**ICT2203 Assignment Report**

Done by: Gu Boyuan#1, Lim Wei Le#2, Ng Ee Zen#3 & Lee Ray Zan, Zane#4

2103210@sit.singaporetech.edu.sg#1;2103205@sit.singaporetech.edu.sg#2;2103204@sit.singaporetech.edu.sg#3; 2103208@sit.singaporetech.edu.sg#4;

***Introduction***

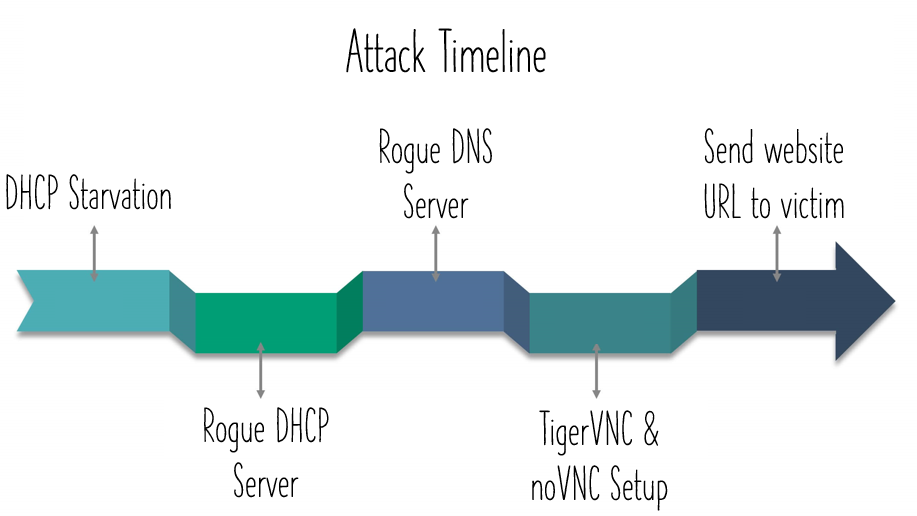
**Our Objective**

2-Factor Authentication (2FA) is a form of security recently adopted by the cyber industry. Hackers are unable to access a compromised account even with the correct username and password, as an additional time-based token (only available on another of the user’s device) is required for successful authentication. The purpose of our project is to create a method of bypassing 2FA which is an industry standard in websites requiring high levels of security. 2FA is commonly used in bank websites and government websites, and there are currently very few ways in bypassing such a high layer of security. Our attack primarily focuses on session hijacking of the victim via a man-in-the-middle (MITM) attack

**The Premise**

Virtual Network Connection (VNC) is a protocol used to connect and remotely control a computer. It is commonly used in network and server administration. Our attack implements noVNC which is a HTTP-based VNC software to allow our victim to control the attacker’s computer on the network, tricking them into logging into a secure website on an insecure machine via a web browser. DNS spoofing is used in a LAN to mask the remote IP of the attacker to the legitimate site which the user is trying to access. With some custom modifications in noVNC, its UI features and desktop can be hidden. This leaves only a web browser connecting to the secure website on the attacker’s computer, which the user then connects to and accesses the website from.

***Attack timeline***



*Fig 1. Attack Timeline*

***Softwares Used***

Scapy (Python module)

Randmac (Python module)

Pynput (Python module)

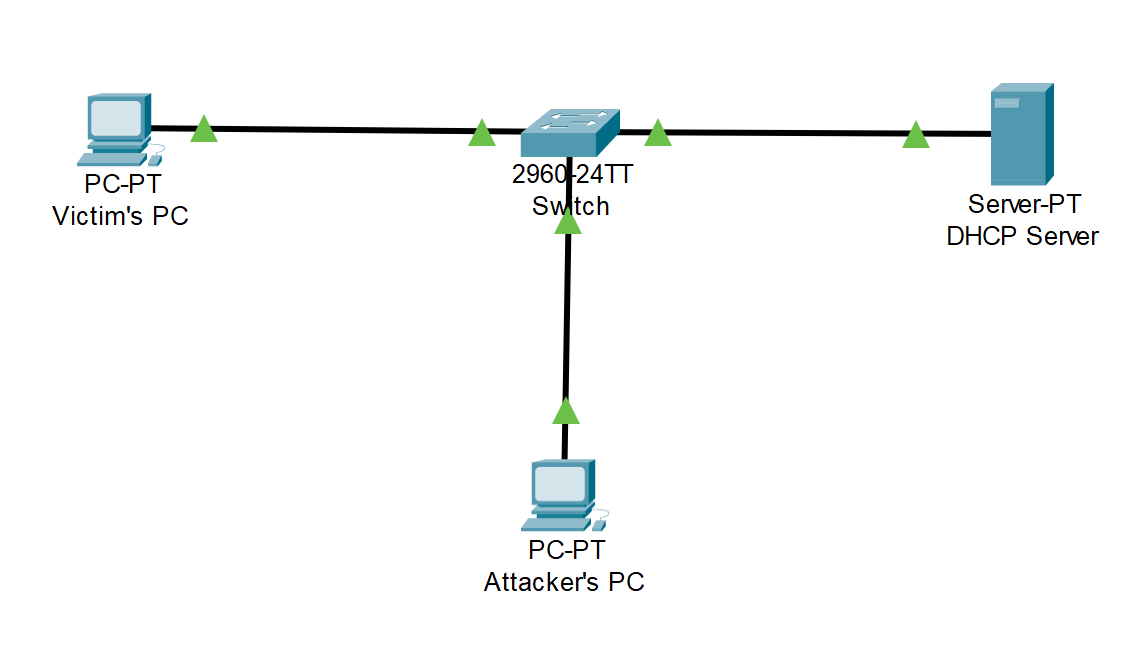
TigerVNC (Linux package)

Xfce4 (Linux package)

NoVNC (Open-source web interface for VNC)

