Task 1

We use arp - a to find ARP table on the target machine

```
C:\Users\user>arp -a

Interface: 10.13.37.103 --- 0xd
  Internet Address      Physical Address      Type
  10.13.37.1            08-00-27-ac-95-4c     dynamic
  10.13.37.104          08-00-27-d2-26-79     dynamic
  10.13.37.255          ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.1.85 --- 0xf
  Internet Address      Physical Address      Type
  192.168.1.254         18-58-80-19-8a-0a     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

Task 2

set the ARP spoofing module to attack the target machine

`set arp.spoof.targets 10.13.37.103`

verify if the arp.spoof.targets is set correctly

```
10.13.37.0/24 > 10.13.37.104 » get arp.spoof.targets

  arp.spoof.targets: '10.13.37.103'
```

Start spoof: `arp.spoof on`

```
10.13.37.0/24 > 10.13.37.104 » set arp.spoof.fullduplex true
10.13.37.0/24 > 10.13.37.104 » arp.spoof on
10.13.37.0/24 > 10.13.37.104 » [12:58:54] [sys.log] [inf] arp.spoof starting net.recon as a requirement for arp.spoof
10.13.37.0/24 > 10.13.37.104 » [12:58:54] [sys.log] [war] arp.spoof full duplex spoofing enabled, if the router has ARP spoofing mechanisms, the attack will f
ail.
10.13.37.0/24 > 10.13.37.104 » [12:58:54] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
10.13.37.0/24 > 10.13.37.104 » [12:58:54] [endpoint.new] endpoint 10.13.37.103 detected as 08:00:27:f9:39:0a (PCS Computer Systems GmbH).
```

Task 3

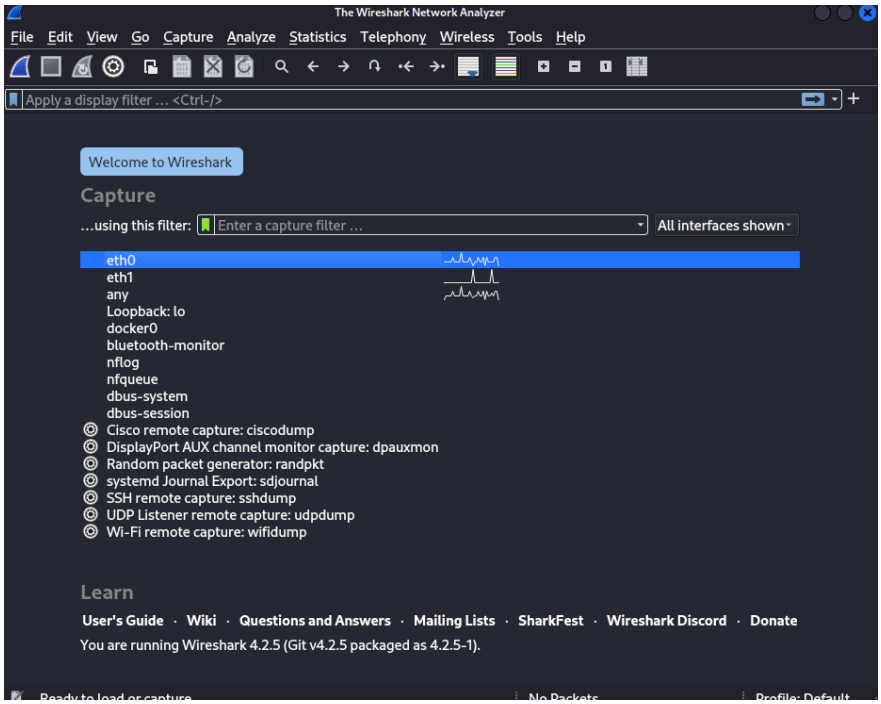
Before the attack: The ARP table showed the legitimate MAC address of the network gateway associated with the gateway's IP address (10.13.37.1).

