DE MONTFORT
UNIVERSITY
LEICESTER

Linux Hardening Against Heartbleed and Shellshock Vulnerabilities

Module:	Linux Security (CTEC-2912)
Academic Year:	2021/2022
Student Name (Number):	Victoria Etchells (p2629898)
Date:	April 2022

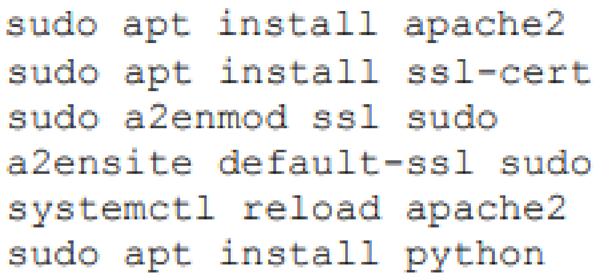
Table of Contents

1 Introduction

Scope of the work and report:

To mitigate against two known bugs, the heartbleed bug and the shellshock bug. These are the prerequisites before starting the report:

Figure.1.1.1 – Install this software.



```
sudo apt install apache2
sudo apt install ssl-cert
sudo a2enmod ssl sudo
a2ensite default-ssl sudo
systemctl reload apache2
sudo apt install python
```

There is more software that I have utilised which is detailed in the relevant sections of the report.

There are two scripts to run to test proof of concept when implementing the remedies chosen and screenshots have been provided throughout. Each must be succinctly described with reasons given for the chosen technique.

There are four tasks to complete for each vulnerability and are as follows:

Task 1 – Vulnerability Explaination.

Task 2 – iptables Firewall Rules

Task 3 – Alarm Trigger Technique

Task 4 – Additional Remediation

Max Word Count Allowed: 3000

Ubuntu IP Address (For proof-of-concept tests): 10.0.2.8

Systems using for hardening and exploits: Ubuntu 16.04.06 VM for the hardening component

Kali-Linux-2021 VM for the testing component

Both VMs are created on the same NAT network for ease. I am using VirtualBox to run the virtual machines from.

2 Heartbleed Vulnerability

2.1 Task 1 – Vulnerability Description

The heartbleed bug was discovered in March of 2012 with the OpenSSL release 1.0.1 and works by exploiting the memory and is considered an extremely dangerous bug should it be utilised by a threat actor effectively. [1]

The Heartbleed Bug aptly named due to a flaw within OpenSSL's Heartbeat Extension designed for the TLS and DTLS protocols. The extension itself assesses TLS/DTLS secure communication links via the heartbeat request message method, the receiver must send back the exact same message. [3] (see RFC6520 for more information) [1]

Heartbleed came about due to a bug not being picked up during creation. It can be exploited in both TLS server and client sessions. It is classified as a buffer over-read meaning data becomes vulnerable from the target system including private keys, customer data, passwords, and session cookies, and is facilitated by improper bounds checking.

This report will be looking at some mitigations techniques to secure against an attack via Heartbleed, including but not limited to; Checking current OpenSSL version, updating the IP Tables rules, monitor traffic over the network. [4]

Heartbleed Vulnerability	
CVE	CVE-2014-0160
NIST Rating:	7.5 out of 10
Date Discovered:	12/09/2014
Date Disclosed:	April 2014
Vulnerable Version:	OpenSSL 1.0.1 through 1.0.1f
Founder:	Neel Mehta
Vulnerable Since:	February 2012
Attack Vectors:	TLS Client & Server Sessions
Possible Mitigations:	IP Table Rules, Traffic Monitoring, Honeypot
Targets Exploited by:	Memory buffer, payload as a message, improper bounds checking

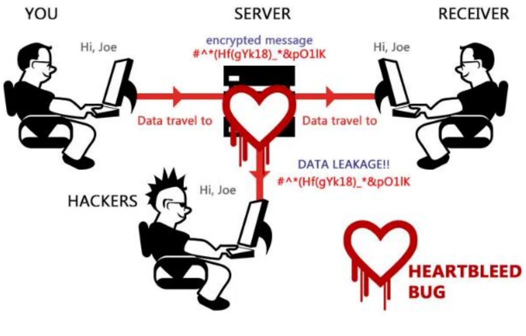


Figure.2.1.1 Depicting heartbleed [5]

2.2 Task 2 – iptables Firewall Rules

Prerequisite Commands:

$ sudo apt install apache2

$ sudo apt install ssl-cert

$ sudo a2enmod ssl

$ sudo a2ensite default-ssl

$ sudo service apache2 reload

$ sudo apt install python

Install Logwatch & mailutils for monitoring of all mitigations and subsequent exploit attempts:

$ sudo apt install logwatch

$ sudo apt install mailutils

Configure Logwatch to send a daily report [15]:

$ sudo nano /usr/share/logwatch/default.conf/logwatch.conf

Here you can set up automatic daily alerts, change the following:

MailTo=root **NB:** This can be any email you want.

Range=All **NB:** Set to all to get everything.

Detail=High **NB:** This can be Low, Medium, or High.

NB: These options selected because it is on local only configuration, to check for mail use**:** *$ sudo mail*

NB: This will show you any mail you have,

Service=All

Command to get a report:

$ logwatch --detail Med --mailto ADDRESS --service all --range today

Check existing rules: *$ iptables -L -n -v*

Delete current firewall rules: *$ sudo iptables -F*

Install Persistent: *$ sudo apt install iptables-persistent*

Save Firewall Rules: *$ sudo netfilter-persistent save*

Rules implemented[3] - Specific rules to heartbleed:

Add log rule:

$ iptables -t filter -A INPUT -p tcp --dport 443 -m u32 --u32 "52=0x18030000:0x1803FFFF" -j LOG --log-prefix "BLOCKED: HEARTBEAT"

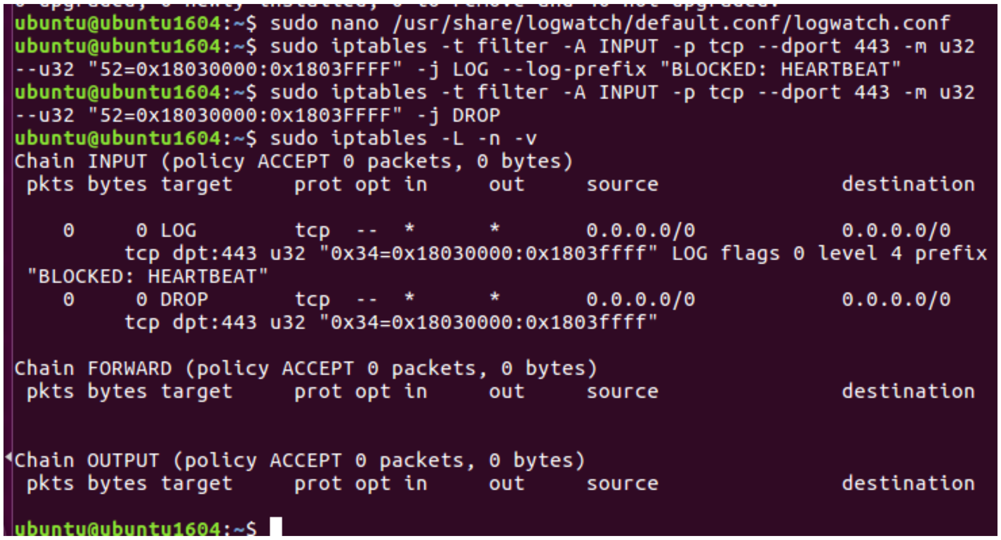
This rule is added to the INPUT table to enable logging of any heartbleed attack, selecting tcp protocol with the destination port selected as 443 and passing in additional module u32, then, checks against the bytes starting at 52, if these are 1803 then it returns a TRUE output and logs it using -j to select LOG.

Add block rules:

$ iptables -t filter -A INPUT -p tcp --dport 443 -m u32 --u32 "52=0x18030000:0x1803FFFF" -j DROP

This rule is added to the INPUT table to enable dropping of any heartbleed attack, selecting tcp protocol with the destination port selected as 443 and passing in additional module u32, then, checks against the bytes starting at 52, if these are 1803 then it returns a TRUE output and drops it using -j to select the DROP option.

Figure 2.2.1 – Screenshot of rules

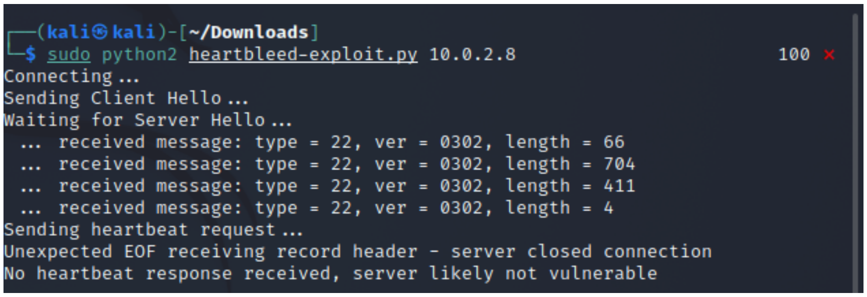


```
ubuntu@ubuntu1604:~$ sudo nano /usr/share/logwatch/default.conf/logwatch.conf
ubuntu@ubuntu1604:~$ sudo iptables -t filter -A INPUT -p tcp --dport 443 -m u32
--u32 "52=0x18030000:0x1803FFFF" -j LOG --log-prefix "BLOCKED: HEARTBEAT"
ubuntu@ubuntu1604:~$ sudo iptables -t filter -A INPUT -p tcp --dport 443 -m u32
--u32 "52=0x18030000:0x1803FFFF" -j DROP
ubuntu@ubuntu1604:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 LOG        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
         tcp dpt:443 u32 "0x34=0x18030000:0x1803ffff" LOG flags 0 level 4 prefix
 "BLOCKED: HEARTBEAT"
    0     0 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
         tcp dpt:443 u32 "0x34=0x18030000:0x1803ffff"

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

ubuntu@ubuntu1604:~$
```

Check the vulnerability with Kali:

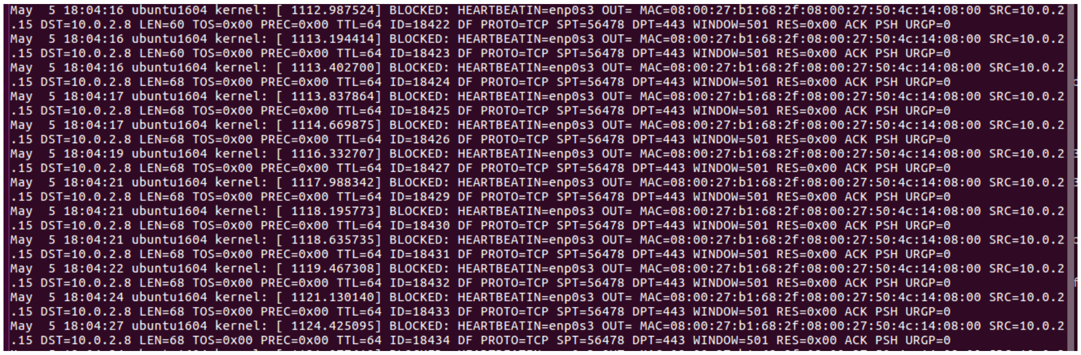
$ sudo python2 heartbleed-exploit.py 10.0.2.8

