Detecting Rogue Ethernet Switches Using Layer 1 Techniques

Author: Jennifer Walker, jwalker0129@gmail.com
Advisor: *David Fletcher*

Abstract

Ethernet is a ubiquitous standard for local area networks (LAN) and commonly used in other networks. Rogue Ethernet switches are a security and performance threat to networks. These rogue switches are a unique challenge as Ethernet switching is invisible to most network detection methods. Most detection methods rely on the Network Layer (Layer 3) or the Data Link Layer (Layer 2) to generate traffic on the network. This paper will analyze three Physical Layer (Layer 1) techniques that do not require network traffic to detect the rogue switch. This paper will evaluate each technique's ability to detect rogue switches, the possibility of generating false readings, and the possibility of automating the technique. Techniques will be done from both the switch and server perspective for comparison.

1. Introduction

Rogue Ethernet switches introduce multiple issues when added to a network. These unapproved switches add insecure devices, affect network performance, cross network boundaries, or stealthily capture packets. Their detection is uniquely difficult compared to Layer 3 devices.

The security implications of Ethernet are far-reaching. Originally just for enterprise networks, Ethernet is now in Industrial Critical Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) networks. A SCADA network for electrical power systems was brought down for hours due to misconfigurations as simple as plugging an additional cable in the wrong spot (NERC, 2018). In one instance, users added a VoIP phone to a conference room by connecting it to the network with two cables. The switch in the phone caused a network loop and created a broadcast storm that brought down the SCADA network. Malicious or otherwise, rogue switches can degrade the network, affecting service level agreements (SLAs) or access to critical systems.

Ethernet's plug-and-play nature makes it easy to change the network. Users add a switch simply for the ease of adding more devices. Even an administrator may add a switch as a temporary fix that becomes permanent but undocumented. These switches are shadow IT used for convenience or as a workaround for a perceived problem (Kopper & Westner, 2016). Major outages or events, such as the COVID-19 pandemic, contribute to shadow IT infrastructure. In these cases, quick solutions become the priority over procedures and security. Employee turnover exasperates the situation, where the new employees fear making changes to unknown yet possibly critical devices. A more malicious purpose is to monitor the network silently by using port mirroring on a rogue switch. Shadow IT's undocumented nature is a major issue when troubleshooting, guaranteeing network performance, and securing the network.

Ethernet switching is invisible to traditional device detection methods. Network administrators use IP addresses, MAC addresses, and various management protocols for tracking switches. However, none of these are required for an Ethernet switch to forward frames (James, 2019). An Ethernet switch does not even need its own MAC address to

work. Instead, it analyzes the frame's source and destination MAC address to determine where to forward it (Kirauvo et al., 2013). Store-and-forward switches can be detected by the delay they cause from reading the entire frame, but not cut-through switches that only read enough of the header to forward it (Alcock et al., 2005). Unmanaged switches and managed switches configured not to generate packets are undetectable by typical packet analysis.

Current research on the Physical Layer, or Layer 1, techniques is limited. The most common Layer 1 technique focuses on changing the switch port status and tracking if the end device network connection also goes down (Quitiqut & Bhuse, 2022). Further validation can be done by interrogating upper-layer protocols, such as DHCP and ARP, to see if they behave correctly during the switch interface shutdown (Bhuse et al., 2019). Only one paper tested bringing down the interface on the end device side of the link and found anomalous behavior. (Petrenko, 2015). Beyond changing port status, Petrenko tested Time-Domain Reflectometry (TDR) and speed/duplex changes, but tests were limited due to the availability of different hardware at the time.

This paper will further test Layer 1 techniques for both switches and servers. The behavior of switches and servers will be compared in executing each technique and collecting the data. Finally, the ability to automate these tests will be considered.

2. Research Method

Three techniques were tested: TDR, auto-negotiation values, and port status. Each technique compared the values of two devices assumed to be directly connected, referred to as link partners in the Ethernet standard (IEEE, 2022). Mismatched values indicated a rogue switch. Each technique was evaluated for its effect on the network availability and the possibility of false positives and negatives.

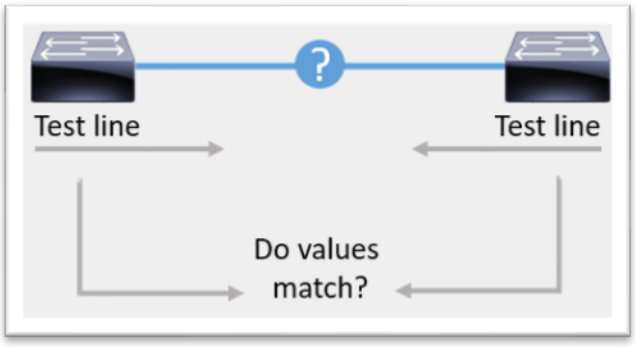

Figure 1. General Test Procedure

TDR measures the physical characteristics of copper cabling. TDR sends an electrical pulse over copper wires to measure each wire pair's length and status (Lunn, 2019). A mismatch in TDR length values implied that the cable was changed to add a rogue switch. TDR is available on many managed switches and is now available on network cards. A cable tester was used to verify the accuracy of the TDR measurements.

Auto-negotiation allows for link partners to negotiate the best speed and duplex. The devices do this by advertising their available speeds and duplex modes. The link partner should see any advertised mode changes on the local device. A rogue switch intercepted the advertised modes if the link partner did not see the change. Changing auto-negotiation values has advantages over hard coding speed or duplex changes to test the line. First, it is possible to change the auto-negotiation value so that it still picks the best possible mode. Second, turning off auto-negotiation also turns off auto medium-dependent interface crossover (auto MDI-X). However, some vendors may implement a specialized version that does not require auto-negotiation (Dove, 2008). MDI-X refers to the fact that the wires used to transmit on one device must connect to the receive wires on the remote device. Auto MDI-X negotiates which wires receive and transmit between the link partners. Manually, the wire order on each end of the cable accomplishes this. Converting to a manual MDI-X configuration can cause a loss of connection to the remote device if the incorrect cable is used.

Port status refers to the interface status as either up or down. Terminology may vary across vendors or commands. On Linux, the ip command uses the term "state". Like the auto-negotiation test, a change on one device causes a related change on the link

partner device. In this case, the port status was changed to down on one device, and the link partner was observed for the status change. A rogue switch kept the connection up if the link partner port status did not change to down. Switch and server behavior of changing port status were compared. Petrenko found that changing the server port status caused the switch's port to go down and immediately back up (2019). The server-side test also attempted to replicate this down/up to isolate the cause.

