

NCY-2 Assignment 3

SNORT

Submitted by

Abdul Sami Qasim (22i-1725)

Ahmad Abdullah (22i-1609)

Submitted to

Prof. Abdullah Abid

Date of Submission: 20/10/2024

Introduction

This assignment involved using snort (an IDS) to detect some well known attacks and produce alerts upon their detection.

Installation and Configuration

To install snort on our devices, we used the command-line interface and input this command

“sudo apt install snort”

After the installation, a simple command like “snort -V” is able to give the output to see if snort is correctly installed or not.

A terminal window with a dark background and light-colored text. The window title is 'gnome@gnome: ~'. The prompt is 'gnome@gnome:~\$'. The command 'snort -V' has been entered. The output shows the Snort logo, version information (2.9.20 GRE Build 82), copyright notices for Martin Roesch & The Snort Team (2014-2022) and Sourcefire, Inc. (1998-2013), and the versions of dependencies: libpcap 1.10.4, PCRE 8.39, and ZLIB 1.3. The prompt 'gnome@gnome:~\$' is shown again at the bottom.

```
gnome@gnome:~$ snort -V

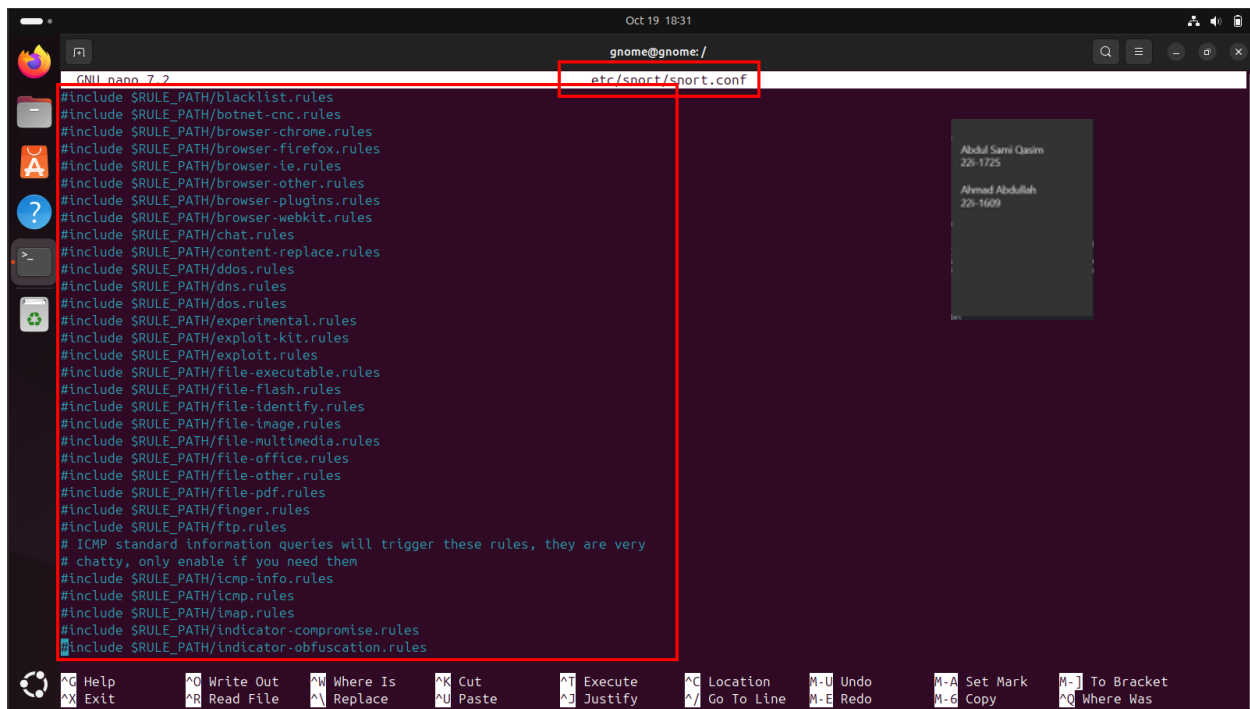
,,_
o" )~
'''

-*> Snort! <*-
Version 2.9.20 GRE (Build 82)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.

Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.10.4 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.3

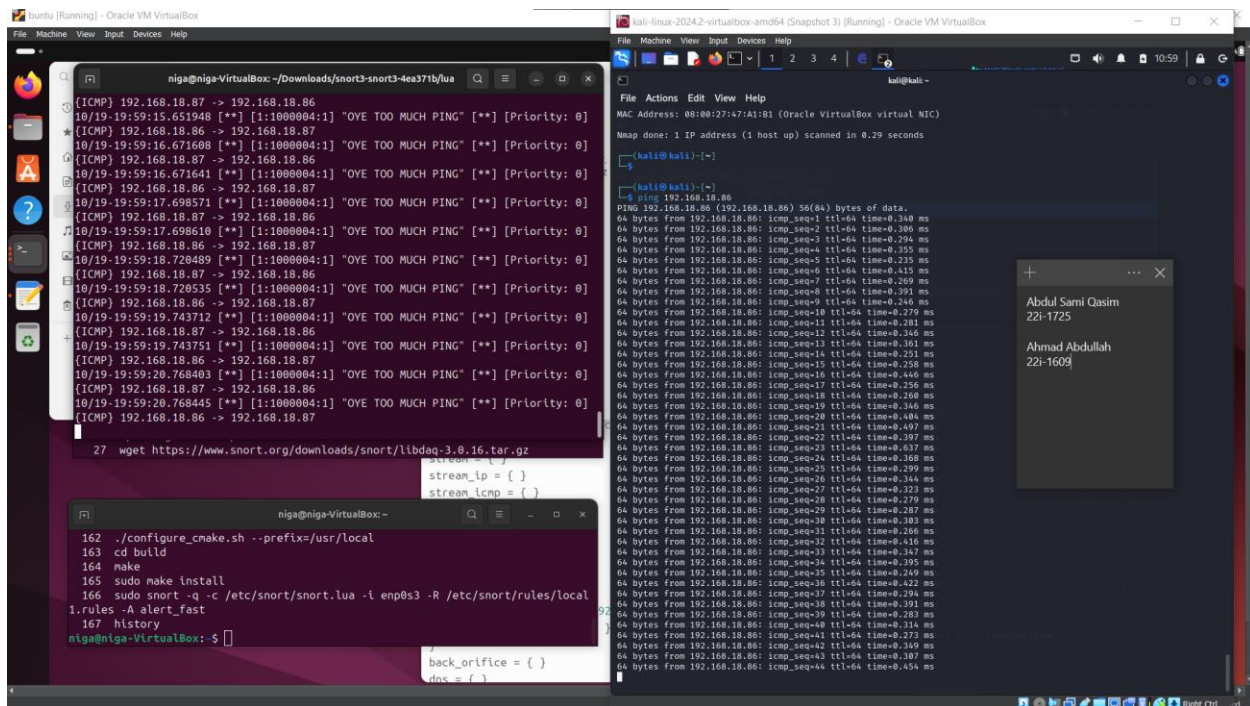
gnome@gnome:~$
```

