Santa Clara University

Department of Computer Science and Engineering

Date: June 14, 2023

I HEREBY RECOMMEND THAT THE THESIS PREPARED

UNDER MY SUPERVISION BY

Jaden Ngo, Roshan Sevalia, Sudhish Sewpaul, Tate Musante

ENTITLED

WARN:
Wi-Fi Attack Recognizer and Notifier

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

DocuSigned by:
Behnam Dezfouli
6ED175D6B91A4FF...

Thesis Advisor
Dr. Behnam Dezfouli

N. Ling
N. Ling (Jun 14, 2023 19:08 PDT)

Chairman of Department
Dr. Nam Ling

WARN:
Wi-Fi Attack Recognizer and Notifier

By

Sudhish Sewpaul
Roshan Sevalia
Jaden Ngo
Tate Musante

Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Science
in Computer Science and Engineering
in the School of Engineering at
Santa Clara University,
Santa Clara, California

June 14, 2023

Acknowledgments

Our group would like to thank our advisors and mentors Dr. Behnam Dezfouli and Ted Kummert for all of their help, guidance, and insight throughout this entire process. This project would not have been possible without them.

Finally, we would like to thank our professors and faculty for all of their support during our time at Santa Clara University.

WARN:
Wi-Fi Attack Recognizer and Notifier

Sudhish Sewpaul
Roshan Sevalia
Jaden Ngo
Tate Musante

Department of Computer Science and Engineering
Santa Clara University
Santa Clara, California

June 14, 2023

ABSTRACT

The Wi-Fi Attack Recognizer and Notifier (WARN) aims to establish a wireless network monitoring system for detection of attacks on Wi-Fi IoT devices. Millions of homes use smart home devices, and many of these are for security. However, these devices are vulnerable to attacks that can disable them or compromise their data confidentiality. WARN is capable of detecting two of these attacks: deauthentication and key reinstallation (KRACK). Deauthentication attacks hoax devices into disconnecting themselves from a network, and KRACK manipulates the authentication protocol of Wi-Fi devices to see unencrypted data. WARN uses Raspberry Pis equipped with long range Wi-Fi antennas to detect these attacks, and users can interact with the system via an iPhone app. In building this project, we discovered that many of the top selling smart home devices are vulnerable to Wi-Fi attacks, so we provided a way for people to know when their devices are under attack. While WARN is not a product ready for commercial manufacturing, this project aims to bring awareness to these attacks, and prove that a low cost solution is possible. Our source code and detailed installation and usage instructions for WARN can be found publicly at `https://github.com/SIOTLAB/WARN`.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AES Advanced Encryption Standard

AID Association Identifier

AP Access Point

API Application Programming Interface

BIP Broadcast Integrity Protocol

BLE Bluetooth Low Energy

BPF Berkeley Packet Filter

CCMP Counter Mode Cipher Block Chaining Message Authentication Code Protocol

CRC Cyclic Redundancy Check

DHCP Dynamic Host Configuration Protocol

DoS Denial of Service

GTK Group Temporal Key

GUI Graphical User Interface

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

IP Internet Protocol

KRACK Key Reinstallation Attack

MAC Media Access Control

OSI Open Systems Interconnection

PMF Protected Management Frame

PTK Pairwise Transition Key

SAE Simultaneous Authentication of Equals

TCP Transmission Control Protocol

TKIP Temporal Key Integrity Protocol

UDP User Datagram Protocol

UI User Interface

WAP Wireless Access Point

WARN Wi-Fi Attack Recognizer and Notifier

WEP Wi-Fi Equivalent Privacy

Wi-Fi Wireless Fidelity

WPA Wi-Fi Protected Access

WPAN Wireless Personal Area Network

Chapter 1
Introduction

1.1 Motivation and Background

Security in one's home cannot be understated. Regardless of walk of life, location, or lifestyle, being safe in one's house is essential. While in the past this has been achieved through conventional methods such as key locks and window blinds, the turn of the century has brought huge technological advancements which has completely changed the face of the security industry. With an endless list of technologies and products, smart homes are proving to be the future of home security. A smart home is a home equipped with internet connected devices that enable the control of critical household functions such as lighting, climate, appliances, and security. They offer ease of management and interactive features which have allowed security to keep up the new era of technology. While devices may be spread among a house, and serve different purposes, they are all connected through Wi-Fi communication. Having Wi-Fi communication in a smart home allows all connected devices to communicate and connect to ensure an effective network with strong security. While it offers various benefits and allows an all-in-one solution for household management, it also presents new vulnerabilities for malicious activity. Within the past eight years, the number of smart homes in the United States has over doubled reaching an overwhelming 77 million households [1]. As the market for smart homes has grown exponentially, so too have the number of attacks. There are numerous re-

ported instances and examples of smart home attacks that have become evident as their list of devices and uses expands. Examples include smart thermostats, smart TVs, IoT-enabled doors, and security cameras. In January to June of 2021, there were an estimated 1.5 billion breaches on Internet of Things (IoT) devices, and a weekly average of 12,000 attacks per smart home in 2021 [2]. In addition, around 40.8% of smart homes have at least one device that is susceptible to cyber attacks [3]. Looking into the future, these numbers will only grow making security a central issue in the smart home industry. Therefore, having countermeasures and security solutions for these new assets is necessary to give users dependability and peace of mind.

1.2 The Importance of Wi-Fi in Smart Homes

Wi-Fi stands for Wireless Fidelity, and is most widely used in smart home communication due to its wide range of effectiveness, high transmission speeds, capability to support numerous devices, and convenience; when looking at the necessary communication requirements that smart homes need for their IoT devices, these are all necessary things that must be met. To start, the typical range of a 2.4GHz Wi-Fi router is about 150 feet, making it more than sufficient for most households; when combined with range extenders, this number can become more than adequate [4]. To meet the transmission and bandwidth requirements to support multiple IoT devices, modern routers can also provide extremely high speeds, once again being more than sufficient to support a smart home with numerous personal internet connected devices. Wi-Fi also presents a convenience factor that makes implementing it as one's home communication protocol quite appealing. These networks allow users and devices to connect in a fraction of a second without any configurations,

hard connections, or setup processes. As homes become more complex with IoT devices, it becomes increasingly important to have a simple and reliable underlying protocol.

A few examples of modern products that employ Wi-Fi technology are the Asus RT-AX86U [5], TP-Link Wi-Fi 6 AX3000 [6], and Netgear Nighthawk AX8 [7] which all provide high throughput, security, and performance. The Asus RT-AX86U is a dual band Wi-Fi 6 router that provides users with up to 2Gbps speeds and advanced security protocols [5]. Regarded as one of the best modern routers on the market, it also offers an effective whole-home mesh network. The TP-Link Wi-Fi 6 AX3000 also offers similar specs including a dual-band operating frequency and speeds up to 2402 Mbps [6]. Finally, the Netgear Nighthawk AX8 also provides users with a dual-band frequency system and speeds of up to 6Gbps [7].

Despite Wi-Fi being the primary choice for smart home systems, there are some downfalls and other communication protocols that offer similar functionality. To start, Wi-Fi uses air as a medium to transmit data between devices. While this allows for ease of wireless communication and can allow devices to connect from distant locations, it also presents vulnerabilities as this signal can be interrupted and prove susceptible to security threats. More modern Wi-Fi routers have begun to address this with dual-band operating frequencies (2.4 GHz and 5GHz) to mitigate signal interference. Wi-Fi also requires high power and high bandwidth in order to properly operate. This can present limitations to numerous users who don't have access to the sufficient resources and environments.

