Lab 5: SSL Strip Attack

Introduction:

This report will explain, in detail, the steps to a man in the middle SSL Strip Attack using an ARP spoof.

The steps to the attack are as follows: The victim does not know the website runs over https or does not care about this detail. The victim types www.website.com and the browser default to http. The attacker intercepts this traffic and forces the victim to do an http interaction with the attacker. The attacker poses as the victim and does the https transaction with the server on the victim's behalf: yet does an http transaction with the victim's browser, therefore a "man in the middle attack." The browser wont ask for the certificate since it is not aware that the page you are asking for is an https protected page. This attack proceeds with no certificate warning.

Preparation Steps:

1. **Creating two VMs in Oracle VM Virtual Box Manager/Selecting a NAT Network:** First I began by cloning my current VM and naming it "Kali 2." (I later added the attribute "[ATTACKER]" to the Kali2 name for better clarification) I then created a NatNetwork in the virtual box manager settings. Finally I connected both VMs to a NAT network. See **Figure 1.1-1.2** below.

Figure 1.1

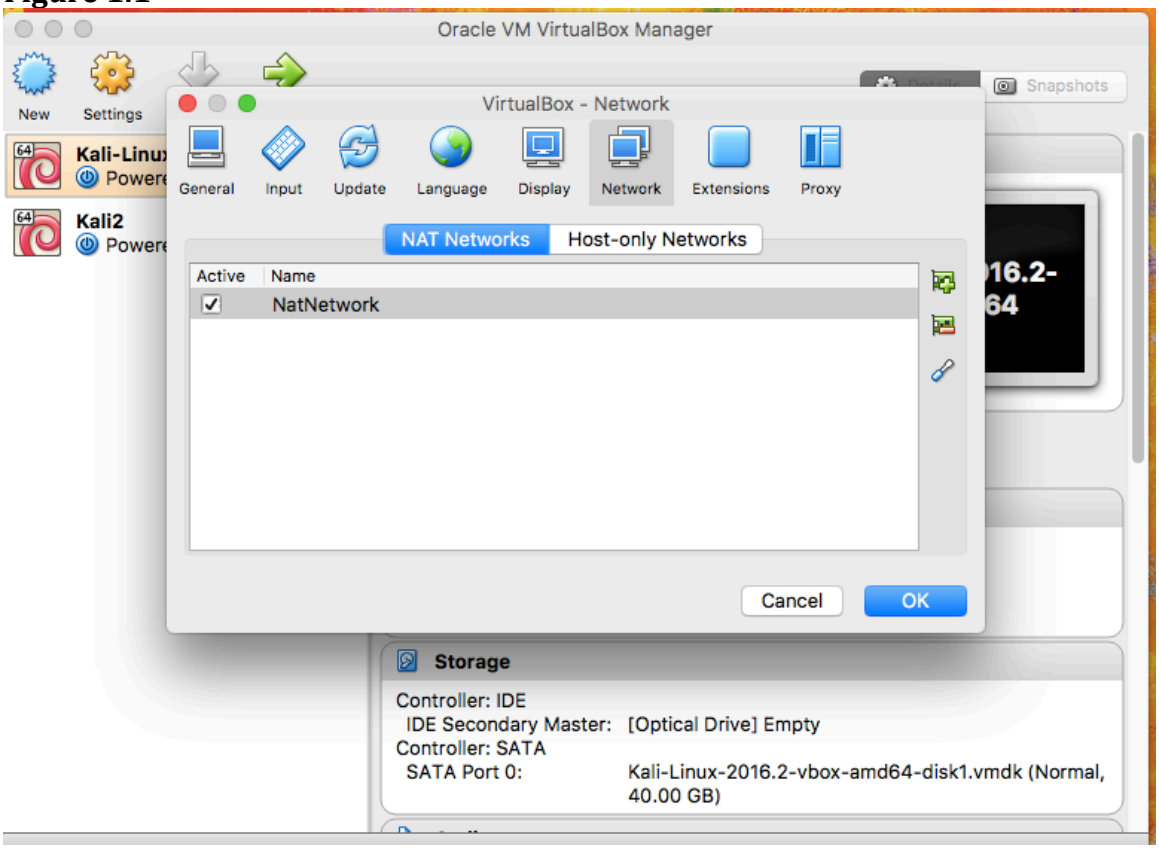
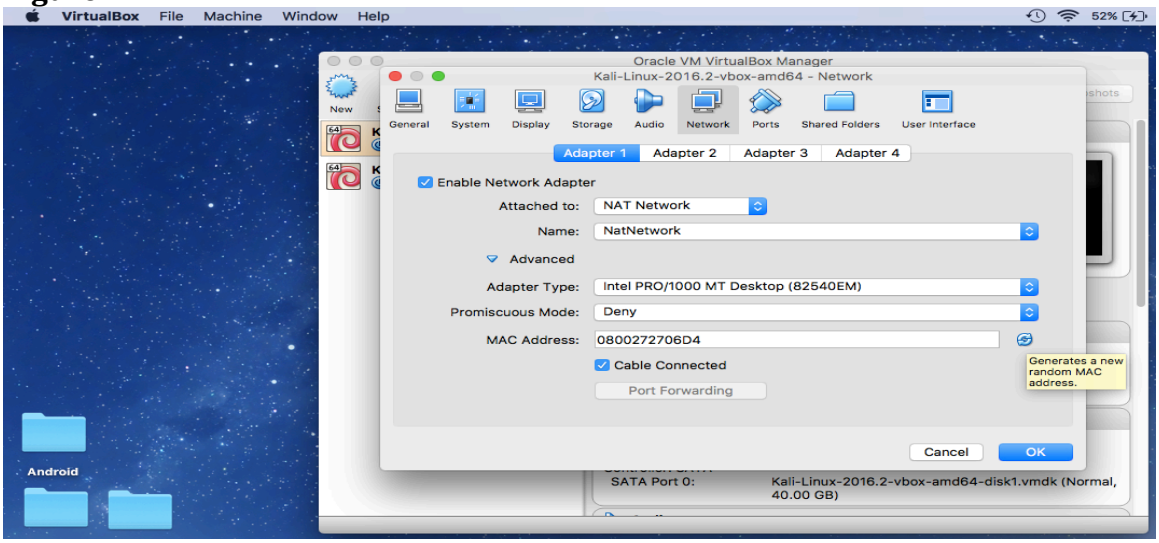


