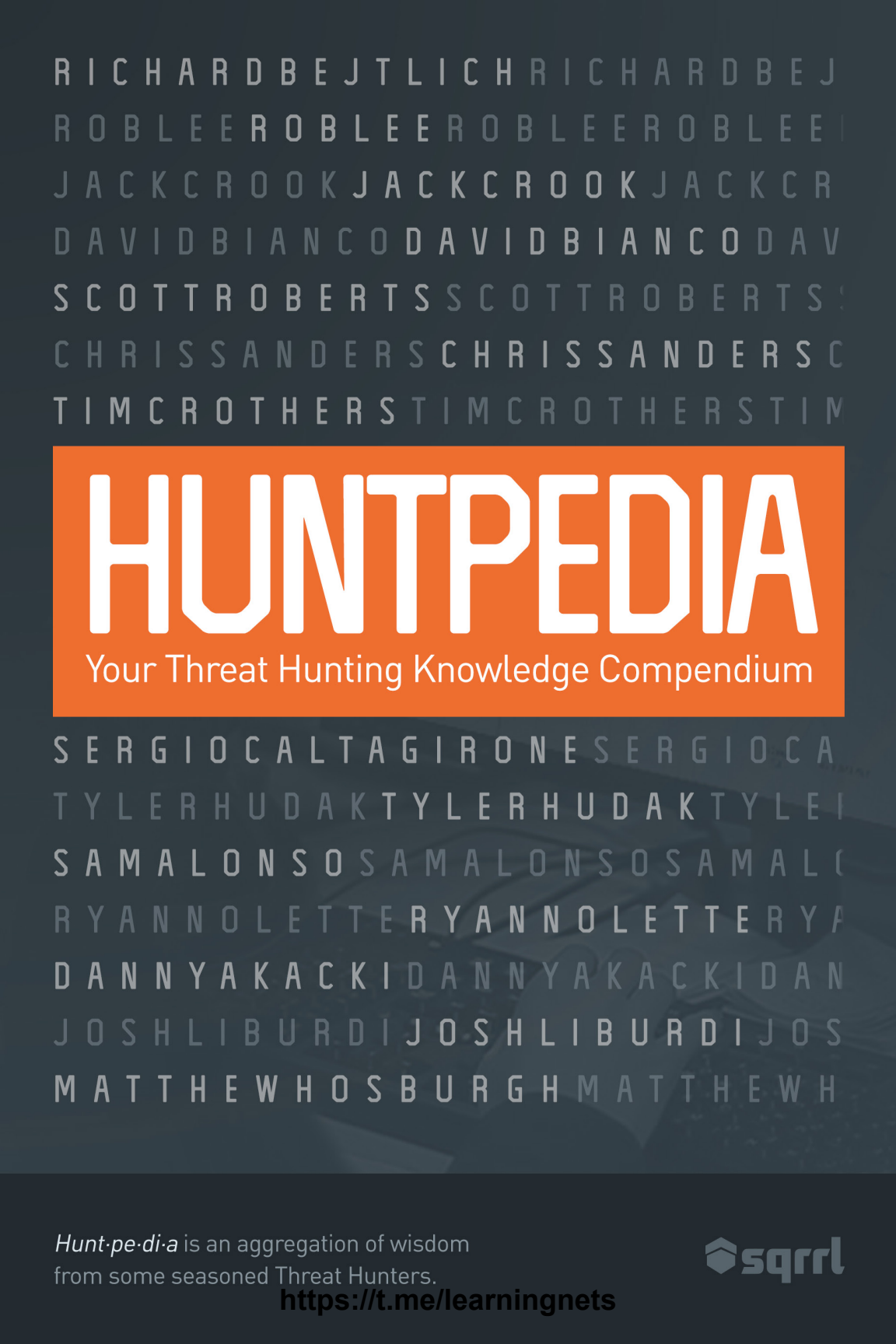


Table of Contents

Foreword

The Origin of Hunting and Why It Matters

Richard Beijtlich

Threat hunting: Everyone's doing it, or at least wants to do it. What is the origin of this term? In 2011 I wrote an article for Information Security Magazine titled "Become a Hunter." I said in part:

"In the mid-2000s, the Air Force popularized the term 'hunter-killer' for missions whereby teams of security experts performed 'friendly force projection' on their networks. They combed through data from systems and in some cases occupied the systems themselves in order to find advanced threats. The concept of 'hunting' (without the slightly more aggressive term 'killing') is now gaining ground in the civilian world."[1]

At the time I was nearing the end of my four-year term as Director of Incident Response at General Electric, and I described the threat hunting work done by my team's incident handlers:

"Senior analysts [take] junior analysts on 'hunting trips.' A senior investigator who has discovered a novel or clever way to possibly detect intruders guides one or more junior analysts through data and systems looking for signs of the enemy. Upon validating the technique (and responding to any enemy actions), the hunting team should work to incorporate the new detection method into the repeatable processes used by SOC-type analysts. This idea of developing novel methods, testing them into the wild, and operationalizing them is the key to fighting modern adversaries."[2]

[1] *https://taosecurity.blogspot.com/2011/12/become-hunter.html*

[2] Ibid.

Why are hunting missions valuable? Why can't intruders simply be stopped altogether, or at least have their actions be automatically detected and identified? Consider the following three factors.

First, high-end intruders have access to most or all of the defensive tools and tactics used by enterprise defenders. Intruders test their own offensive tools and tactics against defender infrastructure prior to deployment "in the wild." Once the attacker knows that his methods will evade or at least frustrate prevention and/or rapid response, he can begin to execute his campaign.

Second, most, if not all, defenders lack the platforms, authority, and resources to fully control, and protect the systems and data for which they are responsible. Defenders struggle to navigate business challenges in which security is, at best, one of many factors to balance in a "risk management" framework, or at worst, an afterthought.

Third, nothing changes as quickly as the technology world. While the grand strategic and operational environments are subject to (relatively) slower disruptive patterns, the tactical and tool-centric landscape is constantly evolving. Microsoft reveals new flaws in its offerings on at least a monthly basis. New applications are released on a weekly basis. New versions of software appear on a daily basis. Developers provision systems in the cloud or in local virtual settings on an hourly basis. The security team is expected to identify, manage, and protect all of this surface area, while keeping a smile on their faces.

Despairing yet? Fear not. My 2007 blog post "Hawke vs the Machine" has a clue to the answer to this problem. In that post I wrote:

"One of the better episodes of an otherwise lackluster third season of Airwolf (one of the best TV shows of the 1980s) was called 'Fortune Teller.' Ace helicopter pilot Hawke finds himself face-to-face with an aircraft equipped with the Fortune Teller, a machine built to out-fly the world's best. My favorite line in the episode follows:

Archangel: They haven't built a machine yet that could replace a good pilot, Hawke.
Hawke: Let's hope so."[3]

Therein lies the secret to the effectiveness of threat hunting. The human heart and mind are the best weapons available to the defender—in any age or environment. Although attackers can test against defensive tools and tactics, they cannot truly test against creative threat hunters. In nearly-impossible-to-protect enterprises, creative threat hunters find ways to unearth treasures using the digital equivalents of toothbrushes and toothpicks. While sitting in the whirlwind of technological change, threat hunters seek the

[3] https://taosecurity.blogspot.com/2007/01/hawke-vs-machine.html

eye of the storm to be effective and efficient. It is only the dedication and flexibility of the human analyst that makes digital defense possible.

Despite advances in robotics, machine learning, and artificial intelligence, I remain confident that no one will build a machine that can replace a good pilot. Thank goodness for that, and enjoy this book!

Richard Bejtlich
Vienna, VA
November, 2017

Part I

Hunting: Theory & Practice

CHAPTER 1

Threat Hunting: People, Process, Technology

"This first chapter is designed to provide a high-level overview of Network Security Monitoring (NSM) and threat hunting. In this chapter, I will discuss modern security monitoring techniques and practices including the overall definition and process of hunting, the people you need to help execute that process, and technology that will help them achieve their goals.

My target audience is the novice threat hunter. That's not to say inexperienced, but those with limited familiarity in matters of threat hunting. This is a jumping off point and, I hope, a productive one. This piece is positioned to be the first in a series of writings that will progressively help lay the foundation, chart the course, and plan the future of a mature threat hunting initiative."

– Danny Akacki

What is "Hunting"?

Simply put, hunting is the act of finding ways for evil to do evil things. Gone are the days of a prevention-centric security posture. While triage is a critically important aspect, it can't be the only goal of a mature SOC. Simply standing up a SIEM and waiting for the alerts to come to you is, at best, counterproductive. It's worth noting that organizations with varied environments and security team goals may define hunting in varied ways such as hunting for vulnerabilities or threat actor attribution. Let's begin our planning with your most valuable assets, your people.

People

Prior to budgeting for and deployment of the latest and greatest security appliance, your first consideration should be the hands at the keyboards. Your greatest assets will always be the people you hire to find evil. Automation is important and necessary, but you still need a set of eyes on the screens to analyze, reason, and make educated assumptions about what your data is telling you. An effective mix of skillsets and experience are required to build a mature Hunt capability.

Individual Style

Each analyst will inevitably develop their own hunt style, processes, tools and tricks. Traditional wisdom has celebrated this, almost to our detriment. The destination mattered more than quantifying the journey, as long as the entire team reached the same conclusion, it was accepted that teaching "style" was a difficult task. The ground truth is quite the opposite; we need to concentrate more on the "how". There should be an utmost importance placed on that journey so we can begin to better identify bias and the components of how we teach others to hunt.

Skillsets

Analytical Mindset. This is, without question, the most important skill an analyst can possess. Without the innate curiosity in and pursuit of the "huh ... that's weird," an analyst can have all the data in the world but they will inevitably find themselves missing pieces of the puzzle. The analyst needs to be able to make reasoned assumptions and chart a new course when the trail runs cold.

Log Analysis. Logs from services and devices are just a couple of the most important and underutilized sources of intelligence for any security department. The ability to analyze logs for anomalies and pivot between data sources to see the big picture is a key competency.

Network Forensics. The ability to read and understand packet capture data and determine the malicious nature of network traffic. If you're fortunate enough to extend your NSM capabilities to the endpoint with an EDR product, a sound foundation in host based forensics is key to compliment your network knowledge.

Network Architecture. An understanding of different network devices and how they operate within the environment.

Attacker Lifecycle. Understanding the different events that happen at any given stage in an attack lifecycle will better prepare your analysts to compartmentalize and prioritize their findings and activities.

Tools. This is an incredibly broad area, but at a foundational level, an understanding of how log aggregators ingest data as well as the function of packet capture analysis tools are essential for the analyst to understand.

OS Architecture. Different operating systems represent different attack vectors. A strong grasp of Windows- and Linux-based operating systems is essential.

Attack Methods. Exploit Kits, Malware, Phishing, and software misconfigurations. Understanding how an attacker attempts to penetrate your network is key to hunting for indicators of the behavior.

Defining The Analyst: Experience and Qualifications

In a perfect world, the size of your security department should be directly proportional to the size of your business, however, this is not always the case. As desirable as a multi-tiered analyst structure might be, sometimes it will work against you. When there are too many cooks in the kitchen, communication can become an issue. Important details can either be overlooked or under-analyzed. Separation of duties will also have to be carved out between different tiers. In order to have a well-rounded team of analysts, several skill levels should be represented within your ranks.

It's important to note that Hunting should be taught and practiced at every skill level. Simply waiting for alerts to pop up on a screen is not an efficient use of your more junior team members. You may also find your organization can't support its own internal threat detection team and you'll need to augment with a Managed Security Services Provider (MSSP). However, you won't know the best course of action until you start doing the legwork and auditing your company's preparedness.

Process

Tools mean little if you don't take the time to build mature processes around them. As a consultant, I often found that more energy and capital are aimed at external vendors than investing in, and management of, internal talent. Processes should be designed around a desire to understand not only what data you have but also the data sources that are missing or misconfigured. You can't secure what you don't know exists.

The aim of a mature security process should be to automate large parts of your threat detection with solid rules and timely altering in order to give your analysts more time to hunt independently.

Gathering Data: Collect, Normalize, Analyze.

Determining what should be logged can be a daunting task. There are many logging standards by which to model your own strategy such as OWASP and NIST. Regulated businesses also have their own logging strategies to consider including HIPAA, GLBA, SOX, etc. One of the key steps in any logging strategy should be to determine what your "crown jewels" are. Data by itself does not equal intelligence, so simply logging all of the things that make noise on your network would be overkill. What are the devices, information or intellectual property that, if compromised, would cripple your organization?
The following are some of the types of logs that may be important to collect in your environment:

- Configuration Management Database (CMDB)
- Application/service logs
- DHCP
- Proxy
- Web and Application Server
- Active Directory/LDAP
- Domain Name Service (DNS)
- Application Firewall
- Database Application and Transaction
- Host-based logs
- Host/Network IDS/IPS
- Firewall
- Antivirus
- Host-based logs
    - Operating System (e.g., Windows Event and UNIX Syslog)
    - Endpoint Detection Response (EDR)
- Virtual Machine Hypervisor
- Network infrastructure logs
    - VPN

- Router
- Firewall
- Load Balancer

From that list, time should be taken to determine how an attacker would go about acquiring those crown jewels. The graphic in Figure 1 illustrates the Applied Collection Framework from the book, Applied Network Security Monitoring by Chris Sanders and Jason Smith. The ACF is designed to help organizations focus on discovering and quantifying those crown jewels.

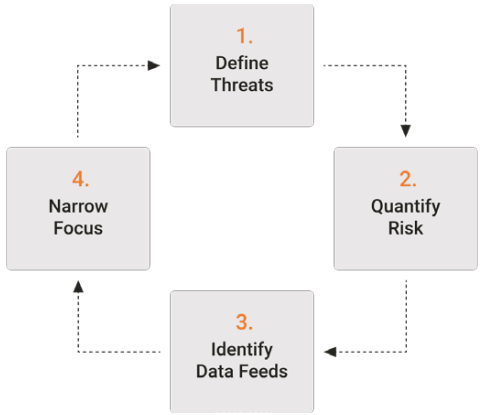


Figure 1 – Applied Collection Framework

Data Retention

Another important part of any logging policy is data retention. How much data should you keep and for how long? A solid retention plan carries many benefits.

Historical: Rarely will you catch an intrusion in real time. The ability to go back through the data to pick up the how and the when of an incident is extremely important.

Operational: Noticing a sharp, unplanned decrease in log flow can signal a problem with your overall processes. Such gaps also hinder your analysts from being able to readily access that data in the event of a security incident.

Strategic: Analytics can be applied to your data in order to find gaps in coverage as well as gaps in software versions.

Legal: If prosecution is part of an organization's goal, forensic data can be used in court to assist in prosecution. In this context, data retention and specifically data integrity

protections are important, otherwise the data will not hold up in court. If you have a General Council, you can leverage them to assist in crafting a policy on chain of custody.

Rules and Alerting

Hunting is as much about automation as it is manual searching. A policy needs to be in place that governs the broader rules that enable hunting. Rules are important to automation, but they must be monitored. Five up-to-date and sharply-tuned rules that give high fidelity alerts are more valuable than 30 broad rules that catch too much and leave your analysts eyeballs-deep in false positives. The policy should outline how often rules and alerts are tweaked and who is responsible for the task.

Rules need tuning to help weed out false positives in order to make it bare actionable fruit. From there, a classification should be made if the rule is worthy of operationalization or utilized as a "one-off" use-case. A feedback loop between the Security Operations and Incident Response functional roles is a key component to a good rules policy. This relationship ensures timely remediation of rules that aren't tuned tightly enough or new intelligence is available to create new or strengthen existing rules.

Metrics

Hunting can be an abstract notion. Directly quantifying your achievements to your management can be a bit of a struggle. "Hunted for this, didn't find it, we're good," is not a measurable achievement. Some people try to approach metrics by indirect measures. A natural question to ask is, has the overall rate of security incidents gone down since you started a hunting program?

The caveat here is that the act of hunting can only increase the number of incidents. It's the post-incident, lessons learned phase that decreases future incidents. A more straightforward question might be: How many incidents did your hunting program uncover that were not previously known? In order for this metric to be most effective, the incidents should be categorized by severity.

The greater lesson is even if you aren't finding evil on the wire, hunting is an opportunity to discover things heretofore unknown or forgotten about your environment as well as create a baseline for normal network traffic. Security gaps directly discovered through a hunting process is an excellent example of a quality metric.

Intel Sources

In order for data to be mined for useful intelligence, context must be applied to make that data actionable. The more context you can apply, the higher the fidelity of your subsequent findings. Intelligence can come from a variety of sources, both internal and external to the organization. Whatever the source, your analysts need a thread to pull to begin their investigations.

- Internal
    - Past incidents
    - Reconnaissance attempts against your infrastructure
    - Threats to your specific line of business and industry verticals
    - Threats to you or your customers' intellectual property
- External
    - Paid intelligence feeds
    - Open Source Intelligence (OSINT)
    - Partnerships with government agencies

Integration

Now that you've cultivated a good amount of evidence, what do you do with it? The first consideration should be the creation of a Knowledgebase or Wiki where you can notate findings for later review. As much as hunting centers around finding the abnormal, documenting the normal is equally important. A ticketing system should be the next consideration for data documentation. This system should be separate from the Help Desk/IT Ticketing system both for ease of use and internal security purposes. This system should have the ability to document false positives in order to tune out as much noise as possible.

Importance of Detection

The truth is dedicated attackers are often times better prepared than the people defending against them. The only thing we know with one hundred percent certainty is that there are no magic bullets when it comes to protecting your company's assets. Prevention is no longer the master plan and hasn't been for quite some time. In addition to robust prevention, defending your organization against current threats requires the development of detection and response capabilities. It's not a matter of if you will be breached, only a question of when. There are various models of attacker methodology

from which we can start to build a foundation for a hunting strategy. Some of these include:

- The Lockheed Martin Cyber Kill Chain©
- The Mandiant Attack Lifecycle
- The MITRE ATT&CK Framework

Overall Strategy

Whichever philosophy you subscribe to, it's agreed that an organization can assist the analysts by at least defining an overall strategy. Like any strategy, it needs to be defined, evaluated, executed and matured. The following is an important, but by no means exhaustive, list of weapons to have in your arsenal to help shape that strategy as you begin to mature your own hunt capability.

- Threat Intel: Paid/Internal/OSINT
- Friendly Intel: Knowledge of critical assets including network diagrams/IP schema
- Approved Service Interruptions (ASI's): When should the admins be making changes?
- Company policy documentation such as an Acceptable Use policy
- Knowledge of data sources

Technology

Combing through your immense data horde is impossible without technology that will not only help you scale but take some of the heavy lifting from your analysts, giving them more time to focus their hunting endeavors. The sheer volume of data we have presents the single biggest challenge to a proactive threat detection strategy. We are essentially trying to find more needles in increasingly larger haystacks. Making things more complicated, the needles are becoming increasingly adept at masquerading as straw.

That being said, needlessly burning capital to crowbar the next new hotness into an environment that can't support or sustain it is counter-productive. Tools need to fit your workflow not the other way around.

Security Information and Event Management platform (SIEM)

Some organizations think of their SIEM as the wall protecting the castle. In reality, it's

more like a guard dog, just one facet of a defense in depth strategy but is by no means a silver bullet. Just like any guard dog, if there is no one to train it or take the time to listen when it barks, it does you no good.

One of the most important questions to ask of your SIEM is, does it enable hunting? If you're unable to quickly pivot between data sources and follow the breadcrumbs without whipping out the manual every 5 seconds, it probably won't be able to stand up to serious investigations. Do you have a dedicated security tool content admin to write rules and tune your SIEM? If not, it will become exponentially harder to get any real value out of your investment.

Other Tools

Not all organizations are created equal. While this paper has been written with the larger enterprise in mind, my hope is to educate and inform organizations of all sizes. If a commercial product is not feasible for your environment, I suggest doing some research into open source tools. Tools like Bro, Snort, Moloch, Wireshark, SOF-ELK and Security Onion can be combined to create a powerful hunting capability.

I would be remiss if I didn't also mention the importance of rolling your own internal tools. You'll be hard pressed to find any security tool, commercial or open source, which will fit snuggly into your existing environment without some customization. Buying a suit off the rack will fit well enough, but for a truly custom fit you go to a tailor. Be your own tailor.

CHAPTER 2

The Pyramid of Pain: Threat Hunting Edition

"As defenders and incident detectors, it's our job to make the attackers' lives as difficult as possible. One way we do this is to consume Cyber Threat Intelligence (CTI), which for purposes of this chapter we'll define as information about an enemy's tools, techniques, capabilities and intentions. Most of us are already familiar with the use of CTI "feeds" in automated detection, a technique often referred to as indicator matching: If we see something in our logs that matches a known piece of CTI (the indicator), issue an alert and have an analyst check it out. But CTI is good for more than just automated detection. Used correctly, it can be a critical part of your threat hunting strategy."

– David Bianco

Using CTI Effectively

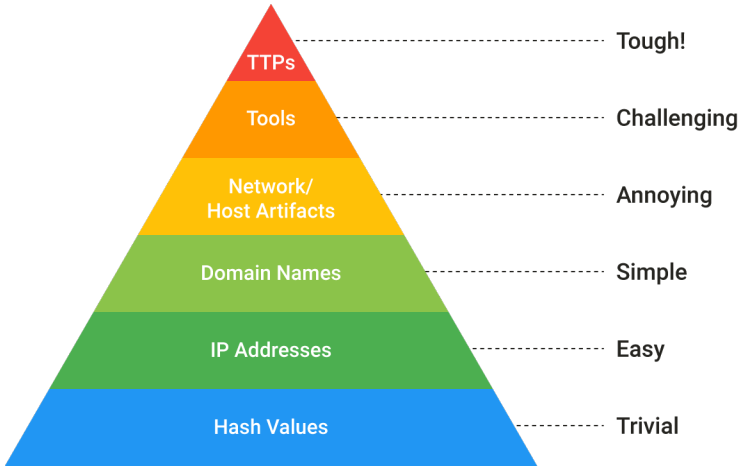
On February 18th, 2013, Mandiant (now a division of FireEye) poked a major hole in the APT intelligence dam when they released their APT1 report profiling a group commonly referred to as Comment Crew. There followed a small flood of reports from other entities like Symantec and the FBI and DHS. Most of the furor over the APT1 report regarded its findings suggesting that APT1 is actually the Chinese PLA's Unit 61398. This was solid work, and a real breakthrough in the public policy area, but I was a lot more excited about the detailed technical information included in the reports' seven appendices (not including the video appendix).

After seeing how these indicators were being applied, though, I came to realize something very interesting: **Almost no one was using them effectively.**

I put that statement in bold, because it's a little bit of a challenge, and I'm sure it will surprise many readers. The entire point of detecting indicators is to respond to them, and if you can respond to them quickly enough, you deny the adversary the use of those

indicators when they are attacking you. Not all indicators are created equal, though, and some of them are far more valuable than others. Furthermore, indicators may be more valuable in some situations than in others.

The Pyramid of Pain



To illustrate the concept of indicator value, I created what I call the Pyramid of Pain. This simple diagram shows the relationship between the types of indicators you might use to detect an adversary's activities and how much pain it will cause them when you are able to deny those indicators to them. Let's examine this diagram in more detail.

Types of Indicators

Let's start by simply defining types of indicators that make up the pyramid:

1. **Hash Values:** SHA1, MD5 or other similar hashes that correspond to specific suspicious or malicious files. Often used to provide unique references to specific samples of malware or to files involved in an intrusion.
2. **IP Addresses:** An IPv4 or IPv6 address. In most cases netblocks or CIDR ranges also fit here.
3. **Domain Names:** This could be either a domain name itself (e.g., "evil.net") or maybe even a sub- or sub-sub-domain (e.g., "this.is.sooooo.evil.net")
4. **Network Artifacts:** Observables caused by adversary activities on your network. Technically speaking, every byte that flows over your network as a result of the adversary's interaction could be an artifact, but in practice this really means those pieces of the activity that might enable us to distinguish malicious activity from

that of legitimate users. Typical examples might be URI patterns, C2 information embedded in network protocols, distinctive HTTP User-Agent or SMTP Mailer values, etc.

