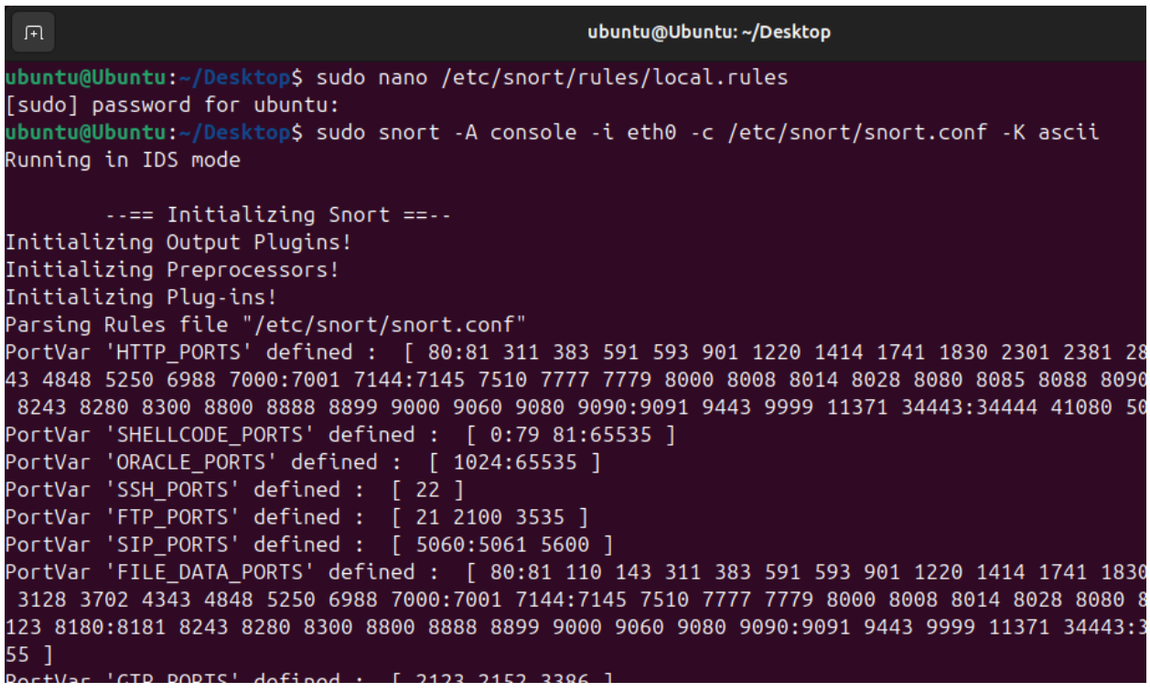
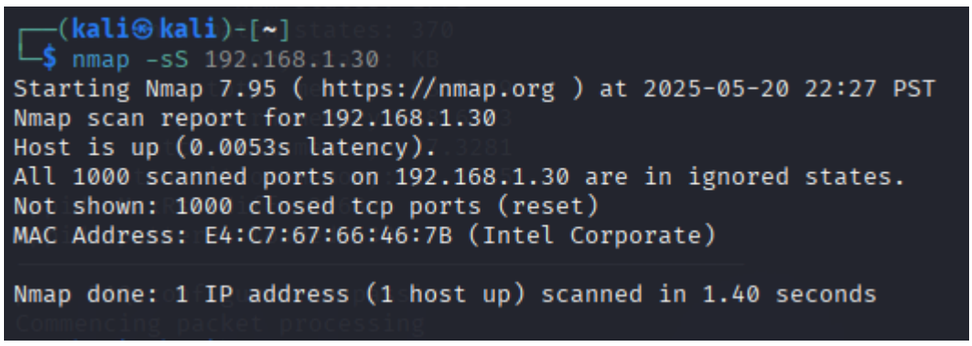
Arzadon, Christian Andrei T.
BSCS 3-B

