Monkey in the Middle: Why Imposter Syndrome Runs Rampant in Security

Abstract

Cyber Security is a field that requires dealing with many complexities; nothing is as simple as it appears on the surface. At times it feels as if you are stuck in an infinite loop of troubleshooting and tracking down errors, which can get quite disheartening, but a lot of the time the user is not at fault. Imposter syndrome runs rampant in this field because failure is so common (as a matter of fact inevitable), and it can be daunting to address these failures because expectations are set so high. In this paper I will highlight my many trials amongst sparse triumphs while attempting to carry out a seemingly straightforward WiFi attack. In the end, the most solidified learning came from struggling through the process, so the importance of perseverance cannot be stressed enough.

Introduction

The initial purpose of this project was to solely demonstrate the ease of stealing peoples' private information using a rogue WiFi access point, and raise awareness to the general public on how they can keep their data secure when joining public networks. Something called an "Evil Twin" WiFi hotspot can be "easily" configured by using the same SSID of any public network, to trick people into thinking they are joining a legitimate wireless network: "taking place at layer 2 in the OSI model, the data link layer", which is difficult to track (Secplicity). Once somebody falls into this trap, the hacker now has an arsenal of attacks to use, and in essence has complete control over everything the victim is doing while connected to the hotspot. The attacker is able to see any credentials sent in the clear via protocols such as HTTP and FTP, set up a backdoor for future access to the victim's device, or riddle their device with other forms

of malware. However it wasn't long before I hit the first of many roadblocks while turning theory into implementation, and it quickly became evident to me that there needed to be a slight shift in the focus of this paper, in order to elaborate on the extent of these obstacles.

