Summer 2018

CARD: Concealed and remote discovery of IoT devices in victims' home networks

Sammie Lee Bush

CARD: CONCEALED AND REMOTE DISCOVERY OF IOT DEVICES IN VICTIMS' HOME NETWORKS

by

SAMMIE LEE BUSH

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER SCIENCE

2018

Approved by

Dan Lin, Advisor
Wei Jiang
Yanjie Fu

ABSTRACT

Smart devices are becoming more common in the standard households. They range from lights to refrigerators and their functionality and applications continues to grow with consumer demand. This increase in networked, complex devices has also brought an increase in vulnerabilities in the average consumerś home. There now exists an Internet of Things (IoT) ecosystem that creates new attack vectors for adversaries to spread malware, build botnets, and participate in other malicious activities. We will overview some of these new attack vectors as well as go over a framework that would allow an adversary to target a userś home network and any other networks that user may join.

ACKNOWLEDGMENTS

I would like to thank my parents for supporting me through life and especiallity as I went through my education.

I would also like to express my appreciation to my advisor who guided me through my degree program and insured everything I covered contributed to both my research and future career.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

The newest app sensation "Swimmy Fish" is getting thousands of downloads a day! You decided to join the craze and see what the game is all about. As you are looking for the game on the app store, you find out that it has been taken off of the app store and is now hosted exclusively on the developer's site. You then take this chance to get it free from the developer's site. When you try to install the download, you get an alert that you need to turn on your "Unknown sources" setting in your phone's security settings. After that, you install the .apk and start playing the hottest sensation for mobile devices!

You go visit your friend Jim's house before you head out of town. Later that day, you get a call from Jim. While you were at his home, a series of events took place as you two look at the new smart home package Jim had installed recently. "Swimmy Fish" began to start new processes on your phone that scanned Jim's home for any interesting devices. It reported its findings to an attacker that compromised the Swimmy Fish developer's site weeks ago and replaced the original game with a modified version. That attacker configured their Command & Control server in such a way that it could view every IoT device on the network and exploit them remotely through Bob's phone. The attacker now has a foothold in Jim's internal network. The attacker also took the time to attempt a takeover of Jim's phone through the broadpwn exploit. After hosting a malicious AP through one of the compromised IoT devices, Jim's phone has been compromised. Once the attacker got everything he wanted from Jim, they decided to flicker all of Jim's lights.

This scenario is made possible by taking advantage of the inherent trust in the common home IoT environment.

This framework has been created to bring attention to the insecurities the current wave of IoT devices has brought to the unsuspecting user. Many of the devices available for purchase have little to no embedded security and bring a range of vulnerabilities to networks that are typically seen as trusted. By bringing attention to this, there is the hope that more research will be done to fix the insecurities in consumer IoT devices.

A previous paper (Vijay Sivaraman and Boreli (2016)) outlines an attack vector that weakens the general security of the network so that a remote adversary can attack through the home router. This method relies on UPnP to reconfigure a home router and create holes from which an attacker can send malicious commands to the internal network.

The problem with this approach comes with those SOHO devices which do not allow this sort of attack. This sort of attack could also raise alarms in some networks and reduce its effectiveness significantly.

We propose an attack vector that does not rely on the existing infrastructure being made insecure, but instead takes advantage of an infected device to infect any network it becomes a part of. This would allow for a moving initial infection point centered at every mobile device that becomes infected.

This paper will go through an IoT attack framework that takes advantage of the weaknesses in the current consumer IoT ecosystem. We will start with different methods of spreading infected applications to users. Next, we will look at the home network specifics used in this experiment and target devices used as infection points. The Command & Control and malicious app features will then be outlined to show some of the capabilities a malware author could implement for this attack vector. We will then discuss potential mitigations for this sort of attack and future works connected to this experiment.

2. LITERATURE REVIEW

2.1. INTERNET OF THINGS VULNERABILITY LITERATURE

The following researchers write about vulnerabilities that can be found in the internet of things. Marin, et al. (Marin (2016)) looks at commercial off the shelf implantable medical devices and their device programmers. Through their research, they found an instance of obfuscation of data as one of the first cases of a security implementation in these systems. they later find issues with privacy, DoS and other vulnerabilities that can be found in other devices. These issues were found through black box reverse engineering, which requires looking at the outputs that result from inputs between the devices. They started by changing the transmitted patient name slightly to see the differences in the encoding. A proposed countermeasure to the attacks an adversary could use would be to jam the channel being used by the device when malicious activity is detected. This would prevent further activity, but potentially disrupt other friendly devices. Other proposed mitigations include having devices send a message that prevents the them form entering a vulnerable state, and creating a key agreement system that authenticates devices to a programmer and prevents other programmers from interacting with unpaired devices. This paper really shows how wide spread the problems in IoT devices are. It has come to a point that physical harm can come to someone if an adversary wished to cause it. Wing, et al.'s (Wing and Zhao (2016)) presentation reviews reliability, security, and privacy in relation to historical examples brought up in the presentation. She talks about the research challenges we need to overcome to ensure users trust these systems when they're more widespread and interconnected. Scanning for Vulnerable Devices in the Internet of Things.

Nicola Dragoni, et al. (Nicola Dragoni and Mazzara (2017)) present an overview of IoT devices and the security issues they face. They make the argument that weak security (default passwords, weak hashing, weak encryption, exposed headers) is the same as no security at all and that there is a lack of security because the focus is on profitability and there is too much difficulty in implementing reliable security practices in low power devices. They reference many incidents where a lack of security in devices has led to big negative outcomes. Many internet connected devices are said to be developed without security in mind. The paper argues that the only fix for these problems is to educate the public on security in general and build a new culture where all people are security and privacy minded. The paper does not present a suggested education plan, but does suggest that exposing kids to these technologies without presenting the concepts of security and privacy is growing the problem. These different works discuss the general vulnerability of Internet of Things environments. There is also a focus on how vulnerable devices can bring kinetic problems to the world.

