A SYSTEMATIC APPROACH TO PRIORITIZE VULNERABILITIES IN IOT DEPLOYMENTS

A Dissertation
Presented to
The Academic Faculty

By

Omar Alrawi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Engineering
Department of Electric and Computer Engineering

Georgia Institute of Technology

May 2023

A SYSTEMATIC APPROACH TO PRIORITIZE VULNERABILITIES IN IOT DEPLOYMENTS

Thesis committee:

Dr. Emmanouil K. Antonakakis
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mustaque Ahamad
College of Computing
Georgia Institute of Technology

Dr. Fabian Monrose
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Roberto Perdisci
Computer Science
University of Georgia

Dr. Douglas Blough
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Michael Bailey
School of Cybersecurity and Privacy
Georgia Institute of Technology

Date approved: April 18, 2023

ACKNOWLEDGMENTS

First and foremost, I would like to express my profound gratitude to the Almighty for bestowing upon me knowledge and wisdom like that of Prophet David and Solomon, as mentioned in the Quran, chapter 27, verse 15: "And We had certainly given to David and Solomon knowledge, and they said, 'Praise [is due] to Allah, who has favored us over many of His believing servants'." I want to thank the Ph.D. committee, whose mentorship, support, and constructive feedback have been instrumental in shaping my research and refining my dissertation. I am grateful for their thoroughness during my defense and for setting high expectations to push me to my full potential. I am deeply grateful to my advisor, Professor Manos Antonakakis, who has been my number-one fan throughout this journey. Your unwavering faith in me and your constant encouragement provided the impetus I needed to persevere through the challenges of completing my Ph.D.

I am indebted to Professor David Mohaisen, who introduced me to research, mentored me, and convinced me to pursue a Ph.D. As a close friend, I have admired your unmatched productivity. I aspire to attain the level of brilliance and dedication that you embody. A heartfelt acknowledgment goes out to Professor Brendan Saltaformaggio, a close friend and mentor. I am very grateful for your willingness to guide me through the intricacies of research and academia. Your generosity in sharing resources, time, and friendship has been a true gift. I owe a special gratitude to Professor Fabian Monrose, who opened my eyes to what it truly means to be a scientist. Your rigorous approach, wisdom, and experience have shaped me as a researcher. I have genuinely enjoyed every moment spent working with you and receiving your invaluable feedback.

To my fellow lab mates, Chaz, Gong, Kevin V, Panos, Rosa, Thanos[1], Thanos[2], Thomas, Yizheng, and Zane, thank you for your support and camaraderie. A special thanks to Chaz, whose mentorship has been invaluable throughout my journey. I am grateful to the numerous outside collaborators from industry and other academic institutions. Your partnership

has been an essential part of my growth and success.

I would also like to express my heartfelt gratitude to my loving mom, sisters, and in-laws, who have been my pillar of strength during these years. Your sacrifices and love allowed me to concentrate on my studies. My sincere appreciation extends to your understanding and support. To my incredible wife, Shireen, and our lovely children, Bayan, Bekr, Belal, and Badr, thank you for your unwavering support, love, and understanding. Your love and patience have sustained me during the most challenging times to achieve this milestone. Thank you, my precious Bayan, for smiling at me from the front-row seat during my first paper presentation and melting my stage fright away. Finally, I am thankful to everyone who has contributed to my journey in big and small ways. This dissertation is not only a testament to my efforts but also a reflection of the collective efforts of those who have touched my life throughout this incredible experience.

Thank you all!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

C&C Command and Control

CDN Content Delivery Network

COTS commercial off-the-shelf

CVE Common Vulnerabilities and Exposures

CVSS Common Vulnerability Scoring System

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

EOL End-of-Life

ICS Industrial Control System

IoT Internet of Things

ISPs Internet Service Providers

LAN Local Area Network

MITM Man-in-the-Middle

NAT Network Address Translation

NIST National Institute of Standards and Technology

OS Operating System

UPnP Universal Plug and Play

VLAN Virtual Local Area Network

SUMMARY

Practitioners must use an informed approach that prioritizes security measures based on risk and cost to improve the security of deployed Internet of Things (IoT). For several reasons, improving the overall security of IoT is essential. First, IoT devices are widely used in various sectors, such as healthcare, transportation, and finance, and often handle sensitive data, making them a prime target for cybercriminals. Second, many IoT devices have limited computing resources, making it challenging to implement robust security measures. This limitation can expose IoT devices to several vulnerabilities, making them easier targets for malicious attacks. Third, many IoT devices are interconnected and communicate with each other, increasing the attack surface for cybercriminals. An attack on one IoT device could compromise the entire network, causing significant harm to the system and its users. Therefore, improving the overall security of IoT deployments is crucial to ensure data privacy, protect against cyberattacks, and maintain the integrity and availability of critical services.

To do so, practitioners require an approach to assess the security of IoT deployments comprehensively, standardize the security assessment across diverse devices, and prioritize security measures for high-risk devices. A comprehensive security evaluation can identify the most significant security threats and vulnerabilities in IoT deployments. This will help practitioners focus on the most critical areas that require the highest priority. Standardizing security assessment will help reduce costs associated with customizing security evaluation for each device and make it easier to compare security across diverse devices. Finally, prioritizing security measures provides a more strategic approach to protecting high-risk devices vulnerable to cyberattacks. Toward improving overall security, *this dissertation contributes a systematic and replicable approach that better prioritizes vulnerabilities in IoT deployments.*

CHAPTER 1
INTRODUCTION

Insecure IoT deployments can cause severe operational issues for critical internet infrastructure, including downtime, data breaches, network congestion, malware propagation, and control system manipulation [1]. Understanding the risk of deploying IoT systems is complex. For example, operators must enumerate their assets, identify vulnerabilities, correlate cyber threats that target these vulnerabilities, and prioritize their resources to protect the most critical IoT deployments. More problematic, IoT deployments often consist of legacy and newer devices, which can be challenging to secure. Legacy devices may not have the latest security features and may be more vulnerable to modern threats. Furthermore, patching legacy devices may be difficult or more costly than replacing them. Many IoT deployments are dynamic and evolve in various aspects through updates, end-of-lifecycle, and changes to their network infrastructure and topology. These changes can cause them to be misconfigured and vulnerable to cyber threats. These compounding complexities overwhelm operators, and it becomes difficult to prioritize what IoT deployments security issues should be addressed quickly. Therefore, assessing the risk of IoT deployments is essential to prioritizing security measures and involves comprehensive, iterative, and large-scale efforts.

1.1 Thesis and Contributions

1.1.1 Thesis Statement

A comprehensive and iterative vulnerability and threat analysis of IoT deployments improves risk assessment models for prioritizing security measures.

1.1.2 Contributions

Prior to my work, risk assessments for IoT deployments focused on custom security evaluations, lacked iteration, and did not consider internet threats. Prior approaches paint a partial picture of the potentially exploitable security issues that exist in IoT deployments leaving gaps in the overall risk assessment. This is far from sufficient, as the modern internet threats evolve and the pace of new IoT applications are rapidly growing while custom security assessment techniques cannot keep up. For example, an IoT vacuum cleaner and a home assistant device may use an embedded Linux operating system, but the software services on the devices will be different. A smart vacuum may expose a web server that accepts commands locally from a mobile application to initiate cleaning tasks, while a voice assistant may accept commands from several remote cloud endpoints to service requests from users: each requiring a customized security evaluation approach for the different type of software services and deployment. In addition, as the devices age and their threats evolve, the device risk levels change over time. There have been several large-scale studies that apply theoretical-based approaches to simulate these complexities, but they do not accurately represent real-world IoT deployments.

My research aims to break from traditional approaches and instead develop a systematic and replicable approach based on end-host binary program analysis and network vulnerability analysis for real-world IoT deployments. In this dissertation, I present three studies that combine binary program analysis and network vulnerability analysis to investigate the fundamental security problems that contribute to high-risk IoT deployments. Specifically, this body of work is built upon systematic methodologies that allow others to reproduce, verify, and extend. *The overall goal of this dissertation is to combine real-world security evaluation observations and malware threat analysis to quantify the risk and prioritize security measures for IoT deployments.* Below, I will briefly introduce these studies and their contributions.

1.1.3 Security Evaluation of Home-based IoT Deployments

This work proposes a component-based security evaluation framework that provides a comprehensive and standardized assessment method for IoT deployments. The framework decomposes complex IoT deployments into manageable core components and uncovers the breadth of the attack surface. Leveraging this framework, we build a test bed of 45 diverse home-based IoT devices and assess their security properties for device services, cloud services, companion mobile applications, and network protocols. We limit the assessment scope to home-based IoT devices because they are readily available and the experimental setup can be reproduced. The results show an IoT deployment can have multiple components, each with vulnerabilities. By systematically combining end-host binary program analysis and vulnerability analysis, we discover that certain device types or vendors can have disproportionate vulnerabilities affecting specific components. Holistically studying these vulnerabilities is essential to understand the potential risks associated with an IoT deployment.

1.1.4 A Longitudinal Security Measurement of Home-based IoT Devices

A systematic security evaluation must consider the temporal component of an IoT deployment by incorporating an iterative approach that can account for vulnerability changes over time. In this study, we conduct 13-month longitudinal assessments to understand how security flaws evolve throughout the device's lifecycle. The longitudinal component enables a more comprehensive understanding of vulnerabilities as they arise and uncovers trends that are otherwise impossible to observe. This study reinforces the replicable approach proposed in the first work and contributes new findings. The results reveal additional vulnerabilities missed in the initial one-time security evaluation. Moreover, the findings evaluate the effectiveness of security patching for different device types and vendors. Lastly, the results highlight groups of vulnerability types by severity and their persistence throughout the lifecycle of an IoT device, which can inform the risk analysis and prioritization of

security measures.

1.1.5 A Large-Scale Analysis of The IoT Malware Lifecycle

Toward providing a more accurate risk assessment of IoT deployments, we study the malware threats that target vulnerable IoT deployments. In this work, we investigate attacks on IoT systems at scale by empirically studying the lifecycle of IoT malware and comparing it with traditional malware that targets desktop and mobile platforms. We build an extensible binary program analysis platform for six Linux-based system architectures and characterize IoT malware infection tactics, their infection payloads, and their capabilities. We leverage the binary analysis platform to carry out a large-scale measurement of more than 166K Linux-based IoT malware samples collected over a year. The results contribute new observations about the IoT malware lifecycle, including device targeting, persistence techniques, abuse tactics, and Command and Control (C&C) communication operation. Specifically, the results identify areas of an IoT deployment that may be particularly vulnerable to attacks. Furthermore, the results discover trends similar to the development of traditional malware for desktop and mobile platforms, which inform the potential evolution of future threats. These findings help more accurately characterize the threats targeting IoT deployment and provide a more informed risk assessment. Lastly, we make our binary analysis platform and the malware dataset publicly available for the research community to encourage reproducibility and validation.

1.2 Dissertation Organization

The dissertation is organized as follows:

- **Chapter 1** presents the challenges in risk assessment and security measure prioritization for IoT deployments. Next, the chapter briefly highlights the gaps in prior works, which do not provide a holistic understanding of risks in IoT deployments. Finally, the chapter overviews the dissertation's contribution and describes three works

that build upon one another to provide a better risk assessment and security measure prioritization of IoT deployments. The chapter closes by providing an outline of the dissertation.

- **Chapter 2** provides the foundational work required to design and evaluate systematic and replicable methods for security evaluation and malware analysis of IoT deployments. First, the chapter systematizes prior work in the security evaluation of home-based IoT deployments. The findings guide the design and inception of a novel security evaluation framework, which later chapters use. Next, the chapter systematizes malware lifecycle studies for traditional platforms like desktop and mobile. The findings inform the design of a systematic and comparative lifecycle framework tailored to modern IoT malware and encompass traditional malware threats.

- **Chapter 3** uses the derived framework from the foundational chapter to evaluate a large-scale and diverse testbed of home-based IoT devices. First, the chapter describes the security evaluation framework and its components. We formalize the framework into a model that allows others to replicate and generalize to different IoT deployments. Next, the chapter presents the testbed design and engineering efforts, including their challenges. This includes the lab setup, evaluation tools, automation, data collection, and analysis. Finally, we present the results and notable findings that highlight the utility of our framework. Specifically, we provide three case studies based on our large-scale evaluation showcasing good, satisfactory, and poor IoT deployments.

- **Chapter 4** presents a temporal security evaluation using the framework from the previous chapter to reinforce reproducibility and uncover new vulnerabilities as they arise. Specifically, we evaluate the security of the device component from the home-based IoT testbed. We describe our experimental setup and data collection using the automated tools built by our first study. Finally, we present the results showing

changes in the vulnerabilities that impact IoT deployments and uncover insightful observations about vendors, device types, and severity of vulnerabilities. We then show how these findings impact the results from the initial security evaluation using two case studies, which informs the risk modeling.

- **Chapter 5** presents a large-scale study of the IoT malware lifecycle. Using the framework derived in the foundational chapter, we empirically evaluate how IoT malware infect, persist, abuse, and communicate on IoT devices. Next, we use the findings to inform the risk of attackers targeting vulnerabilities found in IoT deployments. Moreover, we compare our findings with traditional malware and discuss the future evolution of IoT malware, which helps prioritize security measures and more accurately characterizes risk.

- **Chapter 6** combines the empirical results and proposes a module, risk-scoring model. We derive a simplified risk assessment approach based on the Common Vulnerability Scoring System (CVSS) proposed by the National Institute of Standards and Technology (NIST). However, unlike CVSS, the scoring methodology provides a module risk score that incorporates the components, time, and threats targeting IoT deployments. We demonstrate the risk assessment by presenting two case studies. Both case studies empirically show a more accurate risk characterization than less informed approaches. Together, the module scoring and holistic approach provide a better prioritization of security measures.

- **Chapter 7** summarizes the key findings from each study and lays out future work for improving the overall risk analysis process. Specifically, we highlight additional studies that would provide a more in-depth analysis of the security and threats in IoT deployments. These studies include a longitudinal study into how attackers co-op and abuse infected IoT deployments, a large-scale analysis of IoT cloud backends availability and security, a comprehensive study of the third-party software supply-

chain used by IoT deployments, and toward automating the security evaluation of low-energy network protocols found in IoT deployments. Beyond the risk assessment, we highlight our work's broader impact on the field.

CHAPTER 2
FOUNDATIONAL WORK

This chapter establishes the foundation of prior knowledge to which the dissertation will contribute. Specifically, we present two systematizations. The first systematization examines the security evaluation literature for attacks and defenses against smart-home IoT systems. The second systematization looks at malware threats for traditional platforms like desktop and mobile to compare to IoT malware. Finally, we present a related work section differentiating the contribution of this dissertation from prior efforts. The systematized literature is chosen based on the following criteria:

- **Merit**: The work is unique and among the first to explore a given security predicament.
- **Scope**: The work focuses on the security (offensive and defensive) of home-based IoT systems or malware analysis.
- **Impact**: The work is regarded as significant based on the number of citations.
- **Disruption**: The work uncovers a new area the community is investigating.

2.1 IoT Security Evaluation

We noticed four broad categories of approaches as we surveyed the literature for security evaluation of IoT deployments. These approaches include security assessment of IoT devices through device services, cloud backends, companion mobile apps, and network communication. Using the same differentiae, we organize the literature based on those four IoT components: the device, the cloud backend, the companion mobile application, and the network communication. This organization allows us to understand each component's attack

techniques, proposed mitigations, and stakeholder responsibilities. Table 2.1 presents an overview of the systematized work and their corresponding subsections where we discuss the literature in detail. The component classification highlights the focus of the work while the attack vectors, mitigations, and stakeholders identify the approach. The systematization highlights representative work; hence it does not provide an *all-encompassing* reference to every related work.

2.1.1 Device

Most of the home-based IoT research focuses on the device because the device component is the cornerstone of an IoT deployment.

Attack Vector

Several works ([2, 3, 4, 5]) explored IoT device configuration insecurities. Barnes [2], building on the findings of Clinton et al. [3], demonstrated how exposed hardware pins on a device allowed him to gain privilege access and spy on the end-users. Insecure configurations combined with weak or a lack of authentication can exacerbate the problem as shown by Chapman [6] and Rodrigues [7]. Weak or a lack of authentication in running services is a key contributor to several documented attacks [8, 9, 10, 11]. These attacks demonstrate that device setup and configuration is an important process that the vendor must consider and evaluate for security flaws. Vendors should enforce strict authentication policies and for end-users to configure the device before allowing it to operate.

Table 2.1: Systematization of the IoT literature. Each section corresponds to a component of IoT deployment across attack vectors, mitigations, and stakeholders. The ✓implies the category of attack, mitigation, or stakeholder applies to the discussed literature.

Component	Ref	Attack Vector			Mitigations		Stakeholders	
		Vuln. Services	Weak Auth	Default Config	Patching	Framework	Vendor	End User
Device subsection 2.1.1	Ur13 [4]			✓	✓		✓	
	Costi14 [12]	✓			✓		✓	
	Chapm14 [6]		✓	✓	✓		✓	
	Kaval14 [11]	✓	✓	✓	✓		✓	✓
	Wuess15 [5]			✓	✓		✓	
	Rodri15 [7]		✓	✓	✓		✓	
	Lodge16 [13]	✓			✓		✓	
	Ike16 [3]			✓	✓		✓	
	Franc16 [14]	✓			✓		✓	
	O'Fly16 [15]	- -	- -	- -	- -	- -	- -	- -
	Ferna16 [16]	✓			✓		✓	
	Max16 [8]	✓	✓	✓	✓		✓	
	FlowF16 [17]	✓		✓	✓	✓	✓	
	Oberm16 [10]	✓	✓	✓	✓		✓	
	Barne17 [2]			✓	✓		✓	
	Herna17[18]	✓			✓		✓	
	Morge17 [19]	✓			✓		✓	
	Ferna17 [20]	✓		✓	✓		✓	
	Ronen17 [21]	✓			✓		✓	
	Dolph17 [22]	✓			✓		✓	
	Tian17 [9]	✓	✓		✓	✓	✓	✓
	Wang18 [23]	- -	- -	- -		✓	✓	
		Permissions	Programming	Data Protection				
Mobile Application subsection 2.1.2	Barre10 [24]	✓			✓		✓	
	Au12 [25]	✓			- -	- -	✓	✓
	Egele13 [26]		✓	✓		✓	✓	
	Vienn14 [27]		✓	✓	- -	- -	- -	- -
	Max16 [8]		✓	✓	✓		✓	
	Sivar16 [28]	✓		✓		✓	✓	✓
	Demet17 [29]	✓		✓		✓		✓
	IoTFu18 [30]		✓		- -	- -		✓
		Vuln. Services	Weak Auth	Encryption				
Cloud Endpoint subsection 2.1.3	Max16 [8]	✓	✓		✓		✓	
	Oberm16 [10]		✓	✓	✓		✓	
	Nandi16 [31]	✓				✓		✓
	Blaic16 [32]	✓	✓	✓	✓		✓	
	Wilso17 [33]			✓		✓	✓	✓
	Surba17 [34]	✓			- -	- -	✓	✓
	DTAP18 [35]	✓	✓	✓		✓	✓	✓
		Encryption	MITM					
Communication subsection 2.1.4	BEAST11 [36]	✓			✓		✓	
	Garci11 [37]	✓	✓		✓	✓	✓	
	LUCKY13 [38]	✓			✓		✓	
	Ryan13 [39]	✓	✓		- -	- -	- -	- -
	Foula13 [40]	✓	✓		- -	- -	- -	- -
	Alfar13 [41]	✓			✓		✓	
	Selvi14 [42]	✓			✓		✓	
	POODL14 [43]		✓		✓		✓	
	FREAK15 [44]	✓			✓		✓	
	CRIME15 [45]		✓		✓		✓	
	SMACK15[46]	✓	✓		✓		✓	
	Adria15 [47]	✓	✓		✓		✓	
	Zilln15 [48]	✓	✓		- -	- -	- -	- -
	DROWN16 [49]	✓	✓		✓		✓	
	Jasek16 [50]		✓		✓		✓	
	Kinti16 [51]	- -	- -		✓			✓
	Aptho17 [52]	✓			✓			✓
	Wood17 [53]	✓				✓		✓

Max [8] assessed the security of the August Smart Lock and found that weak authentication and insecure default configuration broke the security of the lock. He found hard-coded credentials and debug configurations that allows modification and introspection of the lock. The work of Obermaier et al. [10] on cloud-based cameras found that although the device had what appeared to be a strong password (36 characters of alphanumeric and symbols), the password was the *MAC* address of the camera reversed and *Base64* encoded. Kavalaris et al. [11] showed that the Sonos device runs undocumented and unauthenticated services on high ports allowing LAN clients to fully control the device. The Sonos device was susceptible to unauthorized device pairing due to the lack of authentication. SmartAuth [9] found that the authentication problem also manifests itself in the IoT application platforms through over-privileged applications. Device pairing establishes a trusted channel between a client and their device. Further, IoT hubs bridge LE devices to IP networks, which have a pre-established trust relationship. An attacker would exploit this specific process to circumvent the device or use it as a pivot point.

IoT application platforms expose a permission-based model to allow third-party applications to run. Fernandes et al. [16, 17, 20] showed how implicit trust to third-party applications can have major implications on the security of the device. There are many subcomponents within the device's platform, which can make securing the device difficult. Many vendors have good practices in place to ensure secure authentication and secure default configurations (as demonstrated by O'Flynn [15]), but core device services can suffer from side-channel information leakage. Ronen et al. [21] showed that although the Philips Hue device was reasonably secure, they were able to extract the master encryption key through a side-channel attack and combine it with a vulnerability found in the communication protocol, which resulted in a wormable exploit.

Flaws in firmware allow attackers to steal WiFi credentials [13], turn smart thermostats into spy gadgets [18], ransom them [14], run arbitrary commands on smart TVs [19], and control home assist devices covertly [22]. Costin et al. [12] conducted a large-scale study

on firmware analysis and found an array of flaws. The literature showed that device security requires defensive approaches to secure side-channel, firmware, and hardware. The toolchain for software and hardware development has a well-defined secure development process that vendors must utilize.

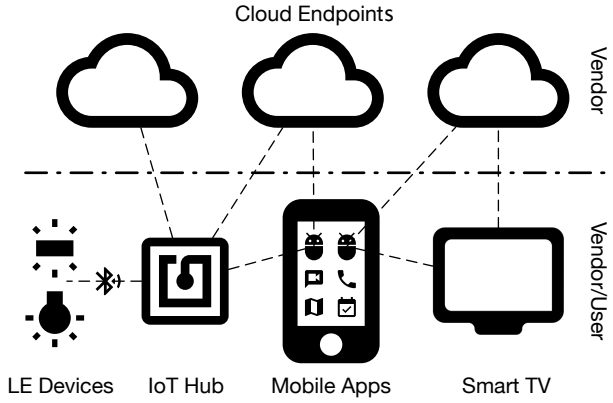
Mitigations

To address vulnerable services, misconfiguration, and weak authentication, vendors patch through device updates, while inherent design flaws in IoT platforms are mitigated through new frameworks. Wang et al. [23] proposed a provenance-based framework to aggregates device activities across a deployment that can detect errors and malicious activities. SmartAuth [9] is a framework that identifies required permissions for IoT applications running on platforms like SmartThings and Apple Home. FlowFence [17] is a framework that splits application codes into sensitive and non-sensitive modules and orchestrates the execution through opaque handlers. This approach burdens developers because they must be mindful of what code operates on sensitive and non-sensitive data. Furthermore, researchers can adapt techniques found in mobile application frameworks to address IoT platform insecurities.

Stakeholders

Table 2.1 shows that the main stakeholder is the vendor. Vendors are responsible for patching and updating vulnerable devices but can delegate some of the responsibilities to users through configurations. For example, users can mitigate insecurities by disabling problematic services on the device. SmartAuth [9] provides a derived authentication approach for applications on the device, but the implementation must be done by the vendor. Users gain control by having a choice about what permissions to authorize for third-party applications. Kavalaris et al. [11] showed how services that the Sonos device exposes create a security risk. Users can mitigate this risk through network segmentation, but it requires

Figure 2.1: Typical home-based IoT setup.



some technical expertise.

Not many devices allow users to fully configure running services or even disable them unless they have privileged access. Based on all the proposed mitigations, end-users can manage configuration or network segmentation residing on the home demarcation side as shown in Figure 2.1. End-users do not have much control and often are given a minimalistic interface, which limits the mitigation of vulnerable services. Vendors, on the other hand, bear the responsibility for keeping the device up to date.

Take Away

The literature addresses some aspects of device security. Devices have many components that contribute to their overall security like the platform permissions, unauthenticated services, insecure configurations, and software and hardware bugs. Further, they are amplified when combined. The device security is not purely in software, but vulnerabilities manifest themselves in hardware and side-channels as well. Embedded Linux is found in many of the devices, but there is no secure open IoT platform, which can incorporate newly proposed frameworks [23, 17, 9] by the community.

System patching addresses most of the vulnerabilities. The patching process is not perfect [18] and can be improved by good practices implemented in other areas of computing [54]. The end-users have almost no control or visibility into the operation of the

device. Securely providing health telemetry and fine-grained configuration parameters can empower users to mitigate immediate risks. Users can deploy the device in many ways that go beyond the vendor's permissive assumptions, hence vendors should assume the device is Internet-facing when building security measures.

Similar problems are faced with general purpose computing systems that are publicly accessible and running vulnerable services or using weak authentication (SSH with guessable password). Adapting techniques from secure platforms and operating systems will improve the security posture of many IoT devices.

2.1.2 Mobile Application

Many of the home-based IoT devices have a companion mobile application to control, configure, and interface with the device. Mobile applications can be leveraged as an attack surface against IoT deployments.

Attack Vector

Acar et al. [55] identified five different areas of Android mobile application issues, namely permission evolution, permission revolution, webification, programming-induced leakage, and software distribution. We adapted Acar's approach and identified three major classes of insecurities that effect IoT devices: over-privilege (permissions [24, 25]), programming errors (programming [26]), and hard-coded sensitive information (data protection [27]). Max [8] showed how programming errors leak sensitive information about the device and the cloud endpoint. Max used the sensitive information to dump credentials, escalate privileges, and circumvent the security of the August Smart Lock. Apart from Max's work, there are no **direct** attacks leveraging the mobile application to circumvent an IoT device.

Chen et al. [30] presented IoTFuzzer that instruments the mobile application within an IoT deployment to find bugs on the IoT device. Chen's approach is unique and leverages the semantics that the vendor programmed into the application. Although there are no re-

ports of this technique used in the wild, theoretically an attacker can use the same approach to escalate privilege on an IoT device. Sivaraman et al. [28] showed how a mobile application can be used on a local network to collect information about available home devices and then reconfigure the router/modem firewall rules to make the devices Internet facing. Hanguard [29] showed how permissive security assumptions by vendors about the LAN can expose an IoT device. Companion mobile applications are an entry point to the device and vendors often assume that the deployment network is trusted and secure. These assumptions can have grave effects on the security of the device especially when devices rely on unauthenticated services or unencrypted communications.

Mitigation

Hanguard [29] proposed a user-space mobile application that interfaces with the router to control access through role-based access control (RBAC). Hanguard's approach will prevent the attack discussed by Sivaraman et al. [28] but cannot stop attacks from a compromised *companion* application. Securing the mobile application by adhering to best practices discussed in Pscout [25], Barrera et al. [24], Egele et al. [26], and Viennot et al. [27], reduces the attack surface. Unfortunately, as Viennot et al. [27] showed, a large portion of the applications in the Google Play Store contain issues relating to permissions, programming errors, and information leakage. Mobile application platforms are mature and have built-in security facilities to promote good practices. Developers and vendors should adhere to best practices and audit their mobile applications periodically.

Stakeholders

The mobile application component relies on both the user and the vendor. This is partly due to the permission model that most mobile platforms provide to end-users. Hanguard [29] provides the user with a system to deploy inside the local network through routing rules (user demarcation Figure 2.1), which does not involve the vendor. Sivaraman et al. [28]

proposes that users should be vigilant when running mobile applications on their networks and only use authorized stores (Google Play, Apple App Store, etc.). The vendors must address programming errors and secure information storage through updates. Vendors must familiarize themselves with the mobile platforms to deploy secure applications or use a reputable third-party developer to provide secure development expertise.

Take Away

The work of Acar et al. [55] showed the maturity of the mobile application security field. An inherent trust is given to mobile applications, which in many cases control core components of an IoT device or a cloud service. Max [8] and IoTFuzzer [30] demonstrated how to abuse the implicit trust between mobile applications and IoT devices or cloud services. IoT vendors and developers should adhere to platform development guidelines and leverage security features to ensure proper deployments. Limiting mobile application access to the device through fine-grained controls is a promising direction that can reduce the attack impact. Lastly, Hanguard's [29] approach should be further investigated to provide end-users with control to mitigate risks.