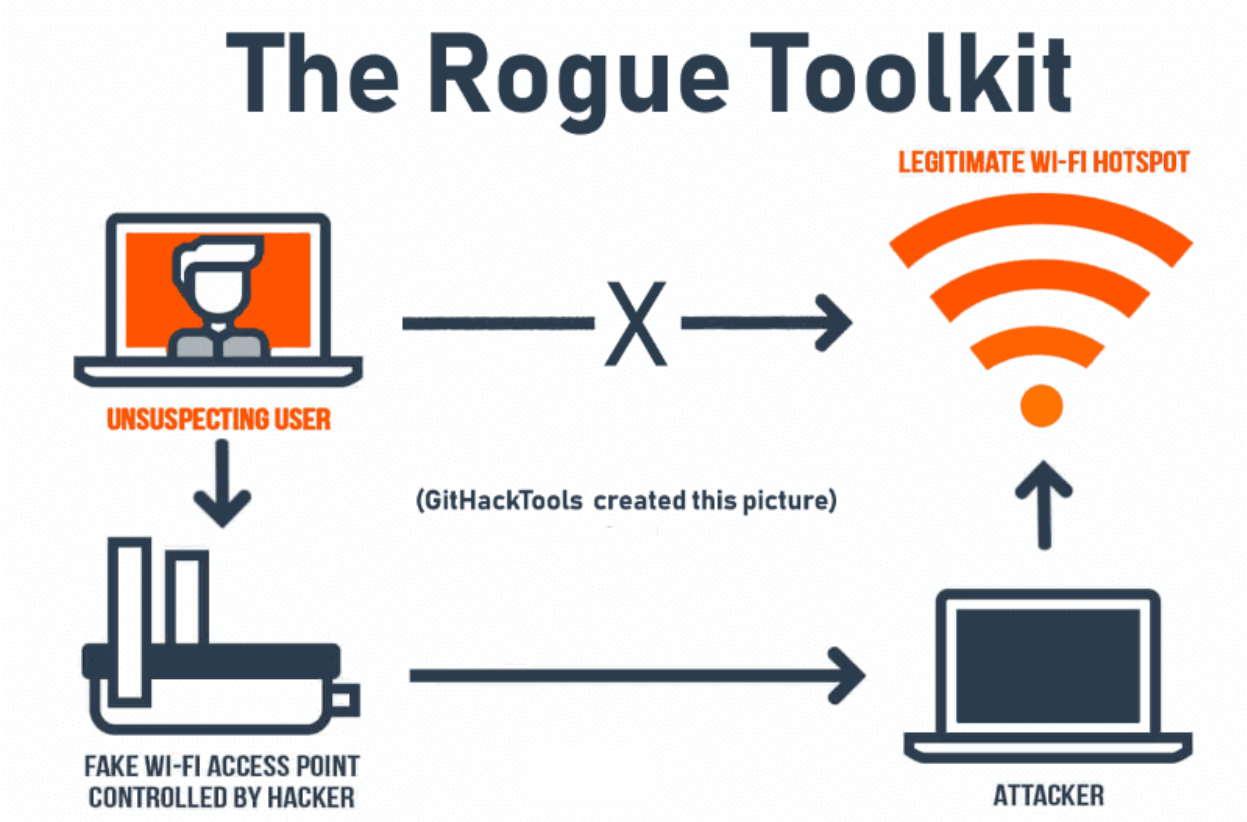
To The Community

There's a moral to be taken from my experience with this project (outside my initial goal to aid the susceptible general public), which I believe anyone in cyber security can relate to, but is especially important for people who are new to this line of work. If you're familiar with Greek mythology, then I'm sure you will have heard of Hydra; a multi-headed serpent, that if you cut off one head, two more would grow in its place. There are so many intricacies in this type of work, that it sometimes feels like you are fighting a never ending battle against Hydra. For example, so much time will be spent getting the hardware up and running, but once that problem is solved, then it turns out there are a bunch of issues with the software. This is where the idea of imposter syndrome comes into play. "Imposter Syndrome is defined as the persistent inability to believe that one's success is deserved or has been legitimately achieved as a result of one's own efforts or skills" (Medium). It may feel like you're just going around in circles breaking everything you touch, making it seem like you're not cut out for the work you're doing, but the reinforcement you get from working through all of the errors is how you learn best. Not to mention, much of the time the user is not at fault for these issues. So the moral is that many people in cyber security tend to undermine their ability to succeed, since the field is just so vast and complex, making it easy to feel inadequate.

In reality, even security professionals still find themselves dealing with these kinds of issues on a regular basis, because that is just the nature of the work in this field.

Details

My planned methodology was the following: First find a wireless USB adapter with a chipset that supports monitor mode (or promiscuous mode), then install software inside a virtual machine, operating Kali Linux, that can use the USB adapter to configure a rogue access point and monitor network activity. The way a rogue access point or "Evil Twin" works is the following: "a wireless device just outside of an organization receives beacons transmitted by legitimate access points within the organization. The evil twin then begins to transmit identical beacons with the intent of having end users connect with it. Once connected, the evil twin can then be used by nefarious individuals" (Techopedia).



Sounds fairly straightforward, right? In actuality, implementing this plan of attack proved to be quite the opposite.

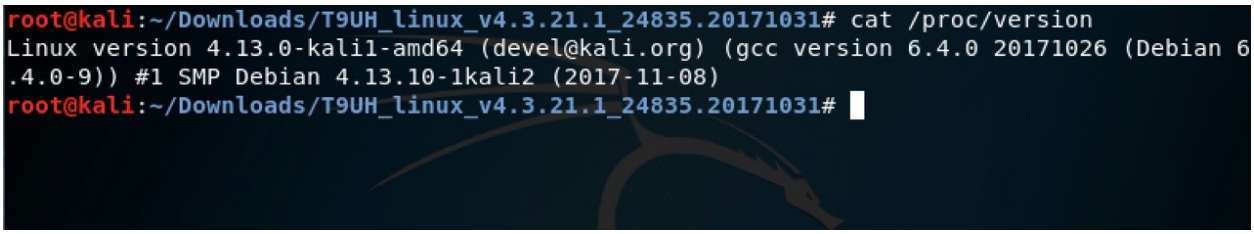
Configuring Hardware

Step one, which I thought would be the most manageable task, admittedly ended up being the biggest setback of the entire project. Due to outdated sources upon my initial research, I was led astray in my search for a compatible adapter. With high hopes, I purchased TP-link's Archer T2UH, high gain wireless dual band USB adapter. Things quickly started to get messy when I downloaded and tried to install the official driver for this device from the TP-link website, "making a mistake in purchasing a network adapter can add up quickly and be discouraging when first learning about Wi-Fi security" (Null Byte). I immediately got fatal makefile errors when attempting to compile the driver. It then became obvious that the driver for this adapter was no longer being maintained by TP-link, but I didn't lose hope just yet because I was not alone in my struggles. Many other people on the web had run into the same problem, so there were a lot of forums pertaining to this issue on helpful sites such as Stack Overflow, Reddit, etc… I tried downloading, and installing a countless number of supposedly patched drivers, which were maintained by disgruntled independent developers, but to no avail. After exhausting all of my options in this domain, I came up with what I thought was a genius idea to downgrade my Linux kernel and version of GCC to match that of the obsolete, but official, rendition of the TP-link T2UH driver. It turns out, again, this was not as easy as typing a couple of short commands into the terminal, but actually required me to download multiple iterations of Kali Linux virtual disk images, and setting up a grand total of 4 new VM's until I finally got it right.