***Attack Workflow***

**Overview**



**Legitimate Default Gateway - 10.0.0.1**

**Legitimate DHCP and DNS server - 10.0.0.1**

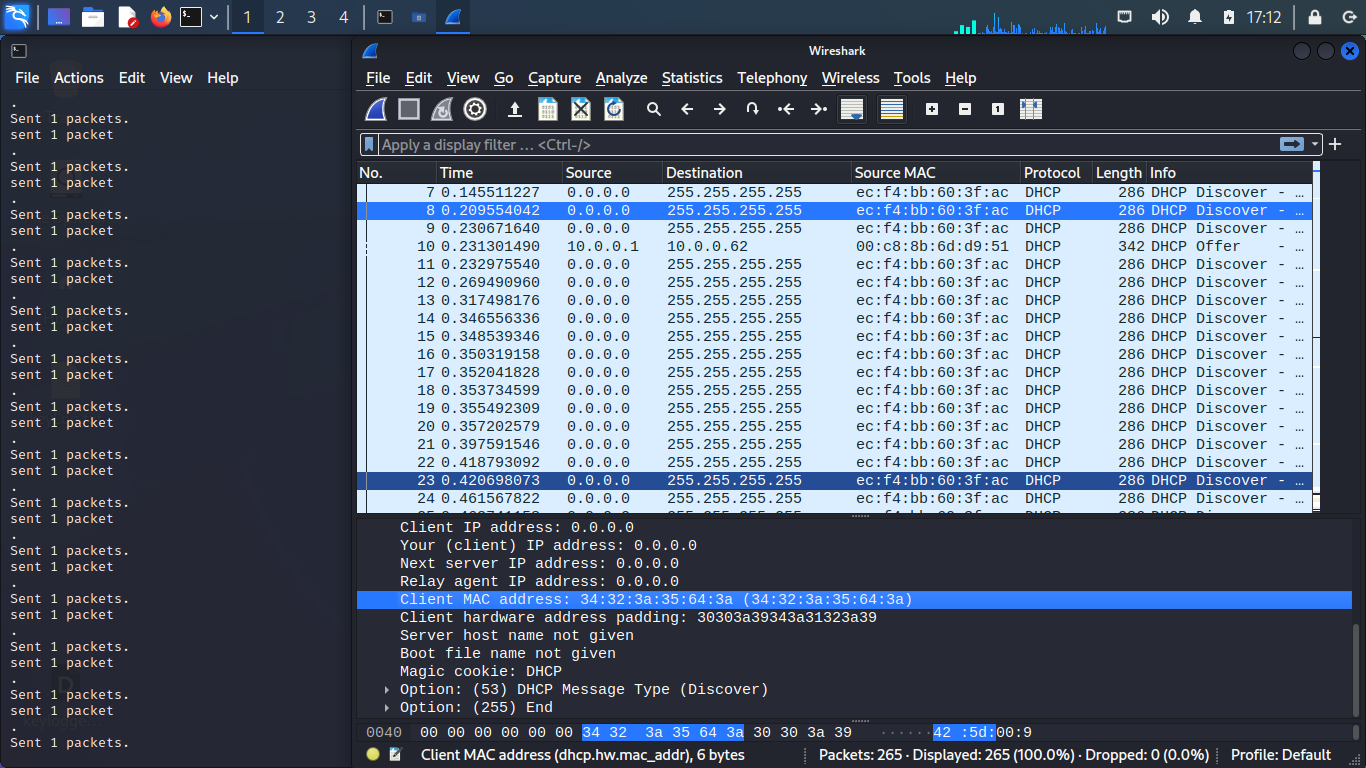
**Attacker IP - 10.0.0.33**

**VNC Web Server IP - 10.0.0.33**

**Victim IP - 10.0.0.128**

**DHCP Starvation**

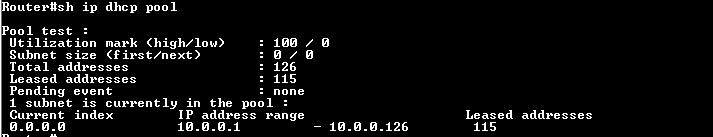
The first stage of our attack involves exhausting all IP addresses of the legitimate DHCP server so that our spoofed DHCP server will be the only one allocating addresses. This attack exploits the fact that a DHCP server reserves an IP address from its available pool of IP addresses to offer the IP address after receiving a DHCP Discover packet. Hence, by flooding the DHCP server with DHCP Discover packets, all the available IP addresses can be exhausted in the pool.



*Fig 2. Screenshot of DHCP Discover Packets Sent by Attacker to Perform DHCP Starvation Attack*

A DHCP server maintains a DHCP binding table that matches a MAC address to an IP address. Hence, the packets sent out need to contain different MAC addresses to successfully exhaust the pool. Since the DHCP server only looks at the MAC address provided by the application level part of the packet, we are able to send out packets with the attacker’s real MAC address in the ethernet layer, and a random MAC address in the application layer, as seen in Fig 2. By sending packets with the same ethernet layer MAC address, we are able to bypass port security, which limits the number of MAC addresses allowed on a port as port security only looks at the ethernet layer source MAC address.

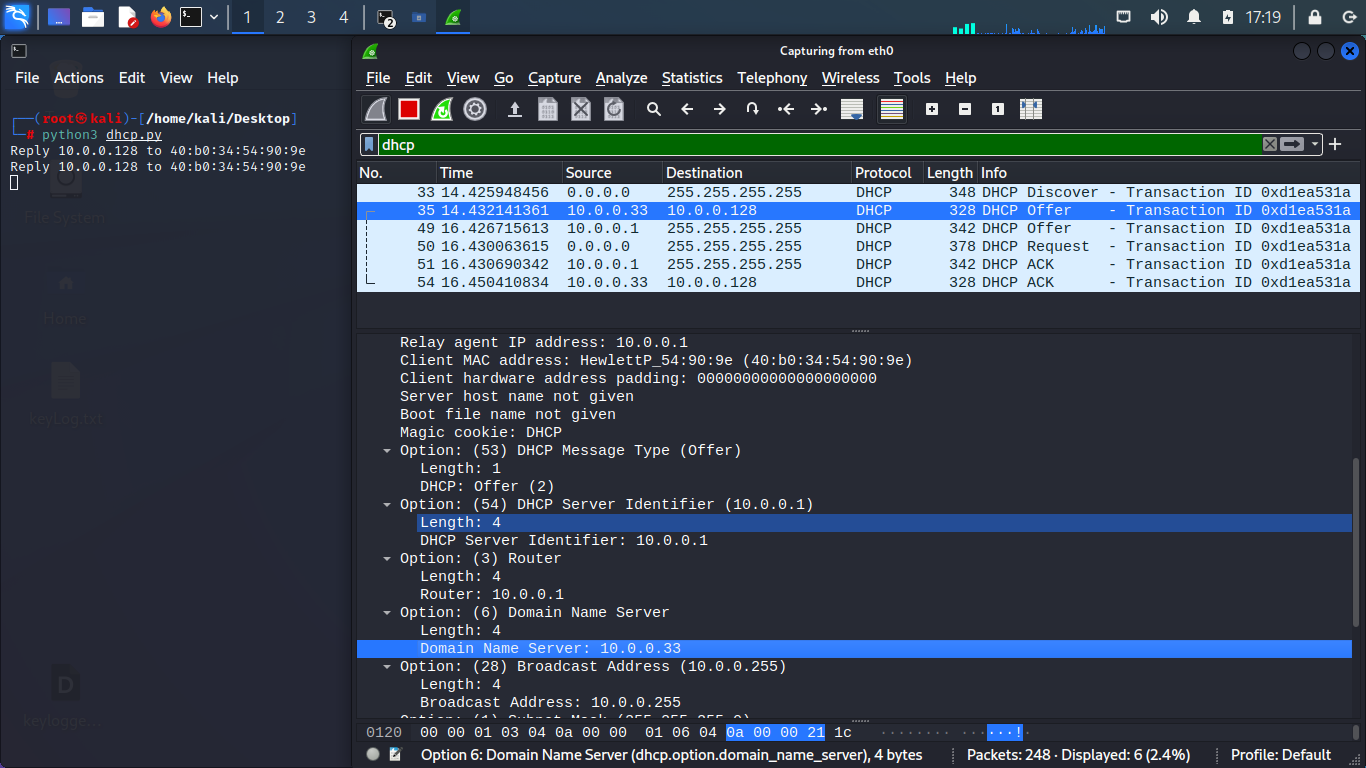
Our DHCPFlood.py script will keep generating random MAC addresses and sending out DHCP Discover packets with those MAC addresses in the application layer, eventually exhausting the DHCP server’s pool of available addresses, as seen in Fig 3., and the DHCP server will no longer respond to new DHCP Discover messages, allowing us to perform the next attack - DHCP server spoofing.



