In the following report, the four attacks we use are:
Attack 1. Use ismtp to perform RCPT TO attacks (username guessing) on Postfix server from another RPI or VM
Attack 2. Use the mail server to perform an open relay attack and send some email spam to our ns-public rpi email server (172.16.0.116)
Attack 3. Spoof an email from outside the organization to appear as if it were sent from an inside address and send it to another inside address
Attack 4. Run a brute-force dictionary attack against POP3 by using Nmap

Part 1

Attack 1

1. Describe the sequence used

```
$ ismtp -h 192.16.1.11:25 -e
/usr/share/wordlists/metasploit/ismtp_test.txt -l 2
```

The details of the above command are:
`-h <host>:` The target IP and port. Here, the IP of the email server on Ubuntu VM is 192.16.1.11 and the port is 25 by default.
`-e <file>:` Enable SMTP user enumeration testing and imports email list. We create an email list ismtp_test.txt locating in the /usr/share/wordlists/metasploit for the enumeration test.
`-l <1|2|3>:` Specifies enumeration type (1 = VRFY, 2 = RCPT TO, 3 = all). Here, we use RCPT TO attack, which is type 2.

2. Provide a python script that executes your attack

```
$ ismtp -h 192.16.1.11:25 -e
/usr/share/wordlists/metasploit/ismtp_test.txt -l 2
```

3. Provide *.pcap network capture file from traffic between attack host and target running the application server

The *.pcap file is attached as part_1_attack_1.pcap.

4. Describe the results of the attack

From the screenshot below, we can notice that the ‘ismtp’ command is using the username guessing method to do the username enumeration. The users on the server are marked as “valid”, while the users that are not on the email server are marked as “invalid”. Through this attack, we can determine the usernames currently on the server.

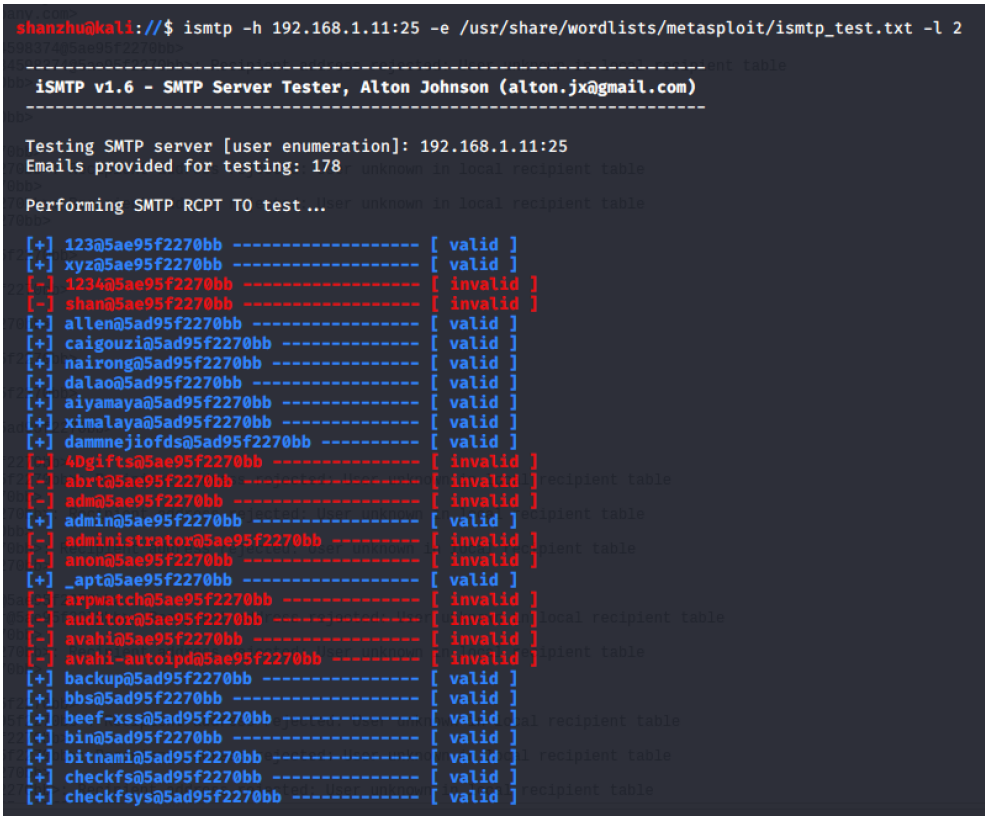

Fig 1. Results for Attack 1 with default configuration

Attack 2

1. Describe the sequence used

`telnet 172.16.0.116 25` *[telnet to the relay server]*
`HELO 172.16.0.116` *[tell the remote server of the client]*
********** *[system auto reply]*
`MAIL FROM:<dummy@172.16.0.116>` *[indicate sender address]*
****
`RCPT TO:<shanz@172.16.0.153>` *[indicate receiver address]*
****
`DATA` *[indicate the following content is data]*
****
`Subject:hi sansan` *[email content body]*
`This is a sample message.`
`.`
****
`QUIT` *[finished sending]*

2. Provide a python script that executes your attack
[See the file attack-2.py]

3. Provide *.pcap network capture file from traffic between attack host and target running the application server

The *.pcap file is attached as part_1_attack_2.pcap.

