Introduction

In this lab, a .pcap file was provided in which the tools Wireshark, Snort, and NetworkMiner can be used to perform intrusion detection on a LAN. Along with this two different events logs are provided in which event viewers and scripts can be run to analyze the files.

Objective

The objective of this lab is to detect any sort of intrusion in accordance with the basic intrusion detection framework on the local network depicted in the packet capture file, along with identifying the type and extent of the attack with TTP information included.

Report

For this lab it is wise to start analysis with a general view of the system, much like the general practice of intrusion detection and investigation, which in this case the system is presented to us in a packet capture file. A general idea of traffic can best be surmised by analyzing basic capture statistics, such as session length and number of packets sent in this time. Provided in figure 1 is this basic information as presented by the statistics panel in Wireshark. The elapsed time of the capture is about four and a quarter minutes, and with a total of 2364 packets, meaning that about 9 packets are sent per second on average. This infers a moderate internal network, supported by the fact that the total amount of bytes is 1921245, which when divided by the time the capture file was taken over comes to about 7623, and further comes to 813 when divided by 9.3, where 813 is average packet size in bytes (*Ethernet statistics*). With the OS of the host of the capture being Win7SP1 and with the fact that there are few endpoints in the capture, we see that this rate of packets is abnormal given what is known, so DDoSing or some other attack that uses rapid segment transfer is a first point of investigation. As seen in figure 1, this information takes averages over all types of packets, so to get an even clearer image of normal and expected traffic for this network, we must investigate the types of protocols shown in the packet capture.

Time

First packet:	2017-07-08 14:09:32
Last packet:	2017-07-08 14:13:45
Elapsed:	00:04:12

Capture

Hardware:	Unknown
OS:	64-bit Windows 7 Service Pack 1, build 7601
Application:	Dumpcap 1.10.3 (SVN Rev 53022 from /trunk-1.10)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
\Device\NPF_{AA48AEC9-CEF5-43EC-B3F1-25D52CAE7321}	Unknown	none	Ethernet	65535 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	2364	2364 (100.0%)	—
Time span, s	252.934	252.934	—
Average pps	9.3	9.3	—
Average packet size, B	813	813	—
Bytes	1921245	1921245 (100.0%)	0
Average bytes/s	7595	7595	—
Average bits/s	60 k	60 k	—

Figure 1 – Capture file statistics

Wireshark allows sorting by protocol, so when analyzing the network the presence of certain protocols informs the structure of the network. The presence of ARP, DNS, LLMNR, TCP, UDP, DHCP, NBNS, and BROWSER protocols infers web connections through a router and the presence of a networked host (*Types of network protocols explained with functions*). What is interesting is that DNS, LLMNR, and NBNS in essence perform the same functions, but a device might use one out of the three depending on fallback in resolving hostnames, telling us that host resolution is a main priority in the network configurations (Miller, 2021). ESP infers a tunnel or VPN connection, ICMP infers pinging commands and network device management, IGMP infers multicasting, showing that the network requires outside commands and remote means of managing and connecting to the network (*What is IGMP? | internet group management protocol | cloudflare*). DCERPC, SPOOLSS, SRVSVS, SMB infers access to shared files and printer devices on the network with the capability for remote command and code execution (Goikhman, *SMB::DCERPC*). SSDP allows for the discovery of new devices, and in conjunction with the remote execution protocols, printer device protocols, and network management protocols, we see that this network has automated processes for adding and using printers and other such devices across the network along with remote management of the network and devices (*SSDP, Wireshark*). There is one more protocol listed, being RK512, however all 3 of these packets are malformed in some way, as seen in figure 2. Even so, RK512 is a protocol that is used to communicate with PLC automats of Seimans Sematic, and with a combination of old legacy and modern protocols, so with just the information found just from the presence of these protocols, an IT network that is directly related to an OT network dealing in some way with printers (*Flexi soft – RK512 - sick germany*). However, to gain context as to the protocols' use, the overall connections and dynamic of the network depicted in the capture must be investigated.

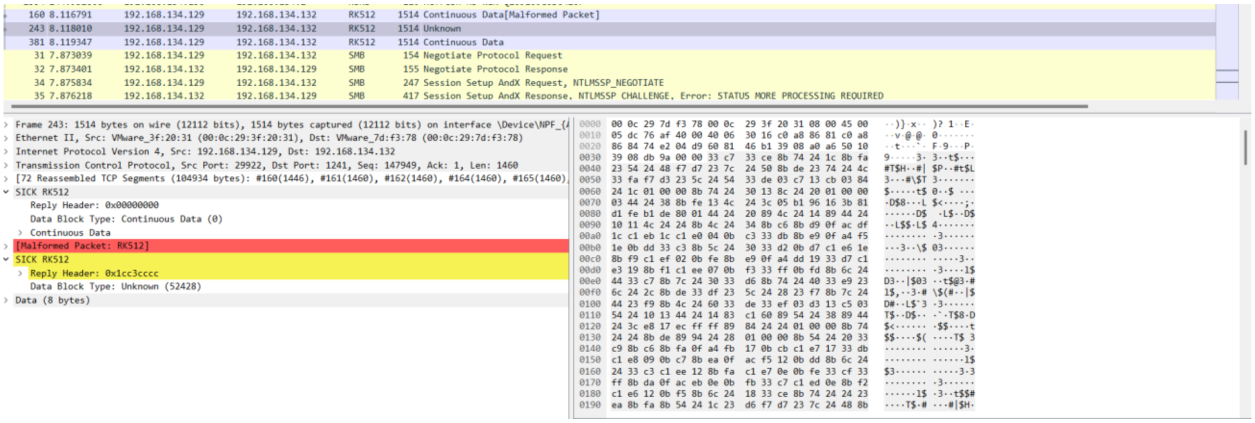
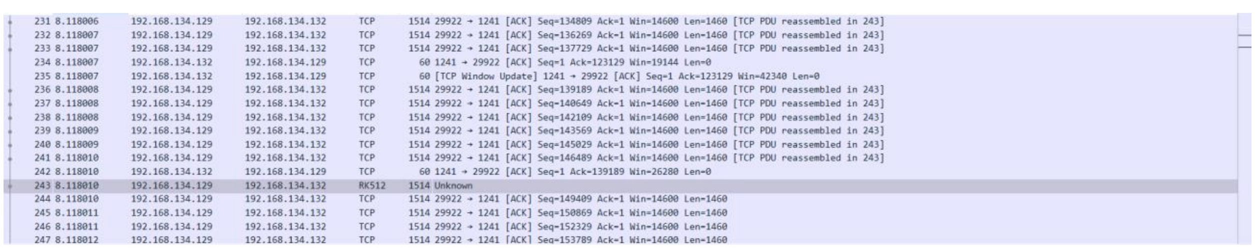


