Interception Analysis

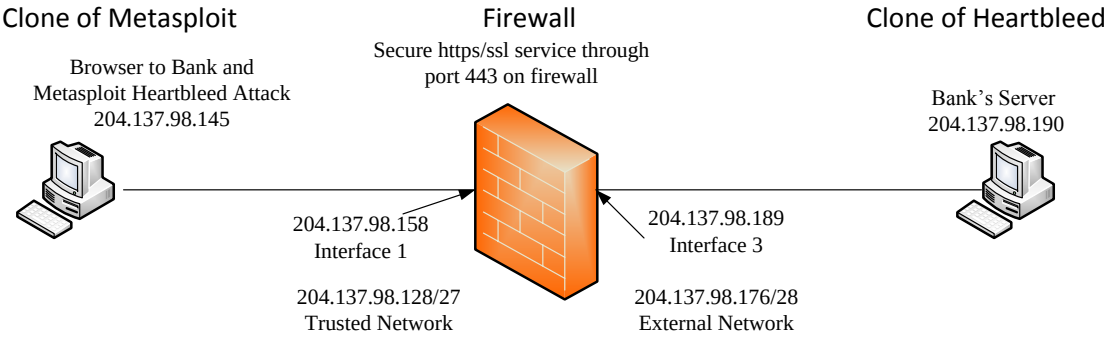
SSL Interception Attacks – Heartbleed

It is assumed that you are familiar with this attack which affects SSL versions 1.0.1a, b, c, d, e, f. For this experiment you will need an HTTPS packet filter on the firewall thus opening the standard 443 port. We will run this from two separate virtual machines - on the trusted side it is called "Heartbleed Client" and on the external/untrusted side is the bank virtual machine called "Rays Heartbleed Bank".

Heartbleed SSL Vulnerability Machine Configuration

You are already using 204.137.98.145 as the admin machine (which you have used to configure the firewall https rule) and 204.137.98.190 as the untrusted external machine. The respective client and server machines to run the Heartbleed experiments are both run on Centos and are preconfigured to 204.137.98.140/27 (client) and 204.137.98.185/28 (server). Make sure that you can ping end-to-end.

Choose different IP addresses but in the correct respective subnets.



Once Firewall configured with 204.137.98.145 use the following VMs:

Heartbleed Client IP address (**204.137.98.140/27**) Rays Heartbleed Bank IP address **(204.137.98.185/28**)

Check that the respective CentOS VMs have these IP addresses configured as 204.137.98.140 and 204.137.98.185 respectively. If not them IP address need to be configured.

Client in trusted network: Centos Metasploit (runs Metasploit)

Bank Server on external network: Centos Heartbleed (runs Apache server)

In all cases when starting these Centos machines say "I moved it".

Open the Bank's Centos machines and log in as *root/P#ssw0rd*

Should you need to provide an IP addresses then enter: ifconfig ethx <IP address>/mask (/27 for trusted and /28 for external networks respectively.

On this Bank's machine create a password file

htpasswd –c /var/www/passwd/passwords <my username in here>

Note space between htpasswd and –c and between –c and /var

(There is an additional switch: –s which will require a SHA hash rather than MD5 (default). It follows the –c switch.)

You will then be asked (twice) for a password which will then be the username and password that will be required to enter the banks website.

Create at least two or three users and passwords but carefully note them down to avoid confusion when you will subsequently login to your bank using these credentials.

This file remains during shutdown so you may well encounter username/password from earlier experiments with this vulnerability.

Open the client Centos Metasploit machine and select *other* and log in as *root/P#ssw0rd*

Provide your IP addresses only if not preconfigured: ifconfig ethx <IP address>/mask (/27 in this case)

You must be able to ping successfully between client and server before proceeding.

On your client (left) machine from the root directory enter *msfconsole*

Enter *search heartbleed*

msf> *use auxiliary/scanner/ssl/openssl_heartbleed*

show options and enter as follows (case sensitive)

set RHOSTS <IP Address of Bank>

set RPORT 443 (not 80 !!!!!)

set VERBOSE true

exploit

From the client Centos Metasploit machine use Firefox and open a browser on the bank server which runs (Apache). *Remember to use https and not http!!*

On the browser turn off "Work Offline" under File if necessary.

Make sure that old certificates have been removed: Firefox > Edit Preferences > Advanced > View Certificates > Servers and remove any of the form "Unknown 204.137.98……

Step through certificate risk process.

To see the certificate on subsequent runs then there is an option at right or left hand end of URL bar. Click on the padlock and the certificate will be displayed.

A user name and password are now requested – the ones you created for your bank.

Openssl returns the Heartbeat response with leakage data. Then select the text *following* "Basic" up to and *including* = for example:

Basic **Y2FuYmVycmE6c3lkbmV5Iy=**

Now do a base64 decode as follows:

Either enter base64 –d <CR> then paste string from above, then a <CR> and then CTL D.

or enter echo <string from above> | base64 –d <CR>

The exploit can be "run" after the web page is opened but it may be necessary to rerun a number of times as it may well be remembering earlier usernames and passwords. You can recognise base64 encoded strings and find a number of usernames and passwords!!

You can also use this web site to obtain a similar result.

See how many IP/password combinations you can leak from this bank.

Checks if this not working:
Have you assigned an IP address in the subnet range?
Are the subnet masks correct – different for each side of the firewall?
Are you using HTTPS and not just HTTP?

MITM (Man In The Middle) Attack using Ettercap (Wired or Wireless)

The following three experiments demonstrate the use of ARP (Address Resolution Protocol) spoofing and the creation of MITM packet redirection. This is a valuable workshop in preparation for the VoIP Forensics workshop and the processing of interception analysis.

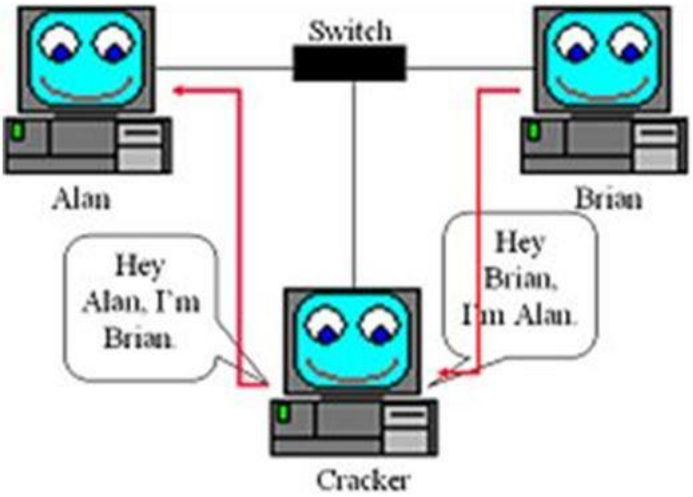
Notes about ARP Spoofing and ARP Poisoning

What Is ARP Spoofing?

ARP spoofing also known as ARP poisoning or ARP attack is a technique in which a host in a LAN can "poison" the ARP table of another host causing it to send packets to the wrong destination. The attacker can modify the traffic in the network such a way that it will redirect all traffic to go through it. ARP Spoofing will allow an attacker to sniff data frames

How ARP Spoofing Works?

The image helps to understand how ARP spoofing/ARP poisoning works. Basically, the Cracker is telling Alan's box that he has the IP that corresponds to Brian's box and vice-versa.



By doing this the Cracker receives all network traffic going between Alan and Brian. Once you have ARP spoofed your way between two machines you can sniff the connection with sniffers like (Wireshark, Ettercap etc.) By ARP spoofing between a machine and the LANs gateway you can see all the traffic that it is sending out to the Internet.

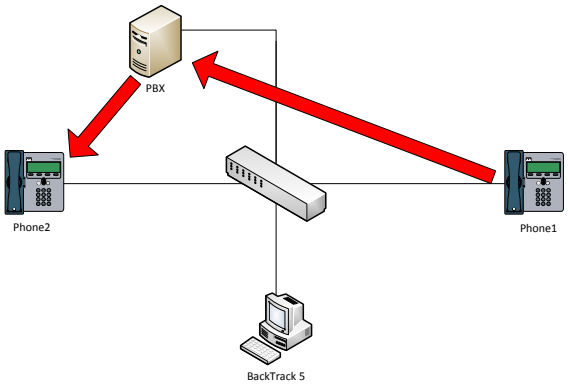
Sample of Tools Used For ARP Spoofing

- Ettercap
- Dsniff
- Cain and Able
- Arpspoof

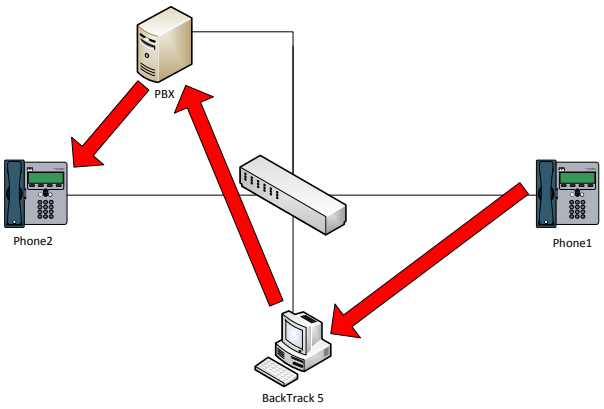
In the VoIP Forensics workshop we operate a similar MITM scheme as shown below:

Because we are operating on a switched network we need to place ourselves within the flow of data. To do this we are going to perform a Man In The Middle (MITM) attack by poisoning the ARP table of using Ettercap out of Backtrack. This changes the data flow (as seen below) allowing us to capture traffic between the client and server – including all ID/password operations.