2.2. INTERNET OF THINGS DEFENSES LITERATURE

These works propose different ways to improve to security and resilliency of devices in the Internet of Things space. A common problem many of these papers face is the power restrictions typically inherent on Internet of Things devices. Agadakos, et al. (Agadakos (2016)) proposes a system to authenticate users via their IoT devices. This requires a user to carry multiple IoT devices on them so that the system can calculate the chance they are in certain locations. This kind of authentication is not reliable if users do not always carry a decent amount of IoT devices on them. It also requires the user to maintain some sort of network connectivity. Bagci, et al. (Bagci (2015)) proposes a method of detecting when wireless transmitters are tampered with. This method requires multiple detectors to be in the vicinity of the malicious party. The detection nodes would read the CSI(channel state information) values presented by the devices to find changes in those values. A change in that value could result from physical changes to the transmitter, or movement of the wireless transmitter. An example given for this involves a wireless camera security system which could alert the user if a camera's point of view or placement has been changed. A problem with this system is the changes in CSI values that occur when people are moving in the area it is used. This requires a system that can see the difference between environmental changes, and tampering. A distance metric is used to find changes in the CSI values. Changes to this measurement are made depending on the number of detection nodes. This detection system would improve the integrity of physical security systems already in place, but may need more data fed to it so that it can properly detect events of tampering. Earlence Fernandes, et al. (Earlence Fernandes (2016)) work proposes new framework called FlowFence. This frameworks goal is to define how applications use data during development. It splits up apps into secure modules that cant directly handle sensitive data, and code that uses the secure modules together to complete tasks. This allows for higher security for the sensitive pieces of information running through these devices. The framework creates a data flow graph to determine what data can go where. This is signified in the code. The policy generated from this prevents sensitive, dereferenced data from reaching certain functions. An average of 99 lines of code was added to each app they modified to use FlowFence. The effectiveness of this framework seems to depend on the implementation of its features. Developers could easily make mistakes that would cause relay attacks and even information leakage to still be a viable threat. Karbab, et al. (Karbab (2016)) proposes a framework for detecting malicious applications not by signatures, but by their relationships to other malicious applications. These are named malicious communities. The framework would take in several apps and compare them to a fingerprint database that would be generated from malicious community graphs. These graphs are generated by finding similarities between app binaries and features that are known to be malicious. As more apps are put through this system, more malicious communities are made. The data sets they used for testing were both known malicious data sets and random android apps from the store. Their results found that their framework was able to detect 87% of malicious applications.Something like this could be very effective if used with Google Play Protect. When a new malicious application would be found, the similar applications that are scanned could be instantly deemed a threat, though the accuracy of finding those applications could pose a problem for keeping users satisfied with the service. Kim, et al. (Kim (2017)) research existing IoT protocols and looks for the challenges these protocols face by using formal symbolic modeling and applying these protocols to scenarios that involve different attackers and security goals. It also goes over crypto DoS attacks that these protocols are also vulnerable to and proposes a new mitigation technique.The attacker/adversary models they look at are D-Y(Dolev-Yao), eCK(Canetti-Krawzcyk), and PFS(Perfect Forward Secrecy).The protocols looked at are SigFox, LoRa, MQTT, CoAP, and JPAKE. They use something called a Tamarin rule to model these protocols. Tamarin rules have three sides or pieces. The first on the left is the premises and consuming facts, the second in the middle is for defining actions, and the last on the right is the conclusion. These are combined with tamarin cryptographic primitives to represent each protocol's functions.The mitigation for crypto DoS involves a PCBC that uses multiple keys. This can be implemented in the protocols they studied, as this mitigation approach relies on only AES/HMAC. Their DoS modeling seems like a good tool to use when verifying the suitability of future IoT protocols. Krentz, et al. (Krentz (2015)) review security in devices that use the IEEE wireless standard 801.15.4. This standard defines how LR-WPANs(low-rate wireless personal area networks) operate. Many IoT devices rely on this standard to communicate with each other. Specifically, they look at the limitations of how devices store anti-replay data and the frame counter values in memory. They propose APKES(Pairwise Key Establishment Scheme. This scheme allows devices to survive reboots, which would require them to lose these values, and reduces the memory and energy used by establishing session keys. This scheme first has neighbors obtain a PAN(personal area network) identifier. Nodes then send out hello packets to establish keys after handshaking. Information on each neighbor is then stored in a table. This is very similar to how OSPF functions between routers, except this scheme needs a centralized node. The implementation of this drastically improves the security of many IoT devices, as one of their flaws originate from the replay attack vulnerabilities they present. Maulana, et al. (Maulana (2016)) proposes a new system for protecting the privacy of users that carry and use BLE(bluetooth low energy) devices. This system prevents an adversary from connecting and even discovering devices the user has configured to be protected by the BLE-Guardian. BLE-Guardian is implemented on an Ubertooth One. Only clients that have be authorized by the user can connect to the protected devices. To prevent adversaries from connecting, BLE-Guardian learns when the devices broadcasts, and jams those advertisements, only opening them up when the client requests to make a connection. The creation of BLE-Guardian is justified through research in the BLE implementation in existence. They found that many of randomly generated address from these devices kept that address for over 24 hours, defeating the purpose of randomizing them. This implementation seems to still give away a users location to the attacker. If an attacker is aware that their target uses this system, they could potentially monitor for these jamming activities to location their target. Shi, et al. (Shi (2016)) reviews IoT devices that need to communicate in environments where attackers have all the source code and can monitor everything put in the air. The name of this environment is referred to as a white-box attack context. The encryption needed to securely communicate in this environment is seen as very costly and time consuming by the authors, so they propose a new lightweight encryption scheme. The contribution is a new symmetric encryption scheme that takes into account: the memory available on IoT devices, the processing power of IoT devices, and the security that needs to be achieved to be protected in this context. The scheme contains three functions: DataGen, Enc, and Dec. DataGen generates to static sets of data for encrypting and decrypting. This function also creates a T-Box with contains a series of adders which output masked values. Enc takes in 128 bits of cleartext and one of the sets of static data to encrypt the cleartext. Dec does the same but with the other static set and ciphertext. Their results showed that their implementation reduced the memory used, while reducing computation costs for 10, 16, and 32 rounds. An implementation like this would make it near impossible to get devices to leak information, depending on how they verify requests. Valente, et al. (Valente and CÃČÂąrdenas (2015)) proposes a method of proving integrity of video feeds through visual challenges like QR codes or strings of plain text. They do this by putting up the challenge in a space the camera is looking at. The camera would see this challenge and then reply with the solution to the puzzle. This system would only require two editions to already existing systems. These are a verifier that serves up visual challenges, and a display for these challenges. If an attacker were to find a way to alter the visual feed or physically change the camera, not being able to get a correct response from the visual challenge would indicate an attack is taking place. Challenges are randomized so that an attacker cannot replay a repeated challenge after recording a large number of them. If QR codes are used for image verification, attackers could take advantage of the QR code vulnerability where the code cannot be decoded and repeat previous undecodable codes. This approach to improving integrity in camera feeds is very interesting and I can only see an issue if the attacker could gain access to the verifier somehow, which the writer says would be difficult to detect in real time due to the discipline of video and image forensics still being new. Yang, et al. (Yang (2016)) proposes two multi-cloud based outsourced attribute-based encryption schemes. These schemes are supposed to both reduce the computations required for these devices, and increase privacy for data, attributes, and access policy. The author makes the assumption that the cloud is an honest-but curious party that will follow the protocol faithfully, but obtain any information leaked. A combination of ElGamal-type ciphertext and multiple cloud servers is used to complete tasks without reveal the original message, accurate user attributes, or the access policy to the cloud. The two proposed schemes are the chain-cloud scheme and the parallel-cloud scheme. In chain-cloud, the receiver specifies the number of clouds they wish to use that can satisfy the distribution of attributes they wish to use. In parallel-cloud, the sender chooses a number of clouds to send their messages to simultaneously and uses only those clouds. the parallel solution showed to have the least delay, as decryption is done simultaneously as opposed to sequentially like the chain scheme. The chain-cloud scheme reminds me of the TOR protocol used to ensure each node in the chain can only see the data they are able to decrypt and then passes it on. Zand, et al. (Zand (2015)) researches administrator network awareness, and where current methods fail. It then talks about a new tool called PARIS which monitors the network traffic and then calculates an importance metric for the devices on the network based on what administrators configure as defensive and prohibitive actions. Through the network traffic, PARIS find network services running, relationships between nodes, and relationships between services. This allows PARIS to build graphs that represent nodes that need to stay up together to complete certain tasks and find activity in the network that harms these tasks. Importance scores are calculated from: the number of bytes and packets sent and received, the number of requests handled by the service, the number of network missions the service has been involved in, the number of failures of the service, and the number of clients of the service. When fed data from a university department network, PARIS was able to rank the most important services on the network and the most important grouped host services. This seems like a really nice network administration tool, with a lot of situational awareness capabilities. Something like this would work well in CCDC. Vinay Sachidananda, et al. (Sachidananda *et al.* (2017)) proposes an implementation of an IoT test bed. This test bed's aim is to allow for security testing of not just the individual devices, but the ecosystem as a whole. It includes a room that blocks any external signals from interfering with the environment, a robust logging system, and access point for allowing internet access to the test devices, and many other features that allow for painless IoT security testing. Through their testing of several popular devices, they were able to build a fingerprint which can be used to find these devices elsewhere on the internet or any other network. These fingerprints also allow for vulnerability scans based on the service versions running on each open port. These sort of fingerprints will be used in my research for targeting specific devices with exploit code.