Figure 1.2



2. **Verifying VM Communication:** The last preparation step is making sure the VMs can communicate with one another and that they are both connected to the internet. First I identified the IPs of both VMs, see **Figure 2.1** below. Using the *ip addr show* with out the interface name, is how I identify the network interface name. After identifying the IPs I then verified that the machines could communicate with each other by using the ping command. See **Figure 2.2** below.

Figure 2.1

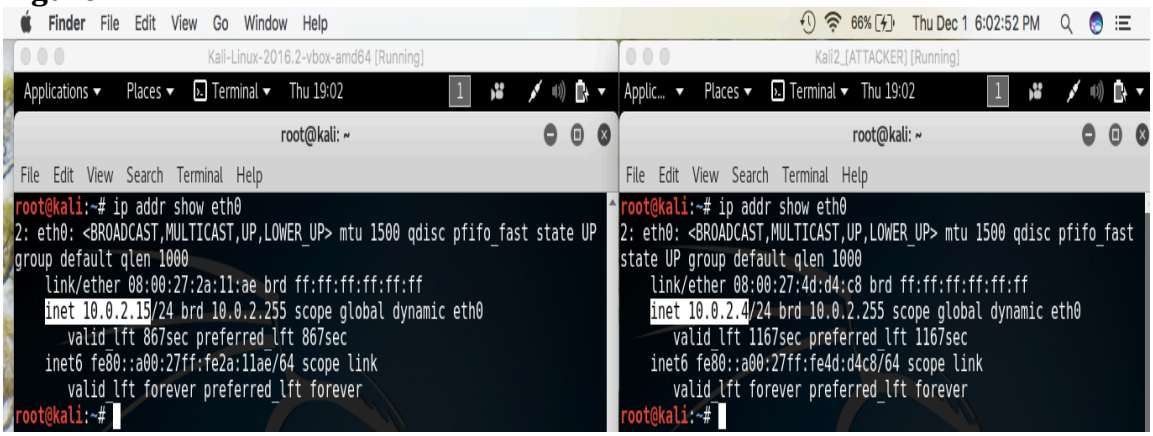
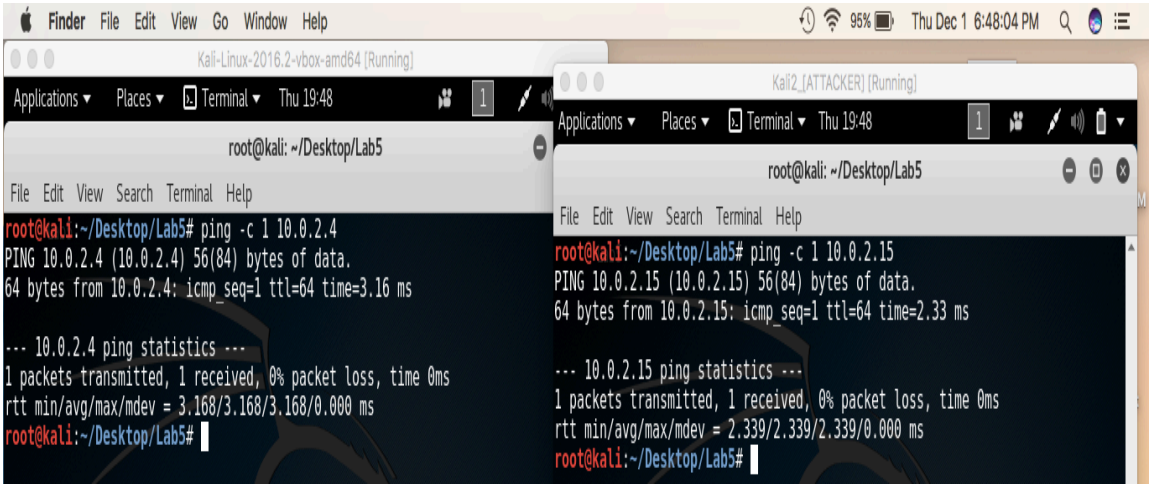