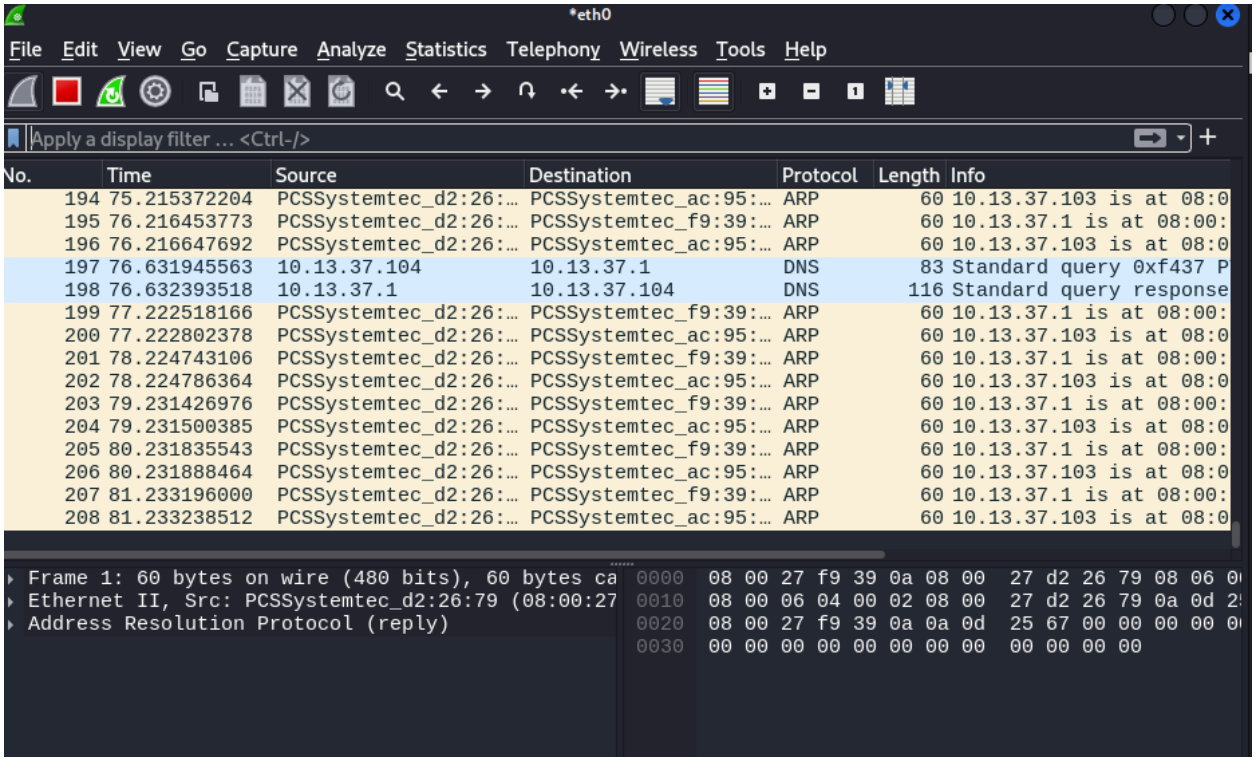
```
C:\Users\user>arp -a

Interface: 10.13.37.103 --- 0xd
  Internet Address      Physical Address      Type
  10.13.37.1            08-00-27-ac-95-4c     dynamic
  10.13.37.104          08-00-27-d2-26-79     dynamic
  10.13.37.255          ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.1.85 --- 0xf
  Internet Address      Physical Address      Type
  192.168.1.254         18-58-80-19-8a-0a     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

After the attack: The ARP table now shows the MAC address of the Kali machine (08-00-27-d2-26-79) as the MAC address for the gateway IP (10.13.37.1). It indicates that the Windows 7 machine is now sending traffic intended for the gateway to the Kali Linux machine instead, which confirms that ARP spoofing has been successful.