2.3. INTERNET OF THINGS MALWARE ANALYSIS LITERATURE

These publications discuss and analyze the functionality and impact various pieces of Internet of Things malware have had on the world. They compare and contrast the different methods used for reconnaissance, delivery, exploitation, installation, command and control, and the different actions taken on infected devices. Kambourakis, et al. (Kambourakis (2017)) analyzes IoT botnets, malware, and their relationships. Their research found that Mirai branched off into many different variants with additional features and different targets. A comparison is done between Mirai and Hajime. It's split up into three parts: infiltration, inspection, and operation. In infiltration, Mirai scans for port 23, while Hajime scans for 5358. Both attempt to use credentials stored in a list of possible creds. Mirai does this sequentially, while Hajime randomly selects a set of credentials. Mirai then tries to enable a linux shell on the device and checks the environment with a command that should return an error. In the inspection phase, both pieces of malware fingerprint the device and look for writeable directories. Mirai will then download a small executable, run it, and then delete it. Hajime will download a small executable that acts as a downloader for the larger piece of malware stored on the C2. In the final operations step, Mirai scans for additional devices to infect. It relays information found to the C2 over http and accepts commands over a tcp socket. Hajime uses BitTorrent for all communications and data transports. It continues to infect other devices and transfer the worm over a random high tcp port, which shows how decentralized it is once it has made its first infection. Once on a device, it opens up certain ports and changes firewall rules.This paper has given me more ideas on how to make a better attack framework that allows for similar functionality within a trusted network. Seungjoo Kim's (Seungjoo Kim (2015)) research looks at situations were IoT devices were used in

DDoS attacks, and specifically in attacks that targeted DNS servers. In case 1, the IP of one of the attacking devices is found and traced back to a home router. Telnet is activated on that router to run an analysis of the processes running. A copy of the firmware is then downloaded and reverse engineered to find the malicious commands used to infect the device and turn it into a bot. It then spreads further by scanning random IP addresses. Case 2 lead to a network switch which was similar to case 1, but pulled entirely different binaries part of a different malware group. Case 3 found an infected CCTV which had its password changed and an infected firmware image uploaded. These CCTVs were initially infected due to their default login credentials, and being exposed to the open internet. The presenter then shows their future work in automatic IoT vulnerability scanning. This case study helps show a lot of the ongoing malicious activities that are plauging IoT devices. Galluscio, et al. (Galluscio *et al.* (2017)) research looks at IoT exploitation as a whole and analyzes the total harm it can bring to the world. The researchers did a series of scans that found many high-value targets were made vulnerable by Internet of Things devices connected to their cyber physical system networks and also exposed to the internet. The authors also created an algorithm which listen to darknet traffic data and tries to find malicious activity towards these devices which can be tracked. With this combination of data, the researchers were able to find at least 11 thousand compromised devices in the wild, with most of the devices being located in China and the second most located in the US. This only accounts for the devices you can see from the internet. With the approach proposed in my research, devices also located inside of internal networks could potentially be scanned.

2.4. INTERNET OF THINGS ATTACK IMPLEMENTATION LITERATURE

These papers demonstrate and analyze implementations of proposed exploits. Only one of these papers covers a unique exploit, while the other two cover ways those exploits can be delivered and how those devices can be manipulated. Ronen, et al. (Ronen and Shamir (2016)) works look at the classification of IoT device attacks. It details some of the classical attacks and then provides examples of attacks that extend the functionality already in these devices. One of these examples exploits lightbulbs in a way that allows for data exfiltration through light flashes. Another extended functionality attack takes advantage of the frequency these lights can be flashed to cause seizures in vulnerable people. This paper helps put light on more of the risks IoT can bring into an environments and some of the vulnerabilities users should be aware of for future mitigations. Vijay Sivaraman, et al. (Vijay Sivaraman and Boreli (2016)) propose a malicious iPhone app that scans an internal network for devices, and exploited vulnerable routers to allow exploits to enter the internal network. This paper has the most similarities to my research, but I see several flaws in its approach. This solution relies on the home router having universal plug and play (UPnP) enabled, and the local networking having a reliable connection to the Internet with no firewall. If the user installs a home router and has UPnP disabled, exploits could not make it into the LAN. If there is a firewall on the network, these exploits could be detected and their source blocked. The use of this framework presents many viable points of failure that could render this attack ineffective. Eyal Ronen, et al. (Eyal Ronen and Weingarten (2016)) reverse engineer the ZigBee protocol and present a newly developed exploit that allows for an attacker to execute code on remote devices and have that code spread in a worm like manner. The writers spread this worm via drone that was flown outside of an office building. ZigBee is one of the most used communication protocols in IoT devices. It's used for commands that control the devices, and allow for them to communicate with each other. The researchers also looked at something called the percolation theory. This is the theory that once a certain amount of people become infected (the percolation threshold) everyone else within that area will become infected instead of the infection being isolated to a small area. This is an interesting theory that I was not aware of previously. It seems that it could easily be applied to any worm-like software. It reminds me of the Myspace Samy worm that had an initial infection point of just his profile but soon spread to everyone on Myspace.

