# Lab #4: Attack Analysis

Inor Wang

Department of Cyber Security, The University of Texas at San Antonio

IS 3523.004

Professor Munoz

May 4, 2025

**Table of Contents**



**Introduction**

In today's threat landscape, the ability to respond to and investigate cyberattacks is a critical skill for any security analyst. This lab provided an opportunity to simulate a real-world security incident by examining a compromised system through collected forensic artifacts. The goal was to analyze the attack, identify key indicators of compromise, and determine how the attacker operated within the network. Provided materials included event logs and a network packet capture file, which served as the foundation for building a timeline of events and uncovering the techniques used by the threat actor. The investigation was conducted using widely adopted forensic tools such as Wireshark, SNORT, NetworkMiner, and the Windows Event Viewer, each of which offered unique insights into the nature and scope of the attack. The lab emphasized the importance of combining network-level data with host-based logs to form a complete picture of the incident. Throughout the analysis, attention was given to patterns of suspicious traffic, abnormal service behavior, and evidence of potential exploitation. Time synchronization issues and logging gaps were also discovered, reinforcing the real-world challenges that analysts face during investigations. In addition, the lab reinforced the importance of correlating findings across tools and applying defensive reasoning to interpret how attacks unfold in stages. By simulating attacker behavior and reconstructing their path, I developed a deeper appreciation for how layered security measures and thorough documentation can impact the success of an investigation. Overall, this lab allowed me to apply key investigative methods in a practical context, sharpen my attention to detail, and better understand how to detect, document, and defend against unauthorized activity in a networked environment.

## Objectives

The primary objective of this lab is to conduct a structured and comprehensive forensic analysis of a suspected intrusion targeting a legacy Windows system. Leveraging forensic artifacts collected during initial incident response, including a network packet capture, security logs, and application logs, students are expected to identify signs of malicious activity, trace the attacker's behavior, and reconstruct the timeline of the incident. The investigation emphasizes practical skills in network traffic analysis, host log correlation, and cross-tool validation using platforms such as SNORT, Wireshark, NetworkMiner, and Event Viewer. Specific focus is placed on detecting denial-of-service patterns, abnormal SMB traffic, printer spooler exploitation, and stealth techniques involving TCP reset manipulation. Additionally, students are challenged to interpret log gaps caused by misconfigured time settings and assess how such oversights can hinder incident response efforts. By the end of the lab, students will have strengthened their ability to detect multi-stage attacks, validate findings across diverse data sources, and document their conclusions using techniques that align with modern cybersecurity investigation standards.

## Methodology

So, after reading the lab instructions, I proceeded to SimSpace and went into one of the Win-Hunt machines. Specifically, I proceeded to win-hunt-25 and proceeded to the sharedfiles

folder to extract the two event logs and the .pcapng file. I then proceeded to the Desktop and created a folder named, “inor-lab4”, to keep all of the documents and files that I use for this lab in the same area and organized. I then copy and pasted the three files into that folder as shown in Figure 1. Since .pcapng is not compatible with most programs, I went into Wireshark and simply converted the file into a regular .pcap file as shown in Figure 2 which was saved into the “inor-lab4” directory as shown in Figure 1. The .pcapng (Packet Capture Next Generation) format is a newer standard used by modern versions of Wireshark that supports additional features such as multiple interfaces, enhanced timestamps, and metadata fields. However, despite its flexibility, many legacy tools and forensic platforms still only support the traditional .pcap format. These older tools often do not recognize or properly interpret .pcapng-specific fields, which can lead to compatibility issues during analysis or automated processing. For this reason, converting .pcapng to .pcap is a common best practice when ensuring broad compatibility with IDS tools like Snort.

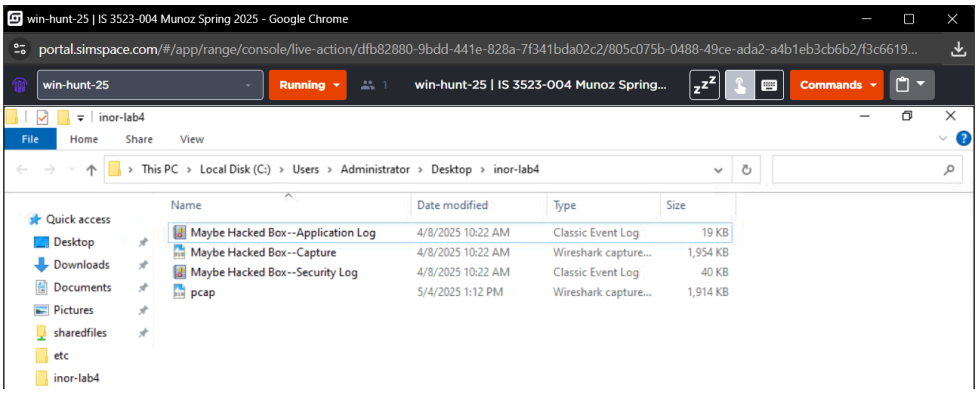



*Figure 1: "inor-lab4" directory*

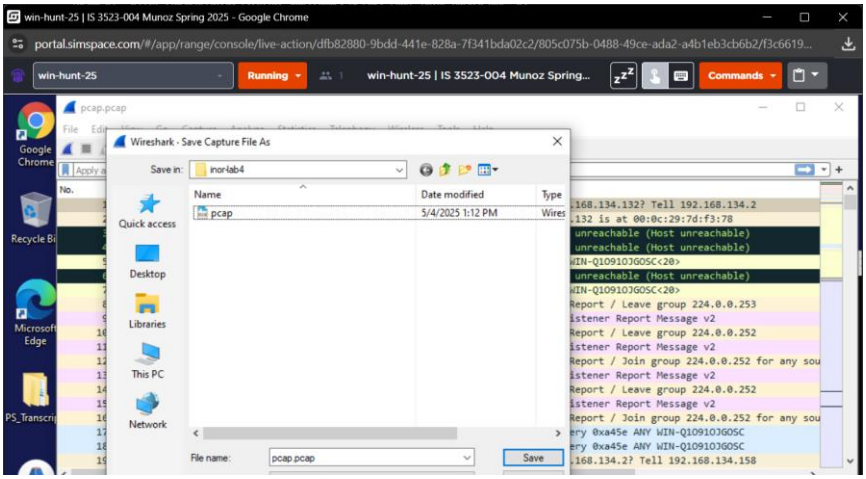



*Figure 2: Converting .pcapng to .pcap in Wireshark*

I then remembered from previous labs about SNORT and the importance of it whilst conducting investigations and hunting. SNORT is a powerful, open-source network intrusion detection system (NIDS) used by security professionals to analyze packet captures and detect malicious patterns based on a comprehensive set of user-defined rules. Its ability to scan for

known threats, suspicious payloads, and anomalous traffic behavior makes it an essential tool in many real-world incident response workflows. I noticed that the SNORT was installed in the Desktop within this VM so I opened a terminal, proceeded to the bin folder, and then wrote the command: snort -c c:\Users\Administrator\Desktop\Snort\etc\snort.conf -r "c:\Users\Administrator\Desktop\inor-lab4\pcap.pcap" -A console > c:\Users\Administrator\Desktop\inor-lab4\pcap.txt to run snort against the .pcap file and spit it out into a .txt file within the directory I created on the Desktop as shown in Figure 3. As shown in Figure 4, this allowed for a clean export of alerts, making it easier to review SNORT's findings and incorporate them into my broader analysis.