As alternatives to Wi-Fi, there are multiple other technologies that present communication protocols, however with differing pros and cons. One alternative technology is Bluetooth Low Energy (BLE). BLE, also referred to as Bluetooth Smart, is the most common communication protocol for IoT devices and offers a

one way communication method that uses advertising and connecting. In a BLE system, the device that advertises a connection is called the Bluetooth Peripheral, while the device searching for a signal is coined the Bluetooth Central device [8]. Some key features and advantages of this technology include extremely low power consumption, low costs, and fast and secure connections. However, there are some large drawbacks to using this technology that make Wi-Fi a more suitable alternative to smart home implementation. To start, BLE's inability to transmit large amounts of data at a high throughput makes it a less preferred network solution to Wi-Fi when incorporating devices that require high bandwidth. In addition, BLE implements different systems for security such as link-layer encryption and device bonding that have yet to be properly implemented into modern day IoT devices. As a result, hackers can easily take control over a network of BLE connected devices which presents serious security risks. A few examples of the types of attacks deployed onto BLE connected devices are passive sniffing, man-in-the-middle, and MAC spoofing [9].

ZigBee is another high-level communication protocol that offers low cost and low power IoT device communication. ZigBee is an IEEE 802.15.4-based specification and is commonly used to create wireless personal area networks (WPAN) that require little bandwidth, low data rates, short range, and small size. In a ZigBee network, devices can expect to transmit between 10-75 meters at a rate between 20-250kbps. A unique feature to ZigBee is its ability for communication redundancy, making single-point-of-failure a thing of the past in mesh networks [10]. However, like many of the drawbacks experienced by BLE, ZigBee possesses some key disadvantages that make Wi-Fi yet again a better alternative to smart home communication. First, Zigbee has a low transmission rate that makes connecting devices that require high bandwidth and high transmission speeds inefficient. Second, ZigBee is

not as secure as Wi-Fi. Due to problems associated with signal interference, ZigBee presents numerous vulnerabilities to potential hackers and malicious activities [11]. Finally, due to its short-range, this protocol limits the flexibility and size of any smart home network users attempt to deploy without many routing-capable devices to create an expanded mesh network.

1.3 Wi-Fi Overview

In this section, we provide an overview of how some properties of Wi-Fi work. This includes information about the physical and MAC layers and the basics of authentication and association when used with WEP, WPA, WPA2, and WPA3.

1.3.1 Physical and MAC Layers

To be able to communicate across the internet, data needs to go through various devices and protocols. Due to the complex nature of this process, the OSI (Open Systems Interconnection) reference model provides us with a way of illustrating the different protocols and independent operations that need to be carried in order to successfully exchange information on a network [12]. The TCP/IP model also defines data exchange on a network and encompasses all of the components of the OSI model differently. A comparison between both models can be found in figure 1.1.

In this section, we are going to focus on the physical and data link layer, which are the lowest levels of the OSI model.

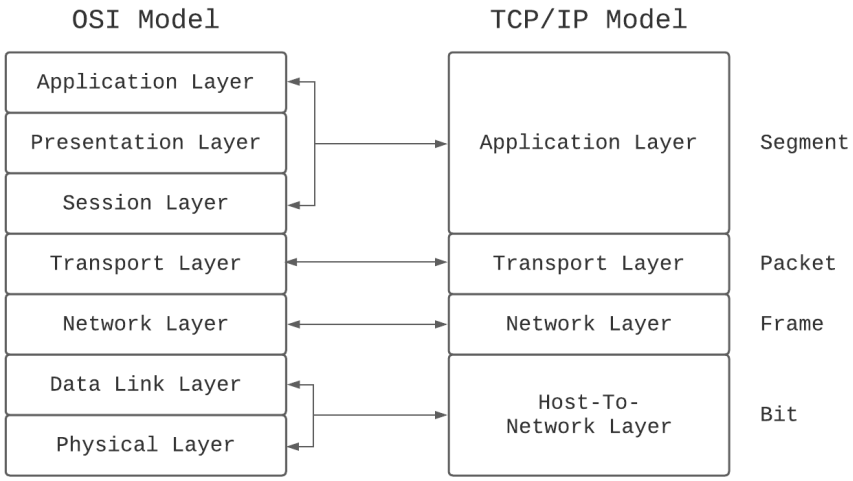

Fig. 1.1: OSI reference model vs TCP/IP reference model

Physical Layer

The physical layer is responsible of controlling the transfer of bits on physical media. Actions, such as defining the rate of transfer of data is performed by this layer [13]. The data being transferred by the physical layer is in the form of frames. Physical layer frames consist of a preamble, a header and a payload. A preamble allows for synchronization of the recipient with the source, the header defines the configuration of the information embedded in the frame, and the payload holds the information that needs to be transferred [14].

MAC Layer

The MAC layer, or data link Layer, is used to manage the transfer of frames from source to receiver, by defining how to send the data frames and the information to be encapsulated within said frame. Alongside this responsibility, the MAC layer also provides means of checking for corruption of data. The data link layer performs a cyclic redundancy check (CRC) and sends the result to the receiver. Upon receipt of the frame, CRC is performed again in order to verify the integrity of the data

received [13]. The MAC layer is an essential component of the network, especially while transferring data, as the layer consists of protocols, such as time division multiple access, carrier sense multiple access with collision avoidance, or polling, which prevent collisions from happening. The protocols define how to handle requests to send frames, thus preventing any conflict to arise between senders on a network [15].

1.3.2 Authentication and Association

Authentication establishes a station as authorized to associate with other stations, and association establishes a mapping between an access point and a station [16].

WEP (Wired Equivalent Privacy) was the first mainstream Wi-Fi security protocol released by IEEE [17]. It was designed to give good enough security while also complying with export laws due to restrictions on encryption technology [17]. It works by using a secret key and randomly generated initialization vector (IV) to seed the RC4 pseudo-random number generator [17]. The generator is then used to generate a data stream that is the same length as the data being sent over Wi-Fi and XORed to encrypt it [17]. The final packet includes the encrypted data and the IV to allow the receiver to decrypt the data [17]. Within a few years after release, however, a vulnerability in the RC4 generator was discovered which combined with the short keys used by WEP, allowed the secret key to be cracked with a few minutes of packet sniffing [18].

WPA (Wi-Fi Protected Access), was a temporary solution to WEP's faults and was quickly replaced by WPA2 [19]. WPA introduced the Temporal Key Integrity Protocol (TKIP) which used RC4 [19]. WPA2 implements new security mechanisms outlined in IEEE 802.11i, which deprecated WEP. Specifically, WPA2 added support for CCMP, which is based on the advanced encryption standard (AES) [19].

The WPA2 authentication process is called the 4-way handshake, where a sequence of four messages are sent between a client and an access point in order to generate and install encryption keys. Unfortunately, a security flaw in this process was discovered. It is known as the key reinstallation attack, or KRACK.

WPA3 was released in 2018 by the Wi-Fi Alliance and replaced WPA2 [20]. It requires the support of Simultaneous Authentication of Equals (SAE), and Protected Management Frames (PMFs) [20]. SAE makes use of a different authentication protocol, known as the Dragonfly handshake, so it is not vulnerable to the key reinstallation attack. It also offers other security properties such as pairwise link confidentiality and integrity protection [20]. PMFs help protect against deauthentication attacks. A station can refuse deauthentication when a PMF fails an integrity check [21]. PMFs are implemented with CCMP and BIP [21]. WPA3 is not yet widely adopted, especially among smaller IoT devices. In mid-2021, a database of access points reported only 16,000 access points using WPA3, compared to 561 million using WPA2 [22].

Authentication and association differ slightly between WPA2 and WPA3. Both involve 5 steps. In the first step, a station probes for compatible access points [23]. In the next step, in WPA2, both the station and the access point share MAC addresses and generate a Pairwise Master Key [23]. In WPA3, this step occurs differently. WPA3 requires an SAE handshake, which is done in this step [23]. The third step assigns an association ID (AID) to the station [23]. Finally, a 4 way handshake is done that generates Pairwise Transition Keys (PTK) and a Group Temporal Key (GTK) [23]. Lastly, DHCP is used to assign an IP address to the station if needed [23].