```
Interface: 10.13.37.103 --- 0xd
  Internet Address      Physical Address      Type
  10.13.37.1            08-00-27-d2-26-79     dynamic
  10.13.37.104          08-00-27-d2-26-79     dynamic
  10.13.37.255          ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.1.85 --- 0xf
  Internet Address      Physical Address      Type
  192.168.1.254         18-58-80-19-8a-0a     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

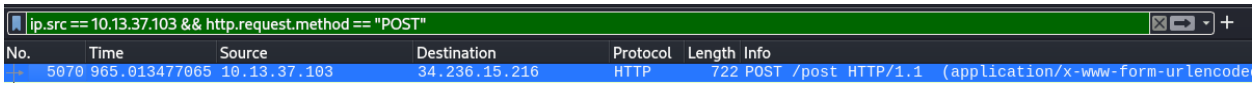
Task 4





Use http://httpbin.org/forms/post

Filter `ip.src == 10.13.37.103 && http.request.method == "POST"`



Task 5

We can’t, wireshark doesn't even have a HTTPS filter. We can only see the DNS lookup for google.com when the browser tries to resolve the domain to an IP address. This is because HTTPS encrypts all communication between the client and the server using SSL/TLS (Secure Socket Layer / Transport Layer Security). This encryption ensures that no one in the middle (like Wireshark) can see the content of the requests and responses, including headers, POST data, and GET parameters.

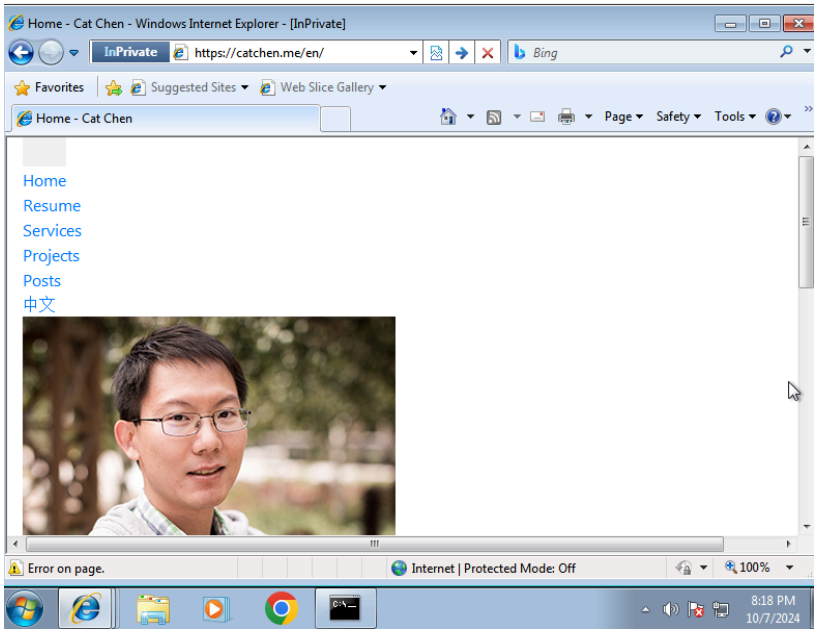
Task 6

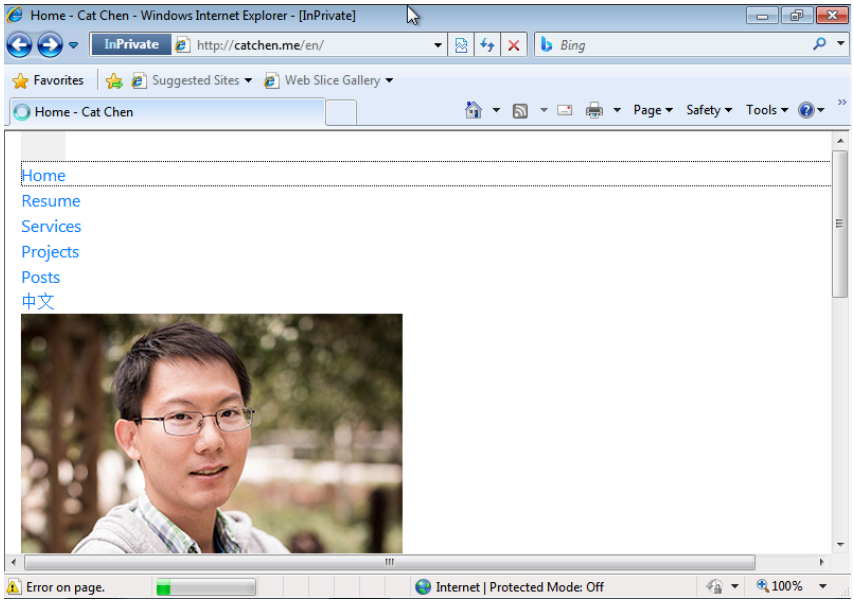
```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT
--to-port 10000
```

- t nat: This specifies we are modifying the NAT table, which is used for packet address translation (like port forwarding).