5. **Host Artifacts:** Observables caused by adversary activities on one or more of your hosts. Again, we focus on things that would tend to distinguish malicious activities from legitimate ones. They could be registry keys or values known to be created by specific pieces of malware, files, or directories dropped in certain places or using certain names, or descriptions or malicious services or almost anything else that's distinctive.
6. **Tools:** Software used by the adversary to accomplish their mission. Mostly this will be things they bring with them, rather than software or commands that may already be installed on the computer. This would include utilities designed to create malicious documents for spearphishing, backdoors used to establish C2 or password crackers or other host-based utilities they may want to use post-compromise.
7. **Tactics, Techniques and Procedures (TTPs):** How the adversary goes about accomplishing their mission, from reconnaissance all the way through data exfiltration and at every step in between. "Spearphishing" is a common TTP for establishing a presence in the network. "Spearphishing with a trojaned PDF file" or "... with a link to a malicious .SCR file disguised as a ZIP" would be more specific versions. "Dumping cached authentication credentials and reusing them in Pass-the-Hash attacks" would be a TTP. Notice we're not talking about specific tools here, as there are any number of ways of weaponizing a PDF or implementing Pass-the-Hash.

The Pyramid Explained

Now that we have a better idea what each of the indicator types are, let's take a look at the pyramid again. The widest parts of the pyramid are cool colors (blue and green), while the higher levels are warmer yellows, with the pinnacle being red, the hottest of all. Both the width and the color are very important in understanding the value of these types of indicators.

Hash Values

Most hash algorithms compute a message digest of the entire input and output a fixed length hash that is unique to the given input. In other words, if the contents of two files varies even by a single bit, the resultant hash values of the two files are entirely different. SHA1 and MD5 are the two most common examples of this type of hash.

On the one hand, hash indicators are the most accurate type of indicator you could hope for. The odds of two different files having the same hash values are so low, you can almost discount this possibility altogether. On the other hand, any change to a file, even an inconsequential one like flipping a bit in an unused resource or adding a null to the end, results in a completely different and unrelated hash value. It is so easy for hash values to change, and there are so many of them around, that in many cases it may not even be worth tracking them.

You may also encounter so-called fuzzy hashes, which attempt to solve this problem by computing hash values that take into account similarities in the input. In other words, two files with only minor or moderate differences would have fuzzy hash values that are substantially similar, allowing an investigator to note a possible relationship between them. Ssdeep is an example of a tool commonly used to compute fuzzy hashes. Even though these are still hash values, they probably fit better at the "Tools" level of the Pyramid than here, because they are more resistant to change and manipulation. In fact, the most common use for them in DFIR is to identify variants of known tools or malware, in an attempt to try to rectify the shortcomings of more static hashes.

IP Addresses

IP addresses are quite literally the most fundamental network indicator. Short of data copied from a local hard drive and leaving the front door on a USB key, you pretty much have to have a network connection of some sort in order to carry out an attack, and a connection means IP Addresses. It's near the widest part of the pyramid because there are just so many of them. Any reasonably advanced adversary can change IP addresses whenever it suits them, with very little effort. In some cases, if they are using an anonymous proxy service like Tor or something similar, they may change IPs quite frequently and never even notice or care. That's why IP addresses are green in the pyramid. If you deny the adversary the use of one of their IPs, they can usually recover without even breaking stride.

Domain Names

One step higher on the pyramid, we have Domain Names (still green, but lighter). These are slightly more of a pain to change, because in order to work, they must be registered, paid for (even if with stolen funds) and hosted somewhere. That said, there are a large number of DNS providers out there with lax registration standards (many of them free), so in practice it's not too hard to change domains. New domains may take anywhere up to a day or two to be visible throughout the Internet, though, so these are slightly harder to change than just IP addresses.

Network & Host Artifacts

Smack in the middle of the pyramid and starting to get into the yellow zone, we have the Network and Host Artifacts. We're getting into the warm colors, because this is where we begin to "turn the heat up" on the adversary! This is the level, at last, where you start to have some negative impact on the adversary. When you can detect and respond to indicators at this level, you cause the attacker to go back to their lab and reconfigure and/or recompile their tools. A great example would be when you find that the attacker's HTTP recon tool uses a distinctive User-Agent string when searching your web content (off by one space or semicolon, for example. Or maybe they just put their name. Don't laugh. This happens!). If you block any requests which present this User-Agent, you force them to go back and spend some time a) figuring out how you detected their recon tool, and b) fixing it. Sure, the fix may be trivial, but at least they had to expend some effort to identify and overcome the obstacle you threw in front of them.

Tools

The next level is labelled "Tools" and is definitely yellow. At this level, we are taking away the adversary's ability to use one or more specific arrows in their quiver. Most likely this happens because we just got so good at detecting the artifacts of their tool in so many different ways they gave up and had to either find or create a new tool for the same purpose. This is a big win for you, because they have to invest time in research (find an existing tool that has the same capabilities), development (create a new tool **if they are able**) and training (figure out how to use the tool and become proficient with it). You just cost them some real time, especially if you are able to do this across several of their tools.

Some examples of tool indicators might include AV or Yara signatures, if they are able to find variations of the same files even with moderate changes. Network-aware tools with a distinctive communication protocol may also fit in this level, where changing the protocol would require substantial rewrites to the original tool. Also, as discussed above, fuzzy hashes would probably fall into this level.

Tactics, Techniques & Procedures

Finally, at the apex are the TTPs. When you detect and respond at this level, you are operating directly on adversary behaviors, not against their tools. For example, you are detecting Pass-the-Hash attacks themselves (perhaps by inspecting Windows logs) rather than the tools they use to carry out those attacks. From a pure effectiveness standpoint, this level is your ideal. If you are able to respond to adversary TTPs quickly enough, you force them to do the most time-consuming thing possible: retrain and learn

new behaviors.

Let's think about that some more. If you carry this to the logical extreme, what happens when you are able to do this across a wide variety of the adversary's different TTPs? You give them one of two options:

1. Give up, or
2. Reinvent themselves from scratch

If I were the adversary, Option #1 would probably look pretty attractive to me in this situation.

Effective use of CTI for Hunting

Now that we've covered all that background, we can finally turn our attention back to the APT1 indicators. I said that almost no one was making effective use of them. What did I mean by that, and what constitutes "effective use?"

What I meant was that I see a lot of discussion about the long list of domain names included in the report. Seth Hall of the Bro NSM project even put out a neat Bro module you could use to automatically alert when those domains are found in your network traffic.

This is all good and proper, but almost all the reports and the chatter about it in the community was oriented towards automated detection of indicators at the lowest levels of the Pyramid. Of course, this report came out well before most companies were talking about threat hunting, so this is to be expected. But the Mandiant APT1 report continues to serve as a model for much of today's vendor threat reporting, so it's important to understand how these CTI reports can be used for both automated detection and for hunting (which, of course, can often be turned into automated detection).

Whenever you receive new intel on an adversary, review it carefully against the Pyramid of Pain. For every paragraph, ask yourself "Is there anything here I can use to detect the adversary's activity, and where does this fall on the pyramid?" Sure, take all those hashes, domains and IPs and make use of them if you can. They're great for automated indicator matching. Keep in mind, though, that these are often "blown" quickly. Adversaries are likely to move on from them once they become known, so they have an inherently limited lifespan. Host and network artifacts are also well-suited for automated detection, and may be slightly less ephemeral, making them the best choice for signatures and other automated detection.

If you really want to bring the fight to the enemy, though, you have to focus the majority of your detection efforts on the Tools and TTPs levels. This is where you'll probably need to invest some time in hunting. At the tools level, you might do detailed reversing of collected samples to fully understand their behavior in your environment, with an eye towards finding the pieces common in all variants. For the TTPs, you may need to do some data analysis to detect patterns or trends that correspond with that actor's known behavior. Hunting at each level involves a certain amount of time and effort on the part of your analysts. However, once hunting is complete, you can usually turn your new insights into automated detections. This is important because swift, decisive reaction is critical to bringing the pain to your enemies, and nothing is faster than automation!

CHAPTER 3

Diamond Model of Intrusion Analysis

"This chapter summarizes the Diamond Model Technical Report which describes a rich and complex model revealing significant insight into analysis and threat mitigation. Figure 1 is The Diamond Model of Intrusion Analysis. An event is shown illustrating the core features of every malicious activity: adversary, victim, capability, and infrastructure. The features are connected based on their underlying relationship."

– Sergio Caltagirone

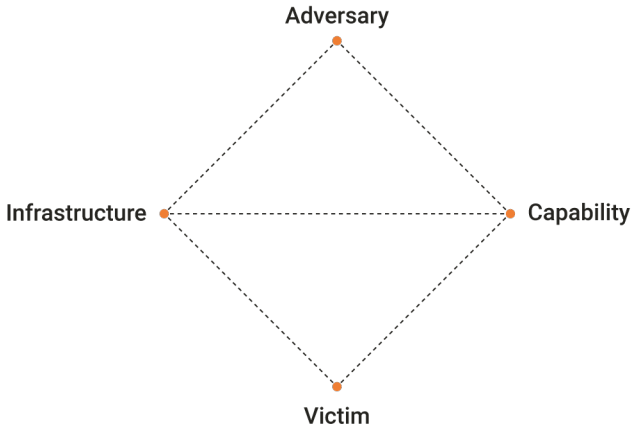


Figure 1 – The Diamond Model

The Diamond Model presents a novel concept of intrusion analysis built by analysts, derived from years of experience, asking the simple question, "What is the underlying method to our work?" The model establishes the basic atomic element of any intrusion activity, the event, composed of four core features: adversary, infrastructure, capability, and victim.

These features are edge-connected, representing their underlying relationships and

arranged in the shape of a diamond, giving the model its name: the Diamond Model. It further defines additional meta-features to support higher-level constructs such as linking events together into activity threads and further coalescing events and threads into activity groups.

These elements, the event, thread, and group all contribute to a foundational and comprehensive model of intrusion activity built around analytic processes. It captures the essential concepts of intrusion analysis and adversary operations while allowing the model flexibility to expand and encompass new ideas and concepts.

The model establishes, for the first time, a formal method for applying scientific principles to intrusion analysis — particularly those of measurement, testability, and repeatability — providing a comprehensive method of activity documentation, synthesis, and correlation.

This scientific approach and simplicity produces improvements in analytic effectiveness, efficiency, and accuracy. Ultimately, the model provides opportunities to integrate intelligence in real-time for network defense, automating correlation across events, classifying events with confidence into adversary campaigns, and forecasting adversary operations while planning and gaming mitigation strategies.

Diamond Model Benefits

- Enables contextual and relationship-rich indicators improving cyber threat intelligence sharing and increasing the range of applicability of indicators
- Integrates information assurance and cyber threat intelligence through activity-attack graphs
- Improves analytic efficiency and effectiveness through easier identification of pivot opportunities and a simple conceptual method to generate new analytic questions
- Enhances analytic accuracy by enabling hypothesis generation, documentation, and testing, thereby applying more rigor to the analytic process
- Supports course of action development, planning/gaming, and mitigation strategies by integrating easily with almost any planning framework
- Strengthens cyber analysis tradecraft development by formalizing first principles upon which new concepts can be explored
- Identifies intelligence gap through a phase-based approach and the inclusion of external resource requirements as a fundamental meta-feature
- Supports real-time event characterization by mapping the analytic process to

well-understood classification and intrusion detection research

- Establishes the basis of cyber activity ontologies, taxonomies, cyber threat intelligence sharing protocols, and knowledge management

Understanding the Diamond Model in Threat Hunting/IR

As a Threat Hunter or incident responder, we try to understand what the threat was in the networks we are defending. We try and answer questions centered around the core components of the diamond model: adversary, victim, capability, and infrastructure. This allows for us to fundamentally detail aspects of malicious activity by using an analytical concept, and to better understand who, what, why, and how. Let's break down the core components into something we can relate to as hunters or defenders.

Adversary

Traditionally this was used to classify certain APT groups (e.g., APT 1) but we can think of this as who the bad guy is behind the activity. Some examples are email, addresses, handles, and phone numbers.

Capability

These are methods or capabilities of how the adversary carries out their objective. Examples would be malware (custom or commodity), exploits, tools used or even commands used, and command and control infrastructure.

Infrastructure

We aim to answer how the adversary completed the attack or action against our network. Some of these examples would include IP addresses, domain names, and email addresses. In most cases of attacks, we want to know how these entities are owned (compromised third party or actual attacker) but this can prove to be very difficult.

Victim

These include network assets, email addresses, the person(s) who were attacked, and

the data that was involved in the attack.

As illustrated in Figure 2, ideally, we would want to link specific incidents together (called activity threads) using the Lockheed Martin Kill Chain to enable grouping of certain activity. The goal here is link relationships between events in the Kill Chain between activity threads.

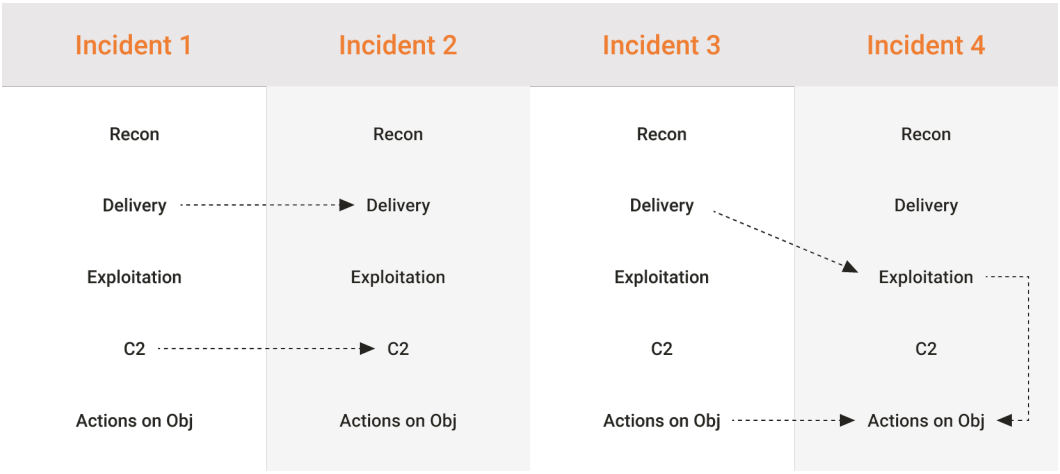


Figure 2 – Incident linking through activity threads

CHAPTER 4

Hunting Through Large Volumes of Logs

"In this chapter, we'll discuss how to hunt through large volumes of logs. We know that companies are consuming huge amounts of data and the people tasked with hunting through these logs often struggle with the volume. The question is: How do we, as hunters and defenders, navigate this sea of data and still be able to find what our current detection technologies aren't finding? This is not an easy task, but I think a lot of it comes down to the questions that we ask."

– Jack Crook

What am I looking for?

I think often we may try and generalize the things we are hunting for. This generalization can lead to broader queries and introduce vast amounts of unneeded logs into your results. Know exactly what it is that you are looking for and, to the best of your knowledge, what it would look like in the logs that you have. Keep in mind that the smaller that "thing" is that you are trying to find, the more focused your query can become.

Why am I looking for it?

Define what is important to your organization and scope your hunts accordingly. Once you define what you will be hunting, break that down into smaller pieces and tackle those one at a time. Prioritize and continue to iterate through your defined process.

How do I find it?

Don't get tunnel vision. If your queries are producing far too many results, know this will

be very difficult to transition to automated alerting. Can you look at the problem you are trying to solve differently and therefore produce different results? This is, by the way, the topic for the rest of this post.

If I look at KC7 for example. Often times the actions that attackers take are the same actions taken legitimately by users millions upon millions of times a day. From authenticating to a domain controller to mounting a network share, these are legitimate actions that can be seen every minute of every day on a normal network. These can also be signs of malicious activity. So how can we tell the difference? This is where looking at the problem differently comes into play. Before we delve into that, we need to look at attackers and how they may operate. We can then use what we may think we know about them to our advantage.

I used the following hypothesis in my slides and I think you would often see some or all of them in the majority of intrusions.

1. Comprised of multiple actions
2. Actions typically happen over short time spans
3. Will often use legitimate windows utilities
4. Will often use tools brought in with them
5. Will often need to elevate permissions
6. Will need to access multiple machines
7. Will need to access files on a file system

Based on the above hypothesis, we are able to derive attacker needs.

1. Execution
2. Credentials
3. Enumeration
4. Authentication
5. Data Movement

I also know during an active intrusion, attackers are very goal focused. Their actions typically can be associated with one of the above needs and you may see multiple needs from a single entity in very short time spans. If you contrast that to how a normal user looks on a network, the behavior is typically very different. We can use these differences to our advantage and instead of looking for actions that would likely produce millions of results, we can look for patterns of behaviors that fall into multiple needs. By looking for indications of needs and chaining them together by time and an additional common entity, such as source host, destination host, or user, we can greatly reduce the amount of data

we need to hunt through.

So how do we get there?

1. Comprised of multiple actions
    b. Actions typically happen over short time spans
    c. Will often use legitimate windows utilities
2. Will often use tools brought in with them
3. Will often need to elevate permissions
4. Will need to access multiple machines
5. Will need to access files on a file system

Attackers will continue to operate in our environments, and the amount of data that we collect will continue to grow. Being able to come up with creative ways to utilize this data and still find evil is essential. I would love to know your thoughts on this or if you have other ways of dealing with billions and billions of logs.

CHAPTER 5

Hunting for Malicious DNS Namespace Collisions

"I've noticed a problem I first saw many years ago come up again in multiple places, and apparently it is still very common—DNS namespace collisions. DNS namespace collisions occur when a private domain name is able to be resolved on the public Internet, whether it is intentional or not. ICANN has a lot of information on this if you are looking for a deep dive on the subject; instead, I will focus on the potential security issues."

– Tyler Hudak

The Issue

Let's start with an example. Suppose you own the Internet domain example.org. This is your Internet presence—all your emails are @example.org, your web servers are in this domain, even your Active Directory domain is corp.example.org. All is well in the world.

When configuring hosts in your organization, one of the things you will do is set up your DNS suffix search list. This is the list of domains your systems will add to a host name if they can't initially resolve it. In our scenario, your DNS suffix search list is example.org and corp.example.org. So, if a host attempts to resolve mail server, they might also try mailserver.example.org and mailserver.corp.example.org.

Let's also suppose that you follow good security practices and have split DNS so no one on the Internet can resolve your internal host names. You also do not allow internal hosts to directly resolve Internet host names.

Any issues so far? Nope. The computer gods are smiling upon us.

As your organization expands, you find the need to add a new internal domain so you choose example.com. Uh oh! You don't own that domain on the Internet, but you'll only be using it on the internal network. Not an issue, right? No, it is a problem.

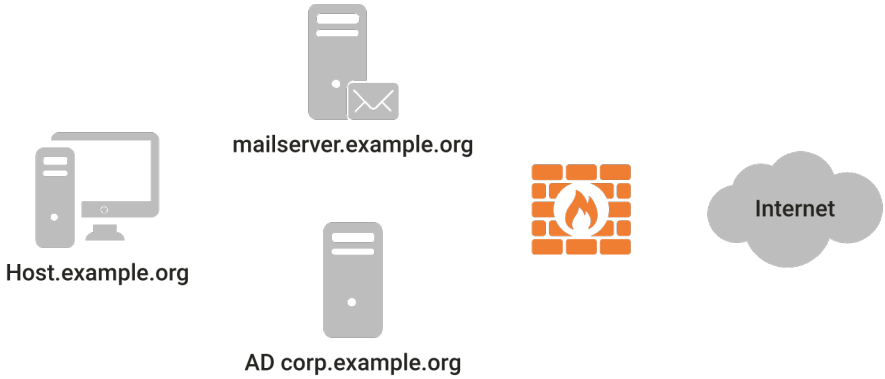


Figure 1 – Your Happy Domain

The issue lies in that you do not own the domain, example.com, but are using it internally; this is a DNS name collision. The issue comes into play as soon as a host accesses the Internet directly (from home, a client's network, etc.). When this happens, they won't be able to resolve hosts with the suffix example.org or corp.example.org—but as soon as they try to resolve with the suffix example.com (which you don't own) they will succeed.

So how is this an issue? In the best case, it isn't. If your hosts try to resolve something that example.com can't resolve then aside from some information leakage, things should be okay. However, what if they try to resolve something that does exist in example.com and then try to start using it?

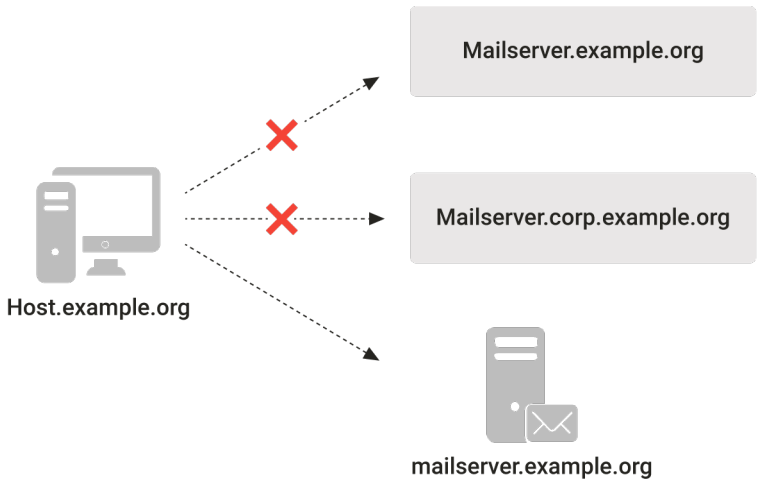


Figure 2 – On the Internet, only example.com will resolve.

For example, our hosts are on the Internet and are trying to reach the internal mail server host name. The only one that is resolvable is mailserver.example.com, which we

don't own. They then start to send emails—your private, internal-only emails—through a server you don't own. See the issue now?

This only happens if that host name already exists in the external domain, right? Wrong.

If DNS wildcards are used, now all of a sudden any host name is being resolved beyond your control and your hosts are sending data to potentially malicious servers. Think of how easy it would be to gain information on your organization or compromise your hosts if I could tell your hosts where their proxy, active directory, or mail servers were when they were outside your organization. And how would you ever know?

This is **not** a theoretical attack. In the last few weeks I have found multiple organizations where this is occurring. Specifically, they are using domains internally that they do not own, their hosts go outside their organization and are resolving these domains to malicious IP addresses.

And there are organizations that are squatting on multiple domains (including obviously internal ones) and setting up wildcard DNS to point them to their own IPs. For what purpose? I don't know, but I suspect it can't be good.

Detection, Prevention, and Response

So how can you detect this? A few ways:

1. Create a list from your DNS for all domains being used by your clients related to your organization. Make sure you own all those domains. If not, what IPs do they resolve to? Consider switching from them. This is also a good threat hunting technique!
2. Windows hosts like to resolve wpad/wpad.dat when browsing. The DNS search suffix tends to get added to that, so look for any HTTP requests to the Internet for wpad.dat, then look for what domains the requests are to. Even if they are not your own hosts (e.g. consultants), you should still be concerned as they could be used as a pivot point into your network.

By the way, wpad.dat is not something you want your hosts doing this with.

Prevention of this is actually pretty easy—just make sure you own any domain you use, or use ones that do not have Internet TLDs. (However, from my research there may be issues with this on some versions of Windows.)

If you do find this happening on your network, I would suggest immediately looking to

see what your hosts are resolving, what data is going out, and more importantly, what is coming back in.

I would also recommend blocking the IP addresses and external domains on your Internet devices to prevent internal hosts from accessing them.

In the end, this is a big problem that I don't think many realize is going on. Fortunately, its fairly easy to detect and start investigating. Doing it now will probably save you a lot of hurt in the long run.

CHAPTER 6

Hunting Anomalous Behavior in Proxy/DNS Queries

"To start hunting threat infrastructure we are going to look at the activity generated by your proxies and DNS. The richness of proxy logs allow you to hunt for different sorts of activities such as drive by downloads. When hunting for evidence of malware in proxy logs, fields such as "Referrer", "URI," "User Agent" and "Host" to name a few are easily stackable and useful pivot points for things like data exfiltration.

I will focus on the URI and Referrer field mainly because these two fields point the user to a resource on the Internet that may be suspicious or will compromise your network. URLs are the value of this field and we will discuss them in this post. These URLs are the link between your organization and the threat infrastructure of the attacker, which is leveraged to compromise your network."

– Samuel Alonso

Components of an attack

There are some interesting components of an attack that I will describe very briefly, such as: drive by downloads, malware delivery networks, fast flux networks, and dynamic DNS. All these components are currently the link between you and your attacker. A simple visit to a URL can redirect you to another URL and serve you an exploit through the landing page. An image is worth one hundred words.

Landing Page – The first item sent by an EK which consists of code that gathers data about a victim's Windows computer and finds a vulnerable application (Flash or Internet Employer) to order to execute code (.exe or .dll).