2.1.3 Cloud Endpoint

Cloud endpoints are the Internet components of the IoT deployment, and in a sense, they define what IoT is. They provide core services like remote administration, alerts, and digital content. The IoT devices and their mobile applications trust these cloud endpoints, which gives adversaries an additional attack point.

Attack Vector

The attack by Max [8] is a great example that touches on all components of the IoT ecosystem. The attack discovered insecure application program interface (API) on the cloud endpoint for the August Smart Lock, which escalated a guest account to an administra-

tor account. Blaich [32] audited the Wifi Barbie doll for various vulnerabilities and found that the cloud endpoints did not authenticate firmware downloads, had multiple cross-site-scripting vulnerabilities, allowed username enumeration, had no brute force limiting, and issued never expiring cookies. Obermaier et al. [10] audited the cloud endpoints of surveillance cameras and showed that an attacker can inject footage, trigger false alarms, and carry out a denial-of-service attack against the camera system. These attacks were possible due to vulnerabilities introduced in the configuration of the infrastructure, vulnerable services, and insecure APIs. Zuo et al. [56] leveraged client-to-cloud trust to implement AutoForge, which forges requests from the mobile applications to the cloud endpoints enabling password brute-forcing, password probing, and security access token hijacking. Implicit trust between IoT components is sensitive and vendors must verify endpoints before allowing them unfettered access.

IoT integration platforms, like IFTTT [57], automate.io [58], and CloudWork [59], are third-party cloud endpoints. They use OAuth tokens to connect multiple IoT devices to perform user programmed tasks. Surbatovich et al. [34] studied the security implications on privacy and integrity when using recipes[1] and showed that some recipes can allow attackers to distribute malware and carry out denial-of-service attacks. Nandi et al. [31] reported a similar type of user-induced programming error through trigger-action programming (TAP), which led to an incorrect event triggering or a lack thereof. Fernandes et al. [35] pointed out that the cloud integration platforms can be compromised, which might expose the user's OAuth tokens publicly. These scenarios are likely to happen based on recent platform compromises like Equifax [60] and Orbitz [61]. The work of Wilson et al. [33] did not identify an attack vector on the IoT ecosystem, but it studied the privacy and trust that users place with IoT vendors. These attacks show that cloud integration services lack fine-grained control and they leak private and sensitive information that can lead to a breach.

[1] recipes are high-level programmable instructions that are used to trigger IoT device actions based on an occurrence of an event.

Mitigation

To mitigate these attacks, Max [8], Obermaier et al. [10], and Blaich [32] recommend proper configuration and secure authentication mechanisms. Surbatovich et al. [34] offered a framework to analyze the cloud platform recipes, which motivated later work. Nandi et al. [31] proposed an automatic trigger generation system that analyzes user-defined triggers for errors and rectifies them by rewriting the triggers. Fernandes et al. [35] proposed the use of a decentralized framework for trigger-action programmable platforms called DTAP. The DTAP platform is a shim between the IoT cloud platform and the user's local network and brokers access to the IoT devices based on transfer tokens (XTokens). The mitigation techniques include securing cloud endpoints, offering tools to analyze third-party integration services, assisting developers in generating correct triggers for their applications, and providing short-lived tokens with constrained access to a device's functions.

Somewhat related, Wilson et al. [33] looked at empowering IoT users that trust the vendors with their private data. The technique is known as TLS-Rotate and Release (TLS-RaR), which requires an auditor entity collecting TLS packets to request the session key from the vendor to decrypt the communication. The vendor then rotates the TLS session key and discloses to the auditor the prior key to decrypt the collected TLS packets. The audit system must be deployed on the end-user demarcation side and collects traffic for devices that they wish to audit.

Stakeholders

The vendor controls the cloud endpoints and the users do not have a way to inspect or control what their device sends to the cloud endpoints [53, 62]. Additionally, third-party cloud providers offer infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS) to IoT deployment. Many of the IoT devices rely on cloud-based infrastructure to run their services. Unplanned outages[63], infrastructure compromises[64], and intentional attacks[65] impact the deployment of the cloud endpoints. When it comes to cloud infrastructure con-

figuration and API implementation ([8, 32, 10]), the vendor is responsible for the mitigation of the vulnerabilities.

Newer IoT devices are taking advantage of managed IoT platforms, which offload much of the security responsibilities to the public cloud providers. On the other hand, the majority of the proposed frameworks ([31, 35, 33]) are user-centric and give end-users visibility and control in a limited way. The work by Fernandes et al. [35] and Wilson et al. [33] is a hybrid approach and can be deployed jointly by vendors and users or by a trusted third-party. As for cloud providers, the vendor can mitigate their exposure by diversifying and over subscribing to different cloud providers.

Take Away

IoT cloud endpoints exhibit insecure cloud deployment through configuration and API implementation, but these vulnerabilities can be addressed with readily available tools for cloud security. Additional measurements are needed to further understand the extent of these misconfigurations in cloud deployments. The Censys Project [66] is a valuable source of data that can allow researchers to historically analyze IoT infrastructure. Further, the IoT cloud integration platforms introduce new challenges that mimic classical work like Decentralized Trust Management [67]. Integration cloud platforms offer users a way to chain multiple IoT devices to execute tasks based on an event, and they suffer from over-privilege recipes and privacy implications, which is demonstrated in the work of Surbatovic et al. [34].

Fernandes et al. [35] utilized prior techniques for the IoT cloud platforms by applying trust management systems and token authentication protocols to the IoT platforms. Vendors are adapting managed IoT cloud platforms, which shifts the security responsibility to cloud providers like Amazon IoT Core [68], Azure IoT Hub [69], and Google Cloud IoT [70]. IoT cloud endpoints are relying more on third-party infrastructure to deploy and run their services, which means vendors should consider a contingency plan for unplanned outages

and infrastructure compromises. Additional studies are needed to understand the managed IoT cloud platforms and what possible weaknesses exist.

2.1.4 Communication

Network communication in IoT deployment fall into two classes of protocols, Internet protocol (IP) and low-energy (LE) protocol. Both communications can exist on the user demarcation (see Figure 2.1) of the network, but only IP communication can go over the Internet. Researchers from industry and academia both are heavily invested in the security of network communication because of its applicability in other areas.

Most home-based IoT systems implement four types of communication protocols: IP, Zigbee, Z-Wave, and Bluetooth-LE (BLE). IoT devices choose to use the IP suite for communication due to its reliability and proven capability of transferring incredible volumes of global network traffic. The IP protocol is stateless and offers no security, but it can be supplemented by the use of TCP and TLS/SSL protocols to provide the security features needed. Based on the literature, we identified five popular application layer protocols that home-based IoT devices use, namely: DNS, HTTP, UPnP, NTP, and custom implementations.

Attack Vectors

The DNS protocol is a lightweight protocol that Internet services rely on, but inadvertently leaks private information based on the recursive and client configurations. Kintis et al. [51] found that open recursive DNS that enable EDNS Client Subnet feature (ECS) [71] (which embeds a truncated portion of the client's IP address) have privacy implications. Selvi [42] demonstrated how a MITM attack on NTP was used to bypass HTTP strict transport security (HSTS). The HTTP protocol gives a more reliable mode of transportation, but like DNS and NTP, it does not provide any confidentiality or integrity. Bellissimo et al. [72] and Samuel et al. [54] demonstrated how an insecure protocol like HTTP allows attackers

to MITM and backdoor the system software update process.

IoT devices widely rely on UPnP protocol to offer easy configuration and control. UPnP uses the HTTP protocol, hence inherits the same flaws [73]. Garcia [37] showed how attackers abuse UPnP because it lacks authentication, validation, and logging. GNUcitizen [74] demonstrated how an UPnP enabled device is vulnerable to cross-site scripting (XSS) vulnerabilities, while HD Moore [75] presented statistics and measurements around UPnP enabled devices on the Internet. Their work demonstrates that unauthenticated and unencrypted use of application layer protocols enables attackers to mass exploit devices, which leads to additional attacks. TLS/SSL sessions provide confidentiality and integrity, which help address the inherent flaws in these communication protocols.

Researchers have thoroughly examined the TLS/SSL protocols and uncovered severe vulnerabilities. Starting off in 2011, BEAST [36] exposed the initialization vector (IV) flaw in TLS 1.0, which allowed attackers to predict the IV of the next message in the stream. In 2012, CRIME [45] showed how TLS sessions that allow compression, like Google's SPDY protocol, were susceptible to session hijacking. In 2013, AlFardan et al. [38] used malformed packets to infer time delays, a side-channel attack, in the MAC verification to statistically infer the plaintext from the ciphertext. AlFardan et al. [41] also showed how the RC4 stream cipher weakens the security of TLS sessions. POODLE [43] exposed a downgrade flaw in SSL 3.0 that allowed for insecure communication between two parties. Beurdouche et al. [46] found flaws in several client and server implementations of TLS/SSL libraries that allow MITM attacks, including the FREAK [44] vulnerability.

Additional attacks disclosed by Adrian et al. [47] and DROWN [49] illustrated the difficulty of implementing secure communication protocols. Many IoT communications are susceptible to MITM attacks because they support older versions of TLS/SSL protocols. TLS/SSL is also widely used in managed IoT platforms to secure the communication channels. Emerging managed IoT platforms, like AWS IoT Core [68], Azure IoT Hub [69], and Google Cloud IoT [70], implement custom protocols that utilize certificates and TLS/SSL.

These protocols and platforms are sparsely documented but rely on time-tested technologies to implement secure end-to-end communication.

The BLE [76], Zigbee [77], and Z-Wave [78] protocols have many security problems. Ryan [39] showed a severe flaw in the key-exchange protocol for Bluetooth, which allows an attacker to passively recover the session key. Jasek [50] demonstrated how attackers can passively and actively abuse the generic attribute profile in the GATT layer found in Bluetooth network stack. Zillner et al. [48] showed how the Default Trust Center Link Key defined by the Zigbee Alliance [77] is the same across all devices. Fouladi et al. [40] showed how a hard-coded constant in the Z-Wave firmware is used to derive session keys, which eventually became publicly known. Legacy versions of LE protocols have critical security flaws, which many home-based IoT devices implement in hardware; hence limits their mitigation options.

Aside from the inherent flaws, LE protocols offer a proximity feature that authentication systems rely on to identify geographical presence. Ho et al. [79] showed how relay attacks were possible against LE protocols by serializing the LE packets and relaying them over IP. Researchers have shown that MITM relay attacks against LE protocols are practical and break the geographical proximity, which authentication systems rely on. These communication channels can have privacy concerns as demonstrated by Apthorpe et al. [52] and Wood et al. [53].

Mitigations

For HTTP, UPnP, DNS, and NTP protocols, the suggested mitigations include disabling the ECS feature in DNS, using updated versions of the NTP protocol (NTPv4), and using TLS/SSL with insecure protocols (HTTPS). For TLS/SSL implementation flaws, upgrading the server-side and client-side libraries to the latest version should address the vulnerabilities. Further, disabling weak or vulnerable TLS/SSL versions reduces exposure but loses backward compatibility. For LE-based communication, the first generation of Zigbee and

Z-Wave protocols have critical flaws and have limited mitigation options. Vendors can disable insecure portions of these protocols [80] at the expense of compatibility.

A recent direction by researchers is the work found in Apthorpe et al. [52] and Wood et al. [53]. Wood et al. [53] proposed a system that monitors the home network and inform users of sensitive data sent by IoT devices. Apthorpe et al. [52] demonstrated how traffic shaping on the home network can prevent side-channel snooping. This direction of research requires additional attention to empower consumers in protecting their networks and privacy.

Devices electing to use Z-Wave must now opt for *Z-Wave Plus*, which has improved security [81] and over-the-air (OTA) update capabilities. Also, Zigbee added a new security model to allow for secure-key distribution known as Trust Center (TC) [82]. TC is a trusted entity within the Zigbee network that is authorized to distribute keys to Zigbee client devices. TC gives each Zigbee connected device a unique encryption key, unlike the legacy key distribution schema. To mitigate relay attacks in LE protocols, Ho et al. [79] introduced a touch-based-intent communication approach using body-area network (BAN) for signal propagation.

Stakeholders

End-users cannot address the communication flaws since the implementation is on the device, the cloud endpoint, or in the mobile application. Further, vendors have limited options in addressing the communication vulnerabilities since some flaws require a hardware upgrade, but in some cases they can disable them [80]. The vendors can patch vulnerable libraries on the device, the mobile application, and the cloud endpoints.

Internet service providers (ISPs) have visibility into the utilization of IP based protocols, but they are not directly responsible for any mitigation. For ISPs to be involved, they must provide network and legal policies that define their role. As for the LE protocols, vendors can mitigate legacy devices by disabling vulnerable pairing. Users can use alternate

methods for pairing LE devices with IoT hubs if such options exists. Users can buy newer devices that offer next generation secure LE protocols, like Z-Wave Plus and Zigbee.

Take Away

Communication channels provide essential functions for home-based IoT. Home-based IoT devices have adapted industry standards for IP and LE protocols, but they suffer from legacy libraries that in some cases cannot be fixed.

Vendors bear the responsibilities for addressing the vulnerabilities in the communication channels. Further, cloud endpoints and mobile applications can be updated by the vendor directly, but vendors must be proactive and informed about vulnerabilities affecting their software. IoT devices continue to rely on insecure protocols like UPnP and, as we will show next, rarely encrypt their communication on the LAN. End-users do not know if their device or mobile application is vulnerable to weak encryption or MITM attacks unless they analyze and test the communication traffic. An informed power-user might segment their local network into trusted and untrusted zones to limit the exposure.

TLS/SSL addresses insecure protocols that are susceptible to MITM attacks, but they also exhibit flaws in their implementation and deployment. The work of Clark et al. [83] provided additional analysis regarding SSL and HTTPS. ISPs can provide reports outlining best network practices and statistics about device and protocol utilization. Managed cloud IoT platforms use custom communication protocols that rely on public-key infrastructure (PKI) and TLS/SSL protocols. Further studies are required to investigate protocols used by managed cloud IoT platforms. These new platforms are not well studied and warn for further investigation to identify any weaknesses.

2.2 Malware Analysis

In order to perform a systematic comparison between traditional and IoT malware, we require a principled framework that describes the malware threat cycle. We systematized

the literature on traditional malware and noticed five main components of the malware lifecycle:

- **Infection Vector** is how the malware attacks a system.
- **Payload** is the dropped malware code after exploitation.
- **Persistence** is how the malware installs on a system.
- **Capabilities** are the functions in the malware code.
- **C&C Infrastructure** is how the malware communicates with the operator.

For each component, we identify techniques discussed in the literature for traditional malware (desktop/mobile) and empirically measure it for IoT malware (chapter 5). We further refine the categories by examining 25 papers from the systematized works to derive subcategories under each component qualitatively. We then use the *MITRE ATT&CK* taxonomy to derive additional subcategories not found in prior academic works but documented by security companies. Table Table 2.2 summarizes the comparative analysis framework components and their definitions.

2.2.1 Infection Stage

Desktop Infection Vectors. In Table Table 2.3, we see desktop malware pioneered many of the infection techniques. Moore et al. [84] document the SQL Slammer worm that exploited vulnerable SQL services on the internet. Although no *large* academic study explored desktop malware use of repackaging, default credentials, and removable media, there are ample instances from security companies documenting these techniques [85, 86, 87]. Desktop malware rely more on infection vectors like drive-by download and phishing. Provos et al. [88] present an extensive study on drive-by downloads, and several prior works measure [89, 88, 90, 91] and propose defenses [92, 93, 94] for them.

Table 2.2: A summary of the proposed comparative framework and definitions for each component.

	Components	Definition for each component's subcategories
Infection	Remote Exploit	**Remote Exploit** refers to exploiting a service or an application running on a device.
	Repackaging	**Repackaging** refers to benign application repackaged with malware (i.e. pirated software).
	Drive-by	**Drive-by** refers to infection by redirecting the system to a malicious resource.
	Phishing	**Phishing** refers to social engineering attacks that trick a user into getting infected.
	Default Cred.	**Default Credentials** refers to the use of vendor default credentials for device access.
	Rem. Media	**Removable Media** refers to the use of USB for infection between devices.
Payload	Packing	**Packing** refers to the use of packers or polymorphic techniques for obfuscation.
	Env. Keying	**Env. Keying** refers to the dependence on the target's environment artifact (i.e. HW id).
	Scripting	**Scripting** refers to the use of a scripting interpreter (i.e. Powershell, sh, etc.).
	Cross-Arch/Plat.	**Cross-Arch/Plat.** refers to using payloads for different architectures (x86, ARM, etc.) or platforms (Windows, Android, etc.).
Persist.	Firmware	**Firmware** refers to persisting by modifying the device's firmware.
	OS - Kernel	**OS - Kernel** refers to persisting as a kernel module.
	OS - User	**OS - User** refers to persisting in user-space through configuration or process/service.
Capability	Priv. Escalation	**Priv. Escalation** refers to exploiting OS vulnerability to elevate privilege on a device.
	Defense Evasion	**Defense Evasion** refers to actively avoiding or disabling security features on the device.
	Info. Theft	**Info. Theft** refers to profiling and exfiltrating sensitive information from the device.
	Scanning	**Scanning** refers to using the device to scan for other devices.
	DDoS	**DDoS** refers to using the infected device to orchestrate a DDoS attack.
	Destruction	**Destruction** refers to actively destroying or ransoming the device.
	Resource Abuse	**Resource Abuse** refers to using the device to run unauthorized services or applications.
C&C	Peer-2-Peer	**Peer-2-Peer** refers to using peer-2-peer network protocol for managing the botnet.
	Centralized	**Centralized** refers to using a central $C\&C$ server for managing the botnet.
	Email/SMS	**Email/SMS** refers to using email or short message service for call-back to the bot master.

For phishing, Abu Rajab et al. [95] present a multi-dimensional measurement into botnets. Their work documents how botnets leverage phishing emails for spreading. Holz et al. [89] and Kotzias et al. [96] empirically show that phishing is a common infection vector affecting desktop users. Desktop malware continued to evolve and make up a large portion of the threats on the internet. The key insight is that desktop malware initially used remote exploitation and default credentials to automatically spread but has evolved to depend on user interaction. Currently, desktop malware's most common infection techniques require user interaction such as phishing (email), drive-by download (browsing), removable media (physical interaction), and repackaging (i.e. pirated software).

Mobile Infection Vectors. Similar to our study, Zhou et al. [108] look at Android mobile malware and characterize the infection techniques. Their work shows that many Android malware use repackaging, drive-by download, and phishing to propagate as shown in Table Table 2.3. Lindorfer et al. [110] identify removable media propagation techniques in their large-scale study. The key insight is that unlike desktop malware, mobile malware is dependent on user interaction. Automated spreading has not been documented for the mobile platform. While worm-based malware for the Android platform do exist, they require

Table 2.3: A comparison between desktop, mobile, and IoT malware using the proposed framework.

Components		Summary			Desktop																				Mobile				IoT	
	Categories	Desktop	Mobile	IoT	Moore03 [84]	Krueg05 [97]	AbuRa06 [95]	Barfo07 [98]	AbuRa07 [99]	Dagon07 [100]	Holz08 [101]	Polyc08 [102]	Kanic08 [103]	Holz08 [89]	Provo08 [88]	Stone09 [92]	Lu10 [93]	Cho10 [104]	Lindo11 [105]	Shin11 [90]	Rosso12 [106]	Inver14 [94]	Kwon15 [91]	Ganan15 [107]	Kotzi19 [96]	Zhou12 [108]	Lever13 [109]	Lindo14 [110]	Tam17 [111]		
Infection	Remote Exploit	✓		✓	✓			✓				✓								✓					✓					✓	
	Repackaging	✓*	✓																							✓				
	Drive-by	✓	✓										✓	✓	✓	✓			✓		✓	✓			✓					
	Phishing	✓	✓				✓						✓											✓	✓					
	Default Cred.	✓*		✓																									✓	
	Rem. Media	✓*	✓																								✓			
Payload	Packing	✓	✓	✓		✓		✓					✓								✓	✓			✓			✓	✓	
	Env. Keying	✓	✓	✓														✓							✓				✓	
	Scripting	✓*		✓																									✓	
	Cross-Arch/Plat.	✓*	✓	✓																							✓		✓	
Persist.	Firmware	✓		✓											✓														✓	
	OS - Kernel	✓	✓	+							✓		✓		✓										✓					
	OS - User	✓	✓	+										✓													✓	✓		
Capability	Priv. Escalation	✓	✓	✓							✓		✓												✓					
	Defense Evasion	✓	✓	✓			✓	✓						✓				✓		✓					✓		✓	✓	✓	
	Info. Theft	✓	✓	✓			✓	✓			✓				✓								✓		✓		✓		✓	
	Scanning	✓		✓	✓		✓	✓			✓								✓					✓					✓	
	DDoS	✓		✓			✓	✓							✓														✓	
	Destruction	✓	✓	✓																				✓			✓		✓	
	Resource Abuse	✓	✓	✓			✓				✓	✓			✓		✓							✓	✓		✓		✓	
C&C	Peer-2-Peer	✓		✓					✓			✓	✓							✓									✓	
	Centralized	✓	✓	✓			✓	✓	✓	✓	✓			✓	✓		✓			✓	✓	✓	✓		✓	✓	✓		✓	
	Email/SMS	✓	✓								✓																✓			

∗ Techniques documented by security researchers. + Unified software layer that integrates OS and firmware.

Sec. subsection 5.3.2 Sec. subsection 5.3.3 Sec. subsection 5.3.4 Sec. subsection 5.3.5 Sec. subsection 5.3.6

users to visit a link to get infected.

2.2.2 Payload Stage

Desktop Payload Properties. In Table Table 2.3, we see that all the payload categories apply to desktop malware. Kruegel et al. [97] predicted the rise of polymorphic payloads and proposed a way to detect them offline. Later, Barford et al. [98] studied the operation of several desktop family bots, such as GT bot, SpyBot, SDBot, and Agobot, and identified polymorphic payload obfuscation using XOR encoding. Moreover, Holz et al. [89] show that the payloads for the Storm botnet are polymorphic and change every minute, which ensures the payload has different static features to evade detection. Rossow et al. [106] studied downloaders, which are bots that download other malware or unwanted programs. Their work identified more than eight different packer techniques in use by downloaders. These findings suggest that desktop malware payloads use polymorphism to evade detection.

On the defense side, Invernizzi et al. [94] propose a technique to detect polymorphic

payloads in large networks by augmenting networking information such as URI and counts. In addition to packing, environmental keying [105, 112] and scripting [113] are key components for desktop malware to bypass network and host defenses. For scripting, the payload is in the form of a text file that is executed by an interpreter such as Powershell, Python, Lua, or sh. Moreover, desktop malware makes use of cross-architecture and platform payloads for banking malware [110]. These observations suggest that the packaging of cross-architecture and platform payloads introduce a novel infection approach by crossing from trusted devices such as mobile phones and desktops.

Mobile Payload Properties. Zhou et al. [108] observe polymorphic and environmental keying behavior in Android apps. They identify malware samples that adopt the use of polymorphic techniques in the Android environment by using code reflection. They also identify malware samples that check the integrity of their code to ensure that the code is not tampered with. Similar to desktop malware, Lindorfer et al. [110] observe Android malware embedding Windows malware with autorun features that execute once the phone is plugged into a desktop. This advanced behavior leads to cross-architecture and platform infection from trusted devices giving attackers further reach. The key insight is that mobile malware use the same techniques as desktop malware but have limited script-based payloads. Script payloads for mobile devices can be invoked from installed applications, WebView, or exposed services like Android Debug Bridge (ADB), which requires the malware to be already present on the device.

2.2.3 Persistence Stage

Desktop Malware Persistence. Table Table 2.3 shows that desktop malware use all levels of persistence. Provos et al. [88] and Polychronakis et al. [102] identify bots that persist through user-space and kernel modules, respectively. Additionally, Stone-Gross et al. [92] document torpig's botnet and the mebroot infector, which both modify the Master Boot Record (MBR) entry on a hard drive's partition allowing them to run before the OS. Desk-

top malware demonstrate the capability to persist on machines at many levels from the user-space all the way down to the firmware, which are outside the visibility of security tools making them hard to detect and remove.

Mobile Malware Persistence. Mobile malware by default installs and persists as a mobile app on devices unless removed by users or security software. Mobile malware can request background service permissions, subscribe to activities, and broadcast receivers giving it multiple entry points for execution. Researchers [108, 110, 111] show that mobile malware leverage all these entry points for persistence on the Android platform. For example, if malware subscribes to a broadcast receiver for SMS, the malware can execute a specific code that reads the SMS content. The key insight is that the event-driven nature of mobile applications provides a unique persistence method for malware. Detecting event-driven methods is more challenging because it requires anti-malware tools to know the triggering event ahead of time, which can be difficult when the malware is obfuscated.

2.2.4 Capability Stage

Desktop Malware Capability. In Table Table 2.3 we find that desktop malware exhibit all of the listed capabilities. Moore et al. [84] document the capabilities of the Slammer worm, which other botnets also borrow [95, 98, 102, 90, 96]. Several works [103, 92, 104, 107] identify information theft and resource abuse (cryptocurrency mining, click fraud, proxy services, spam, etc.) as a common use of infected devices by desktop malware. Additionally, more recent activities include ransoming devices [96] and DDoS attacks [95] for hire.

Another aspect of desktop malware capabilities is the fact that it can escalate privileges [102] by exploitation or keylogging, and they can evade detection by disabling security tools [105, 106]. The key insight is that desktop malware have diverse capabilities, and malware families specialize based on the intended target and the attacker's goal. For example, remote access can be a specialized capability that targets payroll processing sys-

tems. Moreover, the amount of sensitive information and compute resources (i.e. GPU) found on desktop platforms may make them a desirable target for ransom, information theft, extortion, and compute-intensive abuse.

Mobile Malware Capability. Table Table 2.3 shows that mobile malware has the same abusive capabilities as desktop malware with the exception of scanning and DDoS attacks. Zhou et al. [108] identify malware that root mobile devices, evade detection through dynamic code reflection, steal sensitive information, and abuse SMS services by sending messages to premium numbers. Lindorfer et al. [110] present similar findings, but in addition they find ransomware capabilities that lock devices in exchange for payment. Mobile malware implement a subset of the capabilities found in desktop malware, which may be correlated with the features found on each platform. Unlike desktops, mobile devices generally have lower bandwidth, lower compute resources, are energy conservative, and support a single-user profile.

2.2.5 Command & Control Stage

Desktop Malware C&C. Table Table 2.3 shows that desktop malware use all of the listed methods for $C\&C$ communication. Polychronakis et al. [102] show that desktop malware rely on email for $C\&C$ call-back. Moreover, Kanich et al. [103] and Holz et al. [89] study the Storm botnet P2P network to analyze the spam campaigns and estimate the botnet size. They identify a complex layered infrastructure of a hierarchy of workers, proxies, and master nodes based on the Kademlia DHT protocol. They speculate that this complex infrastructure allows the botnet to scale and be resilient to takedowns. However, Rossow et al. [106] found from a large-scale study that centralized infrastructure was more prevalent than P2P.

For centralized $C\&C$ infrastructure to be more resilient, malware use domain generation algorithms (DGA) [92, 94], multi-tier centralized topology [104], fast-flux [101], and bulletproof or hacked [107] servers. The key insight is that desktop malware enhances the

scalability and resilience of their infrastructure by organizing into specific topologies or by incorporating pseudo-randomness in their domains. For example, Holz et al. [101] note content delivery networks (CDNs) and round-robin DNS (RRDNS) provide resilience for internet applications, which malware mimics by using fast-flux.

Mobile Malware C&C. Lindorfer et al. [110] found that even though the majority of malware use centralized $C\&C$ servers, some mobile malware use SMTP to send sensitive information by email. Most empirical measurements [108, 109, 110] identify that mobile malware does not use the same sophistication for $C\&C$ call-back found in desktop malware. Furthermore, Lever et al. [109] compared mobile malware domains with desktop malware domains and found no major differences. The key insight is that mobile malware may not use sophisticated $C\&C$ infrastructure because of their network mobility property. For example, if a mobile device is connected to a network that blocks its $C\&C$ server (mobile network operator), the device will eventually connect to another network (coffee shop WiFi) as it changes its physical location, which may allow connections to the $C\&C$ server.

