Elias Pappas

Exam Project 2 - FI2AMP275

Creating a Home SOC Lab

Approved by Noroff Faculty

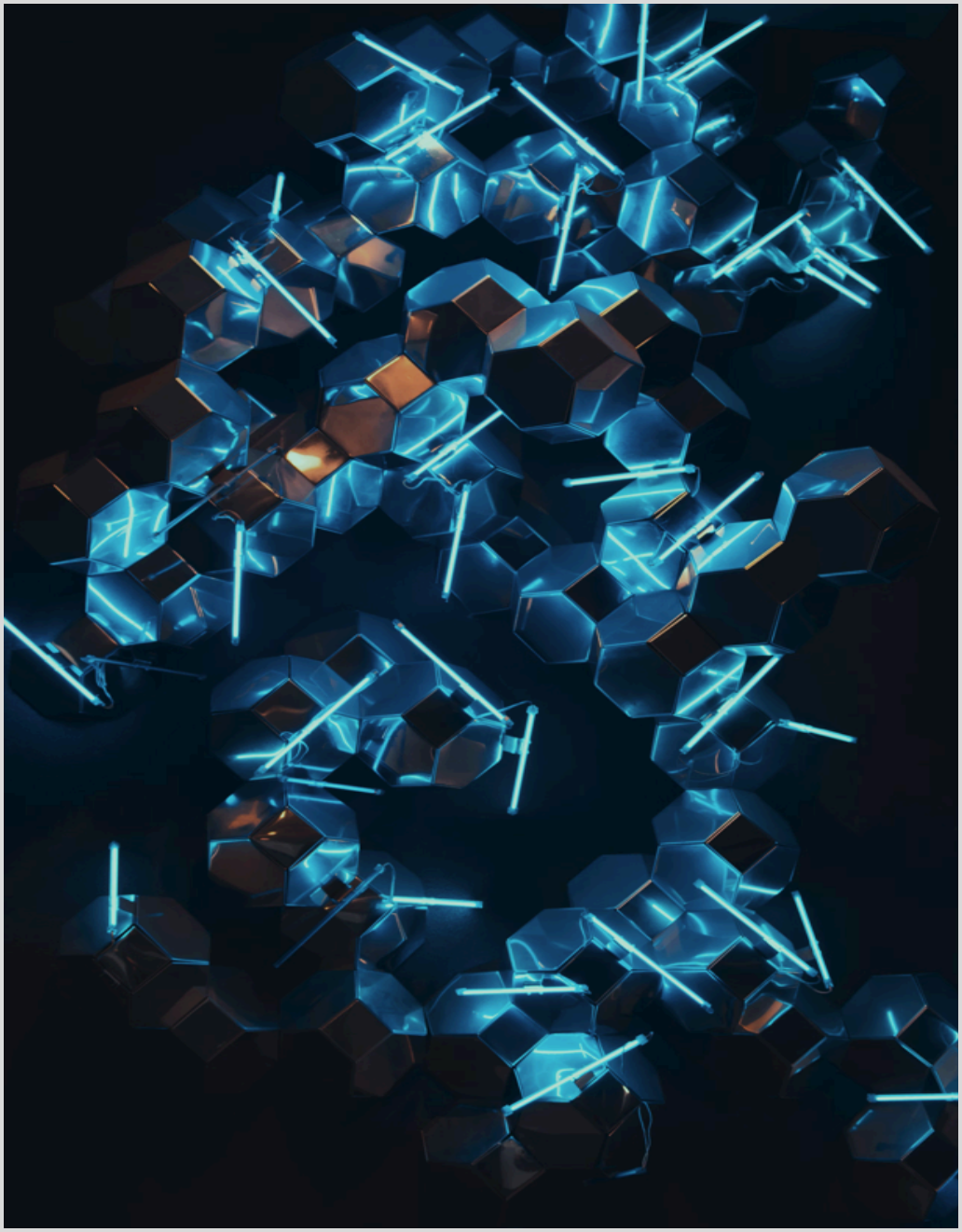
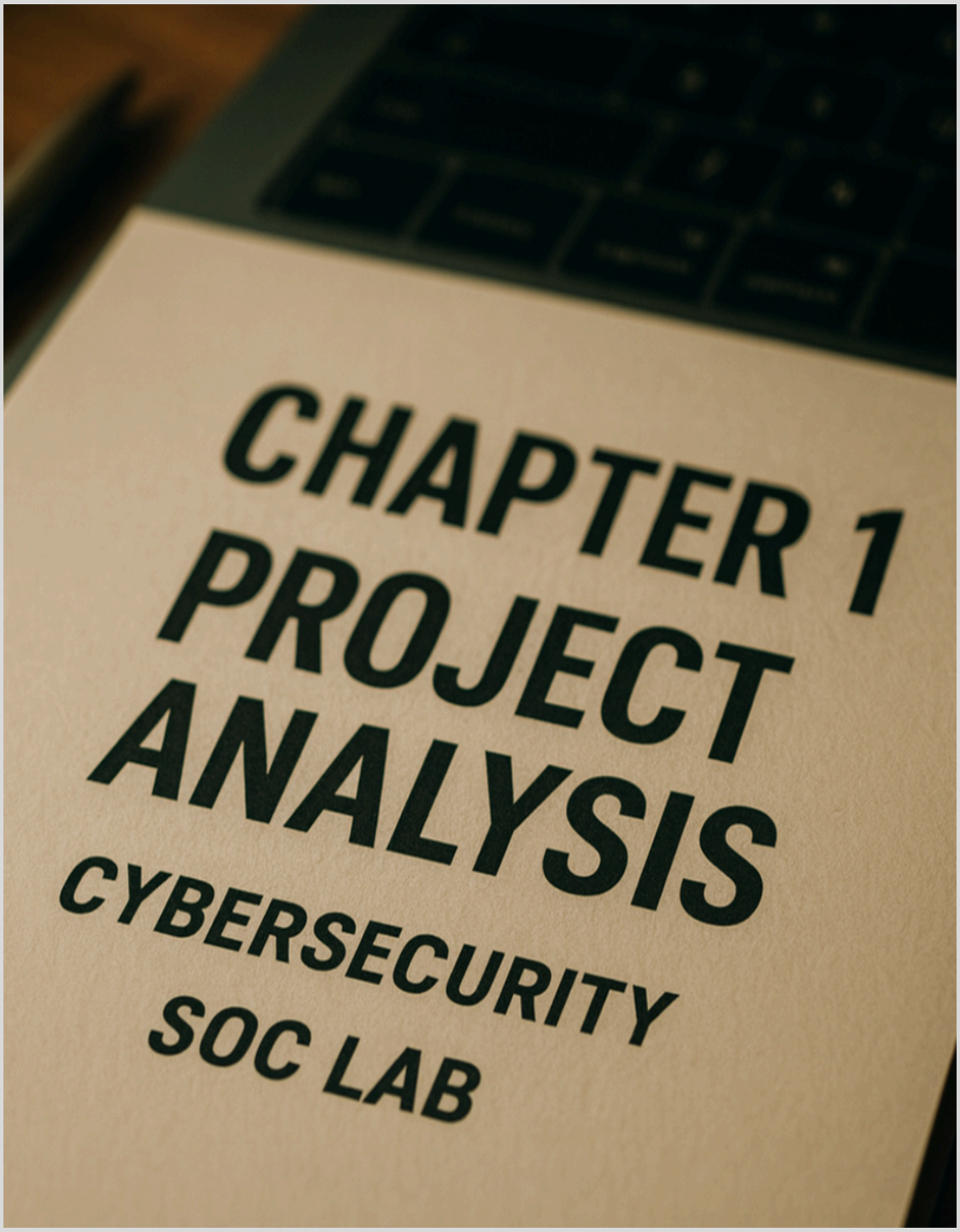


Image by Maximalfocus, Unsplash, 2020

Table of Contents


CHAPTER 1
PROJECT
ANALYSIS
CYBERSECURITY
SOC LAB

1.1 Introduction

I have chosen this project because I believe it combines most of the material covered in the two years of Network and IT Security at Noroff, while also providing much needed hands-on experience that is highly valued by employers.

The lab covers topics such as Virtual Machines, Domains, Networking, Security Fundamentals, Hardening, Ethical Hacking, and Monitoring and Detection of Security Incidents.

It also introduces Splunk - arguably the most widely used SIEM today - which I have been learning as part of my ongoing education, and which serves here as the main platform for log analysis and correlation.

The general design of the lab is a puzzle built by the pieces of the following sources, which I would like to reference here holistically due to them being used as foundational guidelines:

- LetsDefend: Building a SOC Lab at Home (LetsDefend, 2025)
- TryHackMe SOC Level 1 Path (TryHackMe, 2025)
- SOC Analyst - Active Directory & Splunk Home Lab (Leka, 2024)
- Building a Cybersecurity Homelab for Detection & Monitoring (Day Cyberwox, 2021)
- Detection Lab (Long, 2022)
- STEP | Splunk Training and Enablement Platform (Splunk, 2025)
- SOC Home Land and Active Directory (Hebballe, 2024)

Unlike the above sources though, the practical configurations used in building this lab will be properly cited and referenced throughout the work where appropriate.

In addition, to better understand and complete the required structure and tasks within the limits of my hardware and the project's educational scope, I have also relied on many community resources from time to time, which include internet forums, social platforms, Google "rabbit holes", and youtube videos.

These sources were explored throughout the course of my studies for personal training, and because of their random selection and personal nature, have not been formally cited, except in cases where they are directly related to a task.

To improve reading comprehension, the most detailed configurations have been placed in their own respective chapter, titled *Configurations*. Similarly, a separate chapter titled *Problems and Solutions* has been created for the exact same reason.

These chapters are not included in this project due to word count limitations, but they comprise a separate PDF named *Appendixes*, which can be provided if requested.

As it stands, configurations and solutions are properly linked during this project as simple pop-up images, stored in a Google Drive folder that will remain available for access until a final grade is awarded.

Entry-level configurations (e.g., basic installations) are not covered in this project. Configurations based on default or official industry guides have been appropriately linked to their sources but are only expanded upon when technical considerations are required (for example, in the case of Splunk).

This project has been researched, developed, abandoned, and restarted over an extended period of time. Beyond serving as a demonstration of practical skills, it also aims to be a beginner-friendly resource for others who wish to gain practical fundamentals relevant to an entry-level SOC Analyst role.

During my two-year education at Noroff, I have often used the method of creating guides and documenting troubleshooting as a way to better understand and absorb the material. I applied the same approach here.

With the permission of Noroff, this project will be uploaded on my Github page after it has been graded.

1.2 Lab Objectives

This project aims to demonstrate the design and execution of a small-scale Security Operations Center (SOC) lab using **Splunk Enterprise** as the main SIEM tool.

The main objective is to build the entire lab from scratch, and then simulate how specific attack scenarios can be detected and handled in Splunk. In addition, by configuring **Sysmon**, the lab aims to illustrate how security telemetry is needed to enhance incident detection and analysis.

The ultimate goal is to complete a working SOC Lab, which can also be used for further training and education outside of this project.

Sysmon-splunk, Prajwal V., 2025

The decision to center this project around Splunk was deliberate. Splunk is widely used in enterprise SOC environments (Splunk, n.d.), while offering a free learning platform for students at the same time. I am also currently working towards my first Splunk certification, so it made sense to use it in the lab's core.

Unlike many tools that require advanced scripting or complex configurations from the start, Splunk provides a graphical interface (alongside its search language, SPL), making it quite approachable in my humble opinion.

Another key reason for using Splunk was its portfolio value. Employers in the cybersecurity field usually look for candidates who can show that they know their way around a SOC, where a SIEM like Splunk usually sits at the centre.

By building a lab that simulates real attack techniques and then showing how Splunk deals with alerts and uses Dashboards for incident detection and classification, this project will hopefully illustrate valuable hands-on skills.

1.3 Lab Scope

The environment in the lab is limited to four virtual machines, which are enough to create a convincing simulation without crippling a moderately-specced host. The goal is to simulate and detect four key attack techniques that are frequently encountered in today's landscape.

These are:

- **Reconnaissance:** Before launching attacks, adversaries typically scan networks to identify open ports and services (Cymulate, 2025). We will be using **nmap** on the Kali machine to scan both the Windows Server and the Windows Client, and then demonstrate how Splunk detects these scans.

- **Password spray:** This is a Brute Force technique where attackers attempt one password (or a limited number of them) across many accounts, and it is particularly effective against targets that use password sharing (Crowdstrike, 2022). We will use Splunk to correlate authentication logs and highlight failed login attempts, followed by successful ones.

- **Malware execution (EICAR test file):** The EICAR test file is a string that will be executed on the Windows Client to simulate real malware activity (Eicar, n.d.). This generates alerts in Windows Defender and Sysmon, which are ingested by Splunk for detection and analysis.

- **Phishing Triage:** Phishing is one the most common entry points for attackers, and will be simulated by a user of our lab receiving a phishing email with a link, and instructions on how to download a malicious file. Phishing is a technique where an attacker masquerades as a trusted source to steal sensitive information (CloudFlare, 2025), and in our lab we will capture the entire chain in Splunk.

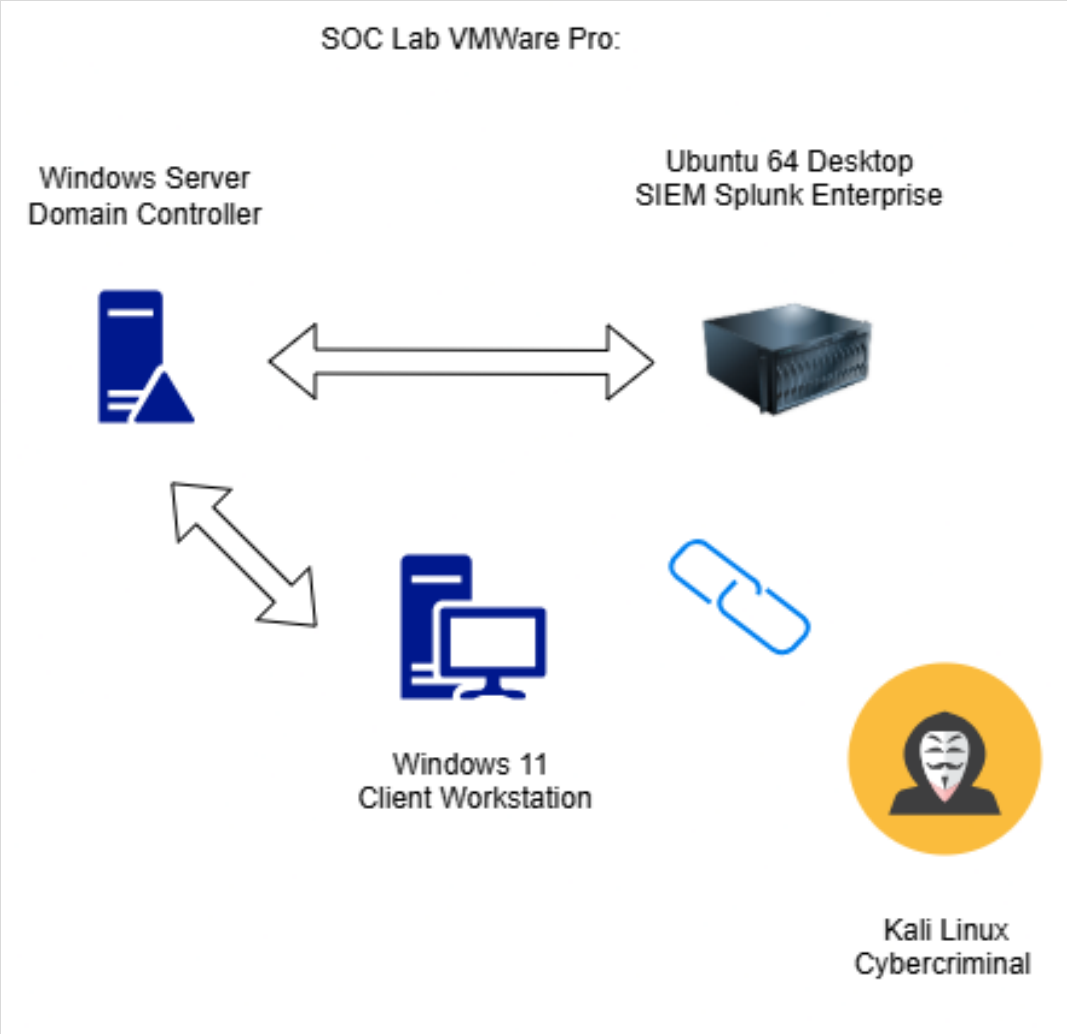
These scenarios fit the tasks SOC analysts are usually expected to perform: log analysis, alert review, and initial incident response.