Exploit Kit (EK) – A server-based framework that uses exploits that target either vulnerabilities in a web browser or third party applications such as Adobe Flash player,

Java Runtime Environment, Microsoft Silverlight, and others.

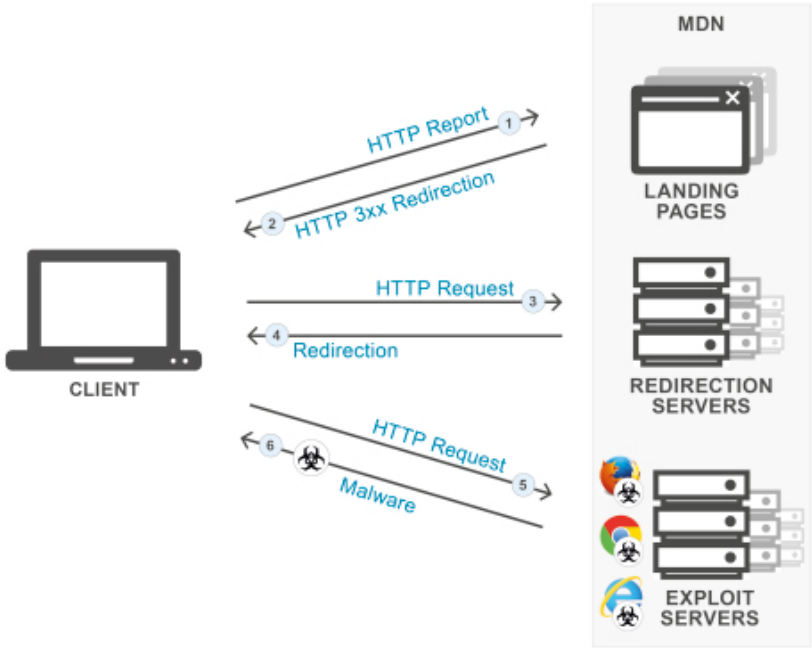


Figure 1 – Basic concepts of a drive by download

Malware – This is the payload delivered by the EK if the exploit is successful. An example of a payload that infected a Windows based computer would be the file types .exe or .dll.

As an example for Exploit Kit detection, a good practice is to search for http code 302 and 200, which could indicate that a URL redirection and a landing page was loaded. Do not forget this could be normal activity, however, it may not be, depending on the value of your URL and the analysis results you obtain from it. This URL is primarily the tip of the iceberg and the first hit against the malware delivery network.

Once the compromise has happened, the malware will phone home to receive commands, extend the foothold, and perform any sort of activity coded in the malware. This last step in which malware is served or is calling home through the malware delivery network, is the most important to understand for a defender. Modern botnets and some malware use what are called fast-flux techniques such as Dynamic DNS and Domain Generation Algorithms (DGA) to evade eradication and add confusion to the defender.

The main idea behind a fast-flux network is to map multiple IP addresses to the same DNS name, so the domain name resolves very quickly, usually in minutes, to different IPs.

These IPs do not host the botnet software but only proxy the query and they send it to a

backend server. This provides the attacker a big grade of resilience, cloaking, and savings since they do not need to duplicate the backend of the malware delivery network.

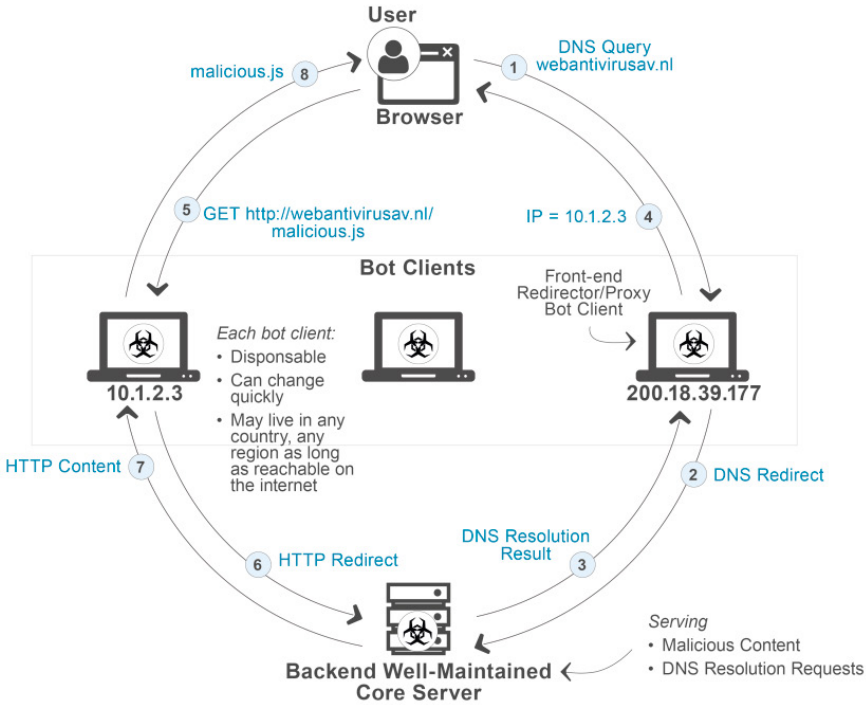


Figure 2 – Attack resilience

Dynamic DNS

Dynamic DNS is a legitimate technology that allows businesses to host resources sitting on constantly changing IP addresses. An example would be an individual or small company which needs to host resources on top of a dynamic IP. When the IP is constantly changing, Dynamic DNS has the benefit of being able to map that resource to that constantly changing IP. This particular feature of DNS makes it very attractive for the criminal who needs to constantly change the IP address of his malware delivery network to avoid detection.

Dynamics DNS can be perfectly detected in the URL field. Not all dynamic DNS domains are malicious, however, they are one isolated indicator that, in conjunction with others, can automatically flag up the malicious nature of the URL. Dynamic DNS is in essence an effective technology to evade IP blacklisting.

DGA

DGA is an interesting algorithm. DGA-based malware generates many domain names and tries to connect to them. Most of these domain names are not registered, they do not exist, and they will generate a high volume of logs with the NXDOMAIN error. Only a few of the domains generated by DGA will be up and running, as the attacker has manipulated the algorithm embedded in the malware to be able to calculate the randomness of the process to generate domains.

This technology helps to avoid domain blacklisting using randomly generated disposable subdomains. Similar to fast flux however, the difference is that for dynamic DNS, the IP falls in the addressing space of one ISP and 1 or 2 ASN´s (autonomous system number). For fast flux, the IP falls in different ASN´s or different IP´s scattered across multiple geographic locations.

Attack Delivery

Let's now look at some potential options for the attacker to deliver the attack: parked domains, legitimate compromised domains, and shady domains.

Parked domains

There are legitimate resources on the Internet, and are usually a single page with ads that provide a very limited value to the user who visits them. These domains are registered by typo squatters (aka domain squatting) or legitimate domain registrars that want to monetize the visit of users who might land on the main page. Oftentimes these domains are used for malicious practices where the page can serve malvertising or malware.

Legitimate compromised domains

This option is easy to understand: A vulnerable site that is not well maintained can be compromised and be used for malicious purposes.

Otherwise suspect domains

In the Internet revolution nearly 40 years ago, we started with six Top Level Domains (TLDs) such as .com, .net, .org, .gov, .mil, and .edu. In later decades, the Internet evolved quickly, and today we have around 1,200 TLDs. The proliferation of TLDs supports Internet

development, however, it also poses a severe risk since it is impossible to monitor them for malicious activity, and that´s how some of these shady domains support exclusively malicious content.

The biggest challenge here for the defender is to block or disrupt the communication to defend against these attacks without causing collateral damage. Things get chaotic when the defender attempts to block or take down shared domain names, IP addresses hosting different sites or block name servers used by different domains.

In a future article, I will tackle this issue and I will explain what tools we have available to detect, block, research, and track this malicious infrastructure. I did not only want to present what is possible to detect, but also how malware currently behaves since some of the activity described above will be seen in the analysis of the URL's you find in your proxy logs. I have seen analysts go crazy since they were not able to explain why tools and automation rules were reporting clean sites or different IP's. Every time the domain is resolved, this created the mentioned confusing situation, effectively evading detection as explained before.

I have taken some time hunting in proxies and they will be helpful to understand what sort of activity is possible to detect in them. As always, it will take time for you to put all the information together but it is the same process I had to go through that will make hunting in your proxy logs second nature. Hunting is an endless learning process in which we need to strive to understand what is possible and also be able to catch up with coming techniques.

CHAPTER 7

Waiting vs. Passivity in DFIR

"To open up this chapter, let me start by sharing one of my favorite songs. In the musical Hamilton, Aaron Burr thinks Alexander Hamilton is a brash, aggressive brute and believes Hamilton thinks him slow and unwilling to make a decision. Burr then sings this song to explain his true goals.

Burr ends up having an iconic line about 3/4 of the way through the song:

BURR: I'm not falling behind or running late
ENSEMBLE: Wait for it, Wait for it, Wait for it, Wait for it
BURR: I'm not standing still, I am lying in wait"

– Scott Roberts

I'd heard this line dozens of times (yeah, I admit, I'm a bit obsessed) but it struck me a bit differently recently. In this case, I was monitoring a resource being used by an espionage group in carrying out their attacks. The question came up: Do we notify the owner and effectively burn the resource, making it unusable by the attacker while losing telemetry? Or do we continue to monitor it and gain intelligence, but risk it still being used against victims?

It was a tough choice that ended with us setting a series of conditions for determining if it was still gaining intelligence. I ended up framing this cost vs. benefit analysis in the terms Aaron Burr uses in Hamilton: In our investigation, what was happening? Were we simply standing still (being passive) or lying in wait (actively avoiding action in furtherance of a more decisive action later)?

Sometimes people claim investigations need to always fix things and move, but sometimes the best thing for an investigation is to take the time to see what's happening and plan the next steps. That shouldn't be mistaken for passivity. But when do you hold and when do you move forward?
Unfortunately, I can't really help you with this one. This takes a combination of experience and intuition. While I can't give you a solid answer, here are some key questions to ask

yourself:

- Are you gaining meaningful intelligence?
- Is gaining more information in the best interest of victims as well as the team collecting?
- Are you able to exploit the information you're gathering in order to act on it in an effective and timely manner?

If you answer yes to those questions, then the best course of action is continuing collection until you've gained as much value as possible. The key is to keep revisiting these questions on a regular basis until there is a change. At that point, you must be prepared to act, whether that's remediating your network, notifying a victim, or sharing the intelligence you've gathered with others.

The last thing to consider about balancing the collection vs. action question is the pressure that can come with it. This can come from many directions—inside your team, your boss, impacted users, law enforcement, etc. The argument is straightforward and compelling: Fix the problem and move on to other things.

In the end, you not only have to decide for yourself but make your case to stakeholders, especially superiors. You have to be prepared to push back against others as long as you're getting meaningful collection and developing intelligence. If you can't get buy in, then the best choice is to hand off information as soon as possible.
Hamilton himself probably said the best thing about the need to remain strong in situations like this ...

> "Those who stand for nothing fall for anything."
>
> **– Alexander Hamilton**

PART II

Hunting – Tools of The Trade

CHAPTER 8

Hunting for Uncategorized Proxy Events Using Sqrrl

"This chapter will address how to hunt through uncategorized proxy events. Attackers rely on the abstraction provided between domains and IP addresses to make their infrastructure more resilient. A domain name can be registered in a matter of minutes, and multiple domains can be configured to point to the same host. This allows attackers to quickly switch between domains and subdomains to avoid detection. One trick experienced hunters use is to rely on the immature nature of these domains and hunt for malicious activity with that in mind. In this post, I'll discuss HTTP proxy categorization and demonstrate how you can use Sqrrl to hunt for malware using previously unseen domains."

– Chris Sanders

HTTP Proxies

Most organizations deploy HTTP proxies as both an efficiency and security mechanism. A proxy is generally a hardware device that all clients must connect through to access the Internet. In doing so, commonly visited pages are cached, delivering a performance boost to the organization's users and saving bandwidth. For the security practitioner, the proxy also creates a single point of protection and detection through its ability to log web browsing and block or alert on entries appearing in blacklists.

These blacklists are a subset of the proxy provider's URL categorization service. By crawling and categorizing sites on the internet, proxies can be configured to deny access

to websites meeting certain criteria. For example, many workplaces choose to block sites classified as pornography, and public schools often block gaming sites. Sites that have been associated with malware are often categorized accordingly and blocked by default with most proxies.

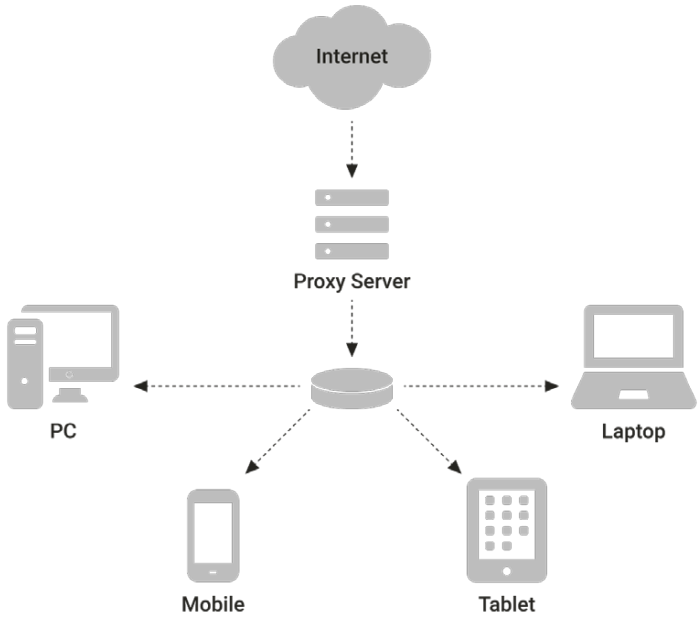


Figure 1 – HTTP proxies serve as a choke point for web browsing

Proxy vendors have gotten good at automating their categorization using things like web crawlers and automated submission analysis from existing customers. That means most of the URLs you encounter will already be categorized. But, what does it mean when you encounter a URL that isn't categorized?

First, the URL could be relatively new. It might also mean that it isn't being indexed by search engines, or that it serves a very specific purpose and has gone undiscovered as a result. Of course, that purpose could be to direct you to a system hosting malware. Many times, it is likely a combination of these things. The important takeaway is that domains registered by malicious actors and used for things like phishing attempts and malware C2 often spend a bit more time as an uncategorized site as far as your proxy is concerned. That means we can hunt for them!

Hunting for Uncategorized Domains

Finding uncategorized domains only requires that you have access to the data feed generated by your proxy, that you're logging URL categorization, and that those logs are

collected somewhere that's searchable. With that in mind, we ask the question, "Did any system on my network make an HTTP request to an uncategorized site?"
This question can be answered with a simple aggregation using Sqrrl Query Language as I've done here:

```
SELECT * FROM Sqrrl_ProxySG WHERE sc_filter_category = 'Unknown'
This query returns all HTTP proxy records (Sqrrl_ProxySG) matching a proxy
category (sc_filter_cateogry) of Unknown.
```

Event Time	c_connect_type	c_ip	cs_bytes	cs_host	cs_method	cs_referer	cs_uri_extension	cs_uri_port
2017-08-16 13:34:3…	Explicit	10.10.1.4	653	mtzlplk.3322.org	GET	http://9991.com/	txt	80
2017-08-16 07:34:3…	Explicit	10.10.1.4	653	mtzlplk.3322.org	GET	http://9991.com/	txt	80
2017-08-15 21:34:3…	Explicit	10.10.1.2	653	mtzlplk.3322.org	GET	http://9991.com/	txt	80
2017-08-15 21:34:3…	Explicit	10.1.23.137	653	9991.com	GET	http://9991.com/	txt	80
2017-08-15 21:34:3…	Explicit	10.10.1.2	653	vqpydhheirk2i.com	GET	http://9991.com/	txt	80
2017-08-15 21:34:3…	Explicit	10.1.23.137	653	bruha.ru	GET	http://9991.com/	txt	80

Figure 2 – A list of all HTTP proxy records for uncategorized sites

If there are only a small number of results, this might be a fine place to start. In large networks or where a lot of unknowns exist, you may be better off performing an aggregation and sorting the visited sites by how many times they were requested, as I've done here:

```
SELECT COUNT(*),cs_uri_stem FROM Sqrrl_ProxySG WHERE sc_filter_category =
'Unknown' GROUP BY cs_uri_stem ORDER BY COUNT(*) LIMIT 100
```

This query selects the URI stem field (cs_uri_stem) from the HTTP proxy data source where the category is unknown and groups and counts all unique entries for that field.

The results are sorted by the count with only the top 100 values shown.

COUNT(*)	cs_uri_stem
1	http://mtzlplk.3322.org/bin2.rar
3	http://vqpydhheirk2i.com/c2_dropkit.js
4	http://mtzlplk.3322.org/c2_dropkit.js
7	http://bruha.ru/c2_dropkit.js

A couple things to keep in mind. First, URL categorization is sometimes an optional

feature or paid add-on provided by proxy vendors. You'll need to ensure that service is enabled. As well, there is no standard naming convention for what proxy vendors call uncategorized URLs. For example, some vendors will refer to these as "Unknown" whereas others refer to them as "Uncategorized." You'll need to edit the above query to account for whatever your proxy identifies these sites as.

Using Risk Triggers with Uncategorized Domains

Uncategorized domains aren't rare enough to be used for alert-based detection, which is why they are well suited as a hunting input. However, these events are also useful during the context of other investigations. You can use Sqrrl's risk trigger functionality to decorate entities with context indicating they've connected to an uncategorized site. The risk trigger below will accomplish this goal.

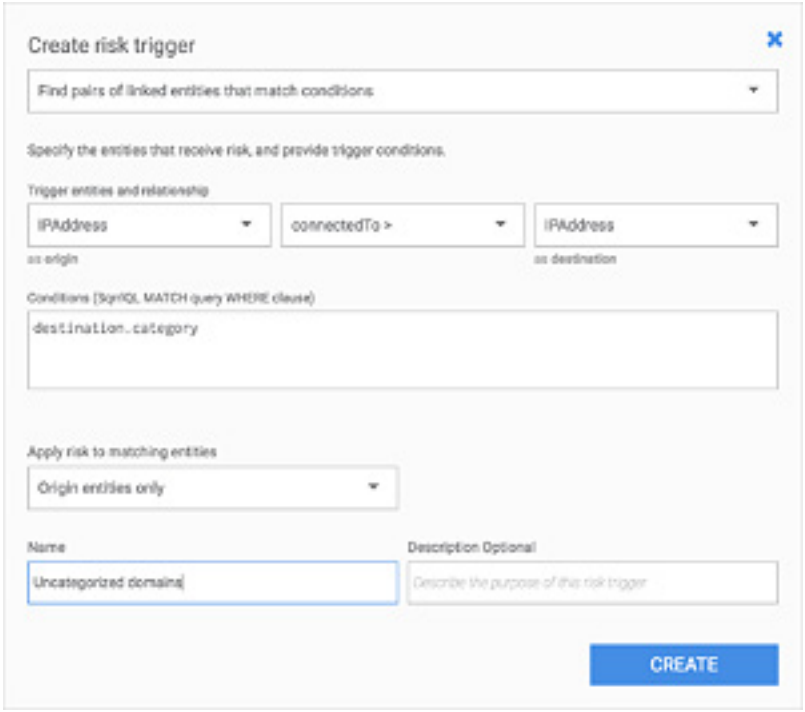


Figure 4 – A risk trigger for uncategorized site visits

Risk triggers are run once a day across the previous 24-hours of data. Should this risk trigger match an entity, it will assign a risk score to it. Those risk scores can be examined during an investigation with higher risk items bubbling up to the main Sqrrl dashboard.

Tips for Investigating Uncategorized Domains

The same strategy you use to investigate any communication with a potentially hostile domain should apply to investigating suspicious uncategorized domains. When you've found one, start by performing a Google search on the domain to see if you can quickly determine that it is legitimate and rule out the finding. If a determination can't be made, you can examine the relationships existing between your internal hosts and the domain.

You should consider examining the following relationships:

- Suspicious files downloaded from the domain. This could indicate the download of malware.
- Unsolicited outbound communication to the hostile domain with no referrer. This could indicate malware command and control.
- The reputation of the domain. This would clue you in if someone else discovered the domain is attacker-controlled.

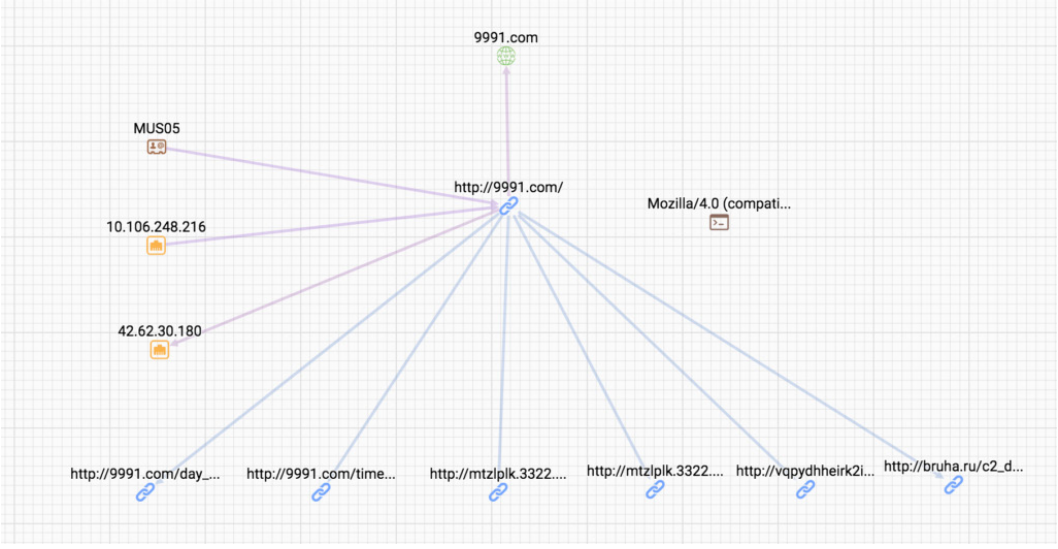


Figure 5 – An uncategorized site reveals interesting relationships

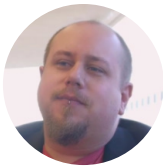
This type of hunting exercise is most likely to find new malware, newer variants of existing malware, or phishing-related domains.

Conclusion

HTTP proxy domain categorization has several uses. Hunters can take advantage of rapid URL classification by searching for communication with yet uncategorized domains to find evidence of malicious activity based on domains that are new or used for a specific purpose. This is a simple technique, but one I've used countless times to find malware that other tools haven't been able to find. By searching for uncategorized domains or Sqrrl or creating an uncategorized domain risk trigger, you'll be able to enrich your investigations similarly.

CHAPTER 9

Hunting Lateral Movement via PSEXEC Using Sqrrl and Bro



"Lateral Movement is a critical step in the process of carrying out an attack on a network. It is a category broad enough that it has its own kill chain step. Although it is a broad tactic, this chapter will survey a specific method that might be carried out by an adversary."

– Ryan Nolette

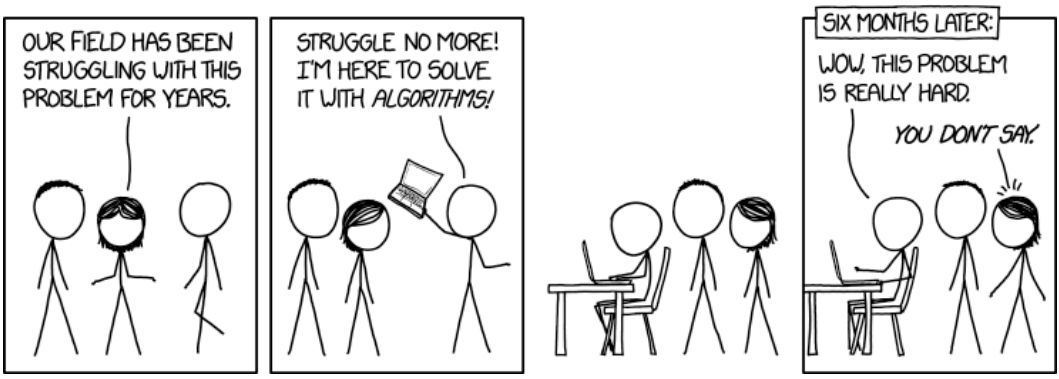


Image source: Randall Munroe, XKCD, 2017

Indicator Searches

The type of dataset you use to hunt for lateral movement depends on what you are hunting for and, by extension, what your hypothesis is. For identifying use of remote access protocols, you will want to focus primarily on network session metadata, including:

- Netflow ("flow" data in general)
- Firewall logs (should log allowed / accepted packets)
- Bro Conn log

For identifying User Access Control (UAC) events, you will want to focus on authentication logs, including:

- Active Directory logs/Windows Security Event logs
    - EventID
        - 528 or 4624 is indicative of a successful logon
        - 529 or 4625 is a failed logon
        - 552 and 4648 are indicative of an attacker attempting to use the runas command or authenticate against a remote host as an alternative user, #privilegeEscalation.
        - 602 and 4698 are indicative of a scheduled task creation
        - 601 and 4697 are indicative of a service creation
    - Account Features
        - Service account
        - Interactive login
- Linux Security Event logs
- OSX Security Event logs
- Multi-Factor Authentication (MFA) logs
- Additional UAC applications if exists

Techniques to Use

After having developed the right hypotheses and chosen the necessary datasets, a hunter must still know what techniques to use to investigate a hypothesis. Here we will survey three types of techniques that you can use to investigate the above.