Figure.2.2.2 Screenshot of Exploit



```
┌──(kali㉿kali)-[~/Downloads]
└─$ sudo python2 heartbleed-exploit.py 10.0.2.8                        100 ✕
Connecting...
Sending Client Hello...
Waiting for Server Hello...
 ... received message: type = 22, ver = 0302, length = 66
 ... received message: type = 22, ver = 0302, length = 704
 ... received message: type = 22, ver = 0302, length = 411
 ... received message: type = 22, ver = 0302, length = 4
Sending heartbeat request...
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
```

Figure.2.2.3 – kern.log – blocked heartbleed request



```
May  5 18:04:16 ubuntu1604 kernel: [ 1112.987524] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=18422 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:16 ubuntu1604 kernel: [ 1113.194414] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=18423 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:16 ubuntu1604 kernel: [ 1113.402700] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18424 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:17 ubuntu1604 kernel: [ 1113.837864] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18425 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:17 ubuntu1604 kernel: [ 1114.669875] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18426 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:19 ubuntu1604 kernel: [ 1116.332707] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18427 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:21 ubuntu1604 kernel: [ 1117.988342] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18429 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:21 ubuntu1604 kernel: [ 1118.195773] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18430 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:21 ubuntu1604 kernel: [ 1118.635735] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18431 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:22 ubuntu1604 kernel: [ 1119.467308] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18432 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:24 ubuntu1604 kernel: [ 1121.130140] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18433 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
May  5 18:04:27 ubuntu1604 kernel: [ 1124.425095] BLOCKED: HEARTBEATIN=enp0s3 OUT= MAC=08:00:27:b1:68:2f:08:00:27:50:4c:14:08:00 SRC=10.0.2
.15 DST=10.0.2.8 LEN=68 TOS=0x00 PREC=0x00 TTL=64 ID=18434 DF PROTO=TCP SPT=56478 DPT=443 WINDOW=501 RES=0x00 ACK PSH URGP=0
```

2.3 Task 3 – Alarm Triggering

Snort Alarm Technique [3]:

This set of rules are set up as an alert to discover any TCP connections that may be transmitting from an external network. It is set up to specify the direction of traffic, output a message related to the heartbleed bug, to search packets for stated information, set out the required parameters, sets the starting point for snort, and tells snort the type of attack we are searching for. Snort offers three modes; sniffer, packet logger and Network Intrusion Detection System(NIDS) mode, and it is open source, meaning costs can be kept to a minimum while having in place a well-known system to alert of any attacks, in addition to this, Snort has a large online community whereby questions can be posed for any issues you might have to further develop your knowledge on the software.

Check Rules Existing: $ /etc/snort/rules

Rule for SSLv3:

```
alert tcp any any -> any any (msg:" Alert: Large Heartbeat Response"; flow:established,to_client; content:"|18 03 00|"; depth: 3; byte_test:2, >, 200, 3, big; byte_test:2, <, 16385, 3, big; threshold:type limit, track by_src, count 1, seconds 600; reference:cve,2014-0160; classtype:bad-unknown; sid: 1000000; rev:4;)
```

Rule for TLSv1:

```
alert tcp any [!80,!445] -> any [!80,!445] (msg:"FOX-SRT - Suspicious - TLSv1 Large Heartbeat Response"; flow:established,to_client; content:"|18 03 01|"; depth: 3; byte_test:2, >, 200, 3, big; byte_test:2, <, 16385, 3, big; threshold:type limit, track by_src, count 1, seconds 600; reference:cve,2014-0160; classtype:bad-unknown; sid: 1000001; rev:4;)
```

Rule for TLSv1.1:

```
alert tcp any [!80,!445] -> any [!80,!445] (msg:"FOX-SRT - Suspicious - TLSv1.1 Large Heartbeat Response"; flow:established,to_client; content:"|18 03 02|"; depth: 3; byte_test:2, >, 200, 3, big; byte_test:2, <, 16385, 3, big; threshold:type limit, track by_src, count 1, seconds 600; reference:cve,2014-0160; classtype:bad-unknown; sid: 1000002; rev:4;)
```

Rule for TLSv1.2:

```
alert tcp any [!80,!445] -> any [!80,!445] (msg:"FOX-SRT - Suspicious - TLSv1.2 Large Heartbeat Response"; flow:established,to_client; content:"|18 03 03|"; depth: 3; byte_test:2, >, 200, 3, big; byte_test:2, <, 16385, 3, big; threshold:type limit, track by_src, count 1, seconds 600; reference:cve,2014-0160; classtype:bad-unknown; sid: 1000003; rev:4;)
```

Table of Snort commands and descriptions:

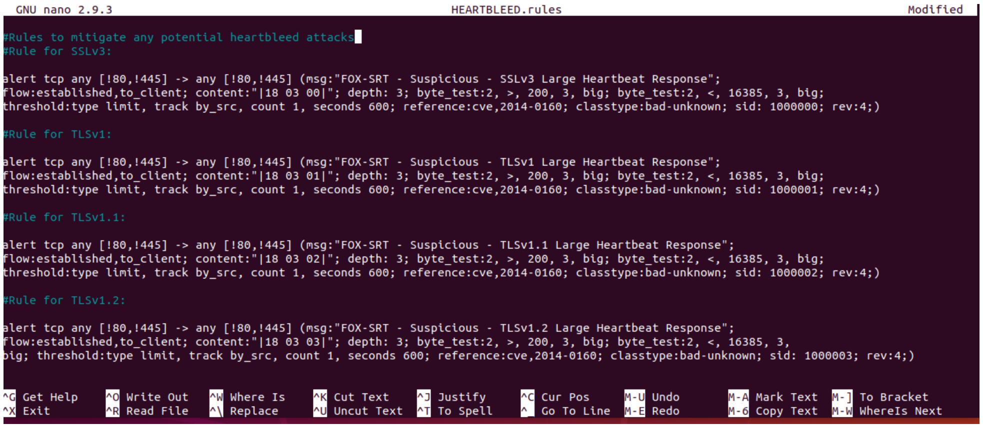
Command	**Description**
->	Traffic direction specification
Msg	Message displayed when alerting, this can be anything you require
Content	Packet search for specified content
Depth	Determines the intensity of the analysis
Offset	Determines the starting point of alert within the packet
Classtype	Attack identifier for the alert
Sid:115	Rule identifier

Commands used:

$ cd /etc/snort/rules

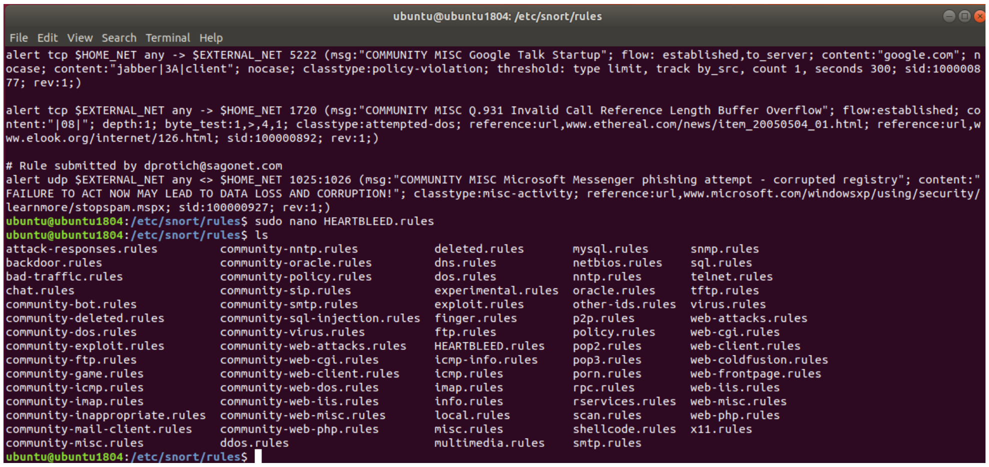
$ sudo nano heartbleed.rules

Figure 2.3.1 Write Rules & Save



```
  GNU nano 2.9.3                                   HEARTBLEED.rules                                   Modified

#Rules to mitigate any potential heartbleed attacks
#Rule for SSLv3:

alert tcp any [!80,!445] -> any [!80,!445] (msg:"FOX-SRT - Suspicious - SSLv3 Large Heartbeat Response";
flow:established,to_client; content:"|18 03 00|"; depth: 3; byte_test:2, >, 200, 3, big; byte_test:2, <, 16385, 3, big;
threshold:type limit, track by_src, count 1, seconds 600; reference:cve,2014-0160; classtype:bad-unknown; sid: 1000000; rev:4;)

#Rule for TLSv1:

alert tcp any [!80,!445] -> any [!80,!445] (msg:"FOX-SRT - Suspicious - TLSv1 Large Heartbeat Response";
flow:established,to_client; content:"|18 03 01|"; depth: 3; byte_test:2, >, 200, 3, big; byte_test:2, <, 16385, 3, big;
threshold:type limit, track by_src, count 1, seconds 600; reference:cve,2014-0160; classtype:bad-unknown; sid: 1000001; rev:4;)

#Rule for TLSv1.1:

alert tcp any [!80,!445] -> any [!80,!445] (msg:"FOX-SRT - Suspicious - TLSv1.1 Large Heartbeat Response";
flow:established,to_client; content:"|18 03 02|"; depth: 3; byte_test:2, >, 200, 3, big; byte_test:2, <, 16385, 3, big;
threshold:type limit, track by_src, count 1, seconds 600; reference:cve,2014-0160; classtype:bad-unknown; sid: 1000002; rev:4;)

#Rule for TLSv1.2:

alert tcp any [!80,!445] -> any [!80,!445] (msg:"FOX-SRT - Suspicious - TLSv1.2 Large Heartbeat Response";
flow:established,to_client; content:"|18 03 03|"; depth: 3; byte_test:2, >, 200, 3, big; byte_test:2, <, 16385, 3,
big; threshold:type limit, track by_src, count 1, seconds 600; reference:cve,2014-0160; classtype:bad-unknown; sid: 1000003; rev:4;)

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text  M-] To Bracket
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text  M-W WhereIs Next
```

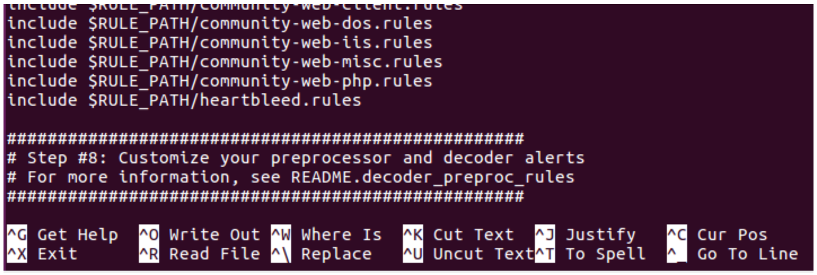
Figure 2.3.2 Check File Saved in /etc/snort/rules



ubuntu@ubuntu1804: /etc/snort/rules
File Edit View Search Terminal Help

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 5222 (msg:"COMMUNITY MISC Google Talk Startup"; flow: established,to_server; content:"google.com"; n
ocase; content:"jabber|3A|client"; nocase; classtype:policy-violation; threshold: type limit, track by_src, count 1, seconds 300; sid:1000008
77; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 1720 (msg:"COMMUNITY MISC Q.931 Invalid Call Reference Length Buffer Overflow"; flow:established; co
ntent:"|08|"; depth:1; byte_test:1,>,4,1; classtype:attempted-dos; reference:url,www.ethereal.com/news/item_20050504_01.html; reference:url,w
ww.elook.org/internet/126.html; sid:100000892; rev:1;)