1.4 Lab Topology

As mentioned earlier, the lab environment is built on four virtual machines running inside VMware Workstation Pro (non-commercial license), with each having a specific role:

- **Windows Server 2022 - Domain Controller:** This VM simulates an enterprise domain controller running Active Directory and DNS. It provides authentication services and the main logs for monitoring via Sysmon.
- **Windows 11 Client - Victim:** This VM will be our end-user workstation, which will be the target for our attack scenarios. Sysmon will also be configured here to allow monitoring events at the user level.
- **Ubuntu 64 Desktop - Splunk Server:** This VM will act as our SIEM platform. Splunk will be installed and configured to collect logs from the Windows hosts and process them by using SPL. They will be monitored and analysed by the various tools Splunk offers, like dashboards and alerts.
- **Kali Linux - Attacker:**This VM will act as our attacker. Kali will be used to launch all our attack scenarios, since it comes with all the tools we need pre-installed.

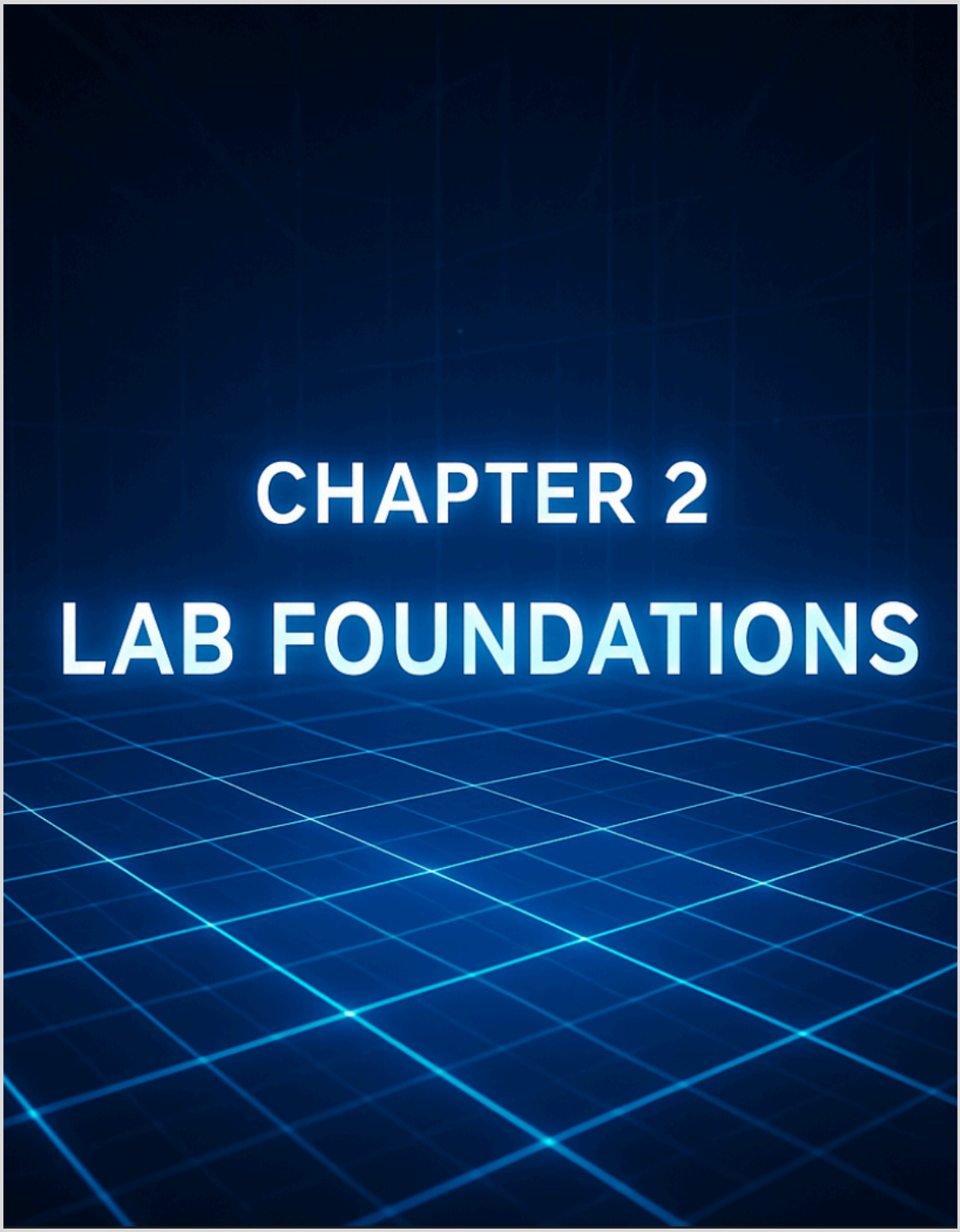
This is our topology:



Design created in draw.io

In the next chapter we will start preparing our VMs for deployment by describing specifications, network configurations and hardening design.

🥷


CHAPTER 2
LAB FOUNDATIONS

2.1 Host Prep & Resources

In order to build and host the virtual lab, the physical host machine must be prepared to ensure stability and smooth operation of all lab workstations.

As already mentioned, the host must have VMWare Workstation Pro installed (there is a non-commercial license) and should have at least 16 GB of RAM, a quad-core CPU or higher, and 150 GB+ of free disk space.

I have used the following splits in VMs during my studies with little to no issue, so I will use them again for this project (for anyone reading the guide feel free to add more resources if your system allows it):

- Ubuntu Splunk Server: 4 GB RAM, 2 vCPUs
- Windows Server (Domain Controller): 2-3 GB RAM, 2 vCPUs
- Windows Client (Victim): 2 GB RAM, 2 vCPUs
- Kali Linux (Attacker): 2 GB RAM, 1–2 vCPUs

ISO files for Windows Server and Windows 11 can be downloaded freely for a limited time, while Ubuntu and Kali Linux can be found here and here respectively. All VMs must be updated **before** proceeding with other installations or configurations, and it is well advised to take snapshots during the course of the project (keep up to 3 for each VM, otherwise you may experience low performance).

2.2 Network Management and Hardening

The lab design requires the VMs to communicate in a controlled way that simulates a corporate environment, which means they can't be exposed to the wider internet without a secure gateway.

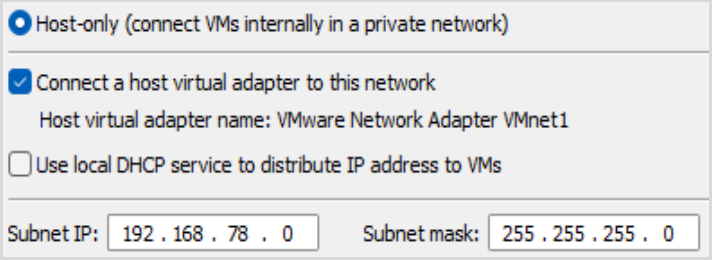
To simulate this, the lab network will have no access to the internet. The Windows Server will act as the local DNS Provider, and will also play the role of a simulated gateway. A Virtual Interface is attached to the host to run the web interface of Splunk.

To keep the project focused on real-time threat monitoring, the attacker in the lab will not perform network breaches; full exploitation is outside the scope of this project and usually belongs to Ethical Hacking exercises.

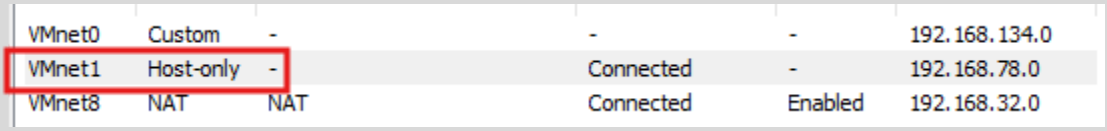
However, some ethical-hacking techniques will be demonstrated where they help illustrate how different telemetry sources correlate within the lab environment.

Network Configurations

To set up our network we start by picking a new *Host-only* network in the Virtual Network Configuration of VMWare Pro, and set our subnet's address. We also tick *Connect a host virtual adapter*.



Then we proceed by placing every VM on that subnet, in this case VMNet1:



After all VMs have been assigned to VMNet1, let's assign static IP addresses to them so they can communicate.

I decided to follow a classic IP set up for ease of use (it is not necessary to follow the same pattern):

- Windows Host (Virtual Interface): 192.168.78.1/24
- Windows Server: 192.168.78.10/24
- Windows Client: 192.168.78.20/24, DNS: 192.168.78.10
- Ubuntu Splunk Server: 192.168.78.30/24, DNS: 192.168.78.10
- Kali: 192.168.78.40/24, DNS: 192.168.78.10

📒 ***- Your own IPs/Names may be different.***

🚦***Reminder: Every time a task that requires internet appears, switch the VMs to NAT (VMNet8 in VMWare Pro), complete the task, and then switch back to our Host-only network, VMNet 1.***

Hardening

We do not need to simulate enterprise-level hardening in this lab. A best-practices hardening for a lab like ours can be comprised of the following:

Windows VMs:

- Network profile in 'Private' so Firewall knows its a trusted network (Plugable, 2024):
- Strong password policy for the Domain.
- One main administrator.
- Client accounts will be local accounts with no other privileges.
- Firewall only allows access from our network
- Windows Defender default state.
- Basic Services like printing, bluetooth, wi-fi will be disabled (just disable them in computer settings).
- Only default ports will be up.

Ubuntu & Kali:

- UFW will be activated (like we did with Private Profile, to simulate full connections in our private network), and two firewall rules will allow connection only from machines in our lab network.
- Basic services like CUPS and Bluetooth will be disabled.
- Kali Linux will be left as is since it is the attack machine.

📒 *- See Configuration 1.1 for the Windows VMs*
📒 *- See Configuration 1.2 for the Linux VMs.*

Problem (solved) During the setup of Kali and Splunk main interfaces were assigned a double IPv4 address due to the automatic activation of local DHCP from VMWare.

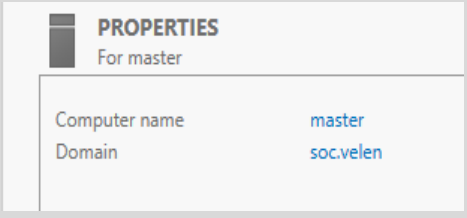
📒 *- Refer to Solution 1: DHCP & Double IPs Solution*

2.3 Domain Configuration - Window VMs

Windows Server
The Windows Server VM will function as the backbone of the lab and will be configured as the Domain Controller (referred to as DC from now on) with Active Directory Domain Services (AD DS) and DNS roles.

These are well-documented procedures, so analytical configurations will not be offered here. We simply follow the guide in Microsoft Learn, *GUI option*, (Microsoft, 2023).

We will be asked to create a new forest and domain:



An Administrator, A Privileged User and four standard user accounts are created in the DC for our Client. As before, we follow the normal guide in Microsoft Learn (Orin-Thomas, 2025).

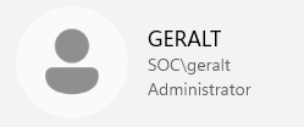
Name	Type	Description
Geralt	User	Domain Administrator

Name	Type	Description
Vesemir	User	Priviliged User

Name	Type	Description
Dandelion	User	Normal User
Triss	User	Normal User
Yennefer	User	Normal User
Ciri	User	Normal User

At this stage, the server provides a central point to control our domain and domain users, and we will use it to generate events and collect logs.

Following best practices, Geralt will be our main administrator from now on instead of the built-in Administrator. We test by logging in on our DC:



In our next step, we are going to join the Windows Client VM to our domain: **soc.velen.**

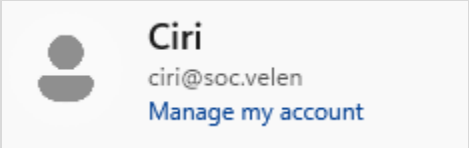
Windows Client

The Windows Client VM represents the everyday end-user workstation in our lab. It will be our primary target for the attacks.

We proceed by joining our client, *WClient1,* to our domain, *soc-velen.* We follow this simple guide in Microsoft Learn, *Control Panel option*, (Xelu86, 2025). We verify that the join-in is successful. We can see that it is:

Full computer name:	WClient1.soc.velen
Domain:	soc.velen

We then test the accounts we created with our Domain Controller in our previous section. We login with each of the accounts. For example, Ciri:



After setting up everything, here is a summary of our domain users:

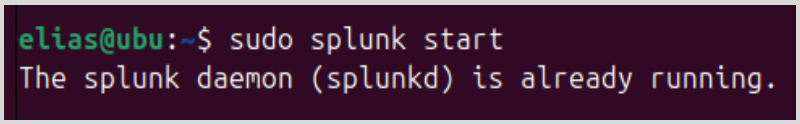
1. **Geralt**: Domain Admin.
2. **Vesemir**: Privileged User
3. **Ciri**: Domain User.
4. **Yennefer**: Domain Use.
5. **Triss**: Domain User.
6. **Dandelion**: Domain User.

2.4 Ubuntu Splunk Enterprise Installation

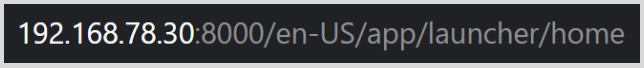
The Ubuntu VM will be our Splunk Enterprise server, acting as the SOC's central logging and analysis tool. We have already assigned a static IP to Ubuntu in our previous section, *192.168.78.30*.

We will now proceed and install Splunk Enterprise by downloading it from here and create an administrator:

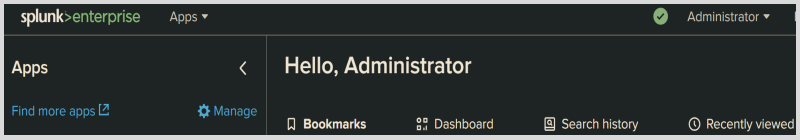
Service running test:



Port working test:



Connecting:



📒 ***- See Config 2: Splunk Initial Config.***

💡***- By using a host-only network setup, our host has a presence in the subnet with a Virtual Interface. That means that we can run Splunk in Ubuntu, but for better performance we can use the host to control it by visiting the default port of Splunk's web interface (Group, 2024): 192.168.78.30:8000***

🥷

The Kali Linux VM and its integrated toolset will serve as the dedicated attacker machine for the lab. Its purpose is to simulate real-world adversary techniques like reconnaissance, password spraying, phishing, and malware execution.

We have already assigned it the static IP 192.168.78.40. We have also configured its firewall.

There is nothing else to do in Kali other than making sure it is fully updated.

2.5 LAB Configuration Summary:

Windows Server

- Description: Domain Controller of soc.velen (AD DS + DNS) – 192.168.78.10
- Users:
    - Geralt — Domain Admin
    - Vesemir — Privileged User

Windows Client (Victim)

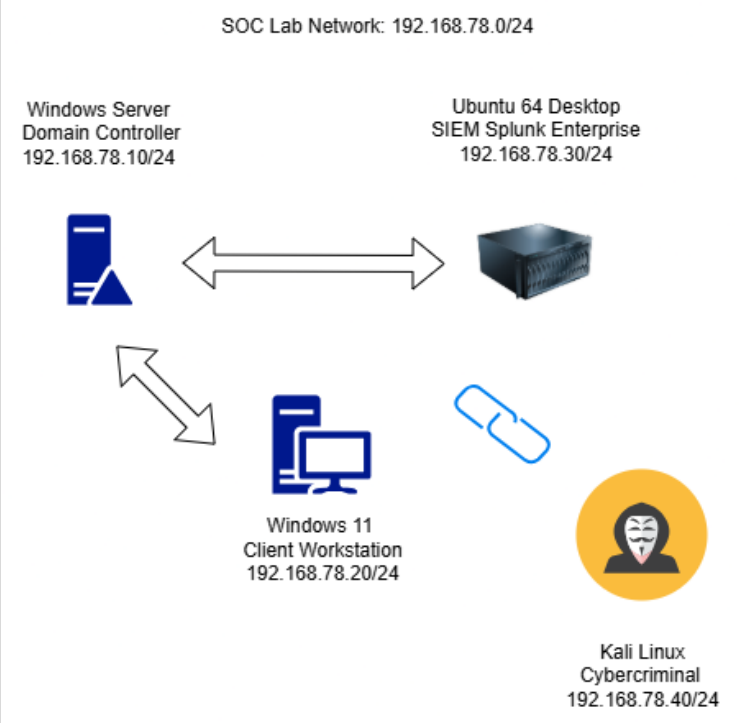
- Description: Victim workstation joined to domain soc.velen – 192.168.78.20
- Users:
    - Ciri
    - Yennefer
    - Triss
    - Dandelion

Ubuntu (Splunk)