Take Away

The systematization highlights the maturity of the desktop and mobile malware landscape. In particular, we observe that desktop and mobile malware have diverse attack vectors, utilize unique tactics in their payload, and persist at different layers in the system. Moreover, desktop and mobile malware tailor their capabilities to extract monetary value from infected systems through resource abuse, extortion, or information theft. Lastly, we find three primary communication tactics used by malware that vary in implementation. These derived categories provide a point of reference that we will later use to compare IoT malware to traditional malware to infer the potential development of the IoT malware landscape. This will inform the risk analysis approach of potential attacks that we must consider.

2.3 Related Work

2.3.1 IoT Security Evaluation

The systematization shows that current efforts focus on one or two components of an IoT deployment, leaving a gap in comprehensiveness. In contrast, in this dissertation, we apply a more complete approach that considers all four components: device services, mobile app, cloud endpoints, and network communication. Furthermore, this work incorporates a time component by iteratively evaluating the security of IoT deployment. Recently, researchers have started to investigate changes in IoT device lifecycle by characterizing TLS communications [114], measuring device updates [115], and attacks on cloud management endpoints [116]. However, these efforts provide a micro-view of a specific protocol or feature over time on the device. This dissertation provides a broader view by examining the device and network communication over a more extended period (18 months), which provides a better perspective on how IoT devices age.

2.3.2 IoT Malware Analysis

Malware targeting embedded Linux-based systems was first reported in 2008 with the discovery of the Hydra IRC bot [117]. Since then, several other bots have entered the scene with various capabilities. Such bots include psyb0t [118], Chuck Norris [119], Carna [120], Tsunami [121], Aidra [122], Dofloo [123], Gafgyt [124], Elknot [125], XOR.DDoS [126], Wifatch [127], TheMoon [128], LUABot [129], Remaiten [130], NewAidra [131], and Moose [132]. Each family had different purposes such as credential theft [132], cryptocurrency mining [133], device destruction [134], internet-wide scanning [120], and cleaning up infected devices [127, 135]. IoT malware development has many considerations due to the heterogeneity of devices. For example, an IP camera and a set-top box can have different processors, C libraries (uclibc, musel, glibc), and kernel versions/features (Linux 2.6, 3.2, 4.6, etc.).

The release of Mirai's source code and recent developments in embedded system toolchains has made it easier for IoT malware development. Antonakakis et al. [1] note that Mirai had a wide impact due to the fact that its small code base runs on diverse devices, spreads efficiently, and targets a large number of insecure IoT devices on the internet [136, 137]. The Mirai botnet took down critical DNS infrastructure [138], disconnected over 900K internet subscribers [139], and attacked a large cloud service provider [140]. Soon after the release of Mirai's code, many variants began to surface with enhancement to its infection vector, payload obfuscation, and C&C communication. For example, Satori [141], a Mirai variant, gained momentum as it exploited a new vulnerability in Huawei routers. These recent developments provide further motivation to understand the IoT malware landscape.

Prior studies looked at IoT malware from different perspectives. Cozzi et al. [142] investigate Linux-based malware but only examine 10K samples, of which 35% are for x86 and x86_64 architecture. Other studies examine specific malware families such as Mirai [1] and Hajime [135]. More comprehensive studies examine individual components of the IoT malware lifecycle. For example, several works [143, 144, 145, 146] examine IoT malware infection tactics and the payload properties. Other works [147, 148] look at how to detect IoT malware by studying its binary static structural features. De Donno et al. [149] organize IoT malware attack capabilities into a taxonomy while Choi et al. [150] study the role that C&C infrastructure plays in the lifecycle of IoT malware.

Additional efforts [151, 152] investigate scanners on the internet to identify if they are infected by IoT malware. Finally, Çetin et al. [153] present a unique perspective on IoT malware infection cleanup by combining multiple data sources and a user study to measure remediation efforts. Our work differs in two aspects. First, we propose a five-component framework that captures the entire lifecycle of IoT malware, which we use to compare with desktop and mobile malware. Second, we conduct the largest and most comprehensive empirical measurement for more than 166K Linux-based IoT malware samples collected over an entire year.

CHAPTER 3

SECURITY EVALUATION OF HOME-BASED IOT DEPLOYMENTS

Figure 3.1: Single IoT deployment.

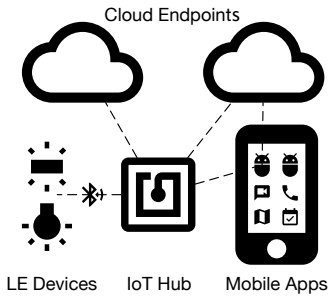
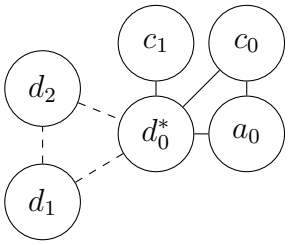


Figure 3.2: IoT graph model.



In chapter 2 we identified four components of an IoT deployment. In this chapter, we formalize this observation and utilize it to evaluate 45 commercial off-the-shelf (COTS) home-based IoT products. The approach involves segmenting each device into its respective topology as shown in Figure 3.1. Formally, we define an IoT deployment as a set of vertices V and edges E as illustrated in Figure 3.2. Overall, our abstract model has four main components: a set of devices (D), a set of cloud endpoints (C), a set of mobile applications (A), and a set of communication channels (E).

$$\begin{aligned} \text{where:} \quad & A, C, D \subset V; \quad D : \{d_i, i \in \mathbb{Z}\}; \\ & C : \{c_j, j \in \mathbb{Z}\}; \quad A : \{a_k, k \in \mathbb{Z}\}; \\ & E : \{e_l, l \in \mathbb{Z}\} \end{aligned}$$

For each device deployment, we construct a representative graph and examine the *security properties* for each component.

3.0.1 Security Properties

The security properties have three categories: attack vectors, mitigations, and stakeholders. Attack vectors are the methods used to circumvent the security of the IoT system. The mitigations define which measures should be taken to address the attack vectors. Lastly, the stakeholders represent the party responsible for mitigation. In this chapter, we will examine the attack vectors on all four IoT components by empirically evaluating 45 COTS IoT devices for the following attack categories:

Attack Vector. The device has three attack categories: *vulnerable services*, *weak authentications*, and *default configurations* that are defined as follows:

- **Vulnerable services** refers to vulnerabilities in running *services*.
- **Weak authentications** refers to weak or guessable *credentials*.
- **Default configurations** refers to the device operating with insecure *factory settings*.

The mobile application has three attack categories, *permissions*, *programming*, and *data protection* that are defined as follows:

- **Permissions** refers to a mobile application being over-privileged.
- **Programming** refers to the mobile application containing vulnerable *implementations*, including improper use of cryptographic protocols.
- **Data protection** refers to the mobile application hard coding *sensitive information*.

The communication of the components have two attack categories, *encryption* and *man-in-the-middle (MITM)* that are defined as follows:

- **Encryption** refers to the lack of encryption or support of weak encryption protocols.
- **MITM** refers to the susceptibility to a man-in-the-middle attack.

The cloud endpoint shares the following attack categories with devices and communication edges: *vulnerable services*, *weak authentications*, and *encryption*, as defined above.

3.0.2 Evaluation Scope and Attack Model

We limit our scope to home-based IoT devices because they are relevant to the systematized work, they are readily available, and the experiment setup can be easily reproduced. For the evaluation, we simplify the attack model to an Internet protocol (IP) network attacker. We recognize that there are more powerful adversaries that can attack low-energy (LE) based devices [21], but they require specialized resources that are not available in many home networks. We consider the exploitation of a hub device (communication bridge between low-energy and IP) to be equivalent to exploiting all the connected low-energy devices because a trust session exists between the hub and the low-energy devices. We exclude direct evaluation of low-energy devices but consider their hubs for evaluation. Finally, we consider the home network to be an *untrusted* network and we make no assumptions about the security state of mobile applications, modems/routers, or web browsers that have complete visibility to the home network ([28]).

3.1 Security Evaluation Methodology

This section will cover the testbed architecture, challenges and limitations, and the security evaluation approach. Our methodology and evaluation require minimal technical expertise to replicate and are deliberately devised to appeal to a wide range of technical audiences allowing them to contribute to this effort. We used a mix of commercial and open-source tools to conduct the evaluation; all of the commercial tools have open-source counterparts. Our evaluation results are summarized in Table 3.3.

3.1.1 Testbed

Building a home-based IoT testbed is challenging and requires careful planning. This section will cover the details of the testbed we developed for this study. We will cover device selection, network architecture, device management, data collection, and testing tools. Ta-

ble Table 3.1 summarizes the automation effort for each stage in the testbed. For each stage, we will present the tools and technical approach.

Table 3.1: A summary of the efforts required for the testbed components and evaluation stages.

Stage	Process	Manual	Semi-Auto	Automated
Deployment	Documentation	✓		
	Network Provisioning	✓		
	Physical Deployment	✓		
	Connectivity Configuration	✓		
	Reachability/Functional Test	✓		
Data Collection	Network Traffic			✓
	Device Scans			✓
	Mobile App Analysis			✓
	Cloud Scans	✓		
Evaluation	Network Protocols		✓	
	Network Interception	✓		
	Device			✓
	Mobile App			✓
	Cloud			✓
Analysis	Network Proto./Intercept	✓		
	Device Vuln.	✓		
	Mobile App Vuln.	✓		
	Cloud Labeling	✓		
	Cloud Vuln.	✓		

Device Selection

We evaluated 45 devices spanning categories that include appliances, cameras, home assistants, home automation, media, and network devices. We chose the most popular devices available on the market at the time of the study but also included some white-label devices with no specific brand (i.e., Chinese Webcam). A complete list of the evaluated devices is in Table 3.2. The device selection process is essential for several reasons. First, the diversity can identify vulnerabilities unique to specific device types or vendors. Second, diversity can help evaluate the effectiveness of security measures across a range of devices rather than just a single device or device type. Third, device diversity helps provide a more realistic representation in a real-world household with IoT deployments, which typically involve a mix of device types and vendors. Lastly and more importantly, representative device selection increases the relevance and generalizability of the results.

Table 3.2: An overview of the devices used in the evaluation.

Device	Category	Hub	Cloud Endpoints	Mobile Application		Communication	
				iOS	Android	IP	Low-Energy
Belkin WeMo Crockpot	Appliance		27	✓	✓	✓	
Roomba	Appliance		11	✓	✓	✓	
Belkin Netcam	Camera		79	✓	✓	✓	
Canary	Camera		22	✓	✓	✓	✓
Chinese Webcam	Camera		1	—	—	✓	
D-Link DCS5009L	Camera		4	✓	✓	✓	
Logi Circle	Camera		341	✓	✓	✓	✓
Nest Cam IQ	Camera		9	✓	✓	✓	✓
Nest Camera	Camera		7	✓	✓	✓	✓
Netgear Arlo	Camera		59	✓	✓	✓	
Piper NV	Camera		42	✓	✓	✓	✓
Withings Home	Camera		20	✓	✓	✓	✓
Amazon Echo	Home Assistant		221	✓	✓	✓	
Apple HomePod	Home Assistant		221	—	—	✓	✓
Google Home	Home Assistant		42	✓	✓	✓	
Google Home Mini	Home Assistant		221	✓	✓	✓	
Harmon Kardon Invoke	Home Assistant		128	✓	✓	✓	✓
August Doorbell	Home Automation		221	✓	✓	✓	✓
Belkin WeMo Link	Home Automation	✓	14	✓	✓	✓	✓
Belkin WeMo Motion	Home Automation		221	✓	✓	✓	
Belkin WeMo Switch	Home Automation		29	✓	✓	✓	
Caseta Hub	Home Automation	✓	221	✓	✓	✓	✓
Chamberlain myQ Garage Opener	Home Automation	✓	1	✓	✓	✓	✓
Insteon Hub	Home Automation	✓	20	✓	✓	✓	✓
Koogeek Lightbulb	Home Automation		1	✓	✓	✓	
LIFX Virtual Bulb	Home Automation		3	✓	✓	✓	
MiCasaVerde VeraLite	Home Automation	✓	74	✓	✓	✓	
Nest Guard	Home Automation	✓	14	✓	✓	✓	✓
Philips HUE	Home Automation	✓	27	✓	✓	✓	✓
Ring Doorbell	Home Automation		9	✓	✓	✓	
Samsung SmartThings	Home Automation	✓	10	✓	✓	✓	✓
TP-Link Wifi Bulb	Home Automation		11	✓	✓	✓	
TP-Link Wifi Plug	Home Automation		11	✓	✓	✓	
Wink 2	Home Automation	✓	12	✓	✓	✓	✓
Amazon Fire TV	Media		174	✓	✓	✓	
Apple TV (4th Gen)	Media		439	✓		✓	
Bose SoundTouch 10	Media		26	✓	✓	✓	✓
Logitech Harmony	Media		17	✓	✓	✓	✓
nVidia Shield	Media		261	—	—	✓	
Roku 4	Media		231	✓	✓	✓	
Roku TV	Media		226	✓	✓	✓	
Samsung SmartTV	Media		182	✓	✓	✓	
Sonos	Media		65	✓	✓	✓	✓
Google OnHub	Network		24	✓	✓	✓	
Securifi	Network	✓	938	✓	✓	✓	✓

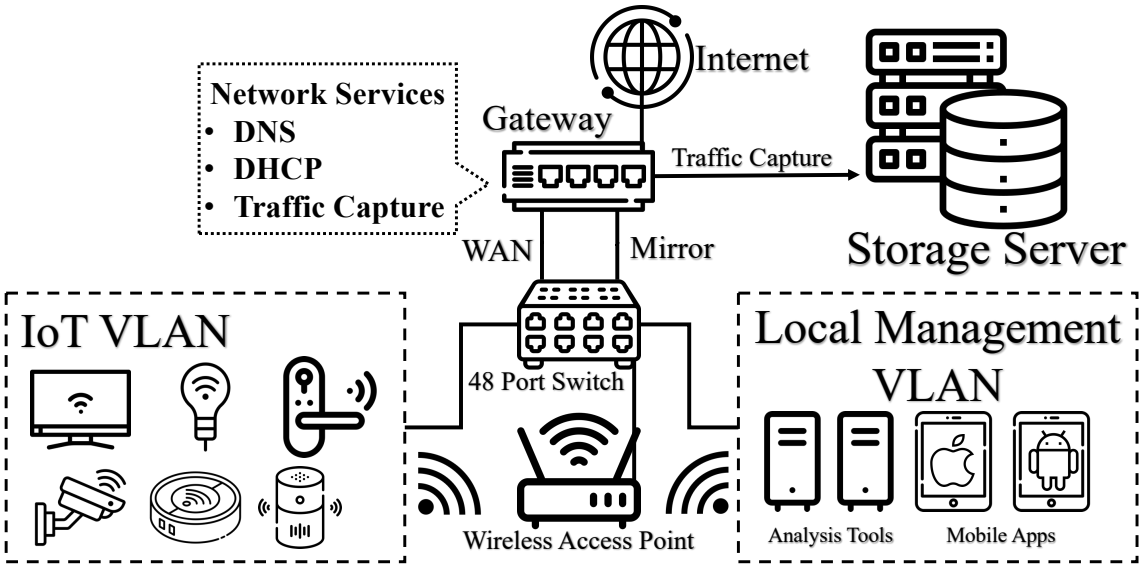


Figure 3.3: An overview of the lab architecture.

Network Architecture

The goals for the testbed network architecture are to ensure isolation from other user networks, reliable connectivity through wired and wireless mediums, scalability and expansion for adding new devices, minimum congestion, monitoring visibility, and access control and logging. We design our network architecture to achieve these goals as shown in Figure 3.3. Our network topology consists of three primary networks: IoT VLAN, local management VLAN, and egress traffic (multiple networks). The gateway configures the VLAN segmentation and assignment, which provides essential network services such as Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), and traffic capture. We configure wired and wireless connectivity to support various IoT devices. Precisely, we deploy an unmanaged 48-port network switch with port mirroring and a wireless access point. We prioritize wired connectivity over Wireless since it provides more stability and less interference with other wireless devices. Moreover, the wired medium allows visibility on East-to-West traffic through port mirroring. On the other hand, Wireless requires specialized networking equipment to gain visibility on East-to-West traffic.

The gateway is a general-purpose x86 computer running Debian Jessie Linux that hosts

dnsmasq for the DNS and DHCP services. The gateway runs full-packet capture using *tcpdump* and remotely stores the traffic capture on a storage server with 50TB capacity. The traffic is sent via an out-of-band management network configured on the gateway and never crosses the local IoT network. The VLAN allocation helps manage and isolate IoT traffic from other devices that need to be on the network. For example, the local management VLANs contain several necessary devices that allow us to interact and configure the IoT devices. Moreover, the analysis tool servers host security tools that enable us to evaluate each device's device and network security. Therefore, the local management VLAN and the IoT VLAN require connectivity managed by the gateway. In the local management VLAN we deploy two tablets, namely an Apple mini iOS iPad and a Samsung Galaxy Android Tab. These two tablets run the IoT companion apps, which interface with the deployed IoT devices.

The analysis servers run a variety of security tools like network vulnerability scanners, UPnP profiler, ARP manipulation tools, TLS/SSL interception, and custom scripts. The local management VLAN provides a testing ground to probe and manipulate traffic in the IoT VLAN. We assign static IP addresses for the VLANs (IoT and management) to track devices' egress traffic and map the network communication to the corresponding device. The static IP address assignment is critical because managing and tracking IoT devices become cumbersome as the lab grows. Lastly, we build redundancy in the lab by deploying a backup gateway (not shown in Figure 3.3) that maintains a copy of the active gateway. We maintain an *Ansible* playbook that automatically configures and deploys the secondary gateway in case of a network, software, or hardware failure.

Device Deployment

When deploying each device into the testbed, we strive for a standardized approach that ensures systematic onboarding. Moreover, we design the testbed environment to have a flat network, similar to a home network (IoT VLAN in Figure 3.3). The device deployment is

a manual process, as noted in Table 3.1, and it is a one-time effort. Our device onboarding procedure is the following:

1. Document device make, model, MAC address, and credentials.
2. Note the configuration method from the manual.
3. Create a DHCP entry in the *dnsmasq* configuration file assigning a static IP from the IoT VLAN pool.
4. If applicable, download the mobile applications for both platforms, iOS and Android.
5. Create an account using the mobile app and document the credentials.
6. Plug the device into a power source and configure it to connect to the internet.
7. Ensure the device has internet connectivity, correct local IP address assignment, and is visible to mobile apps.
8. If applicable, document failure to deploy devices, the reason for failing, and potential workaround.

The device configuration is kept at a minimum to represent the default vendor configurations. We assume many users will most likely use the default *security* configurations on the device and customize the functional features instead. For the initial security evaluation, we disable automatic updates, when it is possible, to limit device behavior change. However, after the initial security evaluation, we turn automatic updates back on and continuously evaluate the device. More on the iterative security evaluation in chapter 4. Lastly, we create a fictitious persona with an email address, a phone number, and a physical address to register all the accounts for the deployed devices.

Data Collection

There are several things to consider for data collection. In the testbed, we collect network traffic, device service scans, mobile app analysis artifacts, and cloud service scans. Each requires a specific approach. Network collection is the easiest, which we passively collect at the gateway. However, we only have partial visibility of the LAN traffic (East-to-West) because the mirroring port on the switch only captures wired traffic. Wireless traffic requires specialized equipment, which we could not accommodate. We capture the traffic on the gateway and remotely store it on a storage server as compressed full-packet capture files. The device service evaluation is an active approach that scans the IoT VLAN once every week. The scanner collects the results and saves them to a managed cloud service. We use Tenable Nessus, a commercial cloud-managed scanner. The mobile app binary analysis requires specialized reverse engineering and analysis pipelines. We initially semi-automated the Mobile Security Framework (MobSF) to analyze the IoT companion mobile apps. Later, we partnered with Kryptowire, who gave us access to their cloud analysis platform. We use their platform to analyze the companion mobile apps and collect the results automatically. Lastly, we semi-automated the scanning task through Tenable Nessus for cloud scanning. However, we observed many IoT devices churn through different cloud backends over time, requiring continuous extraction, manual labeling, and scanning. We could not scale this process and opted to conduct a single scan based on a manual effort. Moreover, we received requests from some cloud backend owners to exclude their backends from the scans, which we manually removed.

Device Functional Features

The deployed devices are in an idle state most of the time. We manually exercise certain features on each device during the evaluation to induce network activity or for security testing. For example, when we want to study the control network paths for a lightbulb, we would turn it on/off and change the brightness and color. We then analyze the network

traffic to see if the mobile app communicated directly with the device on the IoT VLAN or through a cloud backend (across the internet). Moreover, once we identify the path from the control point to the device, we would exercise the functions again but attempt to intercept the communication. This functional device testing allows us to evaluate the security between different components. We document the time for each experiment and analyze the corresponding data collection in the network traces offline.

Challenges and Limitations

Building a home-based IoT testbed presented many challenges, including power provisioning, device deployment, firmware updates, idle-to-off state, emulating a natural household environment, and testbed maintenance. Typically devices' power source can be directly powered by a wall socket. When deploying a large number of devices, we require power provisioning. This provisioning entails the allocation of power sockets and amperage. Most home-based IoT devices typically operate at a 5V or 12V and draw a current of 0.5A to 2A in the United States. However, some devices with heat elements such as *Belkin WeMo Crockpot* can peak to five amperages. We had our facility modify the room power source to allocate 30 amperages per 12 devices. We have 120 amperages dedicated to the IoT testbed to maintain and avoid power interruption.

Power Provisioning. Some devices require specialized power supplies, such as smart doorbells, thermostats, and door locks. Door locks are mostly battery-operated, which requires constant monitoring and battery replacement. Smart thermostats like the *Nest* require unique power brick to supply power to the device, and the device must be connected to an HVAC system to operate. We address these issues using a breadboard to hard wire the power supply and a standalone relay emulating an HVAC system. Similarly, we used a power brick for smart doorbells that output the appropriate voltage and amperage range and hard wire it to the device using a breadboard. Lastly, smart lightbulbs use non-standard power sockets that we had to account for in our budget. We purchase several lamp polls

with multiple lightbulb reciprocals to accommodate multiple smart lightbulbs under one poll.

Device Updates. Tracking device updates presents an additional challenge. In our initial onboarding process, we turn off automatic updates; however, some devices do not allow users to turn off automatic updates. When a device updates, the behavior of the device can change, and prior observed behavior or vulnerabilities may no longer be visible. The updates make the security evaluation less reliable because we cannot repeat the assessment and observe the same results. One potential solution is to block internet access for the device until we scan the services on the device. However, some devices will only activate once they can reach the internet. Another potential solution includes adding the firmware download site to a blocklist to make the device fail in the update process deliberately. We did not implement any of these solutions, but we document them for future iterations of the testbed.

Natural Household Environment. The deployed devices are in a lab with very little foot traffic. This setting reduces the activities around the devices and, therefore, may not represent a natural household environment. Specifically, many devices have an idle-to-sleep process meant to conserve power when they are not actively used. For example, many smart TVs will turn off when idle. Other devices require activation for them to work, like smart vacuums. To address these challenges, we would repeat music or a movie on the smart TV to keep it from sleeping. However, this approach creates additional traffic on the network, which may mask the idle behavior of the smart TV. For devices requiring activation, we created a schedule to activate and perform their function. Other devices require physical interaction to activate their functionality, like home assistants. A potential solution would be to play a recorded message to voice-enabled devices or instrument the power through metered power distribution units to activate the devices. We did not implement those solutions and document them here for future iterations.

Ongoing Maintenance. Lastly, the ongoing maintenance of the lab is cumbersome and

requires due diligence. For example, the testbed requires various approaches to address the abovementioned challenges. As the testbed scales, each custom solution requires monitoring services to ensure they do not fail. With the smart TV playing music, *YouTube* will periodically prompt the user if they are still there. Most video streaming services like *Netflix* have similar features. Moreover, device monitoring is complex because not all devices have the same network behavior or network services we can check via a network probe. Sometimes, the device may be powered but does not have connectivity due to wireless congestion, for example. Network monitoring will report that the device is offline, and we will only identify the problem if we manually inspect the device. Battery-operated devices require a periodic manual inspection to replace the batteries when needed. We have yet to identify an efficient solution to this problem and currently use a manual approach to maintain the testbed.

3.1.2 Evaluation Procedures

We define a threat model for evaluating the security of deployed devices. Specifically, we assume several scenarios including, a nearby attacker, an on-LAN attacker, an on-path attacker (dubious Internet Service Providers (ISPs)), and a remote attacker. These threat models will become more relevant in chapter 6. For each IoT component, we use specific tools to evaluate their security. With the exception of network communication, Table 3.1 shows that the evaluation is mostly automated because we rely on automated commercial and open-source tools.

Device

We use Tenable Nessus Scanner [154] to scan devices for service discovery, service profiling, and vulnerability assessment. Nessus Scanner annotates the CVE [155] information with the versions of running services and provides a summary of their security state. Nessus Scanner uses the CVSS [156] scoring system to rate the severity of the discovered vulnera-

bility on a scale from one to ten and categorizes them into low, medium, high, and critical. We consider any classification of the categories high or critical by the CVSS scoring system as problematic and note it in Table 3.3.

Mobile App

We used MobSF [157], Qark [158], and services from Kryptowire [159] to statically and dynamically evaluate each mobile application for the IoT devices. We looked at both the Android and the iOS applications and presented the vulnerable of the two [1] in Table 3.3. There are 42 devices that have a companion mobile application. We analyzed a total of 83 mobile applications of which 41 are Android and 42 are iOS. We evaluate the apps for permissions and permission use (over-privilege), embedded secrets (passwords, API keys, etc.), and programming errors (incorrect use of cryptographic functions).

Cloud Endpoints

We used Nessus Scanner to discover, profile, and assess running services on the cloud endpoints. We classified each domain into one of four categories: first-party, third-party, hybrid, and unknown. First-party refers to cloud-based services that run on the vendor's infrastructure, third-party refers to subscription services like Content Delivery Network (CDN), hybrid refers to cloud-based infrastructures (IaaS), like Amazon AWS or Microsoft Azure, that host IoT cloud services, and Unknown refers to unclassified infrastructure due to ambiguity. For each cloud endpoint, we evaluated the running services and TLS/SSL configurations, if applicable.

Network Communication

We used Nessus Network Monitor [154], ntop-ng [160], Wireshark [161], and sslsplit [162] to profile the communication edges for each device. We manually inspected traffic and

[1] The portal contains the data for both platforms iOS and Android.

tested them for Man-in-the-Middle (MITM) attack using sslsplit. IoT devices connect with their components using IP-based channels, represented as edges in the model graph (see Figure 3.2). We classified three types of connections, device-to-cloud (D-C), mobile application-to-device (A-D), and mobile application-to-cloud (A-C). We observed 43 devices connecting to cloud endpoints (D-C), 35 mobile applications connecting to cloud endpoints (A-C), and 27 mobile applications connecting to devices through the local area network (LAN) (A-D). We categorized these connections into five application protocols, namely: DNS, HTTP, UPnP, NTP, and custom. The custom category refers to device-specific application protocols. Smart devices utilize many protocols, but in our lab, we only observe the five listed above.

3.2 Results

3.2.1 Device

We evaluated 45 devices and found a total of 84 running services and 39 issues related to those running services. We found devices with running services such as SSH, UPnP, HTTP web server, DNS, Telnet, RTSP, and custom services. Many devices configure TLS/SSL for their services, but their configurations had several issues. For example, the certificates were self-signed, they supported weak to medium ciphers, they used short TLS/SSL keys, they permitted the use of vulnerable versions of SSL (v2, v3, and CBC mode), and had expired certificates. Further, some devices ran outdated and vulnerable services that allowed remote code execution, leaked sensitive information, and ran unauthenticated services.

For example, the *Insteon hub* runs a web server with TLS on port 443 and listens on port 22 for SSH connections. The certificate used for the TLS connection is expired and self-signed, while the TLS service allowed for weak ciphers like RC4 and insecure protocols like SSLv3. Similarly, the *Wink 2*, *Sonos Speakers*, *nVidia Shield*, *Google Home*, *Samsung SmartTV*, and *Samsung SmartThings* all had issues with their certificates or TLS/SSL configurations. The *Wink 2* and *Sonos* both used short SSL keys of size 1024 bits.

Other devices like *D-Link DCS5009L*, *Bose SoundTouch 10*, *Chinese webcam*, and *Securifi Almond* lacked encryption for service authentication, which allows any device on the LAN to snoop.