3. AN ATTACK FRAMEWORK TO DISCOVER IOT DEVICES IN HOME NETWORKS

3.1. AN OVERVIEW OF THE ATTACK FRAMEWORK

This infection framework is designed to take advantage of the weakness currently present in many home local area networks. In many cases, home networks are designed with no seperation of traffic between the devices on the same network. This leads to an environment of inherent trust between devices, which can be taken advantage of. This attack method relies on this home network configuration to exploit devices on the same network and report outside of the invaded network to a malicious command and control (C2) server. It does this by concealing malicious code within the target mobile device. This can be done either through infected mobile applications, or vulnerabilities found in the devices themselves. Once these devices are taken into a wireless network, the malicious code looks for IoT devices in that network and sends the results to the C2. Mobile devices are typically taken into multiple home networks, as they stay with the user wherever they go. Because of this, infections can spread from home network to home network. This is depicted in Figure 3.1. Actions on the local network can then be taken either automatically, dependent on what was found on that network, or a malicious actor operating the C2 can execute commands remotely. Because of how this attack works behind typical network protections like Firewalls, and has a view of the internal network that you couldn't necessarily see from the internet, a plethora of new attack vectors open. With this new found direct access to vulnerable devices, attacks can no longer be prevented in the typical household with minimal protections.

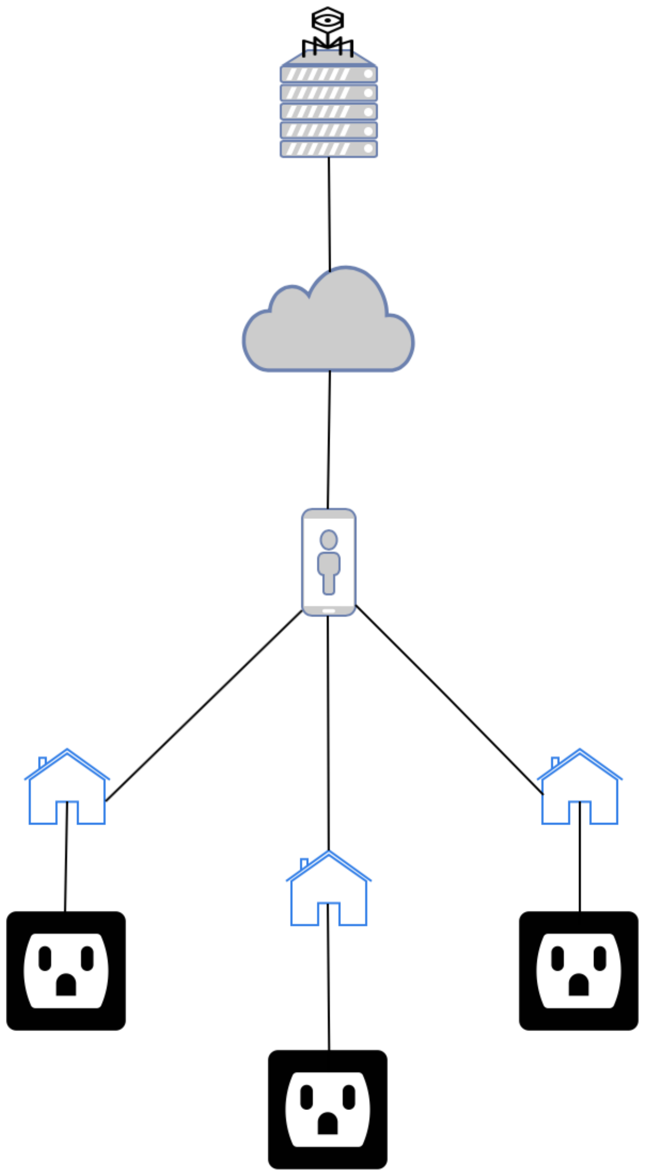


Figure 3.1. Attack Framework Overview

3.2. ATTACK MODEL

There are several methods of distributing malicious applications to users. One of these relies on the use of social engineering to trick users into downloading and installing an application outside of their official app store. In the case of GinMaster (GinMaster: A case study in Android Malware (.)) distribution was done in over 400 different app stores. Well known applications were injected with the GinMaster malicious code and placed in these stores with their original functionality in tact. This allowed for the malicious activities of the application to continue with few detecting it. A more recent example of this technique appeared during the staggered release of Pokemon GO. (Droidjack Uses Side Load Backdoored Pokemon Go Android App (.)) Many users were looking to start playing the game before its release in their country, which led to them downloading and installing the app from untrusted sources. This issue grew to be much worse once media outlets began spreading instructions on how to acquire and install these malicious applications, a process which involved checking a box saying you were okay with installing applications from unknown sources on your device. This has since been changed to only allow approved applications to ask if you would like to install and app from an unknown source, instead of a single global check box.

Another, more direct threat is the presence of malicious applications in the official Google Play Store. Because the Play Store is generally trusted, its users are not typically expecting strange behavior to be connected to malicious behavior. This trust could be exploited by malicious app creators to spread their malware to unsuspecting users with relative ease. One example of a strain of malware that has been found in the app store is called ExpensiveWall. (Expensivewall: A Dangerous Packed Malware On Google Play That Will Hit Your Wallet (.)) This malware maintained connection to a Command & Control and also subscribed users to premium programs. Previous users of some of the infected applications reported that the app was advertised on popular social media websites. This shows how general advertising can bring many of these applications hundreds of thousands to millions of downloads. Google has released a fix for this problem in its most recent OS Android 8 (Oreo) with a security suite called Google Play Protect. (Introducing Google Play Protect (.)) This new suite now scans all downloaded applications before install and periodically scans already installed applications for malicious behaviors. If any unknown