The next step was to configure snort so that we are able to do our task as smoothly as possible. Firstly, we had to comment on all the other rules in the *snort.conf* file so that snort only checks packets based on our rules only.



Attacks and Simulation

1. DDOS/DOS Attack



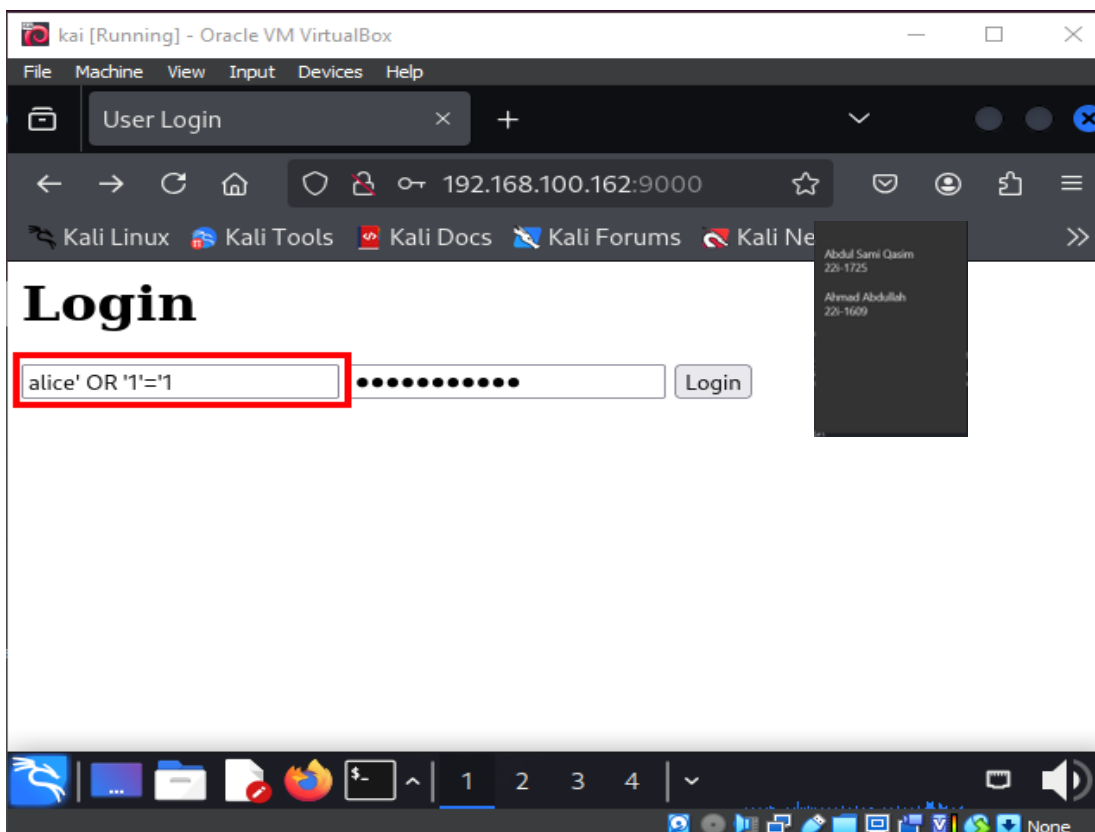
For the DOS attack simulation, we made a rule that generated an alert when more than 5 packets arrived within a minute, yes the bar is set low but it helped us in testing and simulating it.

2. SQLi

We checked the original sql.rules file to get the idea of what alerts for SQL Injections look like and found out that the rules were only based on uri and not on the content itself. URI-based rules are good, but many SQLi are also done on Login forms and these types of injections are detected only when we make rules that detect content-based injections.