# Rule submitted by dprotich@sagonet.com
alert udp $EXTERNAL_NET any <> $HOME_NET 1025:1026 (msg:"COMMUNITY MISC Microsoft Messenger phishing attempt - corrupted registry"; content:"
FAILURE TO ACT NOW MAY LEAD TO DATA LOSS AND CORRUPTION!"; classtype:misc-activity; reference:url,www.microsoft.com/windowsxp/using/security/
learnmore/stopspam.mspx; sid:100000927; rev:1;)
ubuntu@ubuntu1804:/etc/snort/rules$ sudo nano HEARTBLEED.rules
ubuntu@ubuntu1804:/etc/snort/rules$ ls
attack-responses.rules         community-nntp.rules           deleted.rules       mysql.rules      snmp.rules
backdoor.rules                 community-oracle.rules         dns.rules           netbios.rules    sql.rules
bad-traffic.rules              community-policy.rules         dos.rules           nntp.rules       telnet.rules
chat.rules                     community-sip.rules            experimental.rules  oracle.rules     tftp.rules
community-bot.rules            community-smtp.rules           exploit.rules       other-ids.rules  virus.rules
community-deleted.rules        community-sql-injection.rules  finger.rules        p2p.rules        web-attacks.rules
community-dos.rules            community-virus.rules          ftp.rules           policy.rules     web-cgi.rules
community-exploit.rules        community-web-attacks.rules    HEARTBLEED.rules    pop2.rules       web-client.rules
community-ftp.rules            community-web-cgi.rules        icmp-info.rules     pop3.rules       web-coldfusion.rules
community-game.rules           community-web-client.rules     icmp.rules          porn.rules       web-frontpage.rules
community-icmp.rules           community-web-dos.rules        imap.rules          rpc.rules        web-iis.rules
community-imap.rules           community-web-iis.rules        info.rules          rservices.rules  web-misc.rules
community-inappropriate.rules  community-web-misc.rules       local.rules         scan.rules       web-php.rules
community-mail-client.rules    community-web-php.rules        misc.rules          shellcode.rules  x11.rules
community-misc.rules           ddos.rules                     multimedia.rules    smtp.rules
ubuntu@ubuntu1804:/etc/snort/rules$
```

Figure 2.3.3 Add ruleset to the config file



```
include $RULE_PATH/community-web-client.rules
include $RULE_PATH/community-web-dos.rules
include $RULE_PATH/community-web-iis.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-web-php.rules
include $RULE_PATH/heartbleed.rules

###################################################
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc_rules
###################################################

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

$ cd /etc/snort

$ sudo nano snort.config

In section 7 ensure to include the following: $ Include $RULE_PATH/HEARTBLEED.rules

Ctrl + x to save the configuration.

Proof of concept - Run this command:

$ snort -d -l /var/log/snort/ -h 10.0.2.8 -A console -c /etc/snort/snort.conf

Command	Description
-d	Requests that data is shown by snort
-l	Decides on which directory logs are kept
-h	Network specification for monitoring
-A	This allows printed alerts in the console
-c	Configuration specification

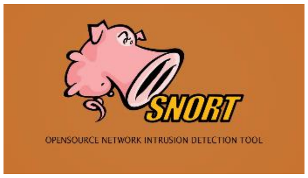
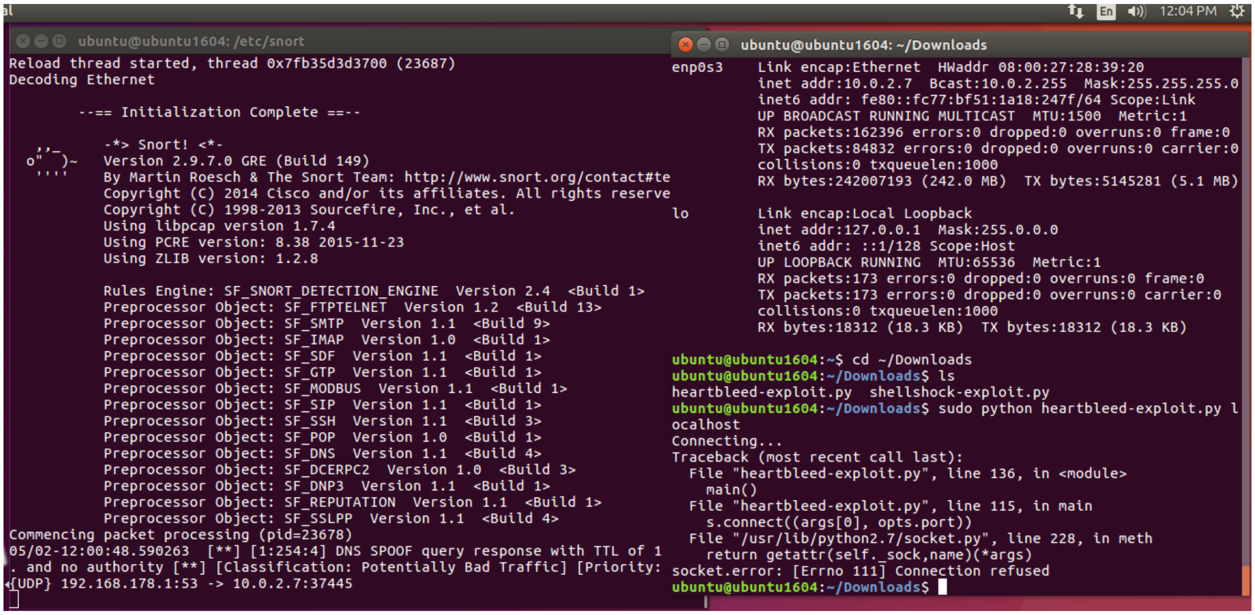


Figure.2.1.4 Snort [6]

Figure.2.3.5 – Proof of concept



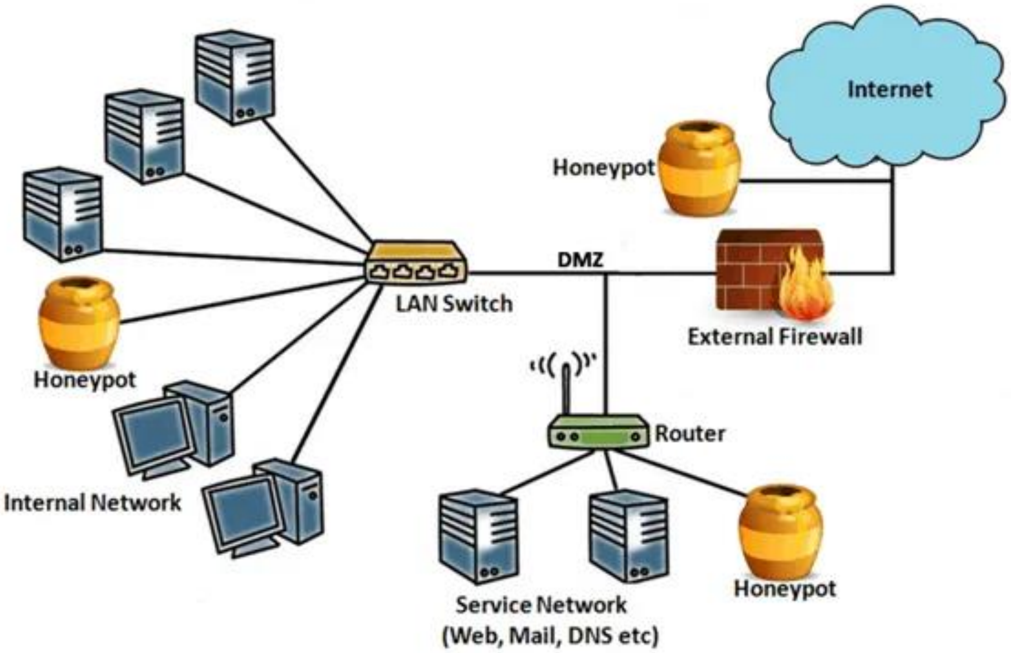
```
Reload thread started, thread 0x7fb35d3d3700 (23687)
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#te
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserve
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.7.4
           Using PCRE version: 8.38 2015-11-23
           Using ZLIB version: 1.2.8

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.4  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
Commencing packet processing (pid=23678)
05/02-12:00:48.590263  [**] [1:254:4] DNS SPOOF query response with TTL of 1
. and no authority [**] [Classification: Potentially Bad Traffic] [Priority:
{UDP} 192.168.178.1:53 -> 10.0.2.7:37445
```

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:28:39:20
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::fc77:bf51:1a18:247f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:162396 errors:0 dropped:0 overruns:0 frame:0
          TX packets:84832 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:242007193 (242.0 MB)  TX bytes:5145281 (5.1 MB)

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:65536  Metric:1
          RX packets:173 errors:0 dropped:0 overruns:0 frame:0
          TX packets:173 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:18312 (18.3 KB)  TX bytes:18312 (18.3 KB)

ubuntu@ubuntu1604:~$ cd ~/Downloads
ubuntu@ubuntu1604:~/Downloads$ ls
heartbleed-exploit.py  shellshock-exploit.py
ubuntu@ubuntu1604:~/Downloads$ sudo python heartbleed-exploit.py l
ocalhost
Connecting...
Traceback (most recent call last):
  File "heartbleed-exploit.py", line 136, in <module>
    main()
  File "heartbleed-exploit.py", line 115, in main
    s.connect((args[0], opts.port))
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 111] Connection refused
ubuntu@ubuntu1604:~/Downloads$
```

3 Task 4 – Additional Remediation Strategy

As an additional remediation strategy, I would recommend setting up a HoneyPot which will offer up a ‘vulnerable server’ to would be threat actors, lure them in and trap them, while the system remains perfectly safe, and the threat actors think they have hit the jackpot. The one I have implemented is a Perl script courtesy of Packet Storm Security, see indexes for share link to the script [7].

Figure 2.4.1 How a honeypot works [9]:

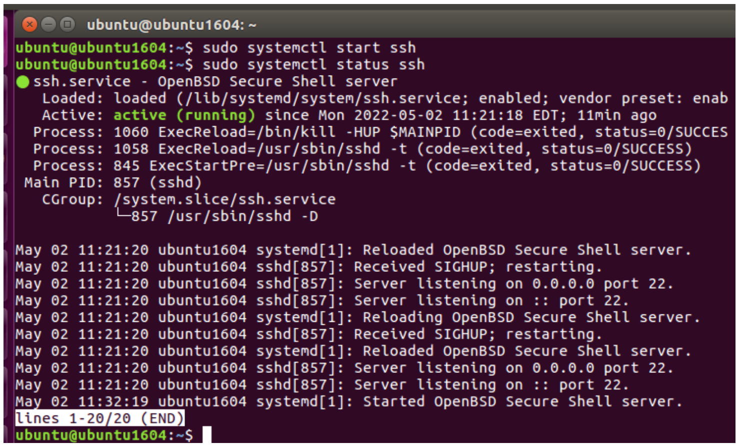


Initial Commands:

$ sudo systemctl start ssh

$ sudo systemctl status ssh

Figure 2.4.2 – Start and check ssh