The lab is a simple design to test three hardware combinations and two rogue switches. The baseline network contains two Cisco 3850 switches and two Dell PowerEdge R530 servers (See Appendix A for more details). Server 2 uses the onboard Broadcom NetXtreme network card, while Server1 has an additional Intel X550T2 network card installed for its TDR capability. The rogue switches include a generic 5-port unmanaged switch and a managed Cisco 3750 switch. The features tested are not specific to these vendors, but availability and implementation may vary across product lines.

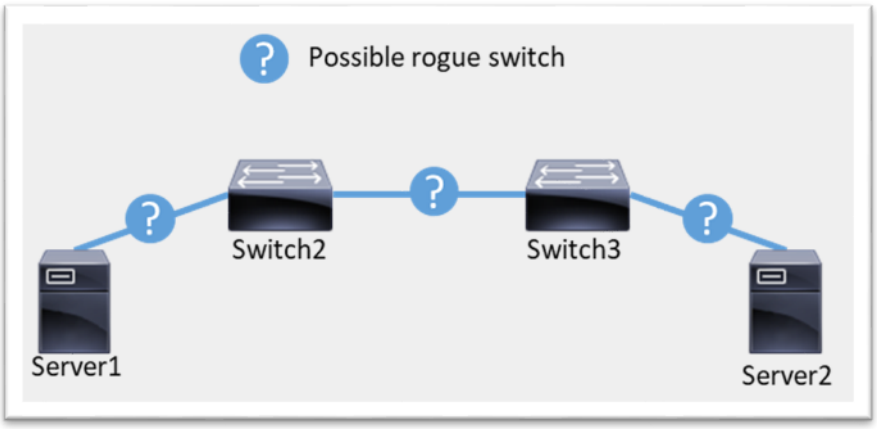

Figure 2. Lab Network

3. Findings and Discussion

Network disruptions were detected by a simple ping script executed for one minute during each test. A baseline ping test was run prior to each rogue detection test. The baseline tests did not have any dropped packets. All detection techniques caused packets to drop except for the TDR test on Cisco switches. Therefore, none of the techniques are guaranteed to be non-disruptive.

```
#Ping every 1/10 second for one minute showing only the summary
ping -q -i 0.1 -c 600 192.168.1.11
```

Figure 3. Ping command used to test network availability

3.1. TDR Findings

The cable length values for TDR are overall not accurate but consistent for a set of hardware. Unexpectedly, the link partner device dramatically affects the cable length value for the Cisco switches. Cable lengths for unplugged cables are accurate within 1 meter for all the Cisco switches. The Intel network card readings were accurate within 1 meter even when connected to a link partner.

To test the TDR cable length accuracy, a NetAlly cable tester measured eight cables varying in length from about 1 meter to 15 meters. All lengths were rounded to the nearest meter as the network device TDR values only gave whole meter lengths. Cables less than 1m were measured with a measuring tape. Cables are named based on length and given an additional letter identifier for cables with the same length (See Appendix A for more details).

Cable Name	Net Ally (meters)	Rounded Length (meters)
1mA	<1 (.9)	1
1mB	<1 (.9)	1
1mC	<1 (.9)	1
3mA	3.2	3
3mB	3.2	3
3mC	3.2	3
8m	8	8
15m	15.3	15

Table 1. Cable Names and Lengths

3.1.1. TDR on Switches

The Cisco switches provide TDR support to troubleshoot cabling. The primary use case for TDR is to find physical faults in the cable. There are three commands related to TDR. On current versions of the operating system, the test command will run in the background after being executed. The results are display or cleared with the related TDR command.

Command	Description
test cable-diagnostics tdr interface [*interface number*]	Executes TDR test
show cable-dianostics tdr interface [*interface number*]	Shows results of TDR test
clear cable-diagnostics tdr [interface *interface number*]	Clear TDR results for all interfaces or per interface

Table 2. Cisco TDR commands

Test results are displayed with the "show cable-diagnostics tdr" command. Per wire pair values include local pair position, remote pair position, length with margin of error, and status. Wire pairs for both local and remote end of the cable are labeled A through D. Generally, the pair length values are within one meter of each other unless there is an issue with the cable. The length of a good cable can be estimated by taking the average of all pairs and rounding to the nearest meter. Per cable values include the interface name and speed.

```
Interface   Speed Local pair Pair length        Remote pair Pair status
---------   ----- ---------- ------------------ ----------- ------------
Gi1/0/1     1000M Pair A     14   +/- 10 meters Pair B      Normal
                  Pair B     14   +/- 10 meters Pair A      Normal
                  Pair C     14   +/- 10 meters Pair D      Normal
                  Pair D     14   +/- 10 meters Pair C      Normal
```

Figure 4. Example Output "show cable-diagnostics tdr interface g1/0/1"

Pair status describes any faults on wire pair. The "normal" status is the expected operation status when the remote end of the wires has physical contact with the link partner device. If all pairs are "open," then the cable is unplugged (or the cable could be cut or has a bad connector, but less likely and assumed not the reason for this research). If only some pairs are open, the cable is bad, as that pair has no pin contact. The status values encountered during this research are listed in the chart below.

Pair Status	Description
Normal	Connected to link partner
Open	Not connected to link partner
Not Completed	TDR test in process
Short/Crosstalk	Shorted wires

Table 3. Cisco TDR Pair Status Description

3.1.2. TDR on Servers

The TDR function on a network card is less common, and finding one that explicitly states TDR support proved challenging. The Intel X550 was selected for its TDR cable diagnostics support. The plan was to use the "ethtool --cable-test-tdr" command, but the Intel X550 driver does not support this feature.

The Intel X550 registers must be accessed directly to execute the TDR test. A total of six 16-bit registers are used for the TDR, along with one register to check the connection state (Intel, 2023). One register is written to execute the TDR diagnostic test. The other five registers provide the results of the TDR test.