Figure 2: malformed RK512 packet

Looking at the I/O graph on Wireshark gives the first indicator that there is anomalous traffic. There is a large spike of TCP traffic at second 8 as seen in figure 3, where many TCP segments are sent with the max size of an ethernet frame, being 1514 bytes. The vast majority are Ack statements sent form the IP 192.168.134.129 to IP 192.168.134.132 with the destination port designation of 1241, meaning that the .129 host is communicating over Nessus to the .132 host with many full tcp packets, possibly scanning for vulnerabilities as is the function of Nessus. As Nessus sends tons of ACK packets with payloads (sending scripts, requests, etc.) where it tries to overwhelm the server with small, rapid probes to enumerate vulnerabilities quickly, we can further prove that Nessus is being used to scan for vulnerabilities on the .132 machine (Mallick, 2025). In the midst of these packet spikes are the rk512 packets, but seeing that, as seen in figure 3, the message that the PDUs were assembled in packet 243, we find that one of the rk512 packets being 243 is likely the cause of errors in the reassembly of the tcp flags, where Wireshark is unable to identify specifics of the stream and relays an error for continuous data. This is the case for all three packets, so it is likely that there is no OT factor on the network, or at least, this aspect is the cause of the scanning we see in the traffic spikes in the packet capture.



There are three main spikes depicted in the graph, and the second spike follows the same rule of thumb and the first discussed earlier, but the last spike, as seen in figure 4, relates to a series of RST flags. This section involves the .129 host sending a syn flag, and the .132 host rejecting and resetting the connection, and with each pair of packets the port numbers increase. This is also indicative of portscanning, as the .129 host is searching for over various ports to establish a connection with SYN, but the .132 host rejects the connections. While in terms of time this takes place about 4 minutes after the first and largest spike, this is not unusual in in-depth scans, as Nessus allows for comprehensive searches that use multiple tools. Right before this spike was a series of ICMP requests and responses, so one the ping phase of the Nessus scan was completed the program tried to directly connect over common ports that are left open such as port 21 and port 80, which is FTP and HTTP respectively. It seems that before the program sought vulnerabilities, but now it is seeking open ports that the host may be able to be exploited by.

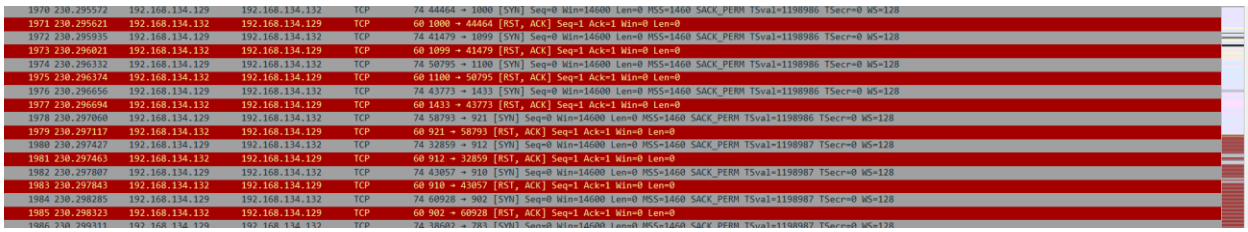


Figure 4: Third traffic spike

When providing the capture file to snort, many alerts are generated, and as seen in figure 5 all the errors relate to small rapid TCP segments with 1 alert being for a suspected portscan. This correlates the suspicions found in Wireshark, where, as seen in figure 6, we see that the .129 host is trying to log into various SMB services through default credentials and requests. What is noteworthy is that there are some alerts generated by the .132 host communicating through SMB to the .129 host, and this means that the .132 host is actively responding to the .129 host, enough to generate alerts, showing that the scanning efforts are returning results for the client. The DCERPC, SPOOLSS, and SRVSVS protocols are also part of the request and response pattern to the vulnerability scanning at the beginning of the packet capture, and as they appear first, it is likely that the .132 machine is a printer and is known as such on the network, as otherwise Nessus would perform regular scans against the OS first (Mallick, 2025). Some of the SPOOLSS packets are malformed, meaning that there is likely some other supplementary program that sits between the .126 and .132 machine that allows for the crafting and sending packets during the Nessus scan (*What is the reason for malformed packet error ?*). Looking more into the printing protocols, as seen in figure 6, we see 15 rejections and 1 acceptance by DCERPC, which is alarming and would surely be desirable information to the user of the .129 host, but as Nessus once started runs to completion, there are no more developments or packets relating to a possible DCERPC connection. This acceptance comes after many login attempts and many full TCP packets, so some form of buffer overflow over DCERPC is likely to allow unauthorized acceptance into the SMB service and DCERPC will provide a c prompt as to its service of remote code execution over the \pipe\spoolss address. To better grasp the architecture as a whole, all the various hosts needs to be mapped as to their relation with the 192.168.134 subnet to see if there are printers that could explain this traffic and if the local network is a standard IT network, as depending on these answers we can figure that most of the traffic and anomalies are a result of the vulnerability scanning.