```
ubuntu@ubuntu1604: ~
ubuntu@ubuntu1604:~$ sudo systemctl start ssh
ubuntu@ubuntu1604:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enab
   Active: active (running) since Mon 2022-05-02 11:21:18 EDT; 11min ago
  Process: 1060 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCES
  Process: 1058 ExecReload=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Process: 845 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 857 (sshd)
   CGroup: /system.slice/ssh.service
           └─857 /usr/sbin/sshd -D

May 02 11:21:20 ubuntu1604 systemd[1]: Reloaded OpenBSD Secure Shell server.
May 02 11:21:20 ubuntu1604 sshd[857]: Received SIGHUP; restarting.
May 02 11:21:20 ubuntu1604 sshd[857]: Server listening on 0.0.0.0 port 22.
May 02 11:21:20 ubuntu1604 sshd[857]: Server listening on :: port 22.
May 02 11:21:20 ubuntu1604 systemd[1]: Reloading OpenBSD Secure Shell server.
May 02 11:21:20 ubuntu1604 sshd[857]: Received SIGHUP; restarting.
May 02 11:21:20 ubuntu1604 systemd[1]: Reloaded OpenBSD Secure Shell server.
May 02 11:21:20 ubuntu1604 sshd[857]: Server listening on 0.0.0.0 port 22.
May 02 11:21:20 ubuntu1604 sshd[857]: Server listening on :: port 22.
May 02 11:32:19 ubuntu1604 systemd[1]: Started OpenBSD Secure Shell server.
lines 1-20/20 (END)
ubuntu@ubuntu1604:~$
```

Commands:

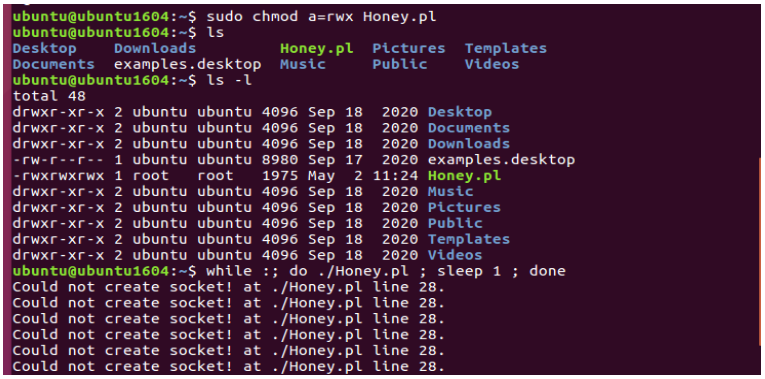
$ sudo nano Honey.pl

$ sudo chmod a=rwx Honey.pl

Run the script in a loop to monitor connections:

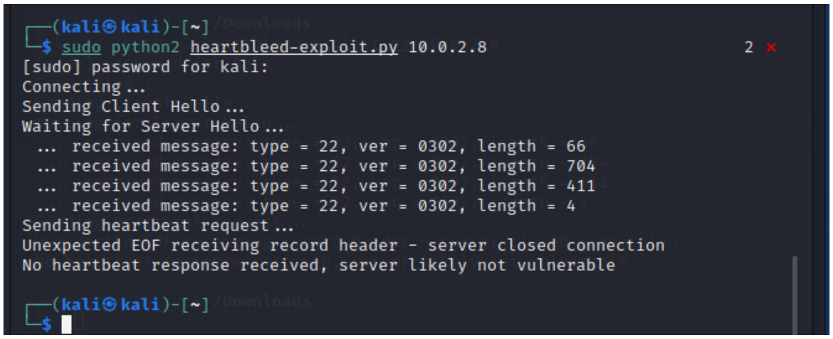
$ while :; do ./Honey.pl ; sleep 1 ; done

Figure 2.4.3 Screenshot of technique:



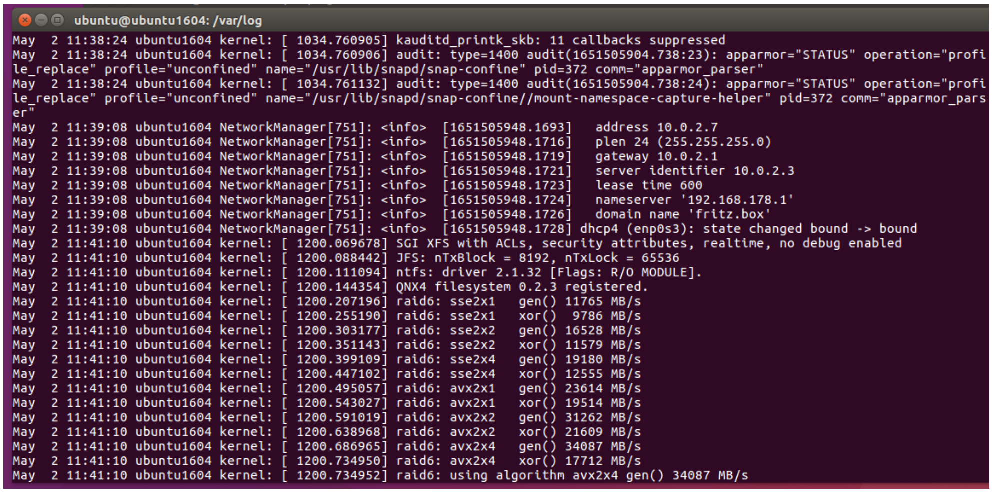
```
ubuntu@ubuntu1604:~$ sudo chmod a=rwx Honey.pl
ubuntu@ubuntu1604:~$ ls
Desktop    Downloads         Honey.pl  Pictures  Templates
Documents  examples.desktop  Music     Public    Videos
ubuntu@ubuntu1604:~$ ls -l
total 48
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Desktop
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Documents
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Downloads
-rw-r--r-- 1 ubuntu ubuntu 8980 Sep 17  2020 examples.desktop
-rwxrwxrwx 1 root   root   1975 May  2 11:24 Honey.pl
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Music
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Pictures
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Public
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Templates
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Videos
ubuntu@ubuntu1604:~$ while :; do ./Honey.pl ; sleep 1 ; done
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
```

Figure 2.4.4 – Screenshot of attempt to exploit heartbleed:



```
┌──(kali㉿kali)-[~]
└─$ sudo python2 heartbleed-exploit.py 10.0.2.8                2 ✕
[sudo] password for kali:
Connecting...
Sending Client Hello...
Waiting for Server Hello...
 ... received message: type = 22, ver = 0302, length = 66
 ... received message: type = 22, ver = 0302, length = 704
 ... received message: type = 22, ver = 0302, length = 411
 ... received message: type = 22, ver = 0302, length = 4
Sending heartbeat request...
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable