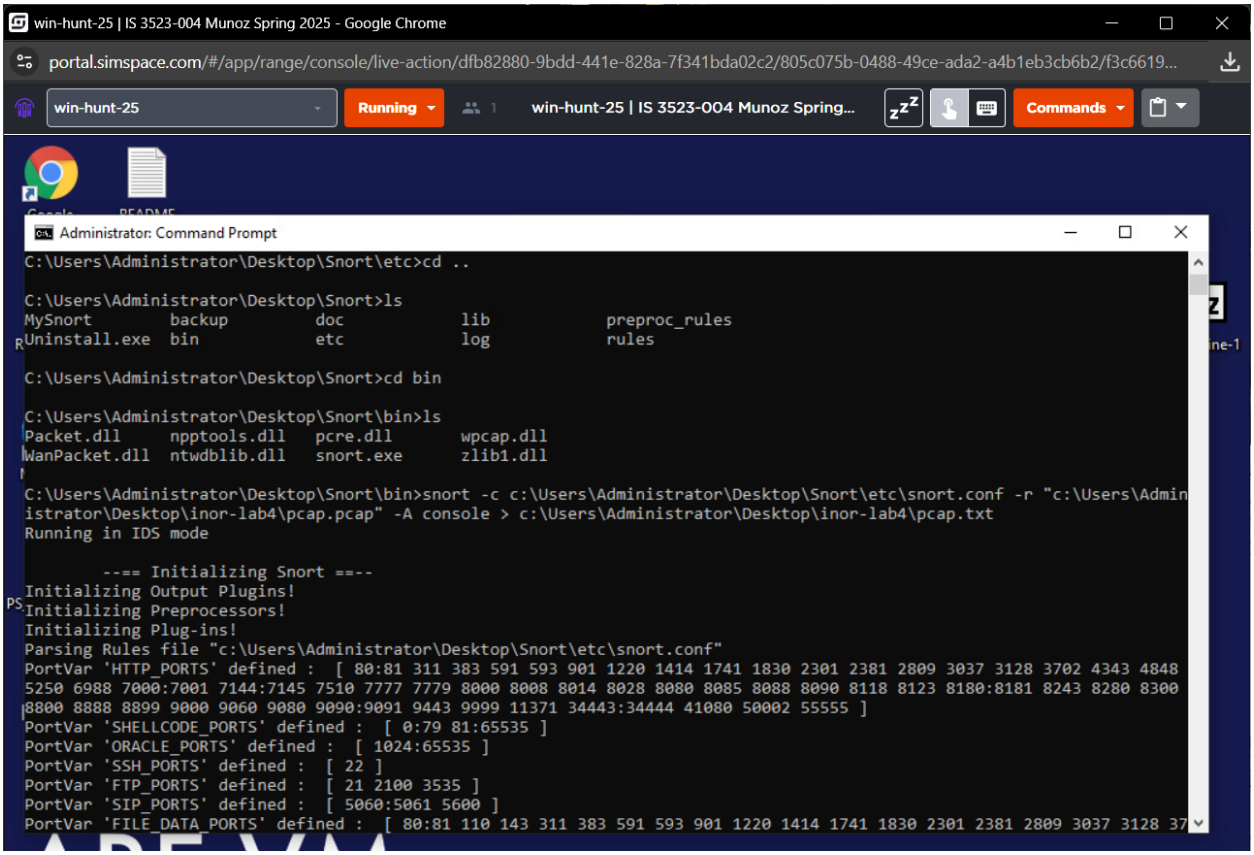



*Figure 3: SNORT command*

As shown in Figure 4, the SNORT output was successfully redirected into the pcap.txt file located in my working directory. The alert logs reveal a repeated pattern of "Consecutive TCP small segments exceeding threshold," which, while not always immediately threatening on its own, may become significant when correlated with other suspicious activity on the network. These types of alerts typically suggest fragmentation, intentionally small packet sizes, or unusual traffic flows, all of which could be indicative of evasion techniques, network mapping, or slow scanning behavior designed to avoid detection by intrusion detection systems. In addition to the fragmentation-related alerts, SNORT also flagged a clear instance of a TCP port scan, with traffic originating from IP address 192.168.134.129 and targeting 192.168.134.132. Port

scanning is a reconnaissance technique often used by attackers to probe a target system for open or vulnerable ports before launching an exploit. This is a more direct indicator of reconnaissance activity and is commonly observed during the initial phases of network-based attacks, where the attacker is gathering intelligence about the environment. Throughout the alert log, only two IP addresses, 192.168.134.132 and 192.168.134.129, are involved, which strongly suggests a focused and possibly targeted exchange. This lack of diversity in communicating hosts could imply that the attacker was specifically interested in this target and not scanning the broader network. This observation guided me to create relevant filters in Wireshark to isolate and analyze the conversation between these two hosts. Moreover, the presence of numerous connections involving high-numbered (ephemeral) source ports from 192.168.134.129 is notable. Ephemeral ports, typically ranging from 49152 to 65535, are dynamically assigned by a client machine during outbound communications. However, they can also be leveraged in malicious activity such as reverse shells, exfiltration channels, or anonymized scanning operations, particularly when used in high volume or with irregular patterns. High ports are dynamically assigned and frequently used by malware or port scanning tools to obfuscate traffic and avoid detection. This initial SNORT output provided valuable early insights into the potential nature of the activity, but it does not provide full context on its own. Therefore, I plan to use additional tools, such as Wireshark for packet-level analysis and NetworkMiner for host-based artifact extraction, to expand my investigation. By correlating alerts from multiple tools, I aim to build a clearer and more reliable timeline of the potential attack. Overall, this step demonstrates how SNORT can play a vital role in the early stages of threat hunting by offering pattern-based alerts that help direct further forensic inquiry. Its ability to surface low-level anomalies gives analysts a crucial starting point for deeper packet and behavior analysis, especially in environments where visibility is limited.

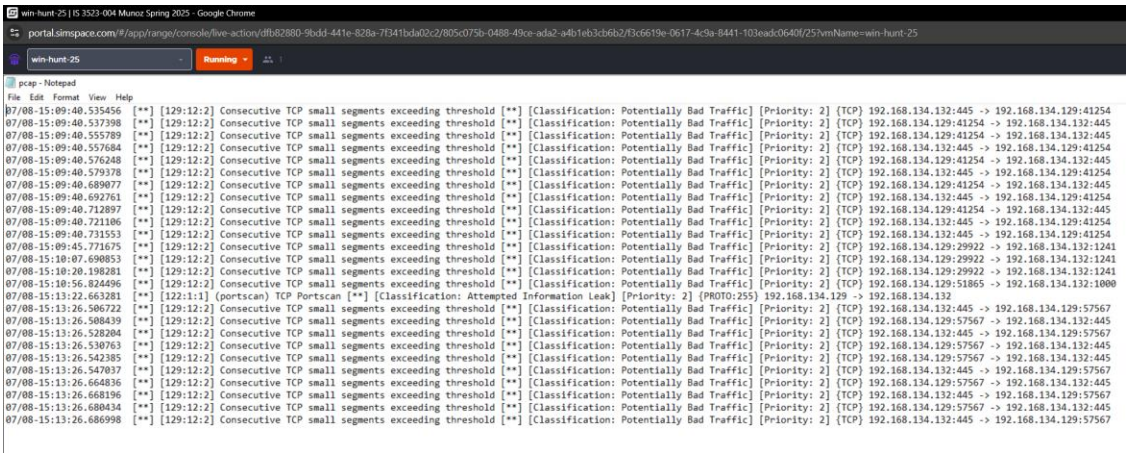



*Figure 4: SNORT output*

I first proceeded to NetworkMiner to help better visualize and understand the overall structure and flow of the incident. NetworkMiner is a powerful network forensic analysis tool that passively parses packet capture data and organizes it into meaningful artifacts such as host information, session metadata, file transfers, and operating system fingerprints. By using this

tool, I was able to gain a better understanding of the network environment, which is essential when attempting to identify compromised systems or suspicious behavior. I specifically navigated to the Hosts tab, where I immediately noticed the same two IP addresses, 192.168.134.129 and 192.168.134.132, that were heavily involved in the SNORT alerts earlier in the investigation, which begs to be further examined as shown in Figure 5. For the IP address 192.168.134.129, NetworkMiner identified the host as running a Linux Debian operating system with 100% confidence, and flagged the presence of Ettercap, a well-known network attack tool frequently used for man-in-the-middle attacks, credential interception, and packet manipulation. The fact that Ettercap was observed on this host is a strong indicator that this machine is not a passive participant, but an active participant or even an active attacker. The MAC address of this device, 00:0C:29:3F:20:31, and the use of VMware also suggest that it may be operating in a virtualized environment, another common tactic among attackers. Its behavior, tools, and system fingerprint collectively support the theory that 192.168.134.129 is the intruder or attacker system in this scenario. In contrast, the second IP address, 192.168.134.132, is labeled with the hostname APLOVERS-765952 and was detected as running an older Windows OS, either Windows XP or Windows 2000. These legacy systems are often more vulnerable to exploitation, particularly when exposed over protocols like SMB (Server Message Block), which is notorious for its role in historical cyber attacks such as EternalBlue and WannaCry. This host appeared to be the primary target of the suspicious communication, and its role became clearer after viewing the consistent exchange of data between these two endpoints. What stood out further was the number of sessions initiated from 192.168.134.129 to 192.168.134.132 over TCP port 445, a port commonly associated with SMB file sharing, remote command execution, and lateral movement. This behavior aligns directly with the SNORT alerts saved in the pcap.txt output as shown in Figure 4, where repeated connections, excessive small TCP segments, and a port scan were detected. Additionally, NetworkMiner's Sessions and Parameters tabs revealed the use of high-numbered ephemeral source ports, such as 41254 and 57567, by the attacker as shown in Figure 6. These dynamic ports are commonly used by client systems but may also be manipulated by malicious attackers. Their repeated appearance in conjunction with known attack protocols strengthens the hypothesis of malicious intent. Overall, NetworkMiner served as a critical pivot point in this investigation. It not only confirmed previous findings uncovered through SNORT, but also expanded on them by providing a more complete picture of host behavior and session context. This multi-layered approach, beginning with raw file extraction, conversion and inspection of the .pcap file, alerting through SNORT, and then deep host-level analysis with NetworkMiner, demonstrates the importance of using complementary forensic tools. With a stronger understanding of the attacker's tactics and the victim’s vulnerabilities, I plan to continue

this investigation using Wireshark for a granular inspection of payload data, protocol behavior, and potential evidence of exploitation.