Anyhow, I was about ready to give up, but instead I chose to use my newfound knowledge to hone my search for a new adapter that actually works. Since, I was crunched for time and all of the most prestigious adapters could only be ordered online, I resorted to purchasing another TP-link device, as this was the only listed device available in stores for my intended use. TP-link's Archer T9UH model uses a Realtek RTL8814AU chipset, which is among the list of chipsets currently supported by Kali Linux for most WiFi hacking tools. Nevertheless I was outraged when I tried installing the official corresponding driver from the TP-link website, only to get the same errors as before. TP-link is flawed here in two ways. One, they falsely advertise Linux as a supported operating system, and two, the manufacturer installation manual conflicts with manufacturer driver installation instructions on their website, in regards to the required version of Linux kernel and GCC.

Next, I found myself on hold with a TP-link technical support representative for around an hour. He had me send him a screenshot of the error message and the Linux kernel version and GCC version I was using.

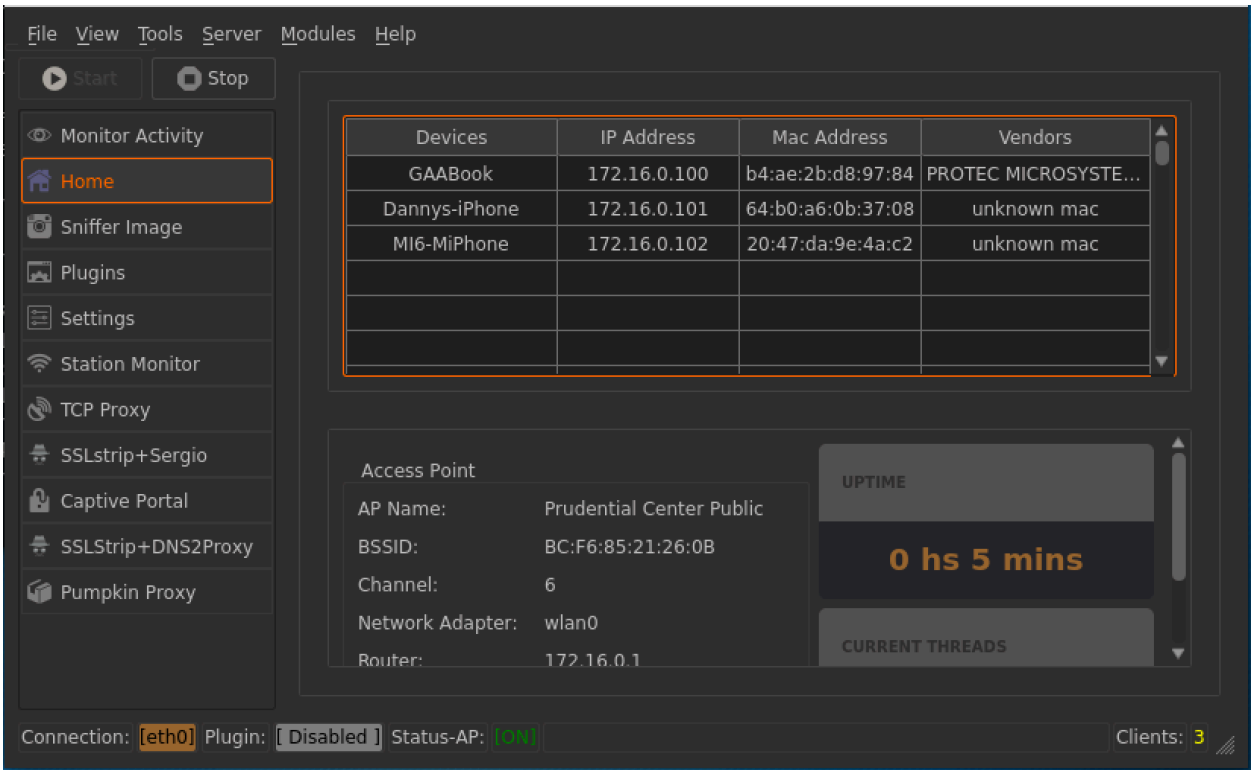
```
root@kali:~/Downloads/T9UH_linux_v4.3.21.1_24835.20171031# make clean
/bin/sh: 1: bc: not found
cd hal/phydm/ ; rm -fr */*.mod.c */*.mod */*.o */.*.cmd */*.ko
cd hal/phydm/ ; rm -fr *.mod.c *.mod *.o .*.cmd *.ko
cd hal/led ; rm -fr *.mod.c *.mod *.o .*.cmd *.ko
cd hal ; rm -fr */*/*.mod.c */*/*.mod */*/*.o */*/.*.cmd */*/*.ko
cd hal ; rm -fr */*.mod.c */*.mod */*.o */.*.cmd */*.ko
cd hal ; rm -fr *.mod.c *.mod *.o .*.cmd *.ko
cd core/efuse ; rm -fr *.mod.c *.mod *.o .*.cmd *.ko
cd core ; rm -fr *.mod.c *.mod *.o .*.cmd *.ko
cd os_dep/linux ; rm -fr *.mod.c *.mod *.o .*.cmd *.ko
cd os_dep ; rm -fr *.mod.c *.mod *.o .*.cmd *.ko
cd platform ; rm -fr *.mod.c *.mod *.o .*.cmd *.ko
rm -fr Module.symvers ; rm -fr Module.markers ; rm -fr modules.order
rm -fr *.mod.c *.mod *.o .*.cmd *.ko *~
rm -fr .tmp_versions
root@kali:~/Downloads/T9UH_linux_v4.3.21.1_24835.20171031# make
/bin/sh: 1: bc: not found
make ARCH=x86_64 CROSS_COMPILE= -C /lib/modules/4.13.0-kali1-amd64/build M=/root/Downlo
ads/T9UH_linux_v4.3.21.1_24835.20171031  modules
make[1]: *** /lib/modules/4.13.0-kali1-amd64/build: No such file or directory.  Stop.
Makefile:1714: recipe for target 'modules' failed
make: *** [modules] Error 2
root@kali:~/Downloads/T9UH_linux_v4.3.21.1_24835.20171031#
```