┌──(kali㉿kali)-[~]
└─$ 
```

Figure.2.4.5 – kern.log file screenshot



```
May  2 11:38:24 ubuntu1604 kernel: [ 1034.760905] kauditd_printk_skb: 11 callbacks suppressed
May  2 11:38:24 ubuntu1604 kernel: [ 1034.760906] audit: type=1400 audit(1651505904.738:23): apparmor="STATUS" operation="profi
le_replace" profile="unconfined" name="/usr/lib/snapd/snap-confine" pid=372 comm="apparmor_parser"
May  2 11:38:24 ubuntu1604 kernel: [ 1034.761132] audit: type=1400 audit(1651505904.738:24): apparmor="STATUS" operation="profi
le_replace" profile="unconfined" name="/usr/lib/snapd/snap-confine//mount-namespace-capture-helper" pid=372 comm="apparmor_pars
er"
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1693]   address 10.0.2.7
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1716]   plen 24 (255.255.255.0)
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1719]   gateway 10.0.2.1
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1721]   server identifier 10.0.2.3
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1723]   lease time 600
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1724]   nameserver '192.168.178.1'
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1726]   domain name 'fritz.box'
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1728] dhcp4 (enp0s3): state changed bound -> bound
May  2 11:41:10 ubuntu1604 kernel: [ 1200.069678] SGI XFS with ACLs, security attributes, realtime, no debug enabled
May  2 11:41:10 ubuntu1604 kernel: [ 1200.088442] JFS: nTxBlock = 8192, nTxLock = 65536
May  2 11:41:10 ubuntu1604 kernel: [ 1200.111094] ntfs: driver 2.1.32 [Flags: R/O MODULE].
May  2 11:41:10 ubuntu1604 kernel: [ 1200.144354] QNX4 filesystem 0.2.3 registered.
May  2 11:41:10 ubuntu1604 kernel: [ 1200.207196] raid6: sse2x1   gen() 11765 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.255190] raid6: sse2x1   xor()  9786 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.303177] raid6: sse2x2   gen() 16528 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.351143] raid6: sse2x2   xor() 11579 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.399109] raid6: sse2x4   gen() 19180 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.447102] raid6: sse2x4   xor() 12555 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.495057] raid6: avx2x1   gen() 23614 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.543027] raid6: avx2x1   xor() 19514 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.591019] raid6: avx2x2   gen() 31262 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.638968] raid6: avx2x2   xor() 21609 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.686965] raid6: avx2x4   gen() 34087 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.734950] raid6: avx2x4   xor() 17712 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.734952] raid6: using algorithm avx2x4 gen() 34087 MB/s
```

4 Shellshock Vulnerability

4.1 Task 1 – Vulnerability Description

Shellshock is a bug that was discovered in 2014 by Stephane Chazelas [12] it works as a remote command execution vulnerability and utilises bash to conduct attacks, creating backdoors into vulnerable systems. It abuses the import of a function, using trail commands to exploit the vulnerability. It is the first, and one of many known vulnerabilities that belong to the family of Shellshock bugs. [9].

Shellshock was initially known as bashdoor and is now a family of bugs known as the Shellshock family, within hours of the announcement of its discovery adversaries had already exploited the vulnerability using botnets created from compromised computers and conducted DDoS attacks and probes of victims' networks and computers.

Technical Details:

Shellshock Vulnerability	
CVE's	(Shellshock) CVE-2014-6271 (Rest of the Shellshock family) CVE-2014-7169, CVE-2014-7186, CVE-2014-7187, CVE-2014-6277
NIST Rating:	10 out of 10
Date Discovered:	12/09/2014
Date Disclosed:	24/09/2014
Bash Version:	1.03
Founder:	Stephane Chazelas
Vulnerable Since:	1989

Attack Vectors:	RCE(Apache & mod_cgi, CGI scripts, Perl, Python)
	RCE(DHCP Client, using Hostile DHCP Server)
	Open SSH RCE/Privilege Escalation)

Possible Mitigations:	Iptables firewall rules, Suricata, force bash to use privilege mode
Targets Discoverable by:	GoogleHacking, Port Scanning, Nmap Shellshock Script, Online Scanners, Metasploit Module

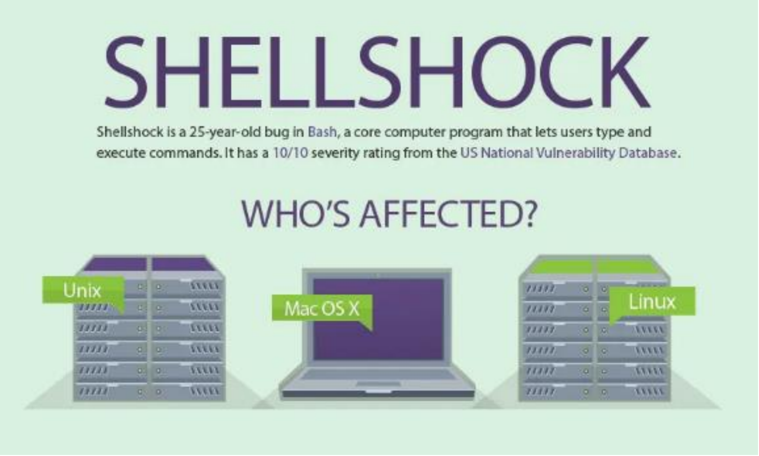

SHELLSHOCK
Shellshock is a 25-year-old bug in Bash, a core computer program that lets users type and execute commands. It has a 10/10 severity rating from the US National Vulnerability Database.
WHO'S AFFECTED?
Unix
Mac OS X
Linux

Figure.3.1.1 Shellshock Bug [8]

4.2 Task 2 – iptables Firewall Rules

We can use IP Tables to mitigate against the Shellshock vulnerability which would drop packets that potentially contain an attack, it is the go-to method when first discovering vulnerabilities to set up remediation on a temporary basis. It is important to note that it is a minimal impact mitigation and threat actors could easily work around these but reducing the amount characters per packet to side-step the signature check. String matching is a last resort and to be used only in emergency situations [12]

Commands:

$ iptables -A INPUT -m string –algo bm –hex-string `|28 29 20 7B ` -j DROP

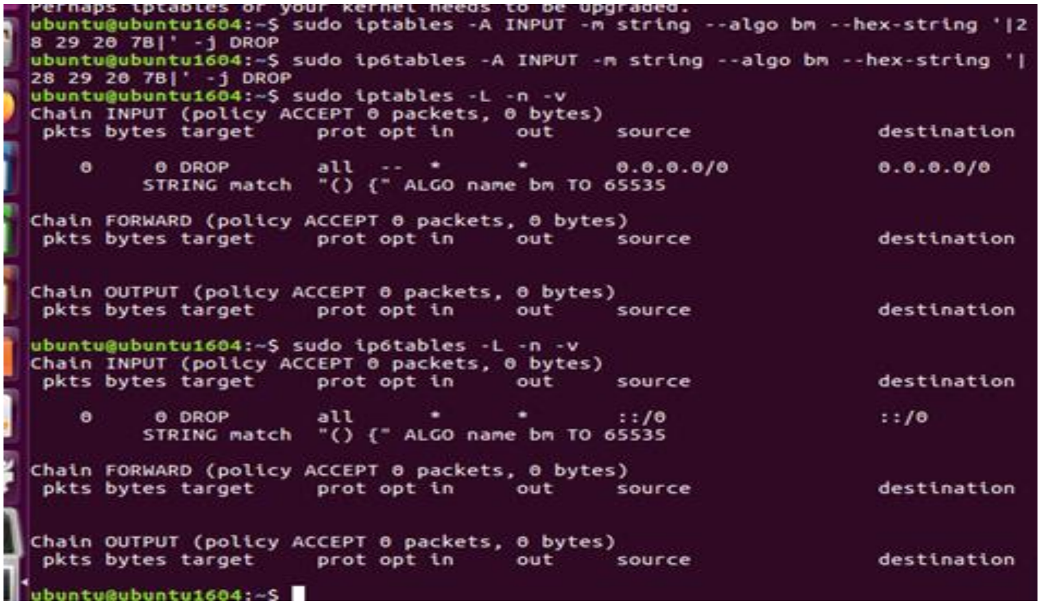
$ ip6tables -A INPUT -m string –algo bm –hex-string `|28 29 20 7B ` -j DROP

$ iptables -A INPUT -m string –algo bm –hex-string `|28 29 20 7B ` -j LOG

$ ip6tables -A INPUT -m string –algo bm –hex-string `|28 29 20 7B ` -j LOG

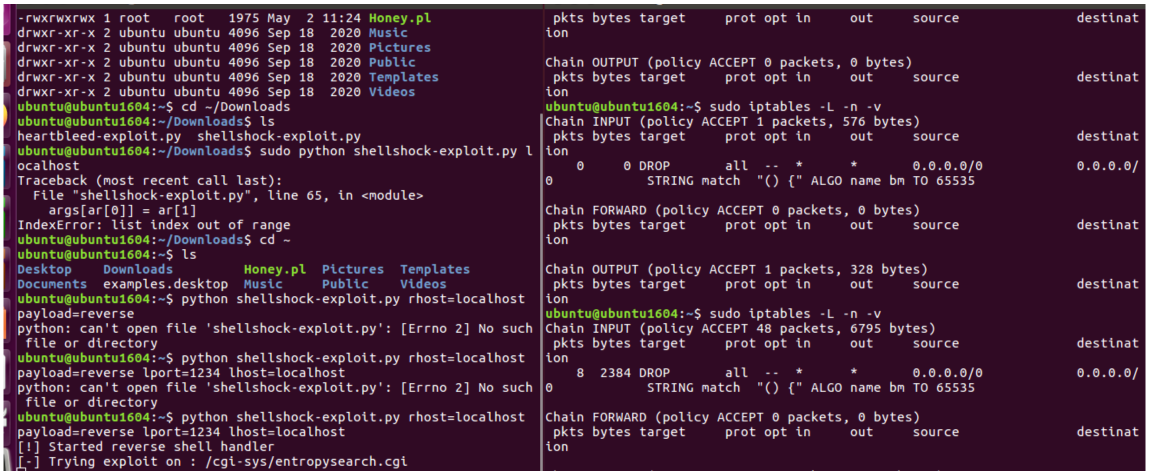
NB: These commands are looking specifically for Shellshock exploits and is setting the string, if it matches, the request is dropped and logged. It is for both ipv6 and ipv4.

-A is appending it to input table, and -j is used to select DROP and LOG options. -m is to select string matching, -hex-string determines the string that the command is looking for. –algo is defining the algorithm to look for, there are two, one is bm (Boyer-Moore) and the other is kmp (Knuth-Pratt-Morris).



```
Perhaps iptables or your kernel needs to be upgraded.
ubuntu@ubuntu1604:~$ sudo iptables -A INPUT -m string --algo bm --hex-string '|2
8 29 20 7B|' -j DROP
ubuntu@ubuntu1604:~$ sudo ip6tables -A INPUT -m string --algo bm --hex-string '|
28 29 20 7B|' -j DROP
ubuntu@ubuntu1604:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
         STRING match  "() {" ALGO name bm TO 65535

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

ubuntu@ubuntu1604:~$ sudo ip6tables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 DROP       all      *      *       ::/0                 ::/0
         STRING match  "() {" ALGO name bm TO 65535

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

ubuntu@ubuntu1604:~$
```

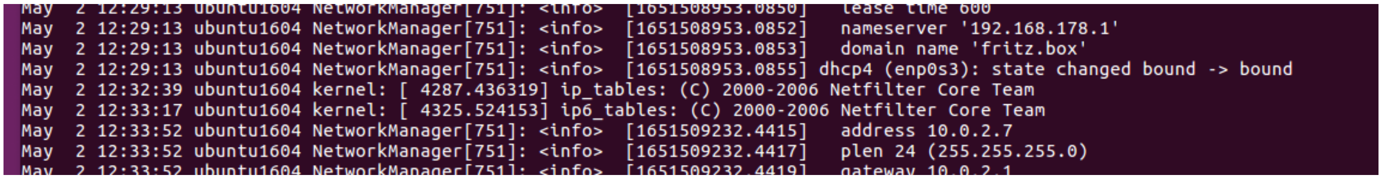
Figure.3.2.1 iptables Firewall Rules



```
-rwxrwxrwx 1 root   root   1975 May  2 11:24 Honey.pl
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Music
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Pictures
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Public
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Templates
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Videos
ubuntu@ubuntu1604:~$ cd ~/Downloads
ubuntu@ubuntu1604:~/Downloads$ ls
heartbleed-exploit.py  shellshock-exploit.py
ubuntu@ubuntu1604:~/Downloads$ sudo python shellshock-exploit.py l
ocalhost
Traceback (most recent call last):
  File "shellshock-exploit.py", line 65, in <module>
    args[ar[0]] = ar[1]
IndexError: list index out of range
ubuntu@ubuntu1604:~/Downloads$ cd ~
ubuntu@ubuntu1604:~$ ls
Desktop    Downloads         Honey.pl  Pictures  Templates
Documents  examples.desktop  Music     Public    Videos
ubuntu@ubuntu1604:~$ python shellshock-exploit.py rhost=localhost
payload=reverse
python: can't open file 'shellshock-exploit.py': [Errno 2] No such
 file or directory
ubuntu@ubuntu1604:~$ python shellshock-exploit.py rhost=localhost
payload=reverse lport=1234 lhost=localhost
python: can't open file 'shellshock-exploit.py': [Errno 2] No such
 file or directory
ubuntu@ubuntu1604:~$ python shellshock-exploit.py rhost=localhost
payload=reverse lport=1234 lhost=localhost
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-sys/entropysearch.cgi
```

```
 pkts bytes target     prot opt in     out     source               destinat
ion

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destinat
ion
ubuntu@ubuntu1604:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 1 packets, 576 bytes)
 pkts bytes target     prot opt in     out     source               destinat
ion
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/
0            STRING match  "() {" ALGO name bm TO 65535

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destinat
ion

Chain OUTPUT (policy ACCEPT 1 packets, 328 bytes)
 pkts bytes target     prot opt in     out     source               destinat
ion
ubuntu@ubuntu1604:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 48 packets, 6795 bytes)
 pkts bytes target     prot opt in     out     source               destinat
ion
    8  2384 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/
0            STRING match  "() {" ALGO name bm TO 65535

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destinat
ion
```

Figure.3.2.2 – Proof of concept