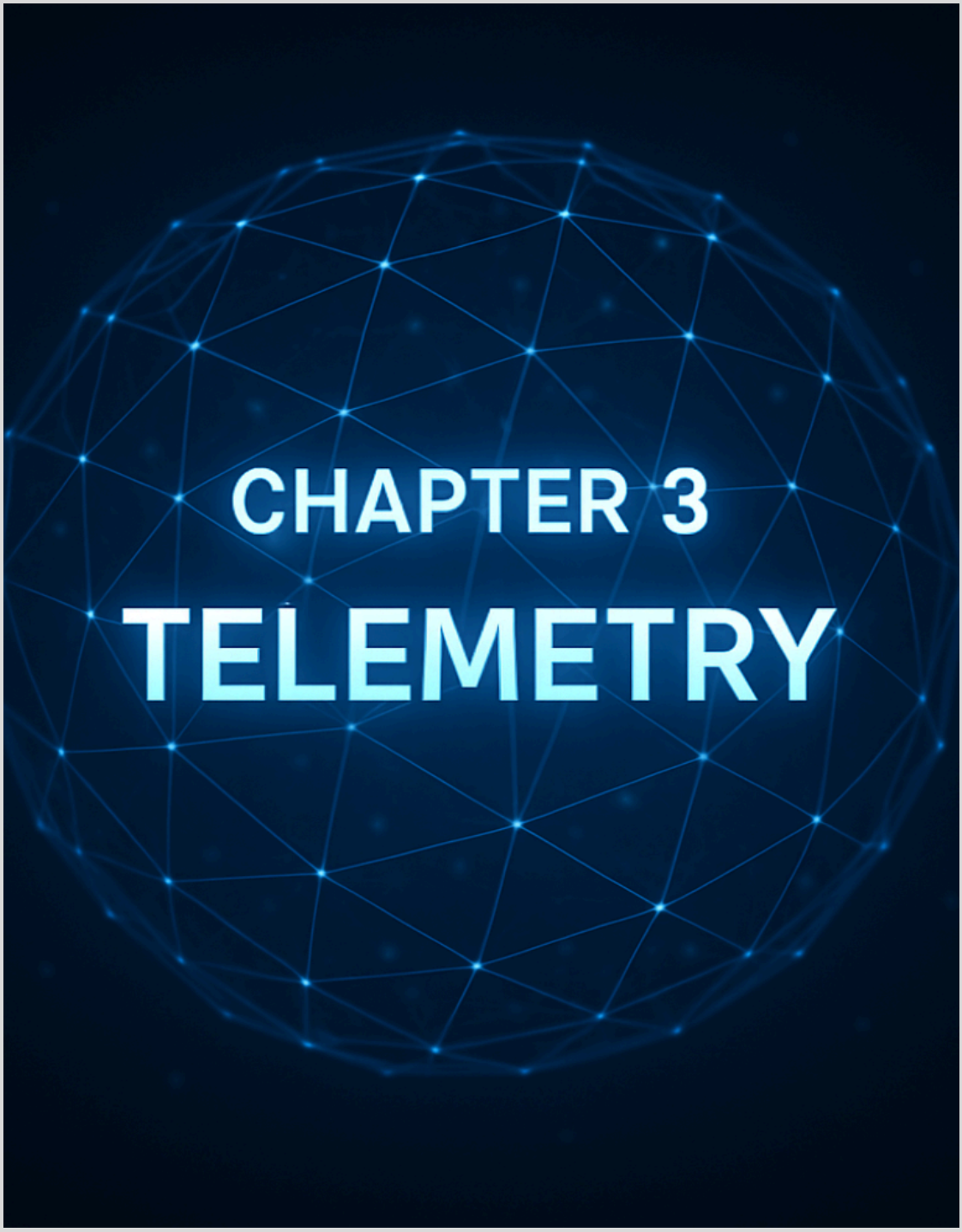
- Description: SIEM server running Splunk Enterprise – 192.168.78.30
- User:
    - Administrator

Kali Linux (Attacker)

- Description: Attack machine for executing reconnaissance, phishing, malware, and password spray – 192.168.78.40
- User:
    - Default User



Design created in draw.io


CHAPTER 3
TELEMETRY

3.1 Windows Auditing

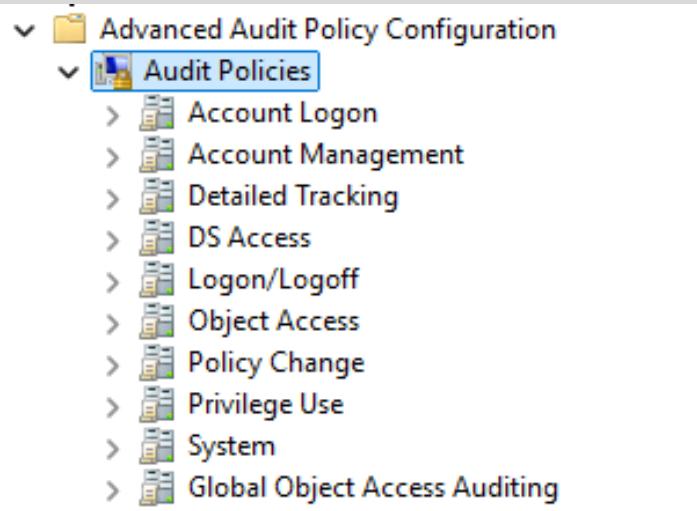
In the previous chapter we set up the foundation of our lab, our domain network, and created our users. For the SOC lab to actually work though, it needs to provide us with proper monitoring. In other words, we need to be able to log in what events take place in our VMs.

To do that, we will enable advanced auditing in our domain's Audit Policies, by pinpointing which elements we need to audit for our lab. Then we must edit the policies **on both the DC and the Client**, in their respective *Group Policy Management Console* (Roberts, 2025).

The reason we are focusing on both is because, according to Microsoft Learn *"Focusing solely on servers or domain controllers is a common oversight, as initial signs of malicious activity often appear on workstations"* (Xelu86, 2025).

So, from the *Group Policy Management* in our DC, and our Local Policy Editor in our Client, we follow this path to reach *Advanced Audit Policy Configuration*:

computer configuration/windows settings/security settings/advanced audit policy configuration



As we can see, there are many categories in our Audit Policies, and each category has multiple sub-categories. We need to enable the sub-categories we need for our attack scenarios (Netwrix, n.d.) (Splunk, 2025):

- Password Spray
- Phishing
- Malware
- Reconnaissance (passive/active)

We will therefore be enabling the following sub-categories for both *Success & Failure telemetry*:

- Account Logon. This means that each time the DC is asked to verify a password for a domain login, a record is created. Useful for our password spray attack.

Subcategory	Audit Events
Audit Credential Validation	Success and Failure
Audit Kerberos Authentication Service	Success and Failure
Audit Kerberos Service Ticket Operations	Success and Failure
Audit Other Account Logon Events	Success and Failure

- Logon/Logoff: This monitors who actually logged in a VM. Useful to see which user, for example, was the victim of a phishing attack or executed malware.

Subcategory	Audit Events
Audit Logon	Success and Failure
Audit Logoff	Success and Failure
Audit Account Lockout	Success and Failure
Audit Special Logon	Success and Failure

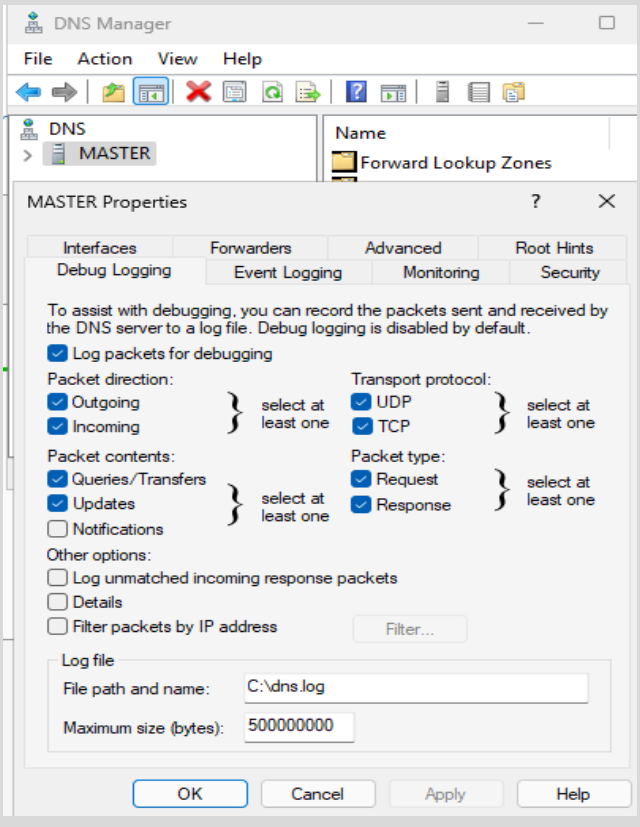
- Object Access: This helps us monitor registry, files, and folders.

Subcategory	Audit Events
Audit Registry	Success and Failure
Audit File System	Success and Failure

- Privilege Use: To monitor any actions by users with privileged rights.

Audit Sensitive Privilege Use	Success and Failure

We also activate DNS logging for our Domain Controller to capture reconnaissance attempts. This can be done by going into *DNS Manager,* and then into *Properties* where we can enable *DNS Debug Logging* (robinharwood, 2025).



This will allow our DC to log any DNS activity, Incoming or Outgoing, and save it on a file in a place of our choosing.

Now, With basic Windows Auditing and DNS Logging out of the way, it is now time to go deeper.

Which means we need to install Sysmon.

3.2 Sysmon on Windows

Everything we just set up is a good way to monitor what actually happened in a system, but it is not enough to tell us how or why. We need an extra tool that will help us correlate that information by providing a more detailed layer of monitoring. And Sysmon does exactly that.

It is a tool that provides details on all system activity, like network connections, processes, file changes, etc., revealing the reasons behind events that have occurred (markruss, 2024).

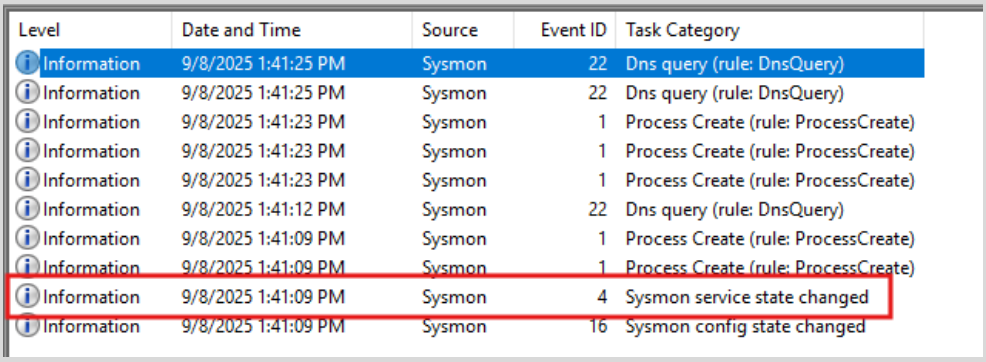
We download Sysmon from here and we install it in both our Domain Controller and our Client.

After downloading Sysmon, we also download a widely known community configuration (e.g., SwiftOnSecurity's Sysmon config), which is a great place to start setting up our configuration with little to no fuss.

After we download the configuration, we place it in the directory where Sysmon is also downloaded, and then we install it via Powershell (IBM, 2025):

.\Sysmon64.exe -accepteula -i sysmonconfig.xml

We check the event viewer in windows to make sure it has started. Indeed it has:



Level	Date and Time	Source	Event ID	Task Category
Information	9/8/2025 1:41:25 PM	Sysmon	22	Dns query (rule: DnsQuery)
Information	9/8/2025 1:41:25 PM	Sysmon	22	Dns query (rule: DnsQuery)
Information	9/8/2025 1:41:23 PM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	9/8/2025 1:41:23 PM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	9/8/2025 1:41:23 PM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	9/8/2025 1:41:12 PM	Sysmon	22	Dns query (rule: DnsQuery)
Information	9/8/2025 1:41:09 PM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	9/8/2025 1:41:09 PM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	9/8/2025 1:41:09 PM	Sysmon	4	Sysmon service state changed
Information	9/8/2025 1:41:09 PM	Sysmon	16	Sysmon config state changed

The config has automatically set the most common events we will monitor (markruss, 2024):

- **Event ID 1 - Process Creation**
  - Records every process, for example opening an email client or a browser. Can track a phishing attachment that has been clicked.
- **Event ID 3 - Network Connections**
  - Tracks details of connections, like reconnaissance attempts.
- **Event ID 11 - File Creation**
  - Tracks down creation of suspicious files. Can track the EICAR test malware for our lab.

🚦***Reminder: both the DC and Windows Client must be on NAT (VMNet8 - default) to have internet and download Sysmon. When the process ends, place them back on VMNet1 and redo their IPs and DNS.***

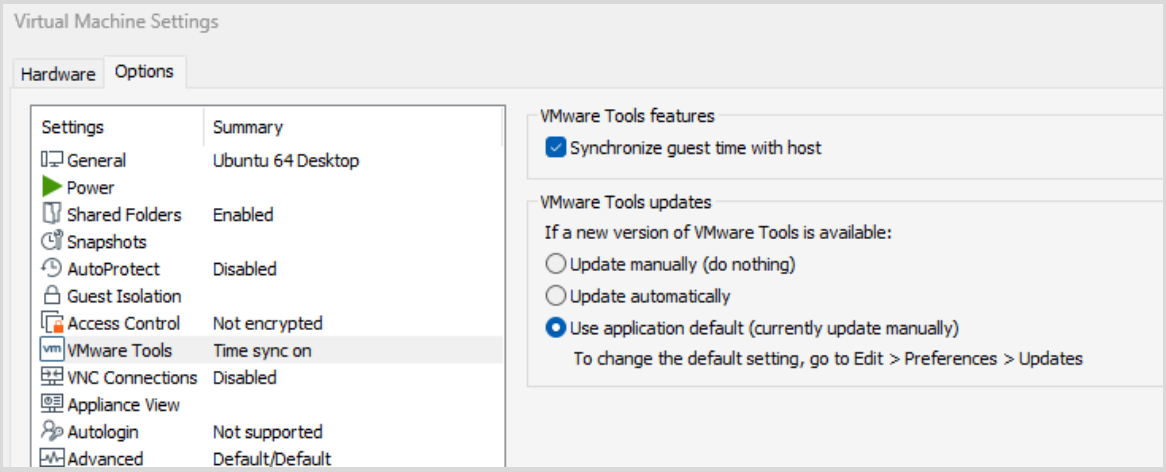
3.3 Time Sync and Sanity Check

Now, it is time for a sanity check on our lab. First, we need to make sure that all the machines we have set up are time-synced, otherwise telemetry cannot be forwarded to Splunk.

While researching time sync in Windows Domains, I discovered that in a multi-domain environment the domain workstations must sync with their DCs, and then the DCs must sync with a special DC with an active PDC Emulator FSMO role, which in turn must sync with an external reliable time source, for example an NTP Server (Kardashevsky, 2024).

Yeah, we are not doing that.

Our lab is not that big, plus it has no internet access anyway. Therefore the easiest solution is to turn on the *Time Sync* option in the settings of each VM in VMWare Pro. This option will allow the VMs to sync with the host machine.



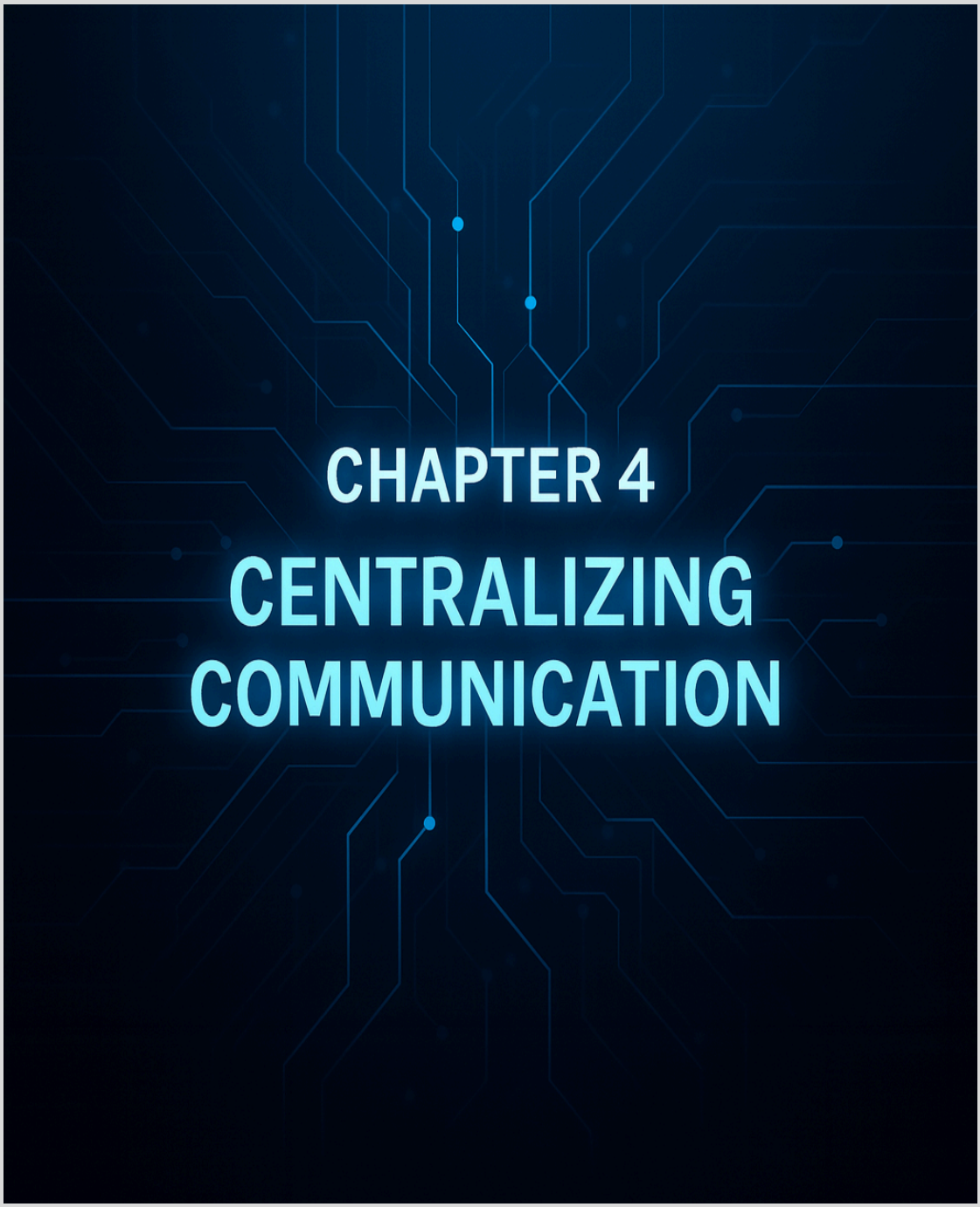
After setting the host time sync on, we can test it by changing the host machine's minutes and then check the clocks of the VMs, where the change will be reflected (there will be a slight delay).