*Fig 3. Screenshot of the DHCP Server’s Exhausted Pool of IP Addresses*

**Rogue DHCP Server**

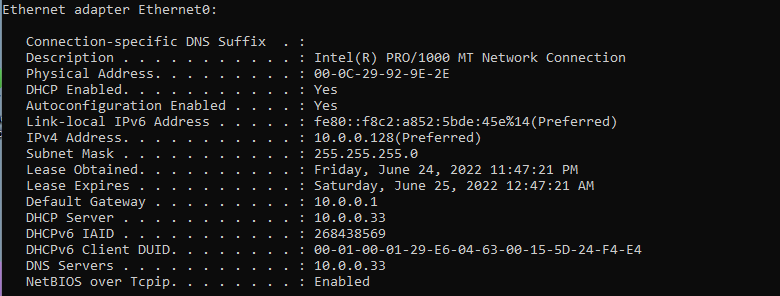
Since the real DHCP is no longer responding to DHCP Discover messages, we are able to run a rogue DHCP server on our attacker’s machine which a new user in the network will request an IP address from.



*Fig 4. Screenshot of rogue DHCP Server telling Victim to Set Its DNS Server to Attacker’s IP*

The DHCP Server not only provides the new user with an IP address, it also provides the user with the IP address of the DNS server. This allows the DNS server of the user to be determined by the attacker. In this case, our dhcp.py script sends back a DHCP offer that tells the new user that the new user that the rogue DHCP server is also the DNS server, as seen in Fig 4.

As seen in Fig 5., the victim now recognises the attacker as the DNS server and will now send DNS queries to the attacker, allowing the attacker to perform the next step - DNS spoofing.

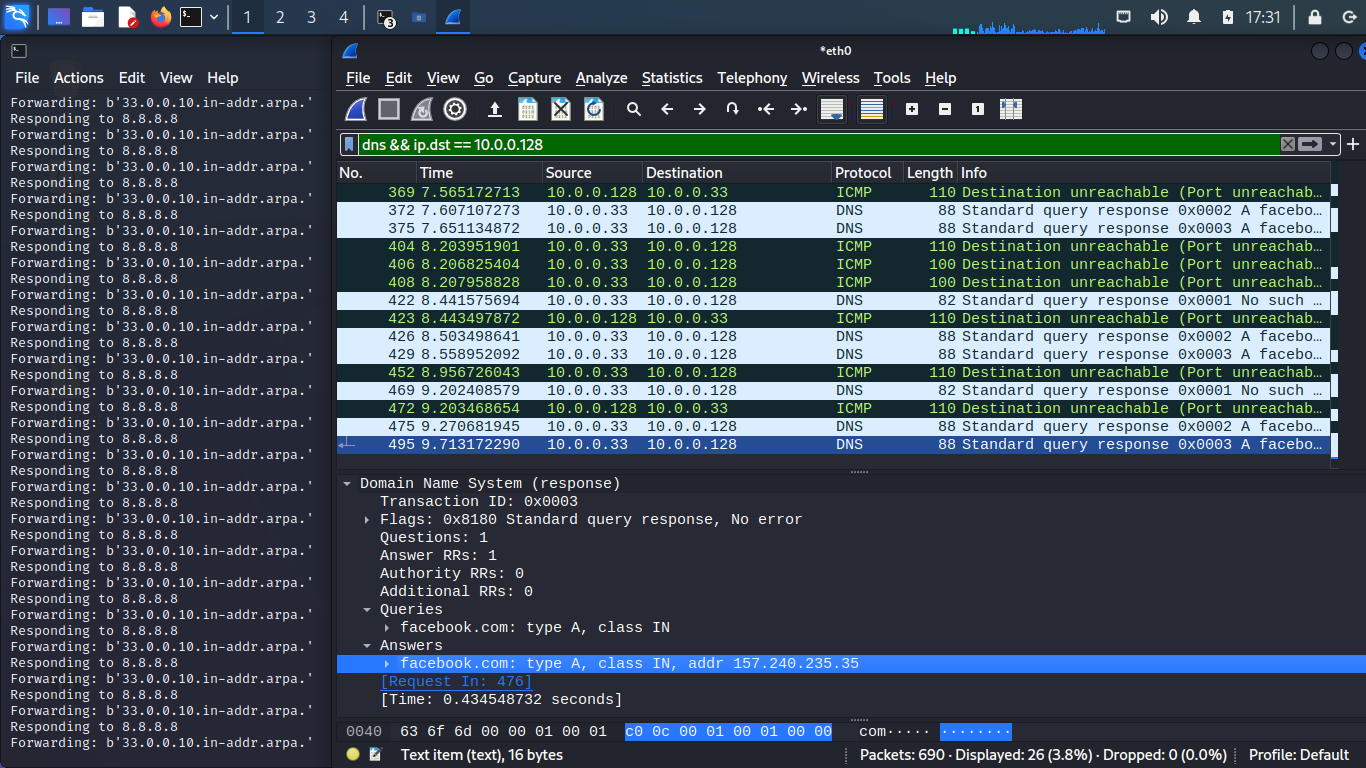


*Fig 5. Screenshot of Victim’s ipconfig after the Attack*

**Rogue DNS Server**

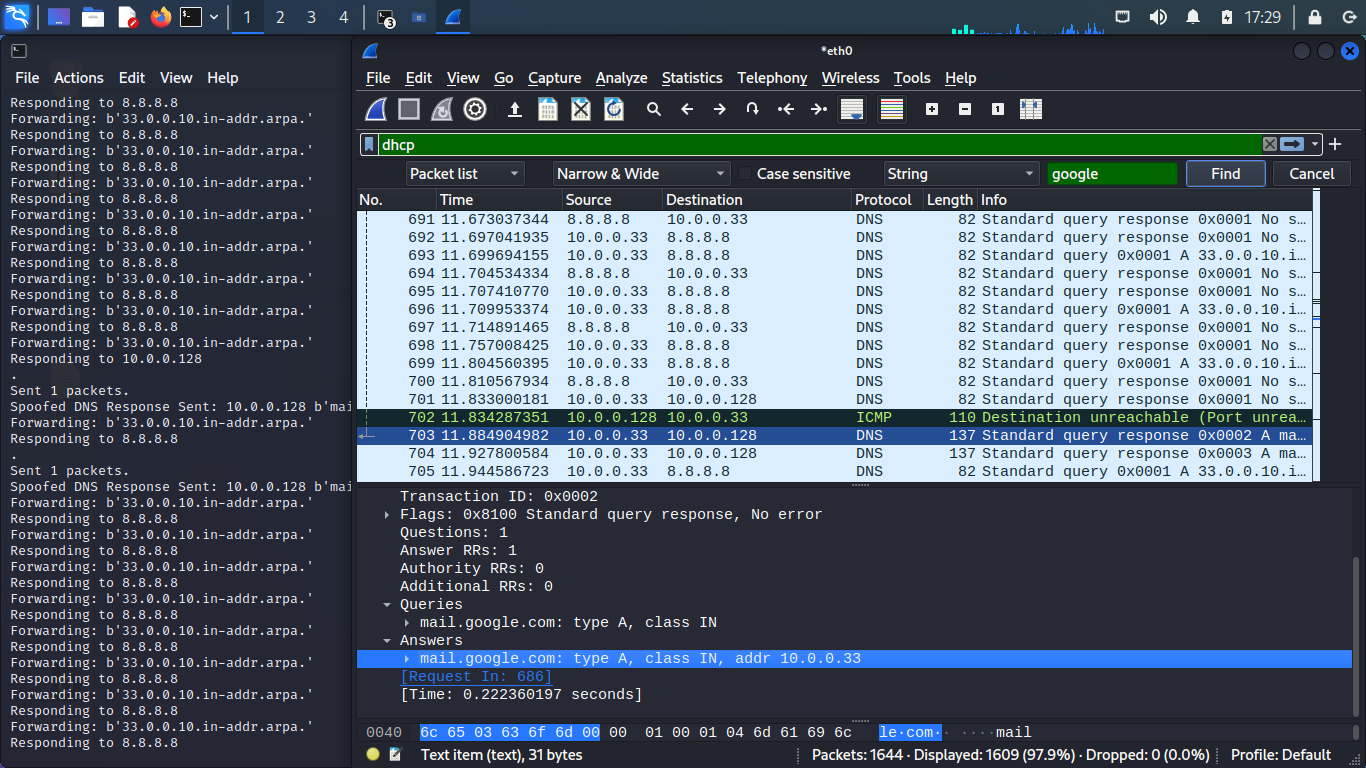
Since the attacker is now recognised as the DNS server by the new user, the attacker can now spoof DNS responses which can direct the victim to websites determined by the attacker.