Normal Data Flow



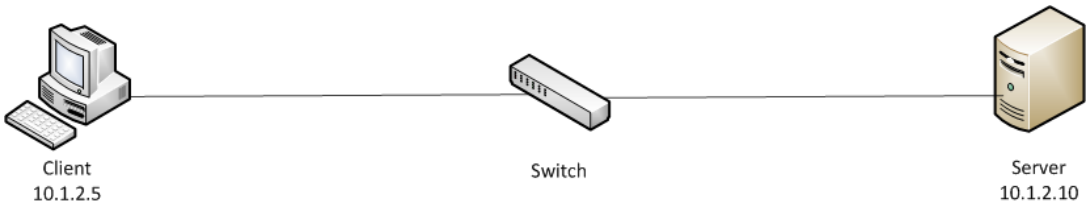
MITM Data Flow



Part 1:

Set up the following network making sure that all VMs are restarted to avoid existing routing and ARP tables.

[Note: 1. Use BT5 Android New and whichever interface is active. This can be checked using ifconfig -a
2. Use /24 mask as if /8 is used then Ettercap will take forever to scan for 255x255x255 hosts]



Simple switch with no MITM configuration

The bank client on the left runs a simple Internet Explorer web browser and has an IP address of 10.1.2.5/24. You can run this either from the base machine or any VM.

The server (bank) on the right runs on 10.1.2.10/24 will be accessed later with the login: http://10.1.2.10/SSL/login.asp

Make sure that you can ping between these two devices through the interconnecting switch. If ping does not work check (i) firewall is off on both machines and (ii) VM settings are "bridged" and not "NAT" nor "Host" on both machines.

It is important to clear the ARP table on the client. This can be done either *with*

arp <IP address> -d for example *arp 10.1.2.10 –d*

or by physically disconnecting the wired/wireless connection device.

```
root@bt:~# ifconfig eth6 10.1.2.5/24
root@bt:~# arp 10.1.2.10 -d
root@bt:~#
```

Before *arp* table cleared ….

```
root@bt: ~
File  Edit  View  Terminal  Help
root@bt:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.1.2.10                ether   00:0c:29:31:a2:b0   C                     eth6
root@bt:~#
```

After *arp* table cleared ….

```
root@bt: ~
File  Edit  View  Terminal  Help
root@bt:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.1.2.10                        (incomplete)                              eth6
root@bt:~#
```

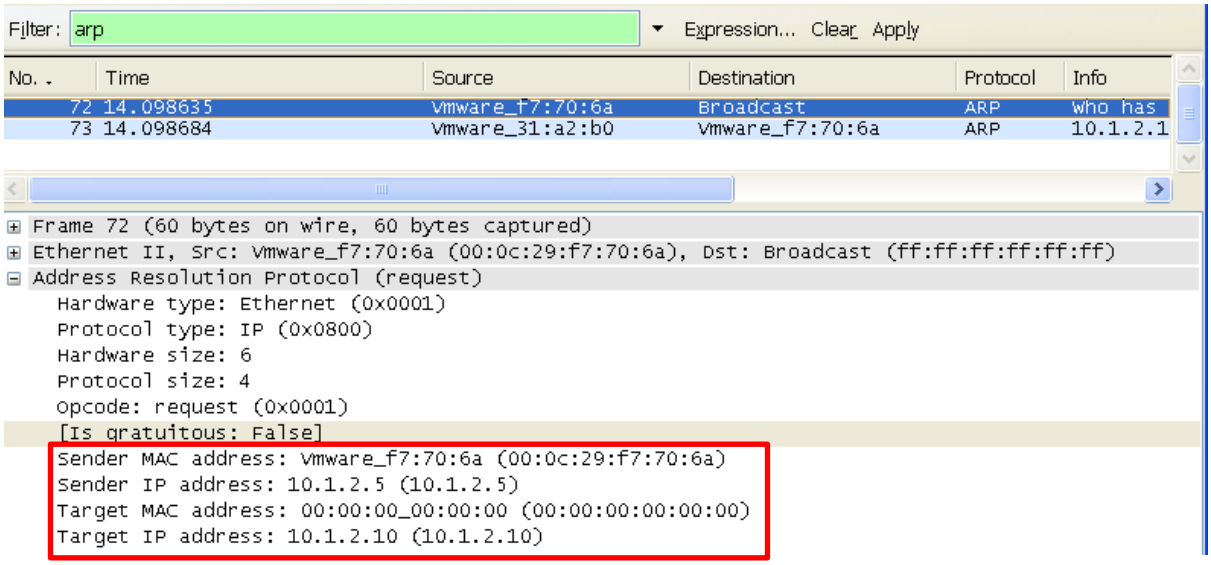
Start Wireshark on the client machine using the appropriate Ethernet interface (likely to be *eth6*) and apply an *arp* filer.

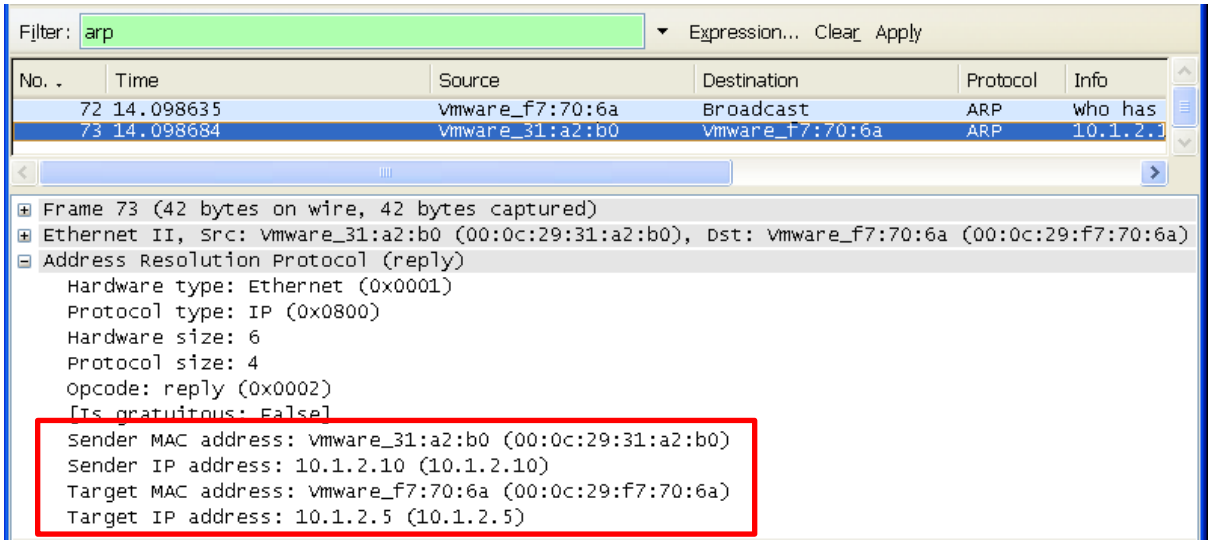
Now open a web page onto the server (bank) with the URL of: http://10.1.2.10/SSL/login.asp

Because there is no MITM in operation the only thing that you should see when you enter the login is an ARP request of the form:

Who has 10.1.2.10? Tell 10.1.2.5

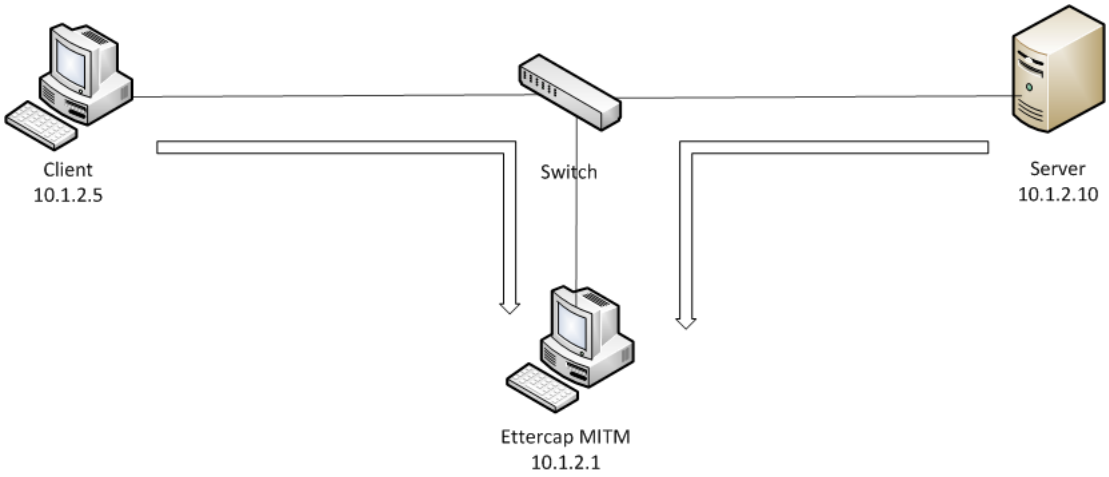
You can see the MAC addresses where appropriate. It is also a good idea to apply an ARP filter to this Wireshark trace.