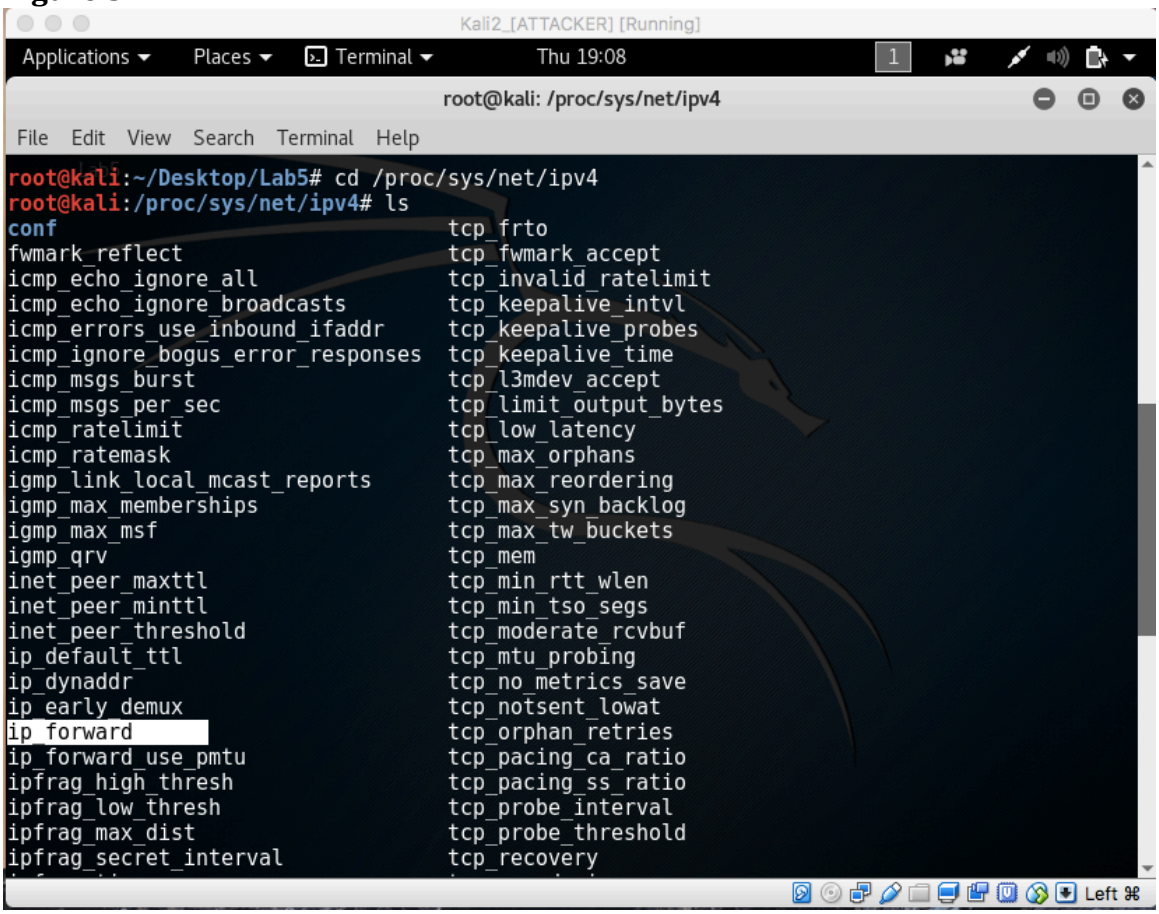
Figure 2.2



Attacker VM Steps:

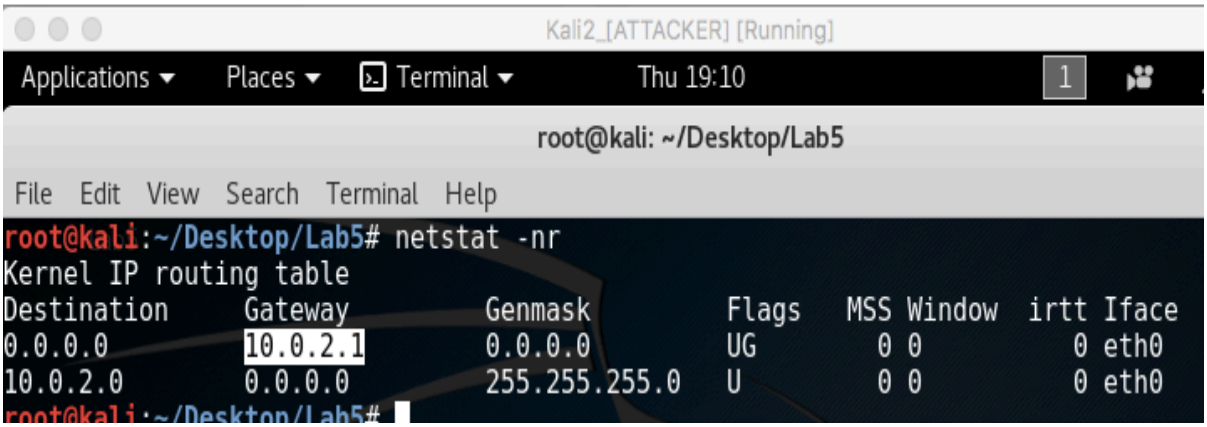
3. **Forwarding Packets:** The first step to preparing the attacker VM is configuring Kali to forward incoming packets. I also verified that the correct file was created in the desired location using the *echo "1" > /proc/sys/net/ipv4/ip_forward* command. **See Figure 3.1** below.

Figure 3.1



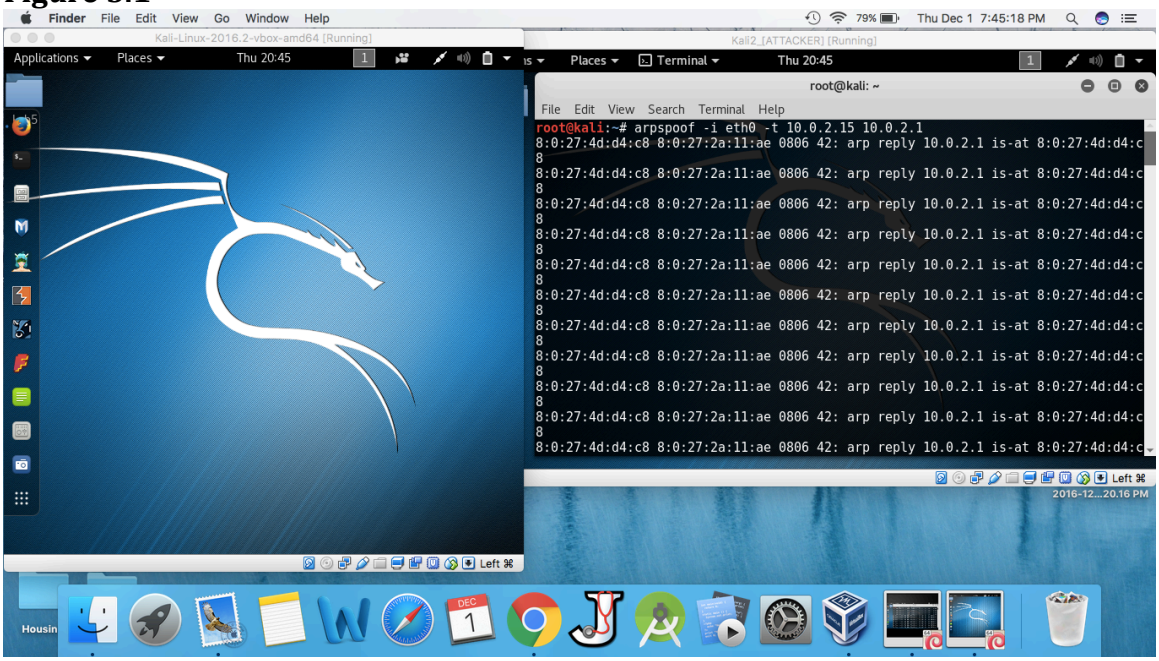
4. **Gateway Address:** The next step is using the *netstat* command and identifying the network gateway address. This will be the same for both VMs. See **Figure 4.1** below.