As seen in Fig 6., in this case, the attacker will forward legitimate DNS responses for most sites, allowing the user to visit the legitimate site, and avoiding suspicion.



*Fig 6. Screenshot of Rogue DNS Server - Real Response for Most Sites*

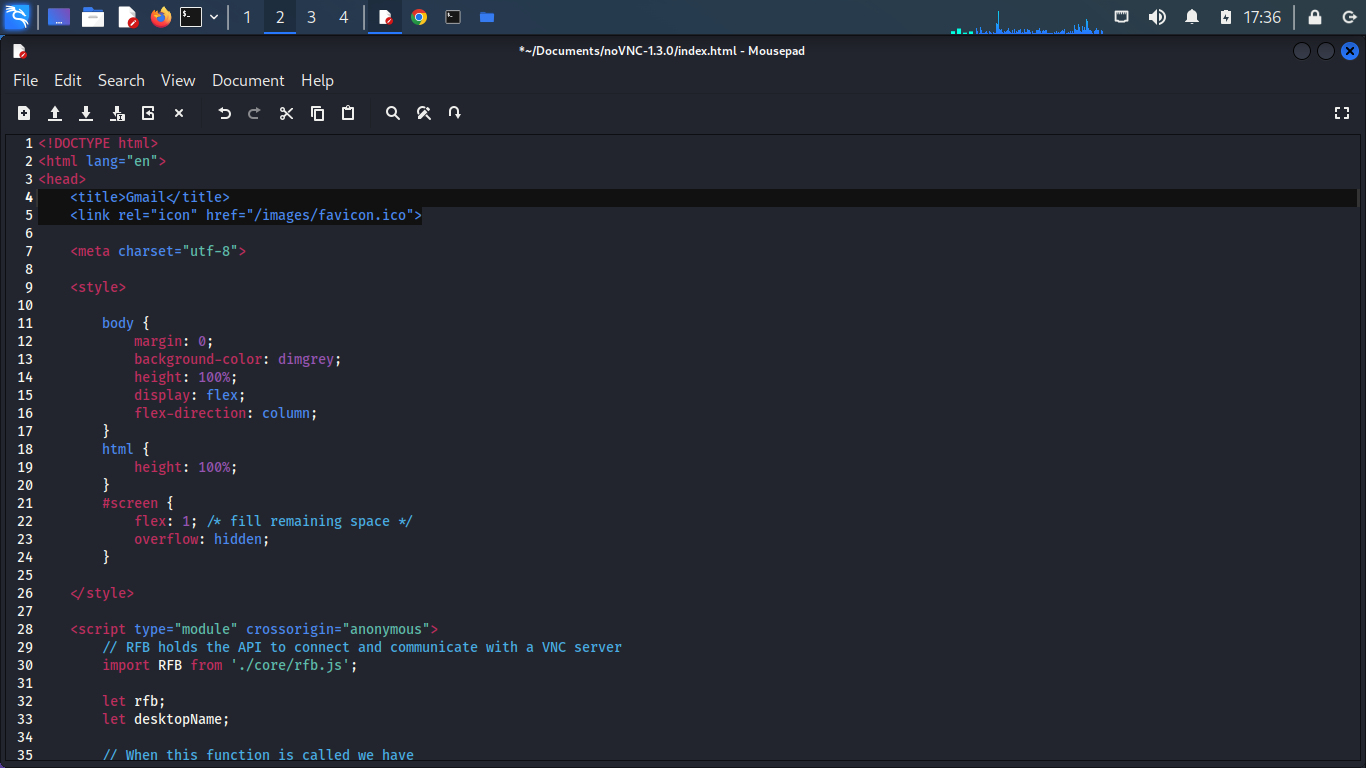
However, the attacker can choose to send a spoof response to the user when certain domains are queried. As seen in Fig 7., when “mail.google.com” is queried, the response is an IP address determined by the attacker, allowing the attacker to direct the user to the attacker’s web server when the user tries to go to “mail.google.com”. In this case, the attacker will direct the user to a VNC web server, where the attacker can obtain various information like credentials, and even take control of the user’s session.