Part 2:

Now set up the following diagram:



Simple switch with MITM configuration using Ettercap from Backtrack5 or Backtrack6

Close the client browser and empty all stored history using the Options menu. Flush the ARP table or disconnect/reconnect at the Ethernet interface or use *arp <IP address> -d* if you are logged in with adequate privileges.

Ensure that your VM machine is "Bridged" under VM > Settings

- Start Ettercap as follows:

  Backtrack5: Backtrack > Privilege Escalation > Sniffers > Network Sniffers > Ettercap-gtk

  Kali/Backtrack6: Applications > Kali Linux > Sniffing > Network Sniffers > Ettercap-graphical
- Select Sniff > Unified sniffing and select your current network interface (e.g. eth4) and click OK.
- Select Hosts > Scan for hosts > Select Hosts > Host list
- Add one of your IP addresses (e.g. client machine) to Target 1 and the other (e.g. server machine) to Target 2.
- Select MITM > ARP poisoning. Select "*sniff remote connection*".

The MITM runs out of Backtrack (initially without Ettercap in operation) and has an IP address of 10.1.2.1/24.

You can start Wireshark (either on the client machine or on the interception machine) and see some of the promiscuous entries. However we need our computer to become a router to forward packets. Thus make sure packet forwarding is turned on, otherwise our machine will drop all traffic between the hosts we are trying to sniff. Thus enter:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

To check that this switch has been turned on enter

```
cat /proc/sys/net/ipv4/ip_forward
```

and the response should be 1.

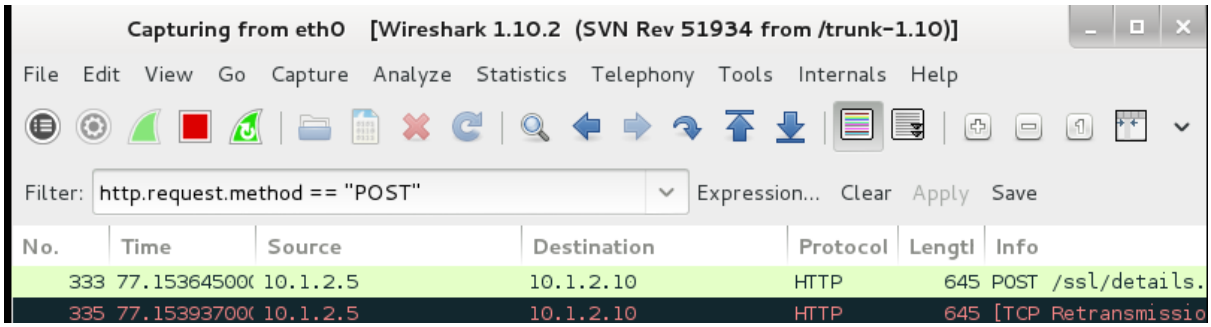
You can do a ping test to ensure that everything is working, but for the final run that you wish to capture, clear the client's ARP table as stated above by *arp <IP address> -d* and/or physical disconnection.

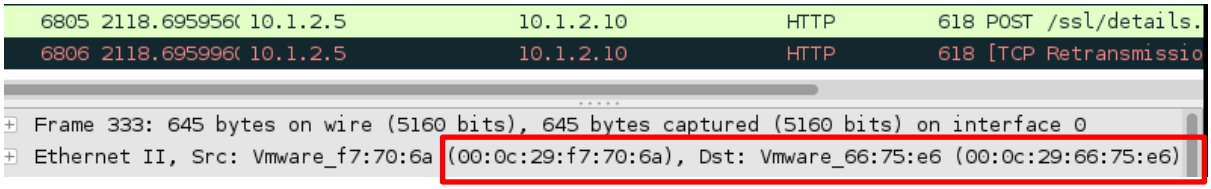
Now using the MITM machine start Wireshark and login from your client machine as above (*http://10.1.2.10/SSL/login.asp*) capturing the *http* traffic with Wireshark. Scan down these *http* segments to find the one carrying the POST command (use a Wireshark filter) and in it you will see the ID and password in clear text.

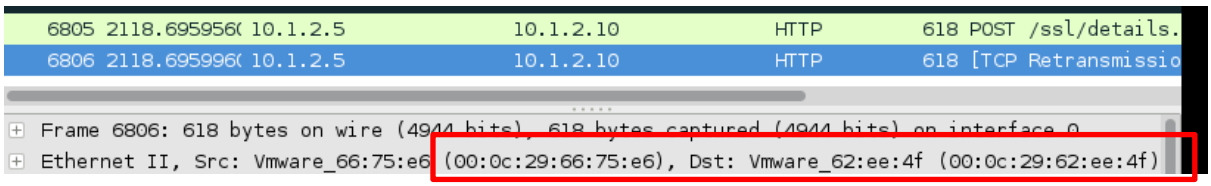
You can use “Follow http stream” pull down menu in Wireshark as well. Alternatively use a Wireshark filter for: http.request.method == POST.

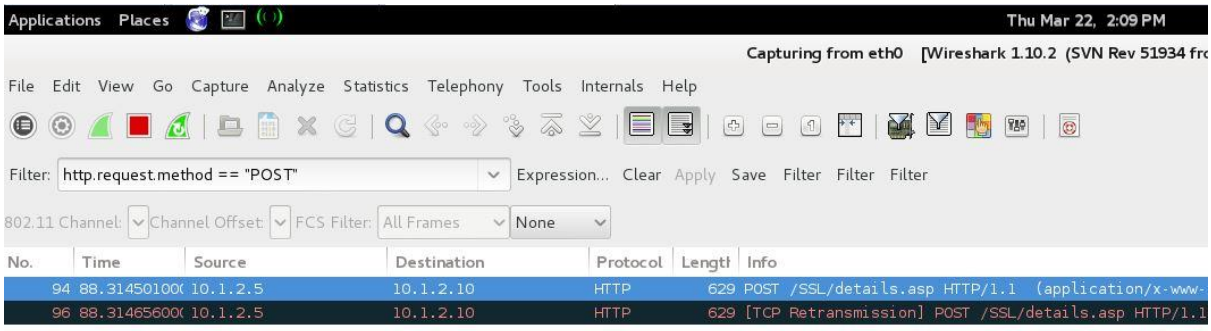
POST/SSL/ details appears near end, i.e. [TCP Out of Order] POST.

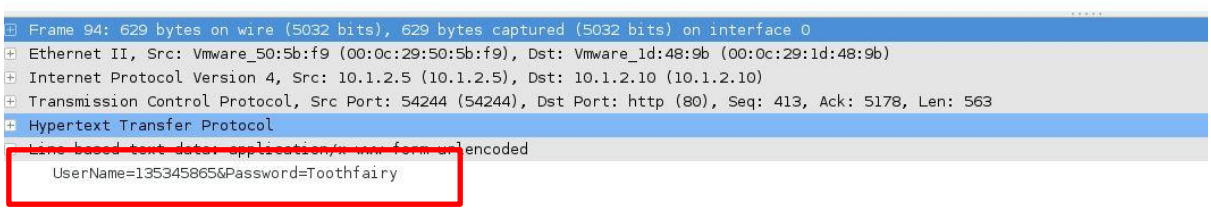
Why do you think that you get two “POST” entries in this Wireshark run on the MITM machine whereas it only appears once on Wireshark at the client or server end? If you are not sure then look at the MAC addresses in this trace and relate to your Ettercap configuration.











Part 3:

Repeat Part 2 using *https* instead of *http*. You will of course not be able to see the ID and password. Thus to proceed further we need to use the *Mallory Interception Engine* (from Blackhat Conference, USA) which is the subject of the next lab to follow (SSL Interception Workshop).

Do you think that this might solve the problem if ID/Password security?

Notes:
Make sure all VMs are bridged.
Make sure no spurious gateway addresses in configurations.
In Wireshark make sure promiscuous mode is turned off.
Make sure Virtualbox is disabled
Make sure that all IP address configurations have a netmask of /24

Interception Analysis: SSL Interception Workshop

(Part 3 of Man-In-The-Middle Attack)

Background

The following workshop will use both Linux (Ubuntu) and Windows Virtual Machines. The client can be a desktop machine although in practice it is more likely to be a mobile and we will run on Android mobiles here and the Man-In-The-Middle interception engine will run on Linux Ubuntu.

There is a widespread belief that a mobile or PC client has a secure encrypted direct connection to a bank for the purposes of Internet Banking. This is almost never the case and the "secure" connection will pass through a number of machines (often proxy servers) along the way. There is also a belief that X.509 digital certificates authenticate the parties. This workshop will demonstrate that secure and encrypted sessions can be intercepted and further, show that without root certificate verification or the use of DNSSEC, the use of X.509 certificates can be manipulated in order to intercept, decrypt, extract ID and password, re-encrypt and continue to send the encrypted data.