A day later I received this response, “This is valuable information, Allow me to forward this information for further check, Thank you!”, and that was the last I heard from the ever so helpful technical support team. At this point I couldn’t bear playing anymore of TP-links’ reindeer games, and decided to outsource from the disgruntled independent developers on GitHub, once again. This time I hit the jackpot. Alas, I was able to compile the makefile of an up-to-date driver for my chipset with zero errors! The really annoying part about dealing with all of these errors is that it’s a lot different than debugging code you write yourself; with that you're actually expecting the errors, but in reality I am dealing with industrial-grade software that is counted on to run smoothly.

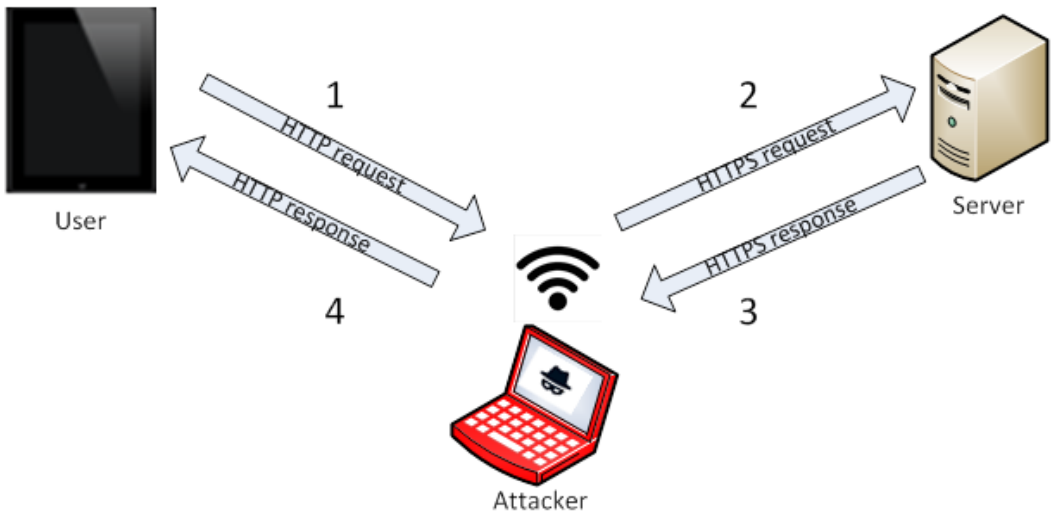
Configuring Software

The WiFi hacking tool I chose to use is called Wifi-Pumpkin. It is a powerful tool with a nice GUI that can do much more than just establish a hotspot and monitor network activity. When fully functional, Wifi-Pumpkin supplies users with many options of attack, such as social engineering attacks, phishing attacks, trojan file injections, XSS attacks, DOS attacks, key logging, and much more. This all sounds so great, is it too good to be true? The answer is somewhat. The tool could be buggy at times, and required me to downgrade certain modules I had installed to previous versions, which took a while to get working. Nonetheless, I was able to get most of the main features functioning after some tinkering. From here, I went to the Prudential Center in Boston, in order to test out my access point. It wasn’t long before I was able to lure in some unsuspecting civilians, in fact it only took five minutes.