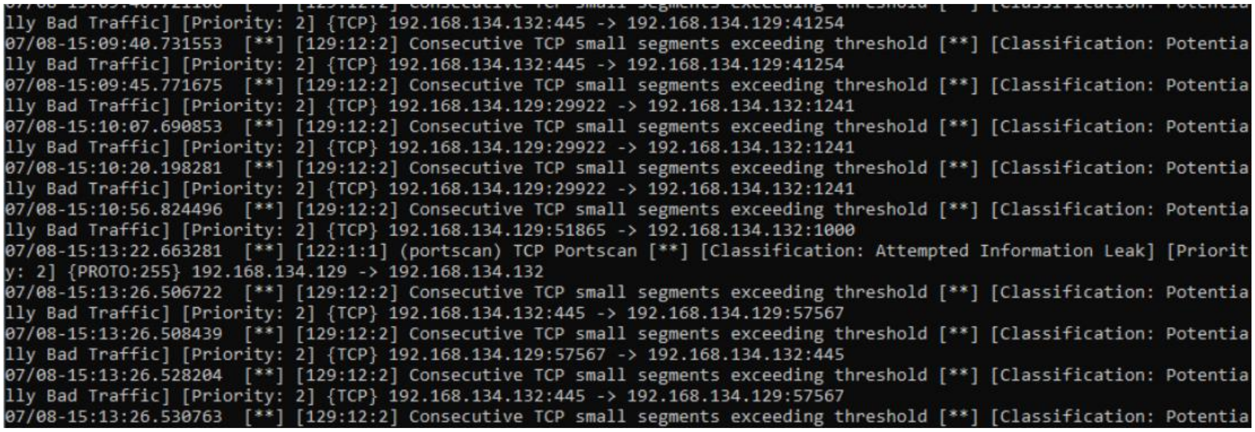


Figure 5: Snort alerts

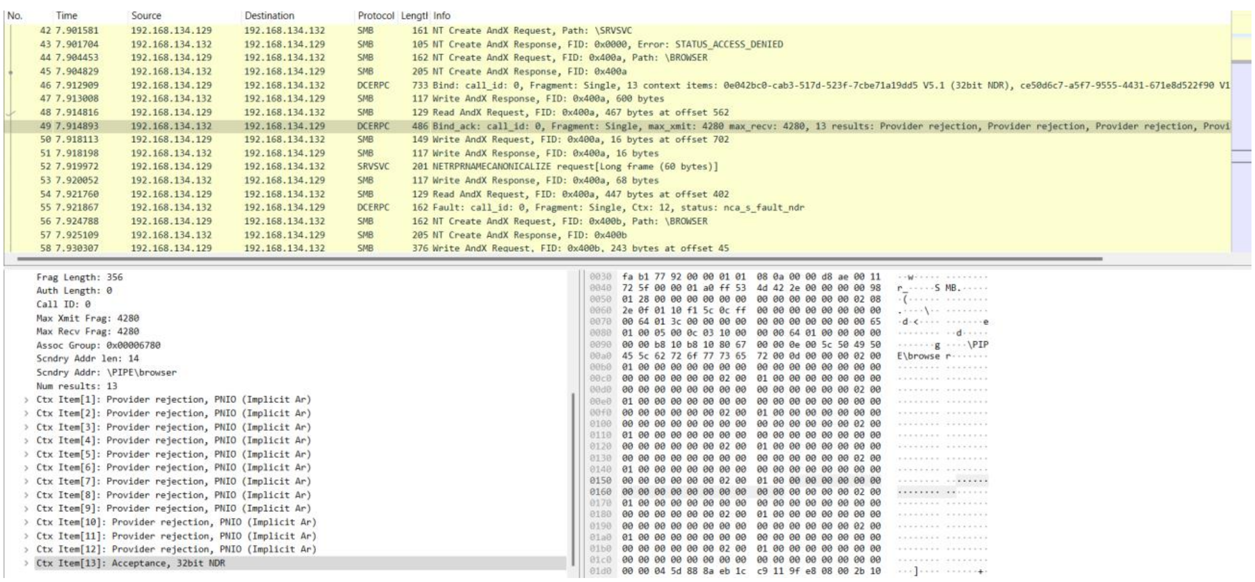


Figure 6: SMB service scanning and login attempts

To find out more about the hosts, we can use Network Miner, which can also help identify the structure of the network such as the router and the related servers. As seen in figure 7, we see that the .129 machine is a Linux machine with Ettercap and that the .132 host is a windows machine and a printer, explaining the presence of the SMB printer services, and that there are 3 other devices on the subnet. The presence of Ettercap on the machine explains how the attacker might have been crafting malformed packets as Ettercap allows packet sniffing and the changing of traffic, so this was likely another tool alongside Nessus that allowed for sniffing all hosted on the Ubuntu OS. Looking at the other devices, as seen in figure 8, the .1 host is a workstation, a server, an NT workstation, and is available to become a browser if needed, showing that the printer device has services and capabilities beyond printing provided by the

Microsoft OS (*Browserprotocol*). The .2 device is a DNS/NBNS resolver and router, and the .158 is another workstation involved in IGMP traffic. One reason we must look outside the two hosts investigated earlier is that we see .2 pinging .158 generating a destination unreachable alert, which warrants further investigation to the other hosts on the network. The .2 device also has the same message returned when communicating with the .132 host, and all of this is likely due to the amount of traffic on the network exceeding normal operational levels, to because the .132 is being communicated with the rapid segments, it is likely designated as unreachable as it cannot respond to the ping, showing that the attack can be seen from indicators on the network as a whole. The .158 device is a workstation that likely utilizes the printer, but due to the scans it is also unable to receive traffic, but to find more information the .129 host should be investigated in how it first interacted with the network and how it utilizes the scans.