Figure 4.1



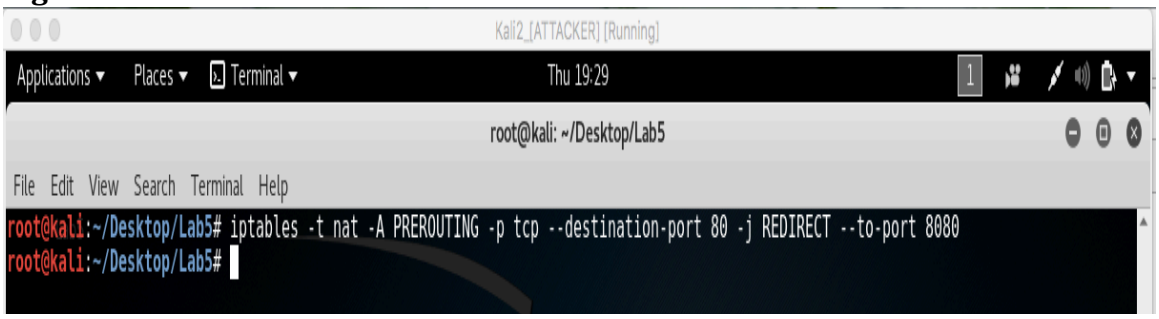
5. **ARPSpoof:** The first step to starting the attack is redirecting the traffic meant for the gateway using the arpspoof tool. See **Figure 5.1** below. After running the arpspoof, I left this terminal window open and opened a new terminal window to continue.

Figure 5.1



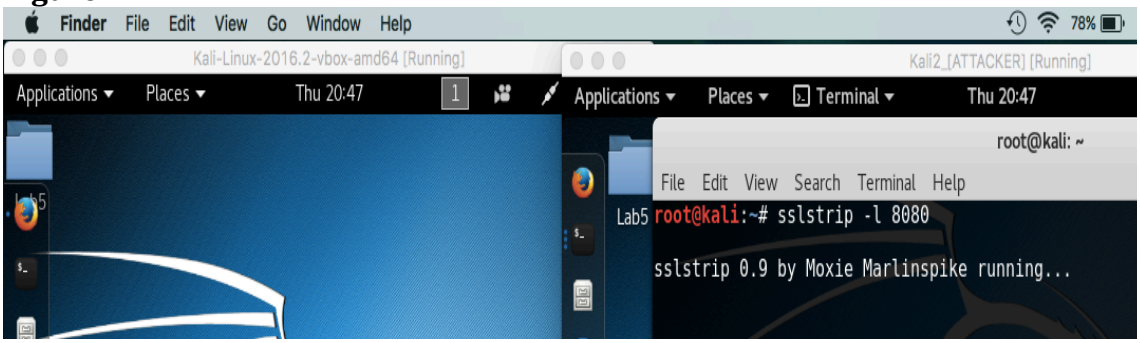
6. **Firewall Rules:** In the new terminal window I set up the firewall rules on the attack machine to redirect traffic received on part 80 to port 8080. See **Figure 6.1** below.

Figure 6.1



7. **Run SSLStrip:** Still working in the new terminal window along with the arpspoof window still running, I began the SSLstrip attack. See **Figure 7.1** below.

Figure 7.1



8. **SSLStrip.log:** I left the SSLStrip and the arpspoof terminal window's open and opened a new terminal window. Now in this new terminal window I viewed the log file produced by the SSLStrip attack. At this moment I switched to the Victim VM and attempted navigating to securtiyisfun.com/CS4331. I noticed that as the victim the attacker only successfully completes the attack if the victim does not enter https into the search bar. Although this only works if the victim begins the domain name with http://www… or just www. If the victim starts the domain name with "securityisfun" with no "www," the browser will default the settings to https and the attack will be unsuccessful. I also noticed when the victim does not use the "http://" the attacker's strip attack throws an "unhandled" error and the website does not direct the victim to the correct page. With the error the attack is still successful. When the victim uses the "http://" the attack and the website behave as expected. **See Figure 8.1-8.5**