Devices that run UPnP services have no authentication or security built in and by default are insecure. Devices like the *MiCasaVerda VeraLite*, *Wink 2*, *Sonos*, *Bose SoundTouch 10*, *Samsung SmartTV*, *Logitech Harmony*, and *Roku* all run UPnP services that allow anyone on the LAN to control the device. Specifically, the *MiCasaVerda VeraLite* uses vulnerable versions of the UPnP service libraries that have public exploits, such as *libupnp 1.6.18* (CVE-2012-5965), *dropbear 2016.72* (CVE-2012-0920), and *UPnP RunLua* (CVE-2013-4863). A complete list of CVEs with CVSS scores of high and critical are found in Table 3.5. We found 16 devices with running services that had no issues and ten devices that did not expose running services. For example, the *Nest* camera uses a push/pull client approach, which limits the exposure of running services.

Findings. The device evaluation found issues related to the device setup, software updates, and service configurations. Additional evaluation results for each device are found in Table 3.4.

3.2.2 Mobile

We found that 39 devices had one or more issues related to permissions, sensitive data, or incorrect use of cryptography. We observed 24 over-privileged mobile applications that ask for permissions on the mobile device that are not used by the application code. As for sensitive data, we found 15 mobile applications to have hard-coded sensitive data like API keys for *Google Geocoding*, *Google Maps*, *fabric.io*, *HockyApp*, *Localytics*, *Microsoft Virtual Earth*, *Umeng*, and other credentials to cloud and device services. We found 17 mobile applications that did not implement cryptographic protocols securely or had hard-coded static keys and initialization vectors (IVs). The cryptographic implementations relied on older or broken algorithms like AES-128 and MD5 hash, respectively. Other applications did not

enforce SSL and allowed for communication over unverified connections.

Findings. The evaluation identifies issues with inherent trust between mobile applications and devices that the systematized work neglects. A summary of our mobile application evaluation is provided in Table 3.3 and additional details are found in Table 3.6.

3.2.3 Cloud

On the IoT network, we observed over 4,000 cloud endpoint domains across the 45 devices. We classified 950 domains as first-party, 1287 domains as third-party, 630 domains as a hybrid, and 1288 domains as unknown. The unknown category includes unattributable domains for a device. For example, the Hulu application running on a Smart TV uses an AWS CloudFront domain, which gives us no indication if the domain belongs to Hulu or the Smart TV. We found 18 devices that used outdated services, leaked sensitive information, lacked encryption for authentication, or ran a vulnerable service. We found eight devices using cloud endpoints that are vulnerable and have public exploits. Additionally, seven devices were authenticated with cloud endpoints in clear text. We found 26 devices using cloud endpoints that have TLS/SSL configuration issues, like self-signed certificates, domain name mismatch, and support for vulnerable versions of TLS/SSL protocol. We found ten devices that used misconfigured cloud endpoints, which allowed for sensitive information disclosure like file paths and running processes on the server. We saw four devices use cloud endpoints that ran outdated operating systems with expired vendor support (Ubuntu 10 and Ubuntu 12).

Findings. The evaluation found issues with the deployment of unsupported legacy OS and sensitive information disclosure. We summarize our findings in Table 3.3 and provide additional details in Table 3.7.

3.2.4 Network

We found 41 devices used the DNS protocol, where 6 of them did not respect the network-configured DNS recursive server, and instead used Google's or OpenDNS's servers. We found that 38 devices used the HTTP protocol and 34 of them used TLS/SSL sessions (HTTPS). We found 21 devices that used the UPnP protocol either by sending a multicast SSDP request or responding to an SSDP request. Additionally, we saw 25 devices that used the NTP protocol for time synchronization. We observed 28 devices that used custom protocols that were specific to a device. For example, Google products (OnHub, Home, and Home Mini) all sent traffic to Google's servers using a custom protocol on ports 5228 and 5223.

The majority of the devices used encryption over the Internet (D-C). We found 25 devices that encrypted all their communication, 15 devices that partially encrypted their communication, and two devices that did not encrypt their communication to the cloud endpoints. As for the mobile applications (A-C), 24 encrypted all their communication, ten partially encrypted their communication, and one did not encrypt its communication to the cloud endpoints. On the LAN (A-D) we observed five devices that encrypted their communication, two devices that partially encrypted their communication, and 20 that did not encrypt their communication. A few devices, such as the Chinese webcam, did not have a companion mobile application but provided an HTTP interface that allows any device on the LAN to authenticate and interact with.

In addition to the communication analysis, we actively MITM attacked every communication edge to test their susceptibility. We found in total 20 devices had one or more of their communication edges susceptible to a MITM attack. We found four device-to-cloud (D-C) communications that were susceptible, two mobile application-to-cloud (A-C) communications that were susceptible, and 20 application-to-device (A-D) communications that were susceptible.

Findings. The evaluation finds that not all communication channels are secured and lack

endpoint verification. We found devices that leak usage information by forcefully using third-party recursive DNS servers. Table 3.3 summarizes the device encryption and MITM attack and additional details are found in Table 3.8.

3.3 Evaluation Cases

Our evaluation shows that some devices have a better security posture than others. In this section, we take a look at three devices that we categorize based on their overall security evaluation. We propose three categories: *good*, *satisfactory*, and *needs improvement*, which highlight good security practices and shortcomings.

3.3.1 Good: Withings Home

Functional Features. The Withings Home device is a camera paired with an air quality sensor. The device has a mobile companion application, integrates with cloud endpoints, and communicates over the Internet and the local network. The device exposes mDNS service, which allows zero-configuration protocols to find and configure the device (i.e. Apple's Bonjour). The device uses a low-energy protocol, Bluetooth, to configure the device initially, then switches to IP communication. Device updates are not applied automatically but require user consent.

Assessment. We found no issues with the *mDNS* service running on the device. The companion mobile application correctly utilizes secure storage facilities to store sensitive data, correctly uses cryptographic protocols, and has proper permission provisioning. The majority of the cloud infrastructure is self-hosted by Nokia and runs services to enable user notifications and control. The network communication between device-to-cloud, app-to-cloud, and app-to-device uses full encryption and is not susceptible to MITM attacks. The device did authenticate in clear-text (an insecure practice) across the Internet to associate the device with the cloud management interface that runs an XMPP server [2].

[2] endpoint located at: xmpp.withings.net:5222

3.3.2 Satisfactory: Nest Cam

Functional Features. The Nest Cam is an indoor camera that senses motion, records video, and notifies users of activities. The device uses forced configuration, which means users have to configure and set up their device before it can operate. The camera uses the Bluetooth protocol to configure the device via the mobile application, which pairs using a pin/barcode located on the back of the camera. The camera does not utilize the local network to control the device, all of the activities and controls operate through the cloud endpoints. Finally, updates to the device are applied automatically with no user consent, ensuring the device always has the latest running firmware.

Assessment. The Nest Cam does not expose any services but uses a client model, where the device acts as a client that communicates directly with the cloud endpoints. The lack of exposed services running on the Nest Cam considerably shrinks the attack vector and limits an IP-based attacker. The Nest Cam uses certificate pinning on the device, which verifies and validates the device to cloud communication is secure. The device setup and configuration require mobile application pairing via Bluetooth, which both ensures the proximity of the end-user and limits remote attack vectors. The mobile application manages all Nest products, including the Nest Cam, which requests access to the microphone, camera/photos, geolocation, and other sensitive services. The cloud endpoints fully manage the Nest Cam, which means without Internet access the device is inaccessible. The Nest products, in general, forcibly use the Google DNS recursive and ignore the DHCP configurations on the local network. A savvy user can configure static routes on their gateway to redirect DNS traffic toward their desired resolver.

3.3.3 Needs Improvement: MiCasa Verde VeraLite

Functional Features. The VeraLite is a smart-home Z-Wave-enabled controller that can monitor and control low-energy sensors and other devices around the home. The device pairs through a cloud portal using a pre-printed pin on the back of the device. The VeraLite

requires manual updates, but the device notifies users of the availability of new updates. The device exposes four services including a web, DNS, UPnP, and SSH server. The mobile device requests excessive permissions like calling, controlling the phone network state (on/off/airplane mode), and access to the camera. The VeraLite is a discontinued product and is no longer offered by the vendor.

Assessment. The VeraLite device provides a hardened setting that disables many of the running services on the device, but they are on by default. The hardened mode forces device management and monitoring from cloud endpoints. The device has several exploitable vulnerabilities as illustrated by Table 3.5. The UPnP services use a vulnerable version of the *libupnp* library, and the SSH services use a vulnerable *dropbear* (2016.72) implementation.

The configuration of the SSH server supports Cipher-Block-Chaining (CBC) mode, specifically *3des-cbc*, *aes128-cbc*, and *aes256-cbc*, which an attacker can exploit to recover the plaintext from the ciphertext. The DNS service is configured to allow queries for third-party domains that do not have the recursion bit set; hence allowing attackers to snoop on the DNS cache. The mobile application requires users to establish an account with the Vera vendor, which allows end-users to manage their controllers. The device does not use certificate pinning, which leaves the deployment susceptible to MITM attacks. The cloud endpoints use clear-text authentication, run exploitable services, expose sensitive information, and run unsupported operating systems.

3.4 An Integrated Security Evaluation

In Chapter chapter 2, we learned that most security evaluation approaches for IoT focus on only a few components. However, our proposed approach provides a more comprehensive evaluation that includes all components. So far, we have evaluated each device's components and demonstrated that security vulnerabilities could occur in one or more of them. In the previous section, we discussed three cases of devices, with a particular focus on the network service component. In the following sections, we will demonstrate how

our approach can identify more security vulnerabilities in IoT deployments by integrating the security evaluation of all components. To accomplish this, we will present an empirical case study of the *Belkin Netcam* device, including its mobile app, cloud backends, and network communication.

3.4.1 Device

The *Belkin Netcam* runs a Linux-based Operating System (OS) based on Linux Kernel 2.6. The device exposes an HTTP server that listens on ports 80 and 81. The HTTP service provides unauthenticated UPnP functions that users on the local network can invoke. When users set up the device, they are guided through a configuration process using the companion mobile application. This configuration process includes connecting the device to the local wireless network to access the internet. The user configures the device by connecting their mobile device to a wireless access point that the *Belkin Netcam* broadcasts. Users authenticate to the wireless access point using a four-digit pin, which is publicly known. To configure the device, the user must also authenticate using the credentials *admin/admin* (username/password) that are also publicly known. The device automatically checks for updates and notifies the user if new firmware is available. However, for an update to occur, the user must manually consent to update the device.

3.4.2 Mobile App

The *Belkin Netcam* companion mobile application supports both Apple iOS and Android devices. Our analysis of the Apple iOS application found no issues related to permissions, embedded secrets, or programming errors. For the Android mobile application, we found an embedded application key for *Localytics*. *Localytics* is an application analytics software that tracks application usage and provides in-depth insights to help developers improve engagement. The application key allows anyone to authenticate to the *Localytics* remote server and spoof activities. However, regarding the security impact on the *Belkin Netcam*

deployment, the app key does not expose any security flaws.

3.4.3 Cloud Endpoints

The *Belkin Netcam* device communicates with 79 different backends. These backends include 13 first-party, 63 third-party, one hybrid, two unknown, and two direct IP addresses. First-party backends are servers owned and operated by *Belkin*, and third-party backends provide analytics, ads, and telemetry services through third-party companies. Hybrid servers are backends that reside on public cloud infrastructure, AWS, Azure, and GCP, but host applications developed by *Belkin*. We could not label two backends and two IP addresses observed in the *Belkin Netcam* evaluation. Our evaluation of first-party backends shows that the *Belkin Netcam* cloud services contain several security flaws ranging from medium to critical in CVSS. Specifically, one server backend contained a JBOSS unauthorized access and remote code execution flaw that we categorize under exploitable services and information disclosure. Moreover, we found some backends to expose their Apache tomcat configuration files, which can potentially contain sensitive information about the server application. We found other services that use basic HTTP authentication without TLS/SSL protection, which allows an on-path attacker to view the credentials. We found that services that use TLS/SSL support SSL versions 2 and 3, which allows an attacker to downgrade secure connections and decrypt the encrypted sessions. Lastly, some backends expose SSH service that supports weak authentication algorithms that attackers can potentially abuse. These findings directly impact the *Belkin Netcam* deployment and potentially give an attacker control over all devices that use the same backends.

3.4.4 Network Communication

The *Belkin Netcam* uses several network protocols, including DNS, HTTP/HTTPS, UPnP, and custom protocols. We found three types of communications, namely mobile app-to-device (A-D), device-to-cloud (D-C), and mobile app-to-cloud (A-C). For each communi-

cation type, we evaluate MITM attacks to assess the manipulation of network communication. Additionally, we look at each communication and assess if they are encrypted (not visible to an on-path attacker). The *Belkin Netcam* A-D communications do not use encryption, which exposes them to MITM attack and information disclosure. The D-C communication uses encryption and is not susceptible to MITM attack. We attempted a MITM attack by serving a fake certificate to the device, but the device raised an unknown/untrusted CA error and failed to connect to the cloud endpoints. However, we found that the *Belkin Netcam* streams unencrypted video on port 5338, which allows anyone on the network path to view and record video footage. These security flaws allow different attacks related to privacy and information disclosure.

3.4.5 Attack Paths

We observe several security flaws impacting the *Belkin Netcam* device, mobile app, cloud backends, and network communication. In traditional security evaluation of IoT deployments, the device and network communication components are the only two considered, which creates a blind spot in the security evaluation results. Our approach has shown that cloud backends and mobile applications contain security flaws. Furthermore, the cloud backend component analysis uncovers serious flaws that not only does it allow attackers to compromise one deployment but many. Numerous flaws can give attackers control of the IoT deployment or disclose sensitive information about the users or systems depending on the IoT deployment. For example, in the case of the device, a nearby attacker can hijack the setup process by configuring the device before the legitimate user allowing complete control of the device. Even if we assume a secure device deployment, a user may deploy the device in an open network that gives open access to any user. An attacker on the local network can abuse the unauthenticated UPnP service found on the *Belkin Netcam* to hijack the device. Furthermore, the attacker on the local network can also view video footage from the camera directly or expose the video stream to the internet.

For the mobile application, no direct attack vectors would give an attacker control over the IoT deployment. However, an attacker can use the hard-code app API key for the service *Localytics* to poison the results and feed false information to the *Belkin Localytics* account. The network communication appears to allow local network attackers to intercept and manipulate traffic from the mobile application to the device. The communication between the mobile app and the device does not use encryption. The cloud backends have critical security flaws that can give an attacker control over many *Belkin Netcam* deployments. For example, an attacker can exploit the JBOSS remote code execution flaw to gain access to the backend server and use it to abuse all *Belkin Netcam* devices that connect to that server. Using our framework, we can systematically map out the entire attack surface for an IoT deployment and find *more* security flaws than a traditional security evaluation approach. In chapter 6, we will use these findings to provide a *more* holistic risk assessment of IoT deployments.

3.5 Proposals

Our large-scale security evaluation of diverse devices gives a unique perspective into good and poor security practices. We found cloud-managed devices to be, in general, more secure. Moreover, devices automatically updating without involving the user tend to be more secure and contain fewer flaws. Lastly, device communication over the local network appears to have a lax security posture, which can allow an attacker to compromise the target device. Based on these observations, we discuss several mitigation strategies and recommendations.

3.5.1 Mitigations

Device. Affected devices should patch through secure channels to ensure the integrity of the update. Vendors can limit running services on IoT devices and follow a client approach where the device is managed through cloud endpoints using push/pull requests.

Device configurations can be remedied using a configure-before-operable approach, where the device will not activate without proper configuration and setup. Many devices follow a configure-before-operable approach, and it should be mandated by industry standards. Finally, endpoint (cloud or mobile) verification ensures that only authenticated parties can interact with the device. Vendors can limit the interaction to a sandboxed environment and assign temporal fine-grained access control for required resources. Trusted endpoints should not operate with unfettered access, and devices should enforce authentication time-outs for **all** parties. Modern home-based IoT devices are equipped with enough compute power ([163, 164]) to apply many of the suggested mitigations, contrary to the popular belief that they are under-powered and energy-constrained devices.

Mobile. Over-privileged applications can have privacy concerns regarding users' activities. Mobile platforms should implement a system to derive permissions based on functional analysis of the application and grant permissions temporarily at runtime. Further, sensitive information, such as API keys, should be derived when the application is installed on the mobile device and stored in an encrypted key store. Cryptographic protocols are difficult to implement correctly, and therefore developers should rely on mature libraries with proper implementations. Finally, developers should adhere to the recommended guidelines that accompany these libraries.

Cloud. Managed platforms and configuration management tools can alleviate the vulnerable services on the cloud endpoints. Vendors should utilize commercial platforms that are managed by experienced professionals. Similarly, automating cloud endpoint configuration through API integration can reduce the chances of misconfiguration. For example, Let's Encrypt [165] can automatically renew certificates for servers. Cloud endpoints should not support insecure protocols, but instead, they should verify both endpoint devices and mobile applications.

Communication. Network communication between all IoT components should adhere to the same security standards (LAN or Internet). Vendors must use the latest secure pro-

tocols, offer limited functionality for backward compatibility, enforce protocol upgrade requirements, and verify endpoints. Endpoint verification will ensure MITM attacks are not successful and protect the integrity of the communication. Vendors should default to a fail state if endpoints are not verifiable. Additionally, vendors can provide an option to install custom certificates in IoT deployments for transparency.

3.5.2 Stakeholders

Vendors. Vendors have to get the security requirements correct for every component at every level, including the design, implementation, and deployment of IoT systems. Our evaluation shows that many vendors strive for device security but often fail with due diligence. Realistically, many vendors do not have **all** the expertise to develop, manage, and deploy these heterogeneous technologies. Vendors that lack expertise in specific areas can outsource to specialized third-parties to develop their products.

End-Users. Home-based IoT deployments transform simple home-based networks into complex enterprise-like networks. End-users can follow good security practices by configuring devices to use encryption, disable remote administration features, and segment their network. Most importantly, consumers can influence vendors by purchasing privacy-aware and secure devices. Our portal is meant to give an objective security assessment of IoT devices and allow consumers to make informed decisions.

Other Parties. Internet Service Providers (ISPs) are not direct stakeholders, but the ubiquity of home-based IoT devices affects the operation of their networks. Much of the traffic seen by the ISPs will be encrypted, but ISPs can identify devices by destinations, service ports, and communication frequency. ISPs can potentially implement technical remedies to block certain ports, but legal policies are needed to intervene. These decisions can pose policy and compliance disputes due to the global nature of IoT and international jurisprudence [166]. ISPs can offer their expertise in running and operating residential Internet networks that can help identify implications around home-based IoT deployments.

Cloud providers offer infrastructure-as-a-service to many IoT vendors and have years of experience in developing, running, and securing cloud infrastructures and platforms. Their offerings are economical and practical for vendors, but they do suffer from outages occasionally [63]. Cloud providers are playing an important role in securing IoT deployments and should continue to offer tailored cloud services that alleviate security responsibilities from vendors.

3.5.3 Recommendations

Measurements. We recommend additional measurements for inter-device communication, mobile application-to-device interaction, and trust relationship between IoT components. Inter-device communications (device-to-device and mobile application-to-device) are not well studied within the LAN. Many IoT systems, like home assist devices, auto-discover and interact with other devices on the LAN without users' consent, which warrants further investigation to understand the security and privacy implications as a result of these communications. Further, conducting longitudinal studies can expose latent flaws that are otherwise difficult to observe without temporal analysis.

Standards. Many well-established vendors have put forth standards for IoT systems, but there is no consensus among the community. Vendors and researchers should combine their expertise to jointly draft industry standards that provide techniques to address common mistakes found in home-based IoT systems. Some home-based IoT systems have cyber-physical components, like connected ovens, fridges, and water heaters. These classes of IoT systems must be regulated by safety mandates and code standards to ensure no physical harm can result from their abuse or component failure. The government must play an active role in the development of these standards to protect consumers' safety and privacy.

Table 3.3: This table is a summary of each evaluated device per graph component in Figure 3.2. The device section summarizes the number of running services and issues found. The mobile application summarizes excessive permissions, sensitive data, or incorrectly use of cryptographic protocols. The communication category summarizes the susceptibility to MITM attack and the communication channel state as fully encrypted (●), partially encrypted (◐), or not encrypted (○).

Device	Device Services Table 3.4		Mobile Application Table 3.6			Cloud Endpoints Table 3.7		Communication Table 3.8	
	Running Services	Security Issues	Over-priviliged	Sensitive Data	Crypto Issues	SSL Issues	Service Issues	MITM	Encryption
Amazon Echo	1	0	✓	✓		✓			●
Amazon Fire TV	1	0	✓	✓			✓		◐
Apple HomePod	4	0	—	—	—	✓			●
Apple TV (4th Gen)	3	0	—	—	—	✓			●
August Doorbell	1	0	✓	✓		✓	✓	✓	◐
Belkin Netcam	1	1		✓		✓	✓	✓	◐
Belkin WeMo Crockpot	0	0		✓		✓	✓	✓	◐
Belkin WeMo Link	1	1		✓			✓	✓	◐
Belkin WeMo Motion	1	1		✓		—	—	✓	◐
Belkin WeMo Switch	1	1		✓		✓	✓	✓	◐
Bose SoundTouch 10	4	1	✓	✓	✓	✓	✓	✓	◐
Canary	0	0	—	—	—	✓			●
Caseta Wireless	2	0	✓			—	—	✓	◐
Chamberlain myQ Garage Opener	1	0			✓	✓			●
Chinese Webcam	4	1	—	—	—		✓	✓	○
D-Link DCS5009L	3	2	✓		✓	✓		✓	○
Google Home	5	2	✓		✓	—	—	✓	◐
Google Home Mini	5	2	✓		✓	—	—	✓	◐
Google OnHub	1	0	✓		✓	—	—		◐
Harmon Kardon Invoke	5	1		✓		✓	✓		●
Insteon Hub	4	6	✓	✓		✓	✓	✓	○
Koogeek Lightbulb	2	0	✓		✓	—	—		●
LIFX Virtual Bulb	0	0	✓		✓	✓		✓	◐
Logi Circle	0	0	✓			✓	✓		●
Logitech Harmony	2	1	✓			—	—		◐
MiCasaVerde VeraLite	4	6	—	—	—	✓	✓	✓	◐
Nest Cam IQ	0	0	✓	✓	✓	✓			●
Nest Camera	0	0	✓	✓	✓	✓			●
Nest Guard	0	0	✓	✓	✓	✓	✓		●
Netgear Arlo	0	0	✓	✓	✓		✓		●
nVidia Shield	2	3	—	—	—	—	—		◐
Philips HUE	2	0	✓	✓		—	—	✓	◐
Piper NV	3	0	—	—	—	✓	✓		●
Ring Doorbell	0	0	✓		✓				●
Roku 4	2	0	✓		✓	✓	✓	✓	◐
Roku TV	2	0	✓		✓	✓	✓	✓	◐
Roomba	1	0	✓	✓	✓	—	—		◐
Samsung SmartThings	1	1	✓	✓	✓	✓			●
Samsung SmartTV	4	1			✓	—	—	✓	◐
Securifi Almond	2	1	✓		✓	—	—		◐
Sonos	3	3		✓		—	—	✓	◐
TP-Link WiFi Bulb	1	0	✓		✓	—	—		●
TP-Link WiFi Plug	0	0	✓		✓	—	—		●
Wink 2 Hub	4	4	✓	✓	✓	—	—	✓	◐
Withings Home	1	0	✓				✓		●

Table 3.4: Device Evaluation.

Device	System Services			System Setup		
	Detected OS	Running Services	Issues Found	Pairing	Config.	Upgrade
Insteon Hub	Linux 2.6	4	6	Wired+Pin	F	C
MiCasaVerde VeraLite		4	6	Cloud+Pin	D	M
Wink 2	Linux 2.6	4	4	Wired	D	M
Sonos	Linux	3	3	Wired	D	C
nVidia Shield	Linux 2.6	2	3	Wired	D	M
Google Home	Linux 3.3	5	2	Wifi	F	A
Google Home Mini	Linux 3.3	5	2	Wifi	F	A
D-Link DCS5009L		3	2	Wired	D	M
Harmon Kardon Invoke	Linux 3.3	5	1	LE	F	A
Bose SoundTouch 10	Linux 2.6	4	1	LE	F	C
Chinese Webcam	Linux 2.6	4	1	Wired+HTTP	D	N/A
Samsung SmartTV	Linux 4.8	4	1	On-Screen	D	M
Logitech Harmony		2	1	LE	F	M
Securifi Almond	Linux 2.6	2	1	Wired	D	M
Belkin Netcam	Linux 2.6	1	1	Wifi+Pin	F	C
Belkin WeMo Link		1	1	Wifi+Pin	F	C
Belkin WeMo Motion		1	1	Wifi+Pin	F	C
Belkin WeMo Switch		1	1	Wifi+Pin	F	C
Samsung SmartThings	Linux 2.6	1	1	Wired	F	C
Apple HomePod	FreeBSD 6	4	0	LE	F	C
Apple TV (4th Gen)	tvOS	3	0	Wired	F	C
Piper NV		3	0	Wifi	F	A
Caseta Wireless	Linux 2.6	2	0	Wired	F	M
Koogeek Lightbulb		2	0	LE+Pin	D	C
Philips Hue	Linux 2.6	2	0	Wired+Button	F	C
Roku 4	Linux 3.3	2	0	Wired	D	M
Roku TV		2	0	Wired	D	M
Amazon Echo	Linux	1	0	Wifi	F	A
Amazon Fire TV	Linux 2.6	1	0	Wired	D	M
August Doorbell	Linux 2.6	1	0	Wifi	F	M
Chamberlain myQ Garage Opener		1	0	Wifi	F	M
Google OnHub	Linux 4.8	1	0	Wired	F	A
Roomba		1	0	Wifi	F	M
TP-Link Wifi Bulb		1	0	Wifi	D	M
Withings Home		1	0	LE	F	C
Canary		0	0	Wifi	F	C
LIFX Bulb		0	0	Wifi	D	M
Logi Circle		0	0	LE	F	A
Nest Cam IQ		0	0	Wired	F	A
Nest Camera		0	0	LE+Pin	F	A
Nest Guard		0	0	Wired	F	A
Netgear Arlo		0	0	Wired	F	M
Ring Doorbell		0	0	Wifi	F	M
TP-Link Wifi Plug		0	0	Wifi	F	M
Belkin WeMo Crockpot		0	0	Wifi+Pin	D	C

(**F**)orced configuration change when device is setup; (**D**)efault device configuration is acceptable and allows device to operate. (**C**)onsent by the user is required for the device to upgrade; (**A**)utomatic updates are applied without user intervention; (**M**)anual device update via user request. N/A means the category is not applicable.

Table 3.5: List of devices and their CVEs with CVSS score of Critical and High.

Device	CVE	CVSS
MiCasa Verda VeraLite	CVE-2012-5958, CVE-2012-5959, CVE-2012-5960, CVE-2012-5961, CVE-2012-5962, CVE-2012-5963, CVE-2012-5964, CVE-2012-5965, CVE-2013-4863	Critical
	CVE-2012-0920	High
Wink 2	CVE-2016-7406, CVE-2016-7407	Critical
	CVE-2016-7408	High

Table 3.6: Mobile Application Evaluation.