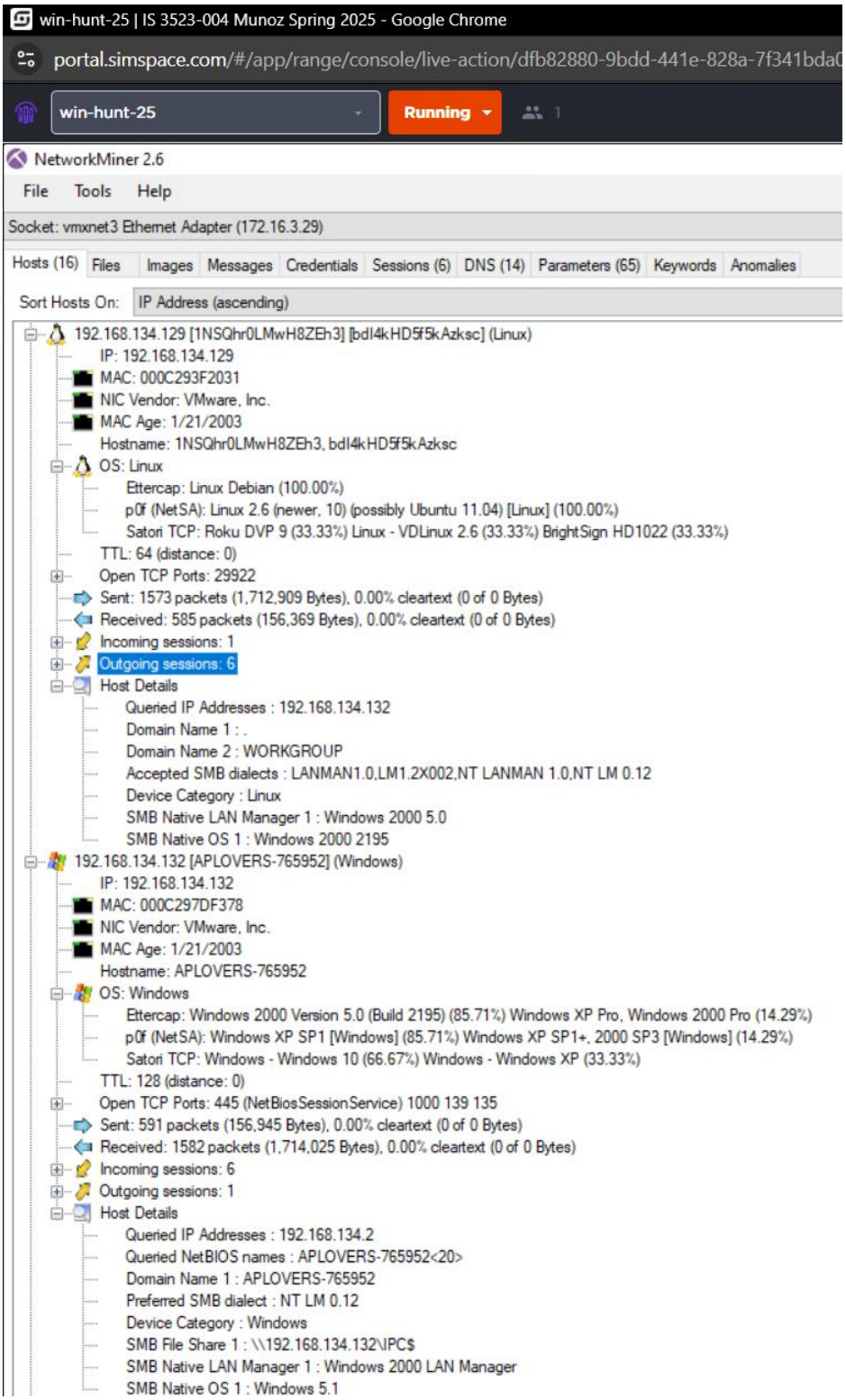
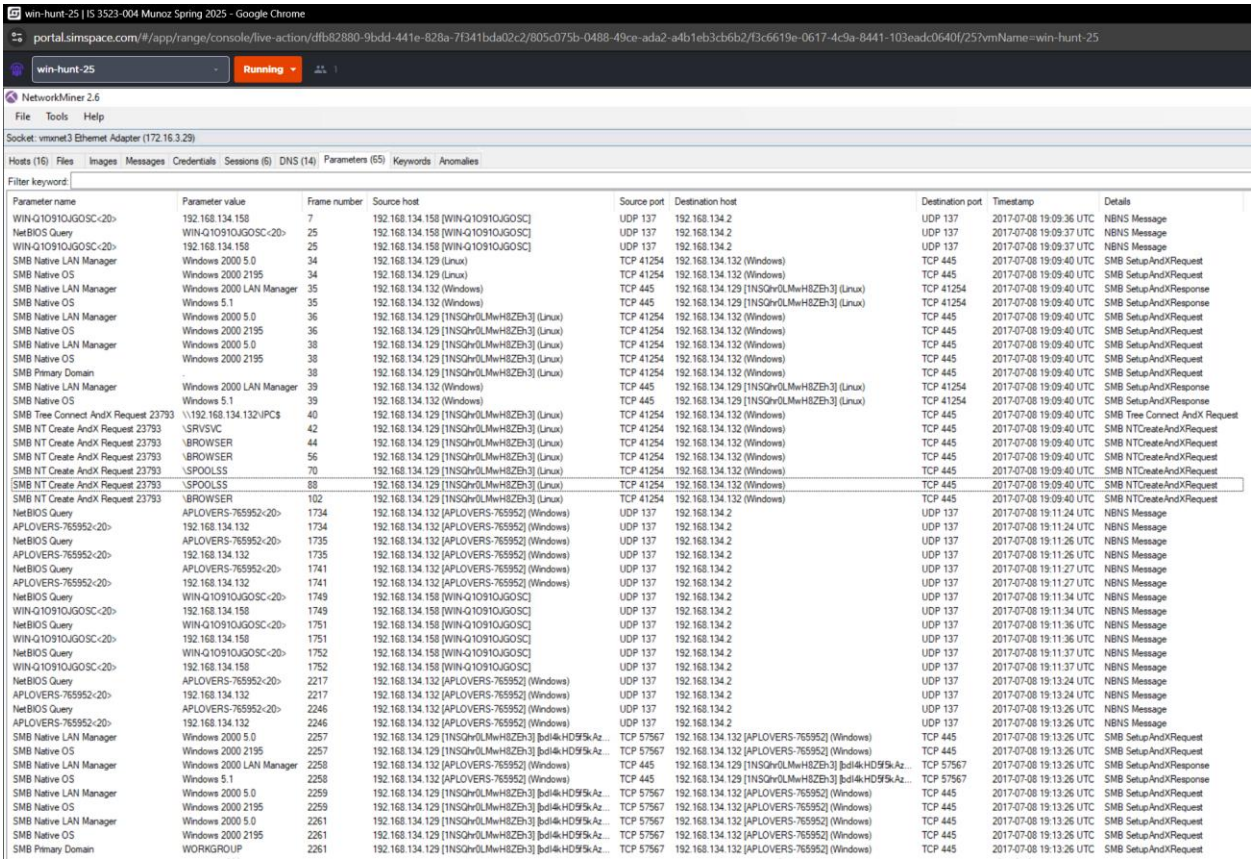