4. Describe the results of the attack
This attack is tested using ns-public open relay and sending spam to the email server of ourselves. And this attack is captured by suricata indicating the server indeed is under attack and by checking the email content, we can see the sender address is ending with the address of our ns public server address, which proves the success of this attack.

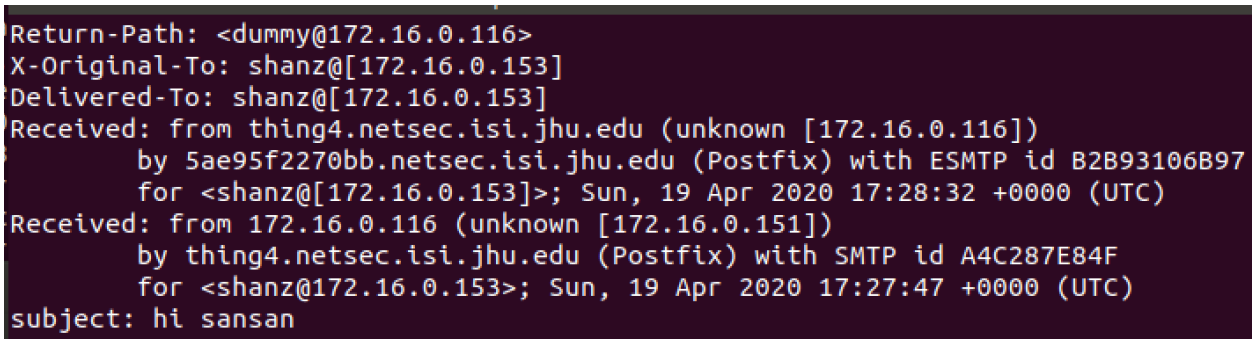

Fig 2. Results for Attack 2 with default configuration

Attack 3
1. Describe the sequence used

`telnet 172.16.107.136 25` *[telnet to the remote server outside of organization]*
`HELO 172.16.107.136` *[tell the remote server of the client address]*
********** *[system auto reply]*
`MAIL FROM:<dummy@172.16.0.116>` *[indicate sender address and this email would look like sending from inside the organization]*
****
`RCPT TO:<stern@172.16.0.146>` *[indicate receiver address]*
****
`DATA` *[indicate the following content is data]*
****
`Hey`, *[email content body]*
`This is a sample message.`
`.`
****
`QUIT` *[finished sending]*

2. Provide a python script that executes your attack
[see attack-3.py]

3. Provide *.pcap network capture file from traffic between attack host and target running the application server

The *.pcap file is attached as part_1_attack_3.pcap.

4. Describe the results of the attack
This attack successfully obtained the expected result, and the content in the below figure shows the accomplishment of spoofing email and using an internal address which appears to be sent from inside the organization. In detail, we can see, in the email it was showing received from 172.16.0.116, which is the internal relay we are using and the sender address 172.16.107.136 is the real ip address of the sender who is outside of the organization. Since the score of this email exceeds the threshold, it was tagged as “spam” and later by search suricata logs, we didn't find any matched records, which means the email spoofing attack is indeed filtered out by SpamAssassin.

```
Return-Path: <dummy@172.16.0.116>
X-Original-To: stern@172.16.0.146
Delivered-To: stern@[172.16.0.146]
Received: by 5ae95f2270bb.netsec.isi.jhu.edu (Postfix, from userid 1007)
        id C36A3101DBA; Tue, 21 Apr 2020 21:13:00 +0000 (UTC)
Received: from localhost by XnzhangX
        with SpamAssassin (version 3.4.2);
        Tue, 21 Apr 2020 21:13:00 +0000
Subject: *****SPAM*****
X-Spam-Checker-Version: SpamAssassin 3.4.2 (2018-09-13) on XnzhangX
X-Spam-Flag: YES
X-Spam-Level: ****
X-Spam-Status: Yes, score=4.5 required=3.0 tests=ALL_TRUSTED,MISSING_DATE,
        MISSING_FROM,MISSING_HEADERS,MISSING_MID,MISSING_SUBJECT,TVD_RCVD_IP,
        TVD_RCVD_IP4 autolearn=no autolearn_force=no version=3.4.2
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----------=_5E9F61DC.80FE71DD"
Message-Id: <20200421211300.C36A3101DBA@5ae95f2270bb.netsec.isi.jhu.edu>
Date: Tue, 21 Apr 2020 21:13:00 +0000 (UTC)
From: spamtester <dummy@172.16.0.116>

This is a multi-part message in MIME format.

------------=_5E9F61DC.80FE71DD
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system "XnzhangX",
has identified this incoming email as possible spam.  The original
message has been attached to this so you can view it or label
"1587503580.V41Ic1116M820005.XnzhangX" 66L, 2524C
```

Fig 3. Results for Attack 3 with SpamAssassin configuration

Attack 4

1. Describe the sequence used

Sequence we used: `sudo nmap -p110 --script pop3-brute 172.16.0.146`

- sudo run nmap with root privilege
- nmap call nmap program
- -p110 the port we are going to scan
- --script select to be a “using script” mode
- pop3-brute indicating we are using scripts for pop3 protocol
- 172.16.0.146 target IP address

2. Provide a python script that executes your attack

[See the file attack-4.py]

3. Provide *.pcap network capture file from traffic between attack host and target running the application server

The *.pcap file is attached as part_1_attack_4.pcap.