*Fig 7. Screenshot of Rogue DNS Server - Spoofed Response For “mail.google.com”*

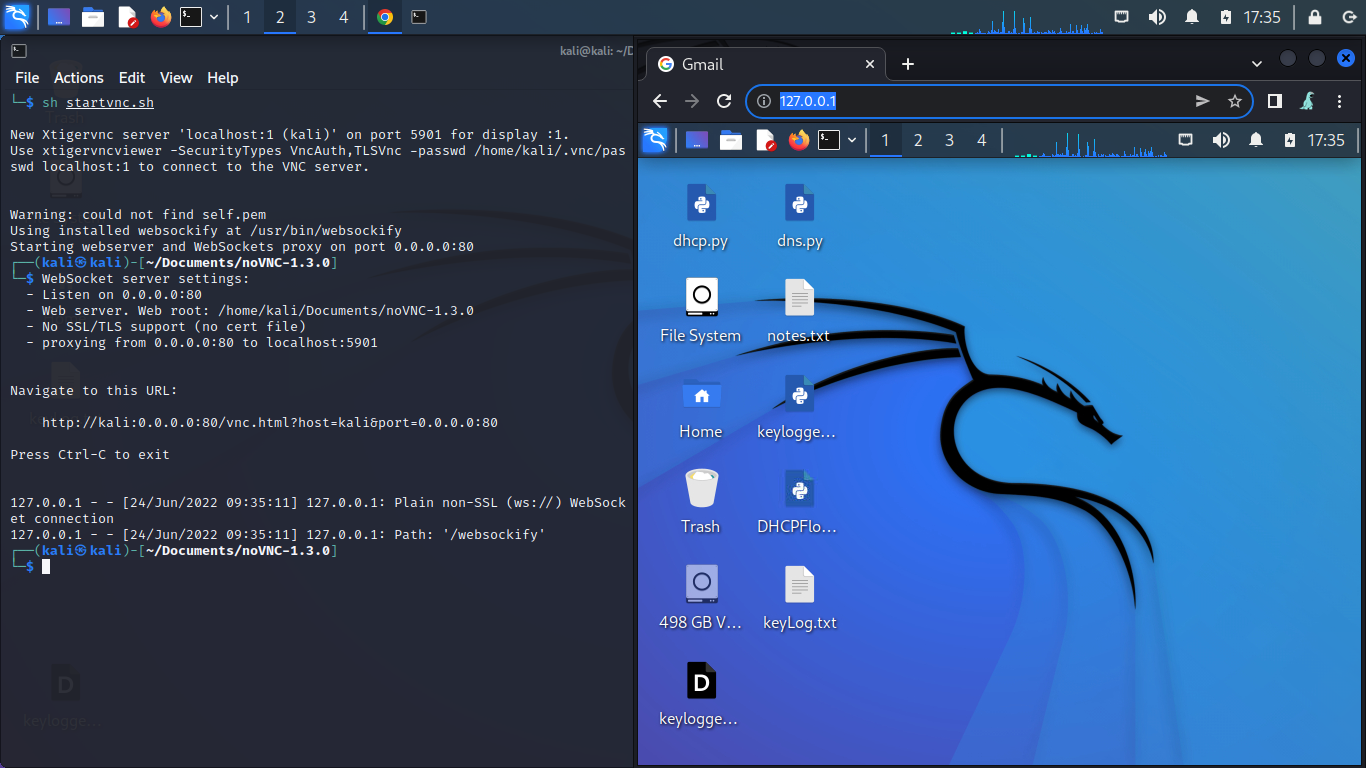
**VNC Web Server Setup**

Before starting the VNC server on our attacker’s machine, we need to change the website’s title and favicon in index.html which will be seen by the victim, mimicking the legitimate Gmail website. The changes can be seen in Fig 8.



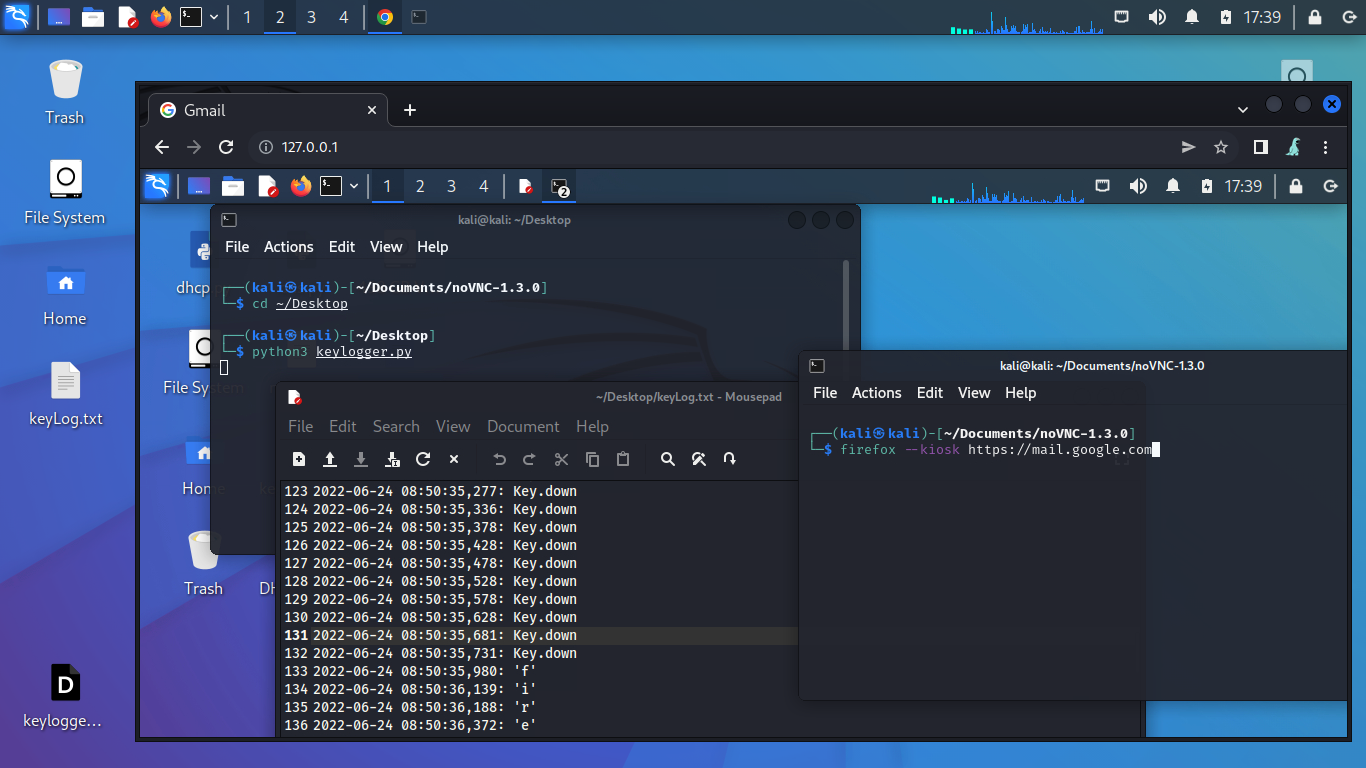
*Fig 8. Screenshot showing Modification of Attacker’s VNC Web Server Title and Favicon to mimic Gmail*

Next, we start TigerVNC on the attacker as the backend VNC server software, and noVNC for the frontend client that our victim will connect to. To test that our setup is working, we can connect to our local address on the attacker’s machine. Shown inside should be an exact replica of the attacker’s Kali Linux desktop accessible from a web browser. The title and favicon can be changed in index.html as explained in the previous step. Afterwards, update TigerVNC’s config (~/.vnc/xstartup) to launch the Xfce4 desktop on startup. Fig 9. shows our setup working successfully.



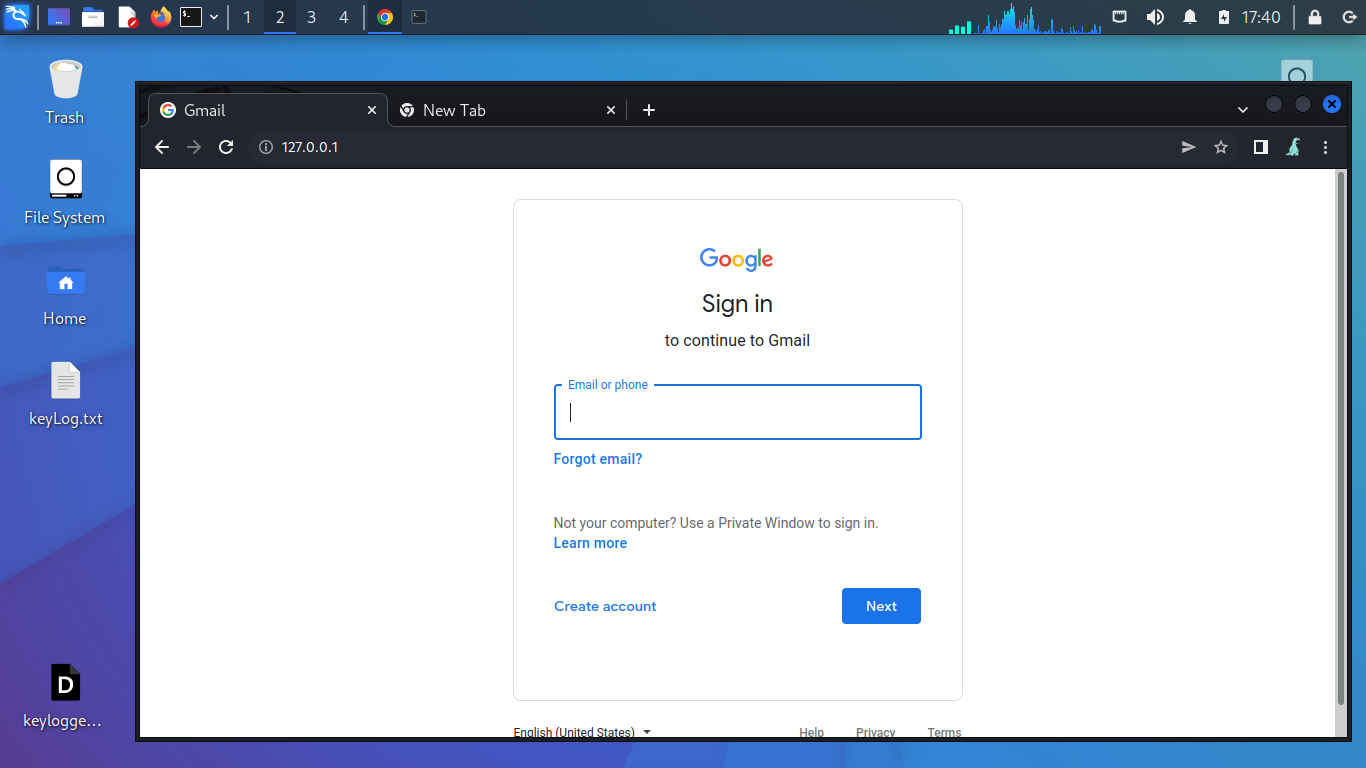
*Fig 9. Screenshot Showing the Remote Desktop Being Accessible from a Web Browser*