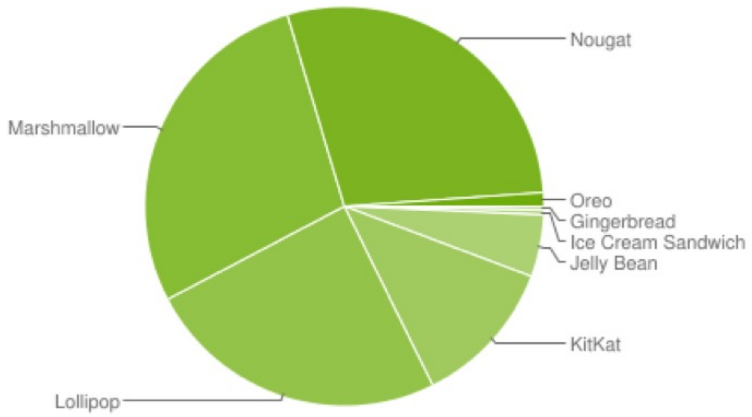
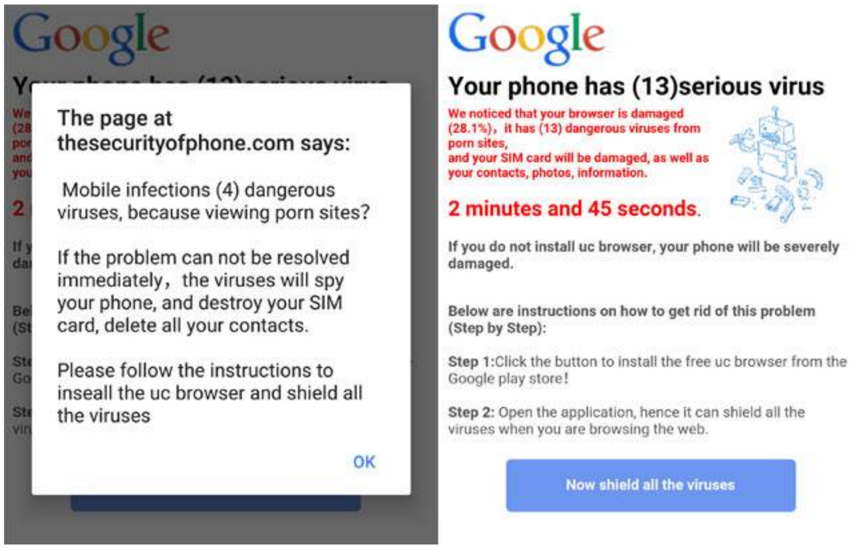


Figure 3.2. Google Dashboard of Installed Operating Systems

applications are found, they are sent to Google for further analysis. If effective, this would solve many of the problems plaguing the Play Store, for Android 8 users. However, for users who have devices that do not support the new OS (Figure 3.2), which is more that 99% of users (Android Dashboards (.)), many of these problems persist.

A common tactic used in the wild to trick users into downloading a malicious app, is to use some sort of scare tactic and present a sense of urgency in the target. This is done through pop-up advertisements for android users [Figure 3.3]. This form of social engineering is used to catch users off guard and cause them to make split second decisions that could potentially infect their devices. It is common the run into these sorts of malicious Ads on high traffic sites, due to the rotational nature of their infection scheme.

A less common method of infecting android devices is done by attacking the system directly. There are few cases where it is possible to infect mobile devices without user interaction. Two recent methods that have come to light are BlueBorne(BlueBorne (.)) and BroadPwn(Broadpwn (.)). These attacks take advantage of vulnerabilities in the firmware of the bluetooth and wireless chips present in many mobile devices. They allow for remote code execution without user interaction, which could be the starting point for this sort of attack.


Google

The page at thesecurityofphone.com says:

Mobile infections (4) dangerous viruses, because viewing porn sites?

If the problem can not be resolved immediately, the viruses will spy your phone, and destroy your SIM card, delete all your contacts.

Please follow the instructions to inseall the uc browser and shield all the viruses

OK

Google

Your phone has (13)serious virus

We noticed that your browser is damaged (28.1%), it has (13) dangerous viruses from porn sites,
and your SIM card will be damaged, as well as your contacts, photos, information.

2 minutes and 45 seconds.

If you do not install uc browser, your phone will be severely damaged.

Below are instructions on how to get rid of this problem (Step by Step):

Step 1:Click the button to install the free uc browser from the Google play store!

Step 2: Open the application, hence it can shield all the viruses when you are browsing the web.

Now shield all the viruses

Figure 3.3. Scare Tactic Example

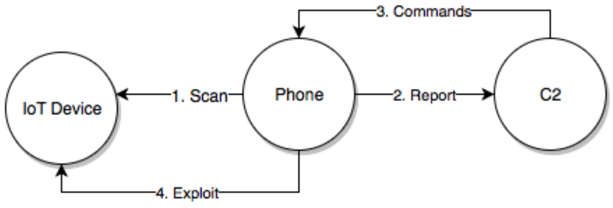


Figure 3.4. Information Flow

3.3. IMPLEMENTATION OF CARD

The CARD framework relies on a few functions to completed the needed functions in any given environment. It needs to be able to analyze its environment, report results of this analysis, and interact with the devices adjacent to it.

- scan() - The scanning function takes no parameters. It does a ping scan, followed by a port scan of devices that appear to be up and saves this information.
- report(c2) - The report function takes in the target command and control location. It transmits the collected information to this command and control.
- update(c2) - Takes in target command and control location. Reads this command location for exploit updates and commands.

- exploit(target, exploit) - Takes in the target IP and exploit choice. Uses this information to push an exploit onto a particular device.

Figure 3.4 depicts the series of actions taken in the system to use the available functions to exploit and/or detect devices on the internal network. The scan function is used to determine what devices are in the infected phone's network. First, the phone determines what subnet it is operating in. Then, the device sets its scan range for this subnet. A ping scan of the network is then executed to determine if a device is up or down. If a response is received, the ip address is saved as active. Once the ping scan is complete, a port scan is conducted on the saved ip addresses to determine what active ports are on these devices. The active ports are then saved to each device's open ports list. This ports list acts as a fingerprint to determine what type of device was scanned and helps with further exploit targeting. Depending on what the fingerprint returns, it can then label this device as either an IoT device or not an IoT device. This allows for the C2 operator to determine what devices they'd like to focus on. Once this information is gathered from the phone's environment, it is transmitted to the C2 through Pastebin. Using the Pastebin API, pastes only visible to the account owner can be made through an encrypted TLS tunnel. The Phone transmits IP, whether it has located an IoT device or not, and the ports active on that particular device. This data can then be examined by the C2 operator. Depending on how the CARD system has been configured, automatic action can be taken depending on what devices are found on the network, or the operator can issues commands to the phone. Commands are transmitted via pastes that are then read by the phone via the Pastebin API. Once this step is complete, the final step is to exploit the target devices and carry out whatever actions have been chosen. The scanning process repeats after this.

