Analysis of Rogue Access Points using SDR

Juan C. Rios
Department of Electrical and Computer Engineering
University of California, Los Angeles
Los Angeles, United States
jcrios@ucla.edu

Charles Mercenit
Department of Math and Computer Science
Stetson University
DeLand, United States
ccmercenit@stetson.edu

Yongxin Liu
Electrical, Computer, Software and Systems Engineering Department
Embry-Riddle Aeronautical University
Daytonaa Beach, United States
liuy11@my.erau.edu

Jian Wang
Electrical, Computer, Software and Systems Engineering Department
Embry-Riddle Aeronautical University
Daytona Beach, United States
WANGJ14@my.erau.edu

Jiawei Yuan
Electrical, Computer, Software and Systems Engineering Department
Embry-Riddle Aeronautical University
Daytona Beach, United States
yuanj@erau.edu

Houbing Song
Electrical, Computer, Software and Systems Engineering Department
Embry-Riddle Aeronautical University
Daytona Beach, United States
h.song@ieee.org

***Abstract* — When people connect to the Internet with their mobile devices, they do not often think about the security of their data; however, the prevalence of rogue access points has taken advantage of a false sense of safety in unsuspecting victims. This paper analyzes the methods an attacker would use to create rogue WiFi access points using software-defined radio (SDR). To construct a rogue access point, a few essential layers of WiFi need simulation: the physical layer, link layer, network layer, and transport layer. Radio waves carrying WiFi packets, transmitted between two Universal Software Radio Peripherals (USRPs), emulate the physical layer. The link layer consists of the connection between those same USRPs communicating directly to each other, and the network layer expands on this communication by using the TUN/TAP interfaces to tunnel IP packets between the host and the access point. Finally, the establishment of the transport layer constitutes transceiving the packets that pass through the USRPs. In the end, we found that creating a rogue access point and capturing the stream of data from a fabricated "victim" on the Internet was effective and cheap with SDRs as inexpensive as $20 USD. Our work aims to expose how a cybercriminal could carry out an attack like this in order to prevent and defend against them in the future.**

I. Introduction

Mobile devices have made an on-the-go connection to the Internet a necessity; with social media deeply integrated into modern society and cell phones dominating human attention, most people check their phones numerous times each day [1]. This brings up an important issue, one that users hardly take into account when finding public WiFi access points: user security.

This research was supported by the National Science Foundation under Grant No. CNS-1757781.

When connecting to free WiFi access points, users rarely consider encrypting their connection using a virtual private network (VPN) and run the risk of unintentionally connecting to a rogue access point (RAP).

An RAP is an access point deployed by a hacker with the intent to siphon sensitive information from those who connect to it. One of the many ways an RAP can be dispensed involves using a tool known as software-defined radio (SDR).

The flexibility of SDR provides a strong advantage over the traditional method of interacting with radio signals; it can simulate the effects of expensive physical equipment (e.g. mixers, filters, amplifiers, modulators/demodulators, and detectors) with a single piece of hardware that manipulates those signals with powerful programs like GNU Radio [2] and *GQRX* [3].

Though in its early stages, SDR has proven to be a revolutionary technology for various agencies, organizations, and corporations: the U.S. military has utilized SDR mechanisms for their tactical radios, the satellite communications business has adopted SDR as a solution to difficulties in changing hardware in space, and the mobile infrastructure market has incorporated SDR to develop faster and more flexible networks [4]. Despite SDR having shown tremendous potential for numerous of applications, individuals with malicious intent have created ways to exploit its powerful features.

With SDR, a criminal could communicate with mobile devices and inconspicuously extract information that the victim believed to be safely transported to the server. This level of anonymity has become extremely dangerous, especially since these attacks often occur in crowded, public areas (e.g. a metropolitan city, a concert, a shopping mall, an airport, etc.).

In 2018, cybersecurity experts at Coronet reported the most likely locations to fall under attack by cybercriminals were airports. They also found that the San Diego International

Airport contained an RAP with a service set identifier (SSID) named #SANfreewifi that ran Address Resolution Protocol (ARP) poisoning attacks to change user MAC addresses and deliver information directly to the hacker. Coronet disclosed that each passenger had a 30% chance of connecting to a medium-risk network and an 11% chance of connecting to a high-risk network [5] at this airport alone.

To top it off, the cost of setting up an RAP falls below $100 USD [6] due to various open-source software programs and legally purchasable hardware devices. For example, many of the methods we followed could be reproduced with GNU Radio, Wireshark, and an RTL-SDR, which brings the total cost to around $30 USD. This method of hacking has become a frightening reality with a huge potential payoff for the attackers.