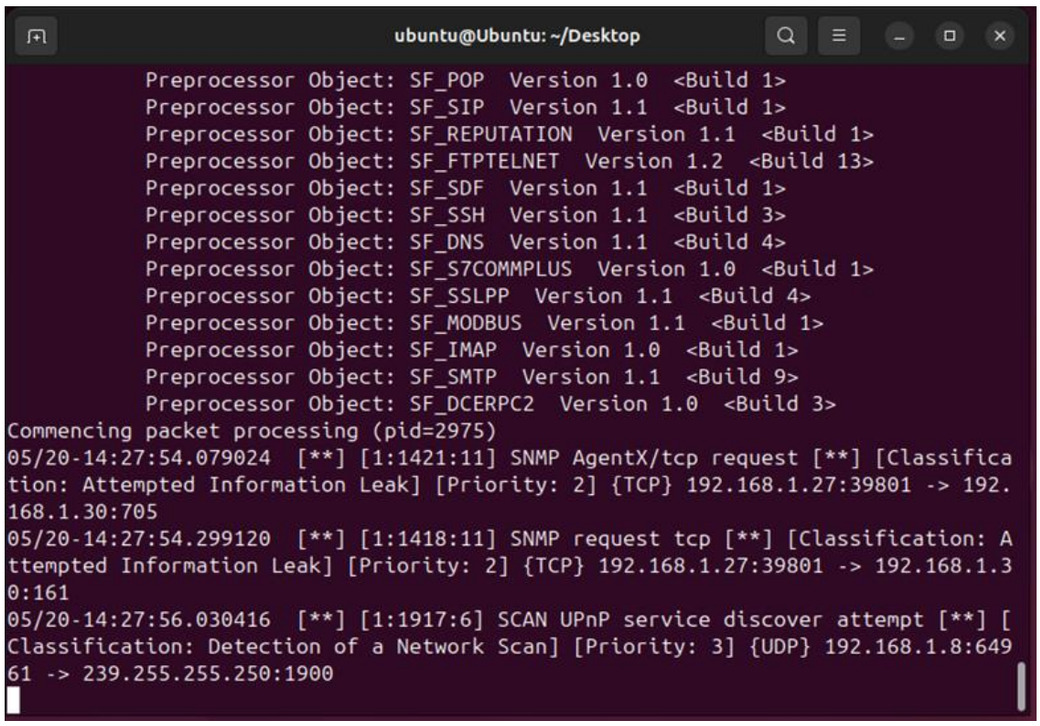
UBUNTU (DEFENDER): Start snort



KALI LINUX (ATTACKER): Kali Linux (Attacker) using nmap to scan Ubuntu(Defender)



UBUNTU (DEFENDER): Snort Detecting attack from Kali Linux (Attacker)



KALI LINUX (ATTACKER): Using nikto

```
┌──(kali㉿kali)-[~]
└─$ nikto -h http://192.168.1.30
- Nikto v2.5.0
---------------------------------------------------------------------------
---------------------------------------------------------------------------
+ 0 host(s) tested
```

KALI LINUX (ATTACKER): Using msfconsole

```
kali@kali: ~
File  Actions  Edit  View  Help

   .00000000.    .;       0     ;   0  ,00000000.
    c0000000.    .00c.      'o00.    ,0000000c
     o000000.    .0000.    :0000.    ,000000o
      l00000.    .0000.    :0000.    ,00000l
       ;0000'    .0000.    :0000.    ;0000;
        .d00o    .0000occcx0000.    x00d.
          ,k0l  .0000000000000000.  .d0k,
            :kk;.0000000000000000.c0k:
              ;k0000000000000000k:
                ,x00000000000x,
                  .l0000000l.
                     ,d0d,
                       .

       =[ metasploit v6.4.56-dev                          ]
+ -- --=[ 2505 exploits - 1291 auxiliary - 431 post       ]
+ -- --=[ 1610 payloads - 49 encoders - 13 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > use auxiliary/scanner/portscan/tcp
msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.1.30
RHOSTS => 192.168.1.30
msf6 auxiliary(scanner/portscan/tcp) > run
[*] 192.168.1.30          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/tcp) >
```