Figure 8.1: Shows the victim navigating to the website with out using http://

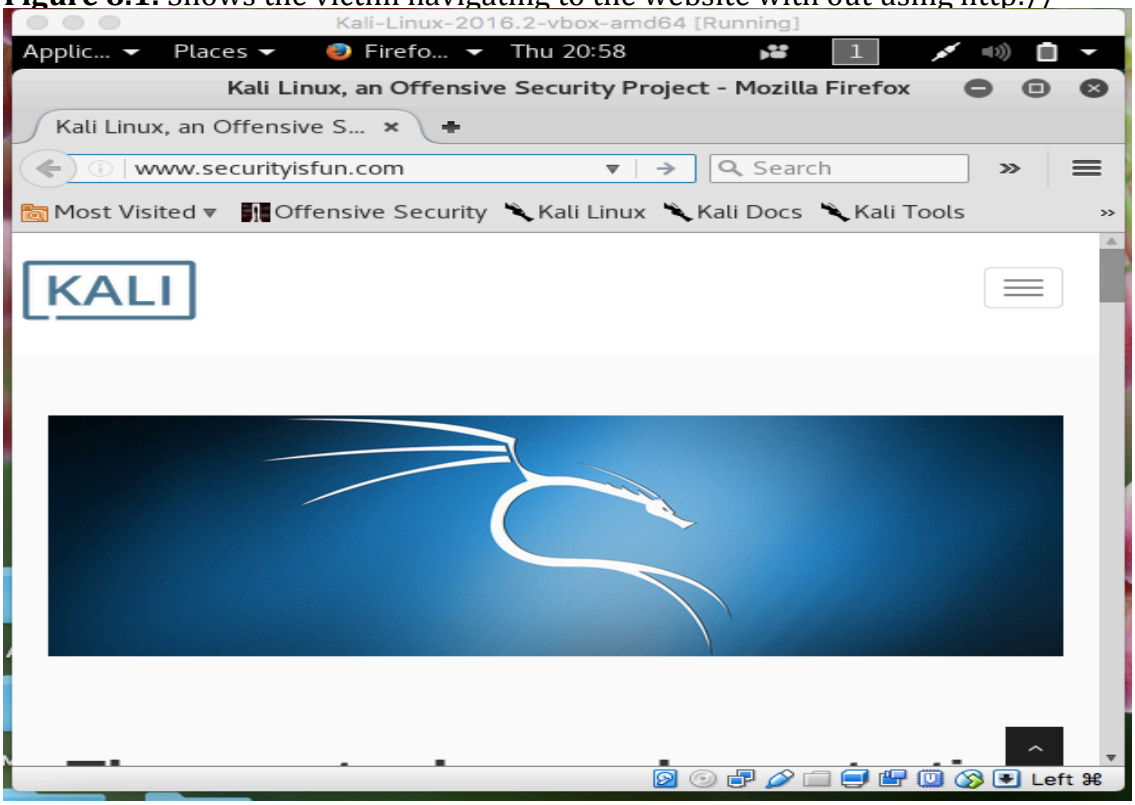


Figure 8.2: Shows the victim navigating to the website using http://

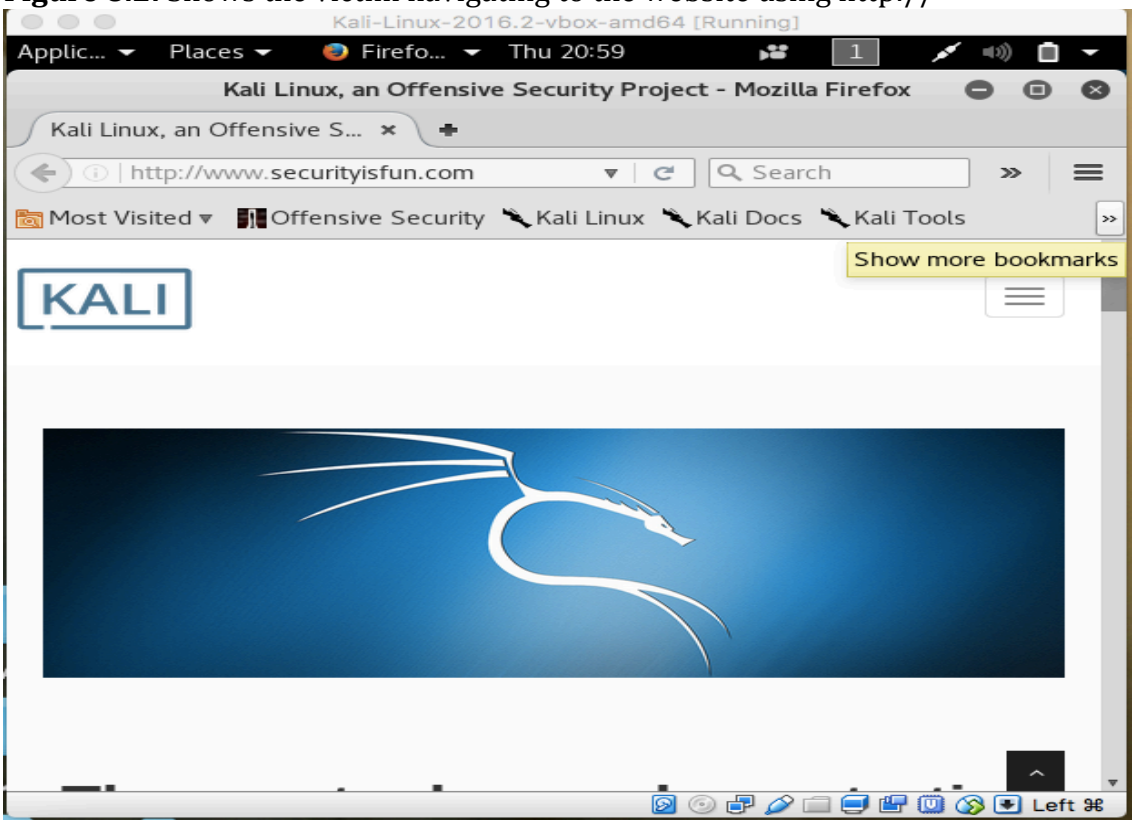