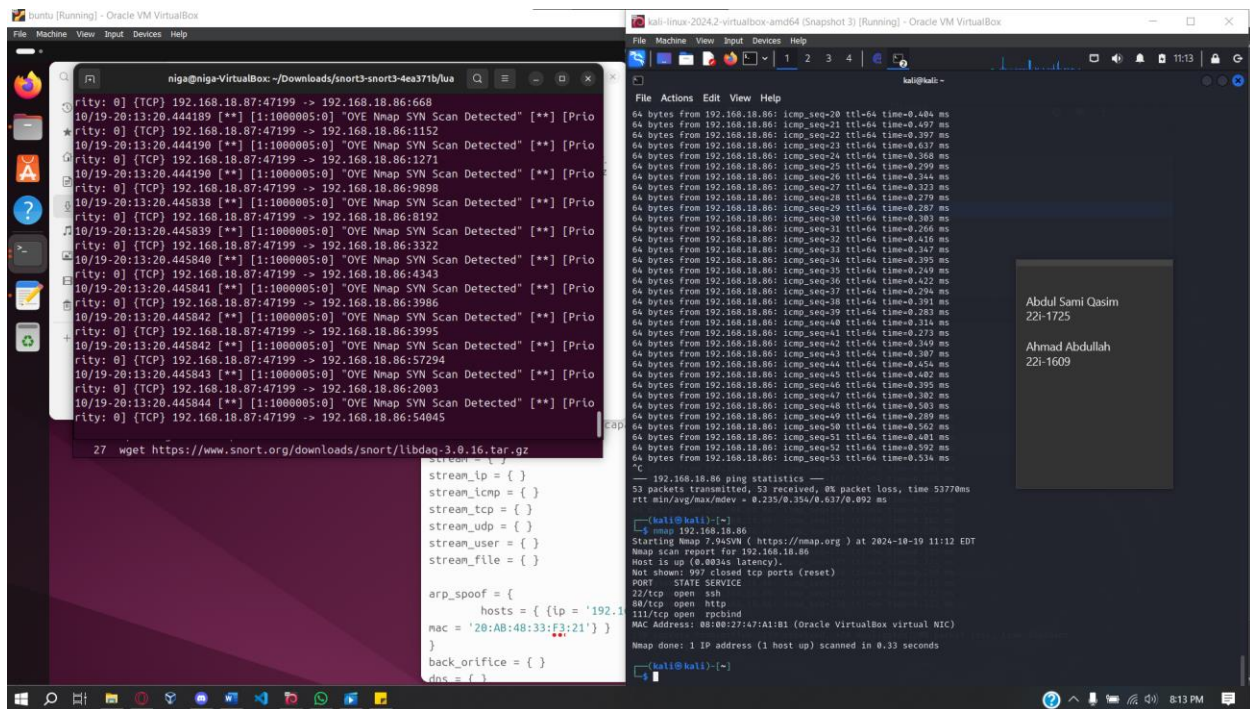
```
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Commencing packet processing (pid=4240)
10/19-15:26:02.110401 [**] [1:1000005:0] Injection Lag Gya SAR!! [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.100.161:3324
4 -> 192.168.100.162:9000
10/19-15:26:09.385940 [**] [1:1000005:0] Injection Lag Gya SAR!! [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.100.161:3798
8 -> 192.168.100.162:9000
10/19-15:26:11.399205 [**] [1:1000005:0] Injection Lag Gya SAR!! [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.100.161:3818
4 -> 192.168.100.162:9000
10/19-15:27:58.388700 [**] [1:1000005:0] Injection Lag Gya SAR!! [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.100.161:3548
4 -> 192.168.100.162:9000
```

1 Short SQLi ALERT



2 Attacking Machin POV

3. Port Scanning



```
niga@niga-VirtualBox: ~/Downloads/snort3-snort3-4ea371b/lua
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:668
10/19-20:13:20.444189 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:1152
10/19-20:13:20.444190 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:1271
10/19-20:13:20.444190 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:9080
10/19-20:13:20.445038 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:8192
10/19-20:13:20.445839 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:3322
10/19-20:13:20.445840 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:4343
10/19-20:13:20.445841 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:3986
10/19-20:13:20.445842 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:3995
10/19-20:13:20.445842 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:57294
10/19-20:13:20.445843 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:2093
10/19-20:13:20.445844 [**] [1:1000005:0] "OVE Nmap SYN Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:47199 -> 192.168.18.86:54045

27 wget https://www.snort.org/downloads/snort/libdaq-3.0.16.tar.gz

stream_ip = {
stream_icmp = {
stream_tcp = {
stream_udp = {
stream_user = {
stream_file = {

arp_spoof = {
hosts = { {ip = '192.1
mac = '20:AB:48:33:F3:21' }
}
back_orifice = {
dns = { }
```