Victims of identity theft spend a tremendous amount of time dealing with emotional stress and financial burdens while seeking to prove their innocence. Often, victims never recuperate their assets and remain unable to recover, leaving them with wasted time, lost wages, and drained bank accounts.

Aside from personal information, criminals have a plethora of knowledge — ranging from sensitive business documents belonging to travelling businessmen and classified intelligence belonging to government officials — within their reach. In the wrong hands, this information has the potential to create catastrophic consequences: bankrupt companies and national security issues [7].

These consequences, combined with the vast amount of public WiFi access points, display how the average person can do little to determine if an access point is legitimate. This makes the public an easy target for attackers.

All of society will benefit from improved methods for detecting these crimes. Our paper will aid the research and development of such detection and prevention systems by revealing the methods used by hackers.

II. Related Works

The average person assumes an access point is secure and remains unaware of the danger that could lie behind the scenes, furthering the importance of detecting RAPs and the need to research and discuss them. In order to aid this discussion, we take the opposing approach and work to disclose how an attacker would launch these attacks. The importance of fully understanding these threats from both sides resides in the fact that the increasing base of Internet users fuels the growth of identity theft in the world.

In 2017, the United States Identity Theft Resource Center reported 1,579 identity breaches with 178,955,069 records exposed. Out of these attacks, digital identity thefts (phishing, ransomware/malware, skimming, RAPs) made up 940 of the breaches (59.5%) and 167,549,245 of the records exposed (93.6%) [8]. This demonstrates that not only does the Internet make up where most of identity theft cases in the United States happen, it also makes the most efficient method of attack for cybercriminals to use. Fig. 1 depicts a rudimentary explanation of how network packets can get stolen.

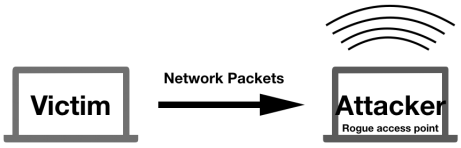


Figure 1. A simplified version of the attack

RAPs present the most common method of accessing sensitive information and consist of an access point with a similar SSID to a well-established and reputable access point nearby (e.g. setting the RAP's SSID to "iHop Free WiFi" next to a legitimate iHop WiFi with a WPA2 key). This baits users to connect to the RAP, and network packets containing sensitive information (usernames, passwords, credit card information, etc.) begin to flow to the attacker for decoding.

Currently, limited work to detect evolving rogue WiFi access point technology with SDR exists. Some work relies on detecting and measuring the strength of signals [6], while other papers use established intrusion detection systems (IDS) such as statistical analysis [9], wireless traffic monitoring, and feature extraction/timing-based solutions [10]. However, hardly any published work explains how an attacker would deploy such an attack using SDR.

We based a lot of our work off of one important paper authored by B. Bloessl et al. [11] Using the techniques for creating an orthogonal frequency-division multiplexing (OFDM) receiver in GNU Radio outlined by Bloessl, we created a receiver and transmitter with SDR to mimic an RAP.

III. Plan, Method, Tools, Etc.

A. Plan

We attempt to recreate the system a hacker would use in order to experience and understand the process an attack could follow. Our final objective involves manufacturing a "victim" by connecting a *Raspberry Pi* [12] to our RAP and subsequently capturing all of the traffic generated by it. By simulating the victim using this system, we can interpret the infiltrated Open Systems Interconnection (OSI) layers.

The first step of building this model includes receiving and transmitting with our USRP B210 [13] and USRP N210 [14] to establish the physical layer. After accomplishing that, unicasting between the USRPs becomes the next important task. Successfully unicasting means the link layer has been introduced. Ultimately, using the USRPs to transceive data establishes the network and transport layers, and our RAP is effectively deployed.

B. Reception

We began by analyzing FM radio waves with GNU Radio. After successfully constructing a flowgraph for listening to the radio, our first major goal included capturing network packets from genuine WiFi spectrums. Fig. 2 shows how we analyzed 2.4 and 5.0 GHz frequencies and observed the multitudes of network packets transmitting in the air using a waterfall graph.

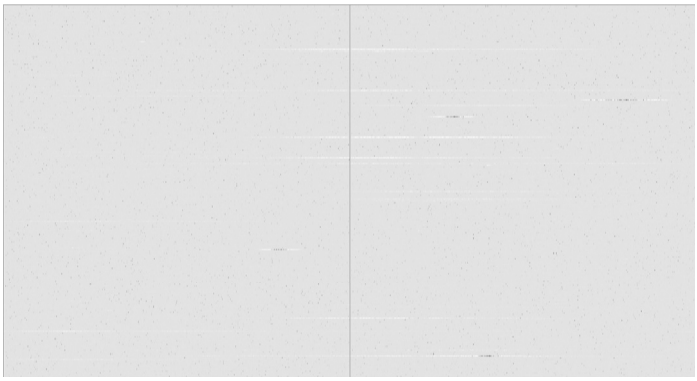


Figure 2. Waterfall graph illustrating network packets (the "hotspots" in the graph) corresponding to a frequency in 802.11g spectrum

Reading these packets required Wireshark [15] and a Wireshark connection block provided by the GitHub repository *gr-foo* [16]; combining these allowed us to analyze the network packets that our SDR captured with GNU Radio.