Inside the attacker’s own remote desktop accessed from the web browser, we need to do some setup. First, we run our own keylogger in the background to capture any keystrokes the victim may input (optional). Next, we run an instance of Firefox in Kiosk mode. Kiosk mode shows a web page while hiding browser features like the site URL and reload feature, making it useful self-serve machines commonly found in fast food restaurants. This is perfect for our attack as it will hide the attacker’s desktop and the web browser interface, showing only content in the legitimate site. Fig 10. shows the various setup that is done.



*Fig 10. Screenshot of Attacker Locally Accessing the VNC server to Set Up and Running the Legitimate mail.google.com in Firefox’s Kiosk Mode*

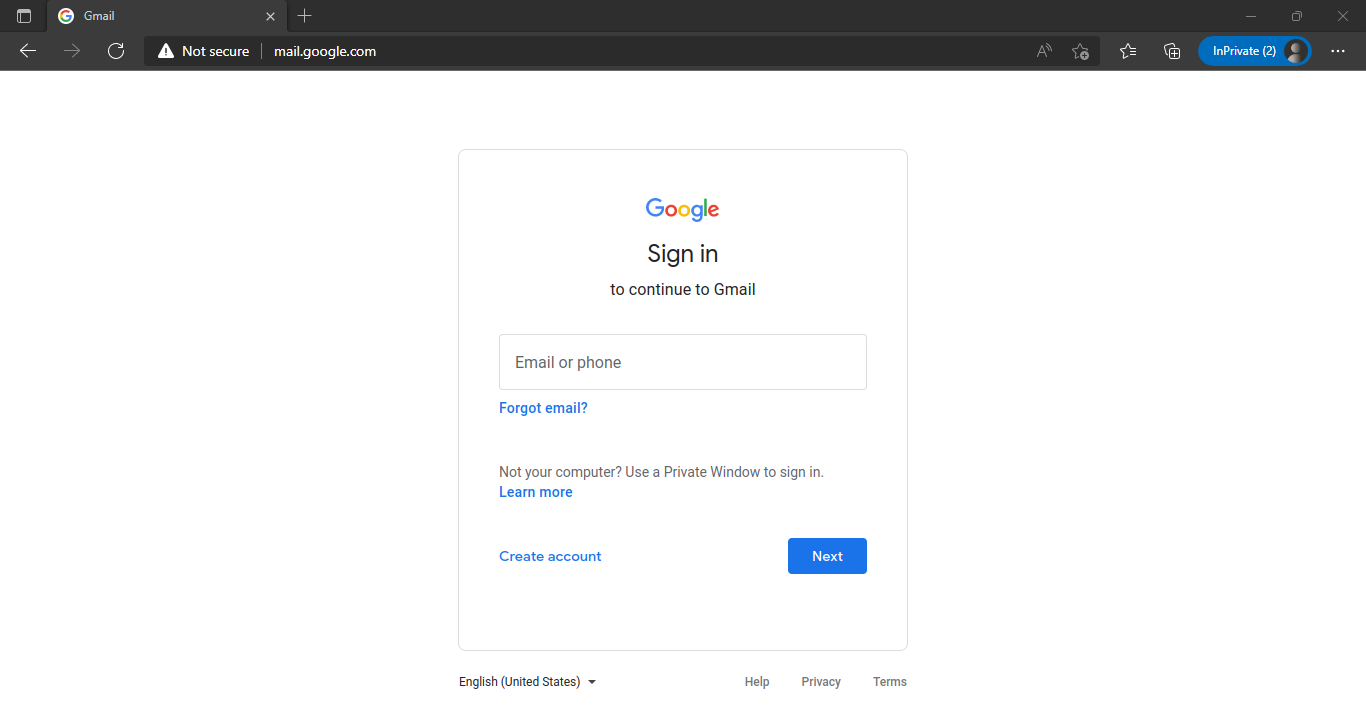
Once we run Firefox in Kiosk mode, the attacker’s VNC web interface is indistinguishable from the actual website. The attacker’s machine is now ready to be used as a MITM for the victim to access any legitimate site.



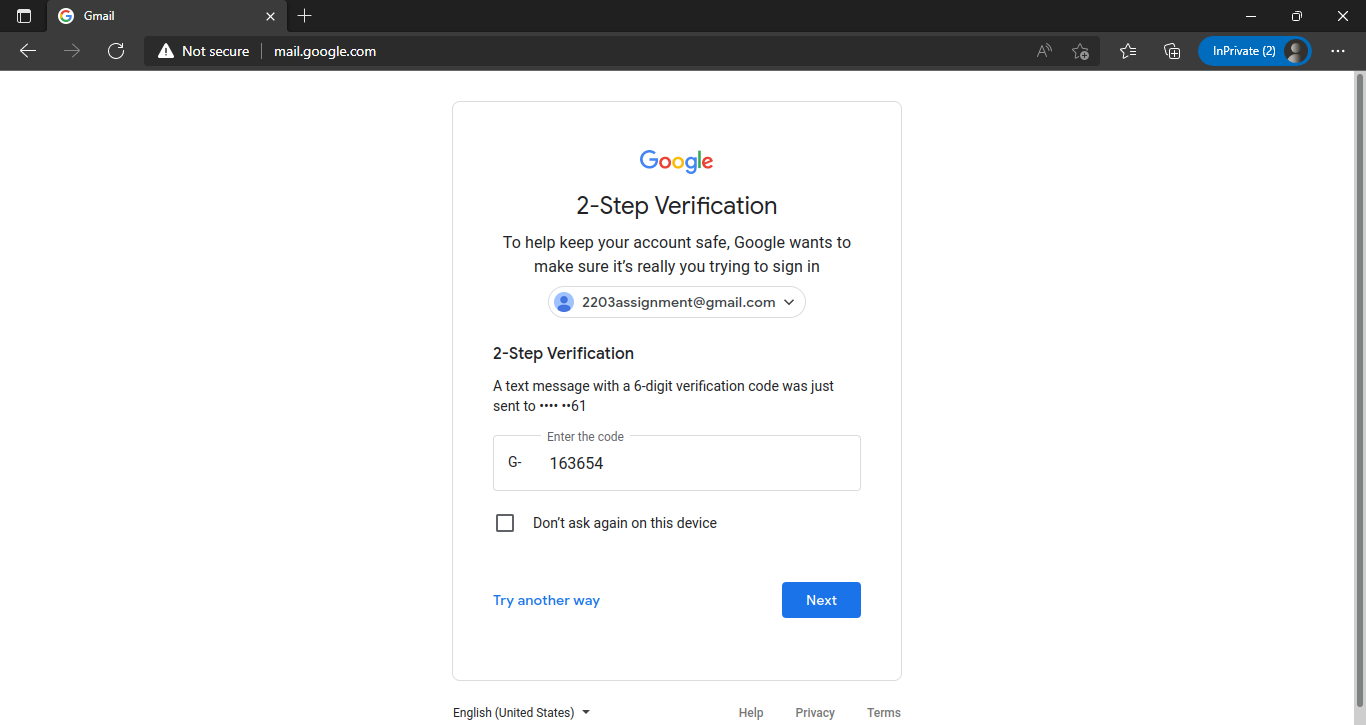
*Fig 11. Screenshot of VNC Web Interface from Attacker’s POV, with Kiosk Mode Running*