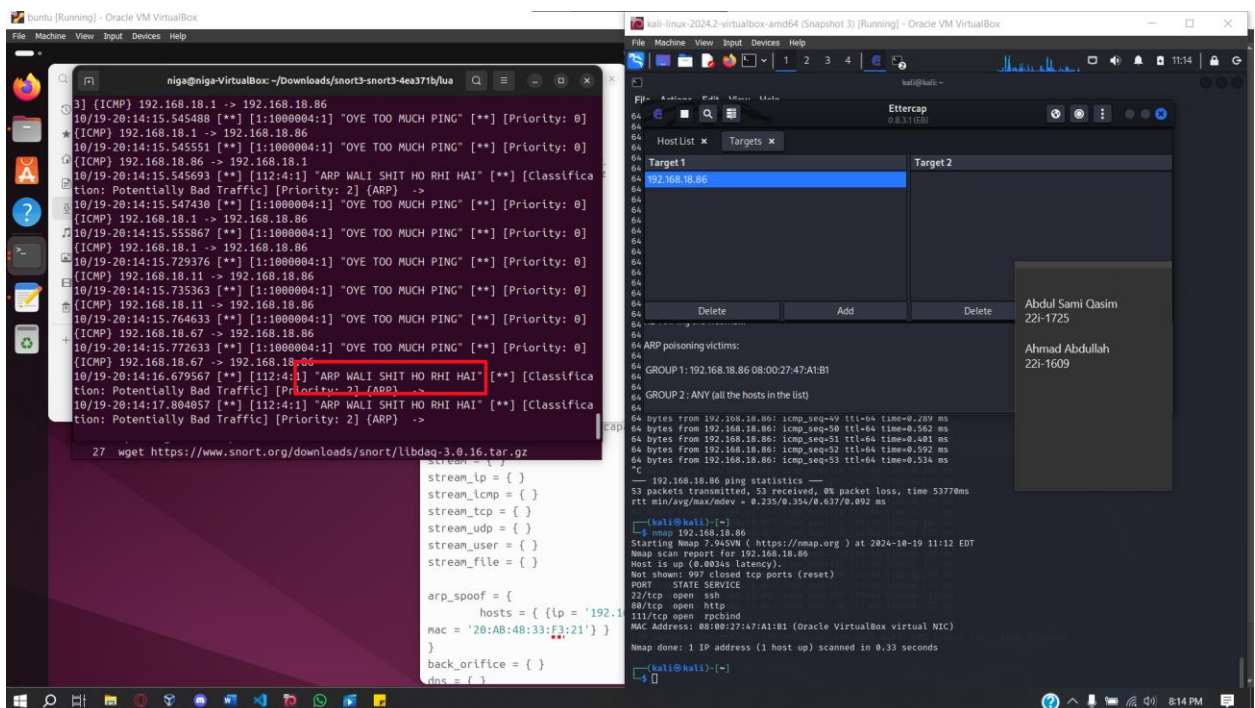
```
kali@kali:~$ nmap 192.168.18.86
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-19 11:12 EDT
Nmap scan report for 192.168.18.86
Host is up (0.0034s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
MAC Address: 08:00:27:47:A1:B1 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 0.33 seconds
```

Abdul Sami Qasim
22i-1725

Ahmad Abdullah
22i-1609

In port scanning we made 5 rules to detect nmap syn, ack, null, fin and xmas flags.

4. Man in the middle



```
niga@niga-VirtualBox: ~/Downloads/snort3-snort3-4ea371b/lua
3] [ICMP] 192.168.18.1 -> 192.168.18.86
10/19-20:14:15.545488 [**] [1:1000004:1] "OVE TOO MUCH PING" [**] [Priority: 0]
[ICMP] 192.168.18.1 -> 192.168.18.86
10/19-20:14:15.545551 [**] [1:1000004:1] "OVE TOO MUCH PING" [**] [Priority: 0]
[ICMP] 192.168.18.86 -> 192.168.18.1
10/19-20:14:15.545693 [**] [112:4:1] "ARP WALI SHIT HO RHI HAI" [**] [Classifica
tion: Potentially Bad Traffic] [Priority: 2] [ARP] ->
10/19-20:14:15.547430 [**] [1:1000004:1] "OVE TOO MUCH PING" [**] [Priority: 0]
[ICMP] 192.168.18.1 -> 192.168.18.86
10/19-20:14:15.555867 [**] [1:1000004:1] "OVE TOO MUCH PING" [**] [Priority: 0]
[ICMP] 192.168.18.1 -> 192.168.18.86
10/19-20:14:15.729376 [**] [1:1000004:1] "OVE TOO MUCH PING" [**] [Priority: 0]
[ICMP] 192.168.18.11 -> 192.168.18.86
10/19-20:14:15.735363 [**] [1:1000004:1] "OVE TOO MUCH PING" [**] [Priority: 0]
[ICMP] 192.168.18.11 -> 192.168.18.86
10/19-20:14:15.764633 [**] [1:1000004:1] "OVE TOO MUCH PING" [**] [Priority: 0]
[ICMP] 192.168.18.67 -> 192.168.18.86
10/19-20:14:15.772633 [**] [1:1000004:1] "OVE TOO MUCH PING" [**] [Priority: 0]
[ICMP] 192.168.18.67 -> 192.168.18.86
10/19-20:14:16.679567 [**] [112:4:1] "ARP WALI SHIT HO RHI HAI" [**] [Classifica
tion: Potentially Bad Traffic] [Priority: 2] [ARP] ->
10/19-20:14:17.804057 [**] [112:4:1] "ARP WALI SHIT HO RHI HAI" [**] [Classifica
tion: Potentially Bad Traffic] [Priority: 2] [ARP] ->

27 wget https://www.snort.org/downloads/snort/libdaq-3.0.16.tar.gz

stream_ip = {
stream_icmp = {
stream_tcp = {
stream_udp = {
stream_user = {
stream_file = {

arp_spoof = {
hosts = { {ip = '192.1
mac = '20:AB:48:33:F3:21' }
}
back_orifice = {
dns = { }
```