Figure 7: NetworkMiner host information

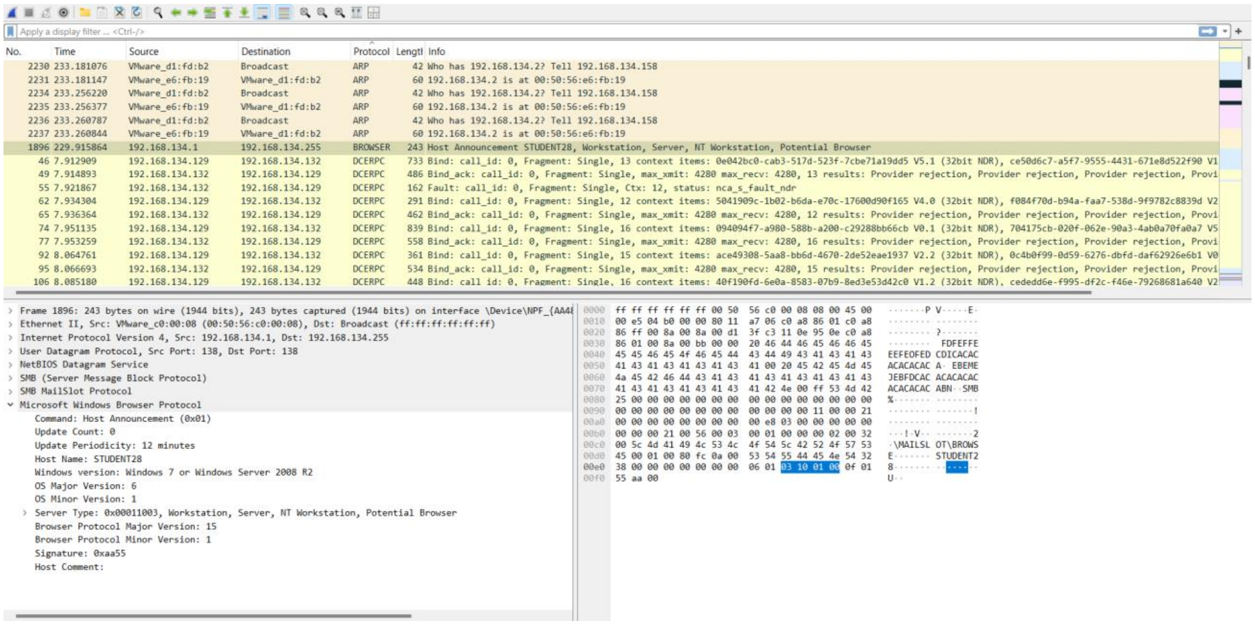


Figure 8: .132 BROWSER announcement

To better grasp the scanning efforts of the .129 host, we can look at the beginning and end of the scanning efforts to see how the host began communication and if the host exploited any vulnerability after the scans. We see that the connection started through a TCP handshake over port 445, or SMB. As far as what can be ascertained by the packet capture, the .129 host was already on the LAN and configured as a regular device. This gives the possibility that vulnerability scans were allowed and condoned by the system manager and owners of the devices, as vulnerability scans can be used for defense just as much as offence. The final sent packet by .129 is over the Nessus port, so with all the information gathered by the pcap, it cannot be ascertained whether the depicted vulnerability scan is being used for malicious purposes, so using the other two sources of information is paramount to investigating the context of the scan. As was found earlier DCERPC allowed a connection but was not followed up on, so it is possible an exploit or a patch was installed, but to figure which is the case the security and application logs provided can provide more insight.

Looking at the first log file being an application log in .evt format proves difficult, but using the date identified in snort we can start our search at the logs generated the day of the attack. When searching through the logs we find a few errors and warnings, but the dates do not match the snort alerts, so it is likely the logs are not synced with the NTP protocol (*NTP - how does it work?* 2022). Looking for SPOOLSS and printer keywords(which is done so because of the attempted enumerations being the first point of scanning) yields logs that the spooler service was stopped and started, and when we look at logs with the source "print", as seen in figure 9, we find that the message that "this event is not installed on your local computer or the installation is corrupted". Looking up the dlls listed in the log as seen in the figure, we see that the cause of this message is likely that the printer driver or print component is missing, corrupted, or mismatched on a ThinPrint machine (*TPPRN.DLL download and fix TPPRN.DLL missing error: Fileinspect*). Later in the logs we see multiple errors indicating a possible compromise, as many services including IPSec and NetBIOS were unable to start due to missing drivers and that the log right after the errors states that the event log service was stopped, as seen in figure 10. These security logs indicate that the printer may be compromised through spooler, and that many security functionalities were disabled by the attacker. To gain more information, looking at the application log that hosts the various SMB services is what is needed to follow up on the current suspicions.