*Figure 5: Hosts tab within NetworkMiner*




| Parameter name | Parameter value | Frame number | Source host | Source port | Destination host | Destination port | Timestamp | Details |
|---|---|---|---|---|---|---|---|---|
| WIN-Q1O91OJGOSC<20> | 192.168.134.158 | 7 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:09:36 UTC | NBNS Message |
| NetBIOS Query | WIN-Q1O91OJGOSC<20> | 25 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:09:37 UTC | NBNS Message |
| WIN-Q1O91OJGOSC<20> | 192.168.134.158 | 25 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:09:37 UTC | NBNS Message |
| SMB Native LAN Manager | Windows 2000 5.0 | 34 | 192.168.134.129 (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB SetupAndXRequest |
| SMB Native OS | Windows 2000 2195 | 34 | 192.168.134.129 (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB SetupAndXRequest |
| SMB Native LAN Manager | Windows 2000 LAN Manager | 35 | 192.168.134.132 (Windows) | TCP 445 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 2017-07-08 19:09:40 UTC | SMB SetupAndXResponse |
| SMB Native OS | Windows 5.1 | 35 | 192.168.134.132 (Windows) | TCP 445 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 2017-07-08 19:09:40 UTC | SMB SetupAndXResponse |
| SMB Native LAN Manager | Windows 2000 5.0 | 36 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB SetupAndXRequest |
| SMB Native OS | Windows 2000 2195 | 36 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB SetupAndXRequest |
| SMB Native LAN Manager | Windows 2000 5.0 | 38 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB SetupAndXRequest |
| SMB Native OS | Windows 2000 2195 | 38 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB SetupAndXRequest |
| SMB Primary Domain | . | 38 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB SetupAndXRequest |
| SMB Native LAN Manager | Windows 2000 LAN Manager | 39 | 192.168.134.132 (Windows) | TCP 445 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 2017-07-08 19:09:40 UTC | SMB SetupAndXResponse |
| SMB Native OS | Windows 5.1 | 39 | 192.168.134.132 (Windows) | TCP 445 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 2017-07-08 19:09:40 UTC | SMB SetupAndXResponse |
| SMB Tree Connect AndX Request 23793 | \\192.168.134.132\IPC$ | 40 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB Tree Connect AndX Request |
| SMB NT Create AndX Request 23793 | \SRVSVC | 42 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB NTCreateAndXRequest |
| SMB NT Create AndX Request 23793 | \BROWSER | 44 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB NTCreateAndXRequest |
| SMB NT Create AndX Request 23793 | \BROWSER | 56 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB NTCreateAndXRequest |
| SMB NT Create AndX Request 23793 | \SPOOLSS | 70 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB NTCreateAndXRequest |
| SMB NT Create AndX Request 23793 | \SPOOLSS | 88 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB NTCreateAndXRequest |
| SMB NT Create AndX Request 23793 | \BROWSER | 102 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] (Linux) | TCP 41254 | 192.168.134.132 (Windows) | TCP 445 | 2017-07-08 19:09:40 UTC | SMB NTCreateAndXRequest |
| NetBIOS Query | APLOVERS-765952<20> | 1734 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:24 UTC | NBNS Message |
| APLOVERS-765952<20> | 192.168.134.132 | 1734 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:24 UTC | NBNS Message |
| NetBIOS Query | APLOVERS-765952<20> | 1735 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:26 UTC | NBNS Message |
| APLOVERS-765952<20> | 192.168.134.132 | 1735 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:26 UTC | NBNS Message |
| NetBIOS Query | APLOVERS-765952<20> | 1741 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:27 UTC | NBNS Message |
| APLOVERS-765952<20> | 192.168.134.132 | 1741 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:27 UTC | NBNS Message |
| NetBIOS Query | WIN-Q1O91OJGOSC<20> | 1749 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:34 UTC | NBNS Message |
| WIN-Q1O91OJGOSC<20> | 192.168.134.158 | 1749 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:34 UTC | NBNS Message |
| NetBIOS Query | WIN-Q1O91OJGOSC<20> | 1751 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:36 UTC | NBNS Message |
| WIN-Q1O91OJGOSC<20> | 192.168.134.158 | 1751 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:36 UTC | NBNS Message |
| NetBIOS Query | WIN-Q1O91OJGOSC<20> | 1752 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:37 UTC | NBNS Message |
| WIN-Q1O91OJGOSC<20> | 192.168.134.158 | 1752 | 192.168.134.158 [WIN-Q1O91OJGOSC] | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:11:37 UTC | NBNS Message |
| NetBIOS Query | APLOVERS-765952<20> | 2217 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:13:24 UTC | NBNS Message |
| APLOVERS-765952<20> | 192.168.134.132 | 2217 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:13:24 UTC | NBNS Message |
| NetBIOS Query | APLOVERS-765952<20> | 2246 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:13:26 UTC | NBNS Message |
| APLOVERS-765952<20> | 192.168.134.132 | 2246 | 192.168.134.132 [APLOVERS-765952] (Windows) | UDP 137 | 192.168.134.2 | UDP 137 | 2017-07-08 19:13:26 UTC | NBNS Message |
| SMB Native LAN Manager | Windows 2000 5.0 | 2257 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 2017-07-08 19:13:26 UTC | SMB SetupAndXRequest |
| SMB Native OS | Windows 2000 2195 | 2257 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 2017-07-08 19:13:26 UTC | SMB SetupAndXRequest |
| SMB Native LAN Manager | Windows 2000 LAN Manager | 2258 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 2017-07-08 19:13:26 UTC | SMB SetupAndXResponse |
| SMB Native OS | Windows 5.1 | 2258 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 2017-07-08 19:13:26 UTC | SMB SetupAndXResponse |
| SMB Native LAN Manager | Windows 2000 5.0 | 2259 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 2017-07-08 19:13:26 UTC | SMB SetupAndXRequest |
| SMB Native OS | Windows 2000 2195 | 2259 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 2017-07-08 19:13:26 UTC | SMB SetupAndXRequest |
| SMB Native LAN Manager | Windows 2000 5.0 | 2261 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 2017-07-08 19:13:26 UTC | SMB SetupAndXRequest |
| SMB Native OS | Windows 2000 2195 | 2261 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 2017-07-08 19:13:26 UTC | SMB SetupAndXRequest |
| SMB Primary Domain | WORKGROUP | 2261 | 192.168.134.129 [1NSQhr0LMwH8ZEh3] [bdl4kHD9f5kAz... | TCP 57567 | 192.168.134.132 [APLOVERS-765952] (Windows) | TCP 445 | 2017-07-08 19:13:26 UTC | SMB SetupAndXRequest |

*Figure 6: Parameters tab within NetworkMiner*

I then proceeded to analyze the packet capture in Wireshark to gain a better understanding of the activity between the suspected attacker and victim systems. Within just a few seconds of applying filters to isolate communication between 192.168.134.129 (the Linux attacker) and 192.168.134.132 (the Windows host APLOVERS-765952), I observed a high concentration of SMB protocol traffic, specifically over TCP port 445, which is commonly used for file sharing and remote device access in Windows environments. This immediately raised a red flag, as SMB traffic is a well-known vector for lateral movement and exploitation, particularly when targeting legacy operating systems like Windows XP or 2000. What stood out even more was the presence of numerous calls to EnumPrinters, a Windows API call typically used to enumerate available printers on a remote machine. In Wireshark, these calls appeared within DCERPC (Distributed Computing Environment / Remote Procedure Call) traffic tunneled over SMB, indicating that the attacker was attempting to query the list of printers configured on the APLOVERS system as shown in Figure 7. The repeated use of this enumeration function suggests the attacker may have been attempting to gather intelligence on networked printers in order to exploit printer spooler vulnerabilities and get access to the network. Digging deeper into the SMB protocol traffic, I observed several references to pipe spoolss, which refers to the spooler subsystem named pipe, a critical component of Windows' print services. Named pipes like spoolss allow for inter-process communication and are often targeted in attacks because they are exposed by default and can sometimes be accessed with limited or no authentication in

legacy environments. The use of \pipe\spoolss confirms that the attacker was not just scanning the system but was specifically targeting print-related services, which have historically been abused in vulnerabilities. The technique for these types of attacks is that they use spooler communication to inject commands, impersonate users, or move laterally within the network. Furthermore, by following the TCP stream associated with this traffic, I could see SMB headers and content embedded in what appears to be printer and device enumeration responses as shown in Figure 8. The string data within the payload (as shown in the TCP stream view) includes identifiers like Windows NT, Remote Printers, and Internet Providers, all of which reinforce that the attacker was successfully querying and retrieving configuration data from the spooler service running on APLOVERS-765952. This confirms that the target host is not only running Windows 2000/XP but also has its print services exposed, likely making it a high-risk endpoint within the environment. The actor behind 192.168.134.129 appears to have been conducting targeted enumeration of print services, likely with the intention of exploiting exposed SMB functionality for lateral movement, persistence, or data manipulation.