4. Describe the results of the attack

The result is shown below. From the following figure, we can justify our successful guess of the credentials on our email server, which matches the user created on the email server.

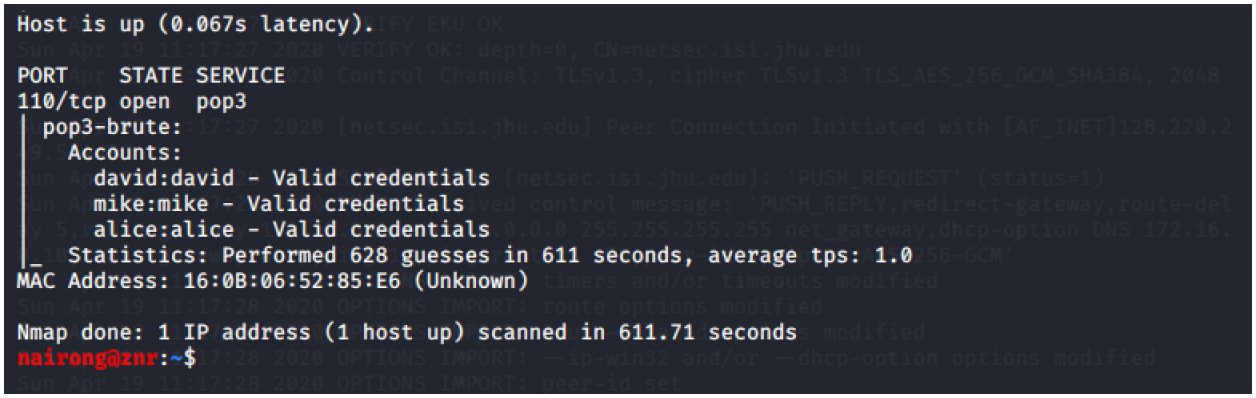

Fig 4. Results for Attack 4 with default configuration

Part 2

1. Provide the details for updated Postfix, other application, and OS settings

Basic modification:

(1) Disable VRFY (verify): The *VRFY command* is short for ‘verify’. It can be used to see if an email address is valid on the mail server. While this is great for troubleshooting, it also allows others to make educated guesses if an account exists and deliver

possibly spam. The VRFY command is not normally not needed for delivery between two mail servers. Thus, we will disable VRFY using:

```
$ postconf -e disable_vrfy_command=yes
```

(2) Network interfaces (inet_interfaces): The first setting to check is the interfaces Postfix is listening to. This setting is called **inet_interfaces** and by default configured with **all**. If you just want to relay messages to other systems, like sending outgoing emails, then there is no need to listen on all network interfaces. Configure Postfix to listen only on the local interface. This can be achieved by setting inet_interface to **loopback-only**.

```
$ postconf -e inet_interfaces=loopback-only
```

Prevent unwanted email relaying

An open relay is a system that accepts email from all systems and forwards them. Spammers use these open relays to send out their messages.

(1) Networks: Relaying is configured with several parameters. The first one is the **mynetworks** setting, which typically only includes the network addresses of the local network interface (lo).

```
$ mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
```

If you want to extend this list, simply add network segments or individual systems. Specify the related network mask, which is /32 for a single IPv4 address, or /128 for IPv6.

Due to the spaces in this setting, add quotes when using the postconf command.

```
$ postconf -e mynetworks="127.0.0.0/8 [::ffff:127.0.0.0]/104
[::1]/128"
```

(2) Domains: The second layer to define what emails to accept. This is called relaying and one of the options is by looking at the destination domain. The related setting for this is **relay_domains**, which specifies for which domains to accept email in the first place.

Incoming email configuration

(1) Enable HELO: Mail servers greet each other with a **HELO** command. Servers that are not using this are typically not properly configured, or simply sending spam. Thus, we will enable HELO using:

```
$ postconf -e smtpd_helo_required=yes
```

Outgoing email configuration

(1) Configure authenticated relaying with a smarthost: Typically only a few machines will accept incoming email and all other servers need to send out emails. So we can define which system(i.e. Relayhost in Postfix) will accept the email using postconf or edit the **main.cf** directly:

```
relayhost = [hostname]:587
```

(2) Enable SASL authentication:

```
# Enable SASL authentication
  smtp_sasl_auth_enable = yes
# Disallow any methods that do allow anonymous authentication
  smtp_sasl_security_options = noanonymous
# Define the sasl_passwd file location
  smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_passwd
```

(3) Edit the **/etc/postfix/sasl/sasl_passwd** file.

```
[mail.example.org]:587 username:password
```

This file can be parsed by postmap to created an optimized version, which is used as the database for lookups: `$ postmap /etc/postfix/sasl/sasl_passwd`

(4) Configuring encryption:

```
# Enable STARTTLS encryption
  smtp_use_tls = yes
# Location of CA certificates
  smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

Cryptography, encryption, and privacy

(1) Enable TLS logging: This is to see the details from TLS, increase the level of Postfix logging. Set **smtp_tls_loglevel** (outgoing) or **smtpd_tls_loglevel** (incoming) to the value one (1).