Introduction to Mallory

The following three diagrams show possible setups for SSL interception, viz. (i) an internal bank server running on the same hardware platform (VM) as the Mallory interception engine and (ii) an external bank server running on a separate and dedicated machine with firewalls protecting both the client and the server and (iii) a multiple/parallel version of (ii) running a DHCP server. The second and third of these two is more realistic and the third is the one that we will configure in this workshop. The client can be a PC (laptop or desktop) or a mobile (Android). In both cases the encrypted SSL session can run over the connection directly or over a VPN tunnel which itself runs over the direct physical connection. In the case of the Android mobile the connection is wireless and runs over a WPA encrypted session thus we have an encrypted SSL session optionally running over a VPN and finally running over an underlying encrypted wireless connection.

The Mallory VM has been setup with the necessary configuration and we will run with individual mobiles connecting to the clients own bank. However this network has been designed to support up to six parallel interception operations.

On the following page is a diagram showing the external (bank) server with a firewall which one might think could protect against such attacks. However the HTTPS packet filter has to be installed and experimenting with a firewall in drop-in mode (where the same IP address is used on both interfaces) demonstrates that the firewall is of no use in protecting against such attacks. Nevertheless it is an interesting addition to the network configuration and makes for an interesting experiment.

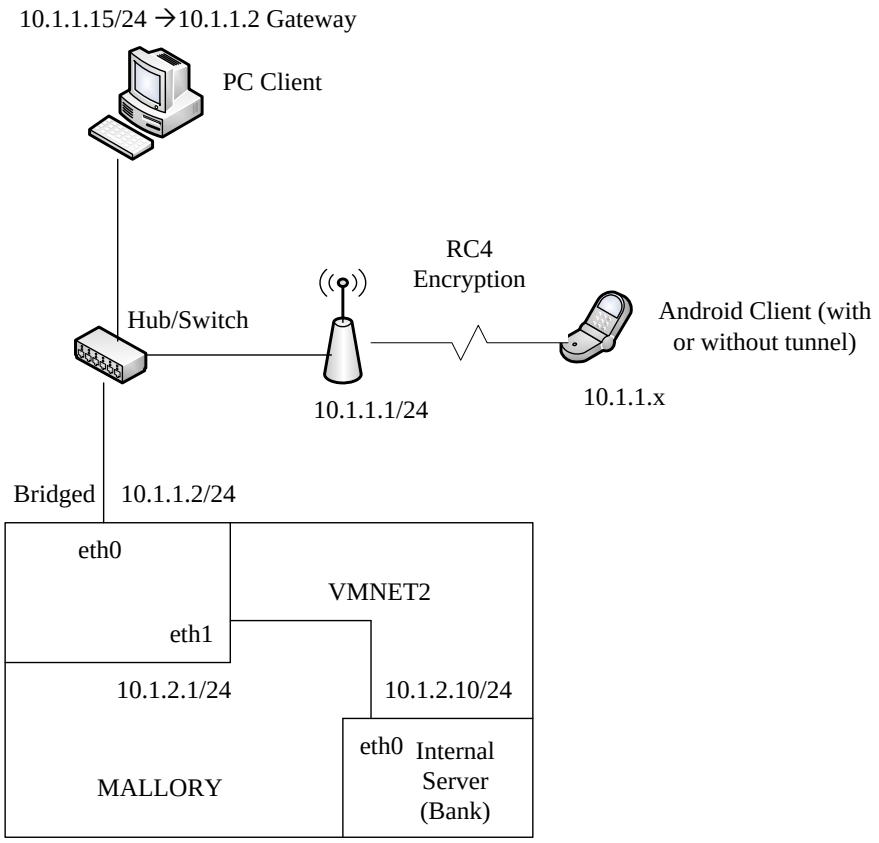


Figure 1: SSL interception using internal server for bank

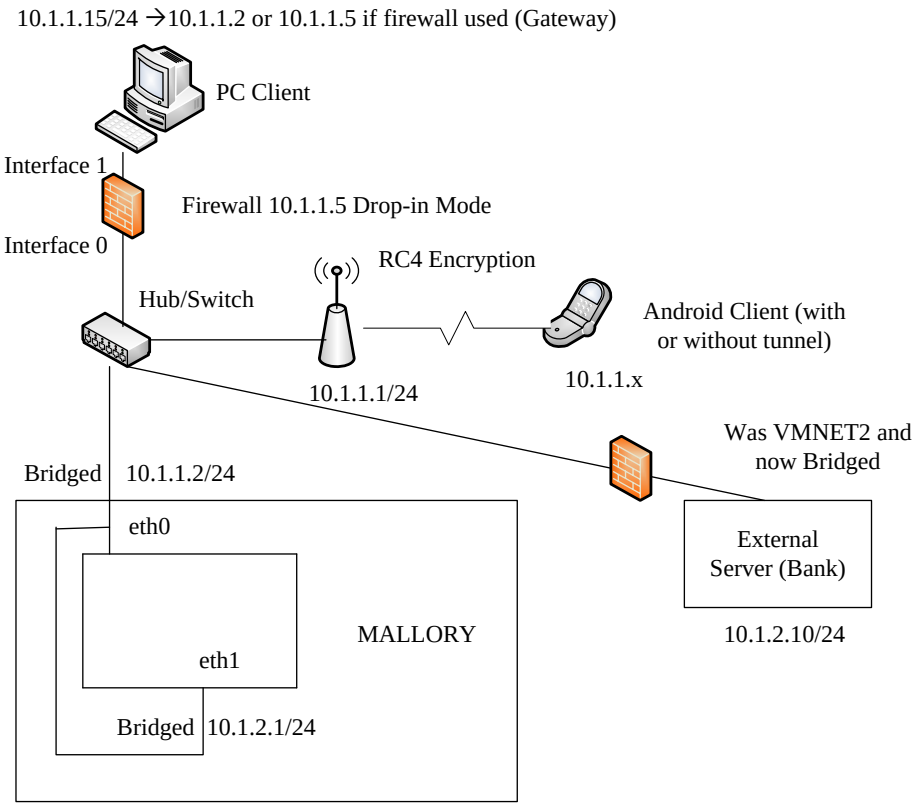


Figure 2: SSL interception using external server for bank (with firewalls)

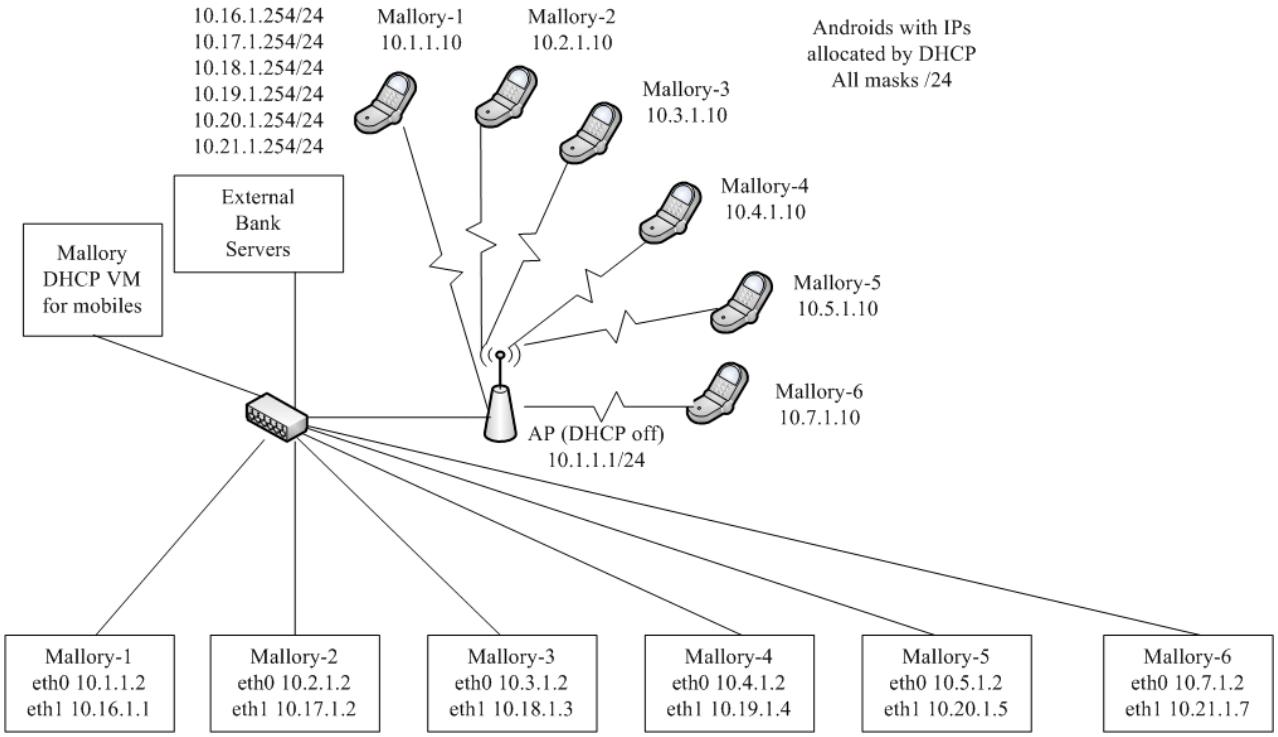


Figure 3: SSL interception using bank servers and running multiple Mallory interception engines

The key components of this lab involve a client (PC or Mobile), an interception engine and the bank server. Because so much Internet banking is now done from mobile we will use Android Mobiles as clients (although exactly the same results occur if we use a PC – desktop or laptop. Figure 3 is an enhanced version of Figure 2 and is essentially the multiuser version in which simultaneous interception attacks can be made to one or more bank servers. Thus the description below using a single access point and multiple bank servers most closely resembles a real life situation of a simultaneous attack on multiple bank servers.