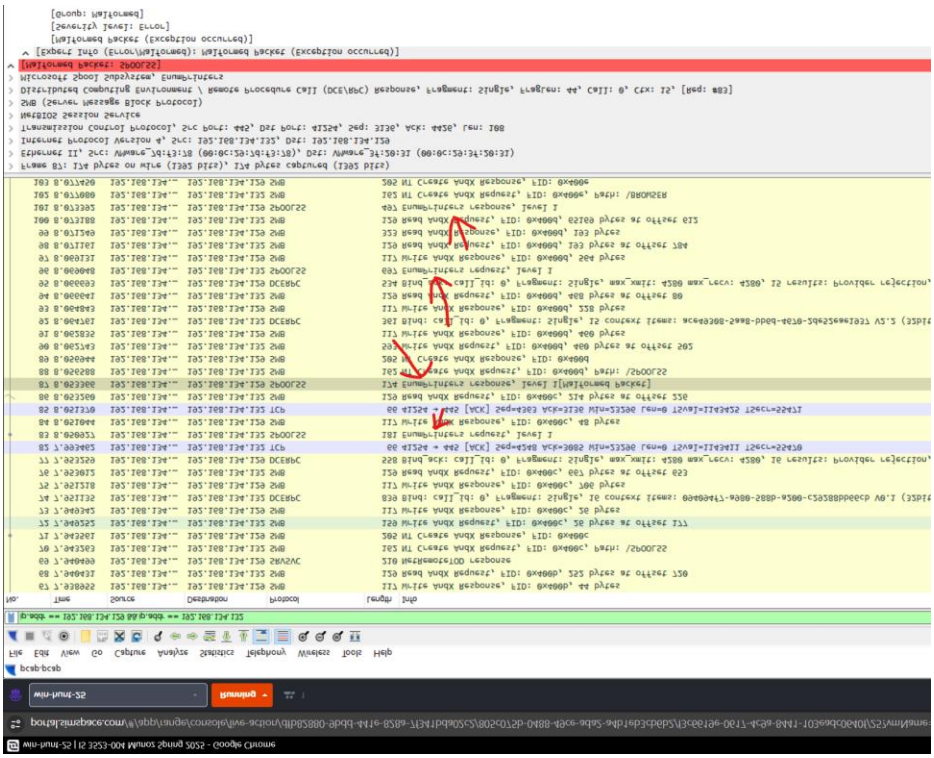


*Figure 7: Wireshark showing EnunPrinters*

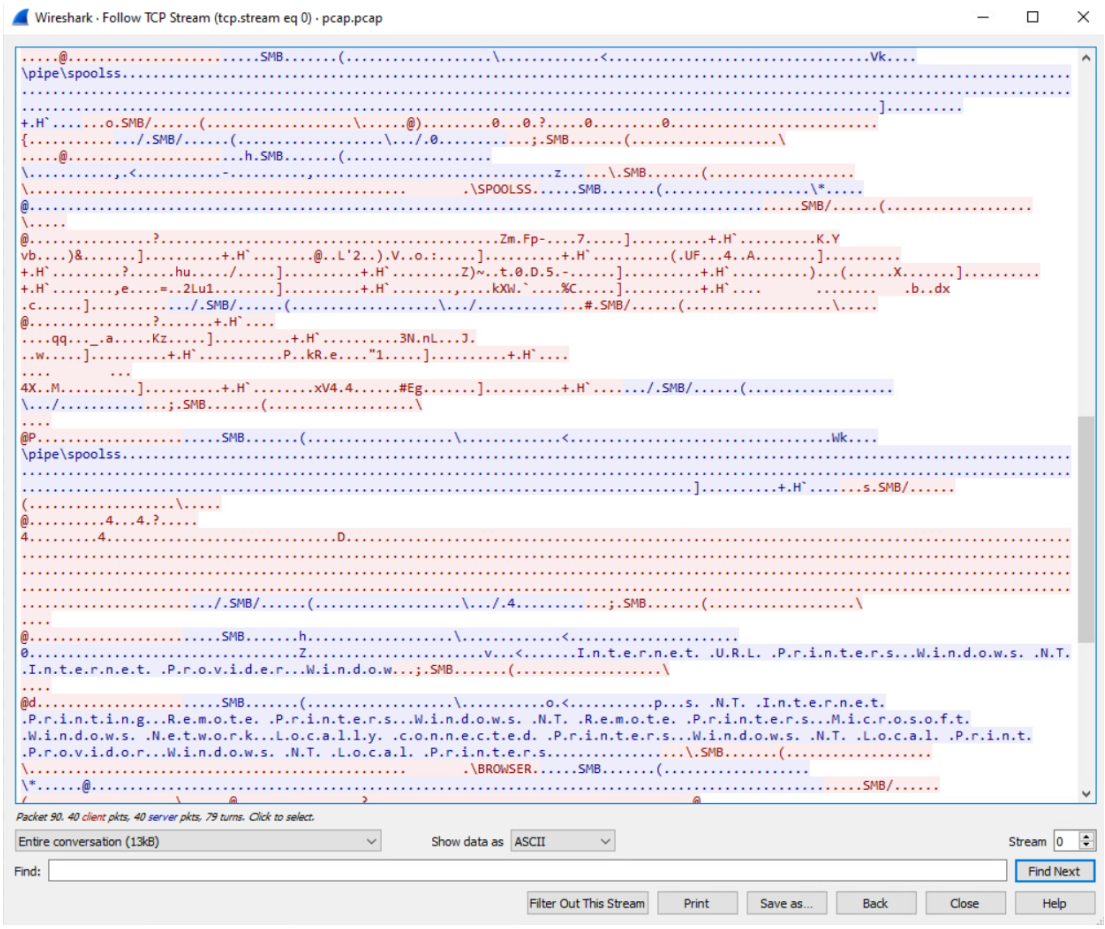



*Figure 8: TCP Stream*

After completing the analysis of the SMB and printer enumeration traffic, I turned my attention to the broader patterns within Wireshark. Using Wireshark to isolate traffic between 192.168.134.129 (the attacker) and 192.168.134.132 (the target), I identified a massive volume of TCP ACK packets with little to no accompanying data. As shown in Figure 9, the attacker flooded the victim with hundreds of ACK segments, creating a high volume of seemingly valid, yet non-productive traffic. This technique is a known Denial-of-Service (DoS) method, often referred to as an ACK flood, which is used to overwhelm the target system's network stack, force CPU consumption, and exhaust memory allocated for connection tracking. Unlike SYN floods, which disrupt the handshake process, ACK floods target established or half-open connections, making them more difficult to block using traditional firewall rules. The intention here was not to establish legitimate communication, but to degrade the performance of the APLOVERS-765952 host and reduce its ability to respond to legitimate requests. A system overwhelmed with excessive ACK packets may struggle with logging, fail to allocate enough resources for processing new requests, or even crash. This behavior demonstrates a clear attempt to destabilize the environment prior to exploitation, ensuring that subsequent intrusion efforts encounter minimal resistance from host-level or network-based defenses. Once the system was under pressure, the attacker transitioned to the second phase of the operation, infiltration using TCP

reset (RST) manipulation. As shown in Figure 10, the attacker initiates TCP handshakes and then immediately follows them with RST flags, which forcefully tear down connections. These TCP reset packets are not meant to disrupt services in this context, but rather to manipulate the connection state, bypass protocol verification checks, or evade detection by simulating broken or terminated connections. This is often done to disguise malicious SMB or RPC-based payloads, especially when probing vulnerable services like the Windows Print Spooler over port 445. By resetting sessions strategically, attackers can fly under the radar of certain intrusion detection systems that expect to see complete TCP flows. Together, these two behaviors, initial ACK flooding to overwhelm and soften the target, followed by RST-based stealth infiltration, highlight a calculated and staged attack pattern. The attacker first induced system fatigue through volume, then moved in with precision to exploit exposed services and enumerate printers using spooler-related named pipes. This layered tactic not only maximized disruption but also ensured deeper compromise with reduced chances of detection. Wireshark played a critical role in exposing these low-level details and validating what higher-level tools had already suggested. By tracing the attack path from SNORT alerts, to NetworkMiner's host view, and finally to Wireshark's insights, I was able to reconstruct the attacker’s likely objectives and assess the potential impact of their actions on the targeted system.