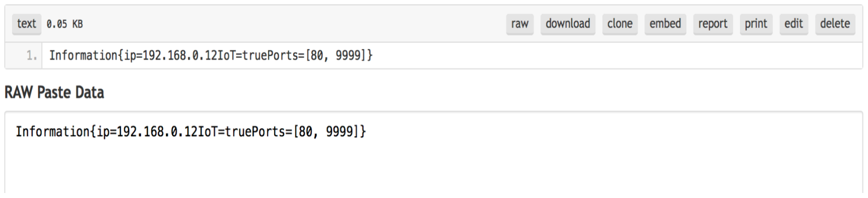
This framework provides several functions for operating within a LAN remotely through the compromised android device. The first of these features allows for the scanning of the internal network.

Internal network scans are done synchronously. Once a device is found to be up, a port scan is started on that device. The framework allows for either a full port scan or a partial port scan based on a list of target ports. Once this scan is complete, a device object is populated with the IP address and open ports. A test is then made to determine whether a found device is an IoT device or not based on its open ports and a list of IoT devices and their corresponding open ports. If a device is found to be an IoT device, a flag is ticked in that device object. Once this phase is complete, The data is sent to the designated Pastebin account. Example of how it is received is in Figure 3.4. The contents of this sent data can be seen in Figures 3.5 and 3.6.

The next feature allows for the user to remotely manipulate devices on the network. This can range from simply issuing commands to said devices, to exploiting those same devices. Commands can be created in the Pastebin and then read by infected devices to be executed on the phone.

Depending on what type of IoT device is found, specific exploits can be run. In this set-up, when a smart plug is detected on the network, a loop of on and off commands is executed on the IoT device. Exploits can be added to the device in a number of ways. This could be a series of system commands carried out by the Phone, or the execution of a program that has been downloaded to the phone. Additional exploits can be added to fit the target environment and effect.

Lastly, this framework allows for communication between the Command & Control and infected mobile devices through the Internet. For this implementation, The Pastebin API is used to do this. Pastebin is a plaint text online storage solution. It is typically used to share large snippets of text in online chat rooms. We chose Pastebin for a few reasons. The first is because it is easily accessible over the Internet. This makes sure it has high availability and can stay in constant contact with infected hosts while also reducing the burden of hosting my own servers for communication. Secondly, Pastebin provides a robust API that allows me to interact with the website over HTTPS, keeping communication

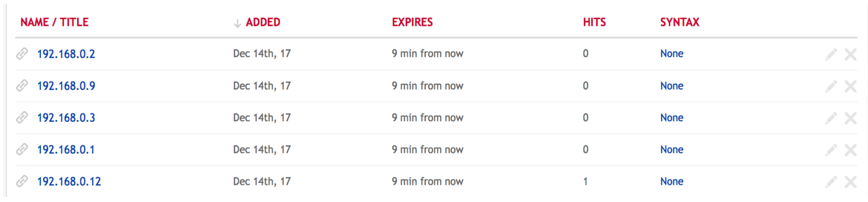


```
Information{ip=192.168.0.12IoT=truePorts=[80, 9999]}
```

RAW Paste Data

```
Information{ip=192.168.0.12IoT=truePorts=[80, 9999]}
```

Figure 3.5. Data Sent from Infected Android Phone



NAME / TITLE	ADDED	EXPIRES	HITS	SYNTAX
192.168.0.2	Dec 14th, 17	9 min from now	0	None
192.168.0.9	Dec 14th, 17	9 min from now	0	None
192.168.0.3	Dec 14th, 17	9 min from now	0	None
192.168.0.1	Dec 14th, 17	9 min from now	0	None
192.168.0.12	Dec 14th, 17	9 min from now	1	None

Figure 3.6. Data Listed by IP Address on Pastebin

encrypted over the wire and hiding amongst other HTTPS traffic in the network. Lastly, We felt that Pastebin would be less likely to be filtered on network due to it being a widely used website. There is also a history of Pastebin being used as a Command & Control in other pieces of malware. An example of this would be Dridex, a banking trojan that was typically spread through word documents. Data collected by infected devices in LANs is sent to Pastebin and read by the controller. The infected device uses post requests to the API to transfer data. A controller can then use the same API to view and manipulate the posted data. To ensure the data collected does not leak, post requests should be made through HTTPS. Posts should also be done to the private bin of a specific user. This can be ensured by generating a user key for the target account. This way, only someone with a valid user key or access to that account can view the collected data.

3.4. THREAT ANALYSIS

There are several paths an attacker could take with this sort of framework in mind. (Table 3.1) Each path requires at least two steps before having access to the home local area network.

Table 3.1. A Description of the Different Methods of Infecting a Target

Staging Site	Restricted to users that visit the attacker's site
Google Play Store	Restricted to users that use the Google Play Store
Malicious Advertisements	Restricted to users with outdated devices and/or browsers
Direct Infection	Restricted to users the attacker has physical access to

- Step 1: Infect the mobile device
- Step 2: Use infected device to gain access to home LAN
- Step 3: Complete some action on the objective

Step one can be accomplished in three ways. The attacker can either: host the malicious app on a staging site, host the application on the Google Play Store, start a malicious ad campaign, or infect the target's phone directly. Depending on the choice for step 1, a different subset of android users would be targeted.

Once you have gotten past phone infection, users would be required to have a wireless local area network with IoT devices to exploit. A Gartner analysis (Gartner IoT (.)) says that the number of IoT equipped homes will only increase as time goes on, increasing the target space.

Once in that network, a variety of actions can be taken depending on the contents of that network. Ronen (Ronen and Shamir (2016)) describes a scenario where seizures could be triggered in public spaces. This could come to be a form of ransom-ware in the future. There is also a plethora of tracking data that could be acquired from these homes. By reading the set temperature of a smart thermostat, attackers could see when users are away from home or alter the devices only when they are home. (IoT Ransomware (.)) This attack framework is also not limited to the user's home. Any network the user connects to is potentially vulnerable to these threats. (Figure 3.1) Figure 3.7 presents a threat model that shows some of the risks in these environments. A Burglar, or someone looking to take physical objects from a victims home would need to do percision targetting while also

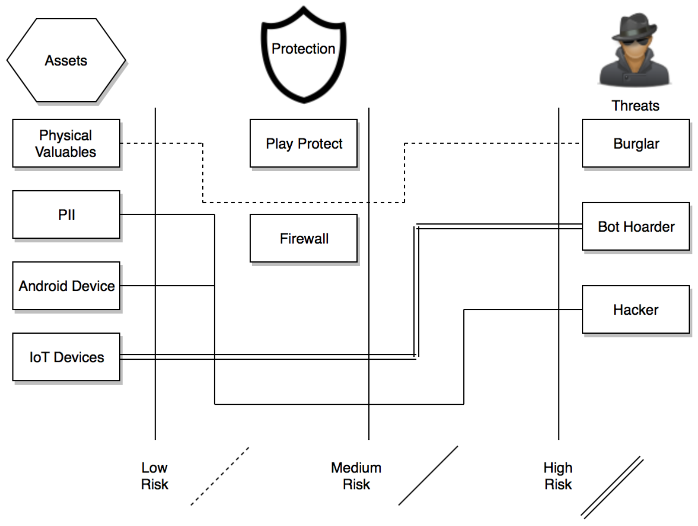


Figure 3.7. Threat Model

facing additional physical protections. A Bot Hoarder would be able to indiscriminately infect many users and acheive their goal of gaining control over many devices. A hacker, or someone seeking digital information or addional access to your device may stop at infecting your phone to obtain personally identifiable information (PII). In the case of an advanced persistent threat, additional post exploitation steps could be taken.