```
$ postconf -e smtp_tls_loglevel=1
```

2. Install SpamAssassin and enable spam filters to reject spam

1) Installing SpamAssassin

```
$ apt-get update
$ apt-get install spamassassin spamc
```

2) Adding SpamAssassin System User

```
$ adduser spamd --disabled-login
```

3) Editing SpamAssassin Configuration File

```
$ vim /etc/default/spamassassin
```

- First change the value of **ENABLED** directive from 0 to 1: `$ ENABLED = 1`
- Next, create a variable named `SAHOME` with the below value:
  `$ SAHOME="/var/log/spamassassin/"`
- Look for the OPTIONS directive and change its value as shown below:
  `$ OPTIONS="--create-prefs --max-children 5 --username spamd --helper-home-dir /home/spamd/ -s /home/spamd/spamd.log"`
- Also, in order for SpamAssassin to pick updates, we are going to set the CRON value to 1 so that the utility can download new rules automatically. `$ CRON =1`

4) Adding SpamAssassin Spam Rules

Next, we are going to add some spam rules. The file we are going to edit is located at /etc/spamassassin/local.cf.
`$ vim /etc/spamassassin/local.cf`
Uncomment the lines shown below and change the values as indicated:

```
rewrite_header Subject [***** SPAM _SCORE_ *****]
required_score          5.0
use_bayes               1
bayes_auto_learn        1
```

5) Editing Postfix

```
$ vim /etc/postfix/master.cf
```

- Look for the line: `$ smtp inet n - - - - smtpd.` Add `$  -o content_filter=spamassassin` below it.
- Add the line below to setup after queue content filter:

```
$ spamassassin unix -     n       n      -       -       pipe
   user=spamd argv=/usr/bin/spamc -f -e
   /usr/sbin/sendmail -oi -f ${sender} ${recipient}
```

3. Repeat the four attacks and provide *.pcap network capture file from traffic between attacks host and target running the application server.

The *.pcap files for the four attacks are attached in the attachment.

4. Describe the results of the four attacks

After we improved the configuration of Postfix and installed the SpamAssassin program to enhance the defence against the attacks for the email server, we repeated the four attacks again. The results are shown below:

Attack 1

Comparing the result below with Figure 1, which is the result of attack 1 with the default configuration of the email server, we can notice that there are more users marked as “invalid” now, which were marked as “valid” before. This indicates that the email server has protected more of its users and makes it harder for attackers to get the valid usernames under the server with the username enumeration attacks. Thus, our improvements have successfully enhanced the email server’s defense against username enumeration attack.

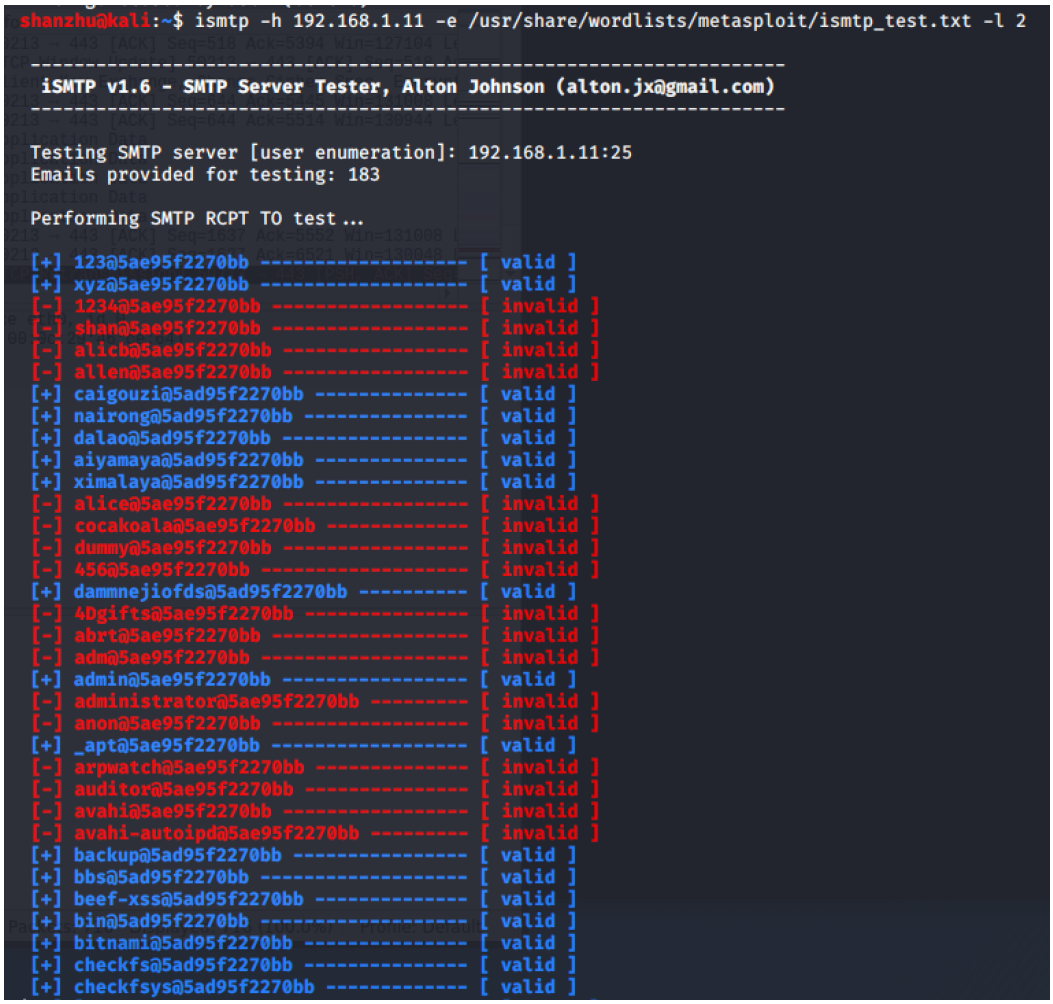

Fig 5. Results for Attack 1 with improved configuration

Attack 2