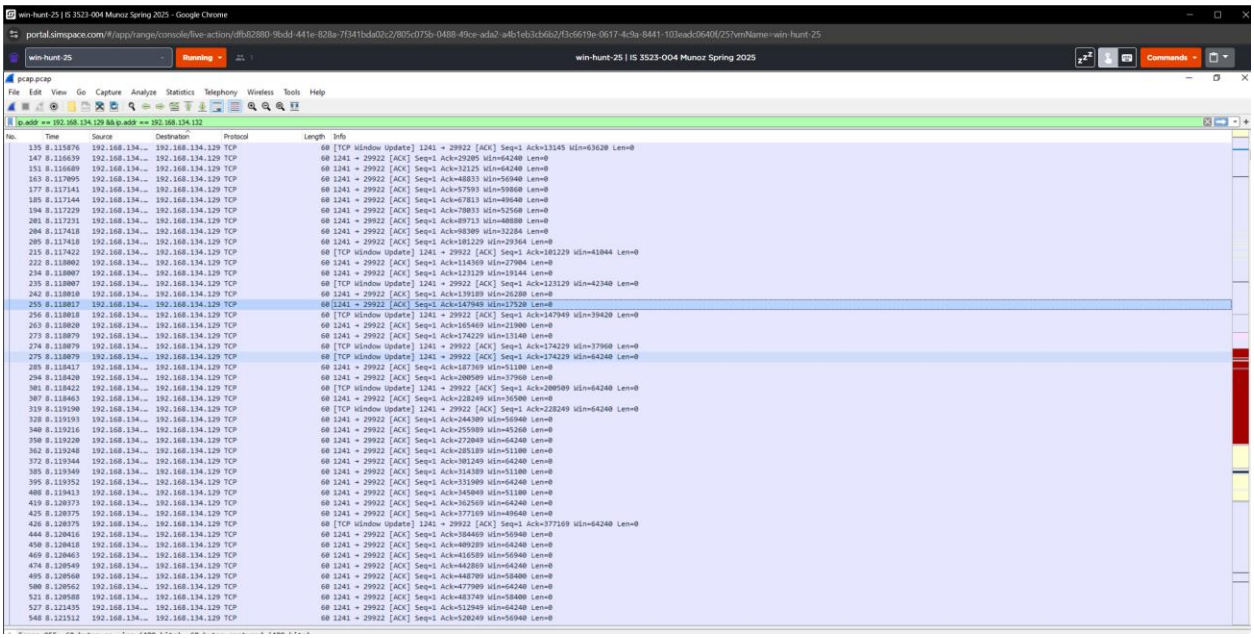


*Figure 9: Wireshark showing multile ACKs*

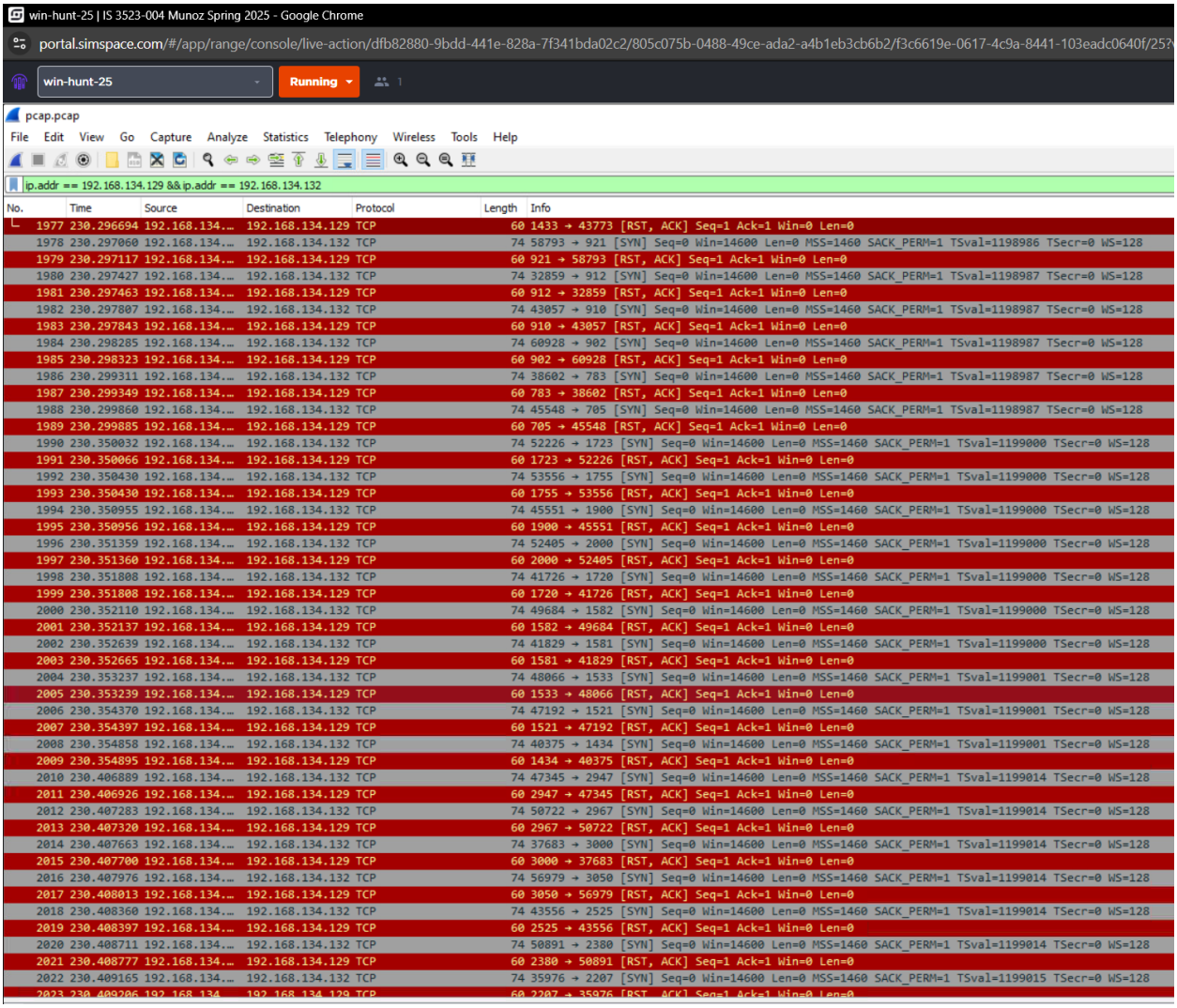
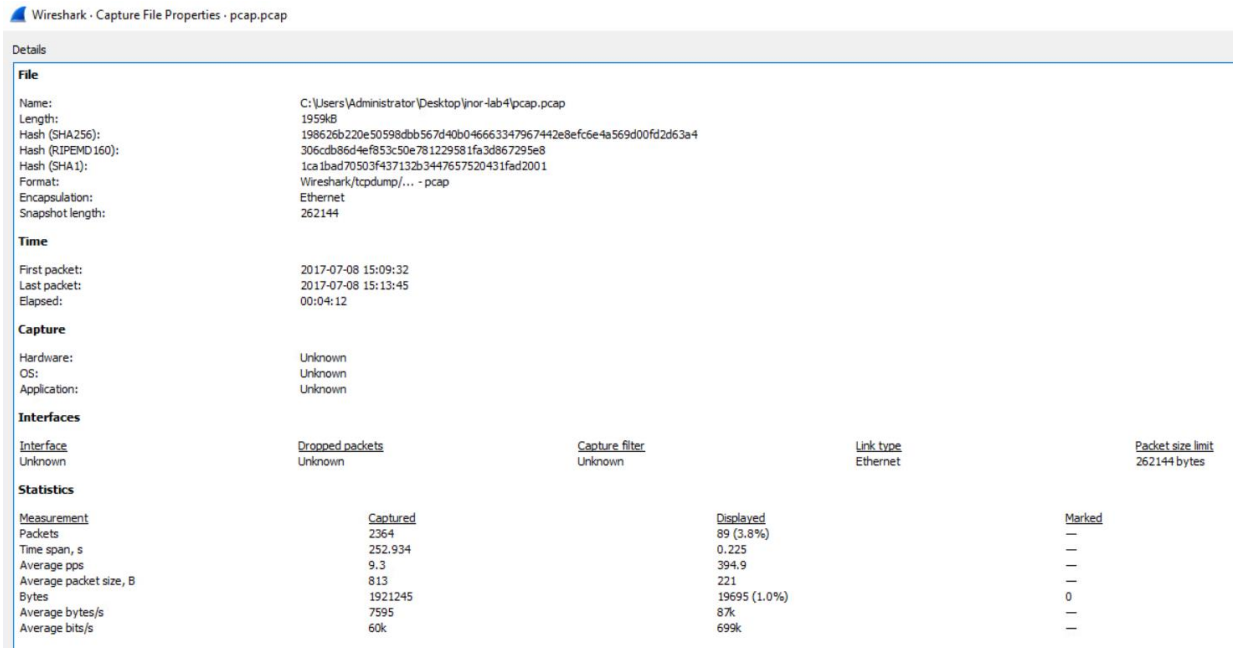



*Figure 10: Wireshark showing multiple TCP Resets*

To correlate network activity with local system behavior, I next examined the capture file properties in Wireshark. As shown in Figure 11, the .pcap file covered a very narrow window of activity, specifically from July 8, 2017, at 1:09:32 PM to 1:13:45 PM UTC. This short time span includes all the suspicious SMB traffic, enumeration attempts, TCP reset infiltration, and the ACK flood DDoS observed in earlier analysis phases. With this timeframe in hand, I turned to the Event Viewer to cross-reference system-level logs and verify if the APLOVERS-765952 system recorded any internal service failures, crashes, or suspicious activity that could support or further validate what was seen in the packet capture. However, after reviewing the security log in the Event Viewer as shown in Figure 12, I discovered that no events were recorded during the packet capture's timeframe, not even standard service messages or system info logs. The logs jumped from well before to well after the 1:09–1:13 PM window. This noticeable gap indicates that the system may not have been using or correctly configured with a time synchronization protocol, such as NTP (Network Time Protocol). The absence of synchronized time is a critical finding in any forensic investigation because it introduces uncertainty when correlating logs

across systems. Without consistent timestamps, it becomes extremely difficult to align network activity with system-level responses or trace an attacker's path across multiple machines in a network. Attackers often exploit this by disabling or spoofing system time to obfuscate their tracks. To gain further insights despite the time discrepancy, I applied a targeted strategy by searching for relevant services within the Event Viewer. Given that my earlier packet-level investigation in Wireshark revealed repeated references to the SPOOLSS named pipe and multiple EnumPrinters SMB calls, I focused my Event Viewer search around the Print Spooler service. This proved fruitful, as shown in Figure 13, where a system log entry (Event ID 7035) confirmed that the Print Spooler service was stopped at 4:13:07 PM on August 6, 2016. While this timestamp is out of sync with the .pcap, it aligns with the attacker's objective observed during the enumeration phase, which was to interact with the remote printing services. Searching for "spool" in the logs helped isolate a relevant activity trail tied directly to the attacker's target, providing strong context to the SMB-based intrusion we observed in Wireshark and NetworkMiner. Using Event Viewer shows that it remains a critical tool in forensic investigations. While network tools like Wireshark and SNORT capture the external interaction, Event Viewer shows the internal reaction, if properly configured. In a real-world setting, ensuring synchronized time and comprehensive logging is essential for reliable forensic correlation, legal admissibility of evidence, and building an accurate incident timeline. Had NTP been enabled and properly set, this would have allowed a perfect overlay between .pcap traffic and system responses, making the attack reconstruction far more precise.