Device	Mobile Application			Overprivileged	Sensitive Data	Crypto Issues
	Name	Platform	Version			
Securifi Almond	com.securifi.almond	iOS	3.5.6	✓		✓
LIFX Virtual Bulb	com.lifx.lifx	iOS	3.8.6	✓		✓
Ring Doorbell	com.ring	iOS	4.1.13	✓		✓
Roku TV Roku 4	com.roku.ios.roku	iOS	4.2.3	✓		✓
Netgear Arlo Camera	com.netgear.arlo	iOS	2.4.8	✓	✓	✓
TP-Link WiFi Plug TP-Link WiFi Bulb	com.tplink.kasa-ios	iOS	1.11.1	✓		✓
Chamberlain myQ Garage Opener	com.chamberlain.myq.chamberlain	iOS	6216.0.0			✓
Google Home Mini Google Home	com.google.Chromecast	iOS	1.28.508	✓		✓
Apple HomePod	—	iOS	—	—	—	—
Wink 2	com.quirky.wink	iOS	6.8.0	✓		✓
Google OnHub	com.google.android.apps.access.wifi.consumer	Android	jetstream BV10127	✓		✓
Samsung SmartThings	com.smartthings.android	Android	2.13.0	✓	✓	✓
Philips HUE	com.philips.lighting.hue2	Android	2.19.0	✓	✓	
Insteon Hub	com.insteon.insteon3	Android	1.9.8	✓	✓	
Sonos	com.sonos.acr	Android	8.3.1		✓	
Nest Camera Nest Cam IQ Nest Guard	com.nest.android	Android	5.17.0.31	✓	✓	✓
Belkin WeMo Motion Belkin WeMo Switch Belkin WeMo Link Belkin WeMo Crockpot	com.belkin.wemoandroid	Android	1.19.0		✓	
Amazon Echo	com.amazon.dee.app	Android	2.2.1615.0	✓	✓	
Belkin Netcam	com.belkin.android.androidbelkinnetcam	Android	2.0.4		✓	
Amazon Fire TV	com.amazon.storm.lightning.client.aosp	Android	1.0.13.18	✓	✓	
D-Link DCS5009L	com.dlink.mydlinkunified	Android	1.0.3	✓		✓
Logitech Logi Circle	com.logitech.circle	Android	2.3.2220	✓		
Canary	is.yranac.canary	Android	2.14.0	—	—	—
Piper NV	com.blacksumac.piper	Android	1.4.0	—	—	—
Withings Home	com.withings.home	Android	1.5.3	—	—	—
MiCasaVerde VeraLite	com.vera.android	Android	7.25.47	✓		
August Doorbell Cam	com.august.luna	Android	6.1.4	✓	✓	
Logitech Harmony	com.logitech.harmonyhub	Android	5.1.1	✓		
Caseta Wireless	com.lutron.mmw	Android	5.1.0	✓		
Bose SoundTouch 10	com.bose.soundtouch	Android	17.170.82	✓	✓	✓
Harmon Kardon Invoke	com.microsoft.cortana	Android	2.10.2.2135		✓	
Roomba	com.irobot.home	Android	2.3.1	✓	✓	✓
Samsung SmartTV	com.samsung.smartviewad	Android	2.1.0.100			✓
Koogeek Lightbulb	com.tomtop.home	Android	1.2.2	✓	✓	✓
nVidia Shield	—	—	—	—	—	—
Chinese Webcam	—	—	—	—	—	—

Table 3.7: Cloud Endpoint Evaluation.

Device	Domains						SSL			Services			
	Total	1st Party	3rd Party	Hybrid	Unknown	Host	Self-Signed	Domain Mismatch	Vuln SSL	Outdated OS	Information Discolsure	Cleartext Auth	Exploitable Service
Amazon Echo	221	15	191	3	12	17	✓		✓	—	—	—	—
Amazon Fire TV	174	100	17	14	43	99	—	—	—				✓
Apple HomePod	182	80	6	76	20	113	✓	✓	✓	—	—	—	—
Apple TV (4th Gen)	439	170	14	188	67	38	✓		✓	—	—	—	—
August Doorbell	55	7	12	34	2	32	✓		✓			✓	
Belkin Netcam	79	13	63	1	2	12			✓		✓	✓	✓
Belkin WeMo Crockpot	27	7	15	5	0	11			✓	✓	✓		✓
Belkin WeMo Link	14	4	6	4	0	11	—	—	—	✓			✓
Belkin WeMo Motion	24	7	12	5	0	9	—	—	—	—	—	—	—
Belkin WeMo Switch	29	5	19	5	0	10			✓	✓			✓
Bose SoundTouch10	26	10	10	6	0	11			✓		✓	✓	
Canary	22	19	3	0	0	9	✓			—	—	—	—
Caseta Wireless	22	2	11	5	4	6	—	—	—	—	—	—	—
Chamberlain myQ Garage Opener	1	1	0	0	0	1	✓			—	—	—	—
Chinese Webcam	1	1	0	0	0	1	—	—	—			✓	
D-Link DCS5009L	4	4	0	0	0	3			✓	—	—	—	—
Google Home	42	29	3	0	10	14	—	—	—	—	—	—	—
Google Home Mini	40	27	3	0	10	17	—	—	—	—	—	—	—
Google OnHub	24	24	0	0	0	15	—	—	—	—	—	—	—
Harmon Kardon Invoke	128	0	108	5	15	9	✓		✓		✓		
Insteon Hub	20	2	12	5	1	5	✓				✓		
Koogeek Lightbulb	1	0	1	0	0	0	—	—	—	—	—	—	—
LIFX Virtual Bulb	3	2	1	0	0	1	✓		✓	—	—	—	—
Logitech Harmony	17	6	5	6	0	8	—	—	—	—	—	—	—
Logitech Logi Circle	341	158	5	178	0	20	✓						✓
MiCasaVerde VeraLite	74	1	30	43	0	40	✓		✓	✓	✓	✓	✓
Nest Cam IQ	9	4	5	0	0	4	✓			—	—	—	—
Nest Camera	7	6	1	0	0	5	✓		✓	—	—	—	—
Nest Guard	14	6	6	2	0	4	✓				✓		
Netgear Arlo	59	23	2	7	27	18	—	—	—				✓
nVidia Shield	261	23	177	3	58	24	—	—	—	—	—	—	—
Philips HUE	27	14	8	0	5	11	—	—	—	—	—	—	—
Piper NV	42	24	16	2	0	16	✓		✓		✓		
Ring Doorbell	9	5	3	1	0	6	—	—	—	—	—	—	—
Roku 4	231	37	177	4	13	28	✓		✓		✓		
Roku TV	226	36	144	6	40	28	✓		✓		✓	✓	
Roomba	11	2	5	4	0	5			✓	—	—	—	—
Samsung SmartThings	10	6	1	3	0	4	✓		✓	—	—	—	—
Samsung SmartTV	182	27	138	2	15	20			✓	—	—	—	—
Securifi Almond	938	9	0	0	929	6			✓	—	—	—	—
Sonos	65	13	34	7	11	1	—	—	—	—	—	—	—
TP-Link WiFi Bulb	11	3	7	1	0	4	—	—	—	—	—	—	—
TP-Link WiFi Plug	11	3	7	1	0	4	—	—	—	—	—	—	—
Wink 2	12	3	7	2	0	4	—	—	—	—	—	—	—
Withings Home	20	12	2	2	4	9	—	—	—			✓	

Table 3.8: Communication Evaluation.
✓+ (TLS/SSL) — ✓- (3rd-party recursive DNS)

Device	Observed IP Communication					MITM			Encryption		
	DNS	HTTP	UPnP	NTP	Custom	D-C	A-C	A-D	D-C	A-C	A-D
Google OnHub	✓-	✓+			✓	✗	✗	—	◐	●	—
Samsung SmartThings	✓	✓+	✓			✗	✗	✗	●	●	—
Philips HUE	✓	✓+	✓	✓		✗	✗	✓	●	●	○
Insteon Hub	✓	✓		✓		✓	—	—	○	—	—
Sonos	✓	✓+	✓		✓	✗	✓	✓	◐	○	○
Securifi Almond	✓			✓	✓	✗	✗	—	◐	●	—
Wink 2 Hub	✓	✓+	✓	✓		✓	✗	✓	●	◐	○
Belkin WeMo Motion Belkin WeMo Switch Belkin WeMo Link Belkin WeMo Crockpot	✓	✓+	✓	✓	✓	✗	✗	✓	◐	◐	○
LIFX Bulb	✓			✓	✓	✗	✗	✓	●	◐	○
Amazon Echo	✓	✓	✓	✓	✓	✗	✗	—	●	●	—
Belkin Netcam	✓	✓+	✓		✓	✗	✗	✓		◐	—
Ring Doorbell	✓				✓	✗	✗	—	●	●	—
Roku TV Roku 4	✓	✓+	✓		✓	✗	—	✓	●	—	○
Amazon Fire TV	✓	✓+	✓			✗	—	✗	●	—	◐
nVidia Shield	✓	✓+			✓	✗	—	—	◐	—	—
Apple TV (4th Gen)	✓	✓+		✓	✓	✗	—	✗	●	—	●
Netgear Arlo	✓	✓+		✓		✗	✗	—	●	●	—
D-Link DCS-5009L			✓			—	—	✓	—	—	○
Logi Circle	✓	✓+		✓		✗	✗	—	●	●	—
Canary	✓	✓+				✗	✗	—	●	●	—
Piper NV	✓-	✓+		✓	✓+	✗	✗	—	●	●	—
Withings Home	✓	✓+			✓	✗	✗	✗	●	●	●
MiCasaVerde VeraLite	✓	✓+	✓	✓	✓	✗	✓	✓	◐	●	○
Chinese Webcam		✓	✓		✓	✓	—	✓	○	—	—
August Doorbell	✓	✓+				✓	✗	✓	◐	◐	○
TP-Link WiFi Plug TP-Link WiFi Bulb	✓	✓+		✓		✗	✗	✗	●	●	●
Chamberlain myQ Garage Opener	✓				✓	✗	✗	—	●	●	—
Logitech Harmony	✓	✓+	✓			✗	✗	—	◐	◐	—
Caseta Wireless	✓	✓		✓	✓	✗	✗	✓	◐	●	○
Google Home Mini Google Home	✓-	✓+	✓	✓	✓	✗	✗	✓	◐	●	○
Bose SoundTouch 10	✓	✓+	✓		✓	✗	✗	✓	●	◐	○
Harmon Kardon Invoke	✓-	✓+		✓		✗	✗	—	●	●	—
Apple HomePod	✓	✓+		✓	✓	✗	—	—	●	—	—
Roomba	✓	✓+		✓	✓	✗	✗	✗	●	●	○
Samsung SmartTV	✓	✓+	✓			✗	—	✓	◐	—	◐
Koogeek Lightbulb	✓			✓	✓	—	✗	✗	—	●	●
Nest Camera		✓+				✗	✗	—	●	●	—
Nest Cam IQ	✓-	✓+				✗	✗	—	●	●	—
Nest Guard				✓	✓	✗	✗	—	●	●	—

CHAPTER 4

LONGITUDINAL ANALYSIS

In the last chapter, we conducted a large-scale security evaluation of 45 COTS IoT devices. The result provided a single snapshot of the security state of the device services, network communication, cloud endpoints, and companion mobile app. The security evaluation needs to iteratively characterize the security posture of the IoT device since the device can update. These updates can improve, degrade, or stagnate the device's security posture. We provide two different longitudinal analyses in this chapter to investigate this matter. The first study takes the original security assessments and compares the device services and network encryption before and after applying device updates. Recall that our initial deployment and configuration of the devices disabled automatic updates. The second study takes 18 months and analyzes the device services and vulnerabilities weekly. The first study provides an overview of the effect of updates, while the second study provides a better characterization of the device's security throughout its lifecycle.

4.1 Background

4.1.1 Longitudinal Studies of IoT Deployments

Two primary works explore the longitudinal aspect of IoT security. One study explicitly focuses on TLS communications of consumer IoT devices [114]. The second study scans the internet for exposed Industrial Control System (ICS) devices and compares the change over time [167]. Our work studies the security of device services and network communications deployed in a testbed. Unlike prior work [114], we study the vulnerabilities for all services and network protocols found on a device. Moreover, our testbed provides better visibility and more control than internet scans because we can physically interact with the

devices and verify observations. Internet studies of IoT devices [167] can be honeypots that may impact the results and conclusions.

4.1.2 Goals

This study aims to understand two aspects of IoT deployments. First, how do updates impact devices' security? Second, what are the trends in device vulnerabilities over time? The first question explains how well a security vendor update improves the overall device's security. This is important to empirically show because the prevailing assumption is that a IoT update will address all security problems. However, as we will see in later sections, that assumption is untrue. The second question provides insights into how quickly a vendor addresses vulnerabilities as they arise. This is important to study because it can inform the risk calculation of the IoT deployment.

4.1.3 Evaluation Scope

We limit our evaluation scope to the device services. We exclude the longitudinal analysis of the companion mobile app, the cloud endpoints, and network communications. As shown in Table 3.1, the analysis aspect of including every component is manual and requires weekly testing of 45 devices, which is impossible given our resources. The manual analysis is essential to characterize each component's security posture accurately. The evaluation component of our framework is automated using a mix of commercial and open-source tools. These tools can have high false positives, and we must manually validate them. To prioritize our time, we study the longitudinal component of the device and network communication. We leave the longitudinal analysis of cloud endpoints and mobile apps for future work.

4.2 Methodology

4.2.1 Devices

Our initial evaluation included 45 COTS devices. Over time, some devices malfunctioned, and we had to decommission them. Moreover, we added additional devices to the lab to account for newer devices. This change only affects the second study that analyzes 13 months of device activity. The following is a list of devices decommissioned:

1. Chinese Webcam
2. Withings Home
3. Caseta Hub
4. LIFX Lightbulb
5. Harmon Invoke
6. Koogeek Lightbulb
7. Roku TV

The following are new device additions to the testbed:

1. Amazon Echo Show
2. Western Digital myCloud Home
3. Western Digital myCloud EX2
4. Rachio 3
5. Eufy HomeBase
6. AVTech Network Camera

7. Axis IP Cam

We apply the onboarding method described in chapter 3 for new devices.

4.2.2 Data Collection and Analysis

We apply the method described in chapter 3 for the data collection. We depend on the Nessus Scanner for automated device scanning. We use Nessus Scanner [154] to scan devices for service discovery, service profiling, and vulnerability assessment. The scanner uploads the results to the Tenable cloud portal, where the Tenable engine indexes the data for analysis. We use the Tenable portal to investigate specific devices and their associated vulnerabilities manually. We use this approach for study one. However, for the weekly analysis (study two), we exported the historical data from the Tenable cloud backend and analyzed it offline.

Study One: Update Impact. Our initial evaluation was done in March 2018, which leaves a big time gap between more recent evaluations. Additionally, we did not have a complete accounting of updates for each device, which can either be automatic, require consent, or manual. To objectively study the lifecycle change of IoT devices, we conduct a baseline evaluation and an update evaluation. The baseline evaluation is a device-only evaluation that focuses on identifying services on an IoT device and any known security issues associated with those services. The update evaluation applies all available updates for each device and then reevaluates them to identify changes in the services and their security issues. Our baseline evaluation was done on 21^{st} of April 2019 and the update evaluation was done on 9^{th} of June 2019.

During these 50 days, we track devices that update automatically to identify if they apply any updates to the IoT device. Specifically, we manually check on a weekly basis if the firmware version numbers for the devices change due to updates. All of the devices that automatically are updated had at least one update between the two evaluation periods. The remaining devices require user consent or manual update, which we applied a week before

(2^{nd} of June 2019) our update evaluation. We confirmed the updates were applied successfully by manually checking the firmware version for each device before the baseline and update evaluation. This controlled experiment ensures that the measurements are accurate and represent an update in the software of the IoT devices.

Study Two: Longitudinal Analysis. Recall that some devices may go into sleep mode or go offline, which the Nessus scanner cannot scan. We include the scan reports for devices that the scanner successfully scanned at least four times throughout the study window (Dec 2021 to Mar 2023) to improve the accuracy of our results. The four scans must be at least one in Dec 2021, one in Mar 2023, and two others in between. The intuition behind our approach is that we need to capture an initial state, a final state, and at least two samples to characterize the device's security lifecycle accurately. If we only have two scan reports, we miss out on potential vulnerabilities arising in the middle of the study period. The filtering resulted in 36 devices out of the 47 in the testbed. The excluded devices include the August doorbell, TP-Link Wifi Bulb, Sonos, Roomba, Ring Doorbell, Piper NV, Netgear Arlo, Canary, and Amazon Echo. The excluded devices had at most two scans. The 37 devices had a minimum of 23 scan reports and at most 57 scan reports.

Table 4.1: Device evaluation based on an initial evaluation, baseline evaluation, and update evaluation. Red cells show an increase for services and issues and green cells show a decrease in services and issues.

Device Name	Initial Eval. 03-2018		Base Eval. 04-21-2019		Update Eval. 06-09-2019	
	Services	Issues	Services	Issues	Services	Issues
Amazon Echo	1	0	0	0	0	0
Amazon Fire TV	1	0	3	4	4	4
Apple HomePod	4	0	3	3	3	4
Apple TV	3	0	6	77	4	0
August Doorbell	1	0	1	0	1	0
Belkin Netcam	1	1	3	0	3	0
Belkin WeMo Crockpot	0	0	2	0	2	0
Belkin WeMo Link	1	1	2	1	1	1
Belkin WeMo Motion	1	1	1	1	1	1
Belkin WeMo Switch	1	1	1	1	1	1
Bose SoundTouch10	4	1	5	24	5	2
Canary	0	0	0	0	0	0
Caseta Wireless	2	0	4	0	4	0
Chinese Webcam	4	1	4	1	- -	- -
D-Link DCS5009L	3	2	1	1	3	3
Google Home	5	2	4	4	5	7
Google Home mini	5	2	4	4	5	5
Google onHub	1	0	2	0	2	0
Harmon Kardon Invoke	5	1	1	1	1	1
Insteon Hub	4	6	- -	- -	- -	- -
Koogeek Lightbulb	2	0	1	0	1	0
LIFX Virtual Bulb	0	0	0	0	0	0
Logi Circle	0	0	0	0	0	0
Logitech Harmony	2	1	0	0	0	0
VeraLite	4	6	3	18	3	18
Chamberlain myQ	1	0	1	0	1	0
Nest Camera	0	0	0	0	0	0
Nest Guard	0	0	0	0	0	0
Netgear Arlo	0	0	0	0	0	0
Next Cam IQ	0	0	0	0	0	0
nVidia Shield	2	3	4	6	4	5
Philips HUE	2	0	3	0	3	0
Piper NV	3	0	1	0	1	0
Ring Doorbell	0	0	0	0	0	0
Roku 4	2	0	2	0	2	0
Roku TV	2	0	0	0	0	0
Roomba	1	0	1	0	1	0
Samsung SmartThings	1	1	2	4	2	4
Samsung SmartTV	4	1	9	0	9	1
Seurifi Almond	2	1	3	1	0	0
Sonos	3	3	4	1	4	2
TP-Link WiFi Plug	0	0	0	0	0	0
TP-Link WiFi Bulb	1	0	0	0	0	0
Wink 2 Hub	4	4	3	4	3	4
Withings Home	1	0	0	0	0	0

4.2.3 Challenges and Limitations

As seen in the prior section, keeping devices active is challenging and impacts data collection. Moreover, wireless congestion caused several wireless devices to go offline, including Roomba, Ring Doorbell, Piper NV, Netgear Arlo, D-Link DCS Cam, Canary, and Amazon Echo. As the number of devices increases in the lab space, the wireless spectrum because more congested, causing some devices to disassociate from the access point and become unreachable via the network. This problem became more severe as other researchers in adjacent space built a router testbed. Because of these circumstances, we could not study the entire 46 devices in the lab. This limitation may impact the result's generalizability since we have fewer representative devices. We are considering using Faraday cages to better partition the wireless spectrum in a constraint room to address these limitations.

Additionally, analyzing Common Vulnerabilities and Exposures (CVE) and CVSS scores derived from automated tools such as Nessus may be inaccurate. First, CVE numbers are not assigned to all security issues found in IoT devices, which makes them difficult to track and rate. Second, the rating system (CVSS) may not accurately capture the threat model for home-based IoT devices. For example, CVSS severity for cleartext protocol is rated low, while SSL/TLS issues such as an expired certificate are rated medium. Although misconfiguration in encryption weakens the security of the protocol, it remains better than having no encryption at all. Third, essential network services that run on IoT devices are flagged as security issues with a medium severity, which are false positives. For example, many home-based IoT devices run mDNS (port 5353) service to announce their features and enable zero-configuration networking.

In traditional enterprise networks, end-hosts are not expected to run the mDNS service, whereas for home-based IoT devices this is a common practice. Fourth, some devices reach End-of-Life (EOL) or become defective like the *MiCasaVerde VeraLite* and *Chinese Webcam*, respectively. These issues can be problematic when trying to evaluate changes in a device's lifecycle. Lastly, assessment tools, such as Nessus [154], require tailoring for

Table 4.2: Summary of device use of encryption (○- None, ◐- partial, ●- full), issues found in SSL/TLS protocol, and vulnerabilities affecting services. Green shows improvement, red shows decline, and yellow shows improvement but poor encryption.

Device Name	Service Eval. 04-21-2019			Service Eval. 06-09-2019		
	Encrypted	SSL/TLS Issues	Vuln.	Encrypted	SSL/TLS Issues	Vuln.
Amazon Fire TV	◐	2		◐	2	
Apple HomePod	◐	2		◐	3	
Apple TV	○		✓	○		
Belkin WeMo Link	○		✓	○		✓
Belkin WeMo Motion	○		✓	○		✓
Belkin WeMo Switch	○		✓	○		✓
Bose SoundTouch10	○		✓	○		
Chinese Webcam	○			- -	- -	- -
D-Link DCS5009L	○			◐	3	
Google Home	●	4		●	8	
Google Home mini	●	4		◐	6	
Harmon Kardon Invoke	○		✓	○		✓
MiCasaVerde VeraLite	◐		✓	◐		✓
nVidia Shield	●	6		●	5	
Samsung SmartThings	●	4		●	4	
Samsung SmartTV	○			◐	1	
Seurifi Almond	○		✓	- -	- -	- -
Sonos	◐	1		◐	2	
Wink 2 Hub	◐	4		◐	4	

home-based IoT devices and their results have to be reexamined to prioritize the security issues. In the first study, we manually examine each evaluation to ensure the accuracy of the characterized change.

4.3 Results

4.3.1 Study One: The Impact of Updates on Security Posture

We present the results in Table Table 4.1, and the highlights change in green for improvement and red for the decline in the device's overall security. The baseline and update evaluation columns present the number of services and security issues found for each evaluation. We find 10 devices that change between the baseline and update evaluation either by services, issues, or both. We find seven devices that have an increase in services or issues and three devices that have a decrease in services or issues. Moreover, we find nine additional devices that have security issues, which do not change (improve or worsen). This implies new features often do not lead to additional exposures.

Device Improvements

The *Apple TV* device does not update automatically, instead, it requires user consent. The device can be configured to automatically update with recent versions of the firmware. In the baseline evaluation, we find the *Apple TV* device to have multiple issues such as vulnerabilities that affect firmware versions 11 and 12. The update addresses all 77 issues and decreases the number of network services from six to four. The update disables Digital Audio Access Protocol (DAAP) on port 3689 and Xsan Filesystem access on port 49152. We highlight the change in green in Table Table 4.2 and document the vulnerabilities in Table Table 4.3.

Similarly, we find the *Bose SoundTouch10* to require user consent to apply the updates. In the baseline evaluation, we find the *Bose SoundTouch10* to have 24 issues. The update addresses 22 of the 24 issues and does not decrease or increase any of the network services. The remaining two issues are low severity in comparison to the 22 issues that have critical severity. For the *nVidia Shield* device, the issues decrease from six to five by upgrading the SSL/TLS service to disable weak hashing algorithms like MD5. We highlight the change in green in Table Table 4.2 and document the vulnerabilities in Table Table 4.3.

Lastly, the *Belkin WeMo Link* requires user consent to apply updates. The baseline evaluation finds that the device runs two services, DNS server on port 49154 and syseventd on port 52367. The syseventd service allows any network attacker to run system commands on the device without authentication as a root user. The update improves the security of the *Belkin WeMo Link* device by disabling the syseventd service. Additionally, the DNS server allows a network attacker to snoop on the DNS cache, which discloses recently resolved records. This example illustrates the challenge in quantifying changes in IoT device security because some vulnerabilities are not tracked by a CVE number.

Table 4.3: A summary of issues for baseline evaluation. Green rows show fixed issues by updates.

Issue Description	CVE	CVSS	Affected Device
Dropbear SSH Server ¡ 2016.72 Multiple Vulnerabilities	CVE-2016-7406	Critical	MiCasaVerde VeraLite
Web Server Directory Traversal Arbitrary File Access	CVE-2014-3744	Critical	Bose SoundTouch10
UPnP Devices (libupnp) ¡ 1.6.18 Multiple Stack-based Buffer Overflows RCE	CVE-2012-5958	Critical	MiCasaVerde VeraLite
Apple TV ¡ 11.4 Multiple Vulnerabilities	CVE-2018-5383	High	Apple TV
Apple TV ¡ 12.1.1 Multiple Vulnerabilities	CVE-2018-4431	High	Apple TV
Apple TV ¡ 12.1 Multiple Vulnerabilities	CVE-2018-4368	High	Apple TV
Apple TV ¡ 11.4.1 Multiple Vulnerabilities	CVE-2018-4261	High	Apple TV
Dropbear SSH Server Use-after-free Remote Code Execution	CVE-2012-0920	High	MiCasaVerde VeraLite
SSL Medium Strength Cipher Supported (SWEET32)	CVE-2016-2183	Medium	Google Home (mini), nVidia Shield
Apple TV ¡ 12 Multiple Vulnerabilities	CVE-2016-1777	Medium	Apple TV
Dropbear SSH Server ¡ 2013.59 Multiple Vulnerabilities,192.168.0.34	CVE-2013-4434	Medium	MiCasaVerde VeraLite
SSL Certificate Signed Using MD5	CVE-2004-2761	Medium	nVidia Shield, Apple HomePod, SmartThings
DNS Server Cache Snooping	- -	Medium	Belkin WeMo (Link, Motion, Switch), Securifi Almond, MiCasaVerde VeraLite, Harmon Kardon Invoke
SSL Self-Signed Certificate	- -	Medium	Amazon FireTV, Google Home (mini), Apple HomePod, Wink 2, SmartThings, nVidia Shield, D-Link DCS5009L
SSL Certificate Fails Basic Key Usage Extensions	- -	Medium	nVidia Shield
Unencrypted Telnet Server	- -	Medium	Chinese Webcam
SSL Certificate Expired	- -	Medium	Amazon FireTV, D-Link DCS5009L
SSL RC4 Cipher Supported (Bar Mitzvah)	CVE-2015-2808	Low	Amazon FireTV, Wink 2
SSL Certificate RSA Keys Less Than 2048 bits	- -	Low	Apple HomePod, Wink 2, D-Link DCS5009L
SSH Support Weak MAC Algorithms	- -	Low	MiCasaVerde VeraLite
SSH Server Support CBC Mode Ciphers	- -	Low	MiCasaVerde VeraLite
Web Server Basic Authentication w/o TLS	- -	Low	D-Link DCS5009L
Web Server Transmits Cleartext Credentials	- -	Low	Bose SoundTouch10

Device Exposure

In Table Table 4.1, we highlight seven devices that correspond to an increase in services or issues following the update in red. The *Amazon FireTV*, *D-Link DCS5009L* Cam, and *Google Home (mini)* introduce new network services. The *Apple HomePod*, *D-Link DCS5009L* Cam, *Google Home (mini)*, *Samsung SmartTV*, and *Sonos* have an increase in the number of issues. All of the issues increase in Table Table 4.1 correspond to SSL/TLS issues. Table Table 4.3 presents a full listing of these SSL/TLS issues and the affected devices. This observation highlights that SSL/TLS issues are common problems found in home-based IoT devices.

For example, in Table Table 4.2 highlighted in yellow, we see that the *D-Link DCS5009L* enables SSL/TLS service for its network service, but introduces three SSL/TLS issues. Similarly, the *Samsung SmartTV* and *Sonos* enable SSL/TLS for one of its network services and introduces an SSL/TLS issue. Table Table 4.3 documents these issues under *SSL Self-Signed Certificate*, *SSL Certificate Expired*, and *SSL Certificate RSA Keys Less Than 2048 bits*. For each issue in Table Table 4.3 we document related CVEs and their CVSS score.

For the *Google Home mini* device, the update introduces a new service that does not

use encryption. We document this result in Table Table 4.2 in red, where the network services drop from fully encrypted to partially encrypted. This observation shows an increase in exposure when introducing non-encrypted services. The *Apple HomePod* changes the TLS/SSL configuration on one of the network services to support a weaker cipher making the device susceptible to the *SWEET32* vulnerability, documented in Table Table 4.3. Other devices that have issues in the baseline evaluation did not exhibit any change after the update, such as most of the *Belkin WeMo* family of devices, *MiCasaVerde VeraLite*, *Harmon Kardon Invoke*, *Samsung SmartThings*, and the *Wink 2*. This observation shows that issues related to IoT devices generally do not get better over time.

IoT Device Vulnerabilities and Remediations

In Table Table 4.3 we observe a wide range of severity for each issue, which we split between *critical/high* and *medium/low*. Moreover, many issues do not have a CVE number. The green rows represent the issues that the device updates fix. Almost all fixed issues are related to a service vulnerability except for *Web Server Basic Authentication w/o TLS*, which is a feature enhancement that improves the device's overall security. Furthermore, the device updates do not fix all critical/high-severity issues.