Register	Bits	Description
1E.C470	4	Initiate cable diagnostics
7.C810	D:9	Connection State
1E.C800	2:0	Pair D status
	6:4	Pair C status
	A:8	Pair B status
	E:C	Pair A status
1E.C801	F:8	Pair A length
1E.C803	F:8	Pair B length
1E.C805	F:8	Pair C length
1E.C807	F:8	Pair D length

Table 4. Intel X550 TDR Registers

The phytool package provides command-line access to hardware registers. The phytool uses the device name, address, and register to read the register's values. The write command of phytool is a similar format, except it includes the target value of the register. All addresses and values are in hexadecimal.

```
#Get current register value
./phytool read enp4s0f0/0x1e/0xc470
0x0e00

#Start cable diagnostic test
./phytool write enp4s0f0/0x1e/0xc470 0x0e10
```

Figure 5. Example Phytool Commands to run TDR for Intel X550

The Intel X550 datasheet describes how to use and read the six TDR registers in detail. In comparison, the onboard Broadcom NetXtreme network card does have two TDR-related registers, but the programmer's guide does not describe their usage (Broadcom, 2022). Therefore, this paper did not attempt to use them.

3.1.3. TDR Accuracy

The TDR accuracy varies across devices. For all the Cisco switches, accuracy drops when the cable is plugged into a remote device. Also, the length values change depending on the remote device. Even so, there were still patterns within the Cisco TDR values. The length readings were consistent across multiple TDR tests for any set of hardware.

Regarding accuracy, the Cisco 3750 switch consistently reported a shorter cable length except when plugged into the Intel network card, when it became pretty accurate. Both Cisco 3850 switches reported longer cable lengths. In contrast to the Cisco 3750, the Cisco 3850 lengths became even longer when plugged into the Intel network card. The Intel card consistently reported the correct length within 1 meter.

Cable	Length	Cisco 3750 Switch1			Cisco 3850 Switch2			Cisco 3850 Switch3			Intel X550 Server1		
		Min	Max	Var	Min	Max	Var	Min	Max	Var	Min	Max	Var
1mA	1	0	1	1	7	16	15	8	19	18	1	1	0
1mB	1	0	1	1	7	16	15	8	19	18	1	1	0
1mC	1	0	1	1	7	16	15	8	18	17	1	1	0
3mA	3	0	4	3	11	20	17	14	24	21	3	3	0
3mB	3	0	4	3	10	19	16	19	24	21	3	3	0
3mC	3	0	4	3	10	19	16	11	21	18	3	3	0
8m	8	1	11	7	15	25	17	16	27	19	8	8	0
15m	15	4	15	11	19	29	14	23	33	18	15	16	1

Table 5. TDR Lengths and Variance for Cable Plugged into a Remote Device

All devices give accurate lengths when the cable's remote end is unplugged. As mentioned previously, cable lengths given here are the average of wire pair lengths. The unplugged cable results had slightly more variable per wire lengths, but the average length is more accurate (see Appendix B for detailed TDR readings).

Cable	Switch1	Switch2	Switch3	Intel X550
1m A	1	1	0	1
1m B	1	0	0	0
1m C	1	0	1	1
3m A	3	3	3	3
3m B	2	3	3	3
3m C	3	3	3	3
8m	8	8	8	8
15m	15	16	16	16

Table 6. TDR Result for Unplugged Cable

The accuracy of the Intel X550 card may be due to it bringing down the connection during the test. The Cisco switches keep the link up during the TDR test. It was attempted to see if a more accurate reading could be obtained by manually bringing the link down for the Cisco switch. Trying to run TDR when the port is shutdown gives an error. Bringing down the link partner's interface causes the Cisco switch to return a length of "n/a" though all pair statuses were still "normal."

```
switch2#test cable-diagnostics tdr int g1/0/2
% Interface Gi1/0/2 is administratively down
% Use 'no shutdown' to enable interface before TDR test start.
switch2#
```

Figure 6. Error when Attempting to Run TDR with Down Interface

```
switch1#sho cable-diagnostics tdr int g1/0/11
TDR test last run on: February 26 04:19:51

Interface Speed Local pair Pair length        Remote pair Pair status
--------- ----- ---------- ------------------ ----------- ------------
Gi1/0/11  auto  Pair A     N/A                N/A         Normal
                Pair B     N/A                N/A         Normal
                Pair C     N/A                N/A         Normal
                Pair D     N/A                N/A         Normal
```

Figure 7. TDR Results when Link Partner's Interface is Down

3.1.4. Failed Hardware

The original lab had Intel X550 cards in both servers, but the card died in Server2 while testing cable lengths. Both ports on the network card stopped working. A TDR test from Switch1 shows a short in the network card as the cause of the issue. This TDR test

used the 15m cable. An additional TDR test (not shown) shows the cable was fine when plugged into a different device.

```
switch1#sh cable-diagnostics tdr int g1/0/24
TDR test last run on: April 14 05:34:36

Interface Speed Local pair Pair length        Remote pair Pair status
--------- ----- ---------- ------------------ ----------- ---------------
Gi1/0/24  auto  Pair A     182  +/- 10 meters N/A         Short/Crosstalk
                Pair B     N/A                N/A         Normal
                Pair C     N/A                N/A         Normal
                Pair D     N/A                N/A         Normal
```

Figure 8. TDR Result on Failed Network Card

Two of the cables failed during testing. The constant plugging and unplugging of cables caused some wires to fail. A TDR test verified the cable failure by showing that some pairs were no longer in contact with the remote device.

```
Switch3#sh cable-diagnostics tdr interface g1/0/21
TDR test last run on: August 14 15:03:01

Interface   Speed Local pair Pair length        Remote pair Pair status
---------   ----- ---------- ------------------ ----------- -------------
Gi1/0/21    auto  Pair A     4    +/- 1  meters N/A         Open
                  Pair B     0    +/- 1  meters N/A         Normal
                  Pair C     3    +/- 1  meters N/A         Open
                  Pair D     5    +/- 1  meters N/A         Open
```

Figure 9. TDR Result on Failed Cable

All the cables lasted through the initial length tests, but 3mA and 3mB failed during the rogue switch detection. These cables were removed from the test due to these failures. It was decided only to use cables 1mA and 1mC (both newer cables) for the rogue switch tests in case other cables also failed.

3.1.5. TDR Rogue Switch Detection

The inaccurate nature of the TDR readings means the comparison of link partner values cannot be used to detect rogue switches. Instead, the consistent nature of the TDR readings can be used to detect rogue switches by comparing each local test to a baseline value. This allows each side to test the link independently. For the baseline to be accurate, it must happen in a known network state with no rogue switches.