Wireshark · Capture File Properties · pcap.pcap

Details

**File**

| | |
|---|---|
| Name: | C:\Users\Administrator\Desktop\jnor-lab4\pcap.pcap |
| Length: | 1959kB |
| Hash (SHA256): | 198626b220e50598dbb567d40b046663347967442e8efc6e4a569d00fd2d63a4 |
| Hash (RIPEMD160): | 306cdb86d4ef853c50e78122958 1fa3d867295e8 |
| Hash (SHA1): | 1ca1bad70503f437132b3447657520431fad2001 |
| Format: | Wireshark/tcpdump/... - pcap |
| Encapsulation: | Ethernet |
| Snapshot length: | 262144 |

**Time**

| | |
|---|---|
| First packet: | 2017-07-08 15:09:32 |
| Last packet: | 2017-07-08 15:13:45 |
| Elapsed: | 00:04:12 |

**Capture**

| | |
|---|---|
| Hardware: | Unknown |
| OS: | Unknown |
| Application: | Unknown |

**Interfaces**

| Interface | Dropped packets | Capture filter | Link type | Packet size limit |
|---|---|---|---|---|
| Unknown | Unknown | Unknown | Ethernet | 262144 bytes |

**Statistics**

| Measurement | Captured | Displayed | Marked |
|---|---|---|---|
| Packets | 2364 | 89 (3.8%) | — |
| Time span, s | 252.934 | 0.225 | — |
| Average pps | 9.3 | 394.9 | — |
| Average packet size, B | 813 | 221 | — |
| Bytes | 1921245 | 19695 (1.0%) | 0 |
| Average bytes/s | 7595 | 87k | — |
| Average bits/s | 60k | 699k | — |