As in all cases of using indicators in hunting, the value of this approach will be impacted by the value of the indicator. Locally sourced indicators will generally provide a high value because they tend to be timely and relevant to the network or systems you might be trying to protect. These types of indicators can be gathered from previous incidents or by internal threat intelligence teams.

It's also important to remember that it is relatively easy for attackers to change the remote infrastructure that they use to conduct attacks; if you are using indicators to hunt, then you should be aware that the indicators may no longer be relevant to a particular attacker or attack tool.

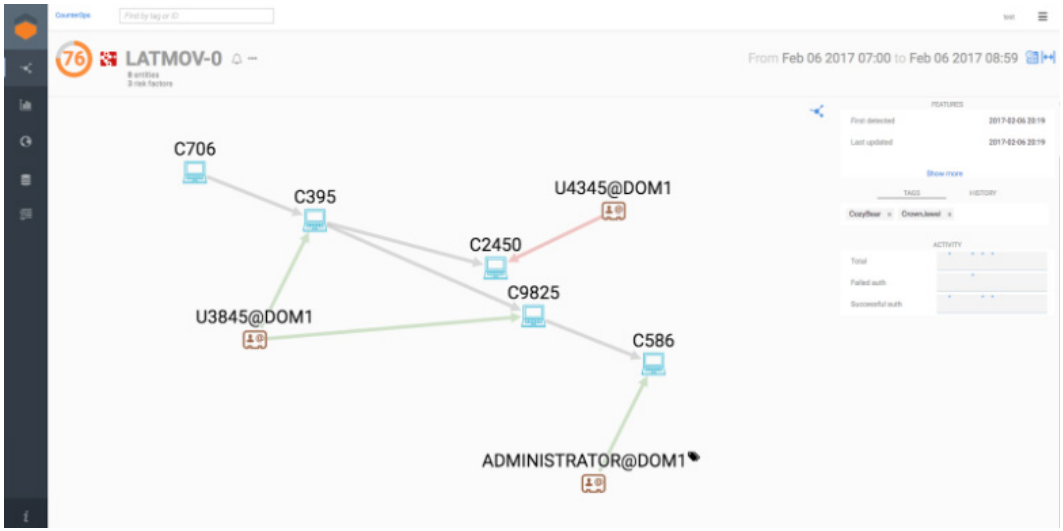


Figure 1 – Indicator search

Some common network session indicators to search for include:

- IP address
- Port
- Top Level Domain (TLD)
- URI
- Unique strings in the connection

Some common host-based indicators to search for include:

- File hashes
- Filenames
- Registry modification
- Process injection

Virustotal

You can submit a file hash or URL to Virustotal to either have it scanned or return existing results if the file/URL has been submitted before and comes with the additional high level information of:

Key	Value	Notes
SHA256	f264700a36a21f96851bddd1a488da66b30c1693544cba5ed2c83562ba62c212	This is the hash value that identifies the file uploaded.
File name	badguy3.exe	This is the name of the file that was uploaded. In addition to this file name there will also be a list of other name the same hash has been known by. (located in the additional information under file names)
Detection ratio	48 / 61	This is the ratio of AV engines that detect the file in comparison to all AV engines that scanned it.
Analysis date	2017-06-19 15:27:17 UTC (1 minute ago)	This is the timestamp of the last analyzed date of the file.

VT also contains a bunch of interesting information about the file, but you can just use the link above to explore that additional information.

Detecting PsExec Activity with Snort

Snort can be used to detect malicious SMB activity. Snort is an open-source intrusion detection and prevention system, and designed to detect attacks via a pattern-matching signature. Below is an example of a Snort rule designed to detect the use of PsExec (Emerging Threats, 2011 http://doc.emergingthreats.net/2010781):

```
alert tcp any any -> $HOME_NET [139,445] (msg:"ET POLICY PsExec? service
created"; flow:to_server, established; content:"|5c 00 50 00 53 00 45 00
58 00 45 00 53 00 56 00 43 00 2e 00 45 00 58 00 45|"; reference:url,
xinn.org/Snort-psexec.html; reference:url, doc.emergingthreats.
net/2010781; classtype:suspicious-filename-detect;sid:201781; rev:2;)
```

Detecting PsExec Activity Using Bro

PsExec is a Windows administration tool used to connect to different systems on a network via SMB, using administrative credentials. SMB is legitimately used to provide file sharing functionality, however, misconfigurations can allow malware to propagate throughout a network. Combine PsExec with the password theft abilities of Mimikatz and you have an equation for lateral movement.

One technique to detect PsExec activity with Bro is by using custom Bro scripts looking for PsExec's use of the C$, ADMIN$, and/or IPC$ shares. These shares added notice messages of "Potentially Malicious Use of an Administrative Share" in the Bro Notice log. The use of PsExec creates an executable named PSEXESVC.exe on the target system. Modified code for my usage. Code is originally from https://www.sans.org/reading-room/whitepapers/detection/detecting-malicious-smb-activity-bro-37472

Figure 2 – Detection of PsExec traffic via a Bro network sensor

In addition to removing control via SMB by PsExec, attackers will upload other binaries to the victim system or use more meterpreter modules. For example, the tool Mimikatz, which is used to dump passwords from memory, can be uploaded to a remote system via the C$, ADMIN$, and IPC$ shares. Bro has the ability to detect Mimikatz getting transferred over SMB and the ability to check its hash against VirusTotal.

If you don't dump your Bro logs into a SIEM or other log aggregation platform, I suggest a simple grep command to search for PsExec usage traffic, "grep -iE "C$|ADMIN$|IPC$"".

Stacking

Stacking is a technique commonly used in many different kinds of hunts. In the case of hunting for command and control activity, a hunter will want to stack for anomalous instances of inbound or outbound traffic. The same metadata types from indicator search above can be used for stacking, including:

- EventID
- UserName
- Account Type
- Hostname

To find lateral movement, you will want to focus on either bidirectional or external to inbound connection flows. The effectiveness of using stacking is dependent on having a finely tuned input. Too little won't reveal enough and too much will flood your ability to tease out meaningful deviations. If a given result set is too large, then consider further filtering of the input data set (e.g., isolate your focus to specific internal subnets). Alternatively, change the metadata that is being stacked (e.g., change from stacking hostnames to stacking Event ID and Username).

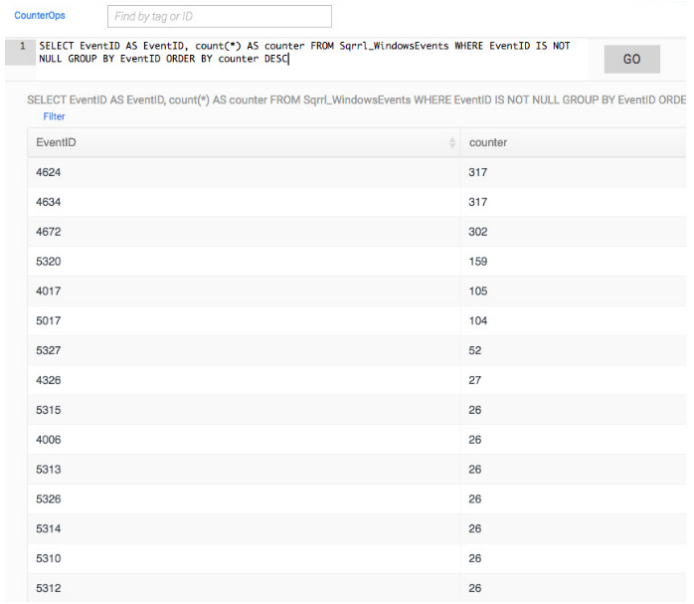


Figure 3 – Stacking EventID

Machine Learning

A more advanced technique involves using machine learning to isolate lateral movement activity. Supervised machine learning uses labeled training data to make predictions about unlabeled data. Given a set of known good and known bad examples, you can create a binary classifier capable of taking in new transactions and deciding if they look more similar to the good training set or the bad training set. After the classifier is trained, assuming you have done a good job, you can feed your network and UAC data through it and get back much smaller set of records that require analyst attention.
For more information on using machine learning for hunting, we highly recommend watching the presentation 'Practical Cyborgism: Getting Started with Machine Learning for Incident Detection' from David Bianco and Chris McCubbin (presented at BSides DC 2016).

Example Hunt

The example below illustrates how to use the hypotheses laid out above with the data and techniques enumerated.

Later Movement (Windows Environment)

1. What are you looking for? (Hypothesis)

Hypothesis:
Attackers may be operating on a C2 channel that uses custom encryption (uncommon protocol) on a common network port.

Look for:
Anomalies in monitored network port channels, i.e. connections that do not have protocol artifacts related to the common port you are looking at. For example, look for connections that have no identifiable SSL metadata over port 443/TCP.

2. Investigation (Data)

Datasets:
For identifying use of common protocols, you will want to focus primarily on application protocol metadata, including:

- Proxy logs, IIS logs
- DNS resolution logs
- Bro HTTP, SSL, DNS, SMTP logs

3. Uncover Patterns and IOCs (Techniques)

1. Use a search to identify legitimate protocol connections on the various common ports you will be inspecting, by looking at protocol metadata
2. If looking at port 80, search for any HTTP protocol records that exist for a given time period
3. Take the output of step 1 and remove the common protocol connections from the session data on the common port. This should leave uncommon protocol connections on the common port
4. Take the results of step 2 and stack the data for what is useful to investigating your hypothesis
5. For example: destination IP, bytes transferred, connection duration/length, etc.

4. Inform and Enrich Analytics (Takeaways)

The destination IP addresses involved in the C2 activity you have discovered can be taken as IOCs and added to an indicator database in order to expand automated detection systems. You can also create packet-level signatures to trigger alerts for cases where the custom protocol you have discovered may appear again.

Always keep in mind that for each instance of a hunt, there will always be multiple different paths that a hunter can take to address a given hypothesis. So what does this activity actually look like in a threat hunting platform like Sqrrl?

First, I need to define relationships between different data sources. For example, my network data and authentication event data both contain the IP address field. This means I can create a relationship between disparate data sets as shown below.

Creating Relationships Between Disparate Data Sets

Figure 1 displays all of the relationships I have created between different entities within

my data. This is a high level visualization of the relationships. The visualization in Figure 2 expands on the high level visualization and illustrates all of the low level field mappings between each data source being ingested and each entity defined. This is where the actual mapping of the IP address field between authentication data and network data.

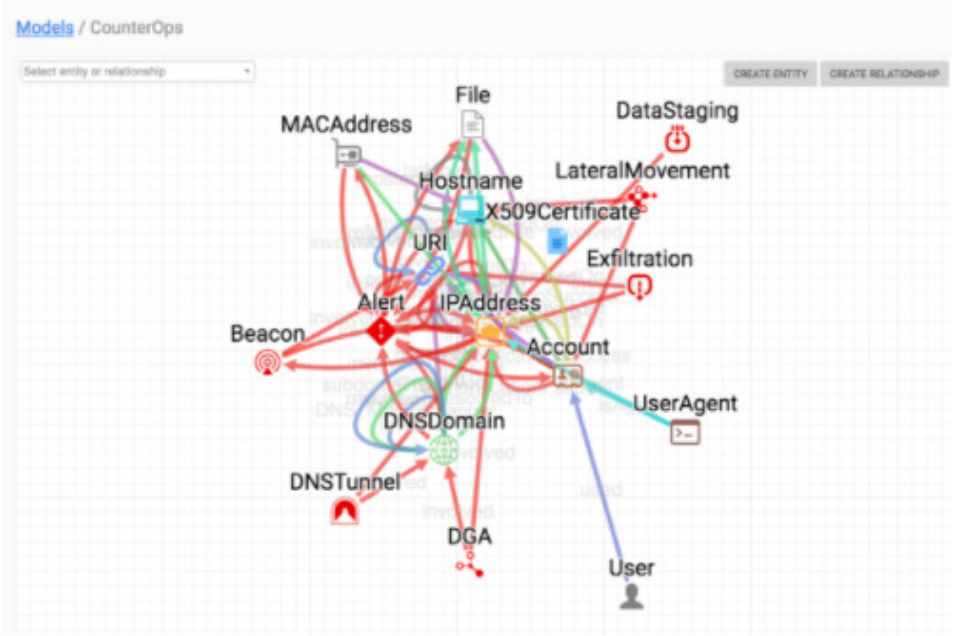


Figure 1 – Relationships

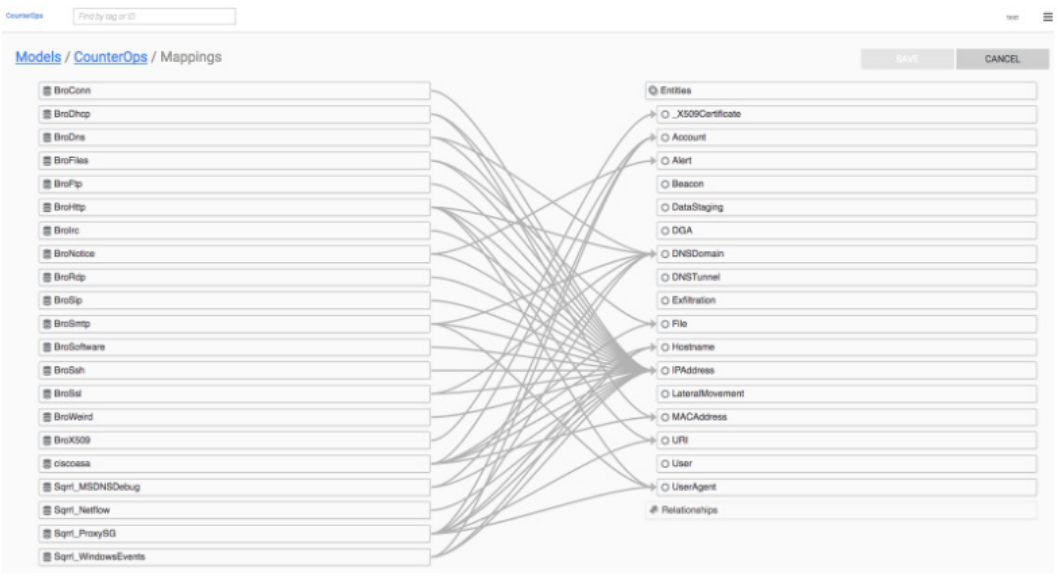


Figure 2 – Low level field mappings

Finally, with all the hard work done, Figure 2 visualizes the event of PSExec being used to move from 192.168.1.100 to 192.168.1.104 as well as incorporate a Bro alert for PSExec to add further validation. While this image only shows the exact activity I am describing for ease of reading, this is what I expect an end result of a hunt to look like. All additional data and possible connections have been investigated and excluded from the original large data set until you are only left with the anomalous/suspicious/malicious event.

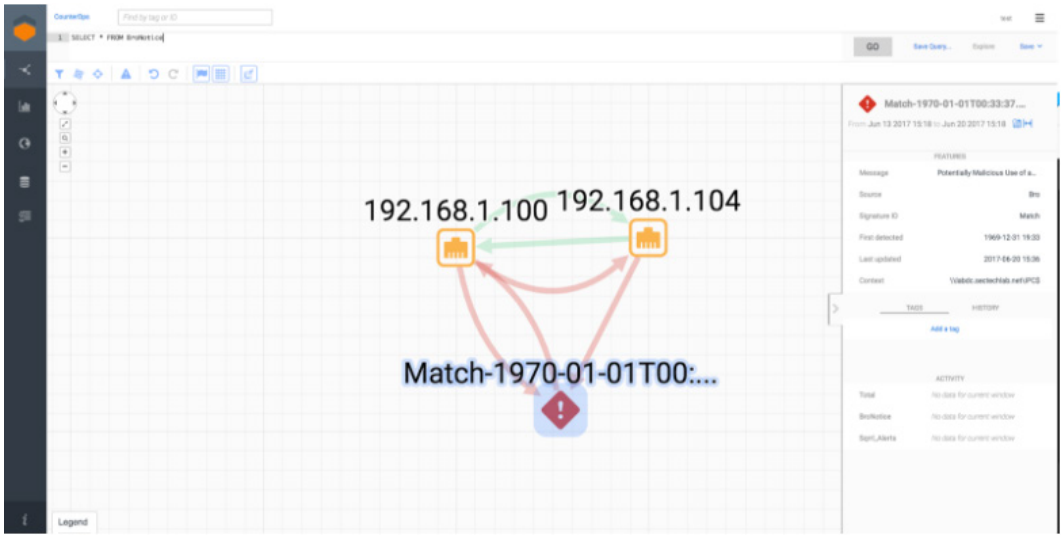


Figure 3 – PSExec movement

I consider this a successful hunt. I would also have considered it a success if I had found nothing at all because the point of a hunt isn't to be a true positive malicious event every time, but instead it is to validate a hypothesis, to answer a question with a definitive yes or no. Good luck to all, happy hunting and as always, remember my motto: Flag it, Tag it, and Bag it.

https://sqrrl.com/the-hunters-den-lateral-movement-part-1/
https://sqrrl.com/threat-hunting-lateral-movement-pt-2-infection/
https://sqrrl.com/finding-evil-when-hunting-for-lateral-movement/

CHAPTER 10

Hunting for Command and Control

"In this chapter we will take a look at how to hunt for Command and Control (C2) activity. Command and control is the process through which an attacker establishes a connection with a compromised asset that they have taken control of in a target network. C2 is a critical step in the process of carrying out an attack on a network. It is a category broad enough that it has its own kill chain step (KC6, "Command and Control"). Although it is a broad tactic, this post will survey the different ways that it might generally be carried out by an adversary."

– Josh Liburdi

Understanding Command and Control

C2 enables remote access for attackers into target networks. Architecturally, C2 is fairly predictable. It will generally follow one of two models for implementation: a Client-Server model or a Peer-to-Peer model. Attackers have multiple options of building their C2 channel, each of which are outlined below.

Command and Control can be identified in either network or endpoint data. Network data, including network session and application protocol metadata, is likely the most common source that organizations have access to for identifying C2 activity. Endpoint data, like process execution and file metadata, can also be used to find C2, but the availability of this type of data is less common. As such, the focus of this post will be on network data.

High-level TTP overview

Attackers generally build C2 channels into common protocols (ComPro) or custom protocols (CusPro). A few examples of common protocols include HTTP/S, SSL/TLS, or DNS. Custom protocols are harder to predict, but include techniques such as encrypting packet data with an XOR cipher.

Like in the case with protocols, attackers generally use common network ports (ComPor) or uncommon network ports (UncPor) for their C2 channels. Examples of standard ports include 80/TCP (HTTP), 443/TCP (SSL/TLS), 53/UDP (DNS). Uncommon ports are difficult to predict, but they typically deviate from ports registered with IANA. Attackers can use any combination of the above protocols and ports:

- ComPro + ComPor
- ComPro + UPor
- CusPro + ComPor
- CusPro + UncPor

In practice, these are the 4 methods through which an attacker may configure a command and control channel. It is instructive to quickly survey each of these methods.

Method 1 – Utilizing a common protocol on a common port

This method of establishing C2 relies very much on the concept of "blending in" with normal network traffic. Often this includes using HTTPS and utilizing high traffic ports in the effort of looking like legitimate traffic. The risk of using this method, from the perspective of an attacker, is that common ports and protocols are often those that are the most regulated by automated detection systems like a SIEM or IDS. As an example, the "Cozy attackers" (Cozy Bear / CozyDuke) have used the standard HTTPS protocol in the past to establish command and control channels. In these cases they were craftily encrypting data inside the HTTP header.

Method 2 – Utilizing a custom protocol on a standard port

Some types of malware have been observed to use custom protocols on standard ports. For example, the FakeM RAT uses a modified implementation of the SSL protocol to transmit data (and it can also mimic common messaging protocols). This can be useful for establishing a connection out of a network because one can expect that in most networks common ports (like 80/TCP, 53/UDP, and 443/TCP) will be open. From the perspective of an attacker, the risk of using this method is that common ports are usually closely watched and the use of a custom protocol might make that transmission stand out from the rest of the traffic. A defender can ask and quickly investigate simple questions to home in on whether this is happening (e.g., "is there any traffic on port 80 that is not HTTP?").

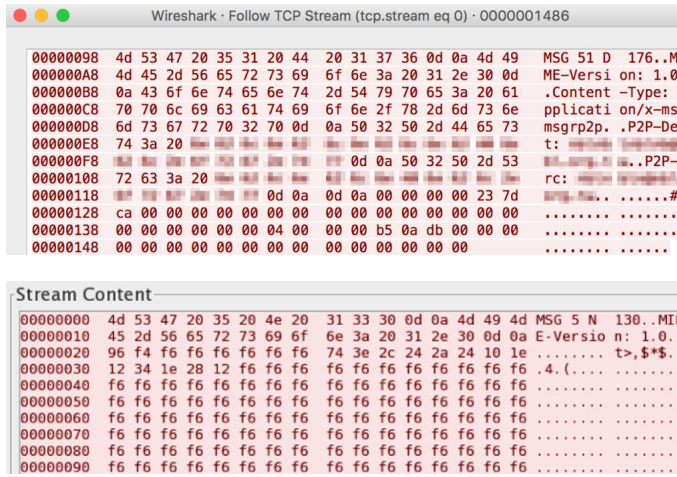


Figure 1 – A comparison of normal MSN Messenger traffic and FakeM RAT's mimicked traffic

Method 3 – Utilizing a common protocol on an uncommon port

Various kinds of malware use common protocols to establish communication links, but do so over uncommon ports that may be open but not as closely monitored as common ones. This might include sending data from common protocols (for example, HTTPS) to ports that will usually be reserved for other protocols (such as DNS). Cisco has documented examples of malware using TLS over uncommon ports.

Method 4 – Utilizing a custom protocol on uncommon port

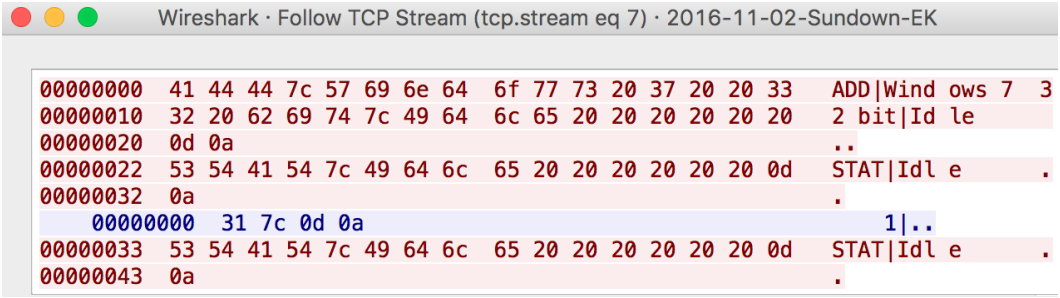


Figure 2 – A packet sample of Kryptik's custom protocol

As a common way to circumnavigate the regular communications channels that might be monitored on a network, attackers can use a custom protocol and send data through an uncommon port. In some target networks, this may be the stealthiest C2 method available to attackers and has been observed as configurations in both commodity

malware and custom attack tools. One example of this method is used by Kryptik malware, which is described on Malware-Traffic-Analysis.