1.4 High-level Solution and Expected Results

We designed and developed a system to detect and alert the user of disconnection attacks on a smart home network. A network of single board computers (Raspberry Pis) monitors the traffic over a small smart home network consisting of various IoT devices used for testing purposes. We have conducted disconnection attacks on this testbed, and suspicious activity is sent to a local server. If this server detects an incoming attack, it will notify the user via a mobile app. In a real smart home, this notification will alert users of home invasions or burglaries that use disconnection attacks as they are happening.

This system is a practical and cost effective way to improve the security of existing smart homes. Rather than redesigning entire protocols or devices, this solution has the capacity to serve the millions of smart homes that already exist without any new infrastructure needed. With a successful solution, we can expect to see a dynamic model that will be able to alert users of disconnection attacks on various smart home products.

Chapter 2

Attacks on Wi-Fi IoT Devices

In this chapter, we detail the existing types of attacks on Wi-Fi IoT devices that smart home networks are vulnerable to. They include the deauthentication, disassociation, and key reinstallation attack (KRACK).

2.1 Deauthentication Attack

WPA/WPA2 networks use management frames to establish and maintain connections. As defined by Cisco, "management frames are broadcast frames used by IEEE 802.11 to permit a wireless client to negotiate with a Wireless Access Point (WAP)" [24]. There are many different types of management frames such as beacon, probe, authentication, deauthentication, association, disassociation, and action. Each provides different functionalities and purposes, and all run behind the scenes allowing devices to connect and maintain network connections. They are critical for the operation of wireless networks and without them, Wi-Fi communication among IoT devices in WPA/WPA2 networks would be impossible. However, while these frames are essential, they are not encrypted and present serious vulnerabilities to network users and devices.

The deauthentication attack is a special type of Denial of Service (DoS) attack that exploits these vulnerabilities, attacking the deauthentication management frame specifically. This attack works when an attacker imitates a client or access

point and sends a deauthentication frame through the MAC header to call for connection termination. In a MAC header, a management frame is specified by two leading zeros and the deauthentication code is included in the reason code with a binary sequence of *1100* [25]. If properly sent and received, the receiving entity will accept this management frame as authentic and close the connection; if the victim devices wish to reconnect, they must go through the authentication process once more. Since these frames are not encrypted, they can be easily spoofed making this type of attack particularly dangerous and easy to implement [26].

Looking at how this attack is implemented in real life, there are various different scenarios when the deauthentication attack can prove quite dangerous. One of the most alarming instances of this attack is the disconnecting of security cameras. Security cameras in one's smart home network is a central way in ensuring that intruders stay clear of malicious activity, and get caught if they don't. Without these devices properly running in a home network, the ability to detect and capture intruders significantly decreases. Another common, yet dangerous, attack is the disabling of smart lights. Much like the security camera vulnerability, if attackers are able to minimize visibility, it becomes increasingly difficult to protect and secure one's home. While these are a couple of dangerous examples, there are many others including turning off thermostats, disabling smart TVs, and IoT-enabled locks.

Table 2.1 lists various IoT devices, and their vulnerability to deauthentication attacks. The column labeled “Minimum Packets/Sec” refers to the minimum number of deauthentication frames sent per second to disable the device. This number was found by continually increasing the frequency of the packets until a device disconnected. Some devices did not attempt to reconnect within 15 minutes, so their reconnect times are marked as ”Unknown.”

Device	Vulnerable?	Minimum Packets/sec	Reconnect Time
Ring Camera	Yes	300	1 minute - Unknown
Nest Camera	Yes	100	1 minute
August Smart Lock	Yes	10	1 minute - 5 minutes
Nest Protect	Yes	20	Unknown

Table 2.1: Vulnerability of IoT devices to deauthentication attacks

2.2 Disassociation Attack

Similar to the deauthentication attack, the disassociation attack also takes advantage of management frames under IEEE 802.11 by spoofing the disassociation frame in order to terminate existing associations. In normal situations, devices disassociate when they leave networks or have changed associations to other networks. Additionally, if given access points cannot support a high number of clients or are restarting their systems, the disassociation frame is broadcasted to the entire network—since disassociation is sent as a notification (not a request) this cannot be refused by receiving stations. Attackers can take advantage of this functionality by imitating clients or access points and send disassociation frames through the MAC header to close connections. In a MAC header, the disassociation code is specified with a binary sequence of *1010*. Once a client is dissociated from a network, it must re-initiate the set-up process in order to rejoin. If an attacker is able to send a continuous stream of these frames, a DoS attack will be caused. Finally, since disassociation frames are also neither encrypted or authenticated, this attack is easy to implement and presents yet more vulnerabilities to WPA/WPA2 devices [27]. The feasibility of this attack is explained by John Bellardo and Stefan Savage [28] and can be executed with tools such as Airjack [29] and KisMAC [30]. Figure 2.1 visualizes how this attack is performed. First, a clients MAC address is sniffed. Then, the spoofed disassociation frame is sent.

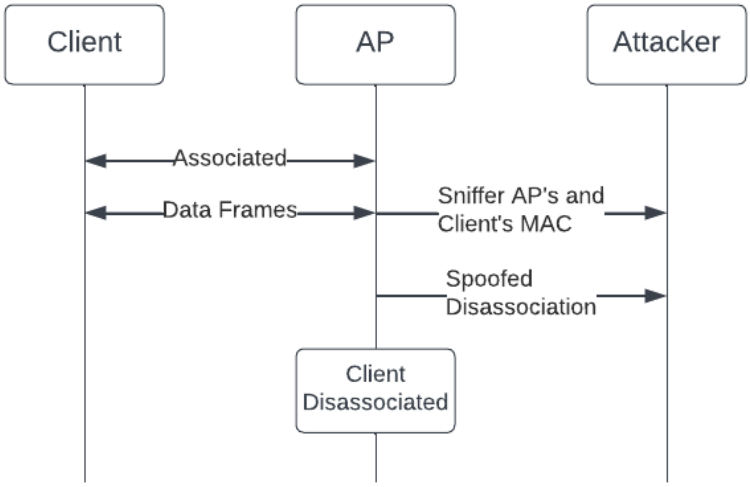

Fig. 2.1: Depiction of disassociation attack

However, while this attack is efficient, it is not as effective as the deauthentication attack. This is because during the association and authentication process between clients and access points, authentication takes place first. Therefore, while the disassociation of devices does interfere with IoT device communication, devices only have to reassociate in order to restore a connection, while deauthenticated devices must both re-authenticate and reassociate.

2.3 Key Reinstallation Attack (KRACK)

The key reinstallation attack exploits design flaws to cause cryptographic protocols to reinstall an encryption key that has already been used [31]. While using CCMP, this allows an attacker to decrypt messages sent from the client to the AP [31]. KRACK is particularly powerful against IoT devices because of insufficient layers of encryption due resource constraints [32].

The key reinstallation attack (visualized by figure 2.2) begins by establishing a

channel-based man in the middle attack between an AP and a client to manipulate the 4-way handshake [31]. The attacker then prevents message 4 from reaching the AP (1). Because message 4 is simply a confirmation, the victim installs its PTK normally (2). When the authenticator does not receive message 4, it assumes either message 3 or 4 was lost in transit, so it retransmits message 3 (3). When the client receives message 3 again, it reinstalls the same PTK as it had before (4). This also resets the nonce and replay counter. The next transmitted frame by the client will reuse the nonce. Once this happens, the attacker can decrypt packets sent from the client to the AP. This explanation is a simplification of the process. There are many different versions of the attack that vary to account for the way different devices conduct the 4-way handshake. However, they all rely on this flaw: the keys are installed immediately after the final confirmation is sent and they can be prompted to reinstall these keys.