After successfully implementing this method, we possessed the ability to inspect the packets transported through the network and captured by GNU Radio. Fig. 3 displays some of the network packets we captured in Wireshark. After we established the information reception phase of the project, our next step became transmission.

C. Transmission

The physical layer, the lowest layer in computer networking, defines the hardware used to physically connect computers together. In our case, it consisted of radio waves that transmitted data between our SDRs.

We began by attempting to transmit an audio file to a nearby FM radio. Using a .wav file and a wide-band FM transmission block, we converted the audio signal to a radio signal and fed it into a rational resampler to increase the frequency. Afterwards, the signal passed through the USRP sink — the last step of transmitting the .wav file — before getting picked up by the radio.

After successfully performing this transmission, we began to implement this system with WiFi signals. Building off of an IEEE 802.11 a/g/p module [17] created by B. Bloessl, we captured network packets from 2.4 and 5.0 GHz WiFi spectrums. In order to pinpoint which frequency our system should listen to, we used GNU Radio to specify the physical medium and phase-shift keying that the physical layer required. For our specific case, we elected to analyze both 2.4 GHz (802.11g) and 5.0 GHz (802.11a) signals with simple binary phase-shift keying (BPSK) rather than quadrature phase-shift keying (QPSK). BPSK is the most simple method to encode data by transmitting one bit per symbol while QPSK offers transmission of two bits per symbol; however, this enhanced rate of transmission results in a higher chance of incorrectly encoded QPSK symbols.

$$\phi(t) = \sqrt{\frac{2}{T_b}} cos(2\pi f_c t) \tag{1}$$

Eqn. 1 represents the signal space in BPSK modulation

D. Unicasting

In the OSI model, the link layer is the second lowest layer. This layer manages the communication protocols that operate with devices directly connected to the host and provides unique MAC addresses to identify network members.

We simulated the link layer by unicasting an encoded text file between the USRP B210 and the USRP N210. Unicasting refers to the one-to-one communication between a sender and a receiver over a network. This represents the link layer by producing the first data connection created between the SDRs.

No.	Time	Source	Destination	Protoco	Length	Info
4827	181.543521	10.33.109.162	255.255.255.255	UDP	176	50222 → 50222 Len=99
4828	181.574982	8e:15:54:aa:0f:65	Broadcast	802.11	433	Beacon frame, SN=3260, FN=0, Flags=........, BI=100, SSID=ERAUNet
4829	181.609894	aa:15:54:aa:0f:65	Broadcast	802.11	438	Beacon frame, SN=3331, FN=0, Flags=........, BI=100, SSID=ERAUStuden…
4830	181.643796	8a:15:54:aa:0f:65	Broadcast	802.11	382	Beacon frame, SN=709, FN=0, Flags=........, BI=100, SSID=EagleNet
4831	181.677500	8e:15:54:aa:0f:65	Broadcast	802.11	433	Beacon frame, SN=3261, FN=0, Flags=........, BI=100, SSID=ERAUNet
4832	181.711963	aa:15:54:aa:0f:65	Broadcast	802.11	438	Beacon frame, SN=3332, FN=0, Flags=........, BI=100, SSID=ERAUStuden…
4833	181.746504	8a:15:54:aa:0f:65	Broadcast	802.11	382	Beacon frame, SN=710, FN=0, Flags=........, BI=100, SSID=EagleNet
4834	181.747819	10.33.109.210	224.0.0.251	MDNS	138	Standard query 0x0002 PTR _CC32E753._sub._googlecast._tcp.local, "QM…
4835	181.748097	10.33.109.149	8.8.8.8	ICMP	135	Echo (ping) request id=0x0c58, seq=25/6400, ttl=64 (no response fou…
4836	181.748373		IntelCor_ee:5a:7f (…	802.11	27	Acknowledgement, Flags=........
4837	181.759927	8a:15:54:aa:0f:65 (…	IntelCor_ee:5a:7f (…	802.11	36	VHT NDP Announcement, Flags=........
4838	181.761953	8a:15:54:aa:0f:65 (…	IntelCor_ee:5a:7f (…	802.11	33	Request-to-send, Flags=........
4839	181.763770	8a:15:54:aa:0f:65 (…	IntelCor_ee:5a:7f (…	802.11	33	Request-to-send, Flags=........
4840	181.779880	8e:15:54:aa:0f:65	Broadcast	802.11	433	Beacon frame, SN=3262, FN=0, Flags=........, BI=100, SSID=ERAUNet
4841	181.814012	aa:15:54:aa:0f:65	Broadcast	802.11	438	Beacon frame, SN=3333, FN=0, Flags=........, BI=100, SSID=ERAUStuden…
4842	181.849267	8a:15:54:aa:0f:65	Broadcast	802.11	382	Beacon frame, SN=711, FN=0, Flags=........, BI=100, SSID=EagleNet
4843	181.849442	10.33.108.53	255.255.255.255	UDP	178	50222 → 50222 Len=101
4844	181.850431	10.33.109.41	255.255.255.255	UDP	178	50222 → 50222 Len=101
4845	181.861797	IntelCor_ee:5a:7f	8a:15:54:aa:0f:65	802.11	43	QoS Null function (No data), SN=0, FN=0, Flags=...P...T
4846	181.862156		IntelCor_ee:5a:7f (…	802.11	27	Acknowledgement, Flags=........
4847	181.882654	8e:15:54:aa:0f:65	Broadcast	802.11	433	Beacon frame, SN=3263, FN=0, Flags=........, BI=100, SSID=ERAUNet
4848	181.916940	aa:15:54:aa:0f:65	Broadcast	802.11	438	Beacon frame, SN=3334, FN=0, Flags=........, BI=100, SSID=ERAUStuden…
4849	181.951635	8a:15:54:aa:0f:65	Broadcast	802.11	382	Beacon frame, SN=712, FN=0, Flags=........, BI=100, SSID=EagleNet
4850	181.952123	10.33.108.65	10.33.109.255	NBNS	127	Name query NB DESKTOP-4SV0OR1<1c>