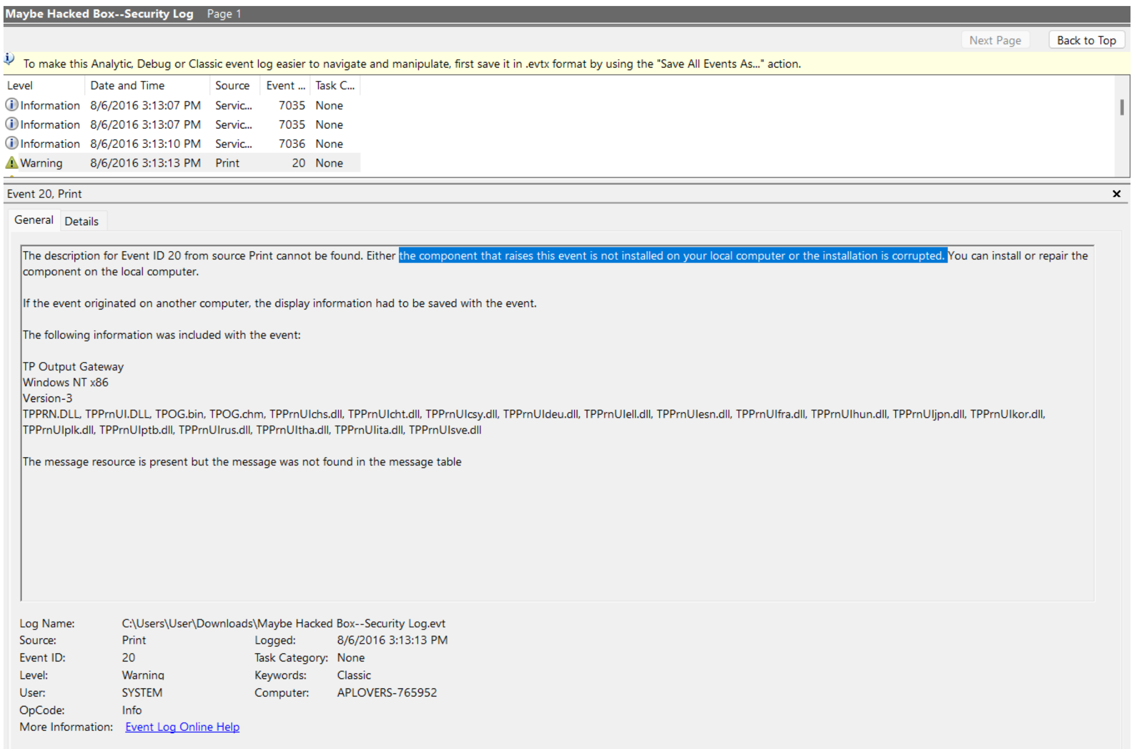


Figure 9: Warning log related to print service

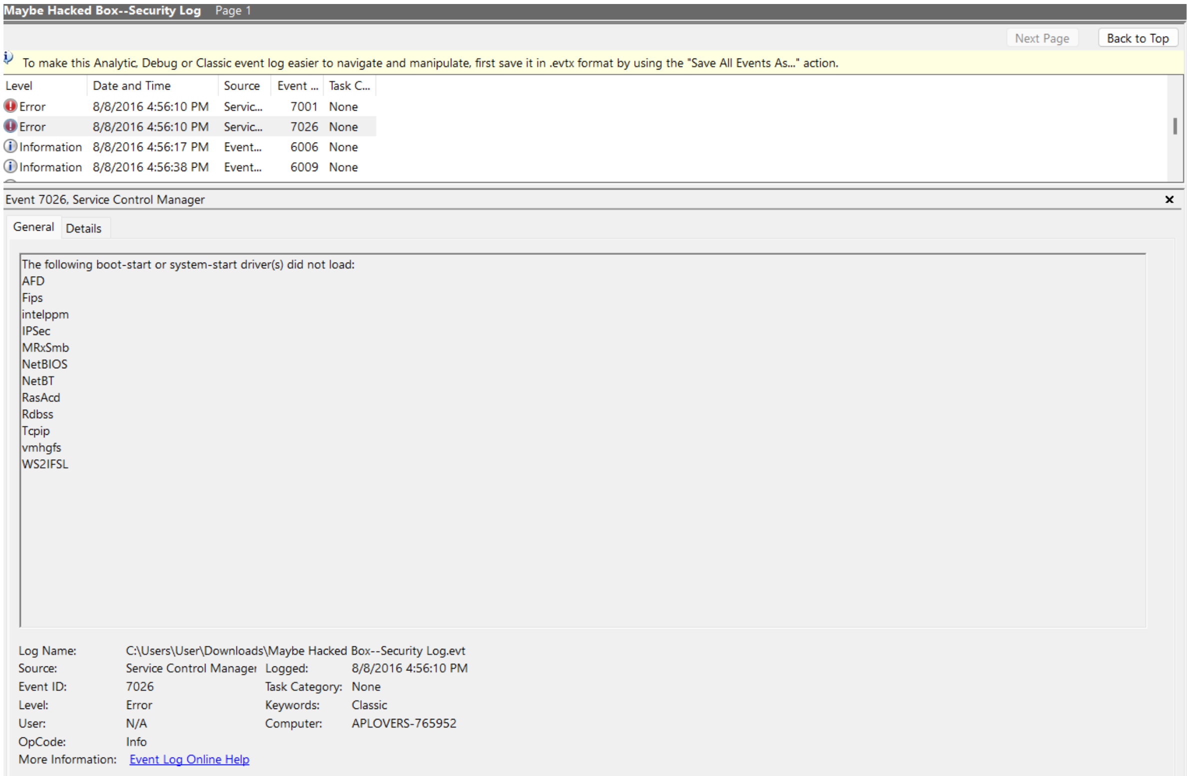


Figure 10: Security services error

In the application log, as seen in figure 11, we find a log when searching for the keyword print that relays information that ThinPrint registered a WMI provider (TPVCGProv) into WMI under Root\ThinPrint, and that either the provider is broken, uninstalled incorrectly, or corrupted (*WMI providers - win32 apps*). The reason for the generated warning is that when Windows tried to query something from that WMI namespace (maybe during boot, or printing), it threw an error (Decker, 2023). Considering that in the application logs these are the only warnings in the file and that the source is WinMgmt, we can figure that with this warning the printer' system monitoring tools will fail (like Performance Monitor and Task Scheduler) along with admin scripts using WMI and remote management via tools like PowerShell, SCCM, or WBEM (*WINMGMT - win32 apps*).The fact that all the logs relating to the printer services are incomplete or corrupted in some way in the event viewer and that they are warnings, it is likely that the printer is compromised on a kernel level, and that the errors generated in the security log are a result of this tampering. In the last few logs in the application file we see that WebFldrs XP has been set up and that the windows update client database was shut down by unexpected means in a dirty shutdown, so while not conclusive, it gives credence to the idea that the attacker is using the printer as a network driver for files and that the updates meant for the printer are not being installed through interruptions. This means that the attacker likely has deep control of the printer and that through programs and the prevention of updates that may remove access or these programs, the printer can be used to manage files on the network and allow lateral movement across the network.