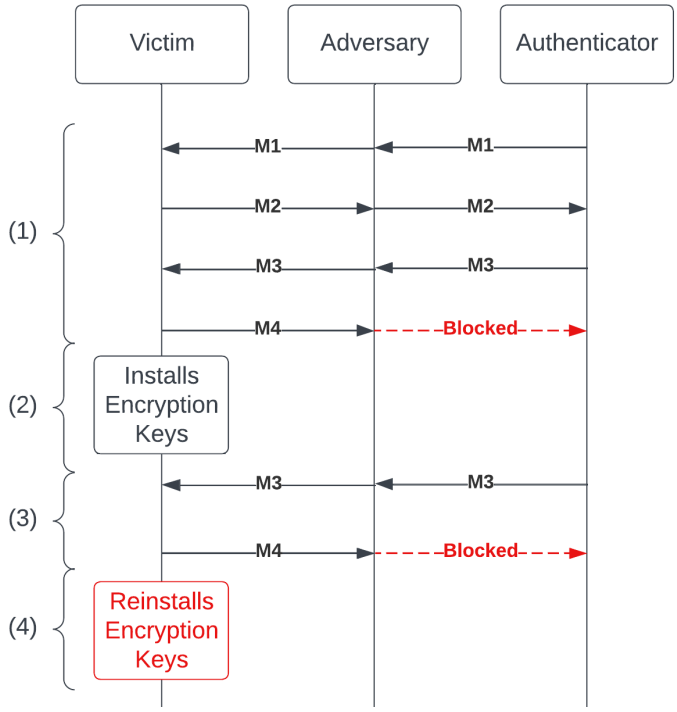

Fig. 2.2: Visualization of key reinstallation attack

The implications of KRACK are huge. When it was disovered in 2017, many im-

plementations were vulnerable including OS X, macOS Sierra, Android, wpa_supplicant, OpenBSD, and others [31]. Authenticators that use CCMP (used in WPA2) were found to be vulnerable to replaying and decrypting frames [31]. Ones that used the older TKIP (used in WPA) were vulnerable to replaying, decrypting and forging of frames [31]. There are even variations on KRACK that target the Fast BSS Transition (FT) handshake and the group key handshake [31]. Luckily, some patches have been made. That being said, older devices that have not been patched are still vulnerable. The IoT devices used in smart homes are particularly vulnerable because they are updated less frequently. Our solution targets older systems that are still vulnerable to this attack.

Chapter 3
Related Work

3.1 Current Solutions

There are a few different solutions that currently exist to protect smart home IoT devices. One approach [33,34] used by Armis Security and Palo Alto Networks is to use a machine learning model to analyze device connection behavior and monitor for attacks. This software is a solution designed for enterprise where it can integrate with existing IT and firewall workflows, but in the home, there isn't enough capability in the networking hardware to support a feature as advanced as this. This type of solution is gaining popularity fast in the enterprise space with companies like Amazon Web Services [35] and Microsoft [36] developing their own versions of it as well.

Another solution is the Netgear Armor [37], born from the collaboration between Netgear and Bitdefender. This approach is similar to the previous, but instead of the detection being handled by a cloud service, it takes place on the router. This approach is targeted towards individuals, since it consists of a standalone piece of software that can run on a residential-class router. But, it requires a hardware replacement which proves unappealing to most consumers.

Another existing solution we researched was the AWS IoT for Connected Homes [38]. AWS IoT is a platform for companies to build smart home devices on. AWS IoT can use machine learning to detect anomalies in the way devices behave [38].

This solution is consumer friendly since manufacturers, not consumers, would be securing their devices. Therefore, it can remain completely invisible to the consumer. This is, however, also a disadvantage due to every device manufacturer needing to implement and potentially update existing devices. Devices that are old and out of date would be left behind. This makes it a good platform for the future, but not an immediate solution to this growing problem.

Ideally, the best long term solution is updating the Wi-Fi protocol itself, and the newer WPA3 protocol attempts to mitigate some of these issues. Two of its new technologies, PMFs and SAE, help mitigate deauthentication and KRACK attacks. PMFs protect the management frames from being spoofed which mitigates the deauthentication attack [21]. SAE is a more advanced authentication system that is designed to block the attack route used by the key reinstallation attack for WPA2. Unfortunately, WPA3 is not yet widely adopted, and it may take many years for the majority of devices to implement the new standard. Additionally, flaws have been spotted in WPA3 as well. The discoverer of the key reinstallation attack also showed that WPA3 is vulnerable to DoS and downgrade attacks [39]. Even with a perfect Wi-Fi protocol, wireless devices will always be vulnerable to jamming attacks. These attacks bombard the communication channel with information and make it impossible for devices to understand each other. These attacks can be easily detected, however, and they can be reported via a wired connection.

Based on this, our goal is to create a solution that requires new hardware but will not replace any existing devices like routers. This will make it easier to deploy in existing systems while also addressing the lack of capable hardware to run this type of device security system in homes today. Another goal of ours is to create a system that can serve older systems that are much more vulnerable to the attacks described previously. While some solutions presented in this chapter will work in

the future, none of them offer an immediately deployable solution without updating the existing system.

Chapter 4

System Architecture

This chapter lists the requirements of our system, as well as details our system architecture and implementation. Our design will include four main components: packet sniffers, a local server, a remote server, and a user app. Raspberry Pi sniffers will be placed throughout our test bed to monitor the network for deauthentication and KRACK attacks. A Berkeley packet filter will be used to filter for frames sent to and from each IoT device. The number of suspicious frames will be sent to a local server that will conduct analysis and notify the remote server when an attack has started and when it ends. The remote server will host a database that includes a list of devices and their statuses, a log of previous attacks, a list of users, and the status of the entire network. The remote server will also manage push notifications to the mobile app. The mobile app will communicate with the remote server to register new devices, view the status of devices, view the network status, view previous attacks, and notify the user of attacks when they happen. Figure 4.1 represents the conceptual model of our design. The components outlined in green are the components of our detection and notification system. The remaining components make up our testbed.

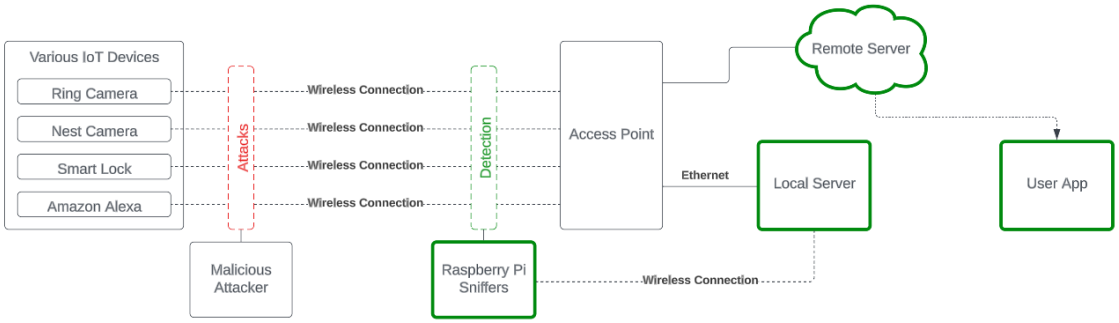

Fig. 4.1: Conceptual model diagram

Table 4.1 is a list of functional and non-functional requirements for our project. These requirements were chosen at the beginning of our design process, and helped guide our decision making process. The functional requirements determine our system behavior, while the non-functional requirements outline our system architecture.

Req. ID	**Functional Requirements**
FR1	Real time deauthentication attack detection
FR2	Real time KRACK detection
FR3	Remote notifications
FR4	Log of previous attacks
FR5	Display current network status
Req. ID	**Non-Functional Requirements**
NFR1	Register and name smart home devices
NFR2	No changes to existing network hardware or software
NFR3	Mobile app to view network and notify of attacks
NFR4	GUI that follows UI design principles
NFR5	Remote server to store attack log and manage push notifications
NFR6	Network of packet sniffers to monitor for attacks
NFR7	Local server to aggregate sniffer data and analyze attacks
NFR8	Simple ”Plug and Play” setup

Table 4.1: Functional and non-functional requirements

4.1 Methods of Detection

Deauthentication attacks require many fraudulent deauthentication frames to be sent in a short period of time. Thus, we will detect these attacks by monitoring for many deauthentication frames within a short period of time. Specifically, more than ten deauthentication frames within ten seconds is indicative of a deauthentication attack. KRACK attacks begin by initializing a man in the middle position between a device and an access point. The attacker then manipulates the 4-way handshake used to authenticate new devices on a network. Specifically, the attacker withholds the fourth message message from reaching the authenticator. This prompts retransmissions of the third message by a device. This may happen multiple times in order to cause the device to reinstall keys it has already used. We can detect KRACK by listening for multiple instances of the third and fourth message in the 4-way handshake that don't have corresponding first and second messages. That way benign handshakes won't be flagged.