- A PREROUTING: This appends a rule to the PREROUTING chain. PREROUTING is used to alter packets as they arrive at the network interface (before they are routed).
- p tcp: This specifies that the rule applies to TCP packets (HTTP uses TCP).
- dport 80: This matches packets that are destined for port 80 (HTTP).
- j REDIRECT: This tells iptables to redirect the packet to a different port.
- to-port 10000: This redirects the traffic to port 10000, where SSLStrip will be listening for HTTP requests.

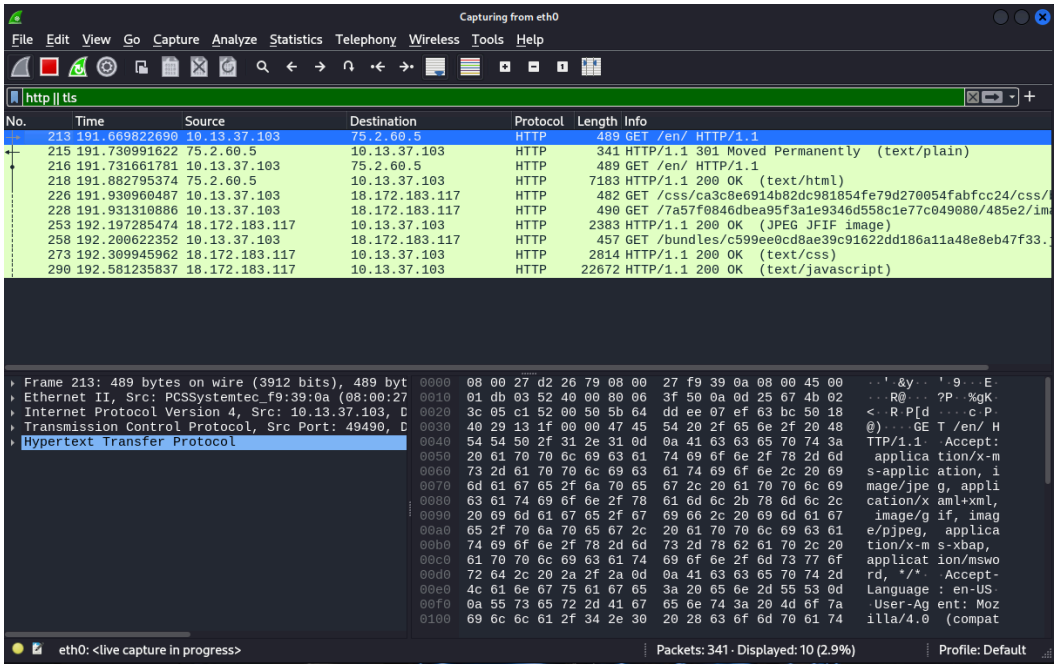
Task 7

Since lots of websites have high security features so I chose a personal website to do the task (Sorry Cat Chen)





We can see when we go to https it automatically downgraded the request to HTTP



We can also see the HTTP traffic in the Wireshark capture

Task 8

SSLStrip is a tool used in man-in-the-middle (MITM) attacks that enables an attacker to intercept and downgrade secure HTTPS connections to unencrypted HTTP connections. It operates by taking advantage of the initial unencrypted HTTP request that many users make when accessing a website. When a user attempts to connect to a website, SSLStrip intercepts the HTTP-to-HTTPS redirection and modifies it so that the victim's browser remains on an HTTP connection instead of upgrading to HTTPS. As a result, sensitive data, such as login credentials or personal information, can be transmitted in plaintext, making it visible to the attacker. Although the connection between the attacker and the target website remains encrypted, the attacker has full access to the plaintext data from the user's side.