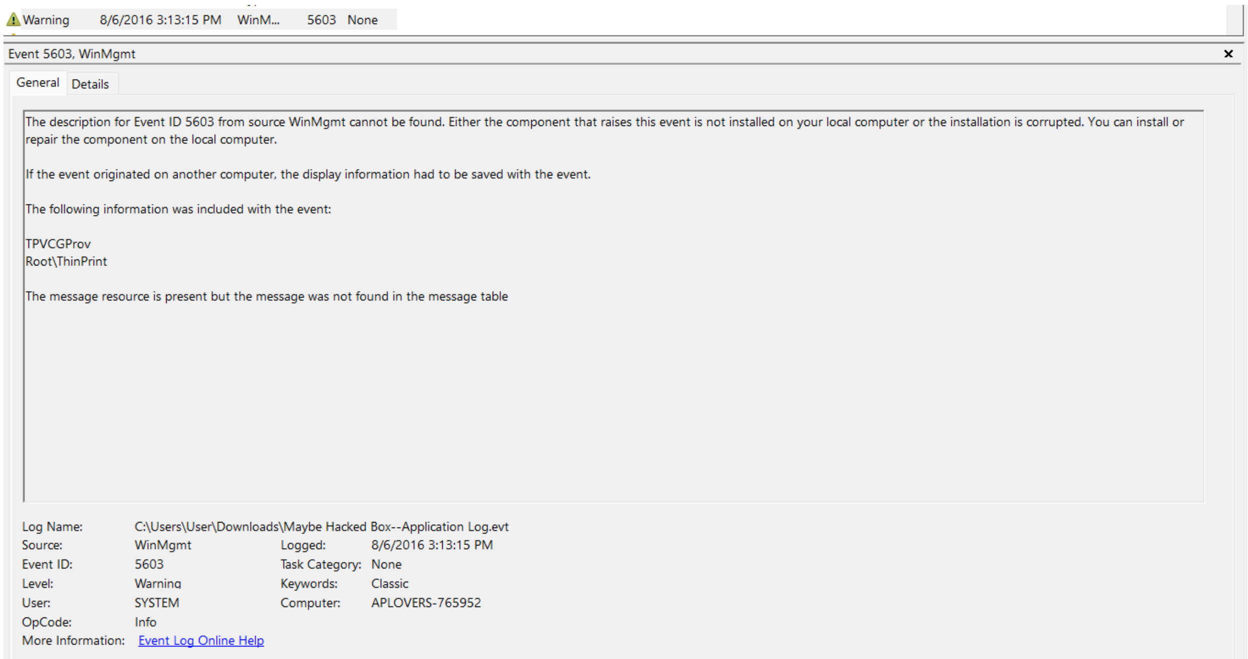


Figure 11: ThinPrint service warning

To conclude a story by summing up the found artifacts, we find the presence of scanning and enumeration by the .129 IP by looking at traffic spikes that include TCP segments relating to the Nessus vulnerability scanner program port and TCP segments relating to attempted connection to common ports. Other scanning efforts can be seen in the SPOOLSS requests and ICMP pings, and along with this we see a connection accepted message in a DCERPC message. Looking at the devices we see a Linux machine with Ettercap trying to connect to a printer running a windows OS and SMB services as to give printer functionality on the network. Looking at the host logs of APLOVER, the printer, we find in the security file that warnings were generated for rapid TCP segments, likely resulting in the RST flags we see in the PCAP, and that there are warnings for security services that were not able to start up. This gives evidence to compromise through the printer services and looking at the application log we see the setting up of WebFldr and the shutdown of the update client. With all of this, we see two stages of an attack, where first comes scanning and then enumeration. Printers are a host such as any other device on the network, and Ettercap and Nessus were able to find a way to exploit insecure printing protocol or configurations, so in this the attacker was able to move laterally through the network, using the printer's SMB and connection statuses to create a moving point for files ang himself throughout the network.