Since the DNS server of the victim is set to our attacker’s IP as explained in our previous attack, we are able to spoof any domain of our choice and direct the user to the attacker’s VNC web server.

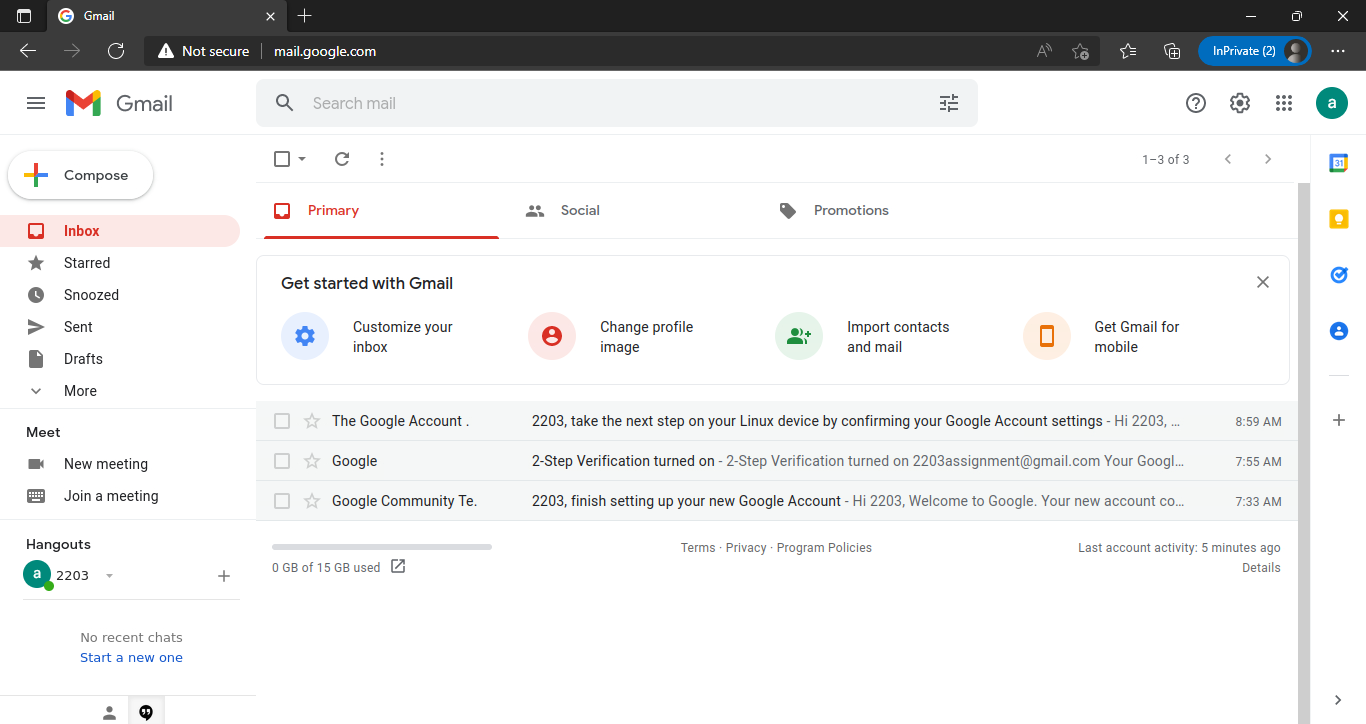
As seen from Fig 12., from the user’s point of view the website is indistinguishable from the legitimate website. The user can then log into their Gmail account, and type in their 2FA code if it is enabled, as seen in Fig 13.