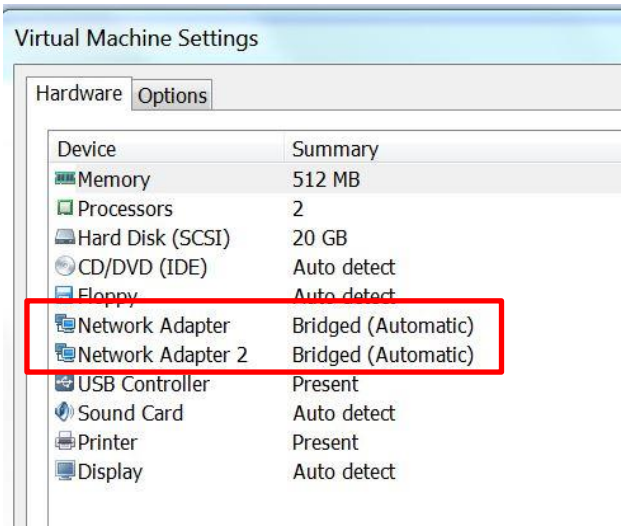
In Figure 3 the DHCP engine provides IP addresses for the mobiles in exactly the same way as occurs in practice. Each Android mobile has wireless access to a single Access Point and then accesses the bank server via the Mallory interception engine.

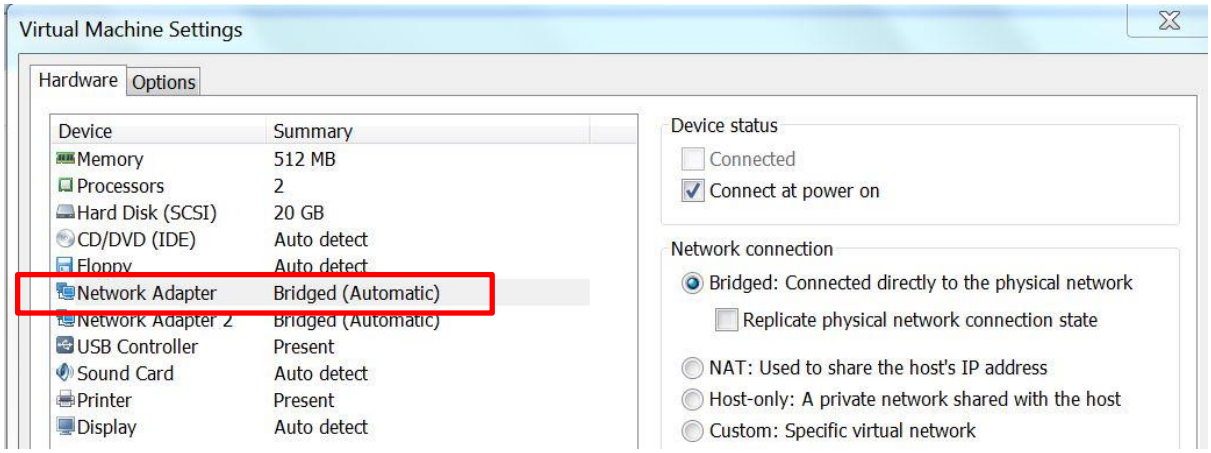
Each Android mobile (or desktop/laptop PC) will need to obtain an IP address in an appropriate subnet range consistent with eth0 on the interception engine. The DHCP server looks after allocating IP addresses for the Android mobiles.

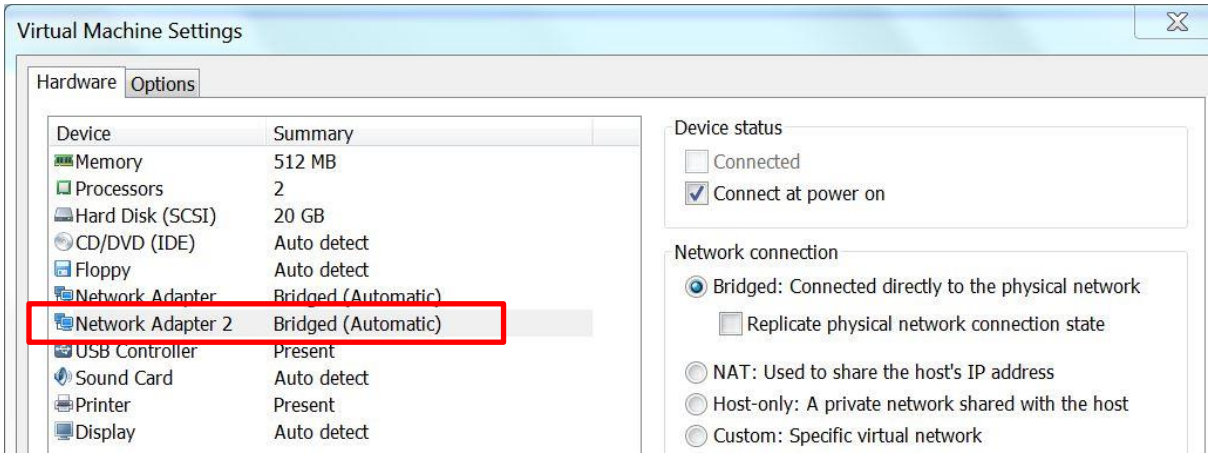
VM Configuration Summary

A summary of the VM configurations is given below:

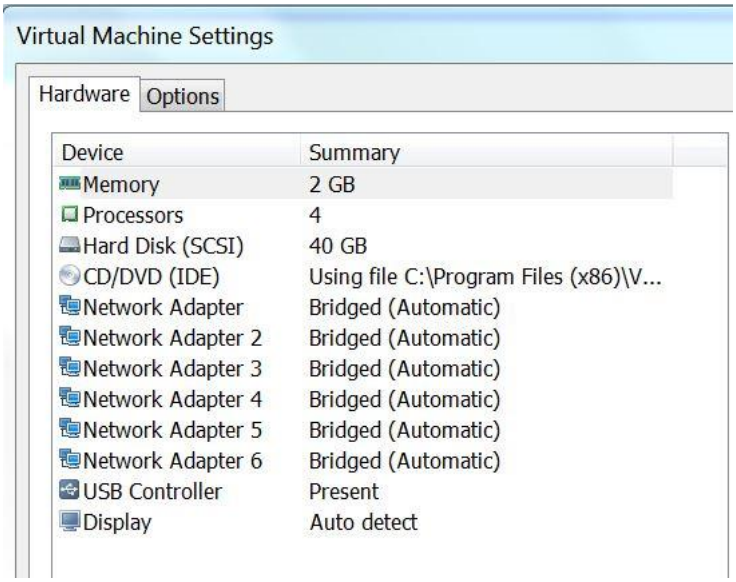
Virtual Machine settings for Mallory interception engine as well as the settings for its two Network Adapters



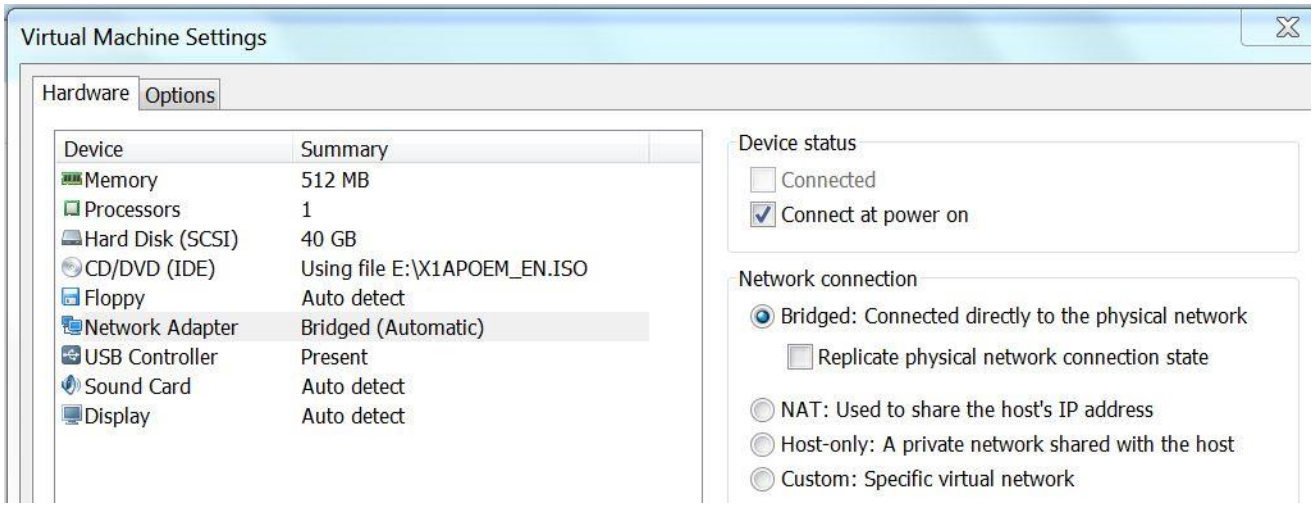




Virtual machine settings for the DHCP engine.