Figure 8.3: Shows the victim entering their credentials

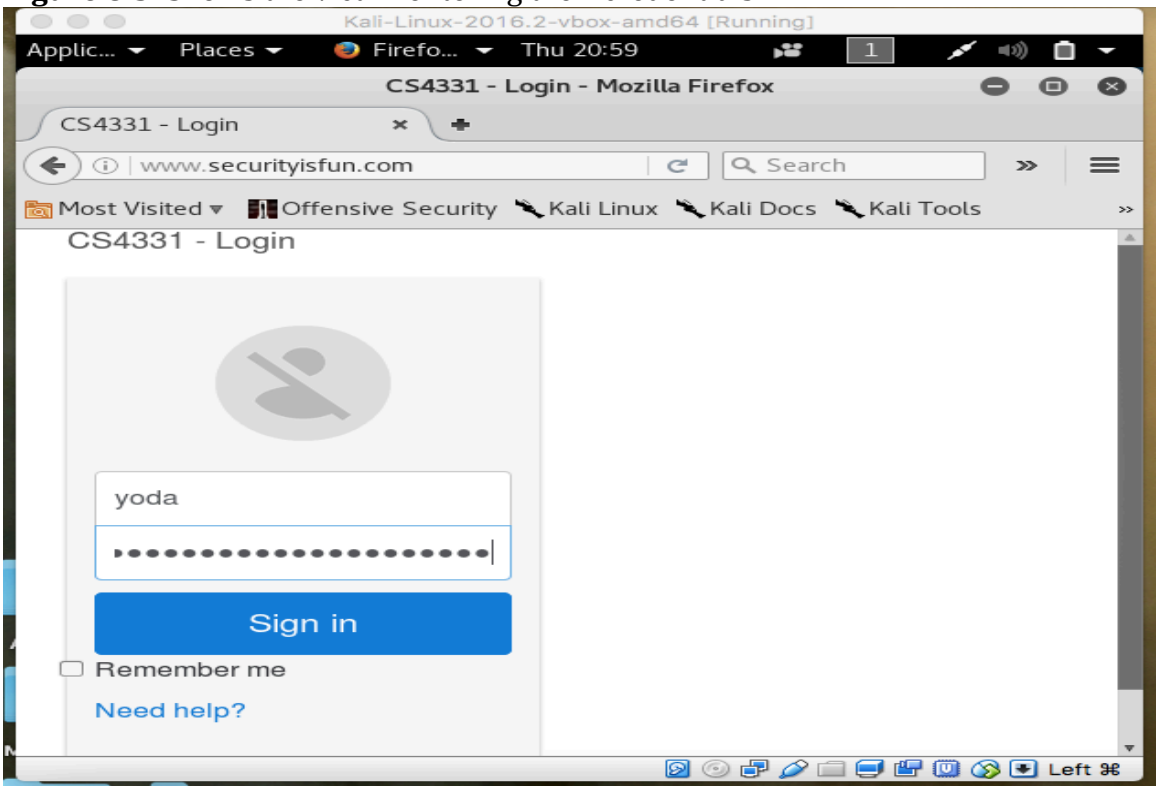


Figure 8.4: Shows the SSLstrip.log (when victim does not use http://) after credentials were submitted to the website