Problem (Solved) When tested timesync of the hosts and VMs, the Linux VMs' clocks wouldn't sync back.

📒 - Refer to Solution 2: VM Machines Syncing Back Problem.

After we have time synced our VMs with the host, we must do a Sanity Check, and make sure everything is still connected and set up properly. Therefore we do a ping for all machines in the network (including the host) and make sure they can all communicate with each other.

In the next chapter we will configure the telemetry from Windows Firewall, Sysmon, and DNS logs to reach Splunk, creating a properly monitored lab.


CHAPTER 4
CENTRALIZING
COMMUNICATION

4.1 Universal Forwarders

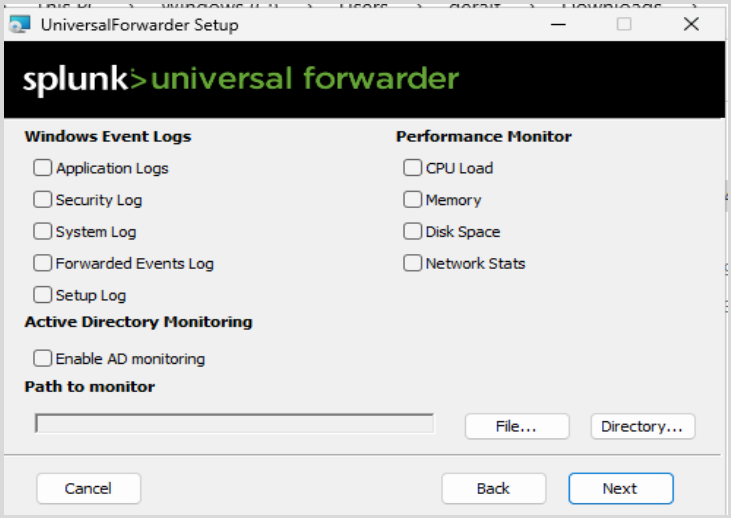
Now we have reached a point where we need to centralize our systems. Configuring telemetry is the first step, but if we do not forward it to our SIEM, it is all for nothing.

We do that by using the **Splunk Universal Forwarders (UFs).** These are little program agents that can be installed in almost any OS and can be configured to collect system data and forward it to Splunk (Group K., 2022).

Forwarders use extremely low resources and come with their own license out of the box, so no need for extra expenses (Splunk, 2025). For our lab, we should install them on the DC and our Windows Client.

After we download the Universal Forwarders, we configure them during installation with the options that suit our SOC lab (we must be very specific on what we choose so we do not overload our VMs).

For both machines we choose *Application Logs, Security Log, System Log,* and *Setup Log.* For our DC we also choose *AD Monitoring* (it is needed to collect more in-depth logs from our Active Directory).



We do not choose *Deployment Server* but *Receiving Indexes (in the screen that follows)***.** We type the IP of our Splunk Server, *192.168.78.30*, since this is where the Universal Forwarders will send data.

We use the default ports in both machines, *9997*, which is the port Splunk listens to when receiving data from the forwarders.

Lastly, we need to configure the *inputs.config* file, which essentially tells the UFs where to focus when looking for logs, and at which index of Splunk to send them to (Splunk, 2025b).

We will be configuring three sources to send data:

a) The logs from Sysmom.
b) The logs from our Windows Firewall.
c) The logs from our DNS.

📒 - Refer to Config 4: Universal Forwarders (Lame Creations, 2024) (Arun KL, 2024) (Labs and Labs, 2020) (Coil, 2023) .

💡- While checking the Splunk App Store and trying to find more about Splunk, Sysmon, and Windows Logs, I found three Splunk addons which are widely recommended for better visualization, parsing and categorization of our logs. I would suggest to install all three, as they are essential:

- ***Spunk add on for Microsoft Windows (Learn A Logic, 2019).***
- ***Splunk addon for Sysmon (Splunk, 2025b).***
- ***Splunk Common Information Model (CIM) (Presidio Splunk Solutions - formerly Kinney Group, 2024).***

Now, we have created our *inputs.conf* file for the UFs. There is also an *outputs.conf* that tells where the forwarding data is sent, but since we only have one Splunk instance, there is no need to configure another, so we leave the file on its default configuration.

For simulation reasons, we make sure that the Forwarders can pass through our firewall:

- *sudo ufw allow from 192.168.78.0/24 to any port 9997 proto tcp.*

At this moment, our lab should be complete and functional. Be aware that, as soon as the Forwarders are correctly set up, and Splunk and our VMs are up running, logging will start reaching our SIEM automatically.

🥷

To test that everything has been installed and configured correctly, we do a test and check our Sysmon EventCodes (webpro255, 2024):

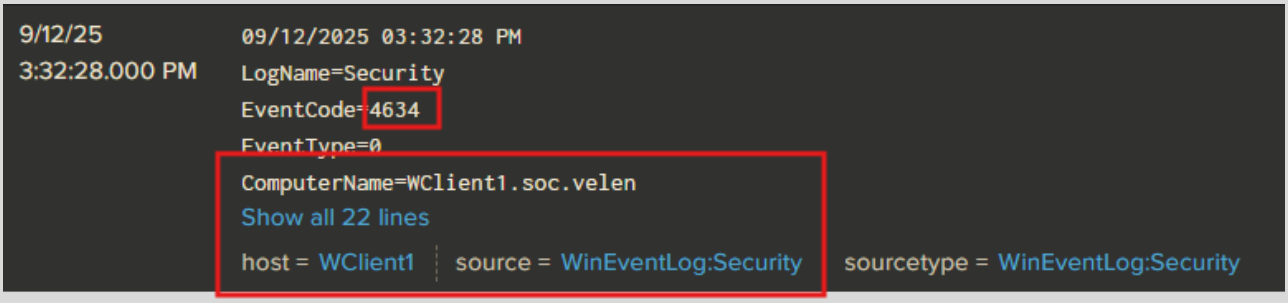
- we login and logout with our Ciri account (event ID 4634 in Sysmon)
- then we try to log in with a failed password (event ID 4625 in Sysmon)
- then we login with the correct one (event ID 4624 in Sysmon)

We capture these events in Splunk by running a very common and broad query (Prajwal V, 2025). This simply says *'look for any event with this code in our index' ('main' is the default index name of Splunk)*:

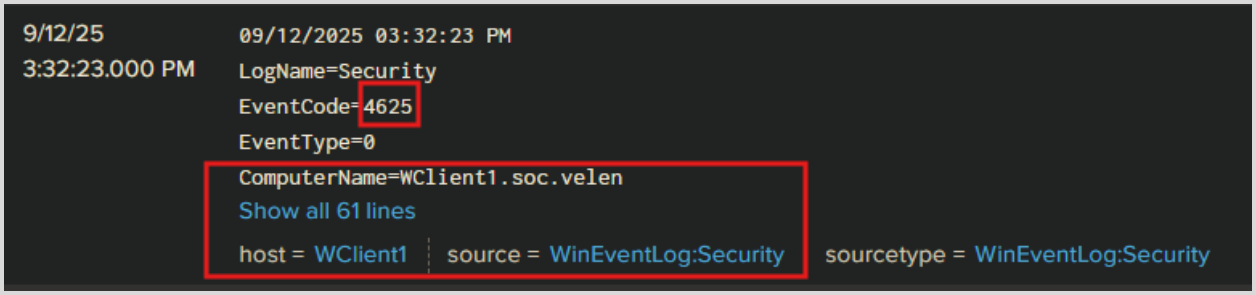
- *index=main EventCode=4624 or EventCode=4625 or EventCode=4634*

If our lab is properly connected and monitored, these events must be captured by Splunk, coming from our domain workstation, named *WClient*. Indeed, they have:

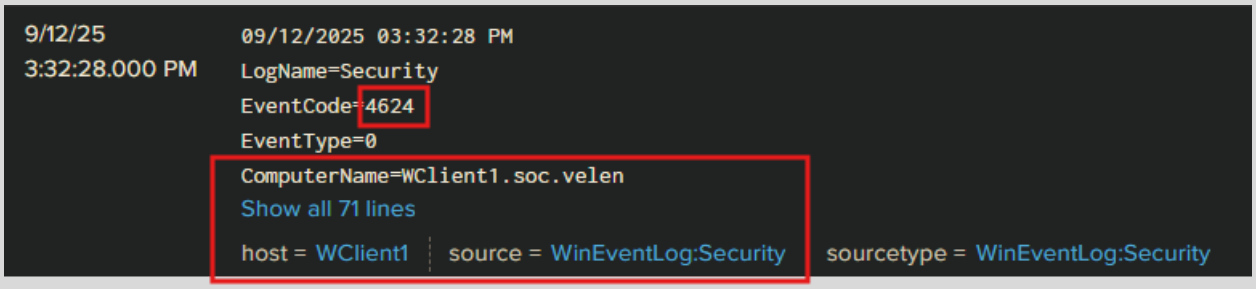
Our logout (4634):



Our failed login attempt (4625):



Our successful login (4624):



Our test is completed and it is now clear that our lab is monitored by Splunk.

4.2 Splunk Basics

Since this is an educational project, it may be a good idea to learn some of the basics of Splunk at the same time, as this will help us better understand how things work when we start hunting for our simulated attacks later on.

In our previous section we showcased the initially successful setup of our monitoring by running:

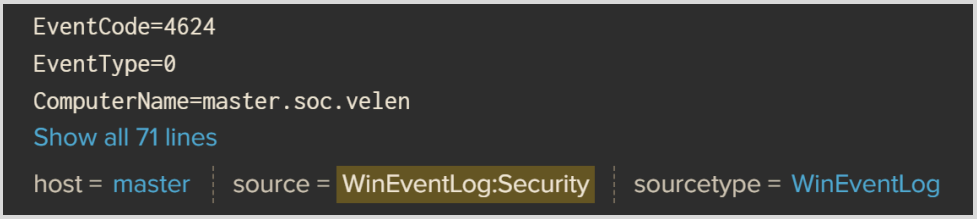
index=main EventCode=4624 #log in or EventCode=4625 #failed log in or EventCode=4634 #log off

This is an SPL (Search Processing Language) query, which is the beating heart of Splunk, and essentially instructs Splunk to retrieve and format data from the logs that it receives (Splunk, SPL, 2025).

We will now use the same events to hunt down for more specific results, building on the above search commands in order to understand SPL structure.

This time we will add a source, specifically from the Security logs, and tell it to look only there to track down failed logins on the DC. We can see it registered one in our *master.soc.velen* indeed:

index=main source="WinEventLog:Security" EventCode=4624



Building on that, we can add a specific account. Let's say that we want to check specific user logon activity, for example, Dandelion:

index=main source="WinEventLog:Security" EventCode=4625 Account_Name="Dandelion"

Now, let us build on that a little bit more. Let's say that we got a suspicious notification regarding Windows notepad, which is a Process, not a user, and it has an EventCode 1.

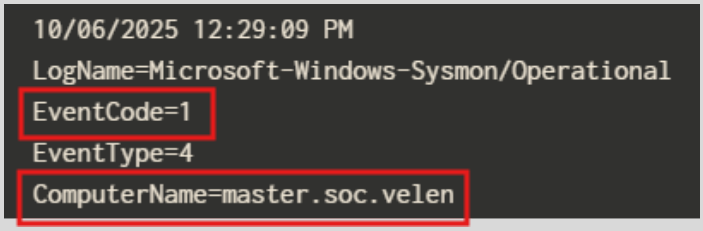
Our administrator, Geralt, has opened the *inputs.conf* of our UFs on the Domain Controller to double-check them. This is a process captured by Sysmon, so let's look at our Sysmon logs via Splunk:

*index=main source="WinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=1 Image="*notepad.exe"*

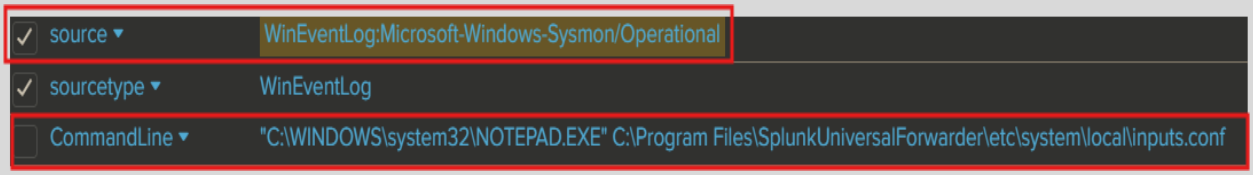
🚦Reminder: In Sysmon, an image is a field which shows the full path of a process/program which was executed (makruss, 2025).

Indeed the result comes back clear (it has been broken into segments for better viewing):

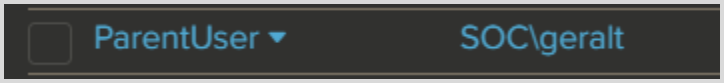
We see the logName, our EventCode 1, and our Domain Controller's computer name:



A little further down we see that *inputs.conf* was accessed:



And little more down we see the User who accessed it:



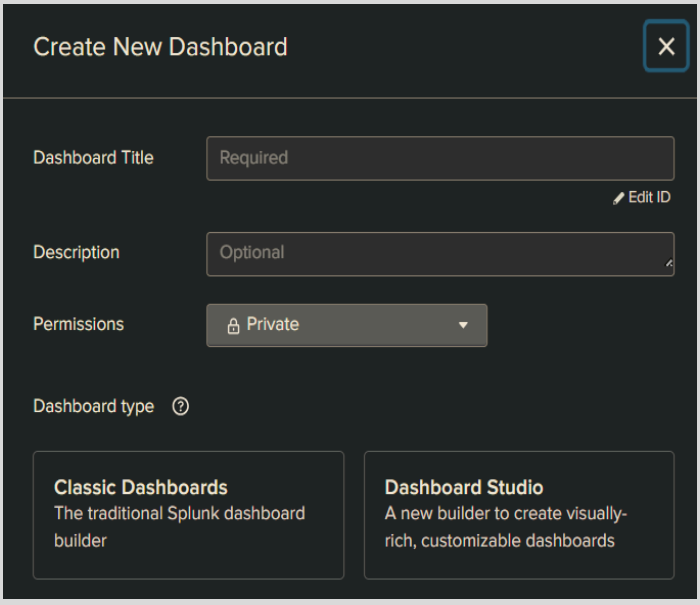
It is now clear that User Activity on files is also monitored and forwarded successfully.

4.3 Dashboards

Dashboards in Splunk are a grouped and more visual approach of the SPL searches, and they contain panels with a variety of modules, such as search boxes, charts, tables and lists (Splunk, 2021).

They provide constant information based on their configuration, without us needing to run SPL searches every time.

We will create and set up four Dashboards based on the four attacks, plus a user Login Activity, give them a name, and choose the *Classic* option for their configurations:



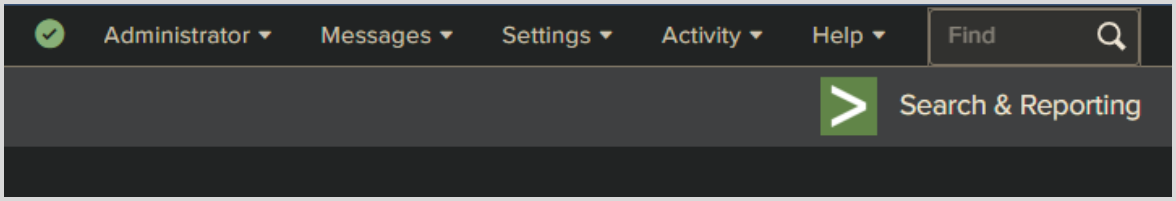
We will configure each panel when we start running our SPL queries for our attack scenarios.