```
Return-Path: <postmaster@172.16.0.116>
X-Original-To: stern@[172.16.0.146]
Delivered-To: stern@[172.16.0.146]
Received: by 5ae95f2270bb.netsec.isi.jhu.edu (Postfix, from userid 1007)
        id 2C7E5101DC6; Tue, 21 Apr 2020 22:17:10 +0000 (UTC)
Received: from localhost by XnzhangX
        with SpamAssassin (version 3.4.2);
        Tue, 21 Apr 2020 22:17:10 +0000
Subject: *****SPAM*****
X-Spam-Checker-Version: SpamAssassin 3.4.2 (2018-09-13) on XnzhangX
X-Spam-Flag: YES
X-Spam-Level: ****
X-Spam-Status: Yes, score=4.5 required=4.5 tests=ALL_TRUSTED,MISSING_DATE,
        MISSING_FROM,MISSING_HEADERS,MISSING_MID,MISSING_SUBJECT autolearn=no
        autolearn_force=no version=3.4.2
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----------=_5E9F70E6.2A2E5ADF"
Message-Id: <20200421221710.2C7E5101DC6@5ae95f2270bb.netsec.isi.jhu.edu>
Date: Tue, 21 Apr 2020 22:17:10 +0000 (UTC)
From: spamtester <postmaster@172.16.0.116>

This is a multi-part message in MIME format.

------------=_5E9F70E6.2A2E5ADF
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system "XnzhangX",
has identified this incoming email as possible spam.  The original
message has been attached to this so you can view it or label
"1587507430.V41Ic113eM201194.XnzhangX" 66L, 2539C
```

Fig 6. Results for Attack 2 with improved configuration

Attack 3

```
Return-Path: <dummy@172.16.0.116>
X-Original-To: stern@172.16.0.146
Delivered-To: stern@[172.16.0.146]
Received: by 5ae95f2270bb.netsec.isi.jhu.edu (Postfix, from userid 1007)
        id C36A3101DBA; Tue, 21 Apr 2020 21:13:00 +0000 (UTC)
Received: from localhost by XnzhangX
        with SpamAssassin (version 3.4.2);
        Tue, 21 Apr 2020 21:13:00 +0000
Subject: *****SPAM*****
X-Spam-Checker-Version: SpamAssassin 3.4.2 (2018-09-13) on XnzhangX
X-Spam-Flag: YES
X-Spam-Level: ****
X-Spam-Status: Yes, score=4.5 required=3.0 tests=ALL_TRUSTED,MISSING_DATE,
        MISSING_FROM,MISSING_HEADERS,MISSING_MID,MISSING_SUBJECT,TVD_RCVD_IP,
        TVD_RCVD_IP4 autolearn=no autolearn_force=no version=3.4.2
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----------=_5E9F61DC.80FE71DD"
Message-Id: <20200421211300.C36A3101DBA@5ae95f2270bb.netsec.isi.jhu.edu>
Date: Tue, 21 Apr 2020 21:13:00 +0000 (UTC)
From: spamtester <dummy@172.16.0.116>

This is a multi-part message in MIME format.

------------=_5E9F61DC.80FE71DD
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system "XnzhangX",
has identified this incoming email as possible spam.  The original
message has been attached to this so you can view it or label
"1587503580.V41Ic1116M820005.XnzhangX" 66L, 2524C
```

Fig 7. Results for Attack 3 with SpamAssassin configuration

Attack 4