There are several actions that can be taken once an actor has gained access to your devices.

- Implant/Spying - Using devices to collect information on subjects
- Disruption/DDoS - Using devices to either disrupt the people using them or deny service to other remote devices
- Botnet Herding - Maintaining longterm control to complete further actions or sell botnet as a service
- Pivoting - Using infected device as a hop point toward higher value targets
- CryptoCurrency Mining - Using devices to mine cryptocurrency

To get to this point, a malicous actor would have to follow a series of steps. These steps are outlined in the Lockheed Martin Cyber Kill Chain Chain (.). These steps are:

1. Reconnaissance: Collection of information on target and connected subtargets
2. Weaponization: Usage of collected information to effect target
3. Delivery: Transport mechanism for effecting target
4. Exploitations: Method of opening hole in target
5. Installation: Method of placing tools on target
6. Command and Control: Method of maintaining connection and control over target and/or tools on target
7. Actions on Objective: Activities to be completed on target

At the Reconnaissance step when using this framework, a malicious actor would determine their target group and use one of the infection points to put the CARD malware on their mobile device. Once this is done, victims will begin enumerating their home environment for an actor to analyze. Depending on what devices the actor finds, they can then weaponize exploits for the located devices and begin updating their malware remotely with the necessary exploits. Exploits can then be delivered directly to the vulnerable devices from the mobile device. Once these devices are exploited, the necessary tools for each action to be taken can be installed for that particular device's architecture. These tools can range from post exploitation tools, scanning tools like nmap(Nmap (.)), or C2 client software that allows for additional control over the internet. Once these tools are installed, Command and Control can either be done through the phone as a proxy, or to the infected IoT device directly over the internet through a reverse shell(Reverse Shell (.)) which would call to the C2 from inside the network, outside, avoiding the protections a firewall may provide.. With this communication tunnel established, any action the malicious actor wishes to make can be made.

4. EXPERIMENTAL STUDY

4.1. EXPERIMENTAL SETTINGS

The target device is a rooted Nexus 5 running android 5.1.1 (Lollipop). The phone is connected to a WLAN with a subnet of 192.168.0.0/24. The phone is also running Pwnie OS on kernel version 3.4.0-g971696c-dirty.

The WLAN consists of a few devices. A TPLINK smart plug, a Linksys Wireless-G 2.4GHz 54Mbps Broadband router, and a Macbook.

4.2. EXPERIMENTAL RESULTS

We performed several experiments to determine the detectibility of the malicious functions on an infected phone. These experiments focused on functionality that could present behaviours that would alert a user to the malicious background functionality of their application.

4.2.1. Battery Drain. We hypothesized that the added functionality to the game application would have little to no affect on the charge of a fully charged battery. The data collected supports this claim(Figure 4.1). Once We began doing network scans, the battery would report an hour less of battery life. This shows that there is the potential of detection with an aware user.

4.2.2. Game Response Time. When looking at the game response, We measured the time it took for the character to fall to the ground at start and initialize the impact sound effect. This came out to a baseline average of 1.906 seconds over 5 trials(Figure 4.2). When looking at game response time during scanning periods, the game's delay was comparable at an average 1.882 seconds.

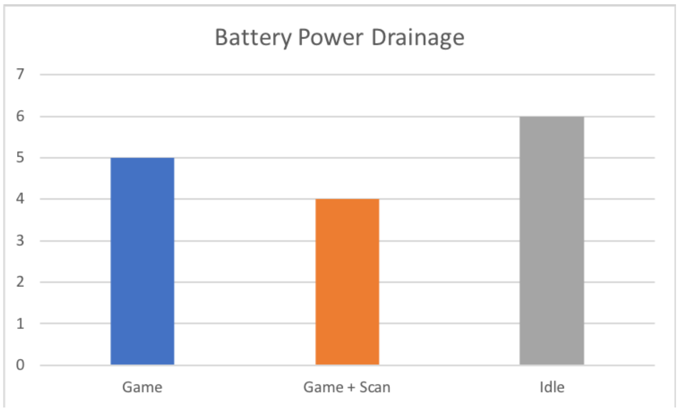


Figure 4.1. Battery Drain

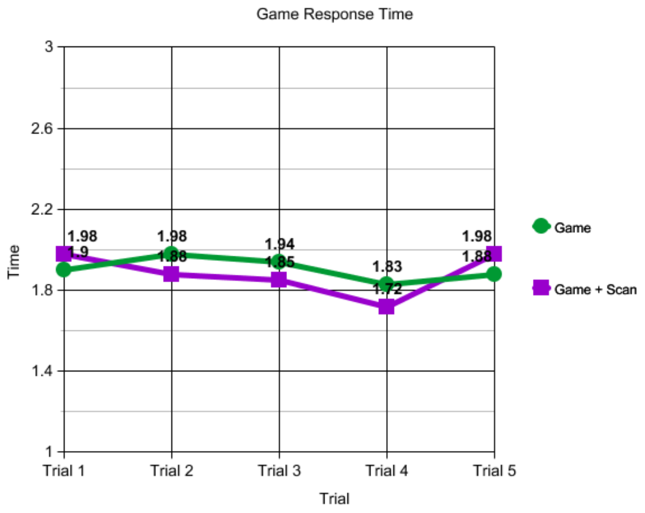


Figure 4.2. Game Response Time

4.2.3. Scanning Time. The time required to scan the devices in the network is dependent on: the number of devices in the network, the number of ports open on each device, the latency present in the network, and the method of scanning. For this implementation, the time to scan the smart plug averages to 6 seconds.