*Figure 11: Capture File Properties*

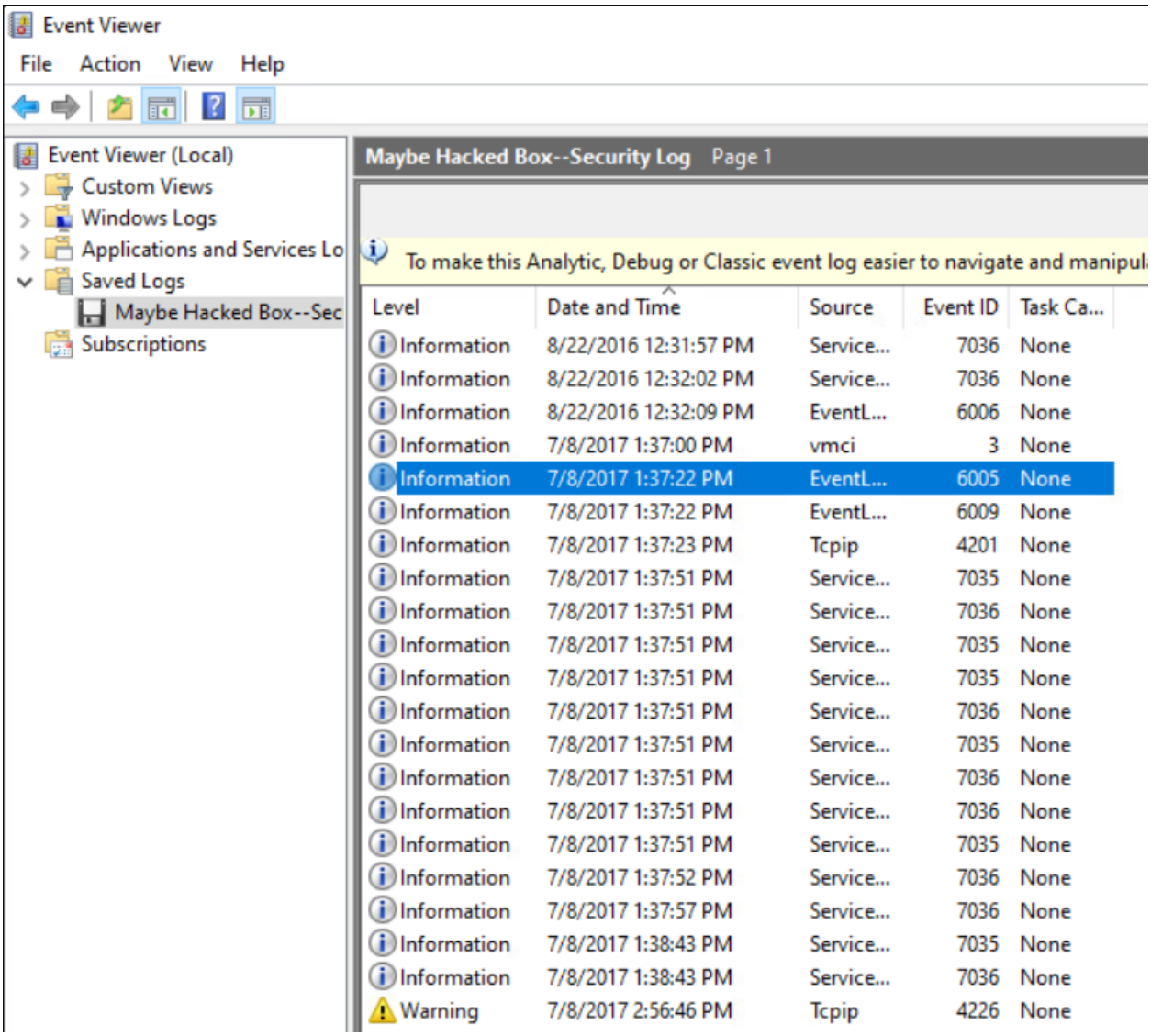



*Figure 12: Event Viewer showing that there is no time for the specified window*

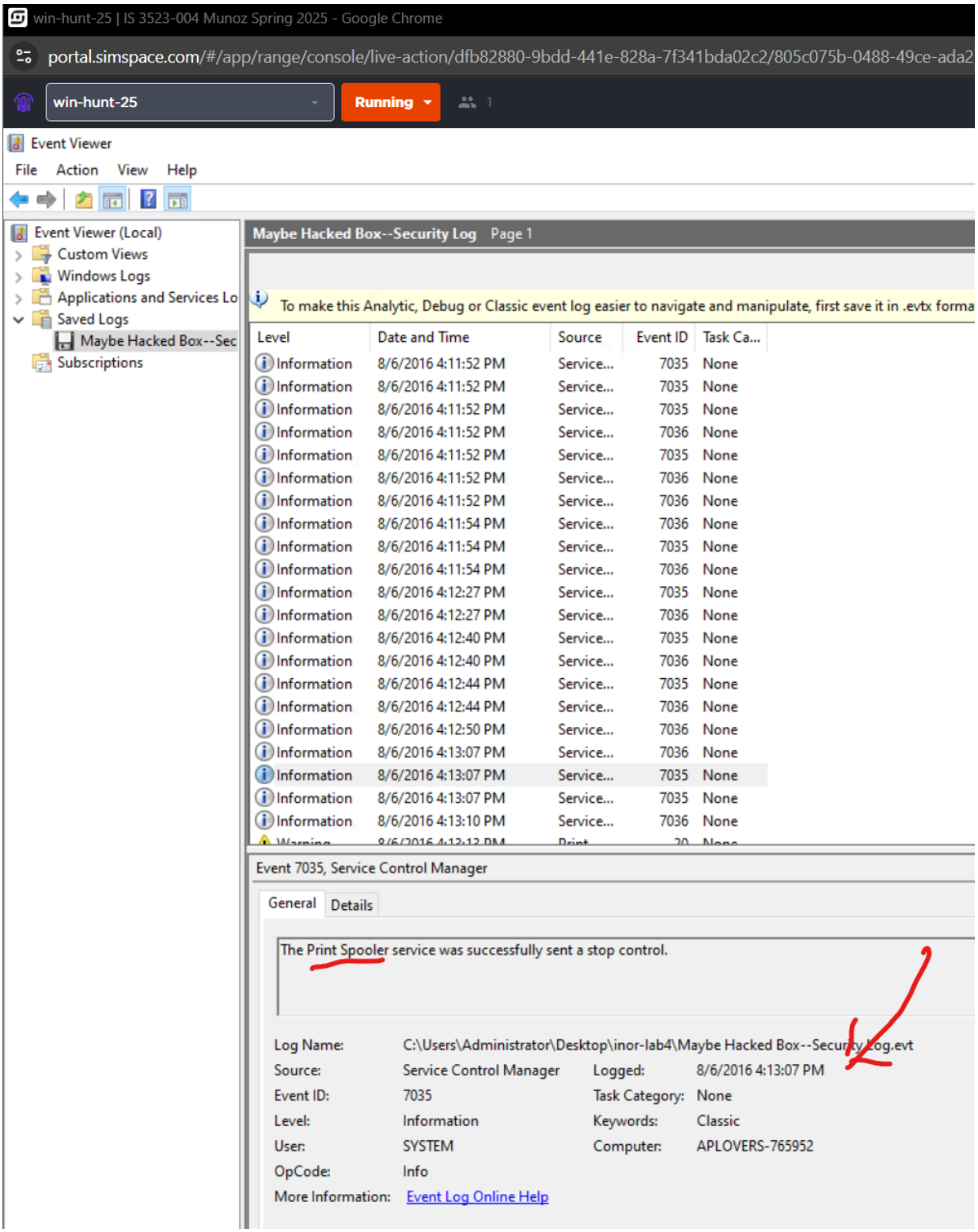



*Figure 13: Event Viewer showing that the incident was indeed logged*

**Story**

The attack began with a high-volume TCP ACK flood originating from IP address 192.168.134.129, a Linux Debian-based host operating within a virtualized environment. This traffic targeted 192.168.134.132, a legacy Windows XP/2000 system identified by its hostname as APLOVERS-765952. The flood consisted of hundreds of TCP packets with the ACK flag set, transmitted without any actual payload or session continuation. This form of traffic is characteristic of an ACK-based Denial-of-Service (DoS) attack. Unlike SYN floods that disrupt the handshake process, ACK floods exploit established or mid-handshake connections, consuming memory and CPU resources on the target by overwhelming the connection tracking tables and exhausting processing capacity. The purpose of this flood appeared to be degradation rather than destruction, intended to slow down the system's responsiveness, reduce the efficacy of detection mechanisms, and possibly suppress logging output, making subsequent actions harder to trace in real time. Once the flood had weakened the target's ability to maintain normal operations, the attacker shifted to the infiltration phase, which involved the use of TCP Reset (RST) packets. These RSTs were injected mid-session, often following a SYN or partial handshake, which forcibly terminated ongoing or forming TCP connections. This tactic is typically employed to evade IDS logging, disrupt stateful inspection tools, or avoid creating complete session records in traffic monitoring solutions. The use of RSTs here indicated that the attacker was probing the system's ports, particularly port 445, which is commonly associated with SMB (Server Message Block) services on Windows environments, without leaving behind clear connection footprints. These probes were not random. They were focused and timed, suggesting a deliberate effort to fly under the radar by manipulating how packet logs are generated and processed. Following the stealthy network probing, the attacker began a targeted enumeration of SMB services. Analysis of the .pcap file in Wireshark revealed repeated SMB-based interactions involving the DCE/RPC protocol and a focus on the Print Spooler subsystem. Specifically, the attacker initiated multiple calls to the EnumPrinters function, which is used to remotely list available printers on a host. These calls were tunneled through SMB sessions and directed at the \pipe\spoolss named pipe, a critical IPC mechanism used by Windows Print Spooler services. The presence of these calls strongly suggested an attempt to either enumerate installed printers for further targeting or exploit known vulnerabilities within the spooler, such as unauthenticated remote code execution or privilege escalation techniques. Such attacks are especially relevant in legacy systems like Windows XP or 2000, where unpatched services are common and spooler hardening is often lacking. At this point in the analysis, tools such as SNORT were used to further validate observed behavior. SNORT generated multiple alerts related to abnormal TCP segment sizes, TCP port scans, and excessive SMB-related activity, including warnings for potential reconnaissance techniques. These alerts confirmed that the activity was not benign. In NetworkMiner, the attacker's host was fingerprinted with high confidence as running Debian Linux, and its associated MAC address was tied to a VMware adapter, further supporting the conclusion that the attacker was operating from a virtualized environment. Notably, NetworkMiner also detected the presence of Ettercap on the attacker's host, a powerful tool often used for man-in-the-middle attacks, credential harvesting, and ARP

spoofing, indicating that the attacker was equipped to escalate access if the intrusion succeeded. Meanwhile, the target system, APLOVERS, was confirmed to be a Windows XP or 2000 host, and exhibited all the hallmarks of an unprotected legacy endpoint. It accepted SMBv1 communications, used legacy dialects like NT LM 0.12, and had named pipes such as spoolss exposed without apparent filtering or access control. These conditions significantly increased the system's attack surface. Importantly, when the Windows Event Viewer logs were examined, there were no events recorded during the time of the attack, which spanned from July 8, 2017, 3:09:32 to 3:13:45 UTC, as recorded in the .pcap file metadata. This gap suggested that the system either lacked a functioning time synchronization protocol like NTP, or that logs were being suppressed or not properly configured. The absence of logging during the critical window of attack significantly impeded correlation efforts and would have severely delayed detection in a real-world scenario. To overcome this challenge, Event Viewer logs were searched for the term "spool", based on the SMB spooler activity seen in the capture. This search led to a discovery of a Print Spooler stop event at a much later time, which, while not timestamp-aligned due to no time protocol, demonstrated that spooler activity was logged under normal conditions. This confirmed that the service was indeed exposed and potentially vulnerable at the time of the attack, even if internal system logs failed to capture the exact intrusion window. In summary, the attacker followed a deliberate, multi-stage approach: beginning with a TCP ACK flood to exhaust the target's resources, then moving to TCP resets to stealthily probe for weaknesses, and finally launching a targeted enumeration of the Print Spooler service via SMB, likely as a stepping stone to deeper exploitation. The use of ephemeral source ports, printer enumeration functions, named pipe communication, and Ettercap all point to a well-equipped and methodical adversary. The victim, lacking proper logging and time synchronization, was not only vulnerable in terms of software but also from a visibility standpoint, making it an ideal target for this type of layered attack strategy.

## Conclusion

This lab served as a meaningful exercise in understanding the complexities and responsibilities that come with cybersecurity analysis and incident response. It emphasized the importance of critical thinking, attention to detail, and a methodical approach when investigating potential security breaches. In the broader context of the cybersecurity community, labs like this are essential for developing the next generation of analysts who are capable of identifying threats, understanding attacker behavior, and preserving the integrity of digital environments. As cyber threats continue to evolve in both scale and sophistication, the ability to dissect and respond to incidents becomes a crucial skill not only for individuals but for organizations at large. This lab reinforced the idea that proactive investigation, proper logging practices, and layered analysis are vital components in maintaining a secure and resilient infrastructure. By participating in this exercise, I gained practical experience that aligns with the real-world challenges faced by security professionals and deepened my understanding of the importance of collaboration, tool integration, and continuous learning in today's ever-changing digital landscape.

**Bibliography**

Beschokov, M. (2025, April 5). *What is TCP Reset Attack (RST)?*. RSS. https://www.wallarm.com/what/what-is-syn-spoofing-or-tcp-reset-attack

BetaFred, msmbaldwin, Justinha, wingtofree, & mdressman. (1998, November 19). *Microsoft Security bulletin MS98-017 - important*. Microsoft Learn. https://learn.microsoft.com/en-us/security-updates/securitybulletins/1998/ms98-017

Chatgpt. (n.d.-a). https://chatgpt.com/

Cisco. (2023, December 14). *Snort 3 Inspector Reference - DCE SMB Inspector [Cisco Secure Firewall Management Center]*. Cisco. https://www.cisco.com/c/en/us/td/docs/security/secure-firewall/snort3-inspectors/snort-3-inspector-reference/dce-smb-inspector.html

*Discover complex attack patterns with event log correlation*. Event Correlation Software, Correlation Rules, Correlate Events. (n.d.). https://www.manageengine.com/products/eventlog/event-correlation.html

Ettercap Home Page. (n.d.). https://www.ettercap-project.org/

Haber, M. J. (2019, March 15). *The implications of Network Time Protocol (NTP) for… | beyondtrust*. BeyondTrust. https://www.beyondtrust.com/blog/entry/the-implications-of-network-time-protocol-ntp-for-cybersecurity

Munoz, J. (n.d.). personal.

N16H7H4WK. (2023, July 16). *SMB enumeration-guide:*. Medium. https://medium.com/@taliyabilal765/smb-enumeration-guide-b2cb5cfb20e6

Pingle, B., Mairaj, A., & Javaid, A. (2018, May 5). *IEEE Xplore*. Real-World Man-in-the-Middle (MITM) Attack Implementation Using Open Source Tools for Instructional Use. https://ieeexplore.ieee.org/Xplore/home.jsp

*Printnightmare, critical Windows Print spooler vulnerability: CISA*. Cybersecurity and Infrastructure Security Agency CISA. (2021, July 2). https://www.cisa.gov/news-events/alerts/2021/06/30/printnightmare-critical-windows-print-spooler-vulnerability

Shavers, B. (2008, November). *A discussion of virtual machines related to forensics analysis*. Forensic Focus. https://www.forensicfocus.com/articles/a-discussion-of-virtual-machines-related-to-forensics-analysis/

*Sourcefire Snort DCE/RPC preprocessor buffer overflow: CISA*. Cybersecurity and Infrastructure Security Agency CISA. (2007, February 19). https://www.cisa.gov/news-events/alerts/2007/02/19/sourcefire-snort-dcerpc-preprocessor-buffer-overflow

What is an ACK flood ddos attack? | types of ddos attacks | cloudflare. (n.d.-b). https://www.cloudflare.com/learning/ddos/what-is-an-ack-flood/