```
May  2 12:29:13 ubuntu1604 NetworkManager[751]: <info>  [1651508953.0850]   lease time 600
May  2 12:29:13 ubuntu1604 NetworkManager[751]: <info>  [1651508953.0852]   nameserver '192.168.178.1'
May  2 12:29:13 ubuntu1604 NetworkManager[751]: <info>  [1651508953.0853]   domain name 'fritz.box'
May  2 12:29:13 ubuntu1604 NetworkManager[751]: <info>  [1651508953.0855] dhcp4 (enp0s3): state changed bound -> bound
May  2 12:32:39 ubuntu1604 kernel: [ 4287.436319] ip_tables: (C) 2000-2006 Netfilter Core Team
May  2 12:33:17 ubuntu1604 kernel: [ 4325.524153] ip6_tables: (C) 2000-2006 Netfilter Core Team
May  2 12:33:52 ubuntu1604 NetworkManager[751]: <info>  [1651509232.4415]   address 10.0.2.7
May  2 12:33:52 ubuntu1604 NetworkManager[751]: <info>  [1651509232.4417]   plen 24 (255.255.255.0)
May  2 12:33:52 ubuntu1604 NetworkManager[751]: <info>  [1651509232.4419]   gateway 10.0.2.1
```

Figure.3.2.3 – Kern.log iptables

4.3 Task 3 – Alarm Triggering

The alarm technique I have employed for detection of the shellshock bug is Suricata, it is open source meaning it keeps costs down while providing a service to detect the shellshock bug. Suricata is quick to implement and offers capabilities of outputting in-depth information of attacks in a JSON file. [11][13] This used in conjunction with Logwatch will help keep track of any successful shellshock hacks into your system.

Prerequisite Commands:

$ sudo apt install suricata

$ sudo -i

$ apt install yum

$ apt install curl

Figure.3.3.1 - Execution Commands:

```
$ yum -y install epel-release wget jq
$ curl -O https://copr.fedorainfracloud.org/coprs/jasonish/suricata-
6.0/repo/epel-7/jasonish-suricata-6.0-epel-7.repo
$ yum -y install suricata
$ wget https://rules.emergingthreats.net/open/suricata-
6.0.3/emerging.rules.tar.gz
$ tar zxvf emerging.rules.tar.gz
$ rm /etc/suricata/rules/* -f
$ mv rules/*.rules /etc/suricata/rules/
$ rm -f /etc/suricata/suricata.yaml
$ wget -O /etc/suricata/suricata.yaml
http://www.branchnetconsulting.com/wazuh/suricata.yaml
$ systemctl daemon-reload
$ systemctl enable suricata
$ systemctl start suricata
```

Check the exploit:

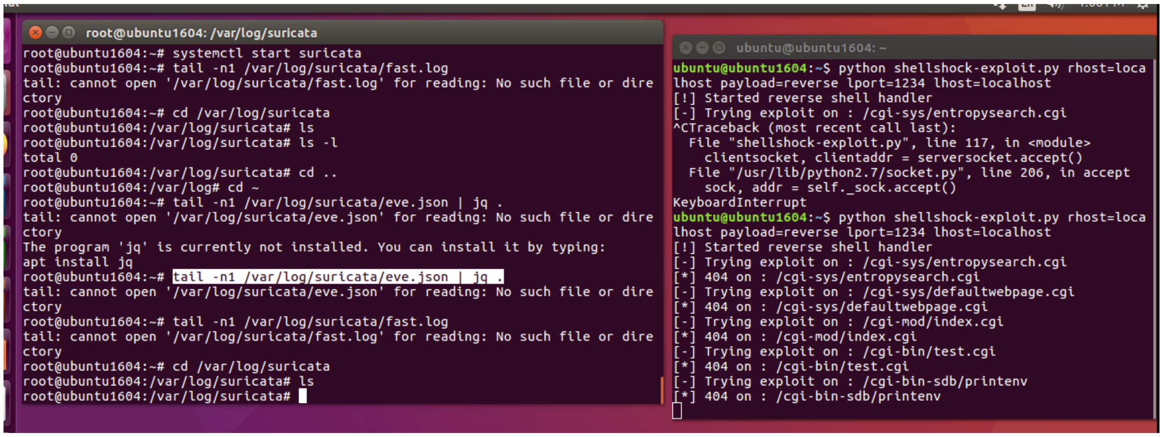
$ python shellshock-exploit.py rhost=localhost payload=reverse lport=1234 lhost=localhost

$ tail -n1 /var/log/suricata/fast.log

A more detailed view would be:

$ tail -n1 /var/log/suricata/eve.json | jq .

Figure.3.3.2 – Exploit attempt



```
root@ubuntu1604:~# systemctl start suricata
root@ubuntu1604:~# tail -n1 /var/log/suricata/fast.log
tail: cannot open '/var/log/suricata/fast.log' for reading: No such file or dire
ctory
root@ubuntu1604:~# cd /var/log/suricata
root@ubuntu1604:/var/log/suricata# ls
root@ubuntu1604:/var/log/suricata# ls -l
total 0
root@ubuntu1604:/var/log/suricata# cd ..
root@ubuntu1604:/var/log# cd ~
root@ubuntu1604:~# tail -n1 /var/log/suricata/eve.json | jq .
tail: cannot open '/var/log/suricata/eve.json' for reading: No such file or dire
ctory
The program 'jq' is currently not installed. You can install it by typing:
apt install jq
root@ubuntu1604:~# tail -n1 /var/log/suricata/eve.json | jq .
tail: cannot open '/var/log/suricata/eve.json' for reading: No such file or dire
ctory
root@ubuntu1604:~# tail -n1 /var/log/suricata/fast.log
tail: cannot open '/var/log/suricata/fast.log' for reading: No such file or dire
ctory
root@ubuntu1604:~# cd /var/log/suricata
root@ubuntu1604:/var/log/suricata# ls
root@ubuntu1604:/var/log/suricata#
```

```
ubuntu@ubuntu1604:~$ python shellshock-exploit.py rhost=loca
lhost payload=reverse lport=1234 lhost=localhost
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-sys/entropysearch.cgi
^CTraceback (most recent call last):
  File "shellshock-exploit.py", line 117, in <module>
    clientsocket, clientaddr = serversocket.accept()
  File "/usr/lib/python2.7/socket.py", line 206, in accept
    sock, addr = self._sock.accept()
KeyboardInterrupt
ubuntu@ubuntu1604:~$ python shellshock-exploit.py rhost=loca
lhost payload=reverse lport=1234 lhost=localhost
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-sys/entropysearch.cgi
[*] 404 on : /cgi-sys/entropysearch.cgi
[-] Trying exploit on : /cgi-sys/defaultwebpage.cgi
[*] 404 on : /cgi-sys/defaultwebpage.cgi
[-] Trying exploit on : /cgi-mod/index.cgi
[*] 404 on : /cgi-mod/index.cgi
[-] Trying exploit on : /cgi-bin/test.cgi
[*] 404 on : /cgi-bin/test.cgi
[-] Trying exploit on : /cgi-bin-sdb/printenv
[*] 404 on : /cgi-bin-sdb/printenv
```

NB: No logs were made meaning attempt did not succeed.

Figure.3.3.3 Suricata

4.4 Task 4 – Additional Remediation Strategy

As an additional remediation strategy, I have considered forcing the vulnerable bash to use privilege mode, it may come with its disadvantages but also proffers some advantages for securing against the shellshock vulnerability. Although temporary, it may prevent attacks while the system is online. This may come with certain issues listed below in the table derived from the bash man page [12]:

Figure.4.4.1 – privilege mode

-p	Turn on privileged mode. In this mode, the $ENV and $BASH_ENV files are not processed, shell functions are not inherited from the environment, and the SHELLOPTS, BASHOPTS, CDPATH, and GLOBIGNORE variables, if they appear in the environment, are ignored. If the shell is started with the effective user (group) id not equal to the real user (group) id, and the -p option is not supplied, these actions are taken, and the effective user id is set to the real user id. If the -p option is supplied at startup, the effective user id is not reset. Turning this option off causes the effective user and group ids to be set to the real user and group ids.

Commands:

Ensure debug and systemtap is updated/installed.

$ sudo apt install systemtap

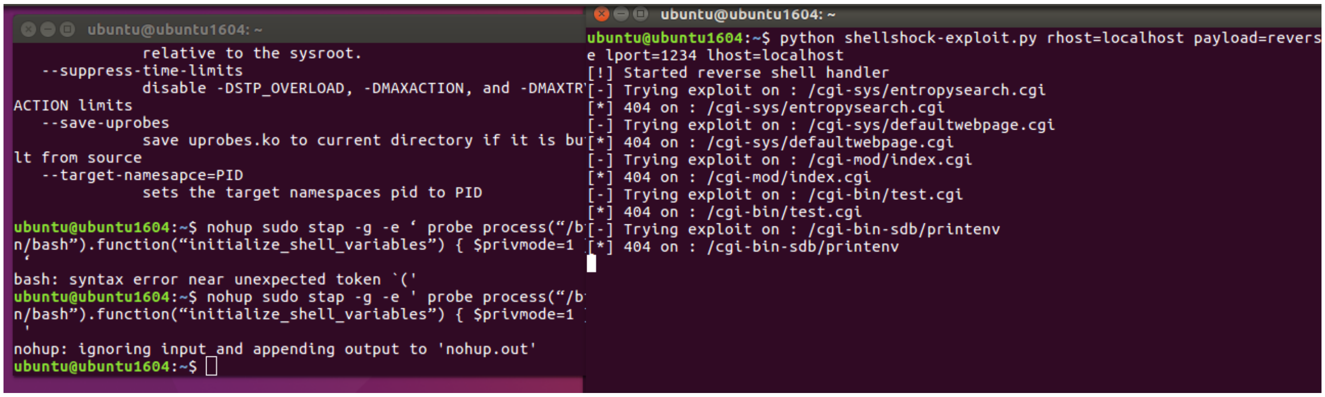
To make sure that privilege mode in bash is always operating:

$ nohup sudo stap -g -e ‘ probe process(“/bin/bash”).function(“initialize_shell_variables”) { $privmode=1 } ‘

Test for the vulnerability:

$ python shellshock-exploit.py rhost=localhost payload=reverse lport=1234 lhost=localhostFigure.3.4.1 – Initial Commands

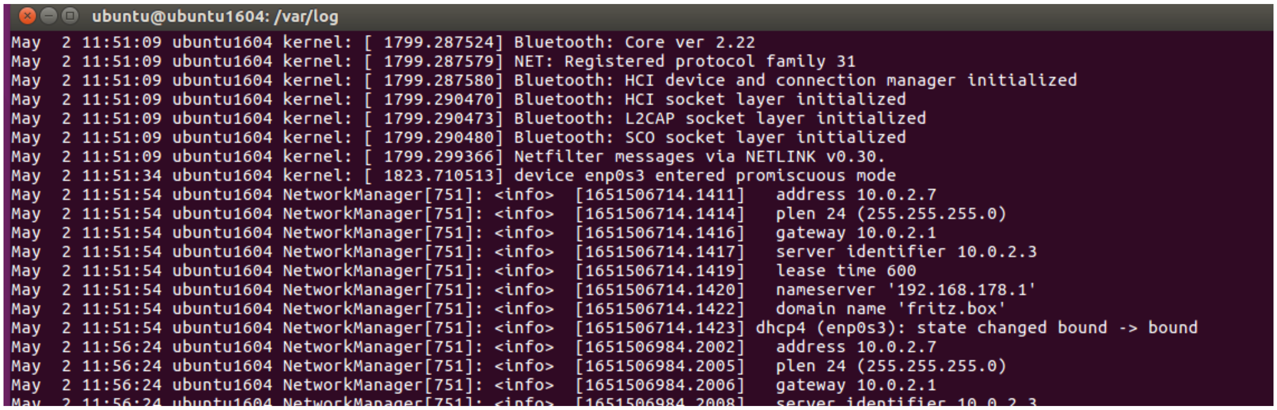
Figure.4.4.2 – Exploit