Example Hypotheses/Sub-hypotheses

As covered in previous posts, hypotheses should be the foundation and the basis for whatever hunt you undertake, specifying exactly what it is you're looking for. There are a number of common hypotheses that you can develop to hunt for C2 that directly correlate to the four kinds of TTPs described above. Hypotheses can be derived from working out details of how an attacker might utilize common protocols or ports. Here are generalized hypotheses that align with each type of C2 approach:

ComPro + ComPor

Attackers may be operating on a C2 channel that uses a common protocol on a common network port. Look for unique artifacts pertinent to the protocol you are interested in. For example, if you are interested in identifying C2 in HTTP traffic, then you might consider looking for anomalous domains, URLs, or User-Agent strings.

ComPro + UncPor

Attackers may be operating on a C2 channel that uses a common protocol on an uncommon network port. Look for identifiable artifacts pertinent to the protocol you are interested in on uncommon ports. For example, look for HTTP artifacts over ports that are not 80/TCP or 8080/TCP. Uncommon ports can be identified externally (for example, ports not registered with IANA) or internally (for example, use statistical analysis of network session metadata to determine which ports are infrequently used).

CusPro + ComPor

Attackers may be operating on a C2 channel that uses custom encryption on a common network port. Look for anomalies in monitored network port channels. For example, look for connections that have no identifiable SSL metadata over port 443/TCP.

CusPro + UncPor

Attackers may be operating on a C2 channel that uses custom encryption on an uncommon network port. Look for anomalous characteristics (e.g., connection duration/

length, bytes transferred) in traffic over uncommon ports. Once again, uncommon ports can be identified externally (for example, ports not registered with IANA) or internally (for example, use statistical analysis of network session metadata to determine which ports are infrequently used).

Datasets to Explore

The type of dataset that you use to hunt for C2 depends on what you are hunting for and, by extension, what your hypothesis is. For identifying use of custom protocols, you will want to focus primarily on network session metadata, including:

- Netflow ("flow" data in general)
- Firewall logs (should log allowed / accepted packets)
- Bro Conn log

For identifying use of common protocols, you will want to focus primarily on application protocol metadata, including:

- Proxy logs, IIS logs
- DNS resolution logs
- Bro HTTP, SSL, DNS, SMTP logs

Techniques to Use

After having developed the right hypotheses and chosen the necessary datasets, a hunter must still know what techniques to use to investigate a hypothesis. Here we will survey 3 types of techniques that you can use to investigate the above.

Indicator Search

As in all cases of using indicators in hunting, the value of this approach will be impacted by the value of the indicator. Locally sourced indicators will generally provide a high value because they tend to be timely and relevant to the network or systems you might be trying to protect. These types of indicators can be gathered from previous incidents or by internal threat intelligence teams.

It's also important to remember that it is relatively easy for attackers to change the infrastructure that they use to conduct attacks; if you are using indicators to hunt, then you should be aware that the indicators may no longer be relevant to a particular attacker

or attack tool.

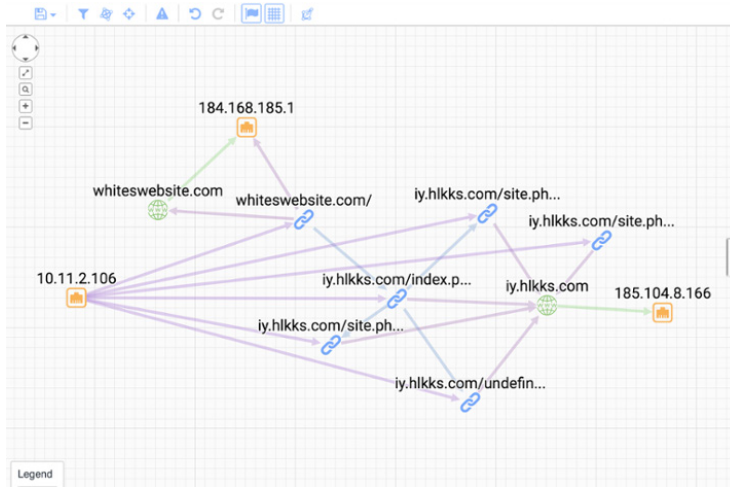


Figure 4 – Searching for indicators

Some common network session indicators to search for include:

- IP address
- Port

Some common application protocol indicators to search for include:

- Domain (HTTP, DNS, SSL)
- URL (HTTP)
- User-Agent string (HTTP)
- X.509 Certificate Subject (SSL)
- X.509 Certificate Issuer (SSL)
- Email address (SMTP)

Stacking

Stacking is a technique commonly used in many different kinds of hunts. In the case of hunting for command and control activity, a hunter will want to stack for anomalous instances of inbound or outbound traffic. The same metadata types from indicator search above can be used for stacking, including:

- Ports
- URLs

- X.509 Certs

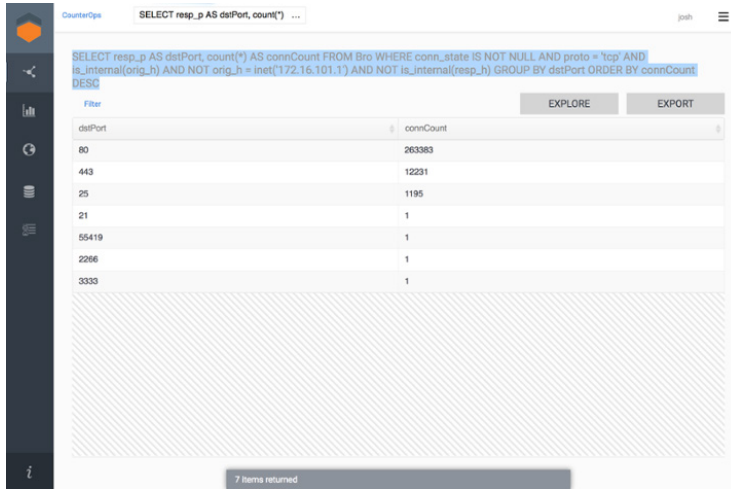


Figure 5 – Stacking for dstport in Sqrrl

To find C2, you will want to focus on either bidirectional or internal to outbound connection flows. The effectiveness of using stacking is dependent on having a finely tuned input. Too little won't reveal enough and too much will flood your ability to tease out meaningful deviations. If a given result set is too large, then consider further filtering of the input data set (e.g. isolate your focus to specific internal subnets). Alternatively, change the metadata that is being stacked (e.g. change from stacking destination IP addresses to stacking destination ports).

Machine Learning

A more advanced technique involves using machine learning to isolate malicious C2 activity. Supervised machine learning uses labeled training data to make predictions about unlabeled data. Given a set of known good and known bad examples, you can create a binary classifier capable of taking in new transactions and deciding if they look more similar to the good training set or the bad training set. After the classifier is trained, assuming you have done a good job, you can feed your HTTP (or other network logs) through it and get back a much smaller set of records that require analyst attention.

Clearcut is a proof-of-concept/demonstration tool created by David Bianco and Chris McCubbin for using machine learning to find C2-like traffic in Bro HTTP logs. In one extreme example, it was used to process approximately 180,000 log entries and produced less than 300 log entries after the machine was trained (a 99.999% reduction). The resulting entries generated by the machine can be manually reviewed to check for C2

activity. You can find Clearcut at https://github.com/DavidJBianco/Clearcut.

For more information on using machine learning for hunting, we highly recommend watching the presentation "Practical Cyborgism: Getting Started with Machine Learning for Incident Detection" from David Bianco and Chris McCubbin (presented at BSides DC 2016).

Example Hunt – Command and Control

The example below illustrates how to use the hypotheses laid out above with the data and techniques enumerated.

1. What are you looking for? (Hypothesis)

Hypothesis:
Attackers may be operating on a C2 channel that uses custom encryption (uncommon protocol) on a common network port.

Look for:
Anomalies in monitored network port channels, i.e. connections that do not have protocol artifacts related to the common port you are looking at. For example, look for connections that have no identifiable SSL metadata over port 443/TCP.

2. Investigation (Data)

Datasets:
For identifying use of common protocols, you will want to focus primarily on application protocol metadata, including:

- Proxy logs, IIS logs
- DNS resolution logs
- Bro HTTP, SSL, DNS, SMTP logs

3. Uncover Patterns and IOCs (Techniques)

1. Use a search to identify legitimate protocol connections on the various common ports you will be inspecting, by looking at protocol metadata
2. If looking at port 80, search for any HTTP protocol records that exist for a given time period
3. Take the output of step 1 and remove the common protocol connections from the session data on the common port. This should leave uncommon protocol connections on the common port
4. Take the results of step 2 and stack the data for what is useful to investigating your hypothesis
5. For example: destination IP, bytes transferred, connection duration/length, etc.

4. Inform and Enrich Analytics (Takeaways)

The destination IP addresses involved in the C2 activity you have discovered can be taken as IOCs and added to an indicator database in order to expand automated detection systems. You can also create packet-level signatures to trigger alerts for cases where the custom protocol you have discovered may appear again.

CHAPTER 11

Hunting Critical Process Impersonation Using Python

"A popular technique for hiding malware running on Windows systems is to give it a name that's confusingly similar to a legitimate Windows process, preferably one that is always present on all systems. The processes that Windows normally starts during boot (which I call the critical system processes) make good targets. Probably the most stereotypical example of this is malware running as scvhost.exe (a misspelling of the actual svchost.exe process). As it turns out, though, if you are collecting information about processes running on your systems, this type of activity is pretty easy to spot.

– David Bianco

When it comes to comparing strings, most of us are familiar with equality since this is a basic operation in most languages.

If str1 == str2, then they match! But string matching isn't always a binary answer. In this case, we specifically don't want to find strings that match exactly, nor do we want to find things that are wildly different. We're really looking for a narrow window of string similarity. That is, we're trying to identify processes that are so similar that they could easily be confused with the real thing, while not actually being identical to the real thing.

As it turns out, there are a number of ways to compute string similarity. They all work a bit differently and have particular strengths and weaknesses, but they typically accept two strings, compare them, and produce some sort of similarity score, like so:

```
score = compare(str1, str2)
```

As long as you're using the same algorithm for all the comparisons, you can use the scores to judge which pairs of strings are more or less similar than the others. Probably

the most well-known algorithm for this is the Levenshtein distance. The resultant score is simply a count of the minimum number of single character insert, delete, or modify operations it takes to convert str1 into str2. For example, the Levenshtein distance between 'svchost.exe' and 'scvhost.exe' (our example above) is 2 (delete the 'v', then add a new 'v' just after the 'c').

Because the algorithm is so simple, the Levenshtein distance would make a pretty decent choice for most uses, though in this case we're going to use a variant known as the Damerau-Levenshtein distance. The only difference here is that the Damerau-Levenshtein distance adds the transpose (switch adjacent characters) operation. Since transposition is one of the common techniques for creating confusingly similar filenames, it makes sense to account for this in our distance algorithm.

With the Damerau-Levenshtein algorithm, the distance between 'svchost.exe' and 'scvhost.exe' is 1 (transpose the 'v' and the 'c'). With that background in mind, let's see how we can apply it across our network to detect malware masquerading as critical system processes.

The Hypothesis

Processes whose names are confusingly similar to those of critical system processes are likely to be malicious.

The Data

In my example, I'm looking at the full path to the binary on disk for any process that actually ran. Binaries that never ran will not be included in my analysis. It's important to note that I'm comparing the path to the binary on disk, not the command line the system says was run, which is arguably the more inclusive check. In reality, you probably want to check both, since it's also quite common for malware to lie about its command line. However, the command lines in my dataset require a lot of parsing and normalization, while the binary paths are pre-normalized. Using only the binaries makes a much clearer demonstration of the technique. Just be aware that for production use, you probably should spend the effort to normalize your command lines and check them, too.

One final note: on medium to large networks, you'll find that it won't be easy to hold all your process execution data in memory at once. My dataset is rather small, so I don't have to worry about it (again, it makes for a cleaner demonstration) but for larger-scale use, you may need to page your search results, convert the code into Spark, or do

something else clever to break things up a bit.

The Hunt

First, import some important modules.

```
import re
from ntpath import basename
from jellyfish import damerau_levenshtein_distance
```

If you are a Python programmer, you probably recognize most of these. The jellyfish module contains a number of routines that compute the "distance" between two strings, which is what this whole hunt is based on. The following is a nested dictionary that defines the system processes we consider to be the most likely targets of this type of process impersonation.

```
CRITICAL_PROCESSES = {'svchost.exe': {"threshold": 2,
                                      "whitelist":['c:\\windows\\system32\\
sihost.exe',
                                                   'c:\\windows\\microsoft.net\\
framework64\\v4.0.30319\\smsvchost.exe']},
                      'smss.exe': {"threshold": 1,
                                   "whitelist":[]},
                      'wininit.exe':{"threshold":2,
                                     "whitelist":[]},
                      'taskhost.exe':{"threshold":2,
                                      "whitelist":['c:\\windows\\syswow64\\
tasklist.exe',
                                                   'c:\\windows\system32\\
taskhostw.exe',
                                                   'c:\\windows\\system32\\
taskhostex.exe']},
                      'csrss.exe':{"threshold":1,
                                   "whitelist":[]},
                      'services.exe':{"threshold":2,
                                      "whitelist":[]},
                      'lsass.exe':{"threshold":1,
                                   "whitelist":[]},
                      'lsm.exe':{"threshold":1,
                                 "whitelist":[]},
                      'winlogon.exe':{"threshold":2,
```

```
                                    "whitelist":[]},
                    'explorer.exe':{"threshold":2,
                                    "whitelist":['c:\\program files
(x86)\\internet explorer\\iexplore.exe',
                                                 'c:\\program files\\internet
explorer\\iexplore.exe']},
                    'iexplore.exe':{"threshold":2,
                                   "whitelist":['c:\\windows\\explorer.
exe']}}
```

For each of these processes, we define a distance threshold (distances greater than zero but less than or equal to this value will be considered suspicious). We also keep a whitelist of the full path names to various legitimate binaries we found that would otherwise still register as suspicious. Feel free to tinker with these thresholds and whitelists, since they are likely to be very system dependent. I had to run through several iterations with a subset of my data before I got values I was happy with. The workhorse of this hunt is the similarity() function, which is defined as:

```
def similarity(proc, critical_procs=CRITICAL_PROCESSES):
    for crit_proc in critical_procs.keys():
        distance = damerau_levenshtein_distance(basename(proc).lower().
decode('utf-8'), crit_proc.decode('utf-8'))
        if (distance > 0 and distance <= critical_procs[crit_proc]
["threshold"]) and not proc in critical_procs[crit_proc]["whitelist"]:
            return (crit_proc, distance)
    return None
```

Pass similarity() the full path to a binary (e.g., c:\windows\system32\blah.exe) and the critical processes dictionary above, and it'll do the checks. If it finds something that's confusingly similar to one of the defined critical processes, it'll return a tuple containing the critical process to which it is similar and the distance score, like so: (crit_proc, distance). It'll stop checking the given process as soon as it finds a close match that's not whitelisted, so at most one result will be returned. If it finds no matches, it'll simply return None.

It's probably important to point out here that this function intentionally only compares the filenames and ignores the path. In other words, it should find c:\windows\system32\scvhost.exe, but isn't concerned with things like c:\windows\temp\svchost.exe, where the filename is correct but the directory path is obviously bogus. We can save this for a future hunt, though it's not hard to imagine combining the two!

At this point you need some process execution data! The exact way you get this will vary from site to site, so I won't try to provide Python code for this. Suffice it to say that my dataset is in a database, so I just queried the entire set via a SQL query:

```
SELECT process_guid, path FROM CarbonBlack WHERE type = 'ingress.event.
procstart'
```

For each unique process execution, Carbon Black provides not only the full file path (the path value), but also a unique ID called the process_guid. If the process turns out to be suspicious, we'll need that so we can investigate further. The code I used to pull this out of the database simply returns a list of tuples, like (process_guid, path), which is what I'll assume yours does, too. If it comes some other way, the following code segment may need a bit of tweaking.

Finally, we just have to go through the results, weeding out the non-Windows processes (I have some Linux servers in my dataset) and compute similarity for each. If the similarity() function returns any results, the code prints them along with the GUID so we can follow up in our favorite threat hunting platform.

```
windows_regex = re.compile("^([a-z]:)|(\\SystemRoot)")

for i in processes_results:
    (guid, path) = i
    if windows_regex.match(path):
        res = similarity(path)
        if res:
            (crit_proc, distance) = res
            print "Process %s is suspiciously similar to %s (distance %d)"
% (path, crit_proc, distance)
            print "\tGUID: %s" % guid
```

When it finds suspicious processes, the code will produce output similar to the following:

```
Process c:\windows\system32\csrrs.exe is suspiciously similar to csrss.exe
(distance 1)
       GUID: 00000009-0000-0477-01d1-f282c48bc278
Process c:\program files\openssh\bin\ls.exe is suspiciously similar to lsm.
exe (distance 1)
 GUID: 00000009-0000-0580-01d1-d883914e50c7
```

```
Process c:\program files\openssh\bin\ls.exe is suspiciously similar to lsm.
exe (distance 1)
 GUID: 00000009-0000-057c-01d1-d87e79c39834
Process c:\program files\openssh\bin\ls.exe is suspiciously similar to lsm.
exe (distance 1)
 GUID: 00000009-0000-1330-01d1-d87e72d3d61e
Process c:\program files\openssh\bin\ls.exe is suspiciously similar to lsm.
exe (distance 1)
 GUID: 00000009-0000-13d4-01d1-d87d971196b0
Process c:\program files\openssh\bin\ls.exe is suspiciously similar to lsm.
exe (distance 1)
 GUID: 00000009-0000-1268-01d1-d30d8b3d6765
Process c:\program files\openssh\bin\ls.exe is suspiciously similar to lsm.
exe (distance 1)
 GUID: 00000009-0000-0254-01d1-d309333a60d1
Process c:\program files\openssh\bin\ls.exe is suspiciously similar to lsm.
exe (distance 1)
 GUID: 00000009-0000-158c-01d1-d3091a4e674b
Process c:\program files\openssh\bin\ls.exe is suspiciously similar to lsm.
exe (distance 1)
 GUID: 00000009-0000-1300-01d1-d3083db995cc
```

As you can see, we did find a few suspicious processes. One of them, csrrs.exe looks pretty suspicious! Fortunately, the rest were all the ls.exe binary provided as part of the OpenSSH package. Since OpenSSH is authorized on our network, this would be a good candidate to add to the lsm.exe whitelist so it doesn't keep coming up whenever we run our hunt.

Conclusions

Even though this hunt is looking at a very specific technique for hiding malware, it can still be effective. If you have endpoint process execution data, looking for processes running with confusing names can be pretty helpful and fairly straightforward to implement.

CHAPTER 12

Hunting for PowerShell Abuse Using Sqrrl

Matthew Hosburgh

Introduction

Earlier in 2017, the world learned about the massive breach that Equifax reported in early September. The Cyber Kill Chain would classify the final stages of this attack as the actions on objective. Unfortunately, Equifax did not detect this activity in time to contain or prevent the data loss. No organization is above the law, especially in terms of a breach; however, failing to learn from this example only paves the way for the next catastrophic breach to occur.

Why PowerShell?

PowerShell is fast becoming the defacto tool for adversaries in nearly every phase of an attack. With Windows holding nearly 75% of the market share, it's no wonder why PowerShell is leveraged so heavily. When time is of the essence, why bring your own tools to the game when you have ready access to them all? As of version 5, the features and capabilities are incredible from both an administrator's and attacker's perspective. One Technique leveraged by known adversaries is the automated exfiltration of data. With PowerShell in play, scripting this type of exfiltration is easy money; however, where is the starting point to begin hunting for something like this?

Formulating the Hunt: Automated Data Exfiltration (Windows Environment)

Begin by leveraging observed techniques to form your hypothesis. MITRE has an

excellent resource, known as the Adversarial Tactics Techniques & Common Knowledge (ATT&CK for short). Contained in this matrix of tactics is a series of techniques that attackers leverage to obtain their objectives. Within each category is an explanation, examples, mitigation, and detection recommendations. Look no further for an outstanding resource to start formulating hunt hypotheses!

1. What are you looking for? (Hypothesis)

Hypothesis:
PowerShell is being leveraged in my environment for automated data exfiltration.

Look for:
Anomalies in HTTP user agent strings, such as PowerShell
Consistent and reoccurring HTTP PUT methods
Cyber Kill Chain Phase:
Actions on Objectives
MITRE ATT&CK:
Automated exfiltration
Example Malware:
TINYTYPHON

2. Investigation (Data)

Datasets:

For identifying misbehaving PowerShell over the network, look at (some easier to examine than others):

Netflow ("flow" data in general)
Frequency and any patterns of PUT methods
Packet Captures
Proxy Logs
Firewall Logs (if logging HTTP headers)

Not required yet, but will be very valuable:

EDR Logs
Sysmon
Windows Event Logs for PowerShell
PowerShell Transcript Logs

3. Uncover Patterns and IOCs (Techniques)

Identify beacons and exfil alerts within the Sqrrl platform
Analyze the frequency of the beacon traffic (figure 2)
Analyze the corresponding exfil alerts. Is it the same?

4. Inform and Enrich Analytics (Takeaways)

The destination IP addresses and user agent string containing PowerShell can be used to help identify other hosts that may be exfiltrating data within environment.

Feed these IOCs into the Risk Trigger in Sqrrl, your IDS, SIEM, or EDR system to automate future detection or response.

Based on one-part hunch, one-part MITRE reference, a viable hypothesis has been created. Similar to the preparation phase of the six-step incident handling process, establishing the resources available to the hunter is a time saver. Now it's time to go hunting!

Network Data: The Broad Brush

Network data is one of the easiest places to start hunting for oddities within an environment; however, encryption or lack of visibility can frustrate your network based hunt. All network traffic has a source. Put another way, there is a process executing on a host at the other end of the traffic you might be seeing. In most cases, the network is a great way to get an idea of where to focus more attention, or what hosts might need to be examined. This is especially true in the case of exfiltration.

PowerShell User Agent Strings

Certain PowerShell actions should stand out among others. One less-than-normal behavior is to see PowerShell making web requests, or even worse, uploading data via HTTP. As an easy starting point, the first technique that can be leveraged is a search for user-agent (UA) strings that line up to the default PowerShell UA. The default can be observed in Figure 1. A good start, but not foolproof. Since version 3.0, there is an option to easily change the default UA to camouflage its identity.

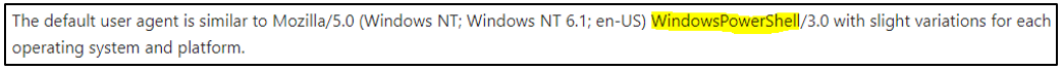


Figure 1 – The default UA string for PowerShell

Frequency Analysis

Since the user-agent is not a reliable indicator alone, Frequency Analysis (FA) can be used to help visualize automated data leakage. In its simplest form, Frequency Analysis illustrates how often a particular set of methods show up and if they seem to be recurring. In this case, the consistent and periodic use of the HTTP PUT method is the primary indicator that will narrow the focus of the hunt.

POST vs PUT

So why not search for POST data, professor? There lies a distinguishing factor between the two. According to the RESTFul API, "POST can be used to add a child resource under

a resources collection." PUT is used "to modify a singular resource which is already a part of resources collection." Confusing, yes, but in my simplistic mind, PUT is more synonymous with uploading an entire file (think cloud file hosting provider).

Got One!

Taking this knowledge and applying it to the hunt can reveal some interesting results. In this case, we can begin our hunt with a suspect PowerShell user-agent, figure 2, at which point we can pivot into connections and detections as can be seen in figure 3. Leveraging FA in combination with the beacon and exfiltration alerts narrows down the misbehaving traffic easily. Figure 2 illustrates the consistent traffic.