Figure 3. Network packets collected by GNU Radio from genuine access points are exported to Wireshark

The initial step involved the B210 broadcasting a text file while the N210 listened to the frequency. Once the N210 located the signal with GNU Radio, it grabbed the wirelessly transmitted packets from the air and fed them into a Gaussian Minimum Shift Keying (GMSK) demodulator. GMSK, a type of continuous-phase frequency modulation, originates from Minimum Shift Keying (MSK). It includes a modification to smooth out the transitions between points in a constellation graph using a Gaussian filter. Using GMSK avoids overextension of the sidebands from the carrier. After exiting the GMSK demodulator, the packet decoder extracted the contents of the signal and stored them into a local, readable text file.

The next step required sending the file directly between the B210 and the N210. To achieve this, we modified the USRP source/sink blocks to transmit and receive only to and from each other. Unicasting the text file avoided the obvious downfall of broadcasting: third parties having the option to grab the information out of the air. Fig. 4 shows this process.

E. Transceiving

The network layer and transport layer make up the third and fourth layers of the OSI model. The network layer is responsible for routing data across intermediate network members, while the transport layer is in charge of Internet protocols in end-to-end communication over a network.

The most common protocols, *Transmission Control Protocol* (TCP) and *User Datagram Protocol* (UDP), are vastly distinct from one another. TCP checks for errors during transmission by waiting for an acknowledgement from the recipient while UDP sends data faster but without confirmation of delivery. These deviating approaches define unique characteristics within each protocol. TCP ensures reliability but uses larger, slower packets while UDP fails to guarantee reception but uses smaller and quicker packets.

We simulated the network layer by combining the reception and transmission of network packets to build a system that could transport information through intermediate hosts with a maximum transition unit (MTU) of 10 kB. We established the transport layer by utilizing a packet data unit (PDU) socket to determine whether a server requests a TCP or a UDP transfer. Once formed, we successfully transceived data with both SDRs on 2.4 and 5.0 GHz (indicated by Fig. 6). The final step included creating an SSID and enabling a connection with mobile devices through network management.

F. Network Management

In computer networking, most network interfaces have an associated physical device that manages the transmission and reception of data packets. For our simulation, we used a virtual network interface to handle packets. Virtual network interfaces differ from traditional interfaces by controlling packets purely with software. Two commonly used virtual interfaces include TUN (network tunneling) and TAP (network tapping). These two interfaces target specific layers within the network: TUN aims to transport IP packets within the network layer while TAP carries Ethernet frames in the link layer, portrayed by Fig. 5. Because of this, TUN has the ability to create point-to-point connections and TAP broadcasts traffic to various hosts. As a result, we use TUN for direct communication between an RAP and hosts.