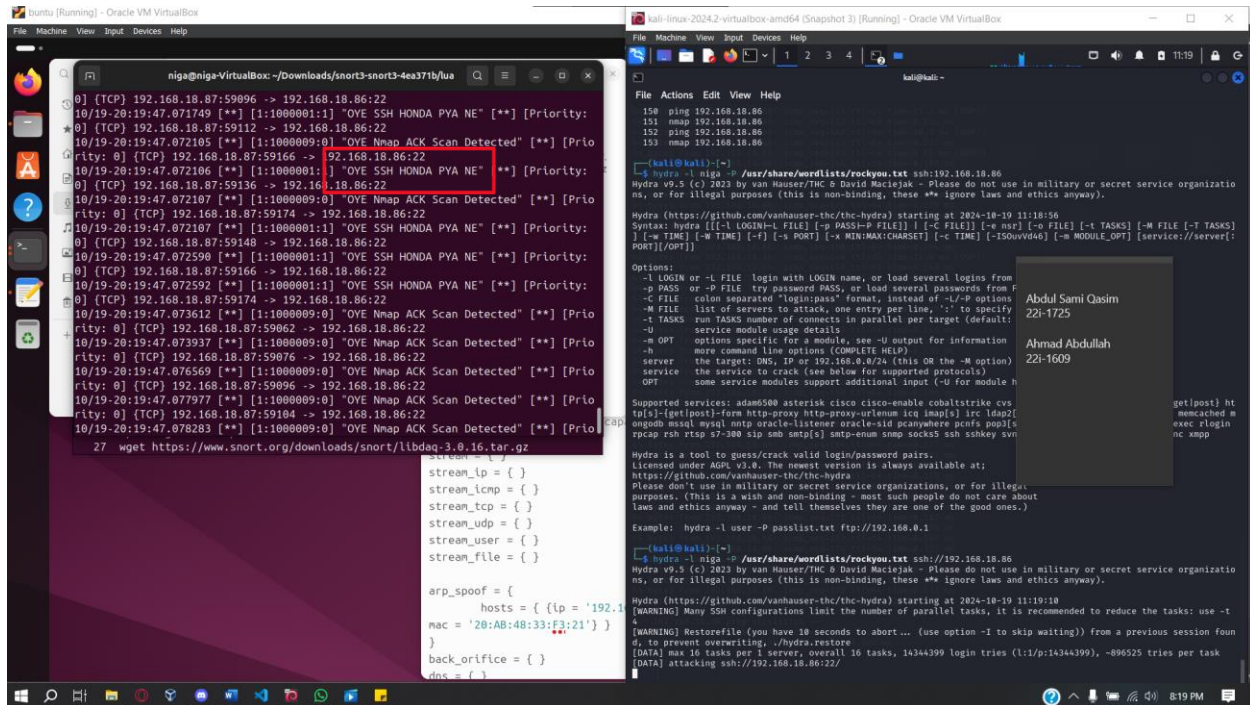
```
kali@kali:~$ nmap 192.168.18.86
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-19 11:12 EDT
Nmap scan report for 192.168.18.86
Host is up (0.0034s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
MAC Address: 08:00:27:47:A1:B1 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 0.33 seconds
```

Abdul Sami Qasim
22i-1725

Ahmad Abdullah
22i-1609

For this we made two rules, one working on the arp packets (apparently snort3 can detect arp packets) and the second rule works on gid:112 which represents the arpspoof preprocessor (the highlighted alert is being generated by the preprocessor rule). On Kali, I'm using the Ettercap GUI to simulate the attack, this is done by first scanning the hosts in the network, flagging the host (ubuntu VM) as the target and then selecting ARP poisoning from the menu.