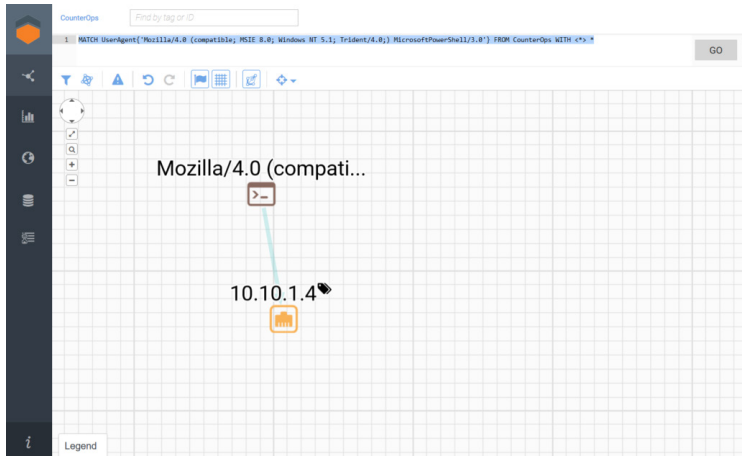


Figure 2 – The PowerShell UA found in HTTP traffic

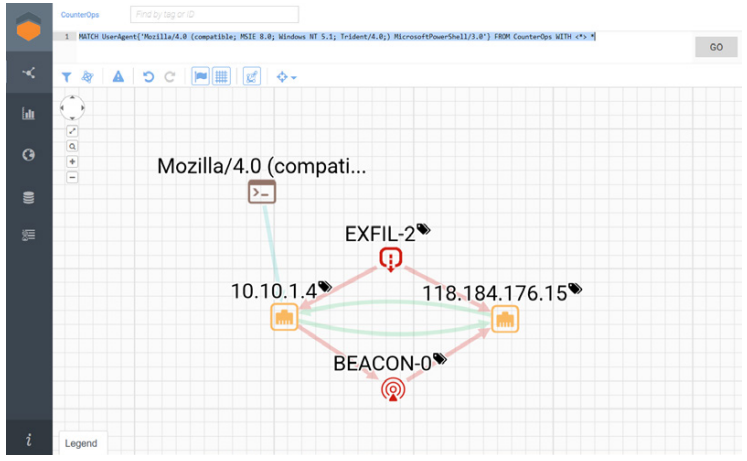


Figure 3 – Connections and detections from initial UA discovery

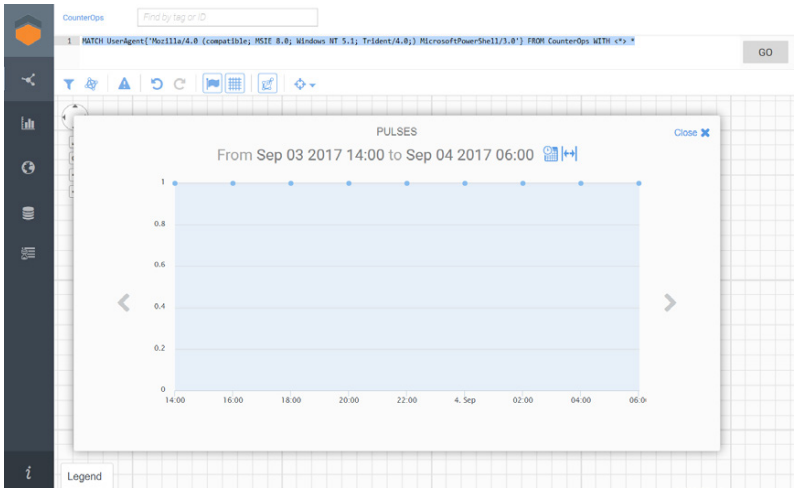


Figure 4 – FA Showing a Consistent Beacon Every Two Hours.

An observation based off of the links and alerts: The beacon might also be the same traffic used to exfiltrate data from the environment. Now it's time to take a look at the host!

Automated Data Exfiltration (Windows Host Environment)

1. What are you looking for? (Hypothesis)

PowerShell is being leveraged in my environment for automated data exfiltration.

Look for:
- Anomalies in HTTP user agent strings, such as PowerShell
- Consistent and reoccurring HTTP PUT methods
- PowerShell processes with related network connections
- Parent processes (and process chain) for the PowerShell processes involved with the network connections

Cyber Kill Chain Phase: Actions on Objectives

MITRE ATT&CK: Automated exfiltration

Example Malware: TINYTYPHON

2. Investigation (Data)

For identifying misbehaving PowerShell on a host, look at:

- EDR Logs
- Sysmon
- Windows Event Logs for PowerShell
- PowerShell Transcript Logs

3. Uncover Patterns and IOCs (Techniques)
• Identify beacons and exfil alerts within the Sqrrl platform • Analyze the processes with matching IP addresses of the beacon and exfil alerts • Examine the process chain of the offending process(es)
4. Inform and Enrich Analytics (Takeaways)
The destination IP addresses and user agent string containing PowerShell can be used to help identify other hosts that may be exfiltrating data within environment. Additionally, with an EDR or process level logs, the MD5s or behavior (scheduled tasks executing PowerShell) could be added to the Risk Trigger in Sqrrl, SIEM, or EDR system to automate future detection or response.

Table 2 – Forming the hypothesis for host hunting

Network Findings

The network indicators pointed to two areas to hone in on: the beacon/exfil IP address. In this case, the address was the same. Additionally, only one host was observed connecting to this IP address, which easily points us (the hunters) to the source, or at least, very close. If you're still feeling uncertain, here's what is known so far:

Description	Indicator
Source IP Address:	192.168.7.61
Destination IP Address:	118.184.176.15
Process (probable, based off of user-agent string):	powershell.exe

Table 3 – Known indicators

So what is needed to continue our hunt? The hunt matrix is a go-to reference that can guide you and your team throughout the duration of a hunt. It serves as the map, or goal, for the duration of the analysis—refer to it often to minimize hunt scope creep (HSC).

The Host in Question

The environment, in this case, was prepared. Every host was setup with Microsoft's Sysmon and detailed host logging before this hunt was initiated. Because of this fact,

the ability to hunt on process-based events is available. Without Sysmon, the ability to hunt on the process would be difficult without an EDR system in place. One other method for an organization without an EDR system would be to enable PowerShell logging and specifically, transcript logging for the greatest visibility.

Transcript logging does wonders for the hunter; however, on a system that might leverage PowerShell frequently, the setting should be tested thoroughly to ensure no adverse effects. The overall process can be seen below.

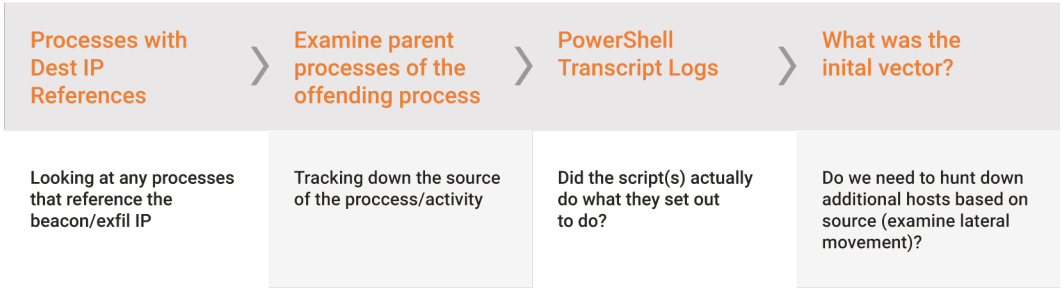


Figure 5 – General process for examining host data with network indicators

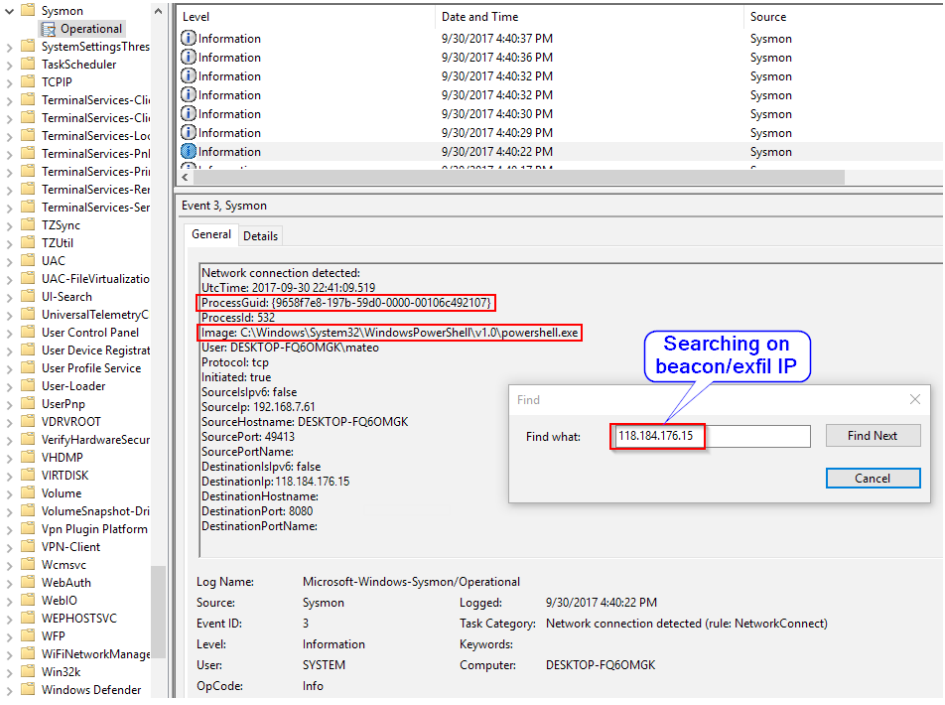


Figure 6 – Searching Sysmon for the exfil IP

Searching for Network Connections within Sysmon logs is a technique to rapidly identify

where the traffic is originating from. Since the exfil/beacon IP is known, taking the host and searching within Sysmon produces results like what is found in Figure 6.

This search yields a positive finding. Examining the log entry, the image (or process name) is identified as powershell.exe. This finding further confirms our suspicion of misbehaving PowerShell.

Process Chaining

The next step in hunting the source is identifying the process chain. This is accomplished by one: an EDR system, like Carbon Black Response, which makes easy work of this technique, or two: manual review. With Sysmon, this is a relatively easy, but tedious endeavor. What we are trying to accomplish is the identification of the process that was created to make the network connection. This can be done by first filtering the log on Event ID 1, which is a process create. Once filtered, searching on the ProcessGUID which was identified in the original search will help point us closer to the source.

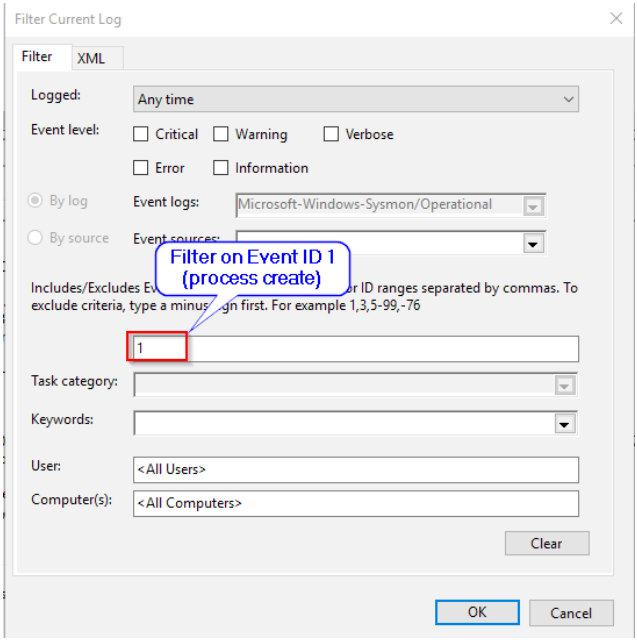


Figure 7 – Filtering Sysmon logs on event ID 1, or process create

With the ProcessGUID, searching for this reference will yield the process that actually made the network connection.

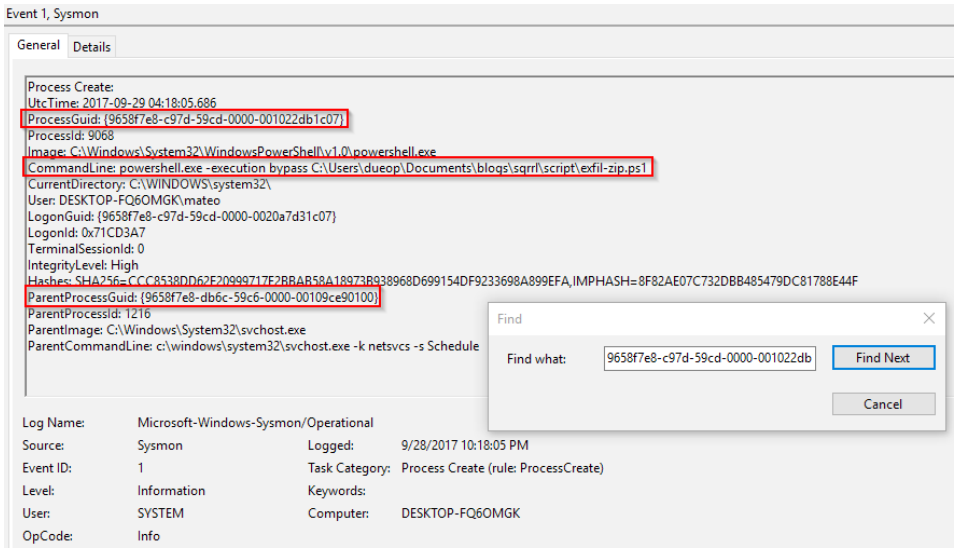


Figure 8 – Search result for ProcessGUID

Another interesting entry is the CommandLine, which points to a PowerShell script. Interestingly, the script is called with the execution bypass flag, easily going around any execution policies set for PowerShell (not to mention the interesting script name). Additionally, the ParentProcessGUID is there, which is the next level up or next process in the chain. The great thing about Sysmon is that this info is also included in the log entry. The ParentCommandLine points to svchost.exe -k netsvcs -s Schedule. This indicates the script is being called on a schedule. As a hunter, the next area to look would be the Task Scheduler and the PowerShell transcript logs.

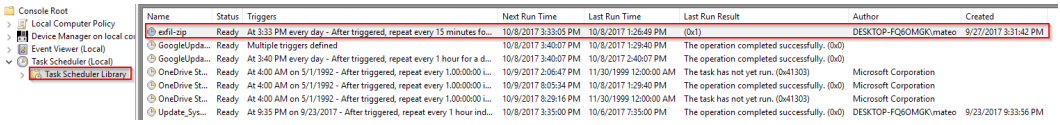


Figure 9 – Task scheduler

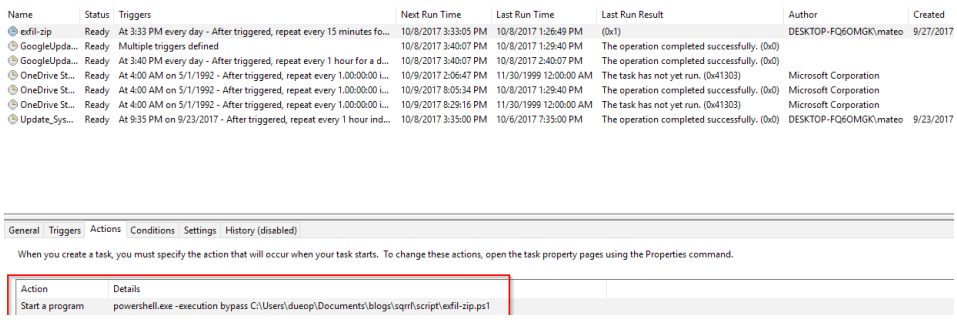


Figure 10 – The task [illegible] scheduled exfil script

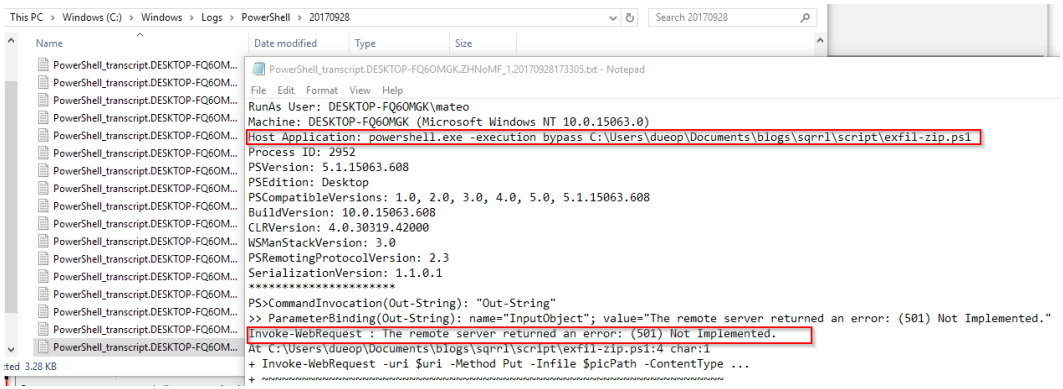


Figure 11 – PowerShell transcript log showing a failed connection

Two things are now known: The script is set to automatically run on a recurring basis and the user that set the job is mateo. Possibly the most burning question of "did we lose data?" can be answered. In this case, the transcript logs show that the web request failed due to a 501 error on the exfil server.

At this point, the next step would be to look at the user mateo to see if there have been other lateral movement attempts/successes and to examine the script. In the case that your organization has the luxury of a SOC or malware analyst team, this could be the juncture where this information is passed along to assist with that analysis, allowing the hunter to continue to pursue the adversary. Looking at these further can help to build additional IOCs or alerts that can be used to build detection capabilities.

Conclusion

In this example, the host was examined for evidence of inappropriate PowerShell usage. The network indicators learned from the first post were leveraged as the starting point to track down suspicious hosts. Leveraging resources such as Sysmon and granular PowerShell logging helped to fully understand what was occurring on the system. The environment had to be ready ahead of time for this type of data to be available, which is an important reminder to any security team. Taking what was found and developing detection capabilities or conducting further analysis can help to increase the odds of the defenders finding adversaries living off the land.

CHAPTER 13

Leveraging Machine Learning for Cyber Threat Hunting

"Machine Learning (ML) and Artificial Intelligence (AI) are two of the hot buzzwords right now. A lot of vendor marketing says they are using ML and AI to solve all of our security problems. Many of us that have been in the security industry for years immediately discount any "easy" solutions offered up by vendors. This is doubly true with Cyber Hunting. That doesn't mean that there isn't a lot of value to be had from leveraging Machine Learning properly. In this chapter, we'll walk through how to achieve some really solid hunting results with ML and a background in Data Science isn't required. We'll also try to give you a better understanding of what ML is and how it works in the process."

– Tim Crothers

One of the reasons to become passionate about leveraging ML for hunting is that it allows us to be more effective across the board and especially for newer analysts to "punch above their weight class" as it were. Allow me to explain.

When I'm hunting in a new organization, one of the first places I tend to look is at HTTP headers flowing out. Why is this? The RFC's for the HTTP protocol don't specify a particular sequence for HTTP headers or whether they should be camel case or something else. However, most legitimate programs use either the Windows, Mac, or Linux libraries to create their HTTP communications. And while the RFC's for HTTP don't specify details like case and order the libraries definitely have conventions. A lot of malicious software creates the HTTP headers internally rather than leveraging the OS libraries. Because the malware authors are rarely familiar with all of the nuances of the standard HTTP libraries you end up with subtle differences that can be used to differentiate manually created HTTP from standard. The downside of this approach is that it takes a lot of time and effort to become familiar with these nuances. With ML, we can let the computer figure out the nuances.

To be clear, ML is not a silver bullet to solve all your cyber hunting needs. With ML the biggest amount of work is up front in curating the data and training your model (both covered in this chapter). But given the need to obtain and curate data to be effective in cyber hunting anyway, a fair bit of the work is already being done. The benefit of ML is once you've got a model trained for your organization, anyone from the least experienced to the most experienced on your team can leverage it to find malicious activity. That is what I mean by "punching above your weight class." What is exciting about ML applied to cyber hunting is that you can have novice cyber hunters achieving results on par with experienced cyber hunters. Given the shortfall of talent in our space, this is very useful.

I want to make sure this piece is not theoretical and allows you, the reader, to work along with me. To that end I'm going to show sample code and work through a real ML project focused on hunting in HTTP headers.

Machine Learning Basics

You don't need to be a data scientist in order to leverage machine learning for some really excellent results. You will, however, need some basic understanding of ML to be effective so we'll start with some fundamentals.

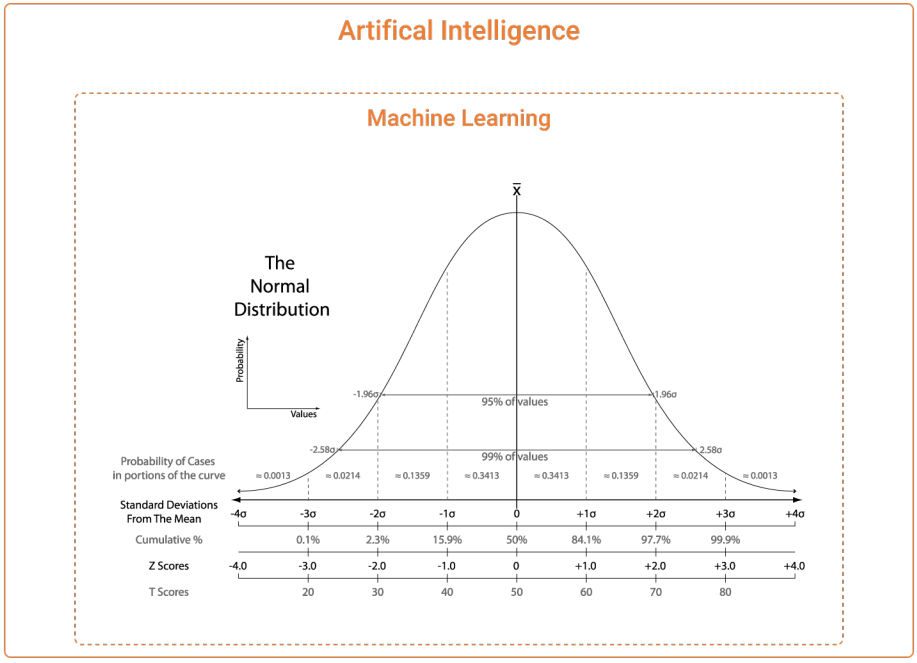


Figure 1 – Statistics image By Heds 1 at English Wikipedia - Transferred from en.wikipedia to Commons by Abdull., Public Domain, https://commons.wikimedia.org/w/index.php?curid=2799839

Machine Learning is a subset of the larger field of Artificial Intelligence. At current levels of maturity, ML is mostly applied statistics. That statement will doubtless offend a lot of data scientists but note the "at current levels of maturity" qualifier. Figure 1 shows the relationship between AI & ML.

Most ML algorithms process a set of data and create a matrix of the characteristics. That matrix of characteristics is called your "model" and it will be used for evaluating additional data against. The model is the output of training your ML algorithm with data and is what is used to evaluate other data to determine what it is. When you use your ML model in hunting, for instance, you run new data against the model and the results are things that an analyst should then look at closer.

Doubtless some of you are thinking "I'm already lost," so let me use an analogy to make it clearer. Let's compare different soda cans. How do you recognize the differences? They are typically the same size and shape. But there are, of course, lots of differences in color, text, and other details. Essentially this is the heart of ML. The algorithm is sampling the data points like color and text in our soda can analogy and comparing them. With a reasonable sample of soda cans, ML can be used to easily determine one from the other.

Classifying soda cans is a simple use case for ML. Where ML gets interesting is in the more challenging comparisons like Dog or Cat. At first glance this would seem equally simple. But if you think about the range of dogs and the range of cats, you can't just tie it to ear shape or face shape or colors because of the many, many varieties of dogs and cats. While in the end the comparison is quite complex, again ML can do a great job here. The key in dogs vs. cats, and indeed most ML classification problems, is a sufficient data set to cover the extremes.

ML becomes a boon for threat hunting because once we have trained a model to differentiate the types of things we are interested in, like malicious HTTP content for instance, the computer is tireless at being able to look through massive amounts of data and be constantly looking (i.e. hunting).

ML is NOT something that is like signatures where a match means you've definitely found something. Rather, think of ML as finding things "like" what you are looking for. A good model will produce accuracy results in the range of 80% to 90%. Anything above 90% accuracy and you have a really good model. This is why ML is generally better suited for hunting cases where you want to take a massive amount of data and let the ML algorithm pare it down to the interesting things a person should look at.

How do you apply ML to a problem like hunting? There are six key steps to leveraging Machine Learning. They are:

1. Choose a problem to solve
2. Decide on approach
3. Have appropriate data
4. Determine appropriate features
5. Build your tool
6. Test & tune

If you complete all six of those steps, you'll end up with a powerful tool in your arsenal. We'll dive into each of the six steps in turn. I am simplifying this, and intend no offense to data scientists. The nuances of each of these steps (and numerous sub-steps) indeed warrant an entire field of expertise. The beauty of the current state of ML, however, is that a lot of people way smarter than I am have taken most of the hard work and boiled it down so that with a basic understanding of ML, one can achieve powerful results. It's very similar to using a car. Someone who knows the details of the mechanics and engineering of a car can modify it to produce much better results than someone who simply knows how to drive it. Because so many talented engineers have built the car and do a lot of complex things under the hood, I can leverage it to accomplish my needs quite effectively. One of my goals in this article is to convince you that ML has gotten to a similar state where you can use it effectively without being a data scientist.