So now that I've swindled a couple of clients into to joining my network, where's all the juicy information? I was able to view and log all URL's that were being accessed by the clients, but I wasn't seeing any user login data. It turns out there's another caveat…

The TCP proxy wasn't recovering any user credentials being sent in the clear because most sites use encryption nowadays. That's when I realized I needed to get the SSLStrip proxy up and running, so that I could force redirect all responses from HTTPS requests sent by the victim into HTTP responses, therefore allowing me to intercept all plaintext information being sent as unencrypted.



Yet again, I ran into more complications with improperly maintained software when deploying SSLStrip. I eventually managed to get it running in the sense that it gave me no errors, but it was not changing HTTPS responses to HTTP like it was supposed to. After spending some time looking for a solution, I concluded that it was time to move on, because it had already turned into a much bigger time commitment than I had anticipated, and after all I had other matters to attend to. Although I felt slightly defeated, I had truly learned a lot along the way.

Defenses

These rogue WiFi access points are alarming, but don't worry there are precautions that can be taken to protect against this type of attack. One tip is to always verify with the establishment or owner of the public WiFi hotspot if you have any doubts that the network you are joining is legitimate. This will eliminate any chance of joining a nefarious access point, but will not necessarily protect you from other MITM attacks carried out on the trusted network. To ensure the safety of your data from SSLStrip proxies and other attacks that redirect or deny HTTPS requests, you must utilize a VPN or similar methods of endpoint encryption. Even then, there is always the question of who you are actually trusting with your data when subscribing to a VPN service, and what it is that they actually do with it. One last suggestion would be to use 4G LTE or whatever form of telecommunication your cellular carrier provides, because it is typically more difficult to hack, so it is typically less polluted than public WiFi.

Conclusion

In theory, the concept of intercepting information via man in the middle attacks is pretty basic. The public's general awareness of potential threats is customarily low and

makes them vulnerable to joining malicious hotspots. Just a few minor tips to public users can significantly reduce risk. The implementation of access points and running the required scripts can be done with tools readily available online using hardware that is commercially available. The challenges I experienced during this project relate to hardware/firmware/software version incompatibilities, and blatant misinformation provided by commercial companies. This is expected to some degree, given that a lot of these tools are not necessarily widely used, and thus are not sufficiently tested for rapid deployment by the average user. Using hardware and software that have been widely tested can minimize compatibility issues. Online forums were extremely helpful in that regard. I hope by now it is clear how something that appears to be pretty basic in this field can so easily blow up in your face, and turn into a massive ordeal. Ultimately, the benefits of acquiring so much knowledge on this subject matter outweighs the pain of going down endless rabbit holes to get there. Next time I run into something of this sort, I will have a better sense of what works and what doesn't, and continue to build a better knowledge base.

Sources

- https://null-byte.wonderhowto.com/how-to/check-if-your-wireless-network-adapter-supports-monitor-mode-packet-injection-0191221/
- https://medium.com/@west.a.dominique/navigating-imposter-syndrome-in-the-world-of-information-security-dc6ec10d9c0f
- https://www.aircrack-ng.org/doku.php?id=compatibility_drivers
- https://www.techopedia.com/definition/4082/rogue-access-point-rogue-ap
- https://latesthackingnews.com/2019/08/09/wifi-pumpkin-wifi-security-audit-framework/
- https://medium.com/bugbountywriteup/how-i-made-a-fake-access-point-to-harvest-login-credentials-6898efb96b3b
- https://blog.envisionitsolutions.com/computer-security-and-fake-wi-fi-hotspots-a-criminals-tool-to-steal-from-you
- https://koolspan.com/beware-of-fake-wifi-hotspots/
- https://binged.it/2piUpvk
- https://miloserdov.org/?p=2196
- https://www.secplicity.org/2017/04/28/wi-fi-hacking-serious-threats-might-missing/