5. Brute Force



```
0] [TCP] 192.168.18.87:59096 -> 192.168.18.86:22
10/19-20:19:47.071749 [**] [1:1000001:1] "OVE SSH HONDA PYA NE" [**] [Priority:
0] [TCP] 192.168.18.87:59112 -> 192.168.18.86:22
10/19-20:19:47.072105 [**] [1:1000009:0] "OVE Nmap ACK Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:59166 -> 192.168.18.86:22
10/19-20:19:47.072106 [**] [1:1000001:1] "OVE SSH HONDA PYA NE" [**] [Priority:
0] [TCP] 192.168.18.87:59136 -> 192.168.18.86:22
10/19-20:19:47.072107 [**] [1:1000009:0] "OVE Nmap ACK Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:59174 -> 192.168.18.86:22
10/19-20:19:47.072107 [**] [1:1000001:1] "OVE SSH HONDA PYA NE" [**] [Priority:
0] [TCP] 192.168.18.87:59148 -> 192.168.18.86:22
10/19-20:19:47.072590 [**] [1:1000001:1] "OVE SSH HONDA PYA NE" [**] [Priority:
0] [TCP] 192.168.18.87:59166 -> 192.168.18.86:22
10/19-20:19:47.072592 [**] [1:1000001:1] "OVE SSH HONDA PYA NE" [**] [Priority:
0] [TCP] 192.168.18.87:59174 -> 192.168.18.86:22
10/19-20:19:47.073612 [**] [1:1000009:0] "OVE Nmap ACK Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:59062 -> 192.168.18.86:22
10/19-20:19:47.073937 [**] [1:1000009:0] "OVE Nmap ACK Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:59076 -> 192.168.18.86:22
10/19-20:19:47.076559 [**] [1:1000009:0] "OVE Nmap ACK Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:59096 -> 192.168.18.86:22
10/19-20:19:47.077977 [**] [1:1000009:0] "OVE Nmap ACK Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:59104 -> 192.168.18.86:22
10/19-20:19:47.078283 [**] [1:1000009:0] "OVE Nmap ACK Scan Detected" [**] [Pri
rity: 0] [TCP] 192.168.18.87:59104 -> 192.168.18.86:22

27 wget https://www.snort.org/downloads/snort3-4ea371b/lu
stream_ip = {
stream_icmp = {
stream_tcp = {
stream_udp = {
stream_user = {
stream_file = {

arp_spoof = {
hosts = { {ip = '192.1
mac = '20:AB:4B:33:F3:21' } }
back_orifice = {
dns = f }
```

```
150 ping 192.168.18.86
151 nmap 192.168.18.86
152 ping 192.168.18.86
153 nmap 192.168.18.86

[kali@kali:~]$ hydra -l niga -P /usr/share/wordlists/rockyou.txt ssh://192.168.18.86
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organization
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser/thc-hydra) starting at 2024-10-19 11:18:56
Syntax: hydra [-l LOGIN|-f FILE] [-p PASS|-P FILE] [-c FILE] [-w MAX] [-e FILE] [-t TASKS] [-M FILE] [-T TASKS]
] [-w TIME] [-W TIME] [-f] [-s PORT] [-v MIN:MAX:CHARSET] [-c TIME] [-ISOuvVd46] [-m MODULE_OPT] [service://server[:
PORT]/[OPT]]

Options:
-l LOGIN or -l FILE login with LOGIN name, or load several logins from
-p PASS or -p FILE try password PASS, or load several passwords from F
-c FILE colon separated "login:pass" format, instead of -l/-p options
-m FILE list of servers to attack, one entry per line, ':' to specify
-t TASKS run TASKS number of connect in parallel per target (default:
service module usage details)
-u OPT options specific for a module, see -O output for information
-h more command line options (CONSISTE HELP)
server the target: DNS, IP or 192.168.0.8/24 (this OR the -M option)
service the service to crack (see below for supported protocols)
some service modules support additional input (-U for module h
OPT

Supported services: adams500 asterisk cisco cisco-enable cobaltstrike cvs
ftp(s)-[get|post]-form http-proxy http-proxy-urlenum icq imap(s) irc ldap2f
omdbox mysql mysqln oracle-listener oracle-sid pcanywhere pcnfs pop3(s
rpcap rpcs rtsp s7-300 sip smb smtp(s) smtp-enum snmp socks5 ssh sshkey svn
Hydra is a tool to guess/crack valid login/password pairs.
Licensed under AGPL v2.0. The newest version is always available at:
https://github.com/vanhauser-thc/thc-hydra
Please don't use in military or secret service organizations, or for illega
purposes. (This is a wish and non-binding - most such people do not care about
laws and ethics anyway - and tell themselves they are one of the good ones.)

Example: hydra -l user -P passlist.txt ftp://192.168.0.1

[kali@kali:~]$ hydra -l niga -P /usr/share/wordlists/rockyou.txt ssh://192.168.18.86
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organization
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser/thc-hydra) starting at 2024-10-19 11:19:10
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t
4
[WARNING] Restoring file (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session four
4, to prevent overwriting: ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (1:1/p:14344399), -896525 tries per task
[DATA] attacking ssh://192.168.18.86:22/

Abdul Sami Qasim
22i-1725
Ahmad Abdullah
22i-1609
get|post| ht
memcached m
exec rlogin
nc nmap
```

For this, we made a rule to detect packets containing SSH in them and if more then 5 packets that fit this category arrive from one source in under 60 seconds, an alert is generated that a brute-force attempt is being made. The highlighted alert is generated as such and the attack was simulated by using hydra.

6. Crafted Malicious Traffic

We simulated a web-based malicious traffic. The alert contained an exact match of what request we made but it can be further improved based on requirement. We can say it is a base rule for more sophisticated rule that will detect different kinds of malicious traffic.

```
Preprocessor Object: SF_S7COMPLUS Version 1.0 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNP3 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_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Commencing packet processing (pid=4602)
10/19-15:45:24.159026 [**] [1:1000002:1] Malicious HTTP Traffic Detected [**] [Priority: 0] [TCP] 192.168.100.161:42052 -> 192.168.100.162:80
10/19-15:45:25.080807 [**] [1:1000002:1] Malicious HTTP Traffic Detected [**] [Priority: 0] [TCP] 192.168.100.161:42066 -> 192.168.100.162:80
10/19-15:45:25.839253 [**] [1:1000002:1] Malicious HTTP Traffic Detected [**] [Priority: 0] [TCP] 192.168.100.161:42068 -> 192.168.100.162:80
10/19-15:45:26.234144 [**] [1:1000002:1] Malicious HTTP Traffic Detected [**] [Priority: 0] [TCP] 192.168.100.161:42070 -> 192.168.100.162:80
10/19-15:45:26.598833 [**] [1:1000002:1] Malicious HTTP Traffic Detected [**] [Priority: 0] [TCP] 192.168.100.161:42074 -> 192.168.100.162:80
10/19-15:45:28.140292 [**] [1:1000002:1] Malicious HTTP Traffic Detected [**] [Priority: 0] [TCP] 192.168.100.161:42088 -> 192.168.100.162:80
```

3 SNORT Malicious Traffic Alerts

We ran a command “curl -A “Malicious” <http://192.168.100.162>” from our attacking machine and SNORT picked the packet and alerted.