Step One – A Problem to Solve

The first step in leveraging ML is deciding what problem you want to solve. This is an important first step because while Machine Learning is a powerful tool, it isn't a panacea to solve all of your problems. The current state of maturity of ML is that it solves very specific problems much more effectively than broader problems.

Here are some examples of problems I've used ML to help solve:

- Find malicious activity in my DNS logs
- Find malicious PowerShell activity in my Windows logs
- Find malicious traffic that is pretending to be legitimate HTTP/HTTPS

Note that each of those are very specific problems. Examples of the broad types of problems ML is not well suited to are to find all malicious protocols on the wire or expecting a single model to find all types of malicious files. This is not to say that ML cannot be leveraged for broader use cases, but as the breadth of the use cases increases, the complexity and difficulty increases as well. Essentially the tighter you can scope your problem, the more likely you will be able to successfully create a model with a high degree of accuracy.

Another key aspect of selecting your problem is you will need appropriate data. We'll unpack this a bit more in step three, but it's worth noting as a prerequisite for choosing your problem.

Step Two – Decide on Approach

The second step is to decide on whether you are going to use supervised or unsupervised ML. This step tends to be a pretty easy and quick decision but requires a bit of explanation as to how each works and the related pros and cons.

Supervised Machine Learning

Supervised ML is so called because you need to direct (i.e. supervise) it's training. This means you'll need a set of data separated between the types you are trying to build your model to recognize. For instance, if you're trying to find malicious PowerShell in Windows logs you will need logs with both malicious and non-malicious separated. When you train your model, you'll tell the program which is which and let it build the model to differentiate in the future.

For the sake of not getting too deep into the science, supervised learning is generally the best choice for use with specific problems like I advocated in the previous section.

Unsupervised Machine Learning

It is probably not a shock that the other alternative to Supervised ML is Unsupervised ML. Unsupervised ML is essentially anomaly detection. Unsupervised ML is extremely powerful but usually better suited to broader use cases.

No doubt some experienced security practitioners perked up at the mention of "anomaly detection," and indeed it can be very useful. The reason I lean away from it though is that I find the majority of anomalies in the real world are not security issues, but things like misconfigured servers or odd (but not malicious) user or application activity. With anomaly detection you end up with, effectively, another data pile you need to hunt. Certainly, this has value, but I find that by training a model for precisely the types of things I want to hunt, I get things of much more interest to look at. To be clear though, both have value and their place.

Another way to characterize the supervised vs. unsupervised approach is that supervised requires a lot more work up front to curate (i.e. label and clean) your data whereas

unsupervised allows you to get running quicker, but will require a lot more analysis on the back side to evaluate what is found. Outside of the scope of this piece but certainly relevant for those wanting to continue going deeper with ML is a hybrid approach where you leverage unsupervised ML to help you find things of interest and then switch to supervised ML to build a model to find those interesting things long term.

Step Three – Select Appropriate Data

Machine Learning's dirty little secret is that this step is where you'll spend most of your time. The actual ML work is quite simple as you'll see shortly. The hard work is curating your data. On a typical ML project, I spend about 90% to 95% of the overall time curating the data. The age-old analogy of "garbage in, garbage out" certainly applies in spades with ML. The better curated your data, the better your results will be.

For my example code and application for this piece I'm using Supervised Learning. The problem I selected to use for this example is to find Malicious activity in HTTP headers. With my example, the first step was to collect a lot of HTTP headers. I'm a huge fan of Bro and so invariably anything involving network traffic I use Bro.

For a use case like mine to find malicious traffic a great place to start is http://malware-traffic-analysis.net. It's a fantastic treasure trove of malware samples including packet captures. So start with a bunch of malicious traffic pcaps. Then if you're using Bro, the next step is to run bro –r against the pcap files. Bro will process each of them as if it's live traffic and produce the standard mix of .log files based on your Bro configuration and modules you have enabled. You can see this in Figure 2.

```
soinull@wodesangediannao:~/Downloads/malware$ ls
2017-11-21-Hancitor-malspam-traffic.pcap
soinull@wodesangediannao:~/Downloads/malware$ bro -r 2017-11-21-Hancitor-malspam-traffic.pcap >
/dev/null
soinull@wodesangediannao:~/Downloads/malware$ ls
2017-11-21-Hancitor-malspam-traffic.pcap  httpheaders.log
conn.log                                  packet_filter.log
dns.log                                   ssl.log
files.log                                 x509.log
http.log
soinull@wodesangediannao:~/Downloads/malware$
```

Figure 2 – Tagged Log Files

The httpheader.log is what we're looking for in this case. This is not a standard Bro module. Instead of breaking each HTTP component out into a separate column/field like http.log does it creates a single serial concatenation of the entire HTTP header. The reason I chose this over HTTP is mostly due to experimentation and testing for different ways to analyze the HTTP headers using ML. Figure 3 shows part of the contents of the above example.

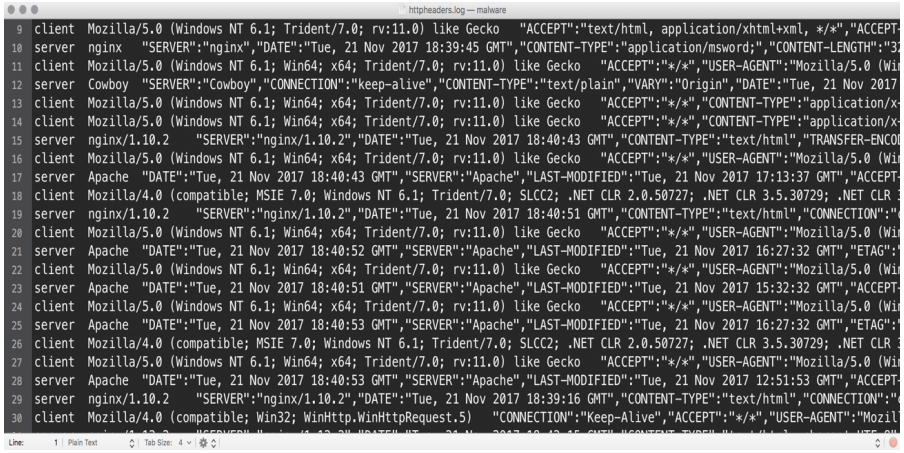


Figure 3 - httpheader.log

The next step is to analyze the sample for anything that isn't part of the malicious activity from the sample. This particular sample is quite clean and just contains the malware activity, so it's suitable for use in training our model. You'll want to get in the habit of checking because a lot of times malware will do things like check for connectivity by hitting things like Google DNS or Microsoft NTP. Those same checks are done by lots of legitimate software as well so you don't want to train your model that those particular activities are malicious.

Getting non-malicious traffic is actually the harder challenge. Ideally you want to pull typical traffic from the environment you want to use the model in. But you're going to need to spend some time pulling out any malicious activity so you don't build a model that thinks malicious traffic is okay. Inevitably, this is where I spend the majority of my time. I find it's easier to edit out the malicious activity in the httpheaders.log files than it is to try to eliminate it up front. To accomplish this, I took an entire day's worth of traffic from my enterprise and ran it through Bro. Then I cross-referenced all of the malicious activity that the team had found for that day and cut those entries out of the httpheaders.log file and pasted them into another httpheaders.log file which I labeled mal_httpheaders.log.

You're going to want a LOT of non-malicious traffic. The wider the variety of traffic you have the better your models will be. Obviously, a LOT is not very specific. There isn't a one right answer for precisely how much a LOT is. I typically try to gather at least a terabyte of non-malicious traffic. That isn't in any way a requirement but I've found in my own experimenting that this is a good threshold that gives good results in my models.

Label and Partition your Data

Once you've collected your malicious and non-malicious traffic for whatever classification

you are going for) you need to make sure you've got it separated for easy labeling of which data is which type. I do this as I collect the data by simply creating 'malicious' and 'non-malicious' folders under my project folder and drop the log files in the appropriate directories as I clean them.

The other thing you need to do is partition off about 20% of each type for later testing. After I have my training data gathered, I create 'malicious-test' and 'non-malicious-test' folders and simply move 20% of the data to each. We'll use those sets of data to determine how effective our models are after we build them. The 80% will be used to create the models.

Step Four – Determine Proper Features

Features are the data points that your ML model will be using to calculate statistical variance from. Getting the right features is definitely critical to the accuracy of your model. This is the step that requires the most data science and statistics understanding to build the best possible models. That said, you can still build very effective models without understanding feature selection inside out and backwards.

Some examples of features for different types of traffic are:

- URL in HTTP headers
- User Agent in HTTP headers
- Event ID in Windows event logs
- DNS Query and Response from DNS logs

You can see from these examples that some features are generally obvious based on subject matter expertise. Sometimes the best features are not obvious, and a proper data scientist will typically not assume any features and build a model using all of the possible data points as a feature and then use a variety of tests like computing the variance between individual features to guide which features should be selected. In the end though, good old experimentation is a large part of it, even for data scientists. I've found experientially that starting with features based on understanding of the protocols or data yields very effective models the majority of the time.

It was experimentation for good features that led me to using the httpheaders module over individual HTTP header values. Remember that unlike building a static IDS signature, we're looking to build a fuzzier result. Our model will never produce 100% accuracy but it will also find things that a static signature won't for those same reasons. This is exactly why I feel that ML is a great hunting tool rather than a standard detection

tool.

The final consideration with feature selection is that more features aren't necessarily better than less. The more features you add the more complex (and thus slower) the computations are. There is a sweet spot where you have sufficient features to be highly accurate but not so many that it takes the model forever to process your data.

Step Five – Build your Tool

Data science libraries exist for virtually every language. My language of choice for this work is Python. There are three modules, Pandas, Sci-Kit Learn, and NumPy that do all of the heavy lifting for us. Pandas takes care of the large data sets. Pandas uses what it calls dataframes to hold large data sets. Sci-Kit Learn and NumPy provide all the data science and mathematical support. All three modules are very actively supported and maintained.

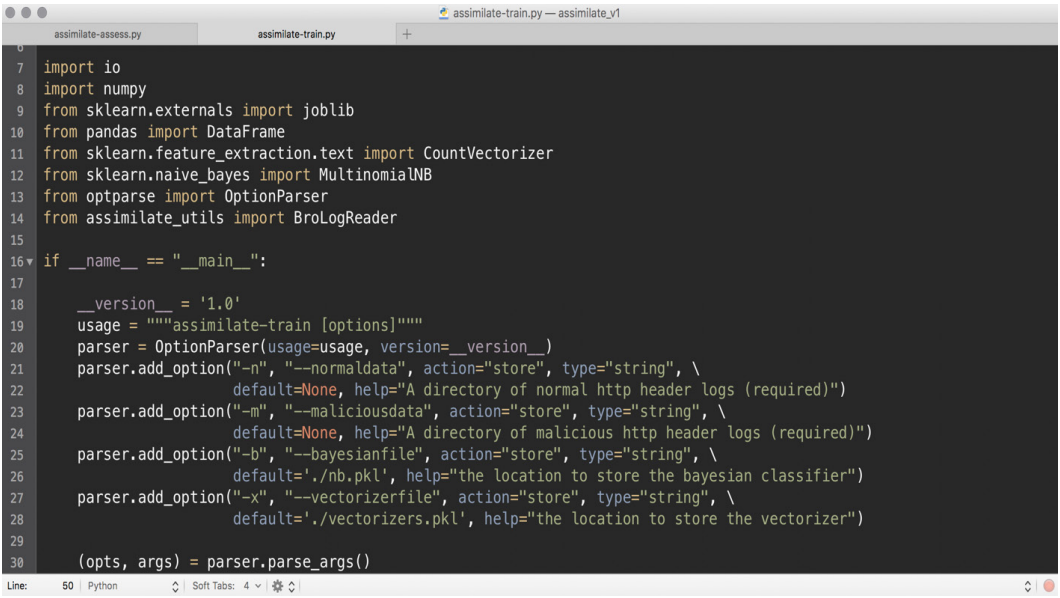


Figure 4 – Modules and Parameters

Our tool is going to need to do a few things. We're going to need to be able to parse our data files into a form that the ML models can process. One key caveat to be aware of is that the calculations can only be done against numerical values. If your data contains text or some other sort of non-numeric data, then it has to be converted into numerical data before the actual processing takes place. This isn't as significant of a problem as it first appears as many of the algorithms actually do the work for you of converting the

data into a numerical form. Speaking of algorithms, you will need to choose which one to use in your tool. Algorithm choice is a topic that could easily be a book itself, so for this discussion, the standard recommendation is to use "Random Forest" if you don't have a specific algorithm you should use. Random Forest is unusual in that it actually uses many algorithms under the hood and calculates which is the most effective internally. Lastly, we'll need two modes for our tool, a training mode and an analysis mode. I usually choose to write two separate pieces of code, but many use a single tool that just has different command line parameters for each mode.

If you want to follow along with the code, you can find it available at https://github.com/Soinull/assimilate. We'll start with assimilate-train.py. That is the code to build our ML model.

The first section loads our modules and sets up the command line parameters.

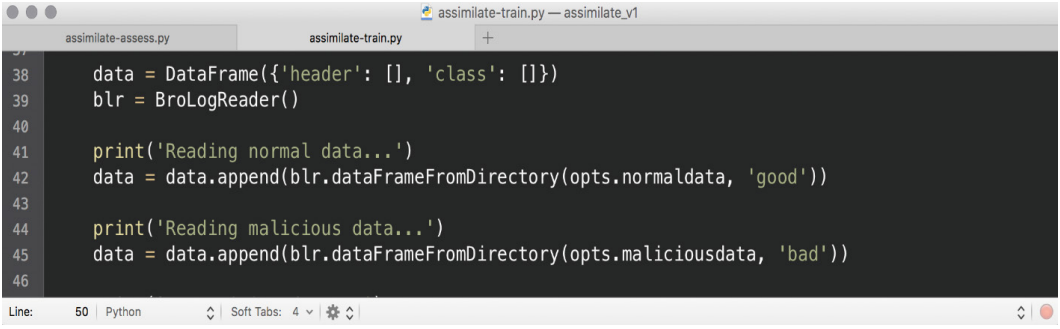


Figure 5 - Pandas

There are a few lines of code to make sure we have the right command line parameters, and then the data is parsed and loaded into a pandas dataframe.

Figure 6 – Data Science code

Line 38 initializes the dataframe. In this case we're using a single feature: the

serialization of the entire HTTP header. We also need a field to label whether the particular row is good or bad (malicious or non-malicious).

Line 42 loads all of the non-malicious data into our dataframe and line 45 loads our malicious Bro http headers.

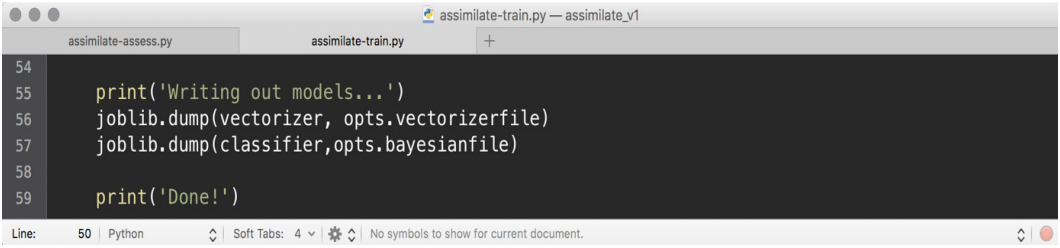


Figure 7 – Writing out modules

Lines 48-53 are where all the data science happens. In Assimilate, I chose to use the Naïve Bayes algorithm. This is the algorithm used to determine whether incoming e-mail is spam or not. We didn't have to convert the textual headers to numerical values because Naïve Bayes already handles that for us.

Figure 8 – Trainer code modules

The final thing we need to do is store our model files. Our entire training code is just 59 lines of Python, although to be fair, there is also a utility module which does all of the Bro parsing work. That file is only 120 lines of code itself.

Next let's look at the analyzer functionality. This script is called assimilate-assess.py and the first lines of code look very much like the trainer code in that it imports the modules we need and then sets up the command line parsing.

Lines 39-45 initialize our various dataframes and model files then lines 48 and 49 load the models created by our trainer code.

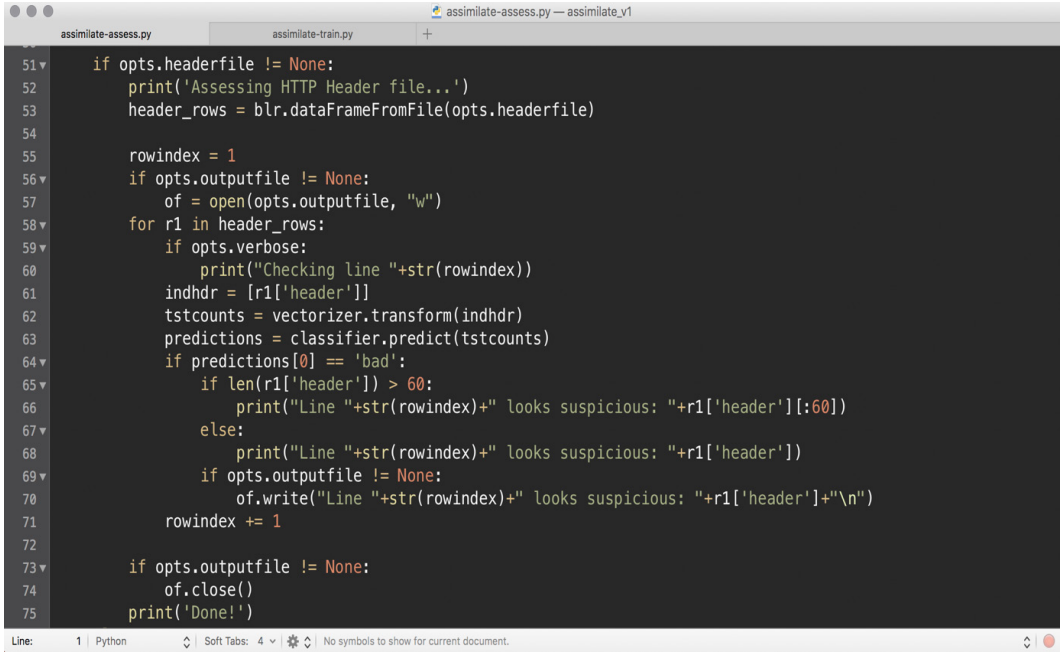


Figure 9 - Writing to a log file

The tool has a fair bit of code for formatting output, but the real work is done in just four lines of code. Line 53 parses and loads the log file to be assessed into a dataframe. Line 61-63 examine a particular header, calculate the vectors on that header, and then compare it to the model to make a prediction if it's malicious (bad) or not. Then the code displays the prediction if it thinks it is malicious and writes it to a log file if the command line option to log to a file was specified.

Lines 77 to 104 do the exact same thing, but with a directory full of log files rather than on a single log file like in Figure 9. And that is all there is too it. Because a lot of really smart people work hard to maintain Pandas, Sci-Kit Learn and NumPy we can create ML models to help us hunt. We do have one more step to do after creating our tools though.

Step Six – Test and Tune

At this point we're ready to create our model and then determine its accuracy.

Assimilate is written to take two directories containing Bro httpheader log files. The "-n" parameter specifies the directory containing the normal files and the "-m" points to the directory containing malicious files. You can see by the output in figure 10 that each folder is read in and parsed and then the model files created and written out.

Once you've built your model files you want to run them against the 20% good and bad you set aside earlier. If your model is 100% accurate then all of your good samples will be classified good and all of your malicious headers will be flagged as malicious. It is common to have just 50-60% accuracy on the first run. If your accuracy is less than 50% then you likely have a problem with your data cleanliness or, less often, with your feature selection. The two key things to keep tweaking are your data set and your feature selection. Good accuracy should get you in the 85-90% range. Anything north of 95% accuracy is considered excellent.

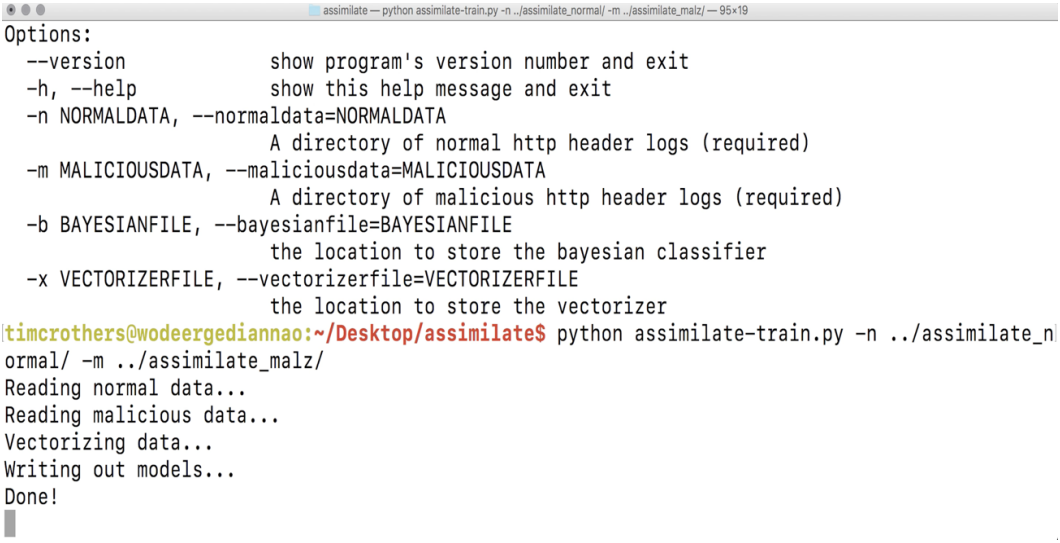


Figure 10 – Creating our model

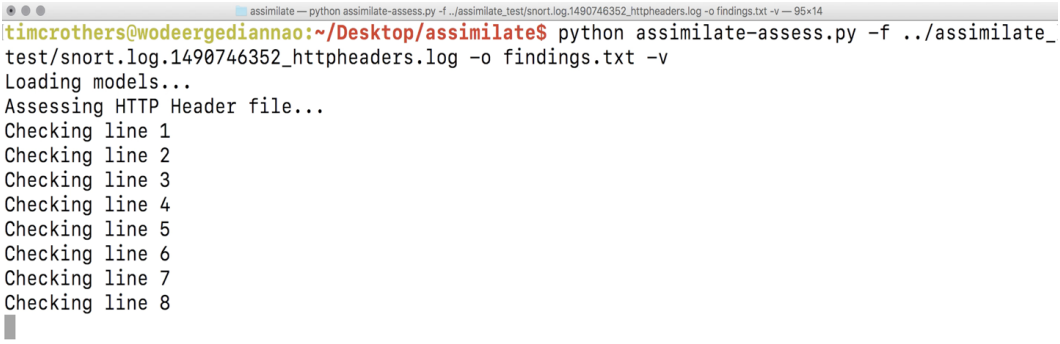


Figure 11

You can see a screen capture of the tool running against one of my test logs. Currently my real models in production are running just over 94% accuracy.

Conclusion

Of course, everything looks easier than it is in real life. Personally, I've been investing 10-20 hours a week for almost three years to get to my current level of proficiency. You don't need to invest that much effort to become proficient in leveraging ML for cyber hunting, however. Scores of books, video training, and classes are now readily available to shorten the learning cycle. And the result is definitely worth the effort. Assimilate, the example I've used here, totals less than 300 lines of code and can be used by novice hunters to accomplish things that previously were only possible with experienced cyber hunters. And just as importantly, Assimilate can be built to run continuously and provide constant hunting, as compared to cyclical hunting as done by people.

Finally, there are a couple other great tools released by others which I strongly recommend checking out. Clearcut, available at https://github.com/DavidJBianco/Clearcut, demonstrates both supervised and unsupervised ML. Clearcut is also designed to find malicious activity in HTTP headers, but uses the standard Bro http.log instead of the httpheaders like Assimilate. Clearcut also supports unsupervised ML, so it will give you some good perspective on how unsupervised ML can be used for hunting.

Malicious Macro Bot, available at https://github.com/egaus/MaliciousMacroBot, showcases an example of how just good ML can be. Malicious Macro Bot includes models and a Docker implementation so you can have it running in minutes. Even more impressively, Malicious Macro Bot is currently running at over 98% accuracy for determining if a macro is malicious or not. Spending a little time studying both of those will give you a lot of sample code you can use to jump start your own ML hunting efforts.