For example, *MiCasaVerde VeraLite* has several critical/high severity rated vulnerable services, but applying the device update did not fix any of the issues. The device is labeled as EOL by the vendor, which may explain the lack of security fixes. In the case of the *MiCasaVerde VeraLite* we observe a firmware version number change but do not see any network service change due to the update. This observation shows that EOL devices receive infrequent updates and ignore critical security issues associated with network services. Additionally, critical/high issues seem to be associated with a single device, whereas medium/low issues are associated with multiple devices.

Finally, as noted earlier, the prioritization for issue types can be misleading since a lower CVSS score for a device's issue has higher severity in the context of IoT devices. We

see cleartext issues such as *Web Server Transmits Cleartext Credentials* and *Web Server Basic Authentication w/o TLS* have lower severity than *SWEET32* and *SSL Certificate Signed Using MD5*. When evaluating the change (improvement/exposure) in the device's lifecycle, researchers must consider the context and threat model associated with home-based IoT devices. These observations provide insights into how complex an IoT device's lifecycle is and bring transparency into what researchers and practitioners should prioritize.

Overall, devices will eventually become vulnerable over time. New vulnerabilities will be discovered, but the critical point is how IoT vendors manage their device's security through updates. The *Apple TV* is a good example showing security prioritization through patches that address all critical/high issues. These practices reduce the exposure period and attack window, which reduces the risk associated with the IoT deployment. Further, we observe medium/low severity vulnerabilities persist across updates and several devices. For example, *Self-Signed Certificate* is a common issue found in multiple devices and persisted across updates. Finally, updates can introduce new issues like the example in the *Apple HomePod*.

Takeaway. In this study, we perform additional security evaluations one year after the first evaluation and present our findings. Our study is among the first to provide essential insights for users to understand the complexity of IoT device updates throughout their lifecycle. Our results show that IoT device updates incorporating new features or functional improvements appear not to increase security exposure; however, our subsequent study shows that vulnerabilities will surface throughout the device's lifecycle. Further, devices must address security issues early on to avoid more exposure throughout the device's lifecycle. We find that EOL devices often receive infrequent updates, and vendors forgo fixing critical/high-severity vulnerabilities associated with network services. Devices that do not expose network services have a limited attack footprint and, in general, are more secure.

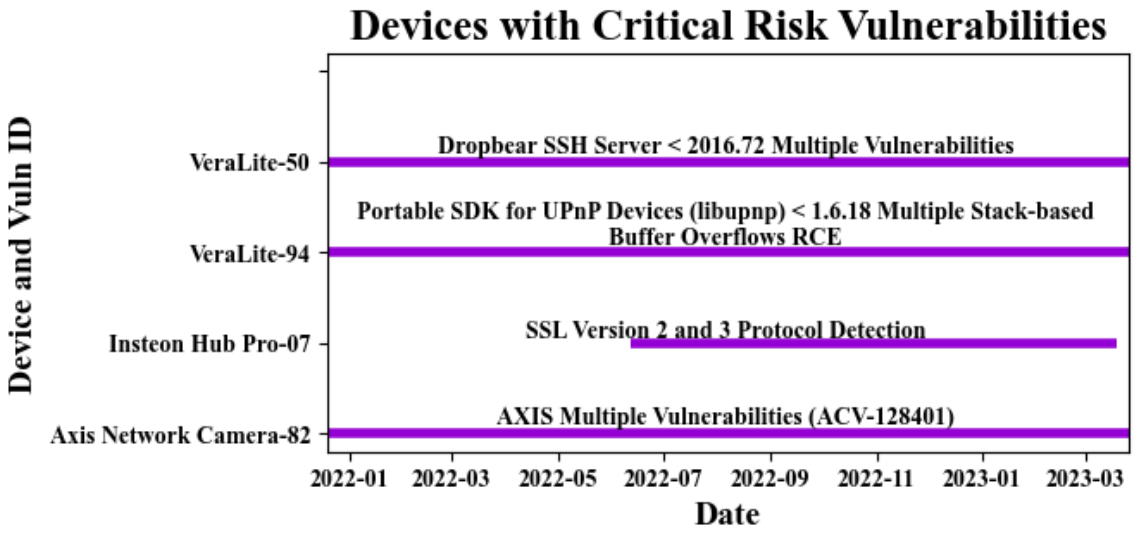


Figure 4.1: Summary of critical CVSS vulnerabilities found in testbed.

4.3.2 Study Two: A Longitudinal Analysis of Devices' Security Lifecycle

In this study, we take a closer look at the update changes with respect to CVEs and their CVSS severity. We already highlighted the challenges and limitations of the current CVE and CVSS scoring system in the context of IoT deployments. For this study, we will use the ratings as is but address the limitations by proposing an alternate risk scoring system in chapter 6.

Critical CVSS Vulnerabilities

In Figure 4.1, we observe three devices with four critical CVSS vulnerabilities. Except for one vulnerability impacting the *Insteon Hub* device, all three vulnerabilities persist throughout the device's lifecycle. The VeraLite, as noted in the prior section, does not address the vulnerability when we apply the vendor-available updates. This device has an EOL status, which can explain the lack of security support. On the other hand, the *Axis Network Camera* requires a manual update. The update process is involved and requires users first to create an account with the vendor, log on to the support page, search and identify the

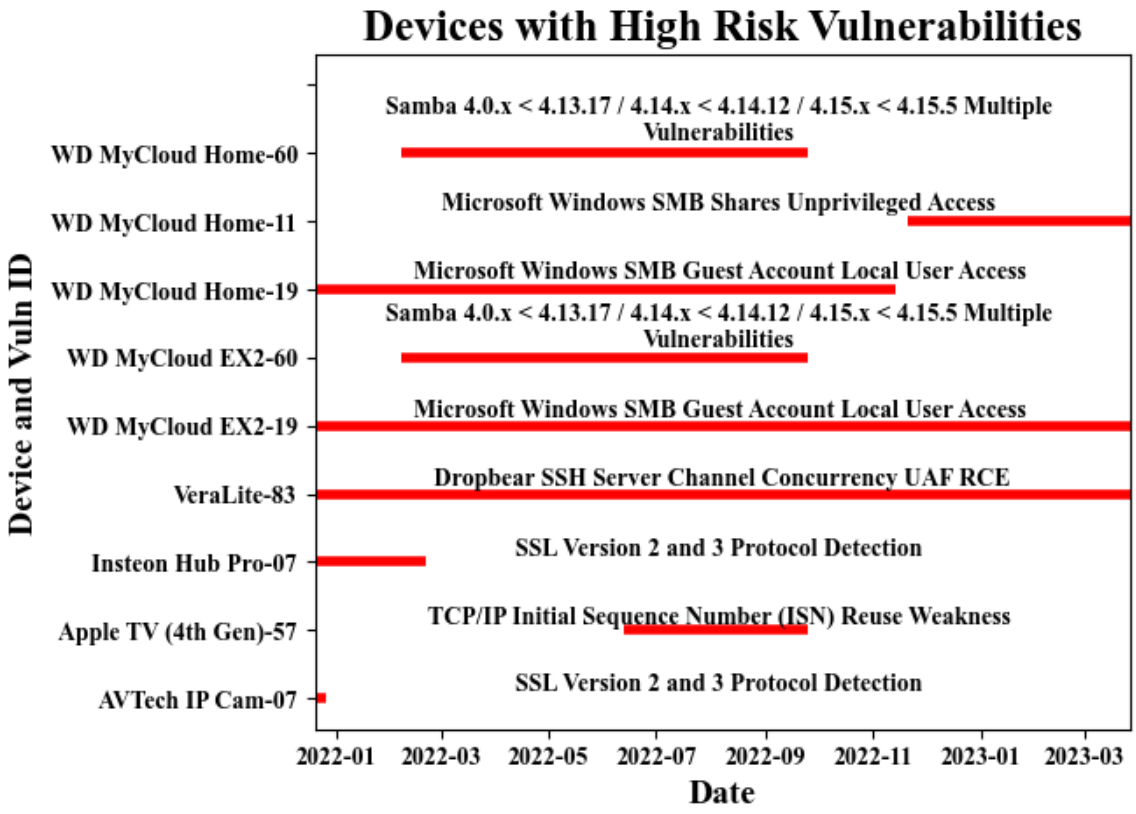


Figure 4.2: Summary of high CVSS vulnerabilities found in testbed.

correct firmware update, download the firmware locally on a personal computer, connect the personal computer to the same network as the *Axis Network Camera*, log on to the camera's backend portal, navigate to the administration page, scroll through several features to find the firmware update link, use the firmware form to upload a copy of the firmware, apply the firmware, restart the device, and finalize the device setup through the camera portal. We went through obtaining the firmware but did not update the device because we were planning to use the current firmware for another experiment. However, the firmware update explicitly addresses the vulnerabilities based on the release notes. Nevertheless, updating the *Axis Network Camera* is esoteric and can be cumbersome for large-scale deployments across enterprise or government facilities. The *Insteon Hub* critical vulnerability emerges around June 2022 and persists onwards.

High CVSS Vulnerabilities

In Figure 4.2, we observe six devices with eight different high CVSS vulnerabilities. The *Insteon Hub* appears to have the same security issue as shown in Figure 4.1. However, the CVSS scoring system appears to have increased the severity from high to critical. Taken together, the SSL Version 2 and 3 vulnerability predates the observation in June 2022. This example highlights the challenges with the current CVSS scoring system and longitudinal tracking of vulnerabilities. The *WD MyCloud Home* has two vulnerabilities: a misconfiguration that users can address and one related to the Samba service. A firmware update after September 2022 addresses the Samba service vulnerability and the Local User Access misconfigurations renamed to Unpriviledged Access. Similarly, we find the *WD MyCloud EX2* suffers from the same issue and updates around the same time (same vendor). The *VeraLite* is an EOL device, and the vulnerability associated with the *SSH* service persists throughout the lifecycle. We observe the *Apple TV* to have a high CVSS vulnerability, but a firmware update fixes the vulnerability two months later. Finally, the *AVTech IP Cam* appears to have a single vulnerability associated with its local TLS service running on port 443. The vulnerability does not appear to persist after the initial observation on December 2021.

Medium CVSS Vulnerabilities

In Figure 4.3, we observe many medium CVSS vulnerabilities. The majority of the vulnerabilities appear to be TLS/SSL related. We find TLS/SSL issues related to deprecated protocol versions, vulnerable versions (Bar Mitzvah, BEAST, POODLE, and SWEET32), weak hashing algorithms, incorrect host names on certificates, expired certificates, and malformed certificates. Some devices address these medium CVSS vulnerabilities like the *Samsung SmartThings*, *WD MyCloud EX2*, *Nvidia Shield*, *Amazon Echo Show*, and *Google Home*. However, we could not understand why some vendors prioritize medium CVSS vulnerabilities while others do not. Interestingly, we observe a distinct pattern between differ-

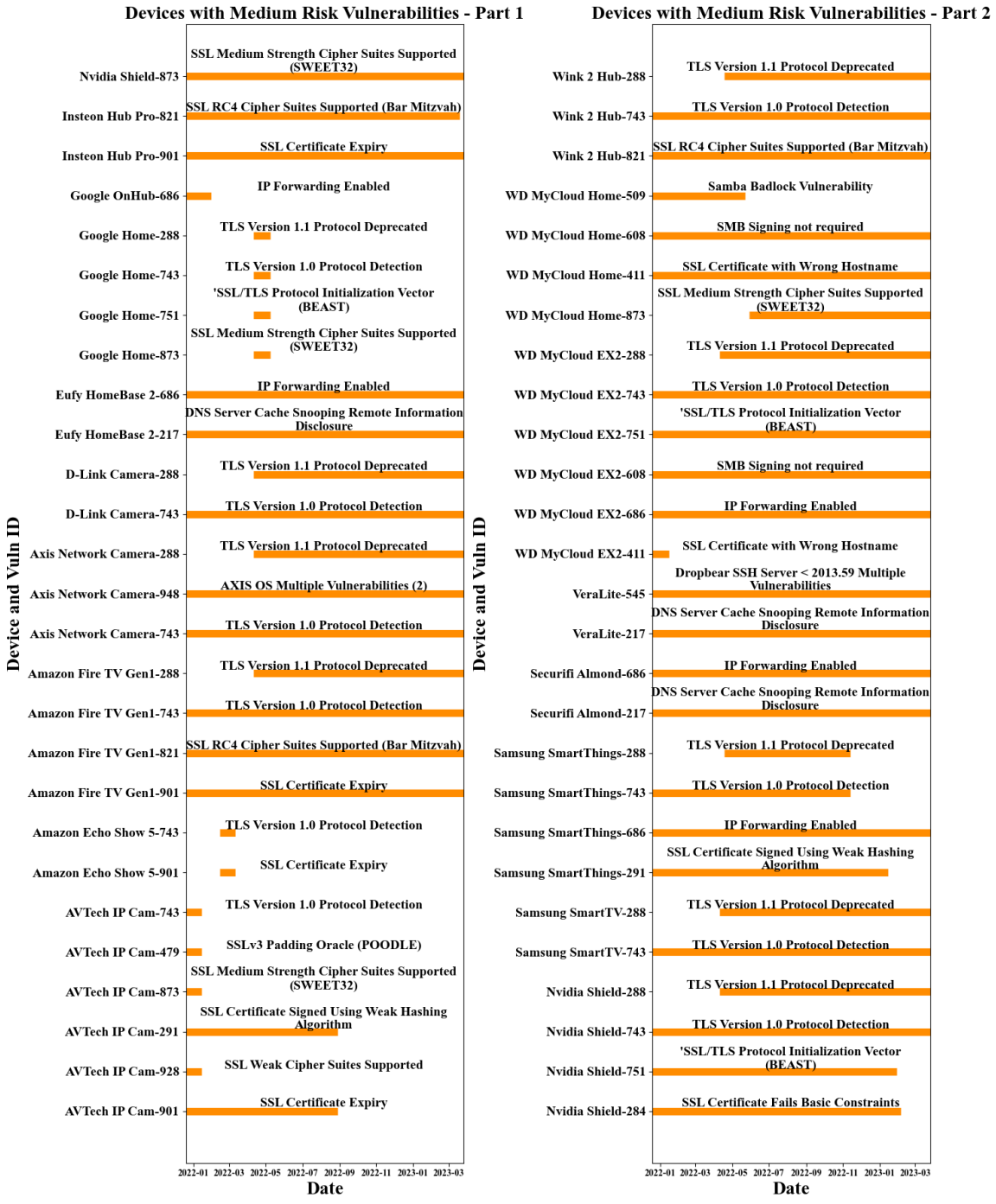


Figure 4.3: Summary of medium CVSS vulnerabilities found in testbed.

ent groups of devices. A few devices have a very short exposure period, while the rest have a long exposure period to the vulnerabilities. For example, the *Amazon Echo Show* and the *Google Home (mini)* have short vulnerability exposure periods. Once vulnerabilities surface, they are addressed almost immediately (within a week).

However, we notice that this behavior is not vendor-related but device-related. For example, the *Amazon Echo Show* and the *Amazon Fire TV* are from the same vendor but have vastly different exposure periods (short vs. long). Another example, the *Google Home (mini)* and the *Google OnHub* have similar exposure periods, but the *OnHub* appears to be longer. The hardware platform and third-party partnerships may contribute to these discrepancies. In the case of Google, the *Google Home* is a Google-based hardware running Google firmware. On the other hand, the *Google OnHub* is a TP-Link-based hardware running Google firmware. For Amazon, the underlying firmware types also differ. The *Fire TV* uses a variant of the Android OS, and the *Echo Show* uses a Linux-based OS tailored for Amazon devices. In these two examples, the heterogeneity between hardware and software platforms potentially contributes to the variation in exposure periods.

Low CVSS Vulnerabilities

In Figure 4.4, we observe six devices with eight low CVSS vulnerabilities. Most of the vulnerabilities appear to impact TLS/SSL and SSH services. These vulnerabilities appear to be configuration-based. It is essential to point out that in the case of the *VeraList* and *WD MyCloud EX2* devices, the users may not have access to the configuration to address the SSH vulnerabilities. Even if the configuration is accessible to users, the users must have the technical knowledge to configure the SSH service properly because the SSH service is not a typical service found or used in IoT deployments. In our testbed, only three out of 37 devices expose SSH service on the network. The SSL Certificate Chain vulnerabilities appear to persist throughout the device's lifecycle. Moreover, low CVSS vulnerabilities are not prioritized by vendors when compared to medium and high CVSS vulnerabilities. We

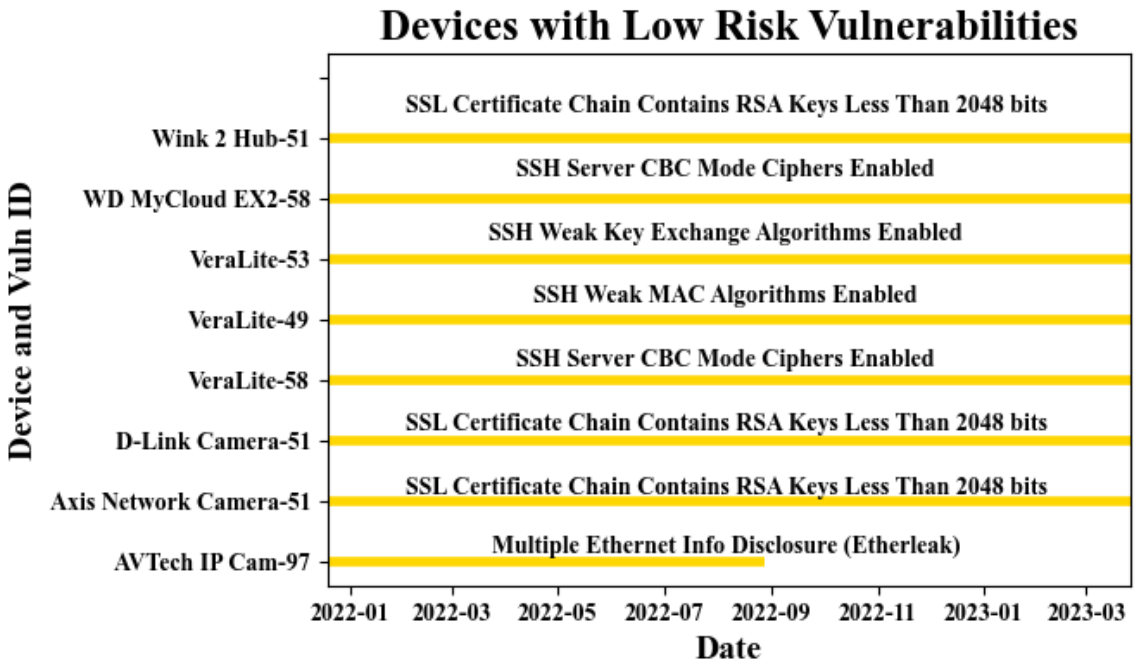


Figure 4.4: Summary of low CVSS vulnerabilities found in testbed.

observe a security fix for a low CVSS vulnerability impacting the *AVTech IP Cam*. The vulnerability relates to an issue that leaks sensitive information via the Ethernet interface. Vulnerable devices may add different data to each Ethernet frame for padding. This data could come from the device's memory or a hardware buffer on its network card. An attacker on the same network as the device can gather sensitive information (passwords, secret keys, and device information).

4.4 Iterative Security Evaluation

Our first study observed that devices not addressing security issues from the beginning likely result in more exposure throughout the device's lifecycle. EOL devices often receive infrequent updates, and vendors forgo fixing critical/high-severity vulnerabilities associated with network services. Devices that do not expose network services have a limited attack footprint and are generally more secure. In our second study, we validated these observations by closely examining the devices' lifecycle for emerging vulnerabilities and

exposure periods. We observe that high and medium CVSS vulnerabilities appear to be prioritized over low CVSS vulnerabilities. Moreover, critical CVSS vulnerabilities in some devices' lifecycles remain for two reasons. First, we show that EOL devices most likely will not receive security updates. Two, devices may have cumbersome update processes making it difficult for users to address critical vulnerabilities. Finally, reclassifying vulnerabilities from high to critical CVSS may create ambiguity in the severity score.

The main goal of the longitudinal analysis is to provide a better understanding of how vulnerabilities evolve over the lifecycle of a device. This information will inform the risk assessment methodology to consider the temporal aspects of vulnerabilities. For example, we learn that critical CVSS vulnerabilities should increase the risk associated with IoT deployment, especially for long exposure periods. On the other hand, high CVSS vulnerabilities can increase or decrease the risk of IoT deployment because the vulnerabilities have varying exposure periods. In the case of the *Apple TV* vulnerability, the exposure period is two months, which can lower the risk associated with IoT deployment because a firmware update fixes the vulnerability. We will use the exposure times as a metric to inform the risk score approach in chapter 6.

CHAPTER 5

LARGE-SCALE ANALYSIS OF THE IOT MALWARE LIFECYCLE

Combining vulnerability and threat assessment can help quantify the overall risk and the potential impact on IoT deployments to protect sensitive information, prevent financial loss, protect physical safety, and ensure compliance. Understanding these threats is necessary to identify potential exploited vulnerabilities and develop security measures or remediations. When conducting risk assessment, we need a basis to understand the patterns and trends in attacks against IoT deployments. Threat trends can inform operators and prioritize decisions about the deployment environment and needed security measures. Unfortunately, traditional threats do not apply directly to IoT deployments because they have unique challenges.

5.1 Challenges and Limitations

Unique Challenges in IoT Attacks. These challenges include device diversity, computational resource limitations, physical exposure, communication protocols, multiple networked components, and limited user interface. The device diversity creates varying hardware and software platforms that require matching threat analysis tools to study their threats. The limited computational resources restrain the security measures that IoT device can implement. The physical exposure of IoT deployments gives attackers additional attack surfaces like low-energy network protocols or physical access to the device. The number of communication protocols found IoT deployment can enable additional attacks. Finally, a compromised IoT device is challenging to identify because the device has a limited user interface to investigate.

The Need for Large-Scale Study. We must conduct a large-scale study to understand IoT threat landscape. A large-scale study can provide a more comprehensive understanding by

analyzing a range of data across time, including malware systems and network behavior, which leads to a better understanding of the IoT threat landscape. Moreover, a large-scale study can identify patterns and trends in attack behavior over extended periods, proactively identifying emerging threats. We use the framework proposed in chapter 2 to study IoT threats. Specifically, we characterize infection, payload, persistent, capability, and C&C communication for 166K IoT malware samples. Our framework deliberately includes reproducibility to allow researchers to replicate the study and validate our findings. This approach increases the scientific rigor of the study and ensures that the results are reliable and can be used to inform security decisions like risk assessment.

Need for IoT Malware Analysis Platform. In order to study IoT malware, we need to develop an extensible malware analysis platform to support the diversity of hardware and software. In Table 5.1, we summarize prior efforts for building an IoT malware analysis platform. We can observe that many solutions either have partial support for analysis, are closed source, or are no longer available online. Open-source dynamic platforms like Lisa [168] Detux [169], and Tamer [170] have partial support for hardware architectures like ARM and MIPS. Furthermore, V-Sandbox [171], which promises many useful features, has yet to be made available by the authors. Lastly, commercial or closed-source sandboxes like IoTBox [143] and Padawan [172] appear to provide a rich analysis platform. However, we need access to the platforms to conduct our large-scale study. We reached out to Padawan's authors, who provided access to the platform. The platform only supported a few architectures and limited the samples we could analyze to a few hundred. As of this writing, the platform is no longer online or accessible. Due to these limitations, we built our own IoT malware analysis platform called *BadThings* [173], which we describe next.

5.2 Methodology

This section will describe our dataset, malware analysis platform, and analysis approach. Our IoT malware analysis platform is more comprehensive than prior efforts and democ-

Table 5.1: A summary of documented or publicly available IoT Malware analysis platforms. ✗ indicates resource no longer available.

Tool Name	Open Source	Open Access	Static Analysis	Binary Emul.	Full System	Supported Arch. ARM	MIPS-EL	MIPS-BE	SPARC	PPC	M68K	SH4
LISA	✓		✓		✓	✓	✓	✓				
Tamer	✓		✓		✓	✓						
IoTBox					✓	✓	✓	✓	✓	✓	✓	✓
Detux	✓				✓	✓	✓	✓				
Padawan		✗			✓	✓	✓	✓		✓		
V-Sandbox	✗					✓	✓	✓		✓		
BandThings	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓

ratizes access to advanced malware analysis techniques for the security community. Our platform includes a large, diverse, and representative malware corpus. Additionally, we made our platform easy to use by building rich analysis virtual machines, containerizing them, and automating their deployment. We create extensive documentation for using the tools, artifacts for each sample in the corpus (reproducibility), and the analysis results (validation).

5.2.1 Data Sources

We list all the dataset sources for our measurements in Table Table 5.2.

VirusTotal. VirusTotal (VT) is a malware analysis and sharing platform that is used by hundreds of commercial security companies and thousands of researchers. We source our dataset from VT and assume that it provides good coverage because of the sheer size of files submitted to the platform, see Figure Figure 5.1. We use VT to identify new binary submissions that meet the following criteria: (1) ELF binaries, (2) never seen by VT before, (3) machine architecture is not $x86$ or $x86\_64$, (4) ELF binary is not *Android* type, (5) submission is not tagged as "shared-lib," "coredump," or "relocatable," (6) file size is less than 30MB, and (7) has at least one anti-virus (AV) detection. We choose these criteria based on the access limitation (10K files/day) and the following assumptions.

First, our work studies malware that target *embedded* IoT systems. The vast majority (82%) of IoT systems rely on Linux-based OS (ELF) [174] and utilize Reduced Instruction Set Computers (RISC) architecture [175], whereas *x86* and *x86_64* are based on Complex

Instruction Set Computers (CISC) architecture, mostly found in servers, desktops, and laptops. We exclude x86, x86_64, and Android malware because (1) they are well covered in prior works [101, 96, 108, 110, 176], (2) are more likely to target mobile or traditional computing devices (servers, desktops, and laptops), and (3) their volume inundate our access capacity, as shown in Figure Figure 5.1.

Second, we found ELF files larger than 30MB to be mostly *coredump* [1], *shared-lib*, or *relocatable* [2]. We found seven files, over 30MB, detected by one or more AV engines and one file detected by five or more AV engines [3]. Third, our analysis pipeline can analyze native ELF binaries, therefore, it does not support Java-based Android apps, but it supports files that run on the *Android Runtime* environment (native). VT classifies files that run natively in Android (*Android Runtime*) as ELF files because Android uses a tailored version of the Linux Kernel. We found a limited number of files for Android IoT and TV, specifically, 113 (AV labels 11 as malicious) and 57 (AV labels 6 as malicious) files, respectively.

We rely on AV detection as a way to identify possible malware, similar to prior works [177]. First, we collect files with one AV detection to stay under the daily access quota (10K/day, see Figure Figure 5.1). Second, we filter files with less than five AV detections to suppress false-positives, which are common in VT [177]. These criteria filter out possible irrelevant samples that are not likely to be IoT malware with minimal impact on the empirical results. However, we do acknowledge this might lead to a bias in the malware dataset since our collection relies on AV detections that can have inherent limitations.

Active and Passive DNS. Our active DNS (aDNS) dataset comes from the ActiveDNSProject [178], which actively resolves many popular zones (COM, NAME, NET, ORG, BIZ, etc.), top sites from the Alexa Top 1M, and public blocklists daily. The passive DNS (pDNS) is an anonymized dataset provided by a large internet service provider (ISP) based in the US. The ISP operates a large set of geographically-distributed local DNS resolvers

[1] A recorded state of a program during a crash

[2] An object file that linkers use to build an executable.

[3] MD5: 3c5a75bd1df81c6f355b3edf61729507, Label: BitCoinMiner

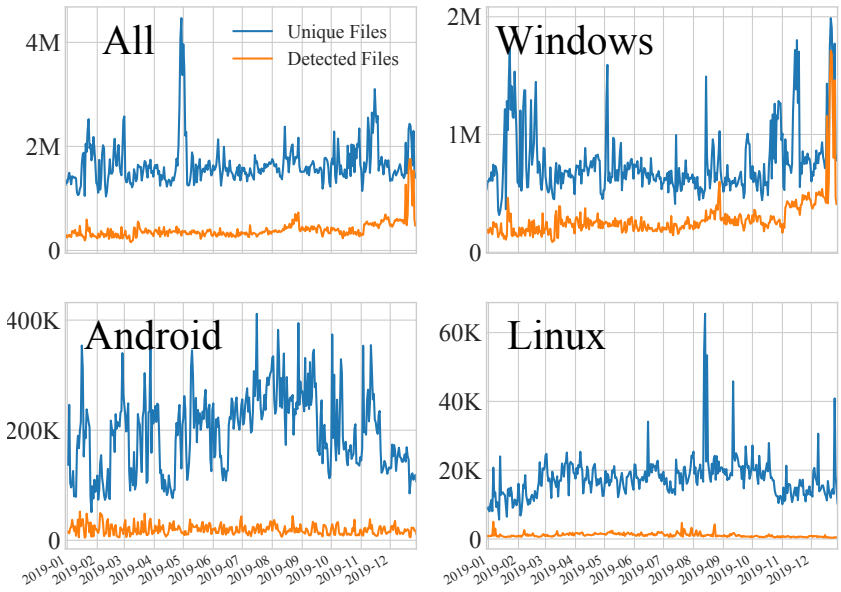


Figure 5.1: The daily volume of files and detected files submitted to VirusTotal in 2019 per platform.