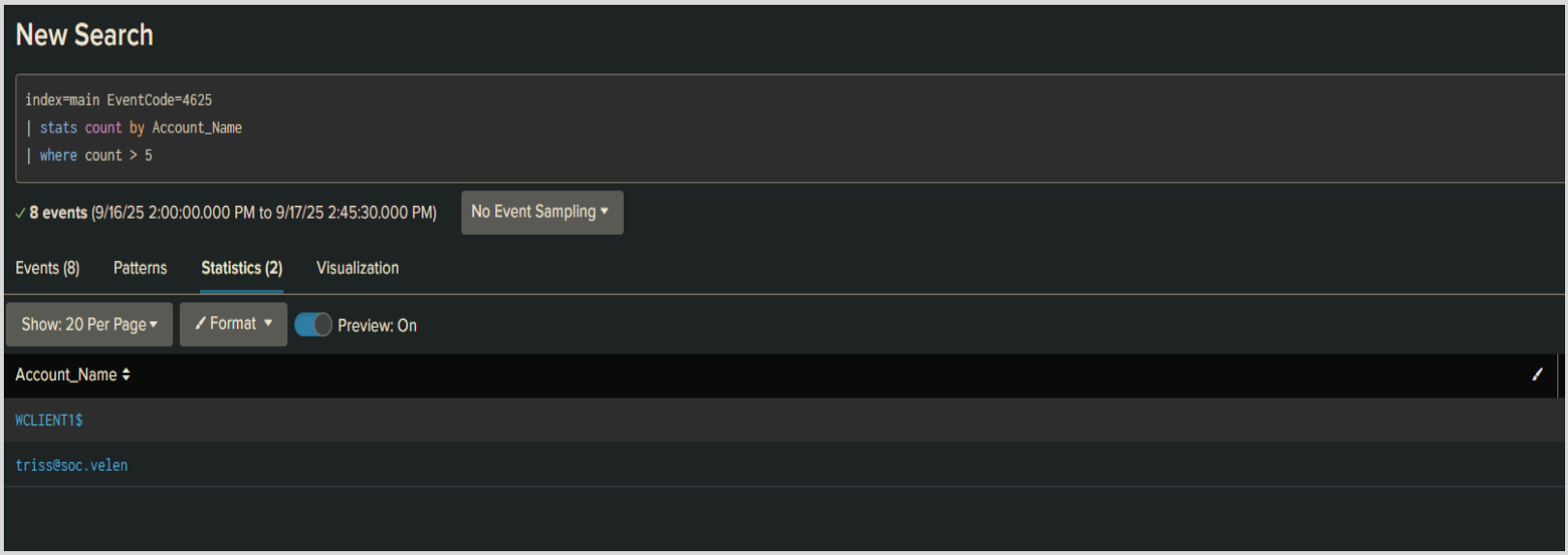
Our Dashboard menu now must look like this:

4.4 Alerts & Reports

After setting up our dashboards, we need to set our alerts up. This is important because the dashboards themselves only update when we click on them or we refresh the page (Splunk, 2021).

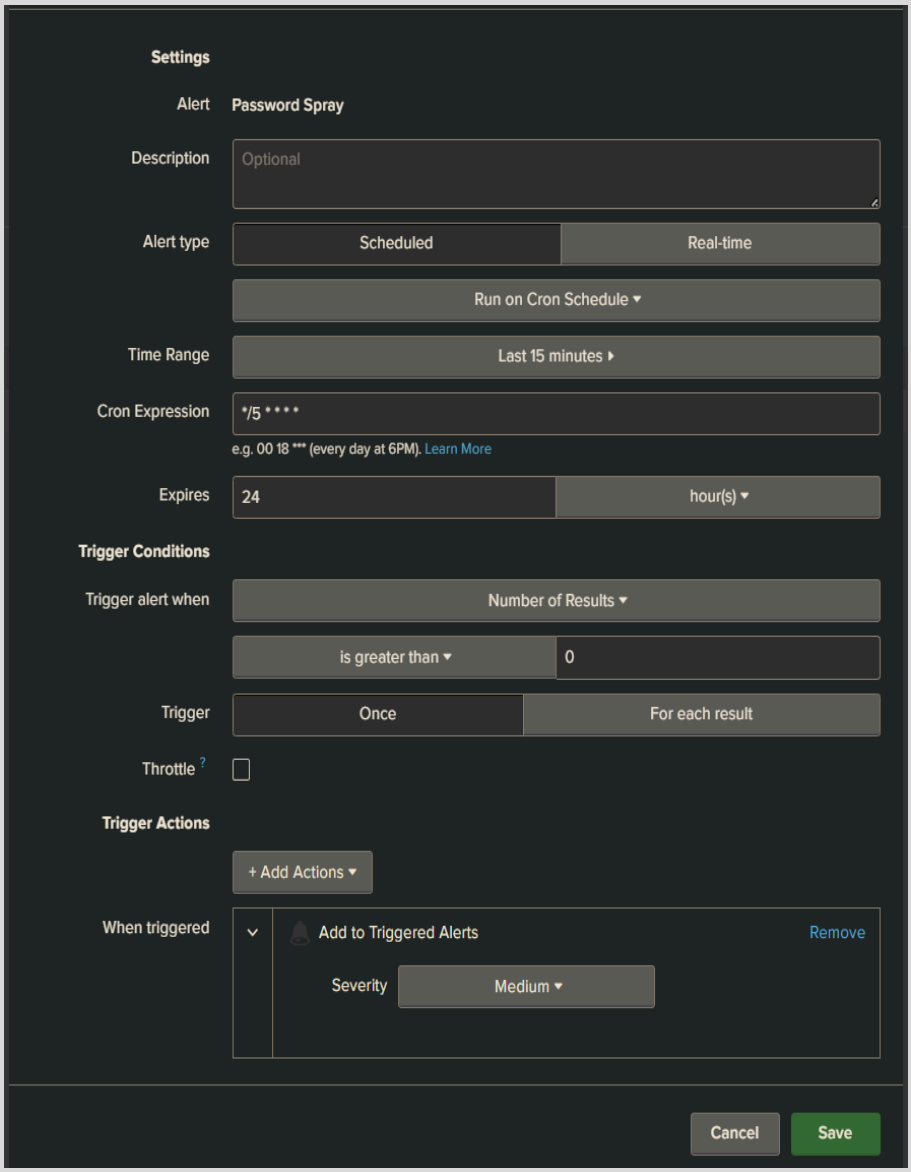
In order to set up an alert, we go to Search and Reporting, and we run our SPL query there.





When the query is done, we can choose to save it as an alert, and set how often we want it to run, what actions to trigger, etc. For the purpose of this lab, let's also run a Cron expression, which will make the alert run every 5 minutes and monitor the last 15 minutes.

As for the *'Triggered Alerts'* we can set the severity on how critical we think are the events being tracked. We can just pick *Medium* for now:

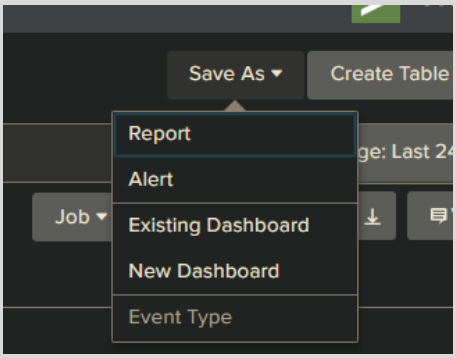


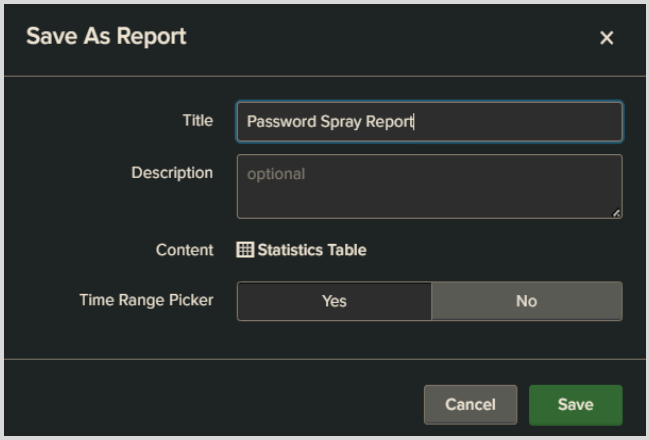
💡 ***- When we configure the creation of each dashboard by using SPL queries, it is smart to aways save these queries as Alerts when we test them. That way, we can quickly click on them on the Alerts menu to get instant results when we need to, without retyping or copy/pasting the commands from our notes.***

🚦 ***Reminder: There are no 'push notifications' for the alerts in the Lab. SOC analysts usually get these via email or other communication channels. We have to manually check to see that the alerts have been triggered.***

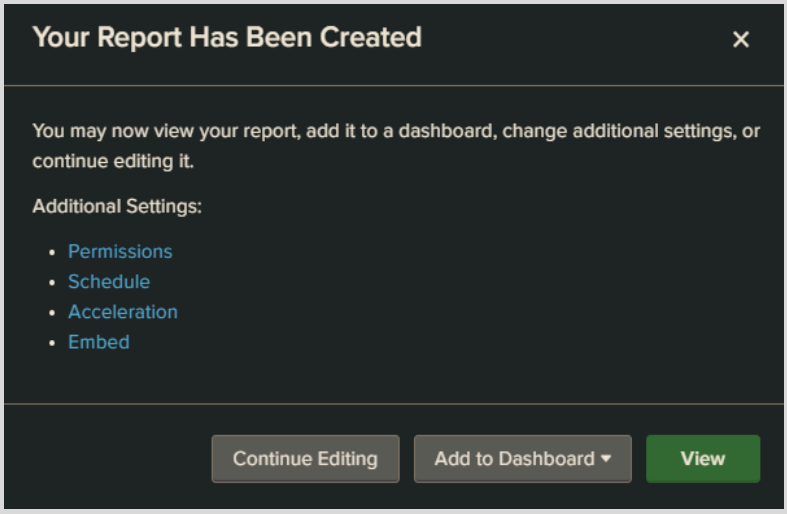
🥷

Reports work the same way as Alerts basically. We set them up by running the SPL query we want, and we save the result as a Report instead of an alert.





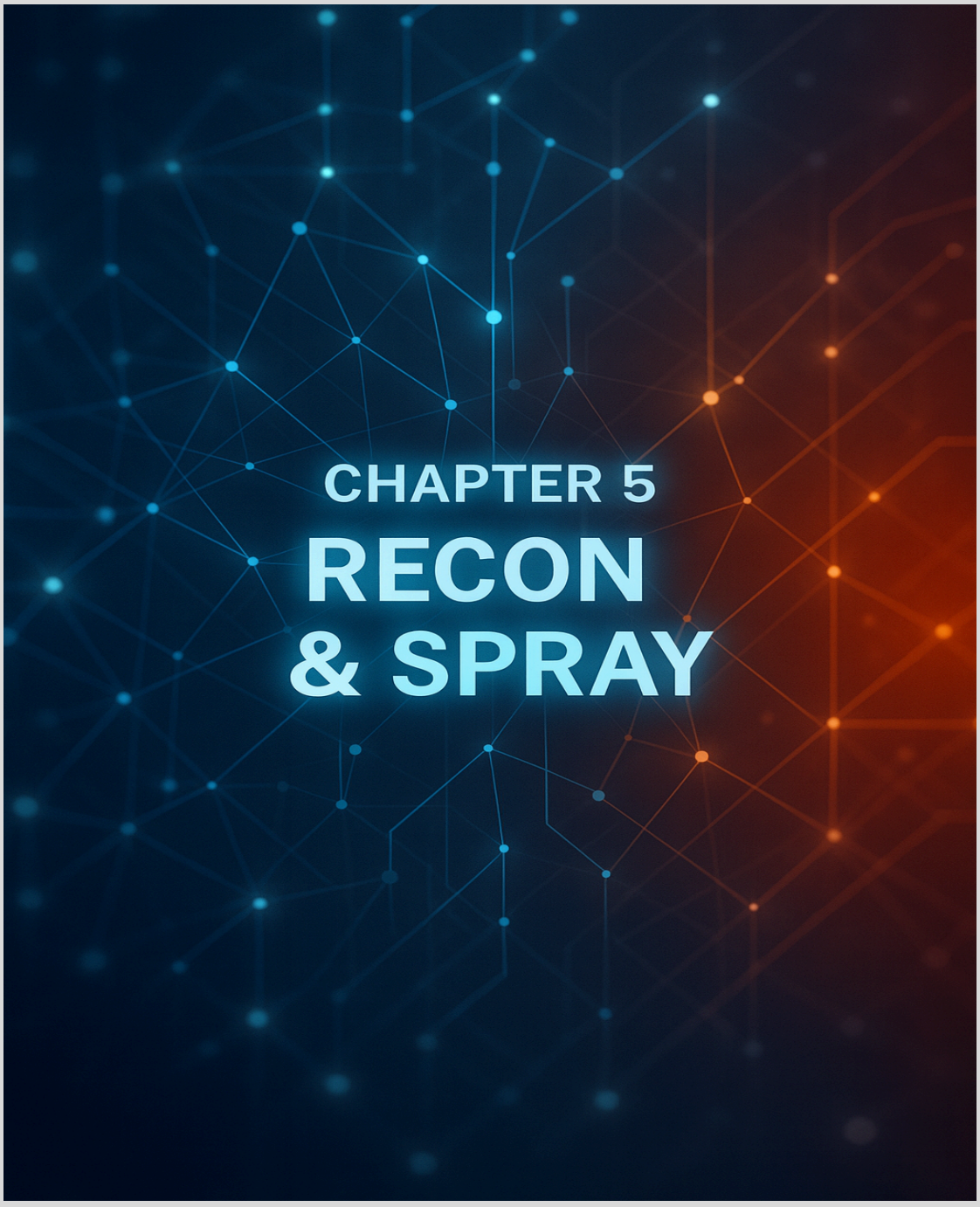
The report can then be configured with an execution timer and sent periodically to other users, containing the results of the query which summarize the suspicious events with the appropriate details, like IPs, Usernames, Timestamps, etc (Splunk Docs, 2025).



We will not be using Reports for our examples, they are too few. But they can be very useful if you decide to expand the lab.

We have now covered the most basic functions of Splunk that will be useful to us in this lab.

It is now time to unleash our attacker, Kali.


CHAPTER 5
RECON
& SPRAY

Introductory Notes

Disclaimer 1: *The results are captured as the commands are executed, so it is easier to discover them as the results are not too many. If you try the same commands days after, you may need extra filtering as the logs would be much more populated.*

Disclaimer 2: *A lot of research has been done to construct the SPL queries, which aim to reflect a beginner's level. SPL is not a set language tool though, and every query is tailored to different contexts (data format, configuration files, etc.), so it is almost impossible to locate specific guides about what we need.*

For that reason, I will mention here the combined sources I used for SPL, while still referencing individual ones where applicable: (Stationx, 2022) (Taillon, 2025) (Lame Creations, 2024) (TruHackMe, 2025) (Splunk, Free Courses, 2025).

Disclaimer 3:
For better understanding of the structure of these simple commands, I strongly advise doing at least some preparation in SPL.

5.1 Reconnaissance

Reconnaissance is the earliest stage of most cyberattacks, and the first step to discover details about a network. It is divided into passive reconnaissance and active reconnaissance.

Passive reconnaissance is basically gathering information about the target without interacting with it (public information, social platforms, simple nmap -unhandshaked- scans etc.), while active reconnaissance is gathering information about a target by actively engaging with it (port enumeration, service scanning, etc.) (Imperva, n.d.)

For our lab we will showcase some active recon. It is expected for a SIEM to drop passive scans to reduce noise, but we will do it anyway to show the process.

We fire up Kali and perform a passive/stealth scan on the network (common move for attackers) by using a simple nmap command (Borges, 2024):

- *nmap -sn 192.168.78.0/24*

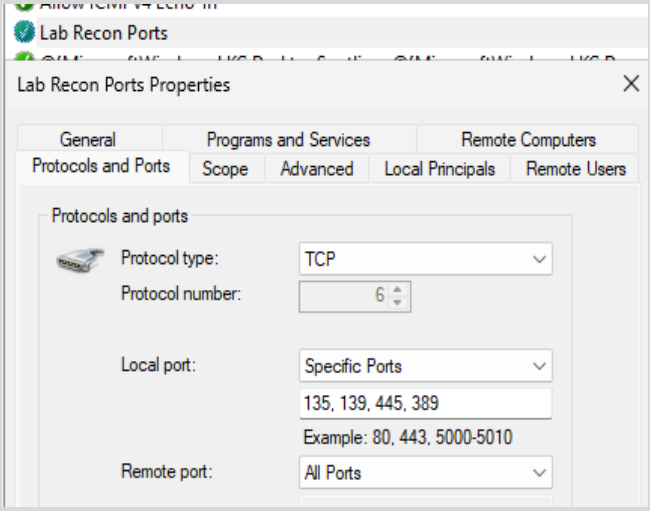
This brings us information about our network machines: 4 VMs, each assigned an IP in the 192.167.78.0/24 network.