Virtual machine settings for the bank server engine.



For the Mallory Engine under VM > Settings notice that there are two Network Adaptor settings. One is for the bridged network connection 10.x.1.2/24 (eth0) through the Access Point to the corresponding Android while the other is 10.y.1.x/24 (eth1) for connection to the corresponding bank server. See Figure 3.

Start-up Sequence is Critical

1. Start VM and configure e0 and e1 IP addresses as below
2. Start Bank Server and configure IP address as well
3. Start DHCP Server (make sure wireless is off and wired is enabled) and start AP with DHCP disabled
4. Start Android and see if DHCP address appears and then ping Android from Bank Server. Mallory e0 and e1 interfaces must be active for ping to operate here.
5. Only now can the two Mallory python scripts be started

IP Configuration for Mallory MITM Interception Engine

When this VM is moved then select “I copied” as “I moved” relates to Centos and Redhat VMs. “copied” knocks out IP addresses while “moved” maintains originals. The password for logging in to the Mallory engines is *mitm*

Set up the IP addresses on the Mallory engine as follows:

sudo ifconfig eth0 up

sudo ifconfig eth0 10.x.1.2 netmask 255.255.255.0 (10.x.1.2/24)

sudo ifconfig eth1 up

sudo ifconfig eth1 10.y.1.x netmask 255.255.255.0 (10.y.1.x/24)

Set to mask of 255.255.255.0 (/24) especially in the Access Point and do not confuse with /8 otherwise route gets screwed up.

The next stage is to run the scripts to bring the Mallory engine’s server and GUI interface into life. However it is most important that the client is first established.

Turn on the Android and verify that the DHCP server has allocated the correct IP address. Then ping this Android from the Mallory engine. Likewise ping the bank server from the Mallory engine as well to ensure that you have connectivity in both directions. You will of course not be able to ping the Access Point. If you are doing the second part for the Android and stablishing a VPN tunnel then this must be established and “connected” before starting the Mallory Server and GUI. For each part of this experiment it is best to restart the Mallory VM.

Go back to Mallory Intercept Engine

cd /home/mitm/mallory/current/src

1. Now to start the *Mallory Server* enter (you can copy these command lines from the commands file on the desktop:
   cd /home/mitm/mallory/current/src
   sudo python ./mallory.py (sudo password is mitm)

   Note now that the server is running and it should be left in the window and a new window used from here on.

2. In a NEW window start the *GUI interface* and enter:
   cd /home/mitm/mallory/current/src
   sudo python ./launchgui.py (sudo password is mitm)

   To shut down – first CTL C the server and then in other window sudo poweroff

Reiterating the note above. The sequence of start-up steps is very important.

1. When experimenting with the Android mobile options (or the desktop/laptop PCs) make sure that the Mallory VM is running – not the server and GUI interface above – just the VM so that IP addresses are established.
2. The connection (a) Android mobile with no tunnel (b) Android mobile with tunnel, (c) PC with no tunnel (d) PC with tunnel must be established *after* the Mallory VM has been started but *before* the server and GUI scripts are run.

Checklist for operational problems:

1. DHCP on the Access Point should already have been disabled.

2. When moving this DHCP machine use “I moved it” as it is a Centos OS. When moving the bank server use “I copied it” and when moving the Mallory interception engine use “I copied it” although this is less critical as we will be configuring the eth0 and eth1 IP addresses.

4. The Mallory interception engine is entitled *Mallory-2-Wireshark* and the DHCP engine is *Mallory DHCP-SSL*

5. **All masks must be /24 (255.255.255.0) but you will not be able to ping the Access Point as it is on a different subnet.**

6. Make sure wireless switches are off for laptops or PCs with wireless access and make sure Virtual Box is disabled on any base machine.

7. All Androids have correct bookmarks and selecting bookmark instantly initiates a connection to the specified URL.

8. When e0 and e1 addresses are configure then we should be able to ping Mallory – Bank. Banks gateway needs to be e1 IP address

9. No need to use uplink on top end hub/switch. DHCP server connected to hub and AP connected to hub. Make sure Wireless has not been disabled on mobiles if not getting an IP address.

10. Do a preliminary test to ensure Androids are getting IP address before proceeding and then ping Androids from Mallory as a check.

11. With cascaded hubs use the uplink connection to connect upstream

12. Only now can the two Mallory python scripts be started

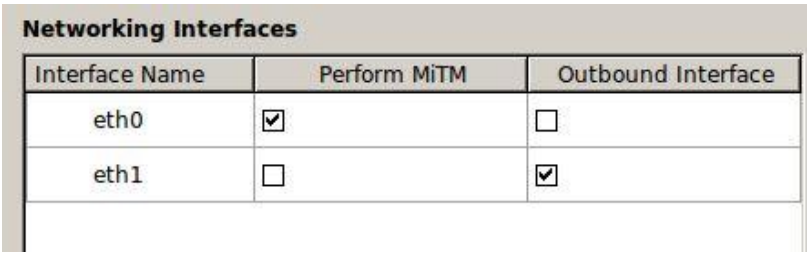
A. Victim is a directly connected Android mobile client (no tunnel)

In this case the Android mobile client has a wireless connection to the Access Point (10.1.1.1) and is running WPA thus all traffic is RC4 encrypted. The SSL encrypted traffic will run over the top of this connection.

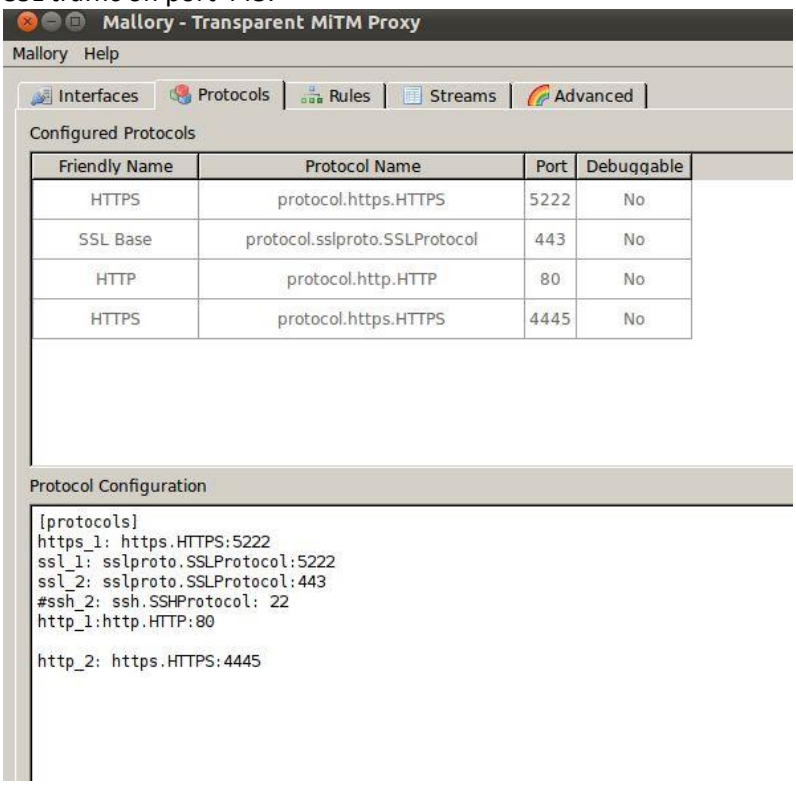
Be very careful that all IP addresses are unique and that pinging works and be careful about the *Android System IP Address* App which does not always update after a change.