UBUNTU (DEFENDER): Detecting metasploit attack using msfconsole from the Kali VM

```
ubuntu@Ubuntu: ~/Desktop
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_S7COMMPLUS  Version 1.0  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=3029)
05/20-14:38:09.962115  [**] [1:1418:11] SNMP request tcp [**] [Classification: A
ttempted Information Leak] [Priority: 2] {TCP} 192.168.1.27:44557 -> 192.168.1.3
0:161
05/20-14:38:09.963170  [**] [1:1420:11] SNMP trap tcp [**] [Classification: Atte
mpted Information Leak] [Priority: 2] {TCP} 192.168.1.27:42661 -> 192.168.1.30:1
62
05/20-14:38:12.250483  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classifica
tion: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.27:35631 -> 192.
168.1.30:705
```

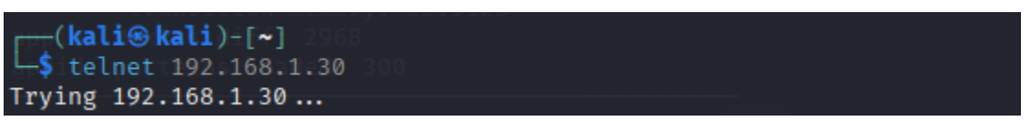
● **The source and destination :** 192.168.1.27:44557 > 192.168.1.3:161

● **The attack type :** SNMP (Simple Network Management Protocol) request tcp

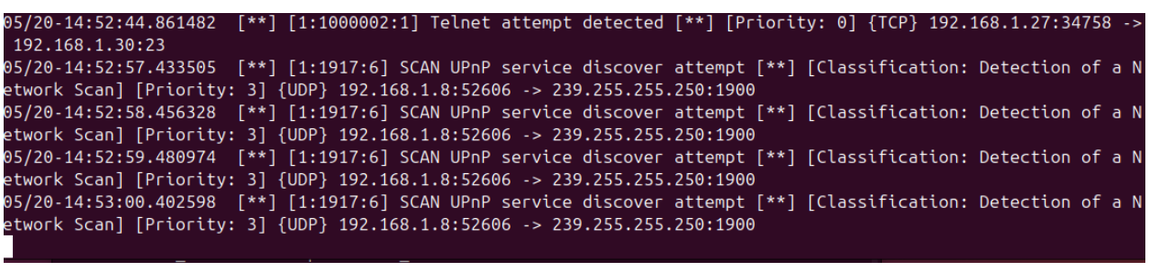
● **Classification and priority :** Attempted Information Leak

UBUNTU (DEFENDER): add rule to /etc/snort/rules/local.rules:

KALI LINUX (ATTACKER): telnet



UBUNTU (DEFENDER):



1. **What types of attacks did Snort detect during your tests?**
   During the tests, Snort detected attacks such as Nmap TCP scans and Metasploit TCP port scans, as simulated using Kali Linux tools.

2. **How does Snort classify and prioritize threats?**
   Snort classifies threats based on the nature of the activity "Attempted Information Leak" for scans like Nmap.

3. **What happens when you trigger your custom rule?**
   When you trigger the custom Telnet rule by attempting a telnet 192.168.1.30 connection, Snort generates an alert with the message "Telnet attempt detected" (as defined in the rule), provided the rule is correctly added to /etc/snort/rules/local.rules

4. **How can Snort be used in real-world networks as both IDS and IPS?**
   IDS, Snort monitors network traffic in real-time, logging and alerting on suspicious activities without blocking them. For IPS, it can be configured in inline mode using NFQUEUE and iptables to actively block malicious traffic based on defined rules, enhancing network security by preventing attacks in addition to detecting them.