Afterword

Threat Hunting: The Newborn Child in IT Security

Rob Lee

As discussed across this book, threat hunting is a focused and iterative approach to searching out, identifying, and understanding adversaries that have entered the defender's networks. Unfortunately, most organizations are still reacting to alerts and incidents instead of proactively seeking out the threats. Threat hunting itself cannot be fully automated. The act of threat hunting begins where automation ends, although it heavily leverages automation. With that said, many organizations are finding success with a focus on core continuous monitoring technologies and relying on more security automation in their environments to make hunting more effective.

Over the past few years, organizations that are adopting threat hunting tactics and strategies are increasing. But most of the growth is limited to the financial, high tech, military/government, and telecommunications sectors. These are the sectors directly afflicted with targeted attacks by numerous threat groups supported by organized crime and nation-states. Healthcare is inching higher in representation, as this sector has been plagued by ongoing ransomware attacks over the past few years.
Threat hunting is new for most security teams, and it is primarily accomplished through unstructured and untested "hunting" capabilities.

Part of the reason so few teams use externally provided methodologies is that there are not a lot of public reference resources for teams seeking to adopt hunting practices. There are very few published guidelines for proper threat hunting tactics and strategies across the security industry. This book goes a long way to closing the gap on setting some guidelines on effective hunting in most organizations.

Moreover, many organizations are not matured enough to fully adopt threat hunting capabilities. The survey demonstrates that most organizations are still struggling to

adopt more formal threat intelligence capabilities into their security operations centers (SOCs), which is a requirement for proper threat hunting to occur. A significant number of groups, outside the largest of enterprises and governments, are still struggling with putting together security operations and mature incident response capabilities, which are prerequisites for dedicated hunting capabilities.

Threat hunting is new, and its emergence is reminiscent of the initial information security operations of the late 1990s and early 2000s. Most of it is internally learned and not well documented. Helping to correct the lack of overall threat hunting experience, an increasing number of larger organizations have dedicated their teams to threat hunting in their daily security operations. These hunting teams are slowly sharing their results, techniques, and findings with the community through conferences, whitepapers, blogs, and small online collaborative groups.

Given the lack of framework, selecting from the many new products or services specializing in threat hunting is a "buyer beware" environment. Some service and product companies have simply relabeled their products as "threat hunting," because they help "detect threats." Although threat hunting is used to detect threats, it is much more than that.

Hunting is about taking a proactive—as opposed to a reactive—approach to identifying incidents. A reactive organization begins incident response (IR) when an alert or notification comes in. The alert could come from a third party, such as the FBI, or it could come from the organization's own security sensors. The best analogy to a reactive approach is that the IR team is largely waiting to be called into action and relies on the accuracy of the notifications it receives. Most organizations start building their IR teams as a reactive organization, and there is nothing wrong with that. In many cases, the IR team is largely composed of augmentation staff that normally fulfill other duties during their regular jobs. As the organization grows larger, or if it has an increasing number of incidents, the team is likely to become permanent. Even larger organizations likely still augment their IR teams with additional internal personnel or might contract to third-party contractors that provide such services.

Organizations move from being reactive organizations to becoming hunting organizations when they realize they are not detecting their incidents early enough. The idea of a hunting-based response doesn't mean it is an either/or approach. A key element of becoming a hunting organization is adopting a mindset that assumes an adversary is already present. That opens the organization's eyes to detect key indicators of an attack.

Most hunting organizations are also reactive. The distinction is that they begin to task their IR teams to engage actively and hunt for adversaries inside their environment. To accomplish this task, the teams will typically be armed with known malware, patterns of

activity or accurate threat group intelligence to aid them in their search.

Organizations that decide to create a hunting organization sometimes fail to see the importance of proper threat intelligence for driving the search in the right areas. Simply tasking a team to "find evil" isn't enough. The team needs to know the difference between normal and abnormal as a prerequisite. They need to know typical hacker tools and techniques, and they need to be skilled in both network- and host-based forensics and response to look for the footprints of these adversaries. Finally, it helps if the organization has invested heavily in a cyber threat intelligence capability that will help guide the team to the right locations on the network to look for specific indicators and techniques associated with threat groups interested in the specific data or capability that the organization owns. Having such capabilities is an achievable goal. But be prepared. Hunting involves both a manual and a semi-automated scanning of systems looking for evil.

Continuous Hunting: Not There Yet

Asking organizations whether they accomplish threat hunting is a hard question to assess correctly. For example, if you present this scenario to an individual: "You are downtown in a large city walking alone to your car. Do you look for potential threats?" He or she is likely to answer "Yes." Threat hunting in an organization is very similar.

In the SANS 2017 Threat Hunting survey, most organizations feel they conduct threat hunting regularly. Because threat hunting is new, respondents could also categorize activities related to antivirus, IDS or security information, and event management (SIEM) alerts as "threat hunting" activities. These data sources are likely used in hunting, but all three are reactive, and response technologies, whereas hunting, is more proactive than reactive.

The "reactive trend" is a very important takeaway because it shows that most organizations are not accomplishing hunting before detecting the event. They are simply responding to alerts provided by intrusion detection systems. This is "business as usual" in the information security industry. Hunting still plays a large role in the "incident scoping" phase of the incident response, given that the incident intelligence is now guiding the hunters on where to find additional compromised hosts. This phase helps determine the total number of compromised systems and measures the level of severity of the breach.

From our perspective, this is very positive and likely indicates that the organization is approaching formal proactive threat hunting operations and is developing skills along the way. If you ask senior or experienced [illegible] they need to "hunt," most will say

it was through their involvement in IR and scoping the extent of the breach.

The IR team is transitioning from reacting to incidents to hunting for threats themselves is a great sign that many organizations are likely on the cusp of formalizing their threat hunting capabilities over the next few years. The only challenge, which we will discuss later, is retaining the staff that is learning these newfound threat hunting skills.

From my perspective, to accomplish continuous hunting, you must have a set methodology. Without a set methodology, organizations are simply guessing where to look, using random techniques that may or may not work.

Mainly SOC-Based Hunting

One of the key questions of the survey asks: "What activities would initiate an active threat hunt in your environment?" Examining the results, it seems that hunting appears to be more centered on "reactive" indicators instead of proactive intelligence.

This means that more organizations are following a traditional SOC "security monitoring" approach as opposed to utilizing predictive cyber threat intelligence (CTI) to do targeted inspections of likely locations of adversarial activity. Most security teams use alerts/alarms from other tools, which indicates the use of more traditional detection approaches that are not as scalable or accurate. Such approaches can increase false positives. The reactive capabilities being used in "hunting" are not customized to the specific threats facing an organization.

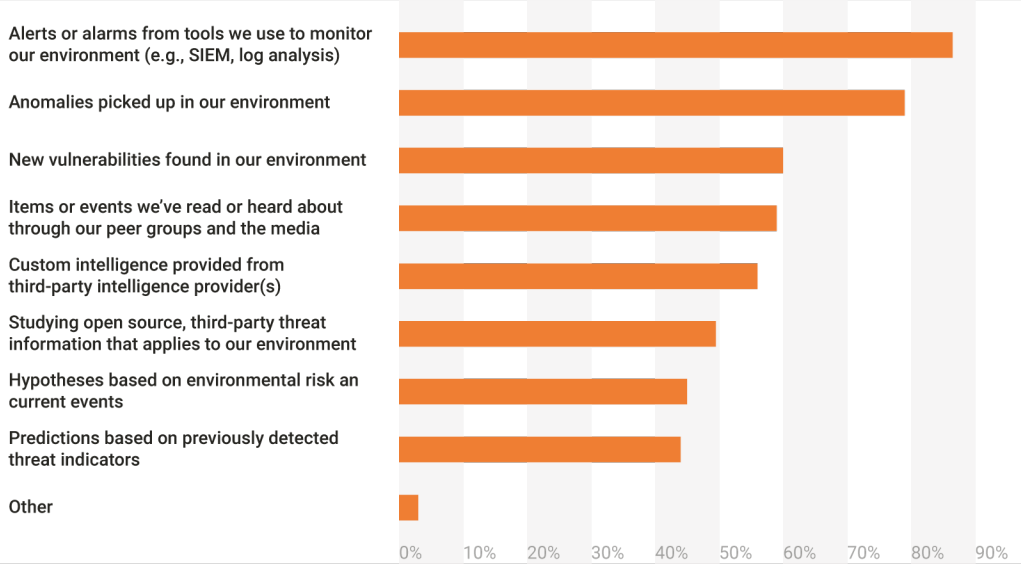
Another contributor to the false positives problem is that according to the SANS Threat Hunting Survey 2017 79% use anomalies in their environments to guide their hunting. Many anomalies are false positives, and differentiating false positives from anomalies is increasingly difficult. Although anomaly detection can provide some success, organizations can leverage more effective hunting capabilities to reduce the time and effort that cyber security teams might encounter while chasing the source of the anomalistic blip on the radar.

What is encouraging is that 49% use open source third-party intelligence in their threat hunting operations, while 57% also use custom intelligence from intelligence providers. Without CTI, organizations attempting proactive hunting are shooting in the dark, as they will have limited knowledge on where to find and uncover likely adversaries targeting their organization. According to the SANS 2017 Cyber Threat Intelligence Survey, many organizations do not yet have a formal internal team working on threat intelligence. To begin this process, we recommend that teams that are just starting out seek companies providing threat intelligence feeds and open source intelligence services, if available.

Bottom line: Organizations that create or ingest modern threat intelligence feeds for tuning sensors to initiate hunting operations are more successful at threat hunting. Organizations need to consider that hunting is more than advanced security monitoring and move their hunting mindset strategically through targeted approaches using threat intelligence to provide hunting accuracy.

In the 2017 SANS Threat Hunting Survey, it is also apparent that many threat hunting organizations are prioritizing "technology" over personnel. Intrusion detection capabilities and proper logging are essential to proper data collection, and hunting requires more than just technology to succeed. Threat intelligence, scalability, and analysts are needed to drive those capabilities to succeed. Threat hunting automation is similar to spell-check in a word processor. While it can help to identify mistakes, it is, by its nature, largely human driven and is more of a tool, rather than true automation.

What activities would initiate an active threar hunt in your environment?
Select all that apply.



Active Hunt Initiators

Threat Hunting Skills and Tools

Threat hunting tools driven by trained analysts can help increase the scalability and accuracy of threat hunting operations. Core technical skill sets and knowledge areas

are also key to a successful threat hunting team. The two key areas defined by the participants include core security operations capabilities and digital forensics and incident response (DFIR) skills. Baseline knowledge of networks and endpoints make up the first tier of knowledge. These knowledge areas are essentially understanding typical network traffic and endpoint services across multiple locations inside an enterprise network. These skills are rated appropriately, as they are core capabilities and are, therefore, mandatory for threat hunting to be effective.

In the same grouping, we also observe both threat intelligence and analytics. Security analytics provide short- and long-term historical views of data in motion and at rest across the network and hosts, enabling threat hunting teams to begin to spot anomalies. Moreover, threat intelligence provides the key difference for hunt teams to focus on areas targeted by adversaries, in addition to being on the lookout for key adversary tactics, techniques, and procedures (TTPs) in use across the enterprise. Without threat intelligence fine-tuning a team's area of focus, most teams find themselves overwhelmed by the massive amount of data they must analyze. DFIR skills make up the second tier of required skills, given that incident response and forensic skills are the baseline capabilities needed to perform hunting on single hosts and at scale across an organization's enterprise network. DFIR skills are usually ranked to follow core capabilities because trained analysts use them to help recognize and extract new threat intelligence used to identify compromised hosts using Tier 1 skills.

The two tiers are grouped perfectly. It is recommended that organizations initially focus on building out the core capabilities when trying to build a threat hunting team. Arm your team with the ability to examine baseline network and endpoints. Then, be able to use security analytics and threat intelligence to identify compromised hosts more efficiently and at a scale that pairs up with the size of your network.

Hunting Automation: Fully Automated or Semi-Automated?

Automation is such a misunderstood word, especially in the context of threat hunting. Hunting needs capabilities to help enhance speed, accuracy, and effectiveness. The best hunting teams heavily leverage automation to aid in increasing the scale and efficiency of hunts across the enterprise. However, by its definition, hunting is best suited for finding the threats that surpass what automation alone can uncover. Threats are, after all, moving targets. Still, it is important to recognize the intertwined nature of automation and the human process of threat hunting.

Tools and capabilities that aid threat hunting is SOC driven. Traditional information

security architecture such as SIEM analytics, log file analysis, intrusion detection and antivirus are largely automated capabilities based on signature-based rules fed and maintained by analysts. When you begin to introduce hunting concepts using these capabilities, they often record, identify, and possibly ignore small anomalies that often are the barely visible tracks of advanced adversaries. Ignoring these trivial anomalies is easy because there are too many to properly vet in even a modest-sized network.
After discovery, most security teams realize that their sensors did, in fact, record the adversaries' activities at the time those alerts occurred. However, the teams were too overwhelmed to pay any attention to them.

These capabilities can be enhanced greatly by utilizing threat intelligence effectively. With proper intelligence, additional threat indicators of compromise and the right analysts using properly tuned tools, some seemingly benign alerts would be identified as major events. Threat hunting, threat intelligence, and security operations can move together in harmony.

Of the 2017 SANS Threat Hunting survey respondents, 16% say they are using threat hunting via fully automated alerts. The number of fully automated hunting capabilities actually in use is a small percentage and is likely tied directly to the fact that hunting is difficult to automate across the many processes and human functions still involved. The majority of respondents stated that their threat hunting capabilities are semi-automated or not automated at all.

The lack of true automation highlights the idea that it is a challenge to fully automate threat hunting. Such a realization brings to the forefront that over-reliance on standard SOC reporting tools such as SIEM/IDS tends to skew reports of hunting effectiveness because they aren't hunting—they are performing intrusion detection. Having said that, SIEM/IDS data can be used to help identify anomalies in hunts.

While I applaud automation in increasing the speed, scalability, and accuracy of threat hunters, understand how much you should automate. Consider seeking tools to enhance and scale hunting activities, but not drive them. It is more appropriate to invest heavily in threat intelligence feeds and training and hiring skilled personnel than it would be to seek capabilities that claim they can fully automate hunting activities.

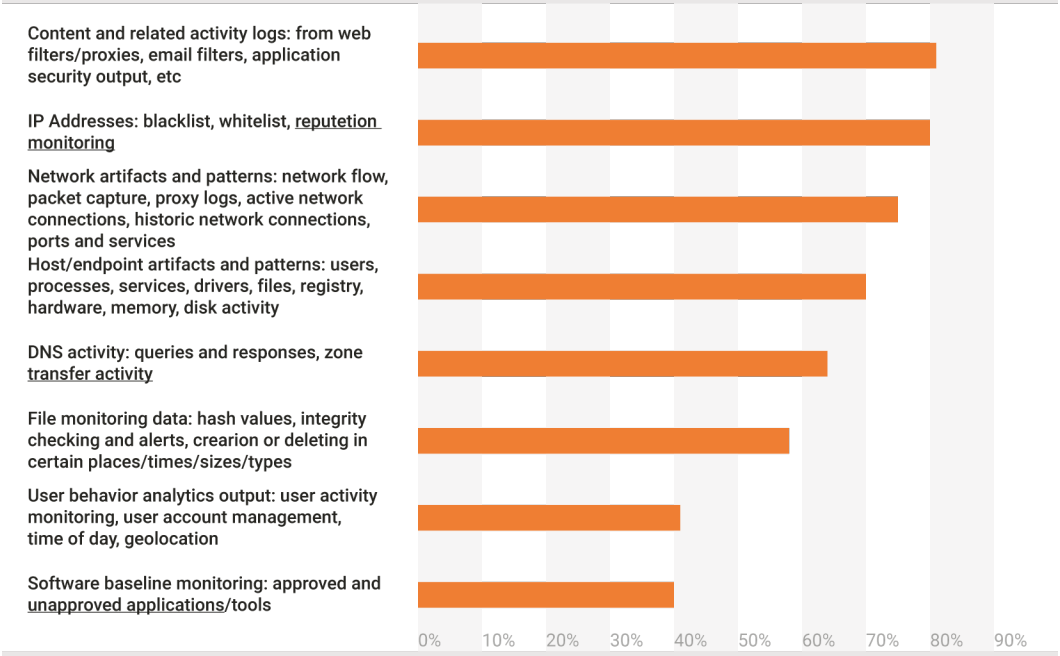
A Choke Point?

In the SANS 2017 Threat Hunting survey, 77% of respondents said endpoint data was critical for conducting hunts, and 73% selected access and authentication logs. Threat intelligence feeds, including vendor and information sharing and analysis center (ISAC) feeds, are rated in the middle of the types of data needed for hunting, being

selected by only 64% of respondents, which also shows that targeted and specific hunts are not being done by respondents because targeted hunts cannot be accomplished without threat intelligence. Targeted hunting with threat intelligence has proven, especially recently, to reduce the dwell time of adversaries in networks, leading to more efficient identification of threats.

While endpoint security data rated high on what is needed, endpoint-related data rated lower on the overall scale of what is actually being collected. Just 71% collect endpoint artifacts and patterns. Another 58% gathered file monitoring data, and 42% monitored user activity. Because many advanced persistent threats (APTs) replicate standard users and credentials, effective threat hunting is limited by the difficulty in detecting activity based on network data alone. The lower numbers in endpoint analytics collected when compared to what respondents believe is needed show that many respondents feel that endpoint data is more obscure and harder to obtain.

What specific collections of data do you collect and analyze during hunt missions?
Select all that apply.



Data Collected and Analyzed During Hunts

Although network data-based data collections were rated high on the survey, endpoint analysis data is still a gaping hole in most hunting operations. Logfile data across multiple hosts can already be easily ingested into SIEM and other analytical systems.

Because it is easier to obtain, the more critical systems you can pull logfiles from, the better the horizon view across an organization. It is often a good idea to log allowed traffic as well as denied or anomalous traffic, since malicious activity, such as data exfiltration, often masquerades as legitimate traffic. Logfile analysis gives perspective to unauthorized credential access, lateral movement by adversaries and malware execution on systems. When collected from multiple systems, logfile analysis can be used to identify anomalies through systems that are not acting in the same manner as the others in your baseline comparisons.

In conclusion

Organizations are beginning to take a proactive approach to threat hunting, taking security into their hands instead of waiting for the breach notice to come across the wire. More organizations need to take that leap. And, they are starting to understand that any threat hunting approach can only be partially automated. In fact, the act of threat hunting leverages the results of automation, but it truly begins where automation ends. Organizations need to ensure that they have appropriately trained staff to provide the needed services. IR team members are the logical ones to tap and expand their knowledge base to engage in threat hunting.

Increased investments will be made in this space. That means there will be organizations, teams, and individuals coming forward with tools, people, and processes that work for them but that may not be well suited for all environments. It is of paramount importance that teams looking to build a threat hunting capability educate themselves appropriately so that they can make wise investments.

Contributors

Index of contributors in alphabetical order.

Danny Akacki

Danny is just a hacker who adores what he does and the community that surrounds him. He's had the honor of working for such companies as Mandiant, GE Capital, Bank of America, and currently works as a Security Evangelist with Sqrrl. He is an enthusiastic speaker with a love of Tribal Knowledge Sharing, Japanese whisky and lock picks. Danny has spoken at events like BSides Augusta, BSides Philadelphia, and is one-third of the Rally Security podcast crew. He is also the creator of DEF CON 610 and InfoSanity.org, a resource and sounding board for those dealing with mental health issues in the Information Security community.

Samuel Alonso

Samuel is a cyber security specialist with a keen interest in incident response and threat hunting. He currently works at KPMG where he is part of a group of highly specialized experts defending KPMG networks. On a daily basis, he responds to different incidents across KPMG's business ecosystems and is part of their threat hunting teams actively chasing threats in their networks.

Richard Bejtlich

Richard Bejtlich is Chief Security Strategist at FireEye, and was Mandiant's Chief Security Officer when FireEye acquired Mandiant in 2013. He is a nonresident senior fellow at the Brookings Institution, a board member at the Open Information Security Foundation, and an advisor to Threat Stack, Sqrrl, and Critical Stack. He is also a Master/Doctor of Philosophy in War Studies Researcher at King's College London. Richard is a graduate of Harvard University and the United States Air Force Academy. His fourth book is The Practice of Network Security Monitoring (nostarch.com/nsm).

David Bianco

David has been involved in computer and network security for 15 years, and has led consulting engagements with Fortune 500 companies, Wall Street firms, public utilities, and major universities.

Sergio Caltagirone

Sergio worked for the U.S. government, Microsoft, and Dragos. He has not only hunted the most sophisticated targeted threats in the world, but also applied that intelligence to protect billions of users worldwide and safeguarding civilization through the protection of critical infrastructure and industrial control systems. He also proudly serves as the Technical Director of the Global Emancipation Network, a non-profit non-governmental organization (NGO), leading a world-class all-volunteer team hunting human traffickers and finding their victims through data science and analytics working towards saving tens of millions of lives.

Jack Crook

Jack is a former US Army Infantryman, and currently works as the principal incident responder for a Fortune 10 company. He has written extensively on threat hunting theory and practice.

Tim Crothers

Tim is a seasoned security leader with over 20 years' experience building and running information security programs, large and complex incident response engagements, and threat and vulnerability assessments. He has deep experience in cyber-threat intelligence, reverse engineering, and computer forensics. He is a recognized thought leader and author/co-author of 14 books to date, as well as regular training and speaking engagements at information security conferences.

Tyler Hudak

Tyler is an Incident Response and Reverse Engineering Specialist with over 15 years of Information Security Experience in multiple industries. He has presented at multiple local and national information security conferences and created and taught multiple training courses on malware analysis and incident response.

Rob Lee

Rob Lee is the curriculum lead and author for digital forensic and incident response training at the SANS Institute. With more than 15 years of experience in computer forensics, vulnerability and exploit discovery, intrusion detection/prevention, and incident response, he provides consulting services in the Washington, D.C. area. Before starting his own business, Rob worked with government agencies in the

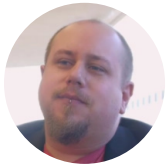
law enforcement, defense, and intelligence communities as a lead for vulnerability discovery and exploit development teams, a cyber forensics branch, and a computer forensic and security software development team. Rob was also a director for MANDIANT, a company focused on investigating advanced adversaries, such as the APT, for five years before starting his own business and co-authored Know Your Enemy: Learning About Security Threats, 2nd Edition.

Robert M. Lee

Robert M. Lee is a SANS certified instructor and author of the ICS Active Defense and Incident Response and Cyber Threat Intelligence courses, is the founder and CEO of Dragos, a critical infrastructure cyber security company, where he focuses on control system traffic analysis, incident response, and threat intelligence research. He has performed defense, intelligence, and attack missions in various government organizations, including the establishment of a first-of-its-kind ICS/SCADA cyber threat intelligence and intrusion analysis mission. Author of SCADA and Me as well as Threat Intelligence and Me, he is a nonresident National Cyber Security Fellow at New America, focusing on critical infrastructure cyber security policy issues. Robert was named EnergySec's 2015 Energy Sector Security Professional of the Year and one of Forbes' 30 under 30 (2016).

Josh Liburdi

Josh is the lead detection engineer at a Fortune 50 company. He has over 4 years' hands-on experience in the threat detection and incident response domain. He is a contributor to multiple open-source network security monitoring projects and communities and has been a presenter/trainer at regional, national, and global information security conferences



Ryan Nolette

Ryan is a security technologist at Sqrrl. Throughout his career he has attained experience in IT/Security planning at a large scale and is proficient in multiple platforms and security techniques. He has experience with troubleshooting, auditing and installations, network intrusion detection, security, incident response, threat intelligence, threat research, computer forensics, and network/systems auditing.

Scott Roberts

Over the past 12 years, Scott has served in a variety of capacities, garnering technical experience in many areas. His career took him through being a Windows System Admin to a Network Admin and finally landing as a Security Specialist. He is currently serving as a Security Assessor in a NERC/CIP and FISMA environment.

Chris Sanders

Chris Sanders is an information security author, trainer, and researcher originally from Mayfield, KY. He is the founder of Applied Network Defense, a company focused on delivering high quality, accessible information security training. In previous roles, Chris worked with the US Department of Defense, InGuardians, and Mandiant to build security operation centers and train practitioners focused on defending defense, government, and Fortune 500 networks.