```
PORT    STATE SERVICE
110/tcp open  pop3
| pop3-brute:
|   Accounts:
|     mike:mike - Valid credentials
|     david:david - Valid credentials
|     alice:alice - Valid credentials
|_  Statistics: Performed 623 guesses in 611 seconds, average tps: 1.0
MAC Address: 16:0B:06:52:85:E6 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 610.95 seconds
```

Fig 8. Results for Attack 4 with SpamAssassin configuration

Part 3

1. Provide the details used to install and configure Suricata so that it could be replicated by someone else.

Installation

First, we install Suricata in a Docker container on the Ubuntu server following the steps:

1) Build the Docker image using: `$ sudo docker build -t tsuricata .`
2) Build the Docker container using: `$ sudo docker run -d --name suricata --privileged -v /sys/fs/cgroup:/sys/fs/cgroup:ro --network host tsuricata:latest`
3) Run the Docker container using: `$ sudo docker exec -it suricata bash`

Configuration

After the installation, we will have Suricata rules under `/etc/suricata/rules`/ and the main configuration file under `/etc/suricata/suricata.yaml`. Next step is to configure the Suricata in the Docker container for our usage.

4) Configure Suricata to differentiate between your internal network to be protected and the external network in the configuration file. This can be done by defining the correct values for the `HOME_NET` and `EXTERNAL_NET` variables respectively under the address groups. We put our IP address in `HOME_NET` and `"!$HOME_NET"` for `EXTERNAL_NET` as default:

```
$ vim /etc/suricata/suricata.yaml
      HOME_NET: "[172.16.0.0/16]" ($Host IP Address Range$)
  ...
      EXTERNAL_NET: "!$HOME_NET"
      Af-packet:
           Interface: tap0
```

5) Next, we want to create a .rules file to set up the regulations for Suricata to detect the possible SYN flood:

```
$ vim /etc/suricata/rules/test-smtp.rules
```

And put these in the file:

```
alert tcp any any -> $HOME_NET 25 (msg: "Possible Spam"; flags: S; flow: stateless; threshold: type both, track by_dst, count 1, seconds 1; sid:1000001; rev:1;)
```

The rule basically fires when there are 1 connection to the local network with port 25 in 1 seconds. The parameter can be changed accordingly.

6) Next, we need to edit the Suricata configuration file to configure Suricata to include this rule:
   a) Change the default rule path to: `/etc/suricata/rules`
   b) Add the rules file under the `rule-files:` section:

```
$ vim /etc/suricata/suricata.yaml
```

```
Default rule path: /etc/suricata/rules
rule-files:
# - Custom Test rules
 - test-smtp.rules
```

Running Suricata to Detect Possible DDoS Attack

7) After configuration, we firstly need to check our network interface using: `$ ifconfig`
8) Then running the Suricata to detect possible DDos Attack:
   `$ suricata -c /etc/suricata/suricata.yaml -i tap0`(The interface get from last step)

2. Provide reporting details from Suricata that show it detected the attack

Attack 1:
Results are shown below, suricata indeed captured the possible SMTP attack when running username enumeration attacks from Kali on the email server on port 25 and recorded in the log file.