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-19 10:43 CEST
Nmap scan report for 192.168.78.1
Host is up (0.00057s latency).
MAC Address: 00:50:56:C0:00:01 (VMware)
Nmap scan report for 192.168.78.10
Host is up (0.00012s latency).
MAC Address: 00:0C:29:D8:1C:32 (VMware)
Nmap scan report for 192.168.78.20
Host is up (0.00076s latency).
MAC Address: 00:0C:29:51:02:19 (VMware)
Nmap scan report for 192.168.78.30
Host is up (0.00057s latency).
MAC Address: 00:0C:29:38:AC:22 (VMware)
Nmap scan report for 192.168.78.40
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 26.75 seconds
```

Now let's say that the attacker wants to know what ports are open on these VM, and starts probing the most common ports that are usually attacked (Schrader, 2024):

- *nmap -p 135,139,445,389 192.168.78.0/24*

📒 ***In a production environment, the most common ports used for enumeration are shut, but the SIEM still picks up the handshake. Our free version of Splunk does not do that, instead it brings up the DC talking to itself, producing a lot of noise, so we must temporarily enable the ports with a firewall rule for the Kali activity to be logged clearly. I couldn't figure out the reason for the loops.***



Here is what our port enumeration brought back:

```
Nmap scan report for 192.168.78.10
Host is up (0.00030s latency).

PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
389/tcp open  ldap
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:D8:1C:32 (VMware)
```

```
Nmap scan report for 192.168.78.30
Host is up (0.00071s latency).

PORT    STATE    SERVICE
135/tcp filtered msrpc
139/tcp filtered netbios-ssn
389/tcp filtered ldap
445/tcp filtered microsoft-ds
MAC Address: 00:0C:29:38:AC:22 (VMware)
```

```
Nmap scan report for 192.168.78.20
Host is up (0.00058s latency).

PORT    STATE    SERVICE
135/tcp open     msrpc
139/tcp filtered netbios-ssn
389/tcp filtered ldap
445/tcp filtered microsoft-ds
MAC Address: 00:0C:29:51:02:19 (VMware)
```

```
Nmap scan report for 192.168.78.40
Host is up (0.000024s latency).

PORT    STATE  SERVICE
135/tcp closed msrpc
139/tcp closed netbios-ssn
389/tcp closed ldap
445/tcp closed microsoft-ds
```

By looking at the results, we can see the.10 VM has multiple ports open for communication, while the .20 VM allows TCP connections and some selective traffic. Therefore, an attacker can deduce that:

- **192.168.78.10 is the DC**
- **192.168.78.20 is the Windows Client.**

Now, since probing for ports is an active scan, our Splunk should have registered the event if we have configured it correctly.

Problem (Solved) When we tried to log for the event in Splunk, we got no results initially. It turns out that NetworkConnect is disabled by default on Sysmon (markruss, 2024) and we need to adjust our Firewall settings.

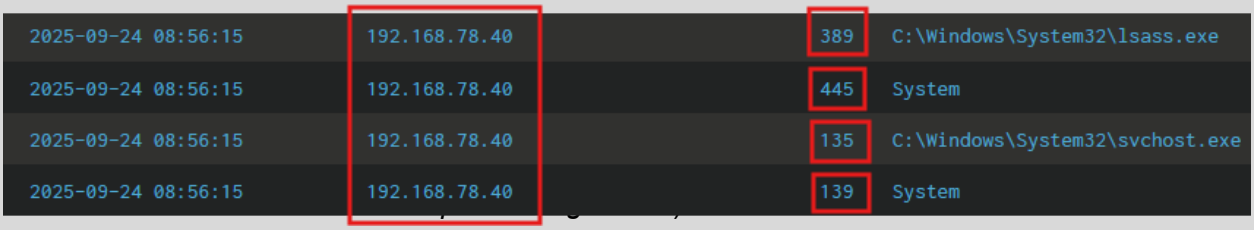
📒 *- Refer to Solution 3: Configuring NetworkConnect.*

After fixing our NetworkConnect problem, we run a simple SPL query with Event 3 to show the network connections from our active recon scan.

Problem (Unsolved/Circumvented) 🛑When looking for Event 3 (Sysmon Network Connections), the results were too many because all VMs were connecting to each other. I tried filtering out the Host, the DC, etc., by excluding IPs, Computer names and whatnot, but there was something going on with the Lab having all machines in the same closed network. So I switched my SPL query to look for the most common ports targeted.

📒 *- Refer to Config 5.1: Port Scanning.*

After running our port-focused query we can see that Splunk has registered that four common ports have been probed by the suspicious IP of .40, our Kali Attacker:



2025-09-24 08:56:15	192.168.78.40	389	C:\Windows\System32\lsass.exe
2025-09-24 08:56:15	192.168.78.40	445	System
2025-09-24 08:56:15	192.168.78.40	135	C:\Windows\System32\svchost.exe
2025-09-24 08:56:15	192.168.78.40	139	System

Maybe it is a good idea to configure Splunk to monitor any DNS activity coming from the .40 IP. So we make a rule and save it in a panel *DNS Queries*, in the Reconnaissance Index Dashboard.

📒 *- Refer to Config 5.2: DNS Logging.*

🥷

But what if the attacker wants to dig deeper and hit our DNS, and maybe try to discover the name of our server, our domain name, and whatever other information is up for grabs?

Let's run **enum4linux**, a pre-installed and common tool used for enumeration (Arr0way, 2015):

- *enum4linux -a 192.168.78.10*

Now we are getting way more results:

- First, we can see the name of our server and our domain:

```
=======================================( Nbtstat Information for 192.168.78.10 )==

Looking up status of 192.168.78.10
        MASTER          <00> -         B <ACTIVE>  Workstation Service
        SOC             <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
```

- Then we can also see that anonymous access in enabled (windows allows some basic anonymous connection with limited rights):

```
[+] Server 192.168.78.10 allows sessions using username '', password ''
```

- In addition, we are getting the Security Identifier for our domain. This is important, because an attacker can use it to find out usernames in the domain (Dansimp, 2023).

```
Domain Name: SOC
Domain Sid: S-1-5-21-3620368929-3162539153-3923263288
```

- Luckily, no info on the OS has been uncovered or any User credentials by enum4Linux:

```
[E] Can't get OS info with smbclient
```

```
[E] Couldn't find users using querydispinfo: NT_STATUS_ACCESS_DENIED
```

Last, let's see if Splunk has captured this DNS enumeration. Let's check the DNS alert we set up in the previous page.

```
9/24/2025 9:57:50 AM 16CC PACKET  000001501BC0D990 UDP Rcv 192.168.78.40  10d2  Q [0001  D  NOERROR] AAAA
```

It is clear that our .40 has received UDP information from our DC, as shown above.

What if the attacker then pivots, and tries to extract our OS and look for another tool that may bring more success?

Let's say that the attacker uses **CrackMapExec**, a widely known open source that is pre-installed on Kali, and is able to automate gathering information, perform lateral movement and execute password attacks, simple or advanced, etc., (Goss, 2024).

What if the attacker executes the following command targeting the SMB (Server Message Block), which is the common client-server protocol for sharing files, printing, assisting login authentication, etc. (Sheldon and Scarpati, 2021)? It uses the 445 port which we have already discovered as open on our DC:

- *crackmapexec smb 192.168.78.10*

This time the attacker's probing has paid off, as the OS we are using has been extracted:

```
SMB         192.168.78.10   445    MASTER           [*] Windows 11 / Server 2025 Build 26100 x64 (name:MASTER)
```

Splunk has also registered this activity by .40 that took place. If we check our DNS alert:

```
9/24/2025 11:29:19 AM 16D0 PACKET  000001501BC3C530 UDP Rcv 192.168.78.40   a8a6   Q [0001
eventtype = windows_ta_data | host = master | source = C:\dns.log | sourcetype = dns_debug
```

At this point, we can conclude that our lab monitoring is configured correctly to capture reconnaissance activities from the .40 IP.

5.2 Password Spray

After revealing that a reconnaissance attempt was made on our network and DC, we now have a valid reason to suspect that the hostile IP 192.168.78.40 will try to escalate its activities.

Therefore, we create a special monitoring table in our Dashboards to capture any activity from the .40 IP.

📒 ***- Please refer to Config 6: Monitoring our Kali Attacker.***

Now, let us assume that our server is slightly misconfigured, for example guest/anonymous access on SMB File Shares is left open, a common attack vector (Xelu86, 2024).

Then the attacker, by using a CrackMapExec command (no need to cover it here as it is more Ethical Hacking area, so we will jist simulate its success - check this Github if you are interested) is able to retrieve the usernames of the domain users and save them in a .txt file.

```
┌──(kali㉿kali)-[~]
└─$ cat users.txt
Dandelion
Triss
Yennefer
Ciri
```

In order to conduct a password spray attack - which is basically when an attacker tries one common password against multiple accounts (Crowdstrike, 2022) - we will use **Hydra**, a pre-installed and common Kali tool, which is designed to attack multiple protocols and show how easy it is to gain unauthorized access (KALI, 2024).

🚦***Kali is not built to communicate with modern SMBs, so we must install the legacy version. We need to install it on our DC by running: Install-WindowsFeature FS-SMB1 (Deland-Han, 2023).***

🥷

After the installation, we can run our SMB attack with this command (Hackviser, 2024):

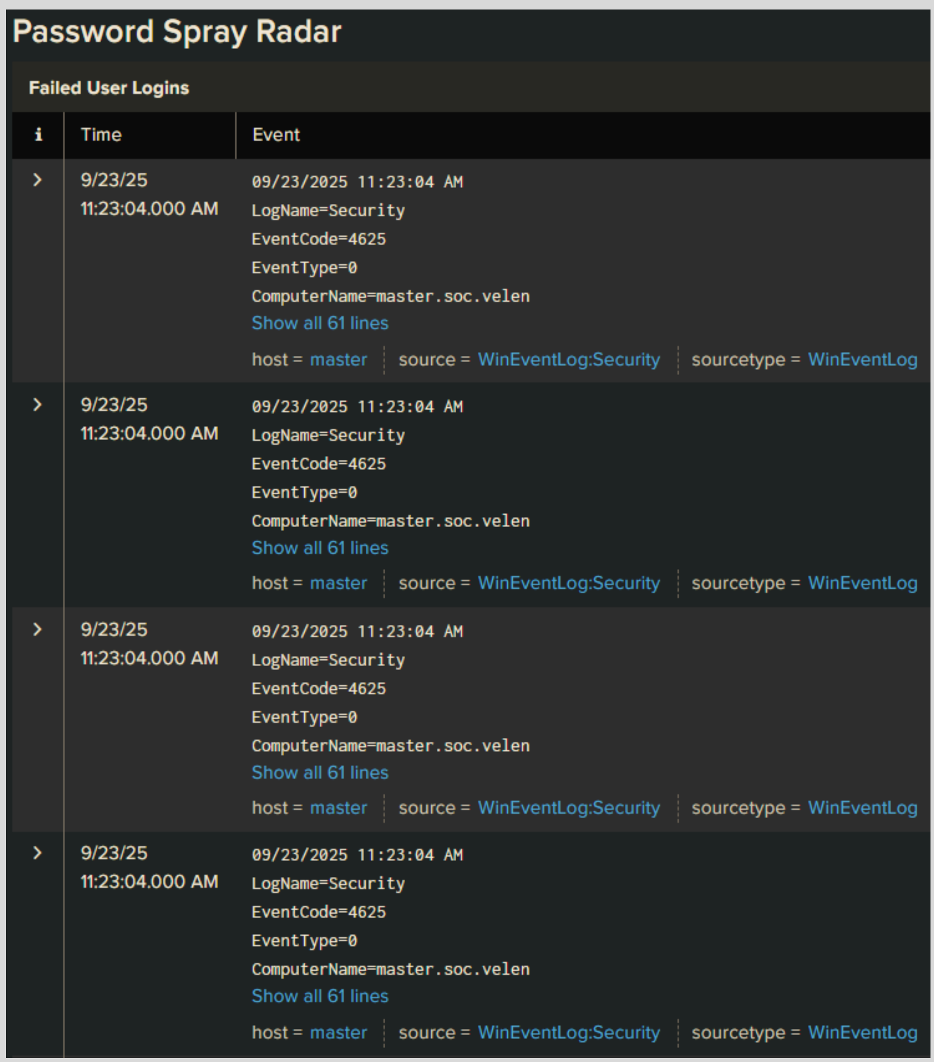
hydra -L users.txt -p Noroff2025! 192.168.78.10 smb

The above basically says, *'check the list of usernames in the txt file, and try the password 'Noroff2025!' by using the vulnerable smb protocol on the .10 IP (our DC)*:

Of course we need to capture this activity in Splunk, so we must construct an SPL query (we saw its most simple version in our Splunk Basics section) to check for the password spray attack.

📒 ***- Please refer to [Config 7](): Monitoring Password Spray Attempts***

Indeed, the activity against our DC has been captured, as we see in the picture on the right:



By checking the details of each event, we can see on which accounts the spray attacked hit:

```
Account For Which Logon Failed:
        Security ID:            S-1-0-0
        Account Name:           triss
        Account Domain:         -
```

```
Account For Which Logon Failed:
        Security ID:            S-1-0-0
        Account Name:           ciri
        Account Domain:         -
```

```
Account For Which Logon Failed:
        Security ID:            S-1-0-0
        Account Name:           yennefer
        Account Domain:         -
```