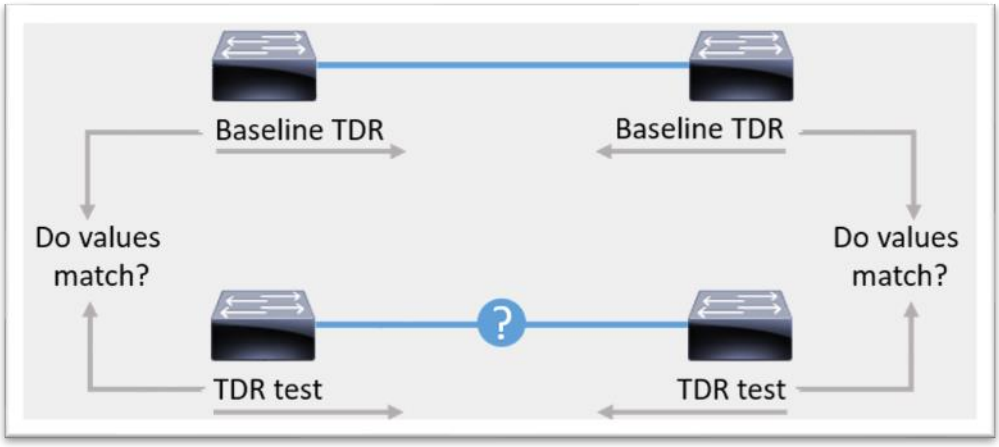

Figure 10. TDR Test Procedure

Between the 3850 switches, both Switch1 and the Unmanaged switch were detected. Switch3 detected both rogue switches while Switch2 detected only Switch1. Switch2 did not detect the Unmanaged switch because the TDR length variation was within 1 meter.

Device	Baseline	Unmanaged	Switch1	Detection
Switch2	7	8	**14**	Detected Switch1
Switch3	7	**10**	**15**	Detected Unmanaged and Switch1

Table 7. TDR Rogue Switch Detection Results between Switch2 and Switch3

Between Switch2 and Server1, only the Unmanaged switch was detected. Since the cables used to add the rogue switch were the same length, Server1 did not detect any changes. Switch2 detected the Unmanaged switch. Switch2 did not detect Switch1 because the TDR length variation was within 1 meter.

Device	Baseline	Unmanaged	Switch1	Detection
Switch2	15	**8**	14	Detected Unmanaged
Server1	1	1	1	No rogue detection

Table 8. TDR Rogue Switch Detection Results between Switch2 and Server1

False negatives are possible. Though the TDR values from the Cisco switches are sensitive to hardware changes, there is still a relatively limited possible value set. It is more challenging to get false positives from both switches. In this case, at least one

switch detected the rogue switch. Interestingly, the Intel X550 has a higher chance of false negatives due to its accurate readings. Finally, false positives are possible from hardware failures. The hardware or cable failure can change the length values. If pair status is also considered, then it should be evident that the change is due to a hardware issue.

3.2. Auto-Negotiation Findings

Link partners exchange negotiation values to select the best speed and duplex. The local device tracks the link partner's auto-negotiation mode and its own. Clause 22 and 45 of the Ethernet standard define auto-negotiation registers, with the latter covering higher bandwidths starting at 10Gb and Energy Efficient Ethernet (IEEE, 2022). Implementation of the Clauses varies across the hardware

Devices	Supported Clauses	Notes
Cisco Switches	22	Max Speed 1Gb.
Broadcom NetXtreme	22 & 45	Max Speed 1Gb, Energy Efficient Ethernet support
Intel X550	45	Max Speed 10Gb, Energy Efficient Ethernet support

Table 9. Supported Clauses for each Device type

All devices could view and change auto-negotiation values, but execution varied. Unlike TDR, the auto-negotiation results were accurate and consistent on all devices. Also, unlike TDR, changing the auto-negotiation value disrupted the network in all cases.

3.2.1. Auto-Negotiation on Switches

Cisco switches provide multiple show commands to see the current speed and duplex. Viewing the advertised modes for the local device and the link partner is more obscure. The auto-negotiation values are viewed through the Clause 22 registers. Register four stores the local device advertised values, and register five stores the link partner's values. These registers only store the modes for 10Mb and 100Mb. The "show controller ethernet-controller <interface> phy" command is used to view the registers. For each of these registers, bits 8 and 7 are the 100Mb modes and the ones used for the testing. These

values specifically are used because the Intel X550 network card cannot see the 10Mb values advertised by the link partner.

```
0000 : 1140                 Control Register :  0001 0001 0100 0000
0001 : 796d                   Control STATUS :  0111 1001 0110 1101
0002 : 0141                          Phy ID 1 :  0000 0001 0100 0001
0003 : 0ed4                          Phy ID 2 :  0000 1110 1101 0100
0004 : 01e1    Auto-Negotiation Advertisement :  0000 0001 1110 0001
0005 : c1e1     Auto-Negotiation Link Partner :  1100 0001 1110 0001
0006 : 000d    Auto-Negotiation Expansion Reg :  0000 0000 0000 1101
0007 : 2001       Next Page Transmit Register :  0010 0000 0000 0001
0008 : 4fef  Link Partner Next page Register  :  0100 1111 1110 1111
0010 : 3870              PHY Specific Control :  0011 1000 0111 0000
0011 : ac08               PHY Specific Status :  1010 1100 0000 1000
0012 : 6400     PHY Specific Interrupt Enable :  0110 0100 0000 0000
0013 : 0000     PHY Specific Interrupt Status :  0000 0000 0000 0000
```

Figure 11. Example Output “show controllers ethernet-controller g1/0/11 phy”

The “speed auto” command can change advertised modes. For example, to exclude the 100Mb speed, the command is “speed auto 10 1000”. This command will turn off bits 8 and 7 in register four. There is no way to turn off 100Mb half-duplex and 100Mb full-duplex separately for Cisco switches.

3.2.2. Auto-Negotiation on Servers

Ethtool on Linux can display all the auto-negotiation values but depends on the network card’s driver to provide the information. The Intel X550 only shows the local device’s advertised values. On the other hand, the Broadcom NetXtreme shows both the local device and link partner values. Below is the Broadcom network card output using the “ethtool <interface>” command.

```
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                             100baseT/Half 100baseT/Full
                                             1000baseT/Full
```

Figure 12. Example Output of “ethtool eno1”

For the Intel X550, the phytool command is needed again to view the link partner advertised modes. In this case, the clause 45 register at address 7.13 holds the bits 8 and 7 with the link partner's 100Mb values. The local advertised modes are viewed with the ethtool command. Changing the advertised modes is the same for both network cards. The command "ethtool -s <interface> advertise <mode value>" sets the combination of modes advertised by the local device. Unlike the Cisco speed command, Ethtool can set auto-negotiation values per speed/duplex mode.