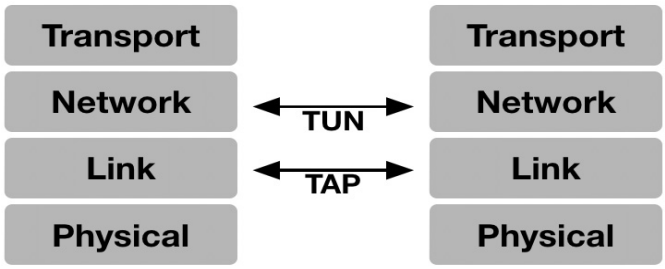


Figure 5. Illustration of the locations of TUN/TAP in the OSI layers

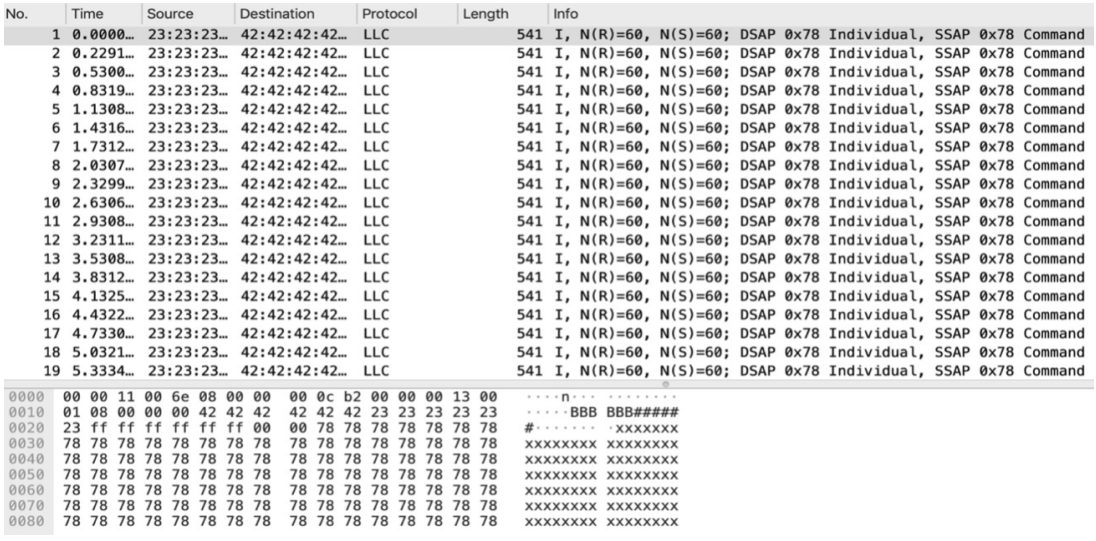


Figure 4. A text file's packets collected from unicasting a textfile from the N210 to the B210 before being decoded

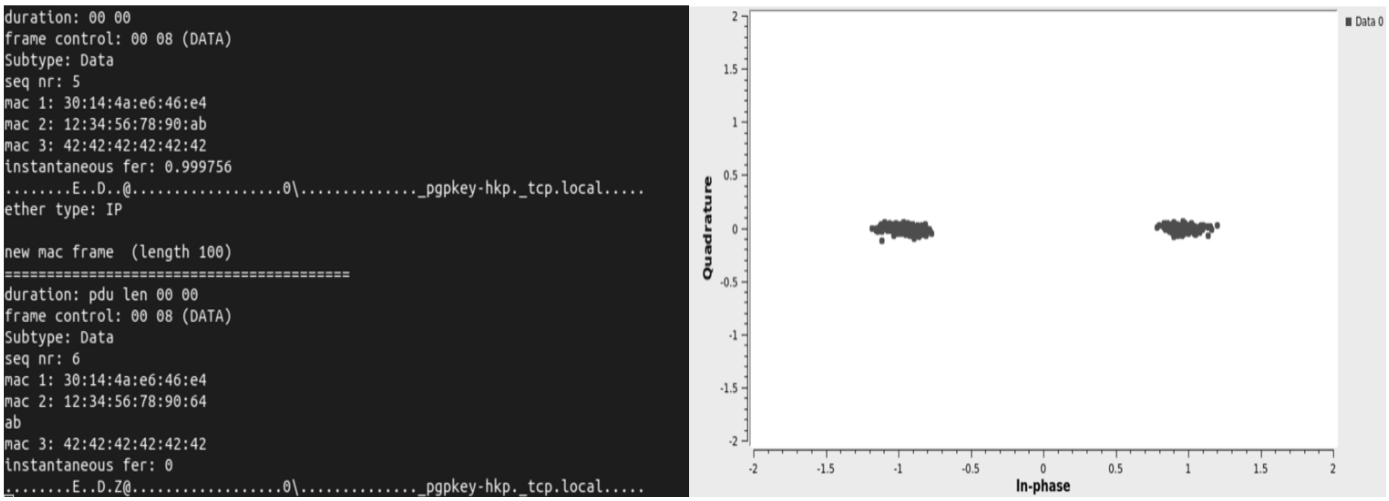


Figure 6. Transceiving network packets on GNU Radio with the N210

Using tunnel.py, a program located within the GNU Radio source files, we pinged between two computers on different networks using our SDRs. Finally, we used *hostapd* [18] to broadcast the SSID to WiFi-enabled mobile devices from a single SDR. In order to integrate hostapd with our flowgraph in GNU Radio, we exposed the inputs and outputs of the data stream and monitored the access point that devices connected to.