1. Start the Mallory VM and establish eth0 and eth1 interfaces as detailed above.
2. Start the Android mobile and do ping tests as detailed above.
3. Start the Mallory Server and GUI as described above.
4. When the GUI window opens, select *Interfaces* tab and select *eth0* as the interface to perform the MITM attack and select *eth1* - which is connected to the server on a separate sub-network - as the outbound interface.
5. Now click “Apply Configuration”. The Network Interface display will depend upon what is connected when Mallory is stated. See window below for this option “A”

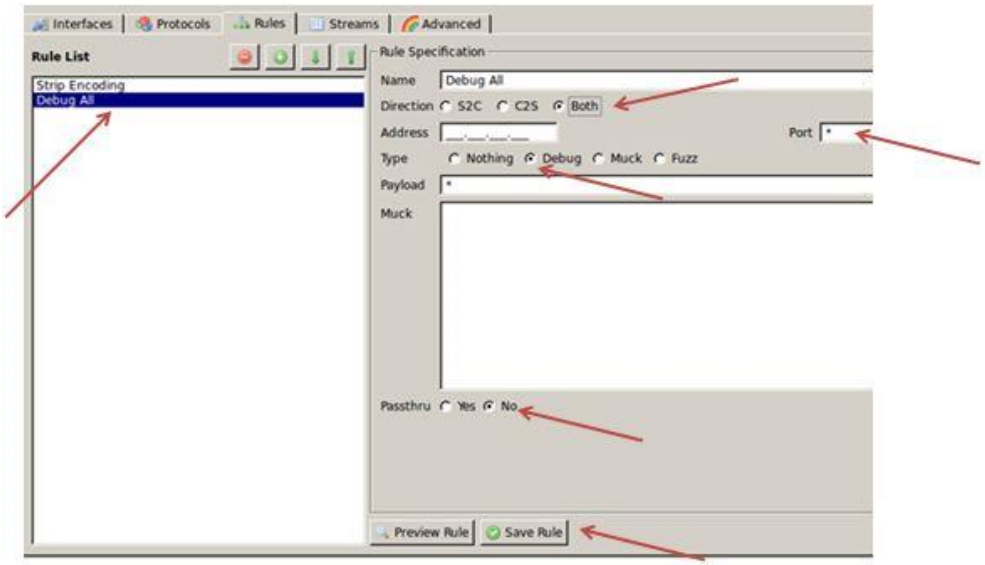
The Network Interface configuration in this case will appear as:



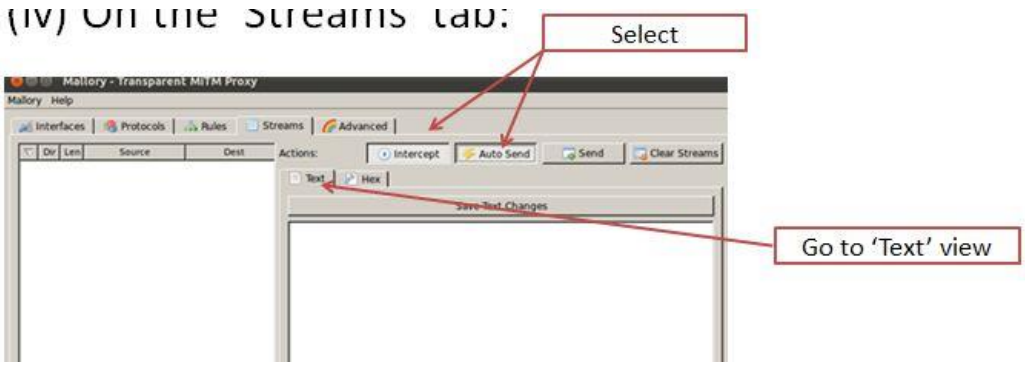
Now select *Protocols* tab and to apply the protocols listed in the Protocol Configuration box click apply and an entry should appear in the top window – see below. In particular we want to perform MITM attacks on SSL traffic on port 443.



Now set up the Rules on the next tab and select Debug All and follow the configurations options in the diagram below. Can select direction to/from client/server as (C2S or S2C) to limit amount of traffic or just Both. Save these rules.



On the Streams tab select Actions: *Intercept* and *Auto Send* and select *Text* view. See diagram below.



On your Android mobile go to bookmarks and open a session on your bank which will be https://10.x.1.254/SSL/login.asp

A client to server and server to client dialogue should appear in the window of the diagram above and the HTTP session dialogue will appear down the left hand side. By selecting a line on the left, the contents of that segment will be displayed as on the right.

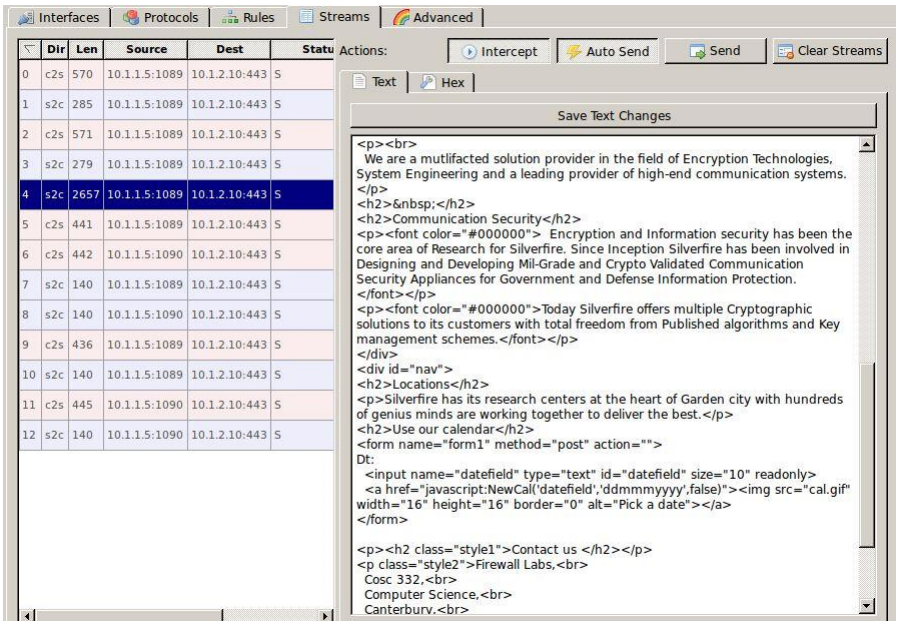
Accept the X.509 digital certificate and after some delay (as there is considerable processing to do with the certificates as well as decryption/encryption of the streams) the web page inviting ID and password will appear.

Note: Once the following URL for the bank is loaded (https://10.x.1.254/SSL/login.asp) wait for at least 30 seconds even if it says "Web page not available". Watch the *yellow* progress bar at the top of Android's screen. Processing is complex and thus slow. Also once ID and password have been entered the reply from bank may also take up to 30 seconds so be patient! There is a lot of complex certificate processing, decryption and encryption going on.

If we want to see the certificate on subsequent runs then there is an option at right hand end of URL bar. Once ID & PW are entered wait about 30 secs for bank account details to be displayed.

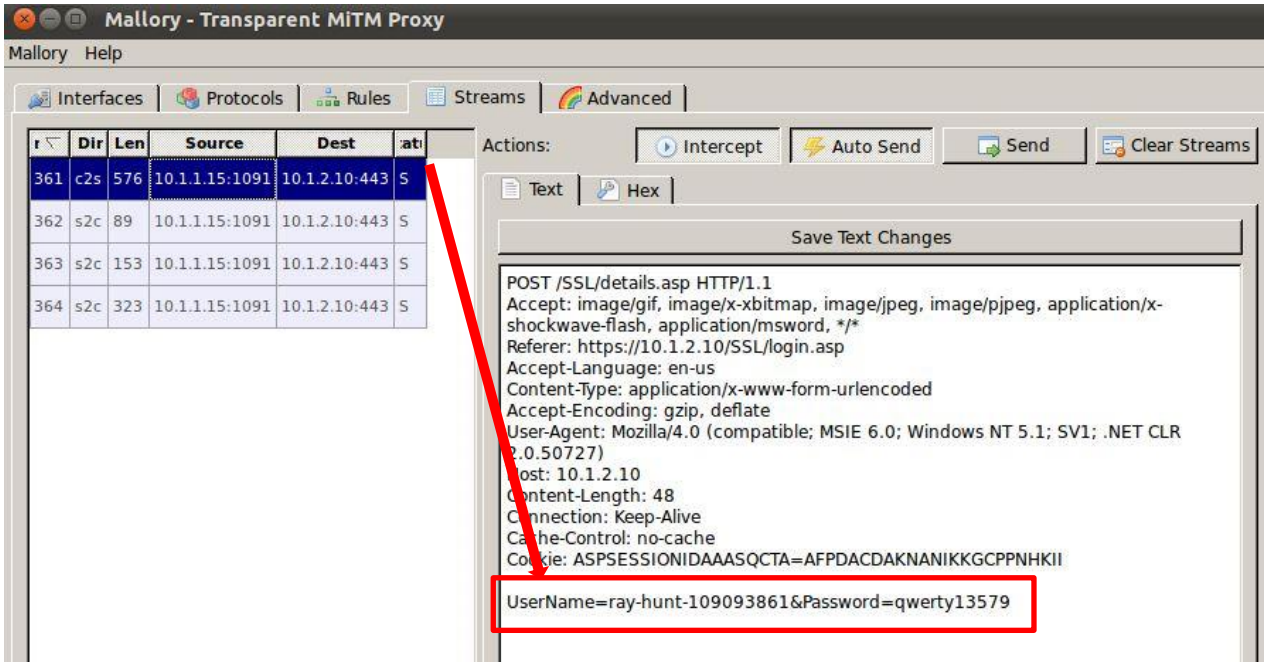
Question: Once web page displayed – how do we get bookmarks back?

Answer: go back …go back left hand button under screen until we get to bookmarks.



Clear the http stream but if forgotten then the ID/Password appears around no 235 and is 625-645 bytes long depending upon ID/PW length.

Clear Streams just before you enter ID and password, as the segment containing your ID and password will appear now. Why does no text appear in right side Mallory window? You must select frame in the left window to see text !!



Check that the Mallory interception engine can see your ID and password. Although in principle you can do this with your real bank, it is better to use the bank server that we have. Some banks can detect this MITM operation and trap such actions - however it varies greatly.