Table 5.2: The data sources used for the empirical study.

Data Provider	**Data Type**	**Role**
VirusTotal	Binaries Metadata Detection & Labels	Binary Analysis Growth & Size
ActiveDNS Project Large ISP	Active DNS Passive DNS	Internet Measurement
Bad Packets	Honeypot	Device Targeting
Tranco	Top Site Ranking	Filtering

that service over 40 million internet-connected devices, which include IoT devices. We use aDNS and pDNS to investigate IoT malware infrastructure. Our aDNS and pDNS datasets cover the period from May 2019 up to Jan 2020. We specifically use aDNS and pDNS to enumerate relationships between observed IPs and domains. We use pDNS data to quantify the lookup volume and the number of anonymized clients resolving the C&C infrastructure.

Bad Packets Honeypots. Bad Packets [179] operates a set of proprietary honeypots that monitor emerging cyber threats targeting enterprise networks, IoT devices, and cloud computing environments. We were provided an aggregate dataset that spans the entire month of June 2019. We use the honeypot dataset to identify attack characteristics observed on the internet and quantify what devices IoT malware target. Specifically, we use aggregate

statistics about internet scans that are classified as IoT malware by Bad Packets.

Tranco Top Site Ranking. We use Tranco's top site ranking [180] to identify and filter benign domains. Our static and dynamic analysis yield large sets of domains and IPs, which may not be related to malware. For example, a link to the UPX packer website is commonly found in samples that are packed by UPX.

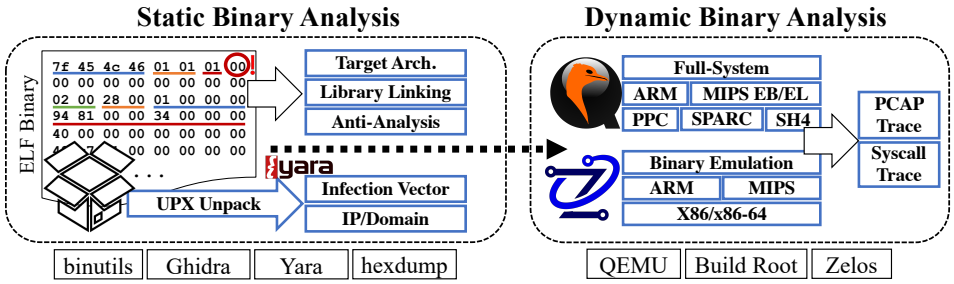


Figure 5.2: An overview of the static and dynamic analysis pipeline.

5.2.2 Analysis Methods

Figure Figure 5.2 presents an overview of our analysis and measurement methodology. We use static, dynamic, and network analysis. We do not claim any of the techniques as a novel contribution, instead, we use them as a means to study IoT malware. We rely on well-established approaches from prior works [142, 181, 182, 183] and tailor them for our analysis.

Metadata Analysis. We use VT for AV detection, AV labels, and in-the-wild names. We combine the AV labels with AVClass [184] to consolidate the labels for each sample. This metadata analysis provides context about the malware samples and helps us to correlate the findings from static and dynamic analysis.

Static Analysis. The goal of static analysis is to identify each binary's target architecture, linking method (static vs dynamic), anti-analysis tactics, packing, embedded domains and IP addresses, and infection vectors. We use a set of tools from binutils suite to perform static analysis, namely readelf, objdump, objcopy, strings, and hexdump. The *file* tool parses the binary information and identifies the target architecture, endianness, and

linking information based on the file headers. Next, we examine the ELF binaries for anti-analysis artifacts by using four heuristics. First, we inspect the ELF file for the first *LOAD* (PT_LOAD) segment in the section headers that is marked for read, write, and execute (RWE). This anti-analysis trick is commonly used to hide the program's entry point and break analysis tools.

Second, we examine the ELF file for fake section headers that overlap the program's entry point by iterating through each segment and section. For each segment, we check if the segment overlaps the entrypoint address. If we detect an overlap, we conclude that the sample has anti-analysis artifacts. This well-known tactic overlays fake data and text sections with opposite flags (switching W and X) to confuse analysis tools by parsing the fake data sections for code. Third, we examine the ELF file for fake dynamic symbol tables by checking the section header for one or more dynamic symbol tables (SHT_DYNSYM). We iterate through each segment and look for dynamic symbol tables that come after the dynamic table (DT_SYMTAB) and check if the dynamic symbol table overlaps the dynamic table (virtual address + size is outside the segment). This anti-analysis technique inserts fake dynamic symbol tables for dynamically-linked binaries that mix up the symbols of functions.

Fourth, we iterate over each segment and check the section header fields (*e_shoff*, *e_shentsize*, *e_shnum e_shstrndx* for zero values. This technique removes critical information about the section headers making it impossible to parse. The Linux kernel does not use the section headers when loading and executing the ELF file, therefore removing the section headers breaks some analysis tools that rely on section headers, but does not affect the execution of the binary. Next, we try to detect UPX packed samples by looking for UPX sections and string artifacts. For UPX packed files, we also check if the UPX header is zeroed out, which usually breaks the UPX decompression utility but not the executable. We then attempt to unpack each sample using the UPX utility. Some files fail to unpack due to corrupt UPX headers, but they execute in the dynamic analyzer.

Finally, we use static analysis to extract IP addresses and domains using *strings* with default settings and regular expressions. For captured domains, we use *tldextract*, a python library, to check for properly formed domain names. For IP addresses, we remove all bogons and invalid IP addresses. We also use static analysis to identify infection vectors by using over 200 Yara signatures. We source our Yara signatures by enumerating a set of IoT and router device vendors, crawl the NVD [185], and identify Common Vulnerability and Exposure (CVE) entries that have public proof-of-concept (PoC) code. We then manually build and verify each Yara signature. For each matched Yara signature, we verify that (1) the offset matches the signature inside the binary and (2) the binary offset is referenced by the code section.

Dynamic Analysis. We build architecture-specific virtual machines that execute each sample and collect their system call and network traffic, which we call *full-system* analysis. We run each sample for 60 seconds and collect system call traces using *strace* and network traces. Further, we use a *binary emulator* that emulates the instructions and system calls of an ELF file to generate system call traces, referenced as *Zelos* [186] in Figure Figure 5.2. The run time of a sample influences the observed behavior as documented in prior works [105]. To account for this limitation, we measure trace divergence between full-system and binary emulation. Binary emulation allows us to skip over sleep system calls and fast forward the execution of malware hence revealing possible hidden behavior. Additionally, we use leaked source code from various IoT malware found online [187] and match them with the execution traces and function symbols to identify capabilities.

We empirically found full-system emulation traces to match 85% of binary emulation traces for ARM. The remaining 15% could not be compared due to application binary interface (ABI) mismatch during full-system analysis or failure to run in binary emulation (missing required libraries or incompatible architecture version). Furthermore, we found that before 30 seconds of full-system emulation about 95% of malware will engage in network system calls that either block or loop infinitely. Hence, we chose 60 seconds

to balance between analysis quality and performance. We count successfully executed samples by two metrics, namely system artifacts and network artifacts. For system artifacts we consider a malware to be active if it creates three or more processes in the VM or if it invokes 100 or more system calls.

These parameters were conservatively chosen by examining diverging traces from full-system and binary emulation. For network artifacts, we collected network traffic from the VM for 72 hours without executing any malware. We then filter out any traffic that matches the baseline or bogon networks. We note that this is a modest attempt to build a dynamic malware analysis system for six different architectures and we recognize the challenges that are documented by earlier works [181, 182, 188]. Nevertheless, we report the results in Table Table 5.3 and make our analysis tools public for the community. Dynamic analysis allows us to study infection attempts, persistence methods, exercised capabilities, and C&C communication. We use these findings to empirically document them in the lifecycle framework and compare them to desktop and mobile malware.

Infrastructure Analysis. We use a three-tiered process to filter and identify C&C indicators. First, we use Tranco [180] top sites to enumerate a list of benign domains. We count the most referenced domains and filter them using the top site list. Second, we manually inspect the new list to remove the remaining benign domains. Third, we build a bipartite graph between domains and IPs to find connected components and filter out additional benign clusters [183]. After removing all the benign indicators, we use historical pDNS and aDNS to expand on the malicious indicators to find common infrastructure. For IP addresses, we look into pDNS and aDNS to identify associated domains. We repeat our method on the newly identified domains and IPs until we remove all benign nodes. We verify each node manually.

Table 5.3: A statistical summary of the dataset, metadata, static, and dynamic analysis grouped by IoT malware's target architecture.

Arch.	Dataset Size	VT Metadata		Static Analysis					Dynamic Analysis		
		Detection (5+)	Honeypot Coverage	Library Linking Static	Library Linking Dynamic	Anti-Analysis	Polymorphic Packed	Polymorphic Unpacked	Network System	Network DNS	Network Outbound
ARM	81,152	57,484	25,406	50,117	4,797	2,570	11,464	9,124	36,660	2,939	42,765
MIPS	19,574	17,675	7,769	17,258	94	323	2,812	2,566	14,536	1,271	13,070
MIPS-EL	15,906	14,757	6,052	14,372	71	314	2,517	2,351	13,481	1,178	12,077
PPC	15,648	14,909	6,393	14,604	74	231	4,232	2,468	13,536	756	12,580
SPARC	11,650	11,218	5,197	10,904	31	283	7	0	10,344	729	9,181
SH4	11,587	11,303	6,667	11,038	67	198	6	0	9,619	414	10,772
M68K	11,255	10,983	6,578	9,420	1,342	221	7	0	- -	- -	- -
Total	166,772	138,329	64,062	127,713	6,476	4,140	21,045	16,509	98,176	7,287	100,445

Table 5.4: Top anti-virus (AV) labels based on reports from VirusTotal.

ARM		MIPS		PPC		SPARC		SH4	
Label	Count	Label	Count	Label	Count	Label	Count	Label	Count
mirai	37,505 (65.244%)	mirai	22,602 (66.61%)	mirai	11,350 (76.12%)	mirai	8,305 (74.03%)	mirai	8,030 (71.04%)
gafgyt	15,468 (26.91%)	gafgyt	8,290 (25.56%)	gafgyt	3,336 (22.36%)	gafgyt	2,810 (25.04%)	gafgyt	3,101 (27.44%)
NOLABEL	1,117 (1.9%)	hajime	1,181 (3.64%)	tsunami	149 (1.00%)	tsunami	62 (0.55%)	tsunami	114 (1.01%)
dofloo	893 (1.55%)	NOLABEL	729 (2.24%)	NOLABEL	62 (0.42%)	NOLABEL	33 (0.29%)	NOLABEL	51 (0.45%)
dvmap	716 (1.25%)	tsunami	418 (1.29%)	mirai-dl	2	wanuk	1	bricker	3
tsunami	544 (0.95%)	dofloo	91 (0.28%)	sshdkit	1	telnetd	1	mirai-dl	2
hajime	531 (0.92%)	ddostf	50 (0.15%)	linksys	1	sshdkit	1	aidra	1
ddostf	264	dnsamp	14	hydra	1	solaris	1	- -	
lotoor	260	aircrack	7	hive	1	snamp	1	- -	
dnsamp	28	bricker	5	healerbot	1	silex	1	- -	

5.3 Results

Using the proposed lifecycle framework, this section presents the results from our empirical measurements and observations. We summarize the results for each subsection by **takeaways (TA)**.

Measurement Setup. We filter our dataset from 166,772 to 138,329 samples that are detected by five or more AV engines. We then analyze each sample statically and dynamically to group the results by architecture as shown in Figure Figure 5.2. We use binutils, Yara, Ghidra, and hexdump to identify the target architecture, library linking, symbols, packing, and anti-analysis artifacts. For packed samples, we attempt to unpack them using UPX [189]. For dynamic analysis, we use Buildroot [190] and QEMU [191] for full-system analysis and Zelos [186] for binary emulation. We build our full-system virtual machines (VM) by using the results from static analysis to identify a common set of required libraries to include in the VMs. However, we were not able to build a VM for M68K architecture

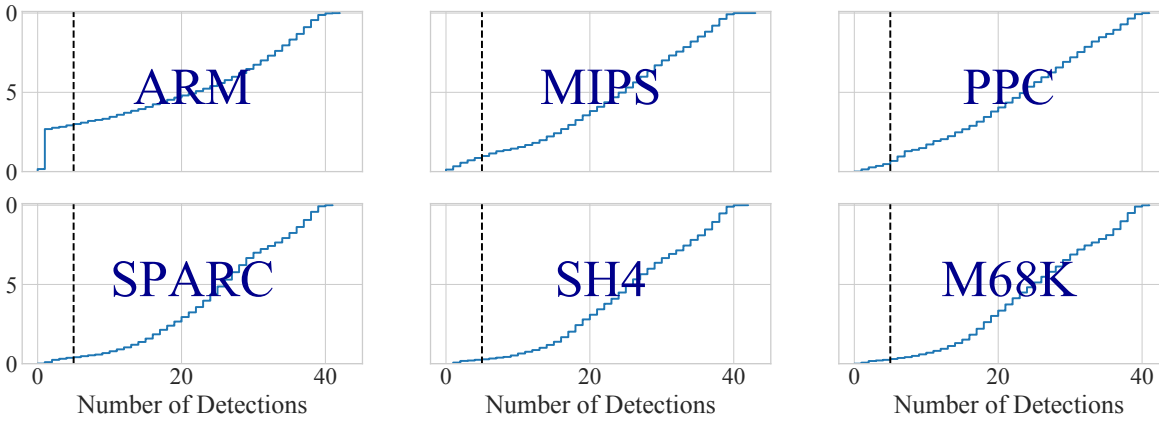


Figure 5.3: The number of AV engines that detect IoT malware per architecture. The dotted vertical line marks five AVs.

due to legacy code incompatibility, therefore, we only considered the M68K samples for static analysis.

Table Table 5.3 summarizes our analysis results by architecture. The VT metadata has two main columns, namely detection and honeypot. Detection refers to the number of samples that are detected by five or more AV engines and honeypot refers to the number of samples seen by the VT honeypot. The static analysis section has three columns, namely library linking, anti-analysis, and polymorphic. The library linking column presents the number of static and dynamic linked samples, the anti-analysis column presents the number of samples that break static analysis tools, and the polymorphic column presents the number of packed samples and how many were unpacked. Lastly, the dynamic section has two columns, namely system and network. The system column reports the number of samples that create three or more processes or invoke at least 100 system calls. For the network, we report the number of samples with DNS and outbound internet traffic.

5.3.1 Detection and Labeling

In Table Table 5.4, we present the top 10 AV labels grouped by system architectures. We use AV engines hosted by VT, which are reported to have better detection coverage than their desktop versions [177]. However, Figure Figure 5.3 suggests that traditional AV engines

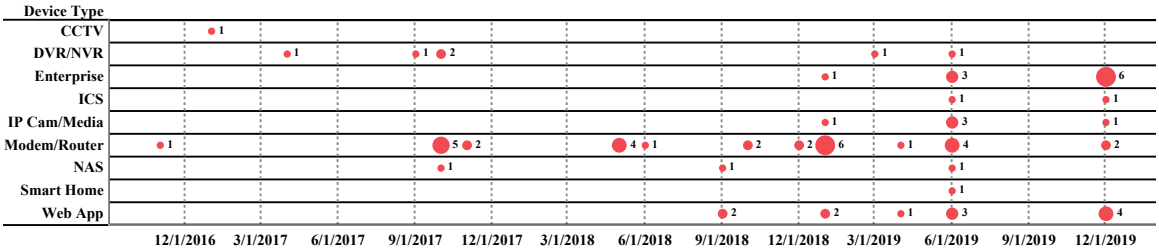


Figure 5.4: A timeline of exploits for Mirai variants based on reports from security researchers.

lack support and detection for IoT malware. VT hosts over 70 AV engines, but only 55 support ELF files. We observe 50% of the malware is detected by less than 25 AV engines and at most by 45 AV engines as shown in Figure Figure 5.3. Furthermore, AV engines appear to detect ARM malware with better coverage, over 25% of the ARM samples are detected by at least 2 AV engines. AV engines provide AV label coverage for at least 97% of the detected malware.

We observe that the *mirai* label dominates in all system architectures and accounts for 76% of the PPC samples. The next most popular label is *gafgyt*. The ARM samples have more diverse labels in comparison with the others. For example, the label *lotoor* and *dvmap* are only found in the ARM dataset. Some labels are exclusive to a set of architectures like *hajime*. Herwig et al. [135] report that Hajime malware is only built for ARM, MIPS, and MIPS-EL, which is aligned with our findings. The inconsistencies in AV detection and labeling are also reported in prior studies [192, 193].

TA1. Given that no host-based intrusion detection systems (HIDS) run on IoT devices, detecting malware after an infection is not possible. However, signature-based scanners can detect suspicious binaries forensically captured from the network or the device. Our findings suggest that many AV scanners lack support or have limited signature coverage (mostly mirai labels) for IoT malware in the wild.

Table 5.5: Device categories and their top vulnerabilities that are targeted by IoT malware based on data from *Bad Packets*.

Category Type	Scans	Top Vuln. in Category	Scans (%)
CCTV	221,340	GoAhead login.cgi	221,340 (100)
Modem/Router	102,690	Linksys	26,239 (25.55)
DVR/NVR	40,998	Kguard DVR	24,069 (58.71)
Enterprise	18,277	Yealink VOIP	11,958 (65.43)
Smart Home	8,806	Google Chromecast	8,422 (95.64))
Web App	6,133	Apache Struts 2	6,094 (99.36)
IP Cam/Media	1,458	WIFICAM Generic	661 (45.34)
NAS	565	QNAP	565 (100)
ICS	11	Schneider U.Motion	11 (100)

Table 5.6: Top exploits found in IoT malware binaries based on static analysis.

Vendor	CVE	Dev. Type	Vuln. Type	Dev. Arch.	AV Labels	ARM	MIPS	PPC	SPARC	SH4	M68K
Huawei	CVE-2017-17215	Router	CMD Inject	MIPS	gafgyt, ircbot, mirai, tsunami	10,046	5,527	2,604	2,352	2,277	2,226
ZTE	- -	Router	Default Cred	MIPS	dlink, exploitscan, gafgyt, mirai, tsunami	3,190	2,038	912	728	735	724
D-Link	CVE-2014-8361	Router	CMD Inject	MIPS	gafgyt, mirai, tsunami	2,378	1,436	656	534	530	534
GPON	CVE-2018-10562	Router	CMD Inject	Unknown	gafgyt, mirai, tsunami	2,016	1,245	539	448	443	435
Zyxel	CVE-2016-10372	Modem	CMD Inject	MIPS	gafgyt, mirai, tsunami	531	356	129	117	132	132
Juniper	CVE-2015-7756	Firewall	Backdoor	ARM	gafgyt, mirai	413	256	115	95	77	82
Multi-Vendor	- -	DVR	CMD Inject	ARM	gafgyt, mirai, tsunami	326	229	74	56	68	70
D-Link	CVE-2013-7471	Router	CMD Inject	MIPS	gafgyt, mirai, tsunami	317	205	79	62	71	71
Synology	CVE-2017-9554	NAS	Info Leak	Various	gafgyt, intercepter, minerd mirai, stealthworker, xmrig	289	145	49	31	34	31
Zyxel	CVE-2017-18368	Router	CMD Inject	MIPS	gafgyt, mirai	191	105	48	41	43	38
Asus	CVE-2018-15887	Modem	CMD Inject	MIPS	gafgyt, mirai	166	92	40	42	53	50
NETGEAR	- -	NAS	CMD Inject	ARM	mirai	112	87	25	21	26	24
HooToo	CVE-2018-20841	Router	CMD Inject	MIPS	gafgyt, mirai, tsunami	112	60	28	17	22	22
WePresent	- -	Router	CMD Inject	MIPS	mirai	98	58	24	21	25	23
LG	CVE-2018-17173	Display	CMD Inject	ARM	mirai	98	58	24	21	25	23
Vera	CVE-2013-4861	Hub	Info Leak	MIPS	mirai	92	52	21	18	21	20
Belkin	- -	Smart Home	CMD Inject	MIPS	mirai	88	50	20	17	20	19
Multi-Vendor	- -	Camera	CMD Inject	MIPS	mirai	85	48	20	17	20	19
Multi-Vendor	CVE-2017-8225	Camera	Info Leak	MIPS	mirai	85	48	20	17	20	19
DreamBox	CVE-2017-14135	Media	CMD Inject	PowerPC	mirai	85	48	20	17	20	19
Multi-Vendor	CVE-2019-3929	Router	CMD Inject	MIPS	mirai	85	48	20	17	20	19
Oracle	CVE-2019-2725	Web App	CMD Inject	x86_64	mirai	85	48	20	17	20	19
Schneider-Electric	CVE-2018-7841	Industrial/Home	CMD Inject	x86	mirai	85	48	20	17	20	19
Linksys	- -	Router	Mem Corrupt	MIPS	mirai	83	50	20	19	21	20
EnGenius	- -	Router	CMD Inject	MIPS	mirai	68	64	13	12	14	13

5.3.2 Infection Analysis

We observe that IoT malware use remote exploitation and default credentials to infect devices. We present a timeline in Figure Figure 5.4 that shows the incorporation of exploits in IoT malware based on reports from researchers. The timeline begins right after the Mirai source code became public and extends to the end of the malware collection period (Dec. 2019). We find nine categories of devices across 70 different exploits [194, 203, 204, 205, 206, 207, 208, 209, 210, 195, 196, 197, 198, 199, 200, 201, 202]. We observe that the number of exploits increases significantly in 2019, which target new categories of devices not seen before such as enterprise network equipment, industrial control systems (ICS), network attached storage (NAS), and smart home devices.

Moreover, in Table Table 5.5 we present results from the Bad Packets LLC [179] hon-

eypot. The table shows a list of device categories targeted by IoT malware in June 2019 ranked by the number of observed scans. We present the top vulnerability in each category to the right and quantify the composition of the scans per category. For example, the *Kguard DVR* vulnerability makes up 58.71% of the scans in the DVR/NVR category. We present our empirical findings in Table Table 5.6. The table shows the vendor of the target device, CVE number, device type, vulnerability type, device architecture, malware labels, and the number of samples containing the exploit.

First, we observe that the exploits affect internet-facing devices and devices behind the NAT. For example, routers and firewalls are typically internet-facing while smart home devices such as hubs should be behind a NAT device. Second, we observe that most of the vulnerability types affect network services by command injection, credential leak, or default credentials. Third, the affected device architectures are mostly ARM and MIPS, nevertheless, IoT malware appears to be architecture agnostic. Finally, we observe that certain malware families, such as *minerd*, *xmrig*, *intercepter*, and *stealthworker* target specific devices like the Synology NAS, which suggests that some IoT malware specializes in device targeting.

TA2. Early IoT malware (see subsection 2.3.2) relied on default credentials or a specific vulnerability to compromise internet-facing IoT devices. Our findings suggest that IoT malware has evolved to rely on a suite of exploits that target *many* diverse device categories not seen before, which can be either internet-facing or behind a NAT device.

TA3. Given most IoT devices are headless, lack a graphical user interface (GUI) or peripheral devices, all observed exploits do not require user interaction. This IoT device property allows malware to efficiently infect many devices very quickly. Additionally, the architecture agnostic nature of IoT malware may potentially make them more of a threat than earlier desktop worms.

5.3.3 Payload Analysis

We observe that IoT malware payloads use packing, environment keying, scripting, and cross-architecture binaries. Table Table 5.3 shows that **at least** 15% of the malware use packing, and we were able to unpack 78% of the packed samples. The remaining samples used anti-analysis tricks that broke the standard unpacker. We observe in dynamic analysis that IoT malware payloads use environment keying before executing. For example, we see payloads profiling the device name, CPU, and memory to check for the right environment.

We found a set of payloads that rely on script interpreters like Python and Lua for functionality. However, most payloads use the system shell for system reconnaissance and persistence. For example, various binaries invoke shell commands like *uname*, *whoami*, *lsof*, *crontab*, and *os-release* to collect information about the device. We observe on exploitation that multi-architecture payloads are delivered to the device to brute force the target system architecture. For example, if the malware cannot identify the device's architecture, they test many variants of the payload for different architectures such as ARM, MIPS, PowerPC, SPARC, SH4, and M68K.

TA4. Our analysis suggests IoT malware uses polymorphism to evade signature-based detection. We estimate at least 15% of the samples use packing and 3.3% use a more advanced anti-analysis method to thwart unpacking. Also, the analysis suggests that the device's system shell interface is a primary component for payload selection and infection.

5.3.4 Persistence Analysis

Before presenting the results, it is important to understand how embedded devices configure their file systems. First, most embedded devices mount their rootfs (file system) as read-only (RO). This reduces wear on flash memory, eliminates system file corruption, avoids accidental overwrites, facilitates device update over-the-air (OTA), and eases factory reset. Still, there are processes on the device that need write-access for passwords, configurations, and keys. Embedded devices designate a non-volatile data region and a volatile temporary

file system region on the flash memory. The data region is used by processes and services to store their configurations. Malware have to consider these file system constraints to persist on the device.

We observe in dynamic analysis that IoT malware attempt to persist on the device's firmware. *We must clarify that firmware refers to the IoT device's OS, which is a customized embedded Linux instance (unified layer, see Table Table 2.2).* In many IoT devices, services run as root, which means if exploited by malware then they will gain root access on the device. We observe that IoT malware use many persistent methods by installing themselves as either a service, a startup script, a system module, or a backdoor. Some samples attempt to remount the file system with read-write permissions to persist on the rootfs. For example, using the command *mount -o remount*, malware can remount the file system with read-write permissions. In several instances, we observe malware using vendor-specific tools such as */bin/cfgmtd* that target Ubiquiti devices to add an SSH backdoor.

Even with volatile memory regions, we observe IoT malware using *tmpfs* paths to persist. On system reboot, the *tmpfs* paths will be wiped, which will remove the IoT malware. However, to prolong the infection, we notice that IoT malware will disable the watchdog process on devices. A watchdog process on an embedded device is a privileged process that mitigates software faults by forcing a device to reboot into a clean state. If malware causes the system to become unstable, the watchdog process will reboot the device and consequently remove the malware. For example, IoT malware will disable the watchdog process by writing the "Magic Close" value ("V") to one of the following locations */dev/FTWDT101_watchdog*, */dev/misc/watchdog*, or */dev/watchdog*.

TA5. The results suggest forensic identification of infections on a device may be difficult because malware can persist in many locations. Although IoT devices mount their file system as read-only, there appears to be many methods to overcome this limitation, which can worsen infection cleanup.

Table 5.7: Scanning methods found in IoT malware binaries based on dynamic analysis.

Protocol	Port Number	Attack Type
Telnet	23, 2323	Dictionary Attack
ADB	5555	Android Debug Bridge Shell
HTTP	5555, 55555, 52869, 37215, 7547, 8080, 8081, 443, 80, 81	Command Injection

5.3.5 Capability Analysis

Initial variants of IoT malware discussed in subsection 2.3.2 focused on DDoS and scanning capabilities. Our analysis shows an expanded set of capabilities found in modern IoT malware. Using dynamic analysis, we observe aggressive evasion by disabling firewall processes, access control modules, ISP remote administration, unblocking restricted domains, deleting access logs, history logs, and service access logs, and modifying timestamps on files. Moreover, we observe privilege escalation attempts targeting the Android Runtime environment. We also observe data theft attempts that look for Sybase database files, collect device profiles, harvest device configurations, and enumerate system files. Perhaps the most prevalent capabilities are network scanning and spreading. Table Table 5.7 is a summary of the observed scanning and exploitation attempts, which includes a subset of the vulnerabilities found in Table Table 5.6. We do not observe direct DDoS attacks, but through static analysis, we find DDoS capabilities in the malware. We identify a set of DDoS attack functions using function symbols in the analyzed samples and match them with public malware source code. Table Table 5.8 presents a list of the DDoS functions found in IoT malware.