4.2 Packet Sniffers

Each sniffer is a Raspberry Pi equipped with an external Alfa AC1900 Wi-Fi adapter. Each one runs a single Python program, which is responsible for setting up communication between itself and the local server via a UDP client/server relationship, periodically updating the list of devices it needs to recognize attacks on, and, of course, sniffing packets and sending information about these packets to the local server. Table 4.2 summarizes the technologies used for the packet sniffers.

Each message sent to the local server has 4 fields: Pi ID, Attack Type, Time, and Counters. A Pi ID of -1 indicates to the local server that a Pi is requesting

Type	Technology
Computer	Raspberry Pi model 4B 4GB
Operating System	Kali Linux
Programming Language	Python
Sniffing Program	PyShark
Wi-Fi Dongle	Alfa AC1900 Wi-Fi Adapter

Table 4.2: Technologies for packet sniffers

the list of devices to listen for. This happens at the beginning of the program and every 60 seconds afterward. If it is found that a sniffer's device list is outdated, the entire program is restarted. This is necessary because it needs to start capturing with a new BPF. Once the correct device list has been found, the sniffer can begin sniffing.

There are two filters that the sniffer uses to speed up the sniffing process tremendously. These are a Berkeley Packet Filter (BPF), and a display filter. BPFs are much faster than display filters because they occur in kernel space. However, there are limitations to BPFs. For this project, we can only use a BPF to filter for the source or destination addresses of wireless frames. The rest of the filtering is done by a display filter. This type of filter simply restricts what frames are seen by the user. We use a display filter to filter for frames specific to the attacks we are looking for. In our project, deauthentication frames indicate deauthentication attacks, and 4-way handshake messages can indicate KRACK attacks.

After the filters are in place, the sniffing process is simple. Each frame that passes through all filters is screened for each attack, and the sniffer keeps count of the number of suspicious frames detected for each attack, for every possible victim. Note that every deauthentication frame is counted, but not every 4-way handshake message is. Only ”rogue” M3s and M4s are counted (those without corresponding M1s and M2s).

Every ten seconds after sniffing has started, these counters are sent to the local server, which uses its own logic to determine if an attack has taken place. The counters are then reset to zero.

4.3 Local Server

The local server is responsible for managing all the devices on the network and keeping track of their current state. This information is communicated to the remote server, which can then perform the adequate action. Aside of device management, it is also responsible for managing all the sniffers on the network. It is the common entity that ties in all of the data collected by the sniffers and runs the analysis required to determine the current state of the network. Additionally, it allows us to enhance security by isolating our sniffer network from the main network. Table 4.3 summarizes the technologies used for the local server.

Type	**Technology**
Computer	Raspberry Pi model 4B 4GB
Operating System	Kali Linux
Programming Language	Python
Device Detection	Nmap

Table 4.3: Technologies for local server

4.3.1 Functionality

To manage all the sniffers and improve security, the local server generates its own access point, using the PyAccessPoint library, for the sniffers connect to. Once connected to the sniffers, the local server opens a UDP socket to allow for the exchange of information between each sniffer and the local server. Upon receipt of datagrams from the sniffers, the local server either sends an updated list of devices

to the each sniffer, or decodes and performs analysis on the sniffer counters. If an attack is detected, the local server sends a POST request to the device endpoint of the remote server and performs a disconnection check to determine if the device has gone offline.

The local server is also responsible for monitoring devices connected on the network and their connection status. To achieve such monitoring, a list of devices to be monitored is obtained from the remote server. Then, making use of Nmap and the list received, the network is periodically scanned. If a device is not detected for a period of 10 minutes, the connectivity status of the device is updated to be offline. If in the process new devices are discovered, their information is relayed to the remote server so as to be available within the app.

4.3.2 Analysis

When analyzing deauthentication counters, if more than ten deauthentication packets are observed within ten seconds an attack might have occurred. From testing we have determined the minimum number of packets per seconds(pps) needed to take a device offline. Based on this number, we can determine whether the reading we have is one that warrants a report as a warning or an attack.

As for KRACK, if four retransmissions of rogue M3s and M4s are observed on the network, the local server reports an attack to the remote server.

4.4 Remote Server

The remote server will be the authoritative storage for attack information, and it will deliver notifications to the user app when an attack occurs. The design of the remote

server is a REST HTTP server that acts as a front end to a PostgreSQL database. The database holds 7 tables: One to store user information (Users), one to store households (Accounts), one that stores device details and current status (Devices), one that stores devices that aren't being monitored yet (UnknownDevices), one that stores attack history for a given device (History), one that stores our pre-existing test info for a particular type of device (DeviceInfo), and one that stores IDs for push notifications (ApnsID). Table 4.4 summarizes the technologies used for the remote server.

Type	**Technology**
Virtualization Software	Docker
Programming Language	Rust
Database	PostgreSQL
Database ORM Library	Diesel
Push Notification Service	Apple Push Notification Service

Table 4.4: Technologies for remote server

4.4.1 Functionality

The remote server provides almost all of the data needed by the user app. It provides the app with account authentication, device attack statuses, and attack/vulnerability history. Most of the endpoints are checked wrappers for getting information from the database such as finding a device entry using it's MAC address. If an attack is detected by the local server, the update sent to the remote server on the device status will also trigger a request to the Apple Push Notification Service to notify the user who owns the device. To get information about what thresholds constitute an attack on a device, a device when added to the system will be crosschecked against our pre-existing test data using the mac address manufacturer code. From there the user will be able to select which device it is and the local server can pull the info it needs to set up the correct thresholds. This will help minimize false positives,

since our testing has observed large differences in the tolerances each device has to certain attack types. Table 4.5 depicts the fields for each table in the database.

Accounts	Users	ApnsID	DeviceInfo	Devices	UnknownDevices	History
account_id	user_id	apns_id	manf_name	device_id	device_id	history_id
	account_id	user_id	device_name	device_name	user_id	user_id
			pps	user_id	device_name	timestamp
				connection_status	device_vendor	attack_type
				severity	timestamp	severity
				info_manf		device_address
				info_name		

Table 4.5: Remote server database tables

4.5 User App

To create our user app, we used the SwiftUI high-level programming language. Created in 2019, SwiftUI is Apple's framework for creating user interfaces in all of Apple's relevant operating systems. For our purposes, the WARN app is an iOS app and therefore is also only currently usable on iOS devices. To notify our users of all necessary attack, network, and device info, we also implement Apple's push notification service. Table 4.6 summarizes the technologies used for the user app.

Type	**Technology**
Programming Language	SwiftUI
Push Notification Service	Apple Push Notification Service

Table 4.6: Technologies for application

4.5.1 UI Overview

The mobile app we created will be the only way that users can interact with our WARN surveillance technology. Our app has a total of three tabs (Devices, Network, Attacks) which all serve a key role in providing our users with a one stop solution to smart home network security. The following three sections will provide a brief

overview of our UI and user interaction capabilities. Figure 4.2 is the icon used for our mobile app.

Fig. 4.2: Mobile app icon

The first tab, "Devices," serves as an overview page for users to see the devices they are tracking on their smart home network. As visible in figure 4.3, users are able to view device connection status and a list of other potential devices they wish to add to their WARN surveillance system. This page also serves as the landing page for users upon opening the app.