3.2.3. Auto-Negotiation Rogue Switch Detection

Changing the auto-negotiation modes detected the rogue switch in all instances. In the case, the baseline values can be retrieved prior to the test even if there is a rogue switch. For this test, only the 100Mb register bits were used and one or both 100Mb modes were removed from the advertised register for the local device. Once auto-negotiation completed, the remote device link partner register was compared to the local device's no rogue values. A mismatch in values means a rogue switch intercepted the auto-negotiation values.

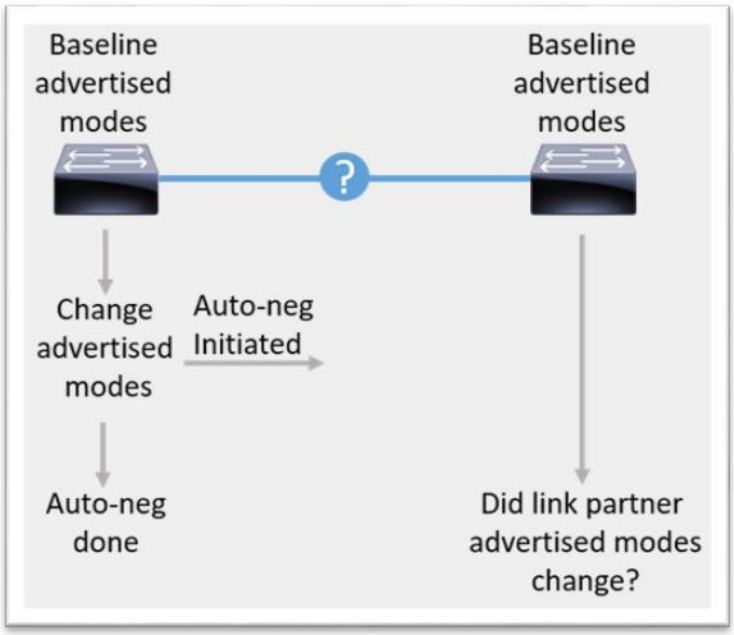

Figure 13. Auto-negotiation Test Procedure

The tests values from changing Switch3 advertised modes are the same as Switch2 values shown in the table below. In all cases, the lack of change from the baseline values detected the rogue switch. Between Server1 and Switch2, the rogue switches could have been passively detected since Server1 does not support half duplex modes.

	Switch2: Advertised Modes		**Switch3: Advertised Link Partner Modes**	
	100Mb Full	100Mb Half	100Mb Full	100Mb Half
Baseline	1	1	1	1
Removed 100Mb Half and Full Duplex values				
No rogue	0	0	0	0
Switch1	0	0	**1**	**1**
Unmanaged	0	0	**1**	**1**
	Sever1: Advertised Modes		**Switch2: Advertised Link Partner Modes**	
	100Mb Full	100Mb Half	100Mb Full	100Mb Half
Baseline	1	0	1	0
Removed 100Mb Full duplex value				
No rogue	0	0	0	0
Switch1	0	0	**1**	**1**
Unmanaged	0	0	**1**	**1**
	Sever2: Advertised Modes		**Switch3: Advertised Link Partner Modes**	
	100Mb Full	100Mb Half	100Mb Full	100Mb Half
Baseline	1	1	1	1
Removed 100Mb Half duplex value				
No rogue	1	0	1	0
Switch1	1	0	1	**1**
Unmanaged	1	0	1	**1**

Table 10. Auto-negotiation Rogue Switch Detection Results

False readings are only possible if the hardware or auto-negotiation values were changed during the test. Both are unlikely to happen during the short time it takes to re-negotiate speed and duplex. Most likely hardware change that could happen in the short time is moving a cable. Changing the auto-negotiation values is very unlikely in normal network operations unless someone happens to run this same test.

3.3. Port Status Findings

Manually changing the port status is the most straightforward technique to execute and observe across the devices. Tracking downtime on the switch side was more straightforward than on the servers. This is due to the variable behavior of the different network cards on the servers. As expected, the test does disrupt the network.

3.3.1. Port Status on Switches

Cisco switches have multiple commands to display the port status of an interface. Also, port status logs display to the console with a timestamp, which is what this research used. The "shutdown" command at the interface configuration mode brings down the port, while the "no shutdown" commands brings it back up. Changing the port status can be automated using the Embedded Event Manager (EEM).

```
event manager applet shut-interface
event none
action 001 cli command "enable"
action 002 cli command "config t"
action 003 cli command "int g1/0/22"
action 004 cli command "shut"
action 005 wait 10
action 006 cli command "no shut"
action 007 cli command "end"
end
```

Figure 14. EEM Commands to Shutdown Interface for 10 Seconds

```
#event manager run shut-interface

 16:29:10.005: %LINK-5-CHANGED: Interface GigabitEthernet1/0/22, changed state to administratively down
 16:29:11.006: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/22, changed state to down
#
 16:29:18.040: %SYS-5-CONFIG_I: Configured from console by  on vty1 (EEM:shut-interface)
#
 16:29:20.025: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/22, changed state to down
#
 16:29:22.652: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/22, changed state to up
#
 16:29:23.652: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/22, changed state to up
```

Figure 15. Example Output of EEM Script

3.3.2. Port Status on Servers

Changing the port status on the servers is the same for all network cards. The command "ip link set dev <interface> down" was used. After changing the port status, each network card's behavior varied. The Intel card had the expected behavior, while the Broadcom card exhibited the down/up behavior. Switch3 shows the interface for Server2

going down but then up within four seconds. The OS shows the link as down during this time, but the Broadcom network card's lights are on. Bringing the link back up using the ip command caused similar down/up behavior

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/11, changed state to down
%LINK-3-UPDOWN: Interface GigabitEthernet1/0/11, changed state to down
%LINK-3-UPDOWN: Interface GigabitEthernet1/0/11, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/11, changed state to up
```

Figure 16. Down/Up Behavior of the onboard Broadcom network card

In an attempt to change the down/up behavior, both PXEBoot and Dell's version of IPMI (iDRAC) were disabled. Wake on Land was already disabled. The behavior persisted, likely due to the server's embedded management system called Lifecycle. Lifecycle is a robust embedded system that works below the operating system with many options that may take control of the network card. The Intel X550 network card did not exhibit the down/up behavior, but it was an additional card added later and not managed by Lifecycle.

The port status can be viewed manually with the ip or ethtool command. The ability to use logs to validate the timing of the port status varied and took more work compared to the switches. Surprisingly, logs to track the link going down were obscure or nonexistent by default. The Intel X550 provided one log entry "removed PHC" when the link went down, while the Broadcom NetXtreme provided none.