Once we deployed our RAP, hostapd handled the threeway handshake that Fig. 7 shows. A three part procedure, this handshake entails both the client and server sending synchronize (SYN) and acknowledge (ACK) packets before establishing a connection. After initiating the connection, we used the N210 with GNU Radio and Wireshark to read the information transferring between the connected devices and our RAP.

IV. Experimental Results

Throughout our research, we found ways that a hacker could build their own computer network modeled after the OSI model in order to gain access to user information. We gained an understanding of how cybercriminals deploy RAPs and the network weaknesses they exploit. Using SDR, we successfully recreated our own physical, link, network, and transport layers to implement an RAP.

The RAP easily listens to user activity and extracts information sent across its network. In our case, we evaluated our RAP by connecting a *Raspberry Pi* and surfing the Internet (i.e. visiting several sites and signing in to numerous accounts).

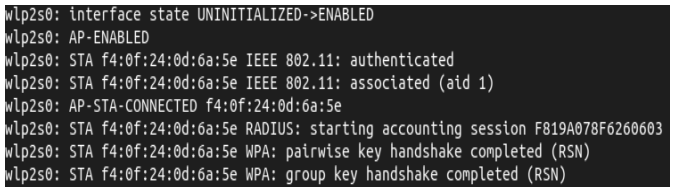


Figure 7. Handshake protocol between the RAP and the host

In our testing, we found that Wireshark can easily decode login credentials on websites with poor security. Fig. 8 reveals some of the packets we captured. According to an article published by the Center for Internet Security, 33-59% of people use the same password for multiple accounts [19]. The high percentage of password reuse makes having access to just one enough for an attacker to begin employing tactics like credential stuffing. With this strategy, criminals seek to use information obtained from one breach to sign into thousands of unrelated accounts [20].

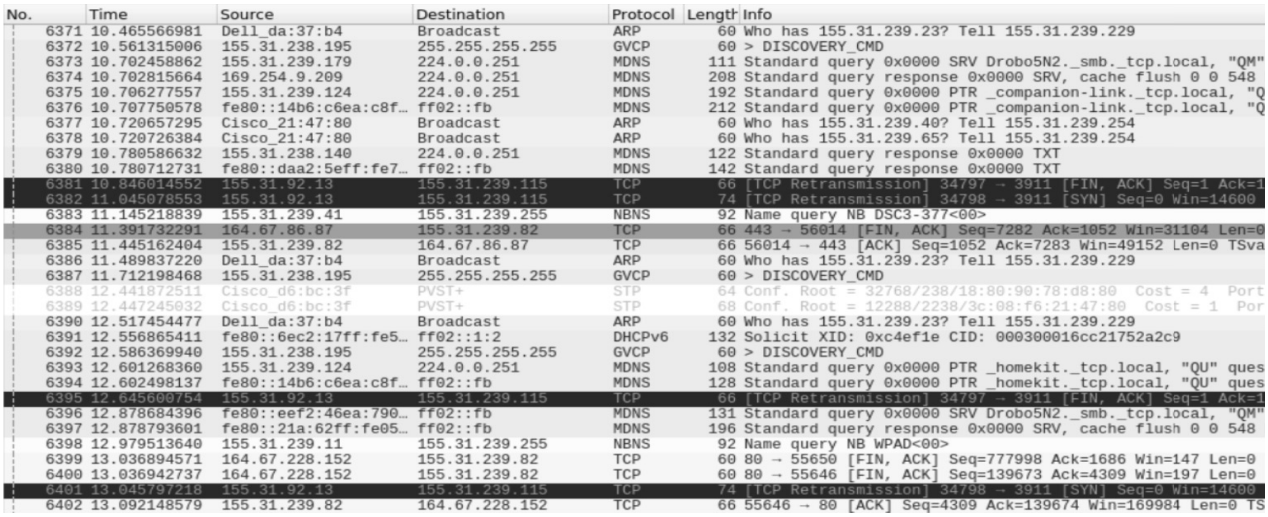
V. Conclusion and Future Work

Identity theft leaves victims helpless and in ruins. Online identity thefts constitute the vast majority of all cases, yet users rarely consider the security of information sent through open access points. In this paper, we demonstrate a hacker's process for setting up an RAP to intrude on a user's online activity. Not only does this display the power of software-defined radio, it also signifies a glaring security flaw in computer networks.