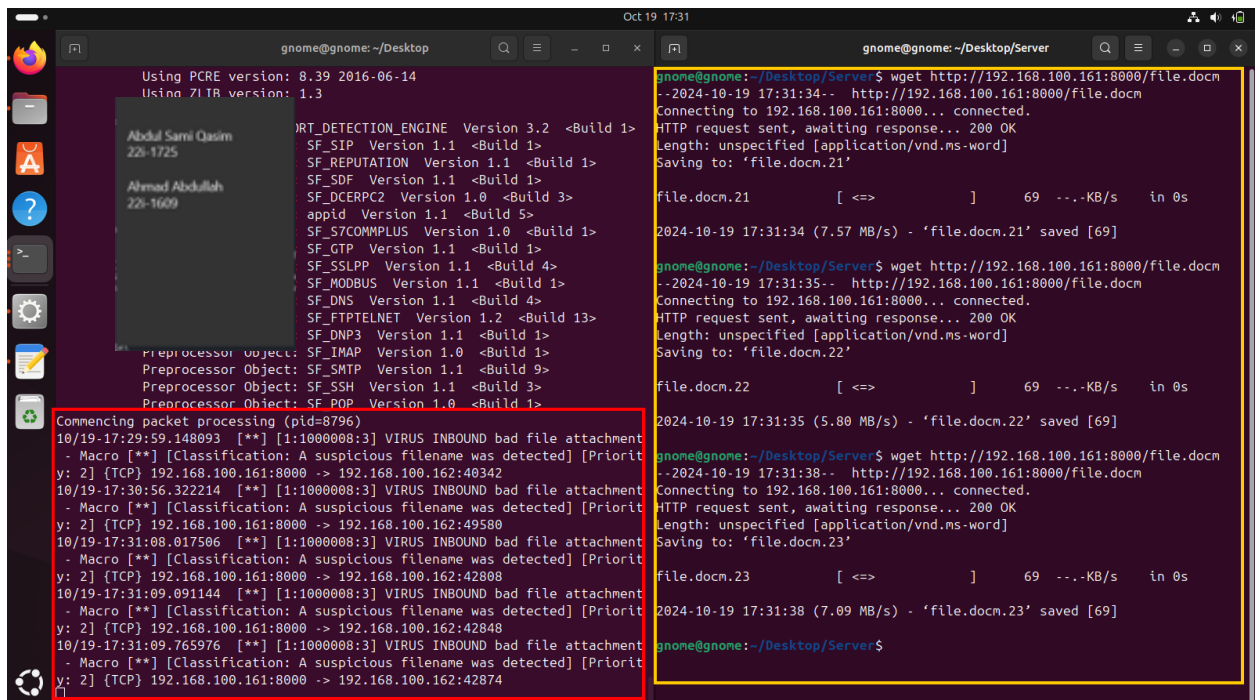
```
File Actions Edit View Help
hal@kali: ~
$ curl -A "malicious" http://192.168.100.162
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
  Modified from the Debian original for Ubuntu
  Last updated: 2022-08-22
  See: https://launchpad.net/bugs/1966006
-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Apache2 Ubuntu Default Page: It works</title>
<style type="text/css">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
  }
  body, html {
    padding: 3px 3px 3px 3px;
    background-color: #D8DBE2;
  }
</style>
</head>
<body>
  <div style="text-align: center;>
    <div>
      <img alt="Ubuntu logo" data-bbox="490 450 510 470"/>
      <br/>
      Ubuntu
    </div>
    <br/>
    <div>
      Apache2 Ubuntu Default Page: It works
    </div>
  </div>
</body>
</html>
```

4 Attacking Machine POV

7. Research-Based

For this, we looked at many rule files to find a *virus.rules* which is a rule file that detects suspicious files with 'exe, bat, etc'. This rule file was missing modern MS Office extensions with macros. So we added those extensions to our custom rules.

On the left, we downloaded a file with the .docm extension, and on the right, SNORT alerted us to a suspicious file download.