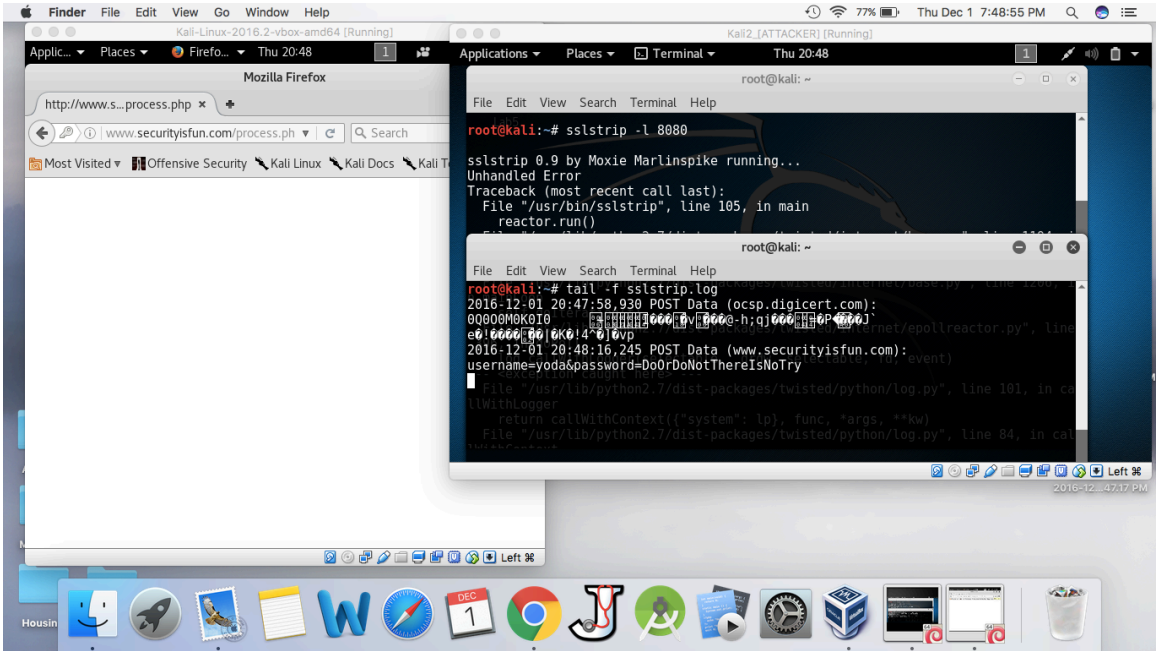
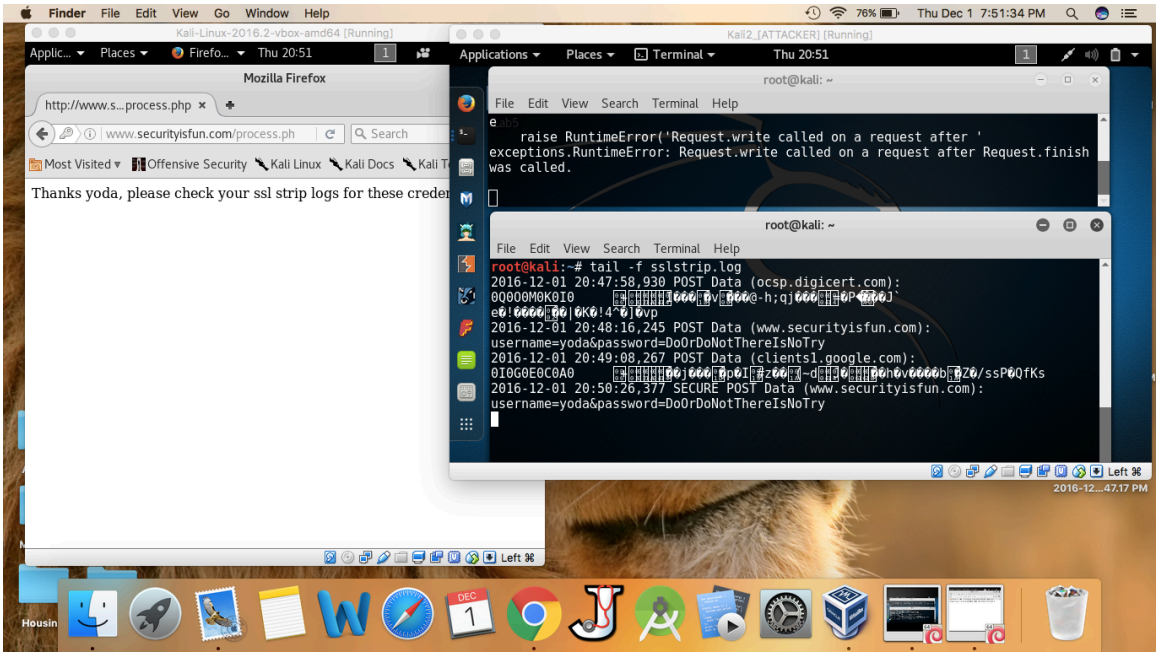


Figure 8.5: Shows the SSLstrip.log (when victim uses http://) after credentials were submitted to the website



Conclusion:
In conclusion the attack is successful because the attacker is posing as the victim and does the https transaction with the server. The browser proceeds without any certificate warnings. The attacker is able to intercept any data the victim enters into the website because of the "man in the middle" interaction with the server. This attack is successful when the victim does not specify the https in the search bar. The attack is unsuccessful when the victim either enters the https or lets the browser determine the presets by beginning the domain name with "securityisfun" leaving off the "www." (The browser will add the https:// automatically) This is due to the HTTP Strict Transport Security, which can protect the victim against this attack. This protocol allows the server to declare to the browser "I must be accessed via HTTPS for the next x amount of years." If this protocol is implemented, even if the victim types www.webiste.com the browser will know that it has to initiate the request as an https.

But the protocol has its weakness. If the server is accessed for the first time, then it has not had a chance to tell the browser about the https protocol. This means the attack would be successful during this instance. This is more likely to occur on obscure, rarely accessed websites.

Also when the period of activity of the HSTS policy expires, this may also be a window of opportunity for the adversary.

There are also counter measures to this weakness. The Strict Transport Security preloaded list is preloaded on browsers. Allowing the browser to contain a list of known sites, which use https. Which covers the problem of first accessing the server. But does not scale to cover all websites in the world.

The best way to safe guard against this attack is to type https:// straight into the browser all the time. There is still a non-zero probability of falling to the attack, but as close to zero as you can get.