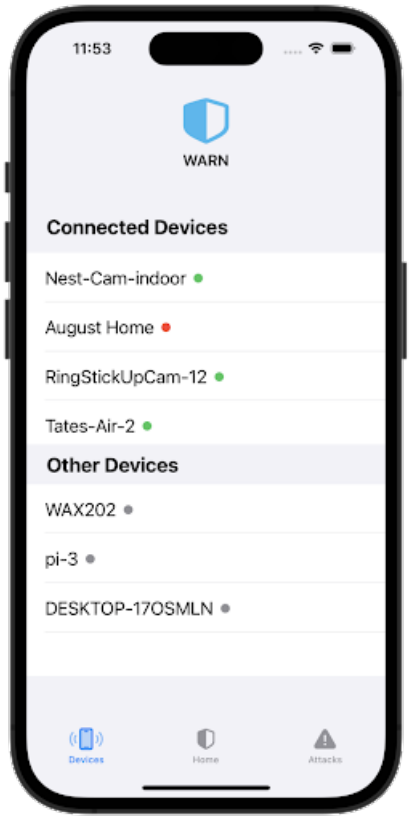

Fig. 4.3: Mobile app devices tab

For each device, a status is given based on network connection status: Connected, Vulnerable, and Disconnected. The "Connected" status indicates that the registered device is connected, and no attacks are being detected. The "Vulnerable" state indicates that the given device is connected, but our WARN surveillance detected suspicious network activity. Finally, the "Disconnected" state indicates that the device is no longer connected to the smart home network, and that an attack might have caused it - whether there was an attack or not is indicated by our push notification system and "Attacks" tab.

The center tab, depicted in figure 4.4, offers our users with three main features: network connection status, add device, remove device. To start, users will be able to view their network name and connection status indicated by the top right icon.

This serves as quick overview of one's smart home network and provides an added level of network understanding to our users. Secondly, users can add devices to track on their WARN surveillance system. This processes is seamless, intuitive, and allows users to track a wider range of devices on their network. Finally, if a user wishes to stop tracking a device, they can simply remove it using the "Remove Device" button. All of these functionalities run in real time and are immediately reflected in all other pages.

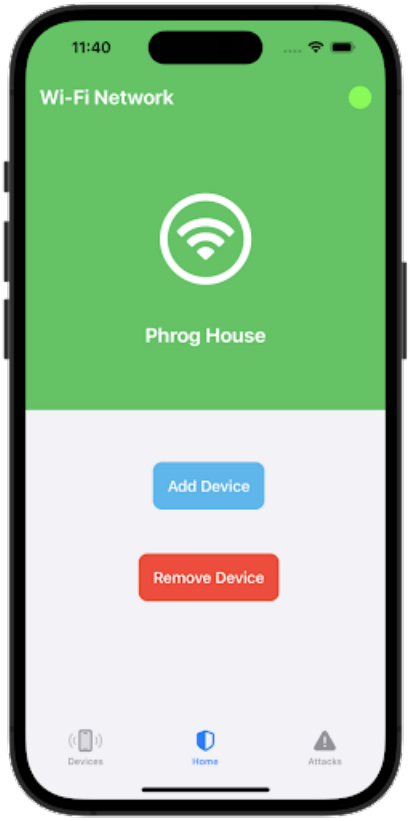

Fig. 4.4: Mobile app network tab

The right tab, "Attacks," (depicted in figure 4.5) displays attacks that have taken place on the smart home network. Users are able to view attack types, attack timestamps, affected devices, and whether those devices are still online. This interface serves as a great place if users want a complete history of attacks and

vulnerabilities on their network.

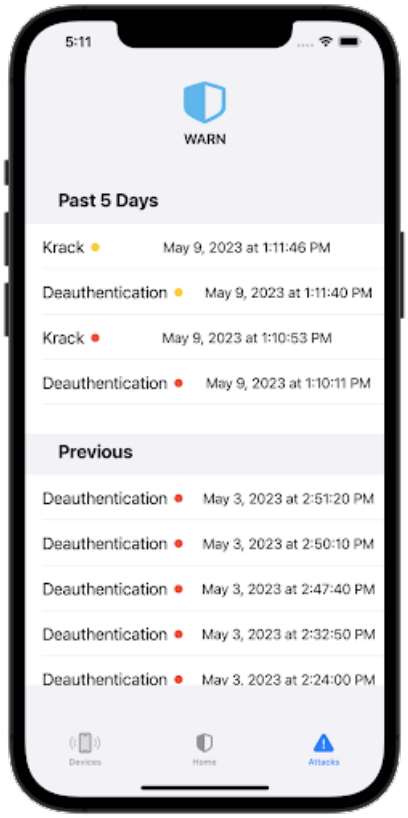

Fig. 4.5: Mobile app attacks tab

4.5.2 Communication with the Remote Server

In order for our app to show live and real data to our users, we use a series of API calls to our remote server. To update and display our users' devices, device statuses, network status, and attack history, conducting these API calls to their relevant endpoints in the remote server is key.

The object that allows these API calls to take place inside our app is called URLSession. This class inside SwiftUI provides programmers with an API for downloading/uploading data to and from endpoints from a URL specified address. There are three different types of URLSessions: default, ephemeral, and background. The

default type behaves like a shared session and for the purposes of our app, is the option we chose to go with. However, ephemeral and background sessions allow for additional functionality such as not writing caches and performing background uploads respectively. In all cases, the underlying functionality of URLSession was key in our app's integration with our remote server - and subsequently local server.

4.6 Design Rationale

As we discussed previously, existing solutions that are currently commercialized either require expensive hardware replacements or are solutions targeted towards manufacturers developing the next wave of IoT devices. There is currently no low-cost easily scalable solution that can be immediately deployed to a house. Our project provides a solution that can function in any home with a simple hardware addition. Moreover, our solution accounts for older IoT devices which are more vulnerable to attacks, something that the existing solutions mentioned in this document fail to account for. Our design was created to have a high performance-to-cost ratio. This design choice inspired us to make use of easily available material to build out our solution, hence the use of Raspberry Pis and easily available antennas for our product. Our goal is to make it as easy as possible to use our product without needing a deep understanding of the subject matter. This is why we chose to make a mobile app as the main way of relaying critical information to a user. Furthermore, the way our infrastructure is structured gives the opportunity to the consumer to add or remove sniffers.

Chapter 5

Evaluation

In this chapter, we will showcase the various tests we have run on our system. Apart from the testing of individual components (e.g., sniffers and local server), we wanted to run tests on the entire system to measure the accuracy of our detection methods. We have performed tests to asses the accuracy and speed of our detection. In other words, how quickly can we alert the users of attacks?

5.1 Deauthentication Attack Testing

The main independent variable we chose to measure the speed and accuracy of our deauthentication attack detection was distance, as this is the main obstacle a real life attacker would have to overcome. Attackers need to be able to perform attacks close enough to be effective, but remain unseen. We designed this experiment to simulate different distances attackers might have to attack from. We performed 6 deauthentication attacks each from 10 to 70 yards, in 10 yard intervals. We performed the attacks on an Alienware m17 laptop running Kali Linux using the internal Wi-Fi card (Intel Wi-Fi 6 AX200). The attack software used was the Aireplay-ng module of the Aircrack-ng suite.

As seen in figure 5.1, we detected 100% of attacks performed within 50 yards. Attacks beyond 50 yards are much more difficult to perform, as these distances are reaching the limits of Wi-Fi technology. Wi-Fi speeds at longer distances are also

severely reduced, making attacks on more resilient devices less effective. That being said, we detected two thirds of attacks at 60 yards, and no attacks at 70. This drop in accuracy is best described by the limitations of our hardware, not the algorithms used for detection.