*Fig 12. Screenshot of Victim Accessing Attacker’s VNC Web Server*

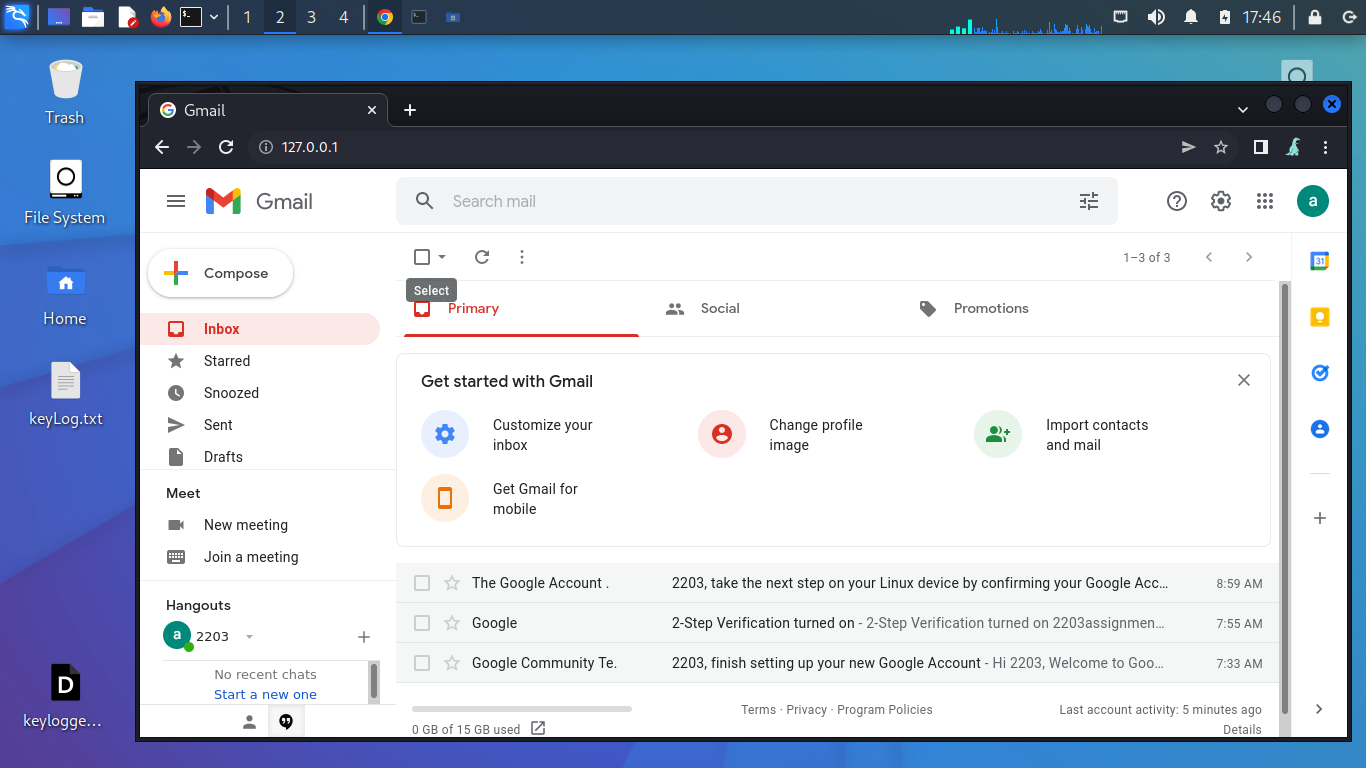


*Fig 13. Screenshot of Victim Entering 2FA for His/Her Account*



*Fig 14. Screenshot of Victim Successfully Logging In to His/Her Account*

From our attacker’s machine, we are able to see any action taken by the user in real-time. If the attacker desires to, they can disconnect the victim and take back control of the VNC connection with the user’s session to any legitimate website. Hence, the attacker can view the various emails in the victim’s account after the victim logs in, as seen in Fig 14., and taking control afterwards, as seen in Fig 15. Our keylogger also captures any keystrokes that may be hidden by the site’s password field.



*Fig 15. Screenshot of Logged In Page from Attacker*

***Attack Mitigation***

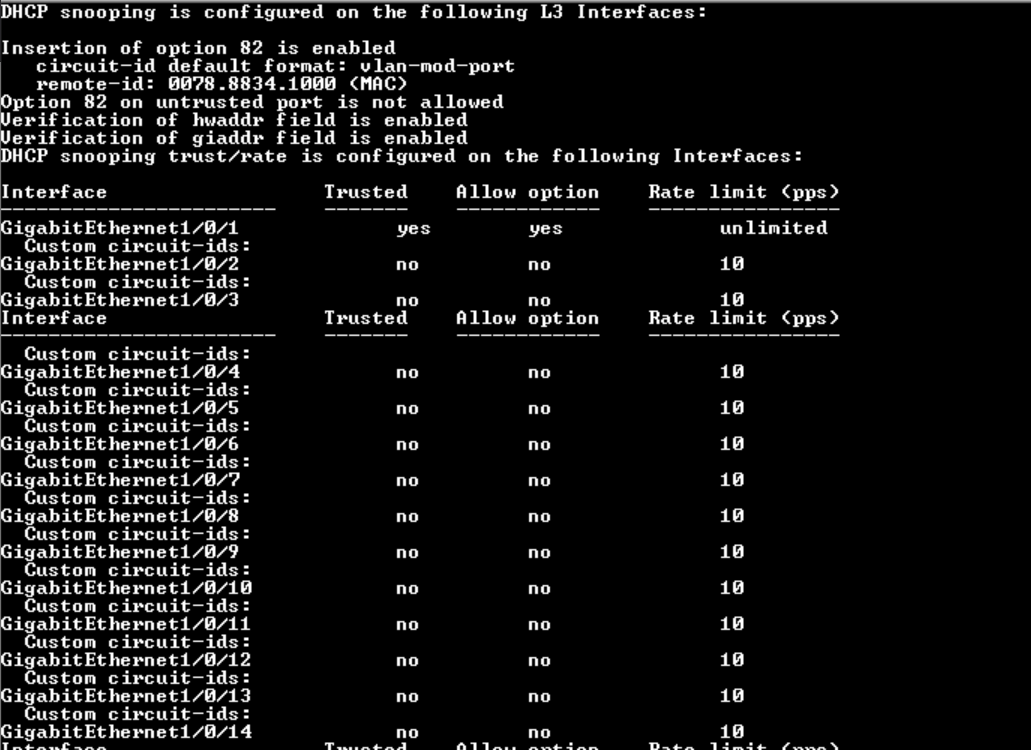
**Mitigate DHCP Starvation**

Since this attack depends fully on the DHCP server and DHCP spoofing attacks, it can be mitigated by enabling DHCP snooping, which allows the switch to perform deep packet inspection of higher layers. The enabling of DHCP snooping can be seen in Fig 16.



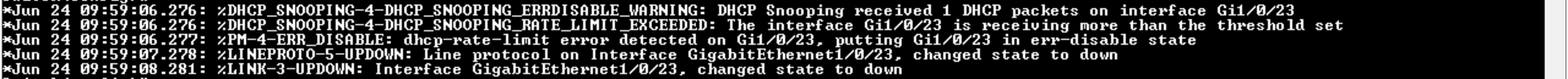
*Fig 16. Screenshot of DHCP Snooping Command Being Issued*

Further configuration of DHCP snooping will allow various functions, such as verifying that the layer 2 source MAC address provided matches the DHCP application layer MAC address, which will prevent DHCP starvation attacks as the malicious DHCP Discover packets will not reach the DHCP server. DHCP snooping rate-limiting can also be configured to prevent the spamming of DHCP packets, as seen in Fig 17.



*Fig 17. Screenshot of DHCP Snooping Rate Limit*

As seen in Fig 18., the attack is now unable to take place as the attacker’s connection to the network is shut down due to the attempted DHCP flooding.



*Fig 18. Screenshot of DHCP Snooping Working Successfully*

***Conclusion***

**Strengths of This Attack**

**Difficult to Detect**

If executed correctly, this form of attack is challenging to detect from the victim’s perspective. DNS spoofing makes the URL of our fake webserver look legitimate, and the title and favicon of our page can be modified to match the site we are trying to gain credentials for. Moreover, the page content itself is completely identical to the legitimate site since it is being accessed from the attacker’s machine. As such, it will be difficult for the victim to learn that their session can be stolen.

**Bypasses Sites that Enforce HSTS**

HTTP Strict Transport Security (HSTS) is a policy mechanism that helps to protect websites against man-in-the-middle attacks such as protocol downgrade attacks and cookie hijacking. It allows web servers to declare that web browsers should automatically interact with it using only HTTPS connections. While the connection from the victim to our attacker’s web server is in HTTP, the connection between our attacker and the legitimate website is in full HTTPS. A website would not be able to detect the HTTP downgrade since our attacker acts as a proxy in which the connection originates from.

**Seamless Session Restoration**

The attacker is able to disconnect the victim and take over his browsing session at any time. Any cookies or local data will still be retained on the attacker’s machine, making it impossible for the legitimate website to detect the session hijack. The attacker will still be logged into the website after the hijack regardless of any form of authentication, be it password or 2FA as long as the victim has logged in once.

**Drawbacks of this attack**

**Spoofed Web Server is Not Secure**

The connection between the victim and the attacker is in plain HTTP. This may raise some suspicion as web browsers may raise a warning when browsing any HTTP site. However, it may be possible to secure the connection between the victim and the attacker if the attacker has a valid SSL certificate for the web server they are hosting. Once this is done, there will be 2 separate HTTPS connections - one between the victim and the attacker; one between the attacker and the legitimate website

**Setup is manual and time-consuming**

The setup of this attack can be tedious since the attacker’s webpage title and favicon has to be changed manually every time they intend to mimic a different website. The DNS spoof script also has to be updated to spoof queries to the new website. While difficult to detect, this attack will only work for 1 user at a time - making it only useful for spear phishing rather than mass phishing.