5. CONCLUSIONS

Many of the mitigations needed to prevent this type of attack are currently in place in newer version of the Android operating system. Google Play Protect prevents most points of initial infection of the mobile device. In situations where the user is using an older version of Android, anti-virus may be a viable solution for avoiding infection. To take the trust of the user protecting their device out of the equation, some sort of local area network protections should be put in place. A system that allows for VLANing devices on the network, while also maintaining functionality of the IoT devices is ideal for ensuring users do not attempt to circumvent these protections. Another potential mitigation could involve placing newly connecting wireless devices in a sort of sandbox to test for malicious activity before being placed on the real wireless network would provice a way to confirm all devices on a network are cleared for usage on that network. In the future, we would like to explore ways a framework like this can map out the vulnerabilities present in many common households. Many of the data available today only looks at IoT devices exposed to the Internet and does not paint a true picture of the vulnerabilities present in most environments. This use-case for the framework would use GPS features of the phone for mapping vulnerable devices to maps to show how wide spread this problem is. We would also like to explore different ways this sort of infection could spread to adjacent mobile devices. There is also work that could be done in more accurate fingerprinting through becoming the DHCP server on a local area network.

REFERENCES

Agadakos, e. a., Ioannis, ‘Location-enhanced authentication using the iot.’ in ‘Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC ’16,’ 2016 .

Android Dashboards, ‘https://developer.android.com/about/dashboards/index.html,’ . .

Bagci, e. a., Ibrahim Ethem, ‘Using channel state information for tamper detection in the internet of things.’ in ‘Proceedings of the 31st Annual Computer Security Applications Conference on - ACSAC 2015,’ 2015 .

BlueBorne, ‘https://www.armis.com/blueborne/,’ .

Broadpwn, ‘https://blog.exodusintel.com/2017/07/26/broadpwn/,’ . .

Chain, L. M. C. K., ‘https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html,’ . .

Droidjack Uses Side Load Backdoored Pokemon Go Android App, ‘https://www.proofpoint.com/us/threat-insight/post/droidjack-uses-side-load-backdoored-pokemon-go-android-app,’ . .

Earlence Fernandes, A. R. D. S. M. C. A. P., Justin Paupore, ‘Flowfence: Practical data protection for emerging iot application frameworks,’ in ‘www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/fernandes,’ 2016 .

Expensivewall: A Dangerous Packed Malware On Google Play That Will Hit Your Wallet, ‘https://blog.checkpoint.com/2017/09/14/expensivewall-dangerous-packed-malware-google-play-will-hit-wallet/,’ . .

Eyal Ronen, A. S., Colin O' Flynn and Weingarten, A.-O., ‘Iot goes nuclear: Creating a zigbee chain reaction,’ 2016 .

Galluscio, M., Neshenko, N., Bou-Harb, E., Huang, Y., Ghani, N., Crichigno, J., and Kaddoum, G., ‘A first empirical look on internet-scale exploitations of iot devices,’ 2017 .

Gartner IoT, ‘https://www.gartner.com/newsroom/id/3165317,’ . .

GinMaster: A case study in Android Malware, ‘https://www.virusbulletin.com/uploads/pdf/conference/slides/2013/yu-vb2013.pdf,’ . .

Introducing Google Play Protect, ‘https://www.android.com/play-protect/,’ . .

IoT Ransomware, ‘http://icitech.org/wp-content/uploads/2016/04/icit-brief-combatting-the-ransomware-blitzkrieg2.pdf,’ . .

Kambourakis, e. a., Georgios, ‘The mirai botnet and the iot zombie armies,’ in ‘MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM),’ 2017 .

Karbab, e. a., Elmouatez Billah, ‘Cypider,’ in ‘Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC ’16,’ 2016 .

Kim, e. a., Jun Young, ‘Automated analysis of secure internet of things protocols.’ in ‘Proceedings of the 33rd Annual Computer Security Applications Conference on - ACSAC 2017,’ 2017 .

Krentz, C. M., Konrad-Felix, ‘Handling reboots and mobility in 802.15.4 security,’ in ‘Proceedings of the 31st Annual Computer Security Applications Conference on - ACSAC 2015,’ 2015 .

Marin, e. a., Eduard, ‘On the (in)security of the latest generation implantable cardiac defibrillators and how to secure them.’ in ‘Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC ’16,’ 2016 .

Maulana, M. A., ‘Protecting privacy of ble device users protecting privacy of ble device users.’ 2016 .

Nicola Dragoni, A. G. and Mazzara, M., ‘The internet of hackable things,’ 2017 .

Nmap, ‘https://nmap.org,’ . .

Reverse Shell, ‘https://www.sans.org/reading-room/whitepapers/covert/inside-out-vulnerabilities-reverse-shells-1663,’ . .

Ronen, E. and Shamir, A., ‘Extended functionality attacks on iot devices: The case of smart lights,’ 2016 .

Sachidananda, V., Siboni, S., Shabtai, A., Toh, J., Bhairav, S., and Elovi, Y., ‘Let the cat out of the bag: A holistic approach towards security analysis of the internet of things,’ 2017 .

Seungjoo Kim, P. F., ‘Ddos attack on dns using infected iot devices.’ in ‘www.slideshare.net/skim71/ddos-attack-on-dns-using-infected-iot-devices,’ 2015 .

Shi, e. a., Yang, ‘An ultra-lightweight white-box encryption scheme for securing resource-constrained iot devices.’ in ‘Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC ’16,’ 2016 .

Valente, J. and CÃČÂąrdenas, A. A., ‘Using visual challenges to verify the integrity of security cameras.’ in ‘Proceedings of the 31st Annual Computer Security Applications Conference on - ACSAC 2015,’ 2015 .

Vijay Sivaraman, D. E., Dominic Chan and Boreli, R., ‘Smart-phones attacking smart-homes,’ 2016 .

Wing, J. M. and Zhao, D., ‘Opening keynote: Crashing drones and hijacked cameras: Cybertrust meets cyberphysical.’ in ‘2016 29th IEEE International System-on-Chip Conference (SOCC),’ 2016 .

Yang, e. a., Lei, ‘A multi-cloud based privacy-preserving data publishing scheme for the internet of things.’ in ‘Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC ’16,’ 2016 .

Zand, e. a., Ali, ‘Know your achilles’ heel.’ in ‘Proceedings of the 31st Annual Computer Security Applications Conference on - ACSAC 2015,’ 2015 .

VITA

Sammie Lee Bush received his B.S. in Computer Science from Missouri University of Science and Technology in May 2017. In July 2018, he also received his M.S. in Computer science from Missouri University of Science and Technology. His emphasis throughout each program was Cyber Security. During his undergraduate career, Sammie acted as chair of the Association for Computing Machinery Local Chapter SIG-Security from 2015 to 2016 and president of the Association for Computing Machinery Local Chapter from 2016 to 2017. Sammie was also a recepient of the NSF's Cybercorps: Scholarship for Service from his Senior year of his undergraduate degree through the end of his M.S.