```
Aug 15 11:03:59 server1 kernel: ixgbe 0000:04:00.0: removed PHC on enp4s0f0
Aug 15 11:04:10 server1 kernel: ixgbe 0000:04:00.0: registered PHC device on enp4s0f0
Aug 15 11:04:13 server1 kernel: ixgbe 0000:04:00.0 enp4s0f0: NIC Link is Up 1 Gbps, Flow Control: None
```

Figure 17. Intel X550 logs for changing port status

This issue is specific to using a command on the server to bring down the link. Unplugging the cable or shutting down the port on the switch side does generate a log entry. In contrast, the link coming up always generated log entries.

3.3.3. Port Status Switch Detection

Changing the port status can detect rogue switches in all instances if the behavior of the specific interface is understood. For the port status test, the baseline values can be retrieved prior to the test even if there is a rogue switch. This test measures the change

state for a specific length of time, compared to the other test that only measure the final test state to a baseline.

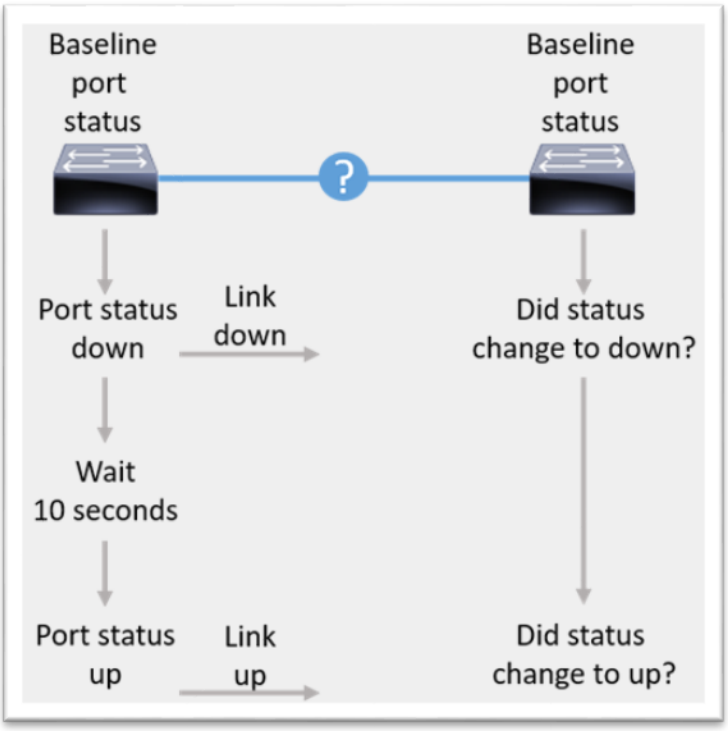

Figure 18. Port Status Test Procedure

In all cases, the rogue switch kept the remote device interface up. When Server2 is the device changing the port status, more is needed to check the end state of the port on Switch3 since this would give a false positive. In this case, the port down/up behavior must be factored into the timing of the test.

Expected Link Partners		Remote Device Status		
Local Device	**Remote Device**	**No rogue**	**Switch1**	**Unmanaged**
Switch2	Switch3	Down	Up	Up
Switch2	Server1	Down	Up	Up
Server1	Switch2	Down	Up	Up
Server2	Switch3	Down/Up	Up	Up
Switch3	Switch2	Down	Up	Up
Switch3	Server2	Down	Up	Up

Table 11. Port Status Test Values

False negatives are possible since multiple things can cause a link to go down. The possibility of this depends greatly on the network. This test relies on the remote device interface staying up if a rogue switch is present. That is, no logs are generated on the remote device. Conversely, when there is no rogue switch, there are always logs when the interface comes back up, and there may be logs of the interface going down.

4. Recommendations and Implications

The network purpose needs to be considered when evaluating the techniques described in this paper. These techniques only identify the issue after the fact and do not provide any protection from the rogue switch. Encryption and network access controls (NAC) can protect the network even when rogue devices are present. While NAC cannot protect against all rogue switches, it can protect against many use cases for rogue switches. Many switches do generate packets, or the attached devices do. Encryption can diminish the usefulness of a rogue switch used for network sniffing by protecting sensitive or proprietary data traveling over the network.

Networks that most benefit from Layer 1 techniques are ones subject to physical espionage or where the physical architecture is critical to the purpose of the network. For many data centers and ISPs, the network architecture directly affects their ability to meet SLA guarantees, recover quickly from outages, and troubleshoot problems quickly. Undocumented changes to these networks can significantly affect its performance and resilience. Additional networks sensitive to physical espionage include ICS/SCADA, defense networks, and other sensitive government networks. Specific commercial or public sector networks may also be targeted.

4.1. Rogue Switch Detection

TDR is the best technique for discovering physical network topology and integrity of wiring. TDR can detect rogue switches but has the highest possibility of false negatives. It is the only technique where baseline values must be taken during a known

network state without rogue devices. TDR's strength is troubleshooting hardware failures and detecting physical network changes beyond rogue switches. TDR may disrupt the network while running but will auto-recover to a working state after completion of the test. The accuracy of cable lengths varies among vendors and even product lines, but accuracy within 1 meter is possible. For data centers requiring all customers to have the same length cable, it may be worth investigating using a product with accurate TDR readings to verify no cable changes.

Automating TDR on the Cisco switches can be done through the simple network management protocol (SNMP). The MIB OID 1.3.6.1.4.1.9.9.390 (CISCO-CABLE-DIAG-MIB) manages the ability to execute a TDR test and view the results. Automating from the terminal would be more complex since the test command takes a variable amount of time to execute and is separate from the command to view results. The Intel X550 is also easy to automate, despite the need to access the physical registers directly, since the Intel card calculates the TDR length and pair status values. The Intel X550 driver supporting the tdr option for ethtool would have made automation even more straightforward.

Manipulating auto-negotiation values to detect rogue switches provided the most accurate and precise results. False readings are unlikely and only possible if the network happens to change during the test. This technique also works on networks without previous baseline values. The baseline can be captured before changing the auto-negotiation values, even if a rogue switch exists. Changing the auto-negotiation will cause a network disruption but will quickly auto-recover to a working state.

Automating is possible for all devices tested, but implementation will vary widely. Changing the auto-negotiation values themselves is standardized per operating system. Reading the link partner's advertised modes varies in how the network device provides the information. For example, the Cisco and Intel products only provided this information via the PHY registers. This test must also consider what modes each device can send and receive. For example, the Intel X550 does not support 10Mb or any half-duplex modes. However, it can track the link partner's ability to support half-duplex for 100Mb and 1Gb. It cannot track any 10Mb modes from the link partner. The Cisco