Task 9

SSLStrip might not work on websites that have been previously visited due to the implementation of a security feature called HTTP Strict Transport Security (HSTS). HSTS is a policy mechanism that informs the browser to always use HTTPS when communicating with a specific website, even if the user attempts to access it via HTTP. Once a browser receives an HSTS policy from a site, it stores this instruction and automatically forces all future connections to be secured with HTTPS, effectively preventing any attempts to downgrade the connection to HTTP. This mechanism protects the website from SSLStrip attacks because the browser no longer makes any unencrypted HTTP requests to that site, thereby maintaining a secure connection.

Task 10

We can check if IP forwarding is enabled

```
┌──(root㉿kali)-[~/Documents/Tools/sslstrip]
└─# cat /proc/sys/net/ipv4/ip_forward
1
```

Ip forwarding is ready
Then for ICMP Redirect

```
┌──(root㉿kali)-[~/Documents/Tools/sslstrip]
└─# cat /proc/sys/net/ipv4/conf/all/accept_redirects

0
```

Looks like they are both ready
If they were not then we can use `echo value > <file path>` to change them
We then need to set up iptables rules to redirect traffic from ports 80 (HTTP) and 443 (HTTPS) to port 8080 where mitmproxy will be listening.

```
┌──(root㉿kali)-[~]
└─# sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 8080
┌──(root㉿kali)-[~]
└─# sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080
```

Task 11

Redirecting ports 80 and 443 to port 8080 is important for mitmproxy because these ports handle the majority of web traffic. Port 80 is the standard port for HTTP communications, where data is transmitted in plaintext without encryption. Redirecting traffic from this port allows mitmproxy to directly capture and analyze all HTTP data flowing through the network. Port 443 is used for HTTPS traffic, which is encrypted using SSL/TLS to ensure data security. By redirecting traffic from port 443, mitmproxy can decrypt this secure traffic and inspect its contents. This redirection ensures that mitmproxy can intercept, view, and modify both HTTP and HTTPS communications and performing man-in-the-middle attacks. Without redirecting these two ports, mitmproxy would be unable to capture most of the web traffic.

Task 12

The --mode transparent flag in mitmproxy configures it can intercept traffic without requiring any manual configuration changes on the target machine. In this mode, mitmproxy acts as an invisible intermediary, capturing and inspecting traffic that is redirected to it by iptables rules, making it ideal for man-in-the-middle attacks. The

--showhost flag instructs mitmproxy to display the hostname of the destination server instead of just showing its IP address in the console output. This makes it easier to understand which domains the target machine is communicating with. Together, these flags enhance mitmproxy's ability to silently intercept and present web traffic in a more user-friendly manner.