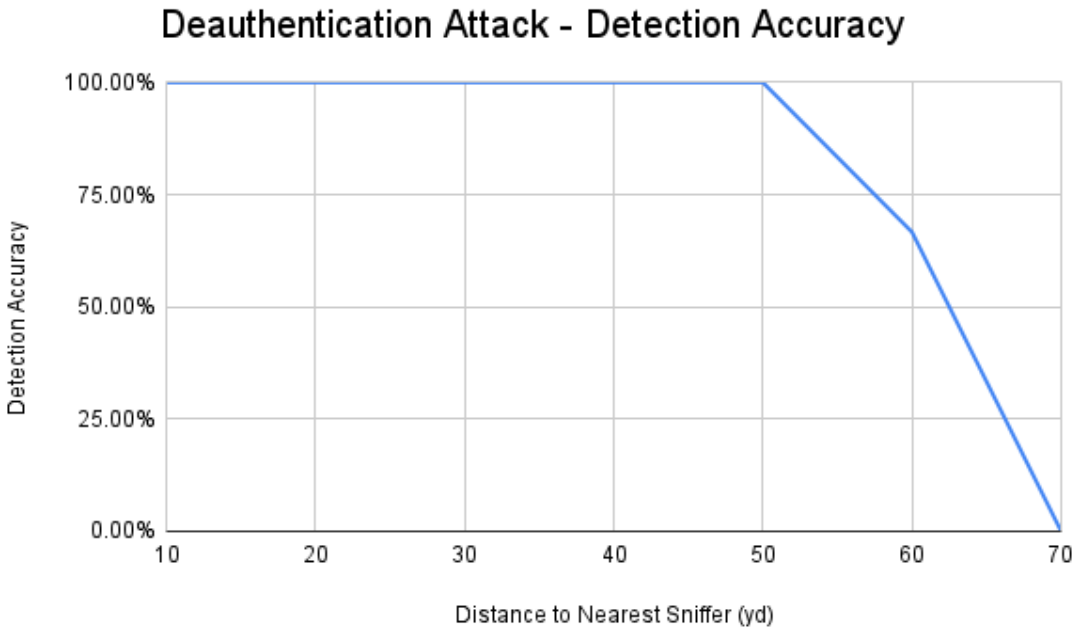

Fig. 5.1: Detection accuracy of deauthentication attacks

Figure 5.2 shows that the time to detect attacks increased as we continued further away from the nearest sniffer. Up until about 30 yards, the time to attack hovered around 10 seconds, with the attacks at 60 yards taking an average of 25 seconds to detect. None of the attacks at 70 yards were successful, and so that data point has been left out. We were very satisfied with the speed of our detection. Notification within 30 seconds would still allow users to take action as an attack is happening.

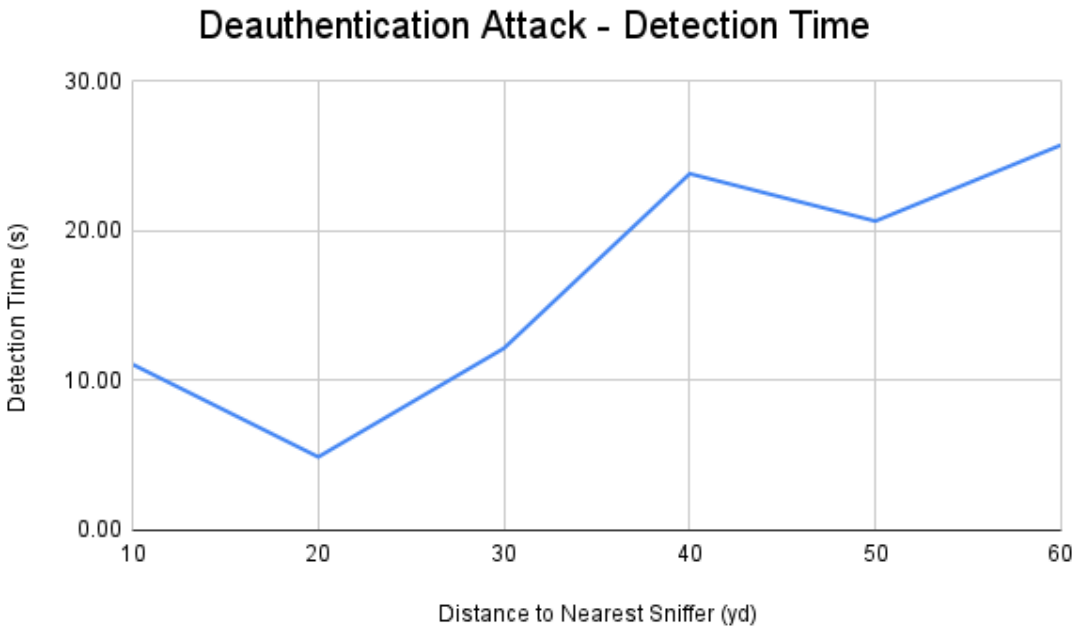

Fig. 5.2: Detection time of deauthentication attacks

5.2 KRACK Testing

Because of the harmful nature of KRACK, we decided not to implement the entire attack. Instead, we tested our KRACK detection by having our sniffers read previously recorded PCAP files that simulated an attack. KRACK's discoverer, Mathy Vanhoef, created software to test whether certain devices are vulnerable to the attack. This software attempts to manipulate the 4-way handshake just like a real attack would, but without malicious affects. We recorded the network activity during this process, and used those recordings to simulate attacks. These simulations helped us create and fine-tune our detection process.

Chapter 6

Future Work

In this chapter, we will further discuss some of the limitations of our project proposed in the previous section, and introduce possible improvements to be made in future work. Most of the improvements suggested in this section detail what work would need to be completed to manufacture and sell our product on a commercial scale.

6.1 More Attack Types

One of the most obvious improvements to be made is added support for detecting more types of attacks. There are many other types of Wi-Fi attacks that are capable of disabling IoT devices, and even more that have yet to be discovered. Adding support for these other attacks will make our system a much more robust and fully fledged security system. Luckily, our design of this system has prepared for this. Because the packet sniffing architecture is already in place, implementing new attack detection would just require the packet filters to be altered, and the detection logic to be put in place. Obviously, an update to the front end (User App) would be required as well.

6.2 Attack Mitigation

Another clear drawback of our system is that it lacks attack mitigation. Right now, we can detect different kinds of attacks, but no action is done to mitigate the effects of attacks as they are happening. The task of attack mitigation is much more difficult than that of detection, and it is not immediately clear how this would be done or what kind of infrastructure is required for such a task. Additionally, these requirements may be vastly different for each type of attack. Although this would be difficult, a system that can mitigate attacks is much more valuable than one that simply detects them.

6.3 Cost

Cost is another drawback from our system. We chose to use Raspberry Pis for our sniffers and local server. While Raspberry Pis are cost effective, even more specialized and lightweight hardware exists that could have been used. Further research and testing is required to determine the exact computational and power requirements of our system, and what hardware could be used that meets these requirements at the lowest cost. In a commercial setting, this would be one of the most important changes to be made.

Another way to reduce the cost of the system would be to merge the local server and sniffer components of our system. This change would be an optional choice for users with physically small networks. Currently, each sniffer sends data to the local server which does most of the computation. This system works well with large networks, where one sniffer does not provide enough range to protect all devices. However, in a small network, only one sniffer is needed. It doesn't make sense to

use two separate pieces of hardware (one sniffer and one local server) when one can do both jobs. This would reduce the cost for users, as they would only have to buy one piece of hardware.

6.4 Physical Design

As computer engineers, we have mainly considered the software and networking design of our system, and not the physical design. Right now, each sniffer is a Raspberry Pi with a power cord and an external Wi-Fi antenna connected via a USB cable. This is no way to manufacture and ship a commercial product. After more specialized and lower cost hardware has been chosen, a chassis would be designed to keep all components neatly packaged.

6.5 Interfaces

Right now, the only way a user can interact with our system is through our iOS app. While this is fine for the scope of our project, if we want WARN to be as accessible as possible, we should look to add as many interfaces as possible. A few ideas we had were an Android app and an online web interface. This would enable our solution to reach a wider range of users and be more usable product.

Chapter 7
Societal Impact

This chapter will address some of the ethical considerations taken while designing this project, and social impact this project aims to have.