Wireshark

Try using Wireshark on the interception machine (as you will not be able to do this with the Android mobile) so that you can trace the traffic. (This will be SSL encrypted). Note that in the Interface options menu of Wireshark you may or may not see separate wire and tunnel interfaces. In this case it is likely that the two will be combined and you will see traffic on the wire with something like (10.x.1.2) as well as the tunnel end-point (192.168.0.x).

To run Wireshark - click on the Wireshark icon and use password of mitm. Otherwise enter sudo wireshark with same password. Just entering wireshark alone will not bring up the interfaces. Can also use gksudo - the primary purpose of which is to run graphical commands that need root.

Generally if things fail it is best to close and restart Mallory as well as restarting the Android. Also ping Android to ensure that the IP address is still working and that there is no IP address conflict. For any issues with address conflict, it is best to restart the bank server including the base machine.

B. Victim is a directly connected Android mobile client (VPN tunnel)

This is a complex option but is very interesting. Over part of the connection there are three layers of encryption in operation (i) RC4 over wireless (ii) PPTP on top of this between client and Mallory interception engine and (iii) SSL encryption end to end. The reason that the tunnel terminates on the interception engine is because in practice there would never be an end-to-end VPN tunnel as SSL provides the normal security. Nevertheless VPN connections to the network interface over the untrusted network are common.

The following procedures are used to set up a VPN tunnel on the Android mobile.

Configuring the mobile Android client

Go to Settings > Wireless & Network Settings > VPN Settings >

Enter Add VPN

Enter Add PPTP VPN

VPN Name: Mallory

Set VPN Server: IP address of the mallory server that is hosting PPTP service (10.x.1.2)

Enable Encryption: Yes (optional)

DNS Search Domains: blank (optional)

Now click Go Back ← bottom left button on Android

Click the Mallory VPN you just setup and enter the username and password set in the chap-secrets file (mallory/mallory). Case sensitive !!

Connect to the VPN as below:

Start-up VPN on Android. Settings > Wireless & Network Settings > VPN settings > Mallory – Connect to network. Username: *mallory* Password *mallory*

If connection fails repeatedly then restart the Mallory VM AND restart the Android mobile.

When connected ….

Notice the IP address allocated for the tunnel connection – both base and virtual end-points. See IP addresses for *eth0* and *ppp0* below.

If there is an issue with *pptp* on the interception engine, enter:

sudo service pptpd start and/or sudo /etc/init.d/pptpd restart

```
mitm@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:da:35:6e
          inet addr:10.1.1.2  Bcast:10.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feda:356e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1606 errors:0 dropped:0 overruns:0 frame:0
          TX packets:525 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:178372 (178.3 KB)  TX bytes:42917 (42.9 KB)
          Interrupt:19 Base address:0x2024

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:56096 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56096 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4882506 (4.8 MB)  TX bytes:4882506 (4.8 MB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:192.168.0.1  P-t-P:192.168.0.234  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1396  Metric:1
          RX packets:148 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:9492 (9.4 KB)  TX bytes:3380 (3.3 KB)
```

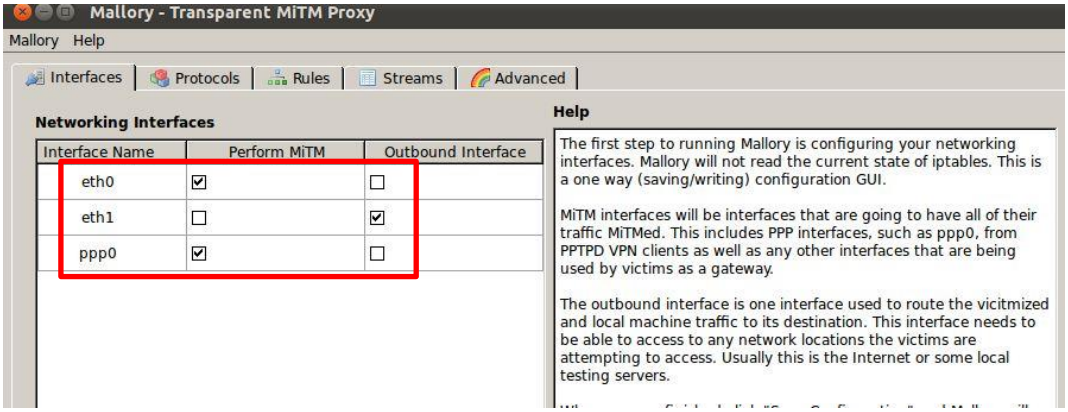
Sample configurations:

Tunnel IP addresses: 192.168.0.234 ---------- connects to ----------192.168.0.1

Lower layer: 10.x.1.3 (client) ----------connects to ----------10.x.1.2 (interface)

If you look at the mallory.py console, traffic should be showing up if you enabled debugging. See the usage guide for more information.

Make sure that the Networking Interfaces in the GUI appear as:



Follow all instructions (as above for Part A) with respect to Networking Interfaces, Protocols, Rules and Streams before commencing. Note that all IP Tables instructions get updated automatically.

Once the server bookmark is selected, communication with the server may take sometime - watch the yellow bar above the URL. There will also be a delay after entering ID and Password. There are essentially *3* layers of encryption in operation: WPA (RC4), PPTP and SSL

Note: Mallory (server and GUI) need to be restarted when moving from no VPN tunnel to VPN tunnel (or the reverse) whether the victim is a PC (wired) or Android mobile (wireless).

C. Victim is a directly connected PC with no VPN tunnel
D. Victim is a directly connected PC with a VPN tunnel

The principles for using a PC (desktop or laptop) are exactly the same and will not be repeated here as use of the Android mobile is of more practical value. *Further there is a bug in the Mallory interception engine which can cause the tunnel to fail as the rules are loaded – for Case D.*

In case you wish to experiment with this option the following steps need to be followed:

Set up VPN tunnel before starting Mallory's server and GUI although Mallory must be running (i.e. the Mallory VM machine – not the Mallory server or GUI. The VPN tunnel is established when the Mallory VM is running but NOT the Mallory server or GUI. ID/password: mallory/mallory. The connection is to 10.x.1.2 (not the bank server!)

Further analysis of the captured file

At bottom of screen reference is made to a tcpdump file, e.g. trafficdb_1384991997.64. This is stored at Home folder > Mallory > current > database. This can be examined using sqlitebrowser. Run the executable file “SQLite Database Browser 2.0 b1.exe” which is in the folder sqlitebrowser in the Mallory folder.

File > Open Database > trafficdb_1384991997.64 (for example)

Select Browse Data > flows and searching is possible (e.g. for ID & password). However it is better to:

File > Export > Table as CSV file (comma separated variables) > Flows > Export

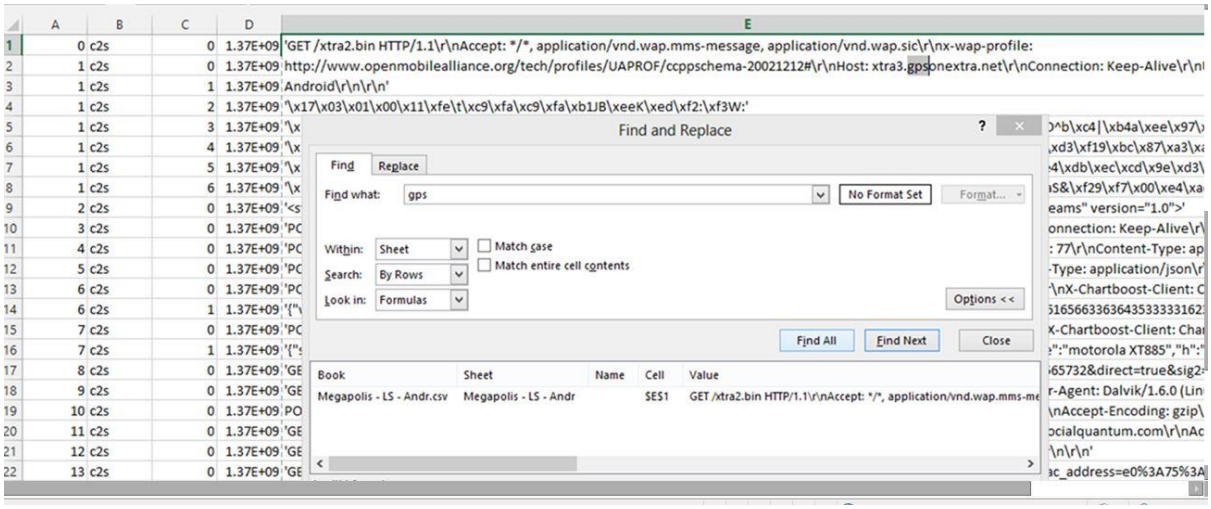
Give the file a name - e.g. test.csv and save it. Should say export completed.

Open this file in XL and there will be a list of C2S and S2C records.

Do a ‘find” for password as follows ….

Do *CTL F* then select FIND ALL. As a result you will get the Cell value where the word “Password” lies. Then double click on this cell.

In the screenshot below we are searching for “gps”. Now double-clicked on cell E1 and the full content is displayed.



Another alternative is to simply open the .csv file with Notepad or Wordpad and do a search for desired string e.g. “Password”