```
Account For Which Logon Failed:
        Security ID:            S-1-0-0
        Account Name:           dandelion
        Account Domain:         -
```

At the same time, our specific Dashboard for the .40 IP should be alerting us of its activities.

Indeed we can see that .40 has been registered for malicious activity, trying to log in and failing (ID 4625) in four different accounts:

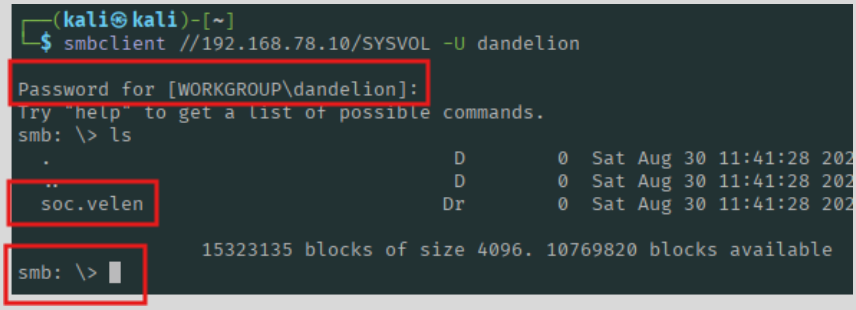
```
master          4625                    -                       192.168.78.40
                                        ciri
master          4625                    -                       192.168.78.40
                                        yennefer
master          4625                    -                       192.168.78.40
                                        triss
master          4625                    -                       192.168.78.40
                                        dandelion
```

Now, what if the password spray worked? For example, by successfully logging in with Dandelion's account?

Then our attacker would be able to log in with *Dandelion* in our SMB file share by using **smbclient**, a simple command line tool (pre-installed in Kali as well), that allows access to network file shares (denizhalil, 2024):

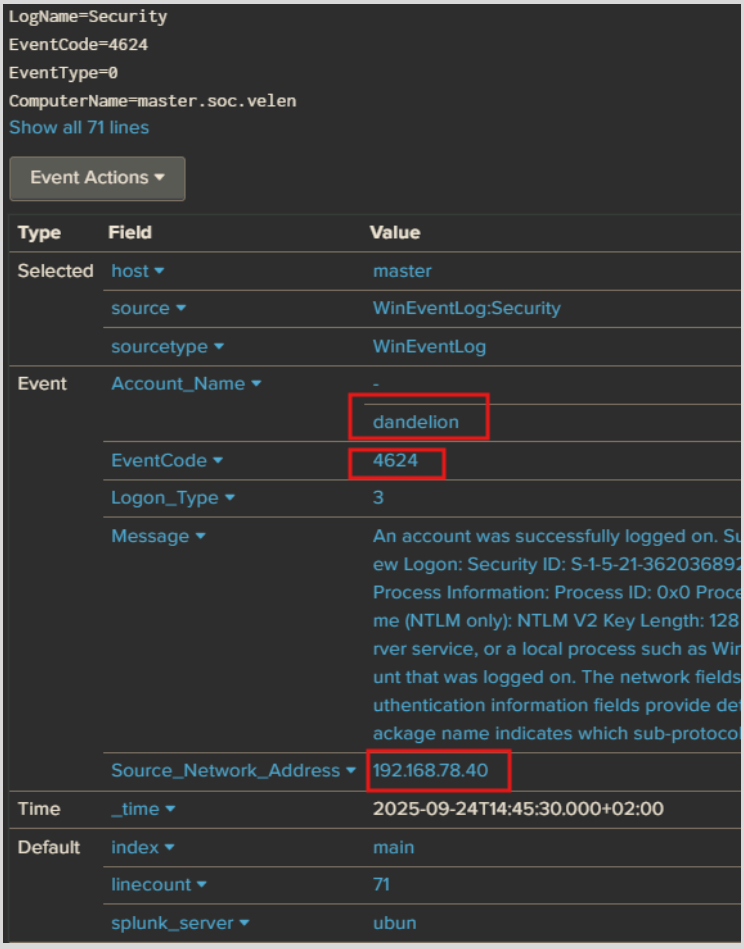
- *smbclient -L 192.168.78.10 -U dandelion*

When we use the above command we are prompted for the password and we log in, arriving at the file share prompt.

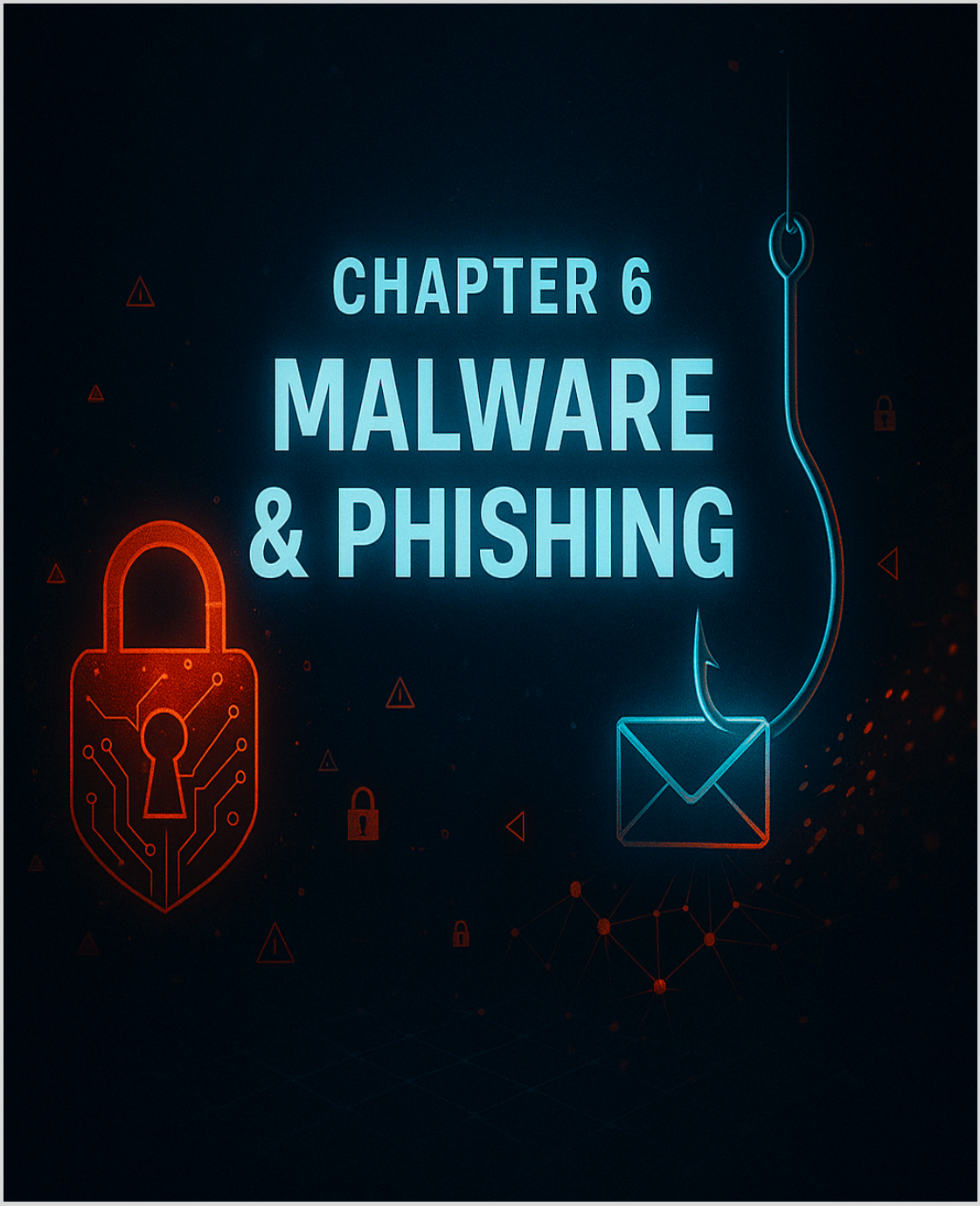


The password spray attack has been successful, but now it is more important than ever to capture that malicious activity in Splunk, because our DC has been breached.

Checking our .40 monitoring panel in Splunk we can see that it has captured the activity. We have now proof that our Kali attacker has used Dandelion's account to successfully connect to our domain (ID 4624):



With this, we have demonstrated that our Splunk can track password spray attempts, both successful and unsuccessful.


CHAPTER 6
MALWARE
& PHISHING

6.1 Malware Execution Detection

Our next attack will be a malware execution. In order to do this we will execute a fake malicious file which simulates malware activity.

The file is called **Eicar** and it was developed by the European Institute for Computer Antivirus Research (EICAR) and Computer Antivirus Research Organization (CARO) to test response and effectiveness of antivirus programs (Eicar, n.d.).

It is widely used as a test method and by executing it we will get security events logged in Sysmon, and as a result security alarms in Splunk.

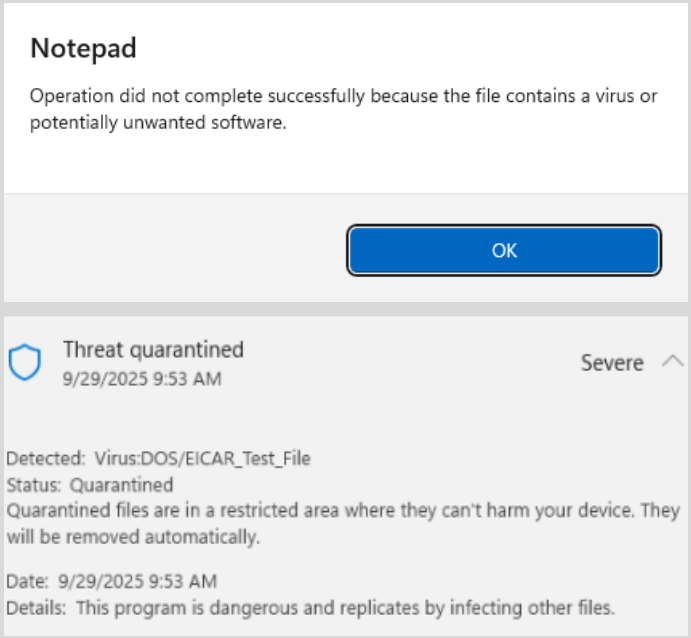
The file can be downloaded by the official site of Eicar, and then be executed on the target system. If we want to create the file on our own, it is very simple, since all we need to do is create a file in notepad and paste the following line (can also be found on the Eicar site):

*X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H**

After we save the file, we log in with a user (let's pick *Triss* this time), and execute the file by simply double clicking on it.

If we have done everything correctly, we should be getting a Windows Firewall alert and a window pop up explaining that execution is blocked because the file contains a virus.

Indeed, our protections seem to be working:

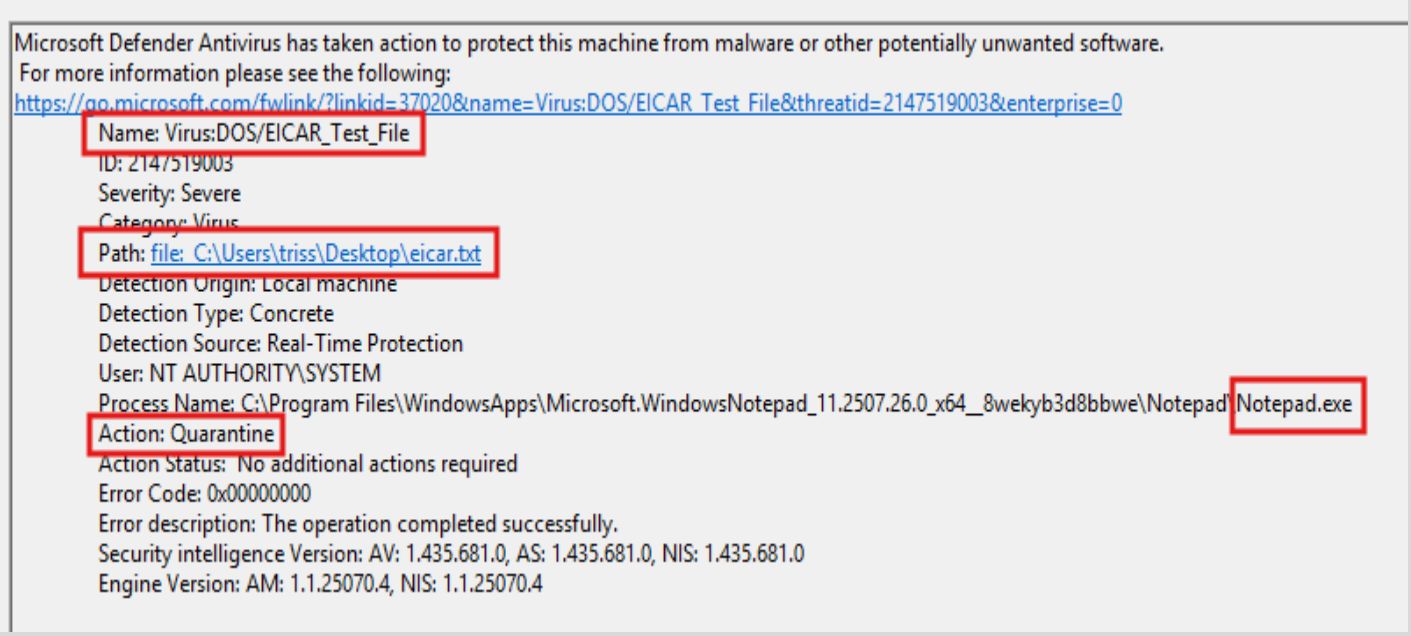


Now, let us see if Sysmon has logged the security event. Unfortunately, it hasn't.

Problem (Solved) When I tried to check Sysmon Operational logs I could not find the created process. It turns out that the process running the eicar file was blocked before it was actually logged, so Sysmon Operational log showed nothing. Therefore, I needed to check the Windows Defender Log.

So, after we check the Windows Defender log in *Event Viewer (Microsoft-Windows-Windows Defender/Operational),* we can clearly see that the event has been logged:

a) The file that was executed.
b) The user who executed it (in this case, Triss).
c) The app that executed (in this case, Notepad).
d) And the action taken by Windows Defender (file has been quarantined).



🚦 ***If you try to locate security events on the Event Viewer on the Domain Controller regarding domain clients, you are not going to find them. Sysmon only logs local events, so it doesn't track cross-domain activity. This is why it is very important to set up Forwarders to all network endpoints (it can also be done with Windows Event Collection) (markruss, 2024).***

🚦 ***While process creations were logged, I couldn't understand why file creation wasn't logging, until I remembered that I am using a well-known community Sysmon Configuration, SwiftOnSecurity, which filters out simple file creation on normal paths like 'Desktop' or 'Documents' to avoid noise. On the other hand, running a file (Event 1) was normally logged and forwarded.***

🥷

Now, let us see if Splunk has caught all the above. The problem is, how do we focus on what we need? In this case, a malicious file execution?

Well, the first thing we do is check our Windows Defender Logs in Splunk, of course.

Problem (Solved) At that point I realized that the client was not set up to forward Windows Defender Logs at all. So I had to set that up in the *inputs.conf* of the client's Universal Forwarders (Labs and Labs, 2020), (Coil, 2023).

📒 ***- Please refer to Solution 4: Windows Defender Logs Forwarding***

After we started forwarding Windows Defender Logs, we simply ran the following in Splunk to capture all events from Windows Defender, which have an EventCode 1116 (siosulli, n.d.):

index=main EventCode=1116

Lo and behold, we can see that Splunk has logged activity from Windows Defender, and has caught *Triss* using Notepad to open a malicious file:

```
Name: Virus:DOS/EICAR_Test_File
ID: 2147519003
Severity: Severe
Category: Virus
Path: file:_C:\Users\triss\Desktop\eicar.txt
Detection Origin: Local machine
Detection Type: Concrete
Detection Source: Real-Time Protection
User: SOC\triss
Process Name: C:\Program Files\WindowsApps\Microsoft.WindowsNotepad_11.2507.26.0_x64__8wekyb3d8bbwe\Notepad\Notepad.exe
Security intelligence Version: AV: 1.435.681.0, AS: 1.435.681.0, NIS: 1.435.681.0
Engine Version: AM: 1.1.25070.4, NIS: 1.1.25070.4
```

We make sure to save our Splunk query as an alert, and add it to our Malware Creation and Execution Dashboard to have constant information regarding events from Windows Defender.

It is clear now that we can monitor Windows Defender Logs and capture the execution of malicious files.

🥷

6.2 Phishing Triage

Now, what if that file was delivered via an email? That would be a Phishing attack, which is still one of the most common attack vectors today.

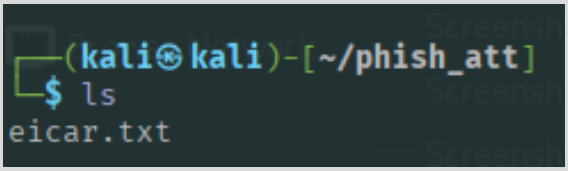
In fact, in 2025, Phishing has risen 49% compared to 2021, while for every 1000 people, a total of 2.330 phishing emails bypass all deployed filters (Baker and Cartier, 2024).

In this demonstration we will use what we have already set up in our previous practical examples, from monitoring Sysmon Events and Firewall logs to cataloguing user activity and malware detection, in order to triage a phishing incident.

Our attacker will be the usual, our Kali Linux, and our victim this time will be the user *Ciri*.

First, we will create our **eicar** file in Kali (as we did in the previous section for our Windows Client) and place it in a new directory (let's call it *phish_att*). We simply use *nano eicar* and paste the below line and just save the file in our directory:

*X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H**



Now that we have our file, let us host it on a simple Apache server, so Ciri can download it when she falls for our phishing bait.

We start Apache in our Kali with *sudo service apache2 start,* and then just copy our file in Apache's web folder (Apache.org, n.d.) so it can be downloaded:

sudo cp ~/phish_att/eicar.txt /var/www/html/

That means that the file is now accessible on *http://192.168.78.40/eicar.txt*

We restart Apache with *sudo service apache2 restart*.

🥷

In order to simulate Ciri receiving an email we simply create a text file which mimics a real email by using notepad (Emlviewer.net, 2023), and then save it as *Security Quick Fix* with the extension of *.eml***.**

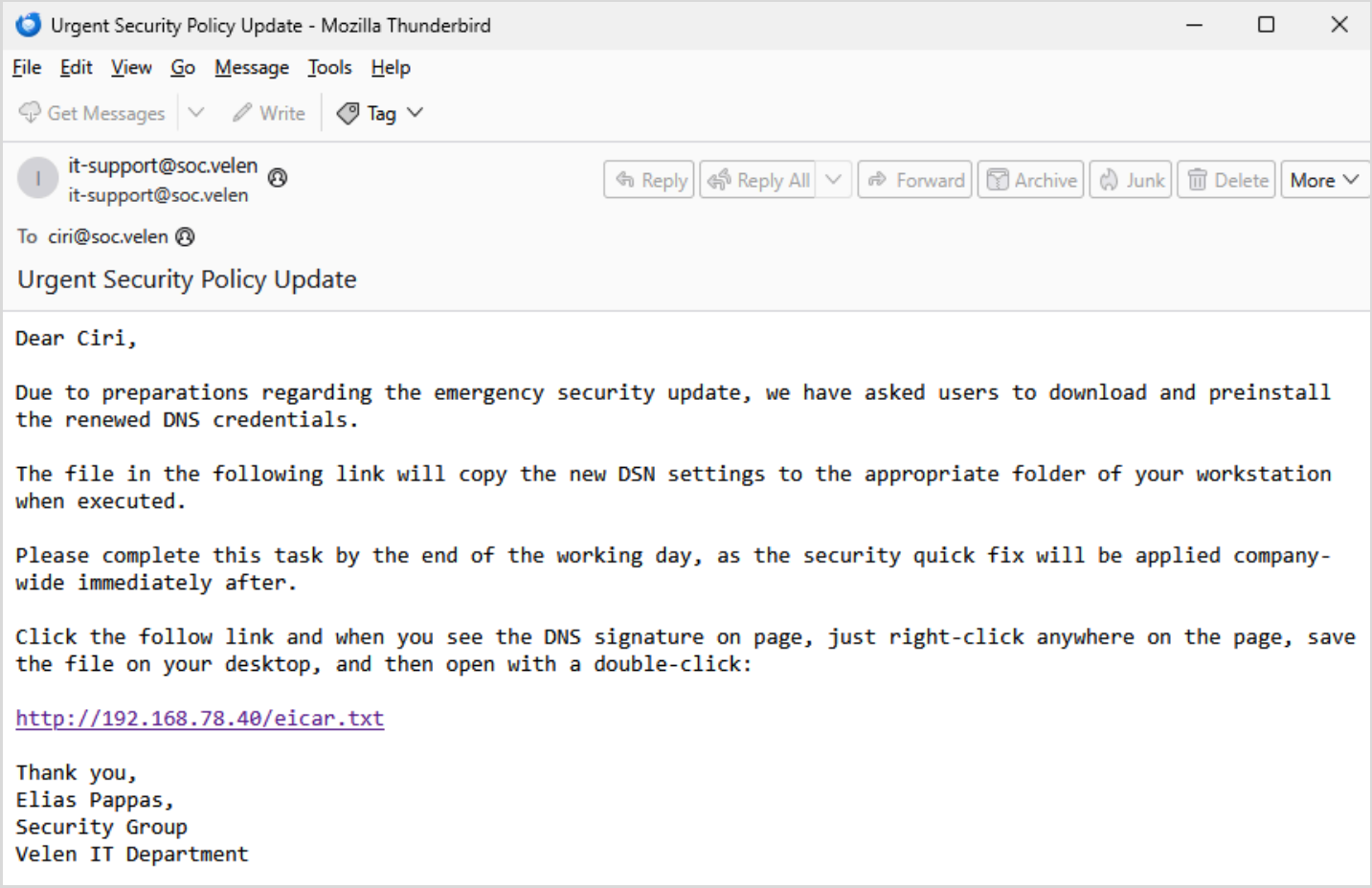
An *.eml* file is basically an email saved as a file, in a format that can give us information about headers and preserve the formatting of the original email, potentially providing valuable intel for digital forensics investigators (Kaler, 2025).

Of course, it is also a common attack vector for criminals, and can contain malicious links or even embedded malware (RecoveryTools and Chard, 2025), as it happens in our case.

Let's open the email.

🚦 ***Outlook cannot operate in offline mode (we haven't set up email accounts or configure outlook since the lab operates in a controlled environment), so in order to open the file I downloaded Mozilla Thunderbird on the Client, and then opened the email file with it.***

When we open the email, it is pretty clear that something urgent is going on regarding security in the company.



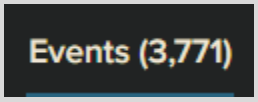
📒 ***- A real phishing file wouldn't be named 'eicar.txt', and the 'DNS signature' is just the eicar string, but hiding all these veers into Ethical Hacking territory, and the lab's goal is to cover detection and monitoring.***

🥷

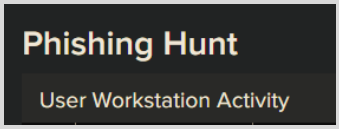
Let us see if the process of loading thunderbird has been captured by Splunk. We run the following simple query to capture processes:

index=main source="WinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=1

Unfortunately, the result is just too much too handle, because it covers ALL processes:

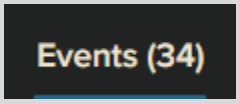


Let us narrow it down by creating a more focused command, targeting User activity, and save that command to our Phishing Hunt Dashboard as:



📒 ***- Please refer to Config 7: Building a User Activity Query in Splunk***

The Splunk results are instantly more manageable (in a production environment there would be more users of course, so we would need an even more focused query):



We can quickly parse these and see that Ciri did indeed open an email, our phishing email. We therefore have our first piece of evidence:

```
OriginalFileName: thunderbird.exe
CommandLine: "C:\Program Files\Mozilla Thunderbird\thunderbird.exe" "C:\Users\ciri\Desktop\Security quick fix email.eml"
CurrentDirectory: C:\WINDOWS\system32\
User: SOC\ciri
```

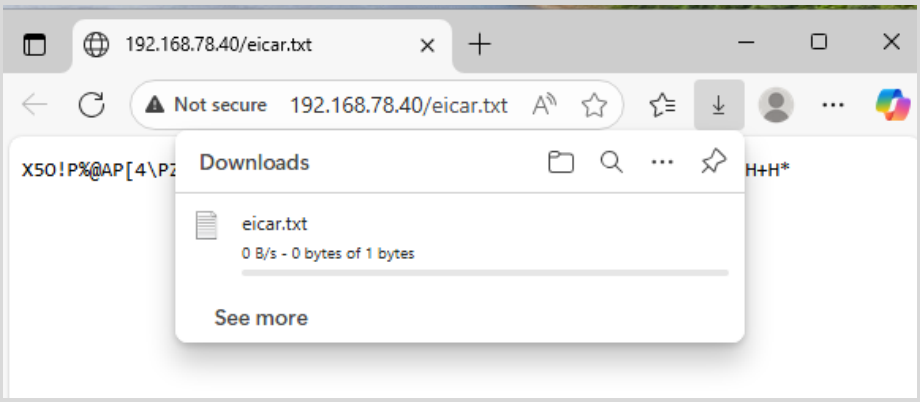
Of course, Ciri, as the victim of our phishing attack, opens the email and clicks the link to download the file, following the instructions. That means that she connected to our attacker, Kali, so there must be a trace of that as well.

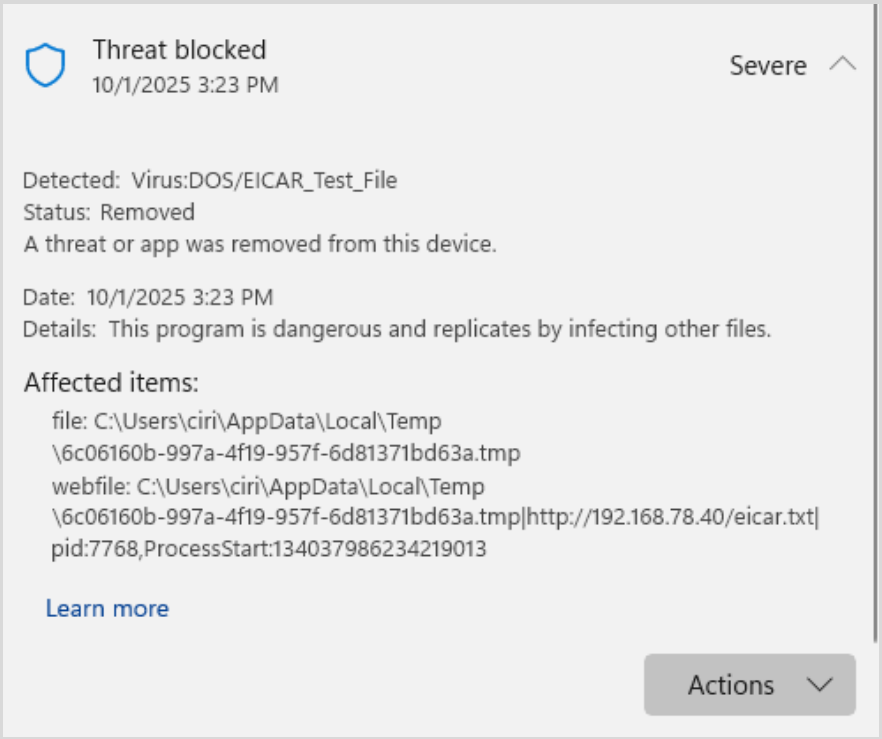
Since we just established an alert in our Phishing Dashboard for user activity, let us see if Splunk has caught Ciri clicking the phishing link and opening the page with Microsoft Edge, where our malware download link is located. Indeed it has:

```
ParentImage: C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
ParentCommandLine: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --single-argument http://192.168.78.40/eicar.txt
ParentUser: SOC\ciri
```

Now, according to the instructions, she needs to save the file on the page.

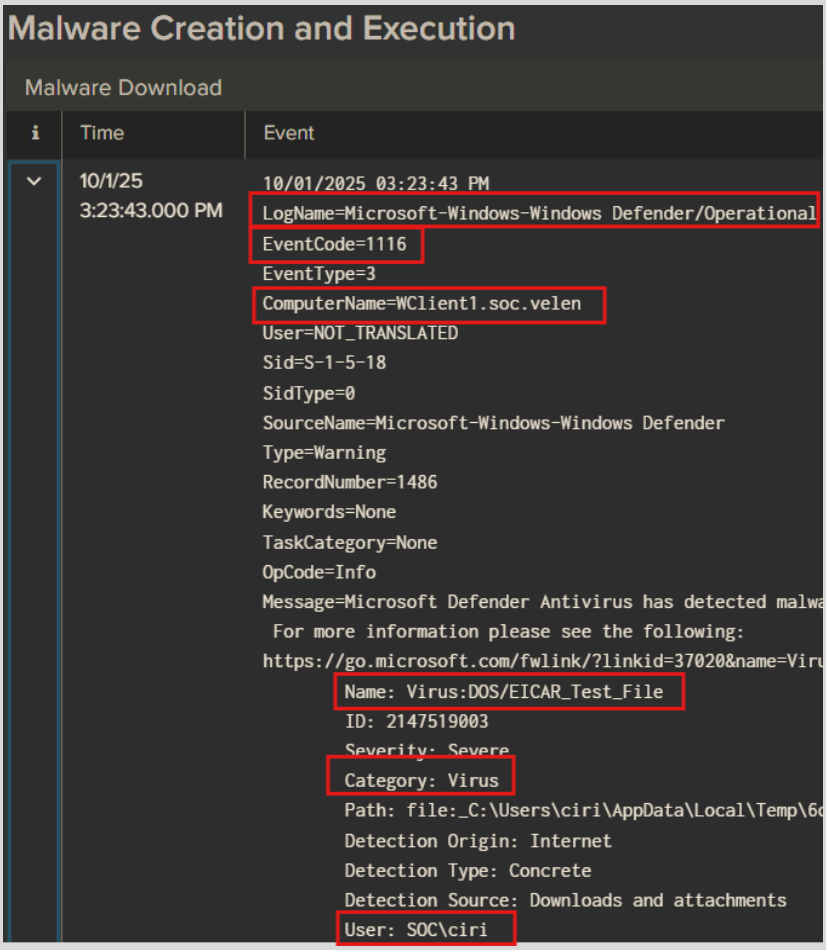
The moment she does this, alerts start popping up immediately. Windows Defender, for example, removes it from the system before it is downloaded:





If we pop over to Splunk, and check the Windows Defender logs for Malware Detection in the Malware Creation and Execution Dashboard, we will see that the event has been picked up:

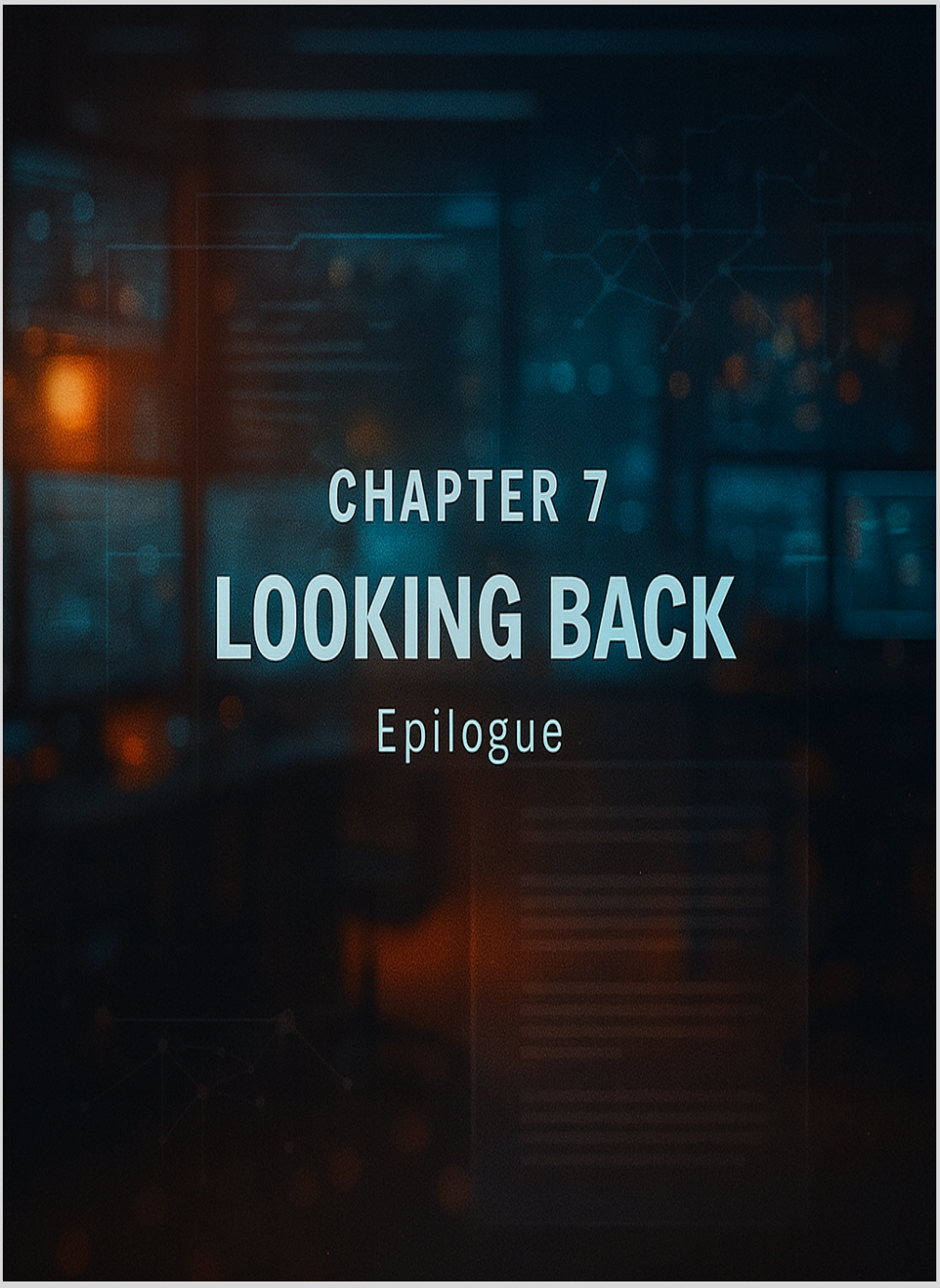
- The Log Name
- The EventCode for Malware Detection
- The Computer name
- The Name of the malware
- The Category of the malware
- The User who was involved



We now have logged in Splunk:

- The process of Ciri opening an email (the .eml file).
- Ciri clicking on a link and opening a page that contains a file.
- Ciri downloading the file.

Therefore, our Phishing Triage scenario has been completed successfully, and our lab activity is now finished.


CHAPTER 7
LOOKING BACK
Epilogue

7.1 Lessons Learnt

Lesson 1: Using an Overly Advanced Sysmon Configuration
The Sysmon configuration I used worked fine, but was far too advanced for the lab's purpose. It filtered out many simple beginner-level events I needed, causing unnecessary troubleshooting and confusion. I thought it would actually be the opposite.

Lesson 2: Splunk Not Logging Expected Events
Because the Sysmon config used custom data formatting, my Splunk queries based on EventCodes returned no results. This forced me to adjust my search structure -valuable learning indeed- but totally outside my intended scope. Proper understanding of data formatting would have saved a lot of time.

Lesson 3: Choosing the Wrong DNS Logging Method
I first enabled DNS Debug Logging, assuming it was standard. While functional, it is resource-heavy and intended mainly for troubleshooting these days. The better choice would have been DNS Analytical Logging, which is the more common choice, especially for my needs.

Lesson 4: Setting Up Linux Telemetry Unnecessarily
I initially configured telemetry on the Ubuntu Splunk server, assuming all hosts required it. Later I realized the Splunk server only receives telemetry, it does not generate its own. It was such a silly misconception.

7.2 Epilogue

This project marked the culmination of my two-year journey in Network and IT Security at Noroff, which was filled with great teachers and valuable educational content. Building the SOC lab from scratch allowed me to apply a wide range of what I had learned, and completing practical tasks using tools that are highly valued by employers, such as Splunk, drove away some beginner fears.

It was exciting to see how all the components came together to create a security posture - from security telemetry and event correlation to SIEM analysis and configuring tools like Sysmon and services like Active Directory, although at moments I felt completely lost and helpless.

As always, I learned a great deal from my mistakes, both big and small, which now seem painfully obvious in hindsight. It goes to show that, in IT, when you think you have everything under control, sometimes it's prudent to examine things more closely.

A simpler and better way is usually around the corner when we take a step back before moving forward.

References & Citations