The image shows two terminal windows side-by-side. The left window, titled 'gnome@gnome: ~/Desktop', displays the output of a Snort rule engine. It lists various rule sets (SF_SIP, SF_REPUTATION, SF_SDF, SF_DCEP2, SF_S7COMPLUS, SF_GTP, SF_SSLPP, SF_MODBUS, SF_DNS, SF_FTPTELNET, SF_DNP3, SF_INAP, SF_SMTP, SF_SSH, SF_POP) and their versions. Below this, it shows packet processing for three connections from 192.168.100.161:8000 to 192.168.100.162, all flagged as 'VIRUS INBOUND bad file attachment' with a classification of 'A suspicious filename was detected'. The right window, titled 'gnome@gnome: ~/Desktop/Server', shows the execution of a wget command to download a file from http://192.168.100.161:8000/file.docm. It shows the connection, the file being saved as 'file.docm.21', and the download progress (7.57 MB/s). It then repeats the process for 'file.docm.22' and 'file.docm.23'.

```
gnome@gnome: ~/Desktop
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.3

Abdul Sami Qasim
226-1725

Ahmad Abdullah
226-1609

SF_DETECTION_ENGINE Version 3.2 <Build 1>
SF_SIP Version 1.1 <Build 1>
SF_REPUTATION Version 1.1 <Build 1>
SF_SDF Version 1.1 <Build 1>
SF_DCEP2 Version 1.0 <Build 3>
SF_S7COMPLUS Version 1.0 <Build 1>
SF_GTP Version 1.1 <Build 1>
SF_SSLPP Version 1.1 <Build 4>
SF_MODBUS Version 1.1 <Build 1>
SF_DNS Version 1.1 <Build 4>
SF_FTPTELNET Version 1.2 <Build 13>
SF_DNP3 Version 1.1 <Build 1>
SF_INAP Version 1.0 <Build 1>
SF_SMTP Version 1.1 <Build 9>
SF_SSH Version 1.1 <Build 3>
SF_POP Version 1.0 <Build 1>

Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_POP Version 1.0 <Build 1>

Commencing packet processing (pid=8796)
10/19-17:29:59.148093 [**] [1:1000008:3] VIRUS INBOUND bad file attachment
- Macro [**] [Classification: A suspicious filename was detected] [Priority: 2] [TCP] 192.168.100.161:8000 -> 192.168.100.162:40342
10/19-17:30:56.322214 [**] [1:1000008:3] VIRUS INBOUND bad file attachment
- Macro [**] [Classification: A suspicious filename was detected] [Priority: 2] [TCP] 192.168.100.161:8000 -> 192.168.100.162:49580
10/19-17:31:08.017506 [**] [1:1000008:3] VIRUS INBOUND bad file attachment
- Macro [**] [Classification: A suspicious filename was detected] [Priority: 2] [TCP] 192.168.100.161:8000 -> 192.168.100.162:42808
10/19-17:31:09.091144 [**] [1:1000008:3] VIRUS INBOUND bad file attachment
- Macro [**] [Classification: A suspicious filename was detected] [Priority: 2] [TCP] 192.168.100.161:8000 -> 192.168.100.162:42848
10/19-17:31:09.765976 [**] [1:1000008:3] VIRUS INBOUND bad file attachment
- Macro [**] [Classification: A suspicious filename was detected] [Priority: 2] [TCP] 192.168.100.161:8000 -> 192.168.100.162:42874

gnome@gnome: ~/Desktop/Server$ wget http://192.168.100.161:8000/file.docm
--2024-10-19 17:31:34-- http://192.168.100.161:8000/file.docm
Connecting to 192.168.100.161:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/vnd.ms-word]
Saving to: 'file.docm.21'

file.docm.21 [ <=> ] 69 --KB/s in 0s

2024-10-19 17:31:34 (7.57 MB/s) - 'file.docm.21' saved [69]

gnome@gnome: ~/Desktop/Server$ wget http://192.168.100.161:8000/file.docm
--2024-10-19 17:31:35-- http://192.168.100.161:8000/file.docm
Connecting to 192.168.100.161:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/vnd.ms-word]
Saving to: 'file.docm.22'

file.docm.22 [ <=> ] 69 --KB/s in 0s

2024-10-19 17:31:35 (5.80 MB/s) - 'file.docm.22' saved [69]

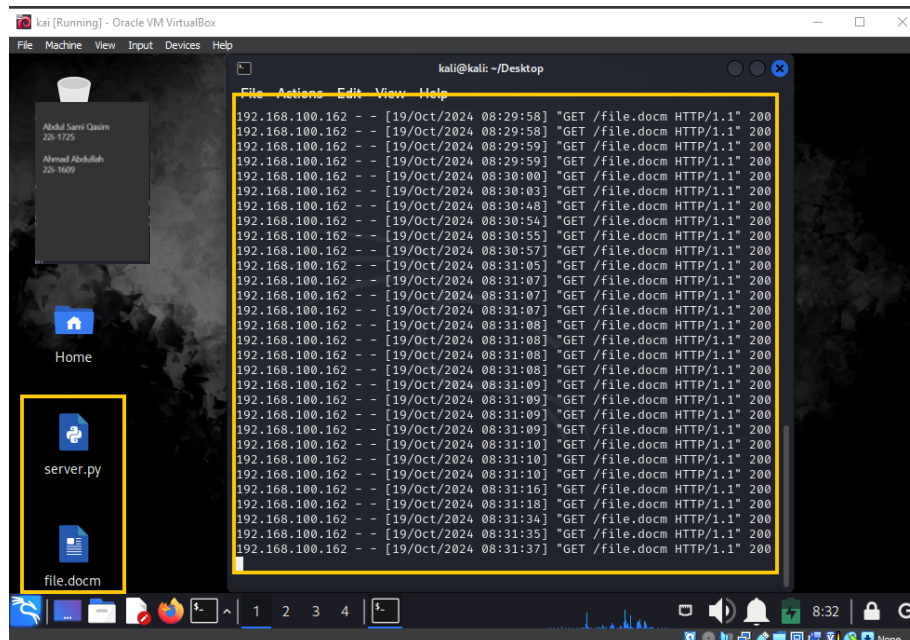
gnome@gnome: ~/Desktop/Server$ wget http://192.168.100.161:8000/file.docm
--2024-10-19 17:31:38-- http://192.168.100.161:8000/file.docm
Connecting to 192.168.100.161:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/vnd.ms-word]
Saving to: 'file.docm.23'

file.docm.23 [ <=> ] 69 --KB/s in 0s

2024-10-19 17:31:38 (7.09 MB/s) - 'file.docm.23' saved [69]

gnome@gnome: ~/Desktop/Server$
```

Hosted a server on attacking machine with the malicious file so that our Ubuntu should download it .



The image shows a Kali Linux desktop environment. On the left, there is a file manager window showing a folder named 'server.py' and a file named 'file.docm'. On the right, there is a terminal window titled 'kali@kali: ~/Desktop' showing a list of HTTP GET requests to http://192.168.100.162:8000/file.docm. The requests are all successful (200 OK) and are being received by the attacking machine.

```
kali@kali: ~/Desktop
192.168.100.162 - [19/Oct/2024 08:29:58] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:29:58] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:29:59] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:29:59] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:30:00] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:30:03] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:30:48] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:30:54] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:30:55] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:30:57] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:05] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:07] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:07] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:07] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:08] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:08] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:08] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:08] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:09] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:09] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:09] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:10] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:10] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:10] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:10] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:16] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:18] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:34] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:35] "GET /file.docm HTTP/1.1" 200
192.168.100.162 - [19/Oct/2024 08:31:37] "GET /file.docm HTTP/1.1" 200
```