Looking at the tactics used by the attacker, we see the tactic T1595.002 being used being "Active Scanning: Vulnerability Scanning" as defined my MITRE (*Active scanning: Vulnerability scanning*). While not able to ascertain a tactic across all the columns of an attack, as seen in figure 12, we can make a diagram that highlights the key points of the adversary, where at which point in could be mapped to threat intelligence data. The most important aspects of this attack are the scanning and the defense avoidance, being T1562.001 or "Impair Defenses: Disable or Modify Tools", where, as seen in the security logs, the logging system was stopped along with TCP/IP and IPSec (*Impair defenses: Disable or modify tools*). This modeling is imperative to the investigative process and incorporates intelligence to both better system response and system defense. As seen in figure 13, we can also map out the attack according to the steps to a cyber-attack, where the portion of the attacker is supplanted by the analysis done with the MITRE framework in the tactics, techniques, and procedures mapping.

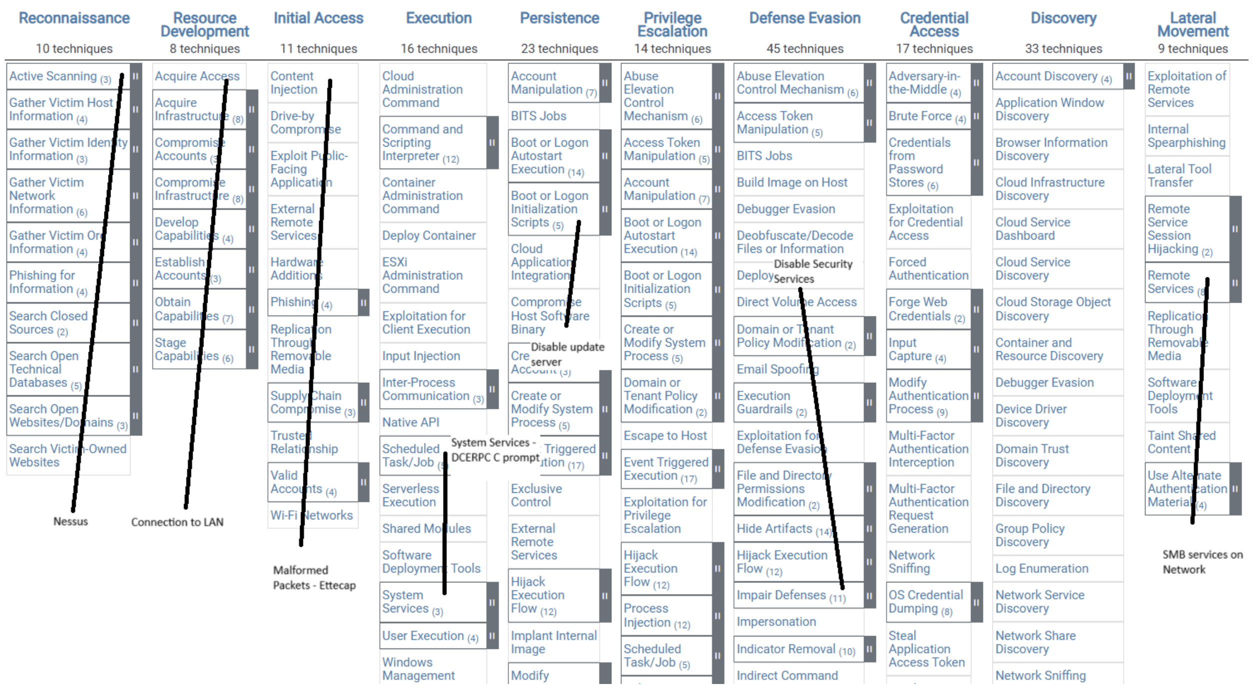


Figure 12 – MITRE attack techniques

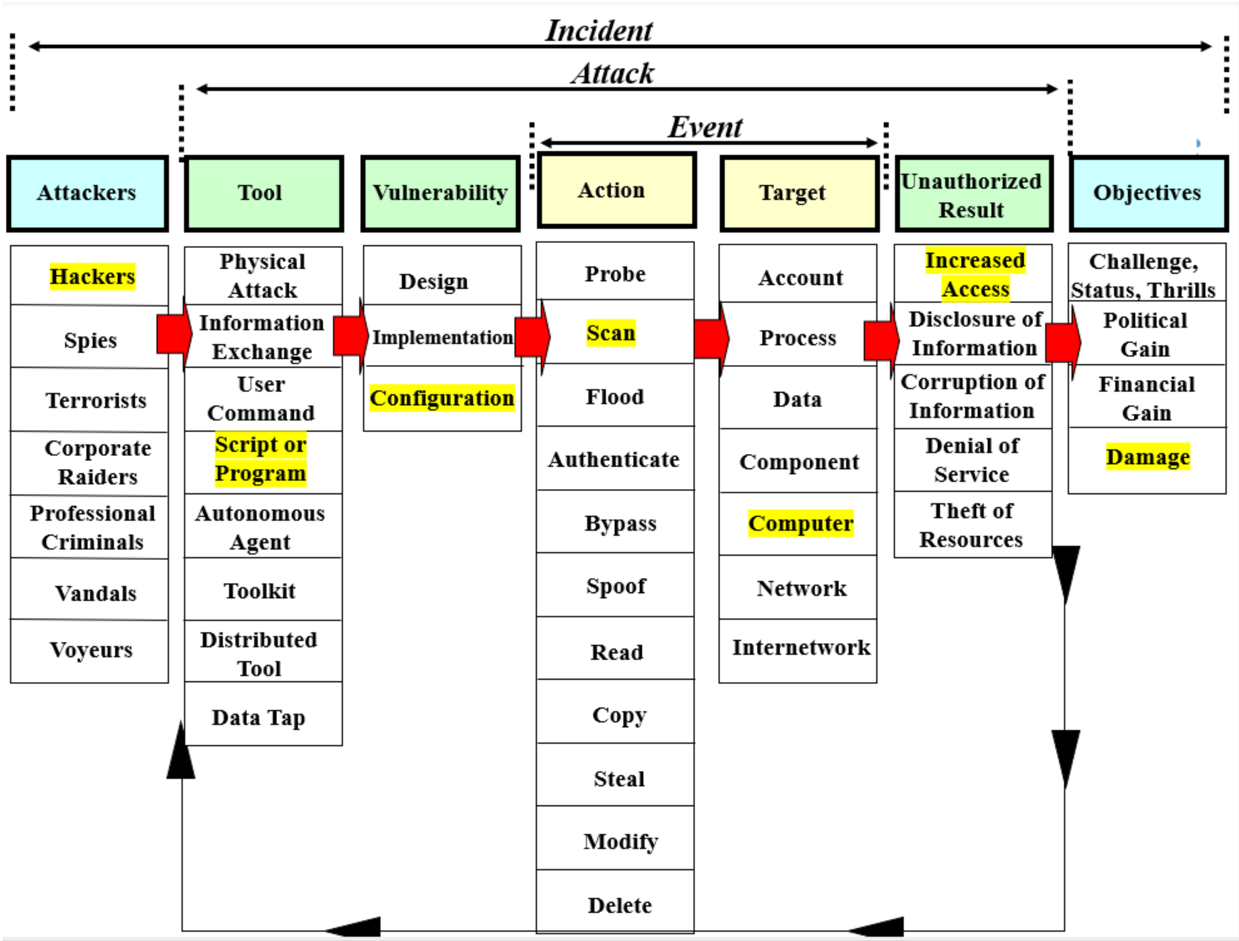


Figure 13 – Attack mapping

Conclusion

In this lab along with a Wireshark pcap was provided, event logs originating from the host system were subject to investigation, and understanding how to extract information form logs in this format is a cornerstone baseline for security analysts. Network logs are often read in combination, but because windows systems provide native logging solutions, many times these sources of information are always available, however this is known to attackers, where an often target and MITRE method is to disrupt and remove the logging process so that one may move undetected. Despite this, in this lab we see that scanning cannot be hidden, at least not if not done so retroactively, and traffic like this is the first sign of a malicious actor, and in this user and intruder profiling can be done to disable connections or perform other means of prevention against an attack when these scanning patterns arise. It also highlights the need for test documentation, as if a vulnerability scan is permitted by system managers, documentation and clear parameters and permissions must be set so that the network can be managed accordingly with the upmost security.

Bibliography