7.1 Ethical

This project will provide users with assurance that their IoT devices (specifically ones designed for security), are working as intended. In this document, we have detailed both deauthentication attacks and KRACK. This is done purely for educational purposes, and to provide motivation for our project.

7.2 Social

Our project aims to increase social awareness of the vulnerabilities that smart home devices face, and hopefully put cybersecurity into the minds of nontechnical people. Many of these devices are advertised as full proof security solutions, when, in fact, many are susceptible to attacks, as we have shown. We are often told that the internet is secure, that your banking information is encrypted, and messages you send are truly private. The reality is that there will always be ways to hack the systems in place. Our project brings this to light.

Chapter 8
Conclusion

In this thesis, we outlined the current landscape of smart home networks and IoT devices, gave an overview of Wi-Fi security and its numerous existing vulnerabilities, examined existing solutions in this space (to which we found was quite limited), offered our own solution, and dived into its accuracy, impact, and future potential.

We designed, developed, and tested a product capable of accurately detecting and notifying users of malicious Wi-Fi attacks on their smart home networks in short amounts of time. As shown in the evaluation section of this thesis, its clear that our solution was successful. We hope that this project will not only inform people of the existential threat on smart home networks (and Wi-Fi networks in general), but also inspire future senior design groups at Santa Clara University to carry on this work. Our source code and detailed installation and usage instructions for WARN can be found publicly at `https://github.com/SIOTLAB/WARN`.

Bibliography

[1] Statista Research Department. (2021) U.S.: Number of Smart Homes 2017-2025. [Online]. Available: https://www.statista.com/forecasts/887611/number-of-smart-homes-in-the-smart-home-market-in-the-united-states

[2] Intersog. (2021) IoT Security Statistics: 6 Facts. [Online]. Available: https://intersog.com/blog/iot-security-statistics/

[3] Smiljanic Stasha. (2022) An In-Depth View Into Smart Home Statistics. [Online]. Available: https://policyadvice.net/insurance/insights/smart-home-statistics/

[4] GIG Internet. (2022) How Far Will Your Wi-Fi Signal Reach? [Online]. Available: https://epb.com/get-connected/gig-internet/how-far-will-your-wi-fi-signal-reach/#:~:text=Wi%2DFi%20signals%20will%20usually,for%20a%202.4Ghz%20frequency

[5] Asus. (2022) RT-AX86 Series(RT-AX86U/RT-AX86S). [Online]. Available: https://www.asus.com/Networking-IoT-Servers/WiFi-Routers/ASUS-Gaming-Routers/RT-AX86U/

[6] TP-Link. (2022) Archer AX3000. [Online]. Available: https://www.tp-link.com/us/home-networking/wifi-router/archer-ax3000/

[7] Netgear. (2022) Nighthawk 8 - Stream Dual-Band WiFi 6 Router 6Gbps. [Online]. Available: https://www.netgear.com/home/wifi/routers/rax80/

[8] BLEMobileApps. (2022) Bluetooth Low Energy (BLE) Technology. [Online]. Available: https://www.blemobileapps.com/ble-mobile/

[9] M. S. H. E. H. Arup Barua, Md Abdullah Al Alamin, "Security and Privacy Threats for Bluetooth Low Energy in IoT and Wearable Devices: A Comprehensive Survey," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 251–281, 2022.

[10] S. C. Ergen, "ZigBee/IEEE 802.15.4 Summary," pp. 2–29, 2004.

[11] Prasanna. (2022) Zigbee Technology Advantages and Disadvantages | ZigBee Technology Architecture and Its Applications. [Online]. Available: https://www.aplustopper.com/zigbee-technology-advantages-and-disadvantages/

[12] B. Turay, "Analysis of seven layered architecture of osi model," *SSRN Electronic Journal*, 2019.

[13] Y. Li, D. Li, W. Cui, and R. Zhang, "Research based on osi model," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, 2011, pp. 554–557.

[14] Tektronix, "Wi-fi: Overview of the 802.11 physical layer and transmitter measurements." [Online]. Available: https://www.tek.com/en/documents/primer/wi-fi-overview-80211-physical-layer-and-transmitter-measurements

[15] Hewlett Packard, "Wireless overview - the mac level." [Online]. Available: https://hewlettpackard.github.io/wireless-tools/Linux.Wireless.mac.html

[16] "Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific

requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021.

[17] "Standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*, pp. 1–512, 1998.

[18] S. R. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of rc4," in *Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, ser. SAC '01. Berlin, Heidelberg: Springer-Verlag, 2001, p. 1,Äì24.

[19] A. H. Adnan, M. Abdirazak, A. S. Sadi, T. Anam, S. Z. Khan, M. M. Rahman, and M. M. Omar, "A comparative study of wlan security protocols: Wpa, wpa2," in *2015 International Conference on Advances in Electrical Engineering (ICAEE)*, 2015, pp. 165–169.

[20] Wi-Fi Alliance. (2020) WPA3 Specification v3.0. [Online]. Available: https://www.wi-fi.org/file/wpa3-specification

[21] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements. part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 4: Protected management frames," *IEEE Std 802.11w-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, and IEEE Std 802.11y-2008)*, pp. 1–111, 2009.

[22] G. Sagers, "Wpa3: The greatest security protocol that may never be," in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2021, pp. 1360–1364.

[23] V. K. Ramanna, J. Sheth, S. Liu, and B. Dezfouli, "Towards understanding and enhancing association and long sleep in low-power wifi iot systems," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 4, pp. 1833–1845, 2021.

[24] Cisco. (2017) Frequently Asked Questions About Management Frame Protection (MFP). [Online]. Available: https://www.cisco.com/c/en/us/support/docs/smb/wireless/cisco-small-business-wireless-access-points/smb5442-frequently-asked-questions-about-management-frame-protection.html

[25] S. Madsen and J. Schoen, "Jamming Attack Workaround Study," 2021.

[26] A. Amoordon, V. Deniau, A. Fleury, and C. Gransart, "A single supervised learning model to detect fake access points, frequency sweeping jamming and deauthentication attacks in IEEE 802.11 networks," *Machine Learning with Applications*, vol. 10, 2022.

[27] B. Aslam, M. H. Islam, and S. Khan, "802.11 Disassociation DoS Attack and Its Solutions: A Survey," *2006 Proceedings of the First Mobile Computing and Wireless Communication International Conference*, 2006.

[28] J. Bellardo and S. Savage, "Proceedings of the 12th USENIX Security Symposium," *802.11 Denial-of-Service attacks: real vulnerabilities and practical solutions*, 2003.

[29] airjack. [Online]. Available: sourceforge.net/projects/airjack/

[30] kismac. [Online]. Available: binaervarianz.de/projekte/

[31] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2017.

[32] A. Terkawi and N. Innab, "Major impacts of key reinstallation attack on internet of things system," in *2018 21st Saudi Computer Society National Computer Conference (NCC)*, 2018, pp. 1–6.

[33] Armis Inc. (2020) Armis Threat Detection. [Online]. Available: https://info.armis.com/rs/645-PDC-047/images/Armis-Threat-Detection-SB.pdf

[34] Palo Alto Networks. (2020) IoT Security. [Online]. Available: https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/datasheets/iot-security

[35] Amazon Web Services. (2022) AWS IoT Device Defender. [Online]. Available: https://aws.amazon.com/iot-device-defender/

[36] Microsoft Azure. (2022) Microsoft Defender for IoT. [Online]. Available: https://azure.microsoft.com/en-us/products/iot-defender/#overview

[37] Netgear. (2022) Netgear Armor. [Online]. Available: https://www.netgear.com/home/services/armor/

[38] Amazon Web Services. (2022) AWS IoT for the Connected Home. [Online]. Available: https://aws.amazon.com/iot/solutions/connected-home/

[39] M. Vanhoef and E. Ronen, "Dragonblood: Analyzing the Dragonfly handshake of WPA3 and EAP-pwd," in *IEEE Symposium on Security & Privacy (SP)*. IEEE, 2020.