switches and Broadcom NetXtreme support 10Mb to 1Gb in both half and full duplex, but they cannot see any higher bandwidth advertised by the link partner.

Manipulating the port status values does detect rogue switches, but the variability of behavior among devices makes this more challenging to verify. False readings are possible because the network connection can go down for various reasons unrelated to the test. This technique does work on networks without previous baseline values because it measures status change over a specific time. This is the only technique that requires intervention to get the link back to a working state.

Automation of the port status change is most straightforward from the switch side. The logging behavior is consistent on both the switch and end device when the port status is changed on the switch. Conversely, logging behavior varied when bringing down the interface on the server side. Additionally, the network interface for servers may not stay down when using embedded management systems such as IPMI. This may be why most current research only uses the port status technique on the switch side. Automating this method requires accurate timestamps for both link partner devices, so a time server is needed. Since SNMP is known to have timestamp accuracy issues, syslog is the better option for tracking port status (Roughan, 2010).

Changing port status can be used for manual troubleshooting by IT support. In this case, the person verifies the port status in real-time on each device, so timestamps are unnecessary. Instead, commands with network status outputs can be used to verify the interface's status.

Once identified, further verification may be needed to find the rogue switch if the location is not immediately apparent. In this case, the cable must be physically traced to the rogue switch location. A tone generator and probe tracer help with this task. The tone generator is placed on one end of the cable, and the probe tracer can follow the tone by moving along the outside of the cable. A shielded cable will be more difficult to trace since the shielding blocks the tone. A high-powered tone generator can overcome the shielding issue.

4.2. Mapping Physical Network

The TDR pair status can be used to map the physical network connections. The pair status detects a remote device on the line even if that remote device is off. This physical link audit can be used to update network diagrams and detect unknown devices. As mentioned before, significant outages, disruptive events, or employee turnover can cause large, undocumented changes to the network. Using TDR to audit the network can help bring it back into compliance and properly documented.

4.3. Hardware Supply Chain Verification

Each device's unique behavior and available registers can be used to verify hardware authenticity. The network card can be verified on the local device by comparing the available registers and their behavior to the network card's datasheet. Differences between the datasheet and the network card's registers could signal a compromised supply chain. The remote device can also be enumerated based on advertised link partner registers and the Cisco switches' TDR results. In this case, a baseline of expected values for approved hardware is created, and any outliers signal a possible unapproved device is connected. This would be easiest to implement in a network with limited vendors and device types to track.

4.4. Future Research

Further research for TDR includes tests on higher speeds, longer cables, OTDR for fiber, and behavior on other operating systems and vendor products. Even with the small set of devices in this research, the amount of variable behavior shows the potential for further research by testing a wider variety of vendors and devices.

Further exploration can be done with the network card registers, especially Clause 45 devices and vendor-specific registers. For example, the Intel X550 has registers to record the link partner's firmware version, and the Broadcom NetXtreme can detect if the link partner is an IP phone. The PHY ID registers common to Clause 22 and 45 can be used for local hardware supply chain verification. Further research could even identify passive detection possibilities.

This research did not consider detecting Layer 1 and Layer 2 taps. Previous research by Petrenko shows Layer 2 taps can set both sides to the same speed and duplex, but it is unknown if they can send consistent auto-negotiation link partner data to each side of the tap. Additionally, port status does not detect Layer 1 or 2 taps, and TDR may detect taps by the change in impedance on the wire.

5. Conclusion

All Layer 1 techniques can detect rogue switches and even other network hardware changes. The auto-negotiation technique is currently the most accessible to implement across devices. The TDR technique provides multiple uses, but server-side driver support still needs to be improved. Overall, device variability is the biggest hurdle in implementation and automation. A standardized method for implementing and gathering test information would help overcome current impediments. For Linux, the ethtool command has potential as it already provides options for TDR and auto-negotiation. The network card driver simply needs to implement the ethtool features. Universal implementation of these ethtool options would benefit administrators for troubleshooting and automating these techniques. A standardized network diagnostic SNMP mib or network device API can gather information across switch brands. Each technique is capable of being scripted with current technology and integrated into network monitoring software.

References