```
                relative to the sysroot.
   --suppress-time-limits
                disable -DSTP_OVERLOAD, -DMAXACTION, and -DMAXTR
ACTION limits
   --save-uprobes
                save uprobes.ko to current directory if it is bu
lt from source
   --target-namesapce=PID
                sets the target namespaces pid to PID

ubuntu@ubuntu1604:~$ nohup sudo stap -g -e ‘ probe process(“/b
n/bash”).function(“initialize_shell_variables”) { $privmode=1
 ‘
bash: syntax error near unexpected token `('
ubuntu@ubuntu1604:~$ nohup sudo stap -g -e ' probe process(“/b
n/bash”).function(“initialize_shell_variables”) { $privmode=1
 '
nohup: ignoring input and appending output to 'nohup.out'
ubuntu@ubuntu1604:~$ 
```

```
ubuntu@ubuntu1604:~$ python shellshock-exploit.py rhost=localhost payload=revers
e lport=1234 lhost=localhost
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-sys/entropysearch.cgi
[*] 404 on : /cgi-sys/entropysearch.cgi
[-] Trying exploit on : /cgi-sys/defaultwebpage.cgi
[*] 404 on : /cgi-sys/defaultwebpage.cgi
[-] Trying exploit on : /cgi-mod/index.cgi
[*] 404 on : /cgi-mod/index.cgi
[-] Trying exploit on : /cgi-bin/test.cgi
[*] 404 on : /cgi-bin/test.cgi
[-] Trying exploit on : /cgi-bin-sdb/printenv
[*] 404 on : /cgi-bin-sdb/printenv
```

Figure.4.4.3 – kern.log file screenshot



ubuntu@ubuntu1604: /var/log

```
May  2 11:51:09 ubuntu1604 kernel: [ 1799.287524] Bluetooth: Core ver 2.22
May  2 11:51:09 ubuntu1604 kernel: [ 1799.287579] NET: Registered protocol family 31
May  2 11:51:09 ubuntu1604 kernel: [ 1799.287580] Bluetooth: HCI device and connection manager initialized
May  2 11:51:09 ubuntu1604 kernel: [ 1799.290470] Bluetooth: HCI socket layer initialized
May  2 11:51:09 ubuntu1604 kernel: [ 1799.290473] Bluetooth: L2CAP socket layer initialized
May  2 11:51:09 ubuntu1604 kernel: [ 1799.290480] Bluetooth: SCO socket layer initialized
May  2 11:51:09 ubuntu1604 kernel: [ 1799.299366] Netfilter messages via NETLINK v0.30.
May  2 11:51:34 ubuntu1604 kernel: [ 1823.710513] device enp0s3 entered promiscuous mode
May  2 11:51:54 ubuntu1604 NetworkManager[751]: <info>  [1651506714.1411]   address 10.0.2.7
May  2 11:51:54 ubuntu1604 NetworkManager[751]: <info>  [1651506714.1414]   plen 24 (255.255.255.0)
May  2 11:51:54 ubuntu1604 NetworkManager[751]: <info>  [1651506714.1416]   gateway 10.0.2.1
May  2 11:51:54 ubuntu1604 NetworkManager[751]: <info>  [1651506714.1417]   server identifier 10.0.2.3
May  2 11:51:54 ubuntu1604 NetworkManager[751]: <info>  [1651506714.1419]   lease time 600
May  2 11:51:54 ubuntu1604 NetworkManager[751]: <info>  [1651506714.1420]   nameserver '192.168.178.1'
May  2 11:51:54 ubuntu1604 NetworkManager[751]: <info>  [1651506714.1422]   domain name 'fritz.box'
May  2 11:51:54 ubuntu1604 NetworkManager[751]: <info>  [1651506714.1423] dhcp4 (enp0s3): state changed bound -> bound
May  2 11:56:24 ubuntu1604 NetworkManager[751]: <info>  [1651506984.2002]   address 10.0.2.7
May  2 11:56:24 ubuntu1604 NetworkManager[751]: <info>  [1651506984.2005]   plen 24 (255.255.255.0)
May  2 11:56:24 ubuntu1604 NetworkManager[751]: <info>  [1651506984.2006]   gateway 10.0.2.1
May  2 11:56:24 ubuntu1604 NetworkManager[751]: <info>  [1651506984.2008]   server identifier 10.0.2.3
```

You can also run a honeypot here too:

Commands:

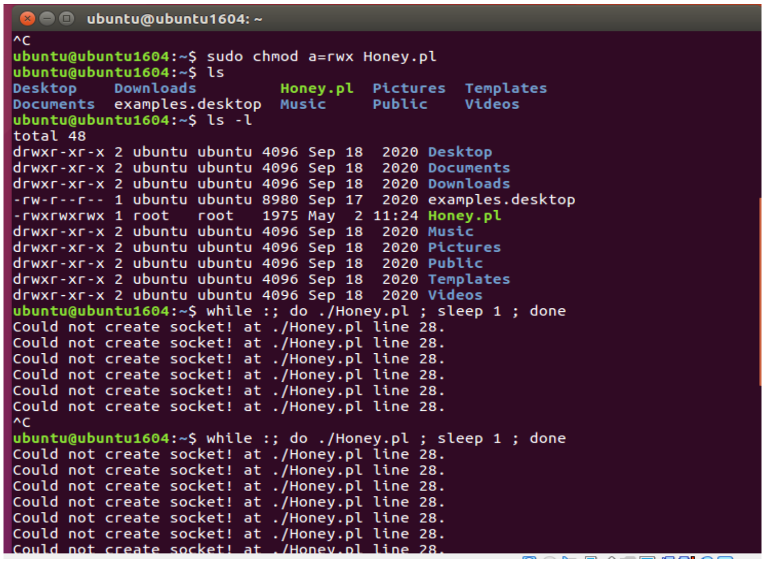
$ sudo nano Honey.pl

$ sudo chmod a=rwx Honey.pl

Run the script in a loop to monitor connections:

$ while :; do ./Honey.pl ; sleep 1 ; done

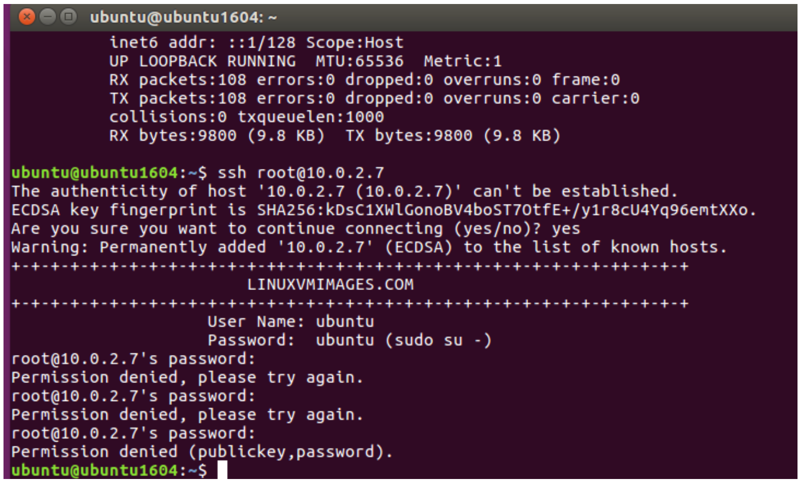
Figure 2.4.3 Screenshot of technique:



ubuntu@ubuntu1604: ~

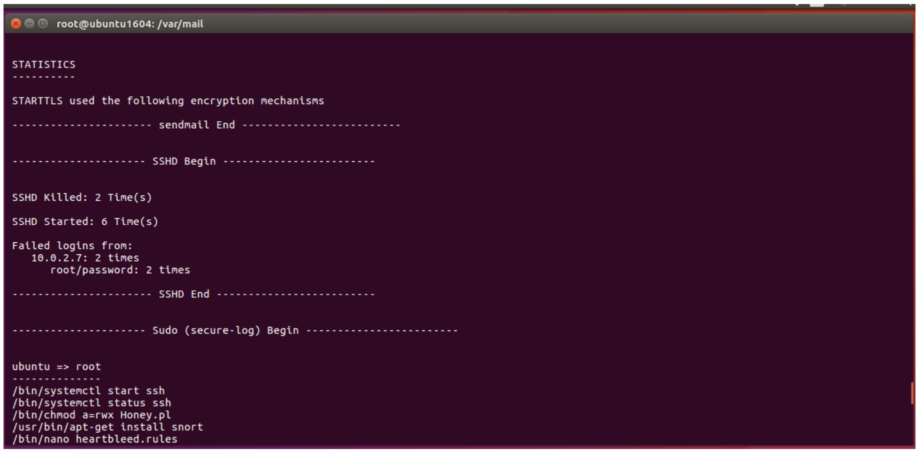
```
^C
ubuntu@ubuntu1604:~$ sudo chmod a=rwx Honey.pl
ubuntu@ubuntu1604:~$ ls
Desktop    Downloads         Honey.pl  Pictures  Templates
Documents  examples.desktop  Music     Public    Videos
ubuntu@ubuntu1604:~$ ls -l
total 48
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Desktop
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Documents
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Downloads
-rw-r--r-- 1 ubuntu ubuntu 8980 Sep 17  2020 examples.desktop
-rwxrwxrwx 1 root   root   1975 May  2 11:24 Honey.pl
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Music
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Pictures
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Public
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Templates
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 18  2020 Videos
ubuntu@ubuntu1604:~$ while :; do ./Honey.pl ; sleep 1 ; done
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
^C
ubuntu@ubuntu1604:~$ while :; do ./Honey.pl ; sleep 1 ; done
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
Could not create socket! at ./Honey.pl line 28.
```

Figure 2.4.4 – Attempt to ssh into system



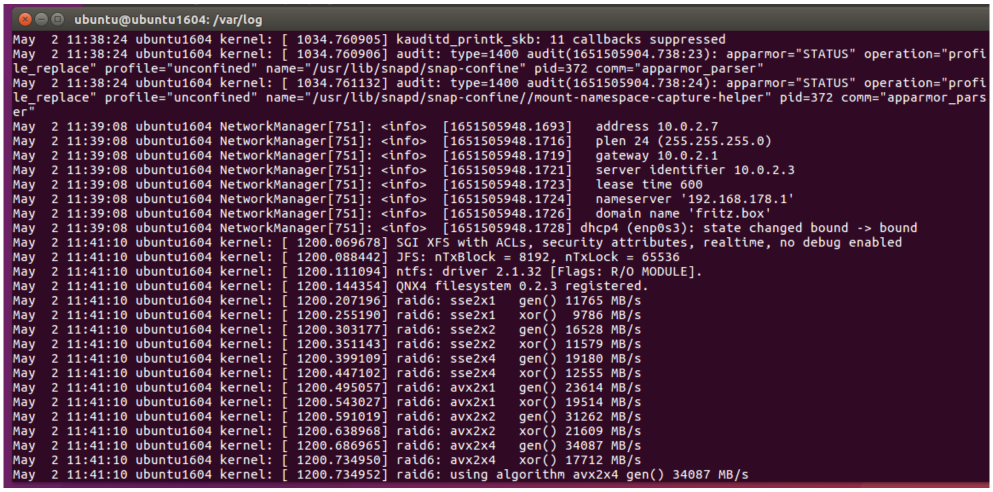
```
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:108 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9800 (9.8 KB)  TX bytes:9800 (9.8 KB)