For the future of this project, we will incorporate the remaining three layers (session, presentation, and application) of the seven-layer OSI model into our recreation of an RAP. With these layers, we can continue to analyze security flaws that hackers exploit and learn how to defend against them.

VI. Acknowledgements

We would like to thank Embry-Riddle Aeronautical University and Ashok Vardhan Raja for providing mentorship during our REU experience.



No.	Time	Source	Destination	Protocol	Length	Info
6371	10.465566981	Dell_da:37:b4	Broadcast	ARP	60	Who has 155.31.239.23? Tell 155.31.239.229
6372	10.561315006	155.31.238.195	255.255.255.255	GVCP	60	> DISCOVERY_CMD
6373	10.702458862	155.31.239.179	224.0.0.251	MDNS	111	Standard query 0x0000 SRV Drobo5N2._smb._tcp.local, "QM"
6374	10.702815664	169.254.9.209	224.0.0.251	MDNS	208	Standard query response 0x0000 SRV, cache flush 0 0 548
6375	10.706277557	155.31.239.124	224.0.0.251	MDNS	192	Standard query 0x0000 PTR _companion-link._tcp.local, "Q
6376	10.707750578	fe80::14b6:c6ea:c8f…	ff02::fb	MDNS	212	Standard query 0x0000 PTR _companion-link._tcp.local, "Q
6377	10.720657295	Cisco_21:47:80	Broadcast	ARP	60	Who has 155.31.239.40? Tell 155.31.239.254
6378	10.720726384	Cisco_21:47:80	Broadcast	ARP	60	Who has 155.31.239.65? Tell 155.31.239.254
6379	10.780586632	155.31.238.140	224.0.0.251	MDNS	122	Standard query response 0x0000 TXT
6380	10.780712731	fe80::daa2:5eff:fe7…	ff02::fb	MDNS	142	Standard query response 0x0000 TXT
6381	10.846014552	155.31.92.13	155.31.239.115	TCP	66	[TCP Retransmission] 34797 → 3911 [FIN, ACK] Seq=1 Ack=1
6382	11.045078553	155.31.92.13	155.31.239.115	TCP	74	[TCP Retransmission] 34798 → 3911 [SYN] Seq=0 Win=14600
6383	11.145218839	155.31.239.41	155.31.239.255	NBNS	92	Name query NB DSC3-377<00>
6384	11.391732291	164.67.86.87	155.31.239.82	TCP	66	443 → 56014 [FIN, ACK] Seq=7282 Ack=1052 Win=31104 Len=0
6385	11.445162404	155.31.239.82	164.67.86.87	TCP	66	56014 → 443 [ACK] Seq=1052 Ack=7283 Win=49152 Len=0 TSva
6386	11.489837220	Dell_da:37:b4	Broadcast	ARP	60	Who has 155.31.239.23? Tell 155.31.239.229
6387	11.712198468	155.31.238.195	255.255.255.255	GVCP	60	> DISCOVERY_CMD
6388	12.441872511	Cisco_d6:bc:3f	PVST+	STP	64	Conf. Root = 32768/238/18:80:90:78:d8:80 Cost = 4 Port
6389	12.447245032	Cisco_d6:bc:3f	PVST+	STP	68	Conf. Root = 12288/2238/3c:08:f6:21:47:80 Cost = 1 Por
6390	12.517454477	Dell_da:37:b4	Broadcast	ARP	60	Who has 155.31.239.23? Tell 155.31.239.229
6391	12.556865411	fe80::6ec2:17ff:fe5…	ff02::1:2	DHCPv6	132	Solicit XID: 0xc4ef1e CID: 000300016cc21752a2c9
6392	12.586369940	155.31.238.195	255.255.255.255	GVCP	60	> DISCOVERY_CMD
6393	12.601268360	155.31.239.124	224.0.0.251	MDNS	108	Standard query 0x0000 PTR _homekit._tcp.local, "QU" ques
6394	12.602498137	fe80::14b6:c6ea:c8f…	ff02::fb	MDNS	128	Standard query 0x0000 PTR _homekit._tcp.local, "QU" ques
6395	12.645600754	155.31.92.13	155.31.239.115	TCP	66	[TCP Retransmission] 34797 → 3911 [FIN, ACK] Seq=1 Ack=1
6396	12.878684396	fe80::eef2:46ea:790…	ff02::fb	MDNS	131	Standard query 0x0000 SRV Drobo5N2._smb._tcp.local, "QM"
6397	12.878793601	fe80::21a:62ff:fe05…	ff02::fb	MDNS	196	Standard query response 0x0000 SRV, cache flush 0 0 548
6398	12.979513640	155.31.239.11	155.31.239.255	NBNS	92	Name query NB WPAD<00>
6399	13.036894571	164.67.228.152	155.31.239.82	TCP	60	80 → 55650 [FIN, ACK] Seq=777998 Ack=1686 Win=147 Len=0
6400	13.036942737	164.67.228.152	155.31.239.82	TCP	60	80 → 55646 [FIN, ACK] Seq=139673 Ack=4309 Win=197 Len=0
6401	13.045797218	155.31.92.13	155.31.239.115	TCP	74	[TCP Retransmission] 34798 → 3911 [SYN] Seq=0 Win=14600
6402	13.092148579	155.31.239.82	164.67.228.152	TCP	66	55646 → 80 [ACK] Seq=4309 Ack=139674 Win=169984 Len=0 TS