```
root@ubuntu:/var/log/suricata# tail -f /var/log/suricata/fast.log
04/19/2020-15:25:44.552496  [**] [1:1000001:1] Possible SMTP attack [**] [Classification: (null)]
 [Priority: 3] {TCP} 192.168.1.12:57924 -> 192.168.1.11:25
04/19/2020-15:25:49.596871  [**] [1:1000001:1] Possible SMTP attack [**] [Classification: (null)]
 [Priority: 3] {TCP} 192.168.1.12:57926 -> 192.168.1.11:25
04/19/2020-15:29:25.188291  [**] [1:1000001:1] Possible SMTP attack [**] [Classification: (null)]
 [Priority: 3] {TCP} 192.168.1.12:57928 -> 192.168.1.11:25
04/19/2020-15:29:30.211365  [**] [1:1000001:1] Possible SMTP attack [**] [Classification: (null)]
 [Priority: 3] {TCP} 192.168.1.12:57930 -> 192.168.1.11:25
04/19/2020-15:35:36.252871  [**] [1:1000001:1] Possible SMTP attack [**] [Classification: (null)]
 [Priority: 3] {TCP} 192.168.1.12:58264 -> 192.168.1.11:25
04/19/2020-15:35:41.301212  [**] [1:1000001:1] Possible SMTP attack [**] [Classification: (null)]
 [Priority: 3] {TCP} 192.168.1.12:58266 -> 192.168.1.11:25
```

Fig 9. Suricata Log for the Attack 1

Attack 2:
Results are shown below, suricata indeed captured the incoming spam on port 25 and recorded in the log file.

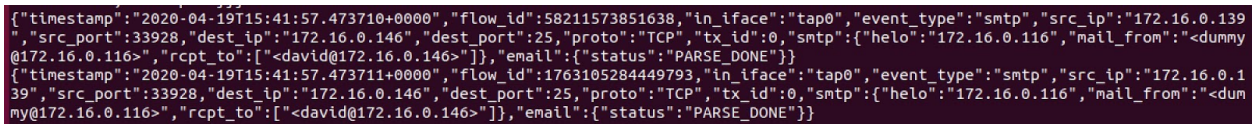
{"timestamp":"2020-04-19T15:41:57.473710+0000","flow_id":58211573851638,"in_iface":"tap0","event_type":"smtp","src_ip":"172.16.0.139","src_port":33928,"dest_ip":"172.16.0.146","dest_port":25,"proto":"TCP","tx_id":0,"smtp":{"helo":"172.16.0.116","mail_from":"<dummy@172.16.0.116>","rcpt_to":["<david@172.16.0.146>"]},"email":{"status":"PARSE_DONE"}}
{"timestamp":"2020-04-19T15:41:57.473711+0000","flow_id":1763105284449793,"in_iface":"tap0","event_type":"smtp","src_ip":"172.16.0.139","src_port":33928,"dest_ip":"172.16.0.146","dest_port":25,"proto":"TCP","tx_id":0,"smtp":{"helo":"172.16.0.116","mail_from":"<dummy@172.16.0.116>","rcpt_to":["<david@172.16.0.146>"]},"email":{"status":"PARSE_DONE"}}

Fig 10. Suricata Log for the Attack 2

Attack 3:

We repeated Attack 3 and in order to bring about any contrast after we improved Postfix setting and installed SpamAssassin, we performed email spoofing attack under the improved configuration. From the results shown above, the email was tagged as spam, and we searched through fast.log and eve.json, there are no such detection records, which in other words proved the effective usage of SpamAssassin who filters out any suspicious email.

Attack 4:

Results are shown below, surcata indeed captured a sequence of connections on port 110 which indicated multiple penetration guesses by nmap.


{"timestamp":"2020-04-19T15:41:33.007959+0000","flow_id":650412413435897,"in_iface":"tap0","event_type":"flow","src_ip":"172.16.0.139","src_port":39052,"dest_ip":"172.16.0.146","dest_port":110,"proto":"TCP","app_proto":"ftp","app_proto_tc":"failed","flow":{"pkts_toserver":10,"pkts_toclient":8,"bytes_toserver":704,"bytes_toclient":611,"start":"2020-04-19T15:40:12.671737+0000","end":"2020-04-19T15:40:32.730179+0000","age":20,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1b","tcp_flags_ts":"1b","tcp_flags_tc":"1b","syn":true,"fin":true,"psh":true,"ack":true,"state":"closed"}}
{"timestamp":"2020-04-19T15:41:33.007991+0000","flow_id":932147235645235,"in_iface":"tap0","event_type":"flow","src_ip":"172.16.0.139","src_port":39048,"dest_ip":"172.16.0.146","dest_port":110,"proto":"TCP","app_proto":"ftp","app_proto_tc":"failed","flow":{"pkts_toserver":9,"pkts_toclient":7,"bytes_toserver":626,"bytes_toclient":540,"start":"2020-04-19T15:40:11.649011+0000","end":"2020-04-19T15:40:32.072144+0000","age":21,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1b","tcp_flags_ts":"1b","tcp_flags_tc":"1b","syn":true,"fin":true,"psh":true,"ack":true,"state":"closed"}}
{"timestamp":"2020-04-19T15:41:33.011595+0000","flow_id":880818081531733,"in_iface":"tap0","event_type":"flow","src_ip":"172.16.0.139","src_port":39050,"dest_ip":"172.16.0.146","dest_port":110,"proto":"TCP","app_proto":"ftp","app_proto_tc":"failed","flow":{"pkts_toserver":9,"pkts_toclient":7,"bytes_toserver":625,"bytes_toclient":540,"start":"2020-04-19T15:40:12.558933+0000","end":"2020-04-19T15:40:32.753369+0000","age":20,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1b","tcp_flags_ts":"1b","tcp_flags_tc":"1b","syn":true,"fin":true,"psh":true,"ack":true,"state":"closed"}}
{"timestamp":"2020-04-19T15:41:33.011661+0000","flow_id":567534438495 76,"in_iface":"tap0","event_type":"flow","src_ip":"172.16.0.139","src_port":39054,"dest_ip":"172.16.0.146","dest_port":110,"proto":"TCP","app_proto":"ftp","app_proto_tc":"failed","flow":{"pkts_toserver":9,"pkts_toclient":7,"bytes_toserver":627,"bytes_toclient":540,"start":"2020-04-19T15:40:12.673128+0000","end":"2020-04-19T15:40:32.705442+0000","age":20,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1b","tcp_flags_ts":"1b","tcp_flags_tc":"1b","syn":true,"fin":true,"psh":true,"ack":true,"state":"closed"}}
{"timestamp":"2020-04-19T15:41:33.011690+0000","flow_id":1347481310609409,"in_iface":"tap0","event_type":"flow","src_ip":"172.16.0.139","src_port":39052,"dest_ip":"172.16.0.146","dest_port":110,"proto":"TCP","app_proto":"ftp","app_proto_tc":"failed","flow":{"pkts_toserver":10,"pkts_toclient":8,"bytes_toserver":704,"bytes_toclient":611,"start":"2020-04-19T15:40:12.671745+0000","end":"2020-04-19T15:40:32.730195+0000","age":20,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1b","tcp_flags_ts":"1b","tcp_flags_tc":"1b","syn":true,"fin":true,"psh":true,"ack":true,"state":"closed"}}
{"timestamp":"2020-04-19T15:41:33.011715+0000","flow_id":1777270097962804,"in_iface":"tap0","event_type":"flow","src_ip":"172.16.0.139","src_port":39048,"dest_ip":"172.16.0.146","dest_port":110,"proto":"TCP","app_proto":"ftp","app_proto_tc":"failed","flow":{"pkts_toserver":9,"pkts_toclient":7,"bytes_toserver":626,"bytes_toclient":540,"start":"2020-04-19T15:40:11.649012+0000","end":"2020-04-19T15:40:32.072150+0000","age":21,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1b","tcp_flags_ts":"1b","tcp_flags_tc":"1b","syn":true,"fin":true,"psh":true,"ack":true,"state":"closed"}}