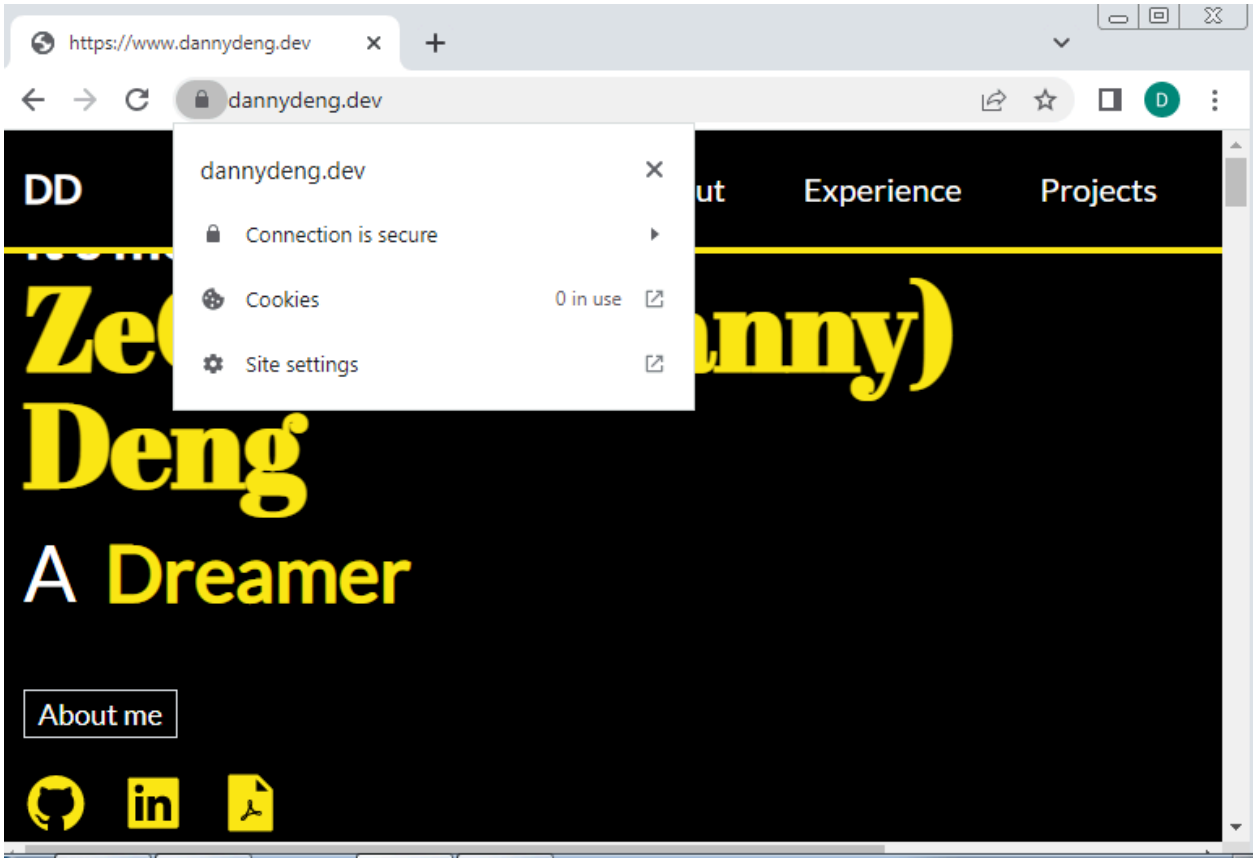
Task13

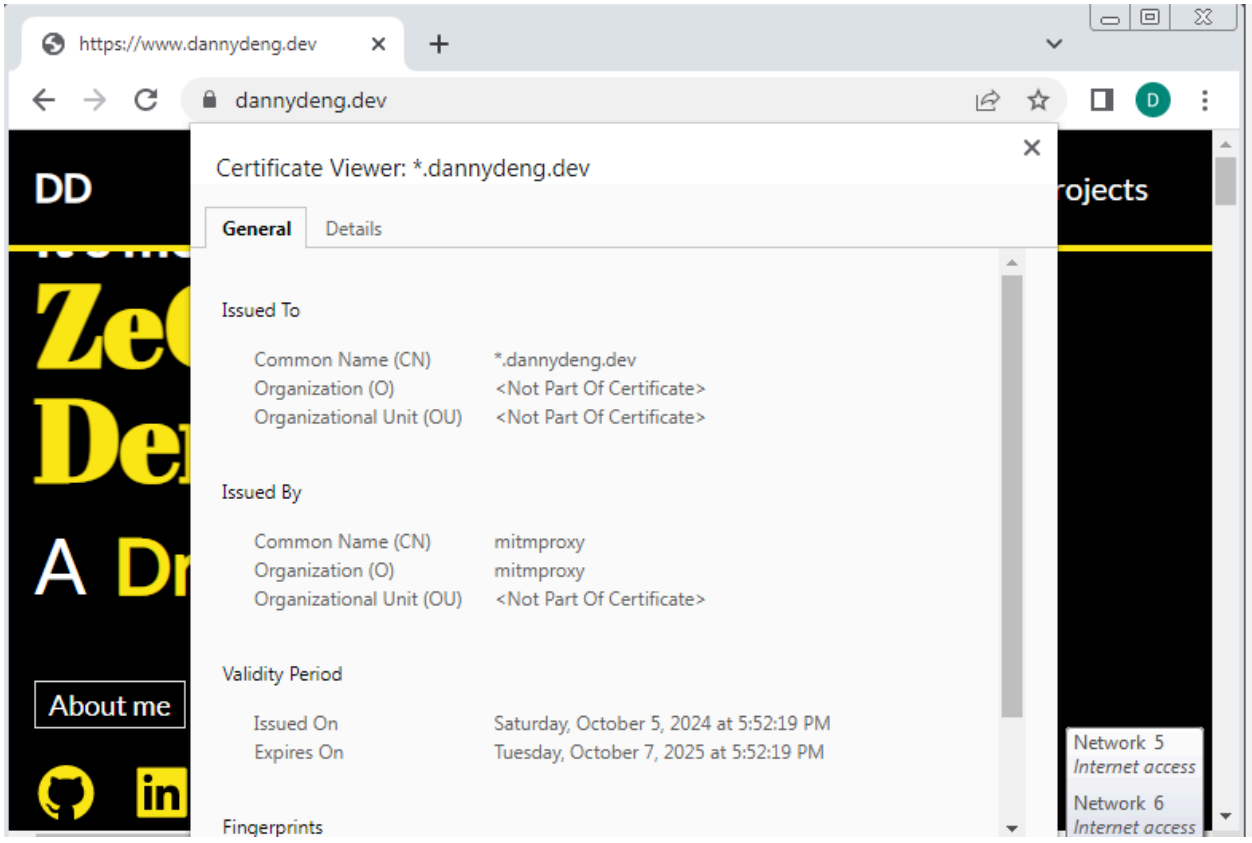
In my opinion, it functions as MITM by intercepting and analyzing web traffic between a client and a server. It works by capturing HTTP and HTTPS traffic that passes through it and allows for inspection, modification, or recording of that traffic. When using --mode transparent, mitmproxy can intercept traffic that has been redirected to it via iptables, without requiring the target machine to configure any proxy settings. This makes the interception process almost invisible to the user
If we did not install the certificate on the target machine, the browser would display security warnings when trying to visit HTTPS websites. This is because the browser would detect that the encrypted connection has been tampered with by an untrusted third party, which breaks the SSL/TLS encryption. Additionally, if we did not use --mode transparent, mitmproxy would not be able to automatically intercept traffic without manual configuration on the client machine. We would have to set the proxy settings in the client's browser or system to point to mitmproxy, which would alert the user to the interception.

Task14

Used my personal website this time

1. Click on the padlock icon in the address bar next to the URL.
2. Click on "Connection is secure"
3. Check the Certificate Issuer


https://www.dannydeng.dev
dannydeng.dev
dannydeng.dev
Connection is secure
Cookies
0 in use
Site settings
DD
Experience
Projects
Deng
A Dreamer
About me


https://www.dannydeng.dev
dannydeng.dev
Certificate Viewer: *.dannydeng.dev
General
Details
Issued To
Common Name (CN) *.dannydeng.dev
Organization (O) <Not Part Of Certificate>
Organizational Unit (OU) <Not Part Of Certificate>
Issued By
Common Name (CN) mitmproxy
Organization (O) mitmproxy
Organizational Unit (OU) <Not Part Of Certificate>
Validity Period
Issued On Saturday, October 5, 2024 at 5:52:19 PM
Expires On Tuesday, October 7, 2025 at 5:52:19 PM
Fingerprints
DD
About me
Network 5
Internet access
Network 6
Internet access