Figure 8. Information captured from our *Raspberry Pi* connected to our RAP and browsing ucla.edu

REFERENCES

[1] NY Post, "Americans Check Their Phones 80 Times a Day: Study". [Online]. Available: https://nypost.com/2017/11/08/americans-check-their-phones-80-times-a-day-study/. [Accessed: 1- Aug- 2019]

[2] GNU Radio, 2019. Available: https://www.gnuradio.org

[3] GQRX, 2019. Available: http://gqrx.dk

[4] Wireless Innovation Forum, "SDR Market Size Study", wirelessinnovation.org, 2011. [Online]. Available: https://www.wirelessinnovation.org/assets/documents/mexp-sdr-11%20final.pdf

[5] "Airport Networks Are Putting Your Devices & Cloud Apps At Severe Risk", Coronet, 2018. [Online]. Available: https://www.coro.net/wp-content/uploads/2018/08/Coronet _Cyber-Insecure-Airports.pdf.[Accessed: 1- Jul- 2019]

[6] J. Wang, N. Juarez, E. Kohm, Y. Liu, J. Yuan and H. Song, "Integration of SDR and UAS for Malicious Wi-Fi Hotspots Detection," 2019 Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, USA, 2019, pp. 1-8. doi:10.1109/ICNSURV.2019.8735296, URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8735296&isnumber=8735100

[7] E. Gardner, "Is airport public Wi-Fi cyber-secure?", Airport Technology, 2018. [Online]. Available: https://www.airport-technology.com/features/airport-public-wi-fi-cyber-secure/. [Accessed: 19- Jul- 2019].

[8] Identity Theft Center, "2017 Annual Data Breach Year End Review", Idtheftcenter.org, 2018. [Online]. Available: https://www.idtheftcenter.org/images/breach/2017 Breaches/2017AnnualDataBreachYearEndReview.pdf. [Accessed: 12- Jul- 2019].

[9] Chao Yang, Yimin Song, Guofei Gu, September 2012, Active User-Side Evil Twin Access Point Detection Using Statistical Techniques, IEEE Transactions on Information Forensics and Security, Volume: 7, Issue: 5, pp: 1638 - 1651.

[10] Mayank Agarwal, Santosh Biswas, Sukumar Nandi, March 2018, An Efficient Scheme to Detect Evil Twin Rogue Access Point Attack in 802.11 Wi-Fi Networks, International Journal of Wireless Information Networks, Volume: 2, Issue: 25, pp: 130 - 145.

[11] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "An IEEE 802.11a/g/p OFDM Receiver for GNU Radio", 2019 Special Interest Group on Data Communication (SIGCOMM), Hong Kong, 2013, pp. 9-16. doi:10.1145/2491246.2491248 URL: https://homepages.dcc.ufmg.br/ mmvieira/cc/papers/OFDM%20receiver%20GNU%20Radio.pdf

[12] Raspberry Pi 3 Model B, 2019. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[13] USRP B210. Available: http://files.ettus.com/manual/page_usrp_b200.html

[14] USRP N210. Available: http://files.ettus.com/manual/page_usrp2.html

[15] Wireshark, 2019. Available: https://www.wireshark.org

[16] B. Bloessl, GR-foo. 2014. Available: https://github.com/bastibl/gr-foo

[17] B. Bloessl, IEEE 802.11 a/g/p Transceiver. 2014. Available: https://github.com/bastibl/gr-ieee802-11

[18] Hostapd, 2019. Available: https://w1.fi/hostapd/

[19] Center for Internet Security, "Reusing Passwords on Multiple Sites". [Online]. Available: https://www.cisecurity.org/blog/reusing-passwords-on-multiple-sites/. [Accessed: 31- Jul- 2019].

[20] Cloudflare, "What Is Credential Stuffing?". [Online]. Available: https://www.cloudflare.com/learning/bots/what-is-credential-stuffing/. [Accessed: 31- Jul- 2019]