Lunn, Andrew. (2019, September). *Ethernet Cable Diagnostic using Netlink Ethtool API* [Conference presentation]. Linux Plumbers Conference 2019, Lisbon, Portugal. Retrieved January 12, 2023, from https://lpc.events/event/4/contributions/436/

IEEE Standard for Ethernet. (2022). *IEEE Std 802. 3-2022 (Revision of IEEE Std 802. 3-2018)*, 1–7025. doi:10.1109/IEEESTD.2022.9844436

Kiravuo, T., Sarela, M., & Manner, J. (2013). A Survey of Ethernet LAN Security. *IEEE Communications Surveys & Tutorials*, 15(3), 1477–1491. doi:10.1109/SURV.2012.121112.00190

Bhuse, V., Kalafut, A., & Dohn, L. (2019). Detection of a Rogue Switch in a Local Area Network. *ICIMP 2019:* T*he Fourteenth International Conference on Internet Monitoring and Protection*, 14-19.

Quitiqut, T., & Bhuse, V. (2022). Utilizing Switch Port Link State to Detect Rogue Switches. *International Conference on Cyber Warfare and Security, 17*, 272–278.

Alcock, S., McGregor, A., & Nelson, R. (2005). Using simple per-hop capacity metrics to discover link layer network topology. *Passive and Active Network Measurement: 6th International Workshop, PAM 2005, Boston, MA, USA, March 31-April 1, 2005. Proceedings 6*, 163–176. Springer. doi:10.1007/978-3-540-31966-5_13

James, V. (2019). Network Automation Methodology for detecting Rogue Switch. *Tech. Libr*, 337. Retrieved January 12 , 2023, from https://core.ac.uk/download/pdf/220125618.pdf

Petrenko, A. (2015). Detecting physical layer attacks on Ethernet networks. Retrieved February 3, 2023, from https://www.theseus.fi/bitstream/handle/10024/98363/Petrenko_Alexey.pdf

NERC. (2018, October 2). Lesson Learned Networking Packet Broadcast Storms. Retrieved February 14, 2023, from https://www.nerc.com/pa/rrm/ea/Lessons%20Learned%20Document%20Library/LL20181001_Networking_Packet_Broadcast_Storms.pdf

Dove, D. (2008). *Apparatus & method for automatically switching media connections when operating in forced speed and duplex* (U.S. Patent No. 7,366,771 B2). U.S. Patent and Trademark Office. https://patents.google.com/patent/US7366771B2/en

Roughan, M. (2010). A case study of the accuracy of SNMP measurements. *Journal of Electrical and Computer Engineering*, Article ID 812979, 1–7. https://doi.org/10.1155/2010/812979

Intel. (2023). *Intel Ethernet Controller X550 Datasheet*. https://www.intel.com/content/www/us/en/content-details/333369/intel-ethernet-controller-x550-datasheet.html

Broadcom. (2022). *NetXtreme/NetLink BCM571X/BCM5720 Family Programmer's Guide*. https://docs.broadcom.com/doc/571X-5720-PG1XX

Kopper, A., & Westner, M. (2016). *Towards a taxonomy for shadow IT*. ResearchGate. https://www.researchgate.net/publication/303769485_Towards_a_Taxonomy_for_Shadow_IT

Appendix A
Lab Details

Phytool was download from github.com/wkz/phytool. The ixgbe driver and NVMUpdate were downloaded from Intel. Phytool did not work until the updated driver and NVMUpdate were applied and the system rebooted. The tg3 driver and ethtool versions are the default for RHEL 8.7.

Hardware	Software
Cisco 3850-24T switch x2	IOS XE version 16.12.07
Cisco 3750X-48 switch	IOS Version 15.2(4) ES
Reidubo JHJ-DOS-K switch	-
Dell R530 server x2	RHEL 8.7, kernel 4.18.0-425.3.1 Ethtool 5.13 Phytool v2 ixgbe driver 5.18.13-1 tg3 driver 4.18.0-425.3.1 NVMUpdate v3.60
Intel X550T2 network card x2	-
NetAlly Linkrunner AT 2000	-

Cable length values are based on the cable tester results except for the 1m cables which were measured with a measuring tape. The cable length values from the tester were "<1" as 1m is the minimum length it can measure.

Cables					
Name	Length	CAT	Type	Gauge	Other
1mA	0.9	5e	UTP	24	-
1mB	0.9	5e	UTP	24	-
1mC	0.9	6	UTP	24	Stranded
3mA	3.2	5e	FTP	26	-
3mB	3.2	5e	STP	26	-
3mC	3.2	6	UTP	24	-
8m	8	5e	FTP	26	Screened
15m	15.3	6	UTP	24	-

Appendix B
TDR Detail Values

The Server1 column without the Intel X550 is the onboard Broadcom network card similar to Server2. This network card on Server1 was not used in the testing of rogue switches. For all switches, port 24 was used for cable length tests.

Switch TDR Values Plugged Into Link Partner						
Switch1						
Cable	Switch2	Switch3	Unmanaged	Server1	Server2	Server1 (Intel X550)
1m A	0	0	0	0	0	1
1m B	0	0	0	0	0	1
1m C	0	0	0	0	0	1
3m A	0	0	1	0	0	4
3m B	0	0	0	0	0	4
3m C	0	0	0	0	0	4
8m	1	1	1	1	1	11
15m	4	4	7	4	4	15
Switch2						
Cable	Switch1	Switch3	Unmanaged	Server1	Server2	Server1 (Intel X550)
1m A	15	7	8	7	7	16
1m B	15	8	10	7	7	16
1m C	15	7	8	7	7	16
3m A	19	12	14	11	12	20
3m B	18	10	11	10	10	19
3m C	16	10	11	10	10	19
8m	23	15	18	15	15	25
15m	27	20	21	19	19	29
Switch3						
Cable	Switch1	Switch2	Unmanaged	Server1	Server2	Server1 (Intel X550)
1m A	16	11	11	8	8	19
1m B	16	11	11	8	8	19
1m C	16	10	10	8	8	18
3m A	20	15	15	14	14	24
3m B	19	23	24	23	23	20
3m C	19	12	12	11	11	21
8m	25	19	19	16	18	27
15m	29	23	24	23	23	33

Intel X550 TDR Values Plugged Into Link Partner				
Server1 (Intel X550)				
Cable	Switch1	Switch2	Switch3	Unmanaged
1m A	1	1	1	1
1m B	1	1	1	1
1m C	1	1	1	1
3m A	3	3	3	3
3m B	3	3	3	3
3m C	3	3	3	3
8m	8	8	8	8
15m	15	15	16	15

While all TDR length values are per pair of wires, only the unplugged cables showed variability of the pair lengths up to two meters. The chart below gives detailed pair length and average length of cables unplugged from the remote device.

Switch1 Unplugged Cable					
Cable	Pair 1	Pair 2	Pair 3	Pair 4	Average
1m A	1	2	0	2	1
1m B	0	2	1	0	1
1m C	0	2	0	2	1
3m A	3	4	3	2	3
3m B	2	3	2	2	2
3m C	2	4	3	2	3
8m	8	8	8	9	8
15m	15	16	15	15	15

Switch2 Unplugged Cable					
Cable	Pair 1	Pair 2	Pair 3	Pair 4	Average
1m A	0	1	0	1	1
1m B	0	0	0	0	0
1m C	0	1	0	0	0
3m A	3	3	3	3	3
3m B	3	3	2	3	3
3m C	2	3	2	3	3
8m	8	8	8	8	8
15m	16	16	16	16	16

Switch3 Unplugged Cable					
Cable	Pair 1	Pair 2	Pair 3	Pair 4	Average
1m A	0	0	0	0	0
1m B	0	0	0	0	0
1m C	1	1	0	0	1
3m A	2	3	3	3	3
3m B	3	3	2	3	3
3m C	2	3	2	3	3
8m	7	8	7	8	8
15m	16	16	15	15	16

Server 1 (Intel X550) Unplugged Cable					
Cable	Pair 1	Pair 2	Pair 3	Pair 4	Average
1m A	1	1	1	1	1
1m B	1	1	1	1	1
1m C	1	1	1	1	1
3m A	3	3	3	3	3
3m B	3	3	3	3	3
3mC	3	3	3	3	3
8m	8	8	8	8	8
15m	15	15	15	15	15