{"timestamp":"2020-04-19T15:41:34.007999+0000","flow_id":1543671121724884,"in_iface":"tap0","event_type":"flow","src_ip":"172.16.0.139","src_port":39056,"dest_ip":"172.16.0.146","dest_port":110,"proto":"TCP","app_proto":"ftp","app_proto_tc":"failed","flow":{"pkts_toserver":10,"pkts_toclient":8,"bytes_toserver":704,"bytes_toclient":611,"start":"2020-04-19T15:40:12.673236+0000","end":"2020-04-19T15:40:33.197122+0000","age":21,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1b","tcp_flags_ts":"1b","tcp_flags_tc":"1b","syn":true,"fin":true,"psh":true,"ack":true,"state":"closed"}}

Fig 11. Suricata Log for the Attack 4

Part 4

1. Discuss what else could be performed to defend against these attacks

    1) Defend against account enumeration attack: Disable VRFY and EXPN or set up email firewalls to limit commands to specific hosts on your network.
    2) Defend against SMTP relay attacks:

- First, we can disable SMTP relay on our email server especially when we don't need SMTP relay, or we can enable SMTP relay for specific hosts on the server or within your firewall configuration.

- Second, we can enable authentication on our email server. We can require password authentication on an email address that matches the email server's domain.

3) Defend against spoofed email attack:
    - Training users: Defending against email spoofing requires a multilayered approach to security. Users, often the weakest link, must be empowered with knowledge and best practices that can help them know how to spot phishing and email spoofing attacks.

    - Attachment Protect. Use software to scan every attachment, search for malicious code, and block users from clicking on malicious links or opening malicious attachments. Suspicious files can be sandboxed or rewritten to a format that enables users to safely access it.

    - Impersonation Protect. Use software to perform a deep scan on all inbound emails to search for header anomalies, domain similarity and specific keywords that may be signs of spoofing.

    - Email filters that use DNS authentication services like SPF, DKIM and DMARC can help to lock potentially fraudulent email.

4) Defend against brute-force dictionary attack using Nmap:

    - Brute force attack can be powerful when some of the POP3 users have weak passwords, and attackers may succeed and get access to the user's mailbox. We can install some third-party applications to protect the POP3 server such as RdpGuard. RdpGuard effectively protects your POP3 server from brute-force attacks by working with POP3 ports or logs on your server to detect failed login attempts. If the number of failed login attempts from a single IP address reaches a set limit, the attacker's IP address will be blocked.

    - Forcing users to set strong passwords is also useful. Security is often rated based on how long it would take an attacker to break it using brute force methods. The time and complexity to break a password grows exponentially because every new character adds 95 possible letters, numbers, and special characters

- Put limits in place that prevent multiple login attempts. This can be done in several ways: Lock an account out after a certain number of failed attempts; Use two-factor authentication so that more than a password is required to log in; If there are multiple attempts from a single IP address, lock this address.

- Finally, store the password securely. A strong encryption is not safe enough to defend brute force attacks because attackers will try to find what password can get them to the database storing all passwords.

2. SMTP is not a secure protocol. Discuss some of the secure protocol alternatives available today.

1) Among all the possible secure protocol alternatives, Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS), are the most common email security protocols. Both of them are application layer protocols, and they provide some security frameworks that work with SMTP to secure email communication.
2) SMTPS (Simple Mail Transfer Protocol Secure) is a method for securing the SMTP using transport layer security. Similar to the way that HTTPS enhances HTTP using SSL (TLS), SMTPS secure SMTP at the transport layer by wrapping SMTP inside TLS. SMTPS provides authentication services and also ensures confidentiality and data integrity among the communication of emails.
3) Sender Policy Framework (SPF) is an authentication protocol that theoretically protects against domain spoofing. It authenticates domains and prevents those domains from being spoofed. SPF can also help a mail server identify whether a mail is sent through a legal domain or the domain is actually used by someone else to hide their true identity.
4) DomainKeys Identified Mail (DKIM) is an extended version of SPF that secures and focuses on the safety of email in transit. It first uses digital signatures to check that the email was sent by a specific domain, and then traces and checks if the domain indeed sent the email.
5) Domain-Based Message Authentication, Reporting, and Conformance (DMARC) is an authentication protocol that checks the validity of SPF and DKIM and also instructs the provider on handling incoming messages.
6) Secure/Multipurpose Internet Mail Extensions (S/MIME) is an end-to-end encryption protocol. It is designed to encrypt email contents but not other parts of the email such as sender and recipient. The protocol requires a digital certificate such that only the receiver is capable of decrypting your email.
7) Pretty Good Privacy (PGP) is a two-way encryption and end-to-end encryption protocol. Its open-source version is OpenPGP. Working similar as S/MIME, PGP only encrypts the contents. OpenPGP gets timely updates, and it’s free to use.