Active scanning: Vulnerability scanning. Active Scanning: Vulnerability Scanning, Sub-technique T1595.002 - Enterprise | MITRE ATT&CK®. (n.d.). https://attack.mitre.org/techniques/T1595/002/

Browserprotocol. BrowserProtocol - Wireshark Wiki. (n.d.). https://wiki.wireshark.org/BrowserProtocol

Decker, E. (2023, May 15). *How to view Windows 10 crash logs and error logs*. TechCult. https://techcult.com/how-to-view-windows-10-crash-logs-and-error-logs/

Ethernet statistics. The Packet Company. (n.d.). https://packet.company/ethernet-statistics

Flexi soft – RK512 - sick germany. (n.d.-a). https://cdn.sick.com/media/docs/2/02/302/online_help_flexi_soft_rk512_en_im0048302.pdf

GmbH, T. (n.d.). *TPPRN.DLL download and fix TPPRN.DLL missing error: Fileinspect*. File Inspect Library. https://www.fileinspect.com/fileinfo/tpprn-dll/

Goikhman, M. (n.d.). *SMB::DCERPC*. MetaCPAN. https://metacpan.org/pod/SMB::DCERPC

Impair defenses: Disable or modify tools. Impair Defenses: Disable or Modify Tools, Sub-technique T1562.001 - Enterprise | MITRE ATT&CK®. (n.d.). https://attack.mitre.org/techniques/T1562/001/

Mallick, C. (2025, March 10). *Why nessus scanner is the Cybersecurity Solution You Want - Spiceworks*. Spiceworks Inc. https://www.spiceworks.com/it-security/data-security/articles/what-is-nessus-scanner/

Miller, M. (2021, March 22). *Vulnerability walkthrough - NBNS and LLMNR spoofing*. Triaxiom Security. https://www.triaxiomsecurity.com/vulnerability-walkthrough-nbns-and-llmnr-spoofing/#:~:text=What%20are%20NBNS%20and%20LLMNR%3F%20Both%20NetBIOS%20Name,through%20the%20organizational%20DNS%20%28Domain%20Name%20Server%29%20server.

NTP - how does it work?. NTP. (2022, June 27). https://www.ntp.org/ntpfaq/ntp-s-algo/

SSDP. SSDP - Wireshark Wiki. (n.d.). https://wiki.wireshark.org/SSDP

Types of network protocols explained with functions. ComputerNetworkingNotes. (n.d.). https://www.computernetworkingnotes.com/networking-tutorials/types-of-network-protocols-explained-with-functions.html

What is IGMP? | internet group management protocol | cloudflare. (n.d.-b). https://www.cloudflare.com/learning/network-layer/what-is-igmp/

What is the reason for malformed packet error ?. Wireshark Q&A. (n.d.). https://osqa-ask.wireshark.org/questions/52190/what-is-the-reason-for-malformed-packet-error/

WINMGMT - win32 apps. Win32 apps | Microsoft Learn. (n.d.-a). https://learn.microsoft.com/en-us/windows/win32/wmisdk/winmgmt

WMI providers - win32 apps. Win32 apps | Microsoft Learn. (n.d.-b). https://learn.microsoft.com/en-us/windows/win32/wmisdk/wmi-providers