ubuntu@ubuntu1604:~$ ssh root@10.0.2.7
The authenticity of host '10.0.2.7 (10.0.2.7)' can't be established.
ECDSA key fingerprint is SHA256:kDsC1XWlGonoBV4boST7OtfE+/y1r8cU4Yq96emtXXo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.7' (ECDSA) to the list of known hosts.
+-+-+-+-+-+-+-+-+-+-+-+-++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-++-+-+-+
                        LINUXVMIMAGES.COM
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                    User Name: ubuntu
                    Password:  ubuntu (sudo su -)
root@10.0.2.7's password:
Permission denied, please try again.
root@10.0.2.7's password:
Permission denied, please try again.
root@10.0.2.7's password:
Permission denied (publickey,password).
ubuntu@ubuntu1604:~$ 
```

Figure.2.4.5 – logwatch mail confirming denied ssh attempt:



```
STATISTICS
----------

STARTTLS used the following encryption mechanisms

---------------------- sendmail End -------------------------


--------------------- SSHD Begin ------------------------


SSHD Killed: 2 Time(s)

SSHD Started: 6 Time(s)

Failed logins from:
   10.0.2.7: 2 times
      root/password: 2 times

---------------------- SSHD End -------------------------


--------------------- Sudo (secure-log) Begin ------------------------


ubuntu => root
--------------
/bin/systemctl start ssh
/bin/systemctl status ssh
/bin/chmod a=rwx Honey.pl
/usr/bin/apt-get install snort
/bin/nano heartbleed.rules
```

Figure.2.4.6 – kern.log file screenshot



```
May  2 11:38:24 ubuntu1604 kernel: [ 1034.760905] kauditd_printk_skb: 11 callbacks suppressed
May  2 11:38:24 ubuntu1604 kernel: [ 1034.760906] audit: type=1400 audit(1651505904.738:23): apparmor="STATUS" operation="profi
le_replace" profile="unconfined" name="/usr/lib/snapd/snap-confine" pid=372 comm="apparmor_parser"
May  2 11:38:24 ubuntu1604 kernel: [ 1034.761132] audit: type=1400 audit(1651505904.738:24): apparmor="STATUS" operation="profi
le_replace" profile="unconfined" name="/usr/lib/snapd/snap-confine//mount-namespace-capture-helper" pid=372 comm="apparmor_pars
er"
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1693]   address 10.0.2.7
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1716]   plen 24 (255.255.255.0)
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1719]   gateway 10.0.2.1
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1721]   server identifier 10.0.2.3
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1723]   lease time 600
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1724]   nameserver '192.168.178.1'
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1726]   domain name 'fritz.box'
May  2 11:39:08 ubuntu1604 NetworkManager[751]: <info>  [1651505948.1728] dhcp4 (enp0s3): state changed bound -> bound
May  2 11:41:10 ubuntu1604 kernel: [ 1200.069678] SGI XFS with ACLs, security attributes, realtime, no debug enabled
May  2 11:41:10 ubuntu1604 kernel: [ 1200.088442] JFS: nTxBlock = 8192, nTxLock = 65536
May  2 11:41:10 ubuntu1604 kernel: [ 1200.111094] ntfs: driver 2.1.32 [Flags: R/O MODULE].
May  2 11:41:10 ubuntu1604 kernel: [ 1200.144354] QNX4 filesystem 0.2.3 registered.
May  2 11:41:10 ubuntu1604 kernel: [ 1200.207196] raid6: sse2x1   gen() 11765 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.255190] raid6: sse2x1   xor()  9786 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.303177] raid6: sse2x2   gen() 16528 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.351143] raid6: sse2x2   xor() 11579 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.399109] raid6: sse2x4   gen() 19180 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.447102] raid6: sse2x4   xor() 12555 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.495057] raid6: avx2x1   gen() 23614 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.543027] raid6: avx2x1   xor() 19514 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.591019] raid6: avx2x2   gen() 31262 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.638968] raid6: avx2x2   xor() 21609 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.686965] raid6: avx2x4   gen() 34087 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.734950] raid6: avx2x4   xor() 17712 MB/s
May  2 11:41:10 ubuntu1604 kernel: [ 1200.734952] raid6: using algorithm avx2x4 gen() 34087 MB/s
```

5 Summary

The heartbleed bug is flawed extension of OpenSSL Heartbeat Extension using TLS protocols and comes with a NIST severity rating of 7.5, it is imperative to implement mitigations as soon as possible. I started by researching the bug itself and looking at various open-source proven methods of mitigation against the heartbleed bug. The first was to implement firewall rules using iptables to stop any heartbleed attacks and log them, as a trigger alarm technique I utilised Snort. The snort rules implemented stopped any potential attacks and output a message to the console and refusing a connection to the attacker. As an additional remediation technique, I chose a honeypot acting as a 'vulnerable server' which would trap would be adversaries to prevent attacks on the actual system, this allows us to keep any eye on any adversaries and gain valuable information on them, such as their IP address.

The shellshock bug is a part of the shellshock family of bugs and facilitates the bash function to exploit a backdoor into vulnerable machines, allowing DDoS attacks and probing. This particular family of bugs has a NIST rating of 10 out of 10 making it catastrophic to any victims if exploited fully. For the mitigations I used iptables firewall rules to log and drop any attempted shellshock attacks, as an alarm technique I utilised Suricata to monitor and get alerts of any potential attacks. As an additional remediation, I have temporarily implemented privilege mode for bash and ran the honeypot script to further harden the system against and ssh backdoor attacks.

These are the techniques employed on a temporary basis and I would not recommend use of most of these mitigations over the long term, it would be good practice to implement a NIDS system for future reference. The techniques all working in conjunction with each other will help mitigate attacks, making it less likely to be exploited, as with all bugs, a patch will be released soon, and I would recommend monitoring everything very closely with the techniques provided until the patches are released.

The issues in summary are; initially the virtual machines were modern and thus were not vulnerable, I opted to go with an earlier ubuntu version to make it easier to exploit, although was not always successful in my attempts to implement rules, software, logs and exploits.

6 Indexes

HeartbleedHoneyPot - https://demontfortuniversity-my.sharepoint.com/:u:/g/personal/p2629898_my365_dmu_ac_uk/EePI1uJvRbtGkqJpmF8SVToBpiIX4sXD77FEc6xY6t4u6Q?e=Sgmcr1

Heartbleed Snort Log:https://demontfortuniversity-my.sharepoint.com/:t:/g/personal/p2629898_my365_dmu_ac_uk/EUzoKGMYVC9HvCcWZVvUDrsBIuzXD7W38Io239b5VATVqw?e=RlLpIh

Logwatch Log for Both Exploits:https://demontfortuniversity-my.sharepoint.com/:t:/g/personal/p2629898_my365_dmu_ac_uk/EeFlq8yvRJRMuQufdrTswZYBIQtpUKXikOo5xRmsi9TF0A?e=A75lPi

7 Bibliography

[1]Heartbleed, "Heartbleed Bug," *heartbleed.com*, Jun. 03, 2020. https://heartbleed.com/

[2]NIST, "NVD - CVE-2014-0160," *Nist.gov*, 2014. https://nvd.nist.gov/vuln/detail/CVE-2014-0160

[3]OpenSourceSecurity, "Heartbleed, the OpenSSL vulnerability. What Should I Do? - Koen Van Impe - vanimpe.eu," *www.vanimpe.eu*, Apr. 09, 2014. https://www.vanimpe.eu/2014/04/09/heartbleed-openssl-vulnerability/

[4]123PCSolutions, "PROTECT YOURSELF FROM THE HEARTBLEED BUG," *Small Business IT Services*, Apr. 18, 2019. https://123pcsolutions.com/protect-yourself-from-the-heartbleed-bug/

[5]Glitch, "Heartbleed Honeypot Script ≈ Packet Storm," *packetstormsecurity.com*, Apr. 09, 2014. https://packetstormsecurity.com/files/126068/Heartbleed-Honeypot-Script.html (accessed May 01, 2022).

[6]Blogarama, "Snort - OpenSource Network Intrusion Detection Tool," *www.blogarama.com*, May 17, 2017. https://www.blogarama.com/technology-blogs/311585-effect-hacking-tools-guides-blog/20308760-snort-opensource-network-intrusion-detection-tool (accessed May 01, 2022).

[7]A. Waweru, "Honeypot Technique Technology and How it Works in Cyber Security," *TechPiton*, Oct. 15, 2019. https://techpiton.com/honeypot-technique-technology/

[8]TrendMicro, "About the Shellshock Vulnerability: The Basics of the 'Bash Bug' - Security News," *www.trendmicro.com*, Sep. 26, 2014. https://www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-exploits/the-shellshock-vulnerability-bash-bug

[9]T. Enache, "Shellshock Vulnerability," 2014. [Online]. Available: https://owasp.org/www-pdf-archive/Shellshock_-_Tudor_Enache.pdf

[10]Wikipedia Contributors, "Shellshock (software bug)," *Wikipedia*, Oct. 20, 2019. https://en.wikipedia.org/wiki/Shellshock_(software_bug)

[11]Suricata, "4. Suricata Rules — Suricata 4.1.0-dev documentation," *suricata.readthedocs.io*, 2016. https://suricata.readthedocs.io/en/suricata-4.1.4/rules/ (accessed May 01, 2022).

[12]Redhat, "Mitigating the shellshock vulnerability (CVE-2014-6271 and CVE-2014-7169)," *Red Hat Customer Portal*, Oct. 06, 2014. https://access.redhat.com/articles/1212303

[13]Wazuh, "Catch suspicious network traffic - Learning Wazuh," *documentation.wazuh.com*, 2014. https://documentation.wazuh.com/current/learning-wazuh/suricata.html#learning-wazuh-suricata (accessed May 01, 2022).

[14]Opensource, “Suricata Detect Dos Attack - Configuring the Suricata IDS to detect DoS attacks by adding custom rule file. - (Suricata-Detect-DoS-Attack),” *opensourcelibs.com*, 2014. https://opensourcelibs.com/lib/suricata-detect-dos-attack (accessed May 02, 2022).

[15]J. Wallen, “How to install and use Logwatch on Linux,” *TechRepublic*, Nov. 02, 2017. https://www.techrepublic.com/article/how-to-install-and-use-logwatch-on-linux/ (accessed May 02, 2022).