Additionally, we observe from dynamic analysis device destruction attempts by IoT malware. Malware will try to delete the root directory of the file system, dbus devices, zero out MMC memory, remove configured devices on general-purpose IO pins, and delete the Linux device table. Furthermore, we observe IoT malware will abuse device resources for cryptocurrency mining and proxy services. Malware will download open-source miners such as *cgminer* and attempt to lock out the device owner by removing restore tools, disabling device upgrade, and hard-coding an IP address to a specific mining pool server. We

Table 5.8: DDoS capabilities found in IoT malware binaries based on static analysis and leaked source code.

DDoS Type	Function Symbol Name
TCP	attack_tcp_syn, attack_tcp_ack, attack_tcp_stomp, attack_method_tcp, attack_tcp_ysynack, attack_tcp_nfo, attack_method_tcpfrag, attack_method_tcpall, attack_method_tcpusyn, attack_method_asyn, attack_tcp_lynx, attack_method_tcpxma
UDP	attack_udp_generic, attack_udp_vse, attack_udp_dns, attack_udp_plain, attack_method_udpgame
GRE	attack_gre_ip, attack_gre_eth
APP	attack_app_http, attack_method_ovh attack_method_miscdestruct, attack_app_cfnull
GENERIC	attack_method_std, attack_method_generic, attack_method_misckill

also observe attempts to set up a proxy service that configures network traffic forwarding on high ports.

TA6. Infected devices can degrade or damage IoT services not only for device owners but also for network operators and device vendors. Additionally, they can facilitate criminal activities by tunneling malicious traffic through infected devices or eavesdropping on local network traffic.

5.3.6 C&C Analysis

We observe from the dynamic and static analysis that IoT malware can use P2P and centralized infrastructure for C&C communication. For example, Hajime [135] uses the *Kademlia* overlay network, which is a P2P protocol. We also observe some malware using the *Tor* network either for C&C call-back or for connecting to a cryptocurrency mining pool. For centralized infrastructure, we find that IoT malware rely on hard-coded IPs rather than domains, as shown in Table Table 5.3. We only observe 7K samples with DNS lookups, which accounts for less than 7% of the network active samples. From network traces, we gather 306 unique domains and 10,895 IPs, which have a very small overlap. This reinforces that IoT malware rely mostly on hard-coded IP addresses for C&C call-back. Lastly, we

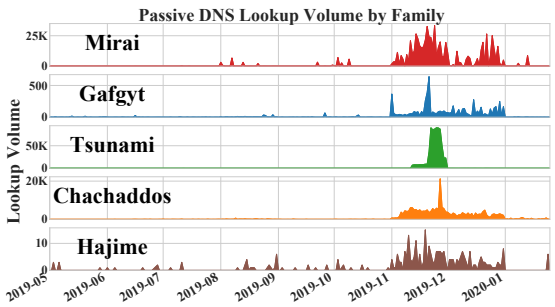
observe that some IoT malware attempt to hide their DNS IP address resolution by using DNS *TXT* records.

We investigate the domains and IP addresses using the pDNS dataset. Table Table 5.9 presents the top six malware families based on the infrastructure analysis described in section 2.2. We rank the rows by the number of unique client IDs found in the pDNS dataset. The columns are as follows, *Labels* is the AV family, *Clients* is the number of unique client IDs, *FQDN* is the number of unique fully-qualified C&C domains, *IP* is the number of unique C&C IPs, *e2LD* is the number of effective second-level C&C domains, *Days* is the number of distinct days the C&C was queried, *Samples* is the number of malware, and *Cluster* is the number of C&C clusters per family. We observe that the *mirai* samples are the most active with 874 clients, 144 e2LD, 151 unique clusters, and 2,607 associated samples. The next largest is *gafgyt*, which shares 63 clusters with *mirai*. Also, Figure Figure 5.5a and Figure Figure 5.5b present the malware activity as seen from pDNS. We observe that the lookup volumes are sporadic throughout the year, then for the period from November to January, there is an uptick in lookup volume especially for the *tsunami* family.

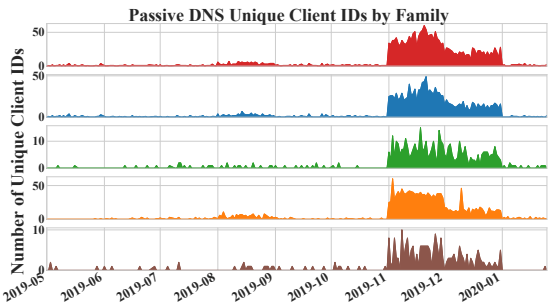
Table 5.9: Top IoT malware clusters grouped by AV Labels.

Labels	Clients	FQDN	IP	e2LD	Days	Samples	Cluster
mirai	874	229	369	144	269	2607	151
gafgyt	687	121	146	69	269	2727	73
chachaddos	300	2	7	2	253	2	1
hajime	156	4	3	3	265	2	3
NOLABEL	132	44	158	24	269	41	29
tsunami	112	41	48	18	268	263	34

TA7. Network detection of malware communication can prove to be difficult with P2P channels and evasive DNS resolutions. However, the use of hard-coded IP addresses makes IoT botnets less resilient to takedowns. IoT malware network activities can be difficult to measure on the internet using DNS since very few samples rely on DNS.



(a) The number of queries per IoT malware family cluster.



(b) The number of clients IDs per IoT malware family cluster.

Figure 5.5: DNS measurement of domains for the top IoT malware family clusters based on the pDNS dataset.

5.4 Case Studies

Motivated by our empirical results in section 5.3, we take a closer look at how IoT malware reuses Mirai's code to provide more context on attacks.

5.4.1 Code Reuse and Evolution

Bugs in the Source. During our dynamic analysis, we noticed a number of IoT malware samples failed to run in the full-system emulation. Further investigation showed that the samples would crash at the beginning of execution. These samples had their function symbols stripped and only affected the MIPS-EL and ARM architecture. We tracked down the issue to a set of faulty compilers that are used in the build script of the leaked Mirai code. These compilers were specifically for ARMv6 and MIPS-EL architecture. To reproduce the bug, we compiled a test program with the faulty compilers and ran them, but they did not crash. However, when we passed the "strip" flag to the compiler, the binaries crashed on execution. This bug was found in over 8,000 ARM samples from our dataset. Moreover, we reproduced this bug on real hardware by running the test program on two physical devices, namely a Raspberry Pi 3 (ARM) and a GLiNet 300M (MIPS) router. The physical hardware exhibited the same behavior as our full-system emulation.

Investigating additional malware samples that failed in the dynamic analyzer, we found

a set of traces that crash in the middle of execution. We analyzed the crash files and found that a segmentation fault is generated when the malware attempts to hide its process name. A snippet of the code is shown at the top of Figure Figure 5.6 based on Mirai's code. However, other samples did not have this bug, which used a different version of the code shown at bottom of Figure Figure 5.6. The bug is caused by a fixed length buffer used to store the process name, which only supports a maximum of 20 bytes including the path of the binary. The newer code fixes this issue by using a variable-length buffer as shown in the lower portion on line three of Figure Figure 5.6.

TA8. Mirai's original code has distinct bugs that transcend into newer variants, but some samples fix them. Although this evolution overall improves the stability of IoT malware, many samples use Mirai's code as a template, which can make them easier to detect by signature-based techniques.

Corrupted DNS Resolutions. We found a large number of malformed DNS packets from our dynamic analysis, which we initially assumed to be a misconfiguration in our analyzer. We came across a set of samples that attempt to resolve a domain but created malformed DNS packets. These samples had very similar system traces to the original Mirai code. We investigated Mirai's code and found an initialization bug that causes DNS queries to be malformed. Specifically, the code does not initialize the buffer where the DNS query is stored, which can contain random bytes from the device's memory as padding. We found this bug to affect all Mirai variants [187] in our study, and it appears to contribute to IoT malware reliance on IPs instead of DNS for C&C call-back.

TA9. Since DNS resolution is unreliable for samples seen in the wild, this may explain the use of hard-coded IP addresses for C&C call-back. Furthermore, given the evolutionary trends observed in other components of Mirai's code, a fix for the DNS resolution function can make new variants more resilient to detection, blocking, and mitigation.

```
// Hide argv0 - Fixed Length Name (Bug)
name_buf_len = ((rand_next() % 4) + 3) * 4;
rand_alphastr(name_buf, name_buf_len);
name_buf[name_buf_len] = 0;
util_strcpy(args[0], name_buf);

// Hide argv0 - Variable Length Name
name_buf_len = (rand_next() % (20 - util_strlen(args[0])))
    + util_strlen(args[0]);
rand_alphastr(name_buf, name_buf_len);
name_buf[name_buf_len] = 0;
util_strcpy(args[0], name_buf);
```

Figure 5.6: Mirai's faulty evasion code (top) and the fixed code found in newer variants (bottom).

5.4.2 Payload Hosting

Having identified the DNS bug in the Mirai code, we wanted to understand how some samples used domains. We study the lifecycle of two different IoT malware C&C infrastructure, specifically, we pick *iwantallthesmoke.club* and *str3sser.com* from the top clusters identified from Section subsection 5.3.6. We manually investigate these domains using DomainTools and VT.

Str3sser Domain. The *str3sser.com* domain was registered by Namecheap on *2018-06-29* and was inactive for almost six months. On *2018-12-27*, the domain records changed to point to Cloudflare. There were two *A* (*104.27.181.96* and *104.27.181.96*) and two *NS* records (*liz.ns.cloudflare.com* and *jobs.ns.cloudflare.com*) created. We speculate that these records were for initial testing before going live because of the low DNS lookup volume (average 16 lookups). After 79 days, the domain's *A* (*35.241.225.135* and *35.205.247.152*) and *NS* records (*dns1.registrar-servers.com* and *dns2.registrar-servers.com*) change to point to Google cloud.

Approximately 50 minutes later, based on pDNS first seen resolution, the domain is detected and reported to *URLHaus*. The domain remained active based on a screenshot captured nine days later but after 14 days the *A* records changed to a residential IP address (*72.5.65.111*). Finally, after two days, the owner created five child labels

(*cuteguyss*, *est1976*, *apneager*, *chivethethrottle*, and *aq*) pointing to OpenDNS infrastructure (*146.112.61.107*) before the domain went offline. We base the shutdown evidence on the abrupt change in pDNS lookups from hundreds a day (average 350 lookups) to zero. The domain remained dormant with no lookups seen by pDNS sensors until it expired. The domain was used for hosting the IoT malware payload, which is downloaded after exploitation. The malware sample associated with this domain checks-in with the C&C server using the hard-coded IP address *35.242.254.121* on port TCP/443 (not TLS). In this case, the payload domain operated for approximately 16 days.

IWantAllTheSmoke domain. The *iwantallthesmoke.club* domain was registered by Namecheap on *2019-01-10*. A day later, one *A* record (*185.141.24.211*) is added to point to a virtual private server (VPS) (Host Sailor Ltd.). Two days later, a screenshot of the domain's front page reads "me nah wan go jail." On day three, 11 lookups are seen by pDNS and the domain goes dormant with no activity for five days. Then, on *2019-01-21* the domain updated the *A* record (*89.46.223.195*) to point to another VPS (Zare.com). Approximately 50 minutes later, the domain is reported to *URLHaus*. The domain's DNS lookups increased to an average of 10 lookups per day, but three days later the lookups stopped. On the seventh day, the domain was no longer available and only operated for six days before going offline.

However, this domain is one of five domains associated with the payload hosting server. Using pDNS data, we observed four additional domains that were used throughout the year (Jan'19 to Jul'19) pointing to IP address *89.46.223.195* and hosting similar payloads, suggesting a rotation of payload domains. The malware sample checks-in with the C&C server using the same IP address on port TCP/9285, but instead of resolving any of the five domains the sample uses the hard-coded IP address. The domains are only used in the initial exploitation followed by payload download. These observations suggest that malware using domains for payload download rely on the device's DNS resolution instead of Mirai's code. Recall, many of the exploits in Section subsection 5.3.2 rely on the device's system shell to download and run the payload, hence the DNS resolution is done by the

device, not the malware code.

TA10. IoT malware uses domains for payload hosting and rarely for C&C call-back. Although payload hosting domains are short-lived (i.e. six and 15 days), their lifespan is sufficient for IoT malware operation because the malware can efficiently infect many devices. This suggests that domain takedowns only affect malware spreading but not the botnet itself.

5.5 Discussion

This section will compare IoT malware and traditional malware to understand common trends. We then present a qualitative assessment of current defensive techniques against IoT malware to understand available security measures that can address IoT malware attacks.

5.5.1 Similarities and Differences

First, we observe that the majority of IoT malware is based on Mirai's code. This is vastly different from traditional desktop and mobile malware, where there are hundreds if not thousands of desktop and mobile malware families. This observation suggests that offline IoT malware detection (TA1) may be relatively easier than traditional malware because a large majority of samples in the wild stem from a shared code base. However, similar to traditional malware, polymorphism and anti-analysis (TA1) found in IoT malware can be effective in evading signature-based detection. Although we only observe 3.3% of the samples to use anti-analysis methods, we can only claim a lower bound.

The infection analysis (TA2 and TA3) suggests that IoT malware can be a bigger threat than traditional malware. For example, desktop malware has more categories of infection (drive-by, phishing, etc.), however, remote exploitation and default credentials for IoT malware apply to a larger set of architecture-agnostic internet-facing devices. Furthermore, we predict as IoT devices advance, repackaging, drive-by, phishing, and removable media will all be practical infection vectors that IoT malware may abuse. The payload analysis

results (TA4) show that IoT malware has already incorporated advanced polymorphic and anti-analysis tactics, which suggests that we may see a wide adoption in the near future similar to desktop and mobile malware. One difference from traditional malware, which can be used against IoT malware, is the reliance on the device's system shell, which can be disabled or limited (i.e. *seccomp*).

Persistent analysis (TA5) shows that IoT malware has to deal with file system constraints not found on desktop or mobile systems. Yet, the unification of user-space, kernel-space, and firmware removes layered protections found in traditional platforms, which can allow IoT malware to have privileged access to the device's hardware. This suggests that although current persistent methods are limited, direct access to a device's hardware can enable stealthier persistence tactics that may require device replacement to remediate. The capability results (TA6) present a spectrum of abuse that can range from infecting devices by scanning and exploitation to more sophisticated such as information theft and network traffic hijacking. The results in Table Table 5.6 show that some IoT malware families target specific devices, which suggests that we may see more tailored IoT device targeting based on the malware's capabilities (rise of specialization). This is analogous to desktop malware that specializes in financial crime, ransomware, and credential theft, for example.

Furthermore, IoT malware C&C communication results (TA7) show a mix of P2P and centralized control infrastructure. Based on the abrupt IoT botnet activity observed on ISP networks, botnet operators may shift to implement a similar layered C&C communication approach to the Storm botnet [89] to achieve scalability, stability, and resilience. However, IoT malware reliance on Mirai's code may have hindered its potential due to inherited bugs (TA8 and TA9). This is further evident by the fact that IoT malware operators use DNS mostly for payload hosting (TA10). It appears based on the infrastructure analysis in subsection 5.4.2, IoT malware operators have adapted to register multiple domains for payload hosting. Since IoT malware uses a very noisy internet-wide scanning and infection approach, the payload domains are quickly detected and blocked. On the other hand, it

seems that short-lived payload domains provide sufficient time for the botnet to spread (TA10).

5.5.2 Stakeholders and Defenses

We identify three primary stakeholders, namely device owners, device vendors, and ISP operators.

Device Owners. Device owners have limited options for detecting and removing IoT malware infections. Device owners, whenever possible, should disable internet-facing services, change default credentials, and segment their network to mitigate some of the risk of infection. Most device owners would reboot their device if it becomes unresponsive or the quality of service degrades, which is also applicable to IoT malware infections. Although most IoT malware may be cleaned up with a simple reboot, we have observed several instances of IoT malware using more persistent methods (TA5). Moreover, re-imaging the device with a trusted firmware may not be possible, is technically difficult, or can damage the device. We believe the impact of this problem is much more serious than reported in prior work [153]. Specifically, we speculate that the current reinfection rates are much higher than what was measured in 2017/2018 (only 5%).

Device Vendors. Device vendors have end-to-end visibility that can provide early detection and remediation of IoT malware infections. For example, device telemetry can help detect system anomalies, device firmware can limit system shell interaction, containerization can limit cross-process interaction, process whitelisting can allow only trusted processes to run, remote attestation via trusted execution can guarantee a clean state, and client-server design can limit the exposed services on the network, therefore reducing the attack surface. These approaches may not all be cost-effective for vendors, but some features can be implemented as default protections for embedded Linux to boost the overall security of Linux-based IoT devices. Moreover, as vendors innovate in the IoT space, they must be mindful of future attack surfaces. For example, future IoT devices may incorporate more human interac-

tions, which can inherit all the attacks from traditional malware such as phishing, drive-by download, and application repackaging. More precisely, incorporating a browser in an IoT device allows IoT malware to reuse attack tactics that are found in traditional malware.

ISP Operators. ISP operators can play an important role in IoT malware infection cleanup as documented by Çetin et al. [153]. Besides using a walled garden for infected customers, ISPs can hinder the infection by deploying IP blocking and redirection for known IoT C&C or payload hosting servers. A more active approach would be for ISPs to intercept payload delivery or C&C communication and instead deliver a therapeutic payload that cleans up and disables vulnerable services transparently without the user involvement. However, this approach requires careful planning and engineering to scale to large networks. Current defenses at the ISP level can disrupt IoT malware infection breakouts, but this requires close monitoring and measurements to detect such events.

5.6 Using Threat Analysis to Inform Risk Assessment

The large-scale study of IoT malware lifecycle yielded several takeaways. These takeaways can inform our risk assessment scoring framework. We provide explicit examples of how the takeaways adjust the risk-score metric. We will not use all the takeaways; the risk score will be weighted based on a multiplicative factor. We will describe the risk scoring methodology in chapter 6.

1. TA1 concludes that host-based detection security measure is not practical. We linearly increase the risk score for an IoT deployment over time to account for exposure.

2. TA2 finds that IoT threats evolve to exploit vulnerabilities on internet-facing and Network Address Translation (NAT) configured devices. We increase the risk score for devices with lax security measures on the Local Area Network (LAN) (unencrypted and unauthenticated services).

3. TA3 infers that IoT threats can spread very quickly without user interaction because

most IoT devices are headless. We increase the risk score for devices without user interfaces and expose network services.

4. TA6 concludes that infected IoT devices can proxy other cybercrime or attacks to degrade critical internet services. We increase the risk score for always-on devices with copious computational resources.

5.6.1 Targeted Devices In The Testbed

Our study identified many targeted IoT devices across different device types (CCTV, ICS, and Smart-Home). We present a list of the top targeted devices in Table 5.5 and Table 5.6. In the smart-home category, we find devices like *Google Chromecast*, *VeraLite Hub*, and *Belkin WeMo (Switch, Link, Motion, Crockpot)*. Our testbed, presented in chapter 3, has deployed IoT devices for both the *Belkin WeMo* and *VeraLite Hub*. We found that the *VeraLite Hub* is an EOL device with many critical and high vulnerabilities that persist with the device. These vulnerabilities increase the risk of deploying a *VeraLite Hub* because we empirically show that IoT malware actively targets the device. IoT malware also targets the *Belkin WeMo* devices. However, the deployed instances of the *Belkin WeMo* devices addressed the targeted vulnerabilities through a firmware update. These two examples show the need for studying IoT malware at scale to identify trends in device targeting and inform the risk assessment approach.

CHAPTER 6

RISK ASSESSMENT FRAMEWORK FOR IOT DEPLOYMENTS

The risk assessment framework aims to help practitioners prioritize security measures for at-risk IoT deployments. A risk assessment framework can quantify risk based on vulnerability and threat analysis of an IoT deployment and provide a ranking. Practitioners can use the ranking to prioritize their limited resources to implement optimal security measures strategically. The framework provides a basis for users to adjust the score calculation based on their environmental needs. In this chapter, we will demonstrate an instance of the framework, which can be subjective. However, the exercise aims to showcase how users can use the framework to generate a risk score. The framework will highlight the importance of conducting a comprehensive (chapter 3) and iterative (chapter 4) security and threat (chapter 5) evaluation for risk assessment and prioritization of security measures.

6.1 An Informed Risk Assessment Model

Our first study in chapter 3 provides an overview of the vulnerabilities found in the components of an IoT deployment. The study provides a holistic approach that accounts for *more* attack vectors against IoT deployment than traditional security evaluations. Using each component's security properties (see subsection 3.0.1), we derive metrics contributing to each component's risk score. There is a base score for each component that users can adjust for vulnerability exposure period and exploitation by threats. The longitudinal observations can adjust device risk score weight based on the exposure window or the time delta between vulnerability identification and patching. For example, we find that EOL devices do not address any of the vulnerabilities on the device, and we can increase the weight of such devices to account for the exposure. We use the IoT malware lifecycle study (see chapter 5) to adjust the weight on devices's risk score that have actively exploited

vulnerabilities.

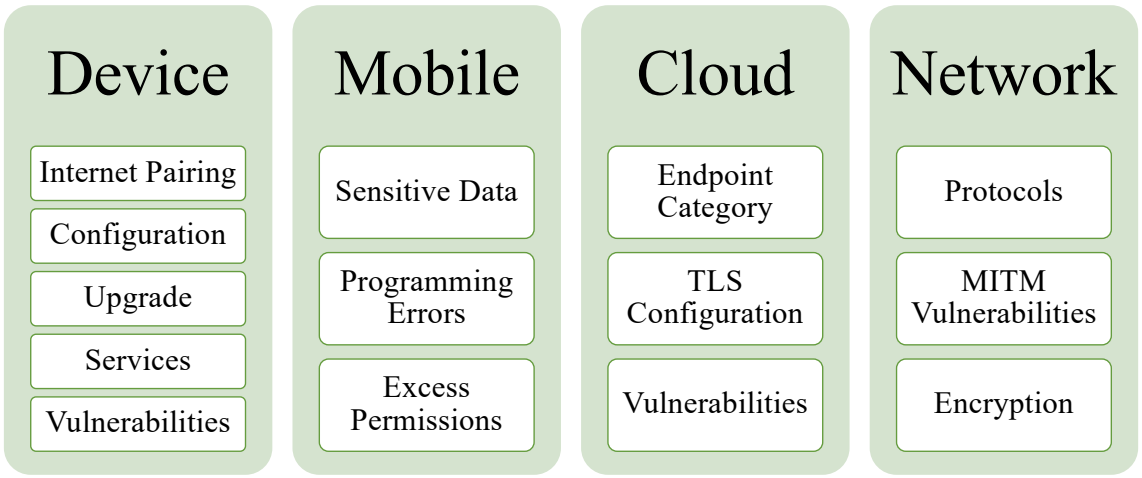


Figure 6.1: The main components of IoT deployment and their security properties.

6.2 Risk Assessment Framework

The following section defines key properties of the risk assessment framework, including the security properties and the threat models. We describe the security properties for each component. We define three threat model types and provide examples for how the threat models can mount an attack against the device.

6.2.1 Overview and Terminology

We summarize the components and their security properties in Figure 6.1. Smart-home devices have four main components:

1. The device - the hardware purchased (Alexa, SmartThings, Sonos, etc.).
2. The mobile application - the companion mobile application that interacts with the device (Android or iOS application.)
3. Cloud endpoints - Internet services with which the device or the mobile application communicates.

4. Network communication - Network traffic between each component (local and Internet traffic).

For each component, we can use the following security properties to compute the risk score:

Device Properties. The device properties are the following:

1. Internet Pairing - The configuration of network credentials to connect the device to the Internet.
2. Configuration - The device configuration during the setup phase, creating an account, and setting up preferences.
3. Upgrade - The device's upgrade options (automatic, consent, or manual).
4. Services - Network services on the device, like UPnP, mDNS, and HTTP server.
5. Vulnerabilities - Vulnerable services on the device.

Mobile Application Properties. The mobile application properties are based on static and dynamic binary analysis to identify three types of security issues:

1. Sensitive Data - Sensitive data includes artifacts like API keys, passwords, and cryptographic keys that are hard-coded into the application.
2. Programming Issues - Implementation errors and incorrect use of libraries include weak initialization vectors in cryptographic functions or guessable seeds to pseudo-random number generators.
3. Excess Permissions - Mobile applications request excess permissions not required or used in the application code.

Cloud Endpoints Properties. We use the device and the mobile application's cloud endpoints for evaluation. There are three security properties:

1. Endpoint Categories - Endpoint categories define three main categories: first-party, third-party, and hybrid. *First-party domains* are endpoints that are owned and managed by the vendor of the product. *Third-party domains* are endpoints that use external services like Google Maps. *Hybrid domains* are endpoints run on cloud infrastructure like Amazon or Azure but managed by the device vendor.

2. TLS Configuration - TLS configuration refers to the proper set up of TLS/SSL, including using a trusted and valid certificate and avoiding legacy versions of TLS/SSL with known vulnerabilities.

3. Vulnerabilities - Vulnerable services on the cloud endpoint includes cleartext authentication, misconfigured services, exploitable services, or unsupported legacy operating systems as the host for the cloud endpoint.

Network Communication Properties. We use the network communication properties observed between the three components; the smart-home device, the mobile application, and the cloud endpoint. Each pair of components can have a network connection, namely device-to-cloud (D2C), mobile application-to-device (M2D), and mobile application-to-cloud (M2C). For each connection, there are three security properties:

1. Protocols - Inspect connections for the use of third-party DNS, HTTP, UPnP, NTPv3, or custom protocols.

2. MITM Vulnerabilities - Identifies whether the communication is vulnerable to MITM attacks.

3. Encryption - Identifies whether the communication uses encryption.

6.2.2 Threat Model and Attacker Types

We define three types of attackers. We categorize them by their accessibility to the device component.

Attacker Types

Type 1: LAN Attacker. The LAN attacker is on a network where the devices are deployed and can carry out direct attacks. This attacker is the most capable since they can access device services on the local network.

Type 2: Internet Attacker. An Internet attacker can access only Internet-facing devices. The attacker does not have direct access to the local network, but they can use software vulnerabilities, known flaws, or default credentials to exploit Internet-facing devices to access the local network. Although we consider a Type 2 attacker less capable than Type 1, a Type 2 attacker can become a Type 1 attacker by pivoting from a compromised Internet-facing device to the LAN.

Type 3: Nearby Attacker. A nearby attacker is physically near the deployed device but does not have direct physical access (malicious neighbor). A Type 3 attacker can perform attacks against the initial device setup or using a low-energy medium, such as Bluetooth, Zigbee, or ZWave. Although Type 3 attacker is less capable than Type 1, they can access the LAN through a compromised device (become Type 1 attacker) via flaws in low-energy protocols.

Attack Examples

We provide examples of how the three types of attackers can target devices on the network.

1. Internet Pairing - A nearby attacker (Type 3) hijacks the configuration setup over insecure Wifi or low-energy (Bluetooth, Zigbee, ZWave) protocols. A device is secure if it requires manual credentials input or a wired connection.

2. Configuration - An attacker (Type 1 or 2) knows a device's default configurations and uses this information to attack the device. A device that requires configuration before operating is more secure.

3. Services - An attacker can directly access unauthenticated services locally (Type 1) or on the Internet (Type 2). A client-mode device that runs no services is more secure.

4. Vulnerabilities - An attacker can use one or many vulnerabilities on the device to expose sensitive information or gain control. The vulnerabilities are in four categories; critical, high, medium, and low CVSS. The critical and high categories mean the vulnerability can lead to a complete or partial device takeover. The medium category means there are misconfiguration issues that could disclose sensitive information, which may lead to a device takeover. The low category means the device has minor issues, such as supporting a weak hashing algorithm in encrypted services.

6.3 Risk Model

This section formalizes the risk model and provides examples of risk assessment. The risk model has two types of scores; a base score and a weighted score. The base score provides a risk score for each component (device, mobile, cloud, and network) in an IoT deployment. We calculate the risk score from the security properties in subsection 3.0.1.

6.3.1 Base Risk Score

The base score is a composite normalized score between zero and one that accounts for each security propriety. A higher base score indicates a higher risk component, and a lower base score indicates a lower-risk component. Formally, we can calculate the base score for

the device using the following equation:

$$
\begin{aligned}
Device_{base\_score} &= \frac{Pairing_{score} + Config_{score} + Upgrade_{score} + Services_{score} + Vulnerability_{score}}{Pairing_{total} + Config_{total} + Upgrade_{total} + Services_{total} + Vulnerability_{total}} \\
Pairing_{score} &= max(WiFi, LE, Wired, Manual) \\
Config_{score} &= max(Default, Forced) \\
Upgrade_{score} &= max(Manual, Consent, Auto) \\
Services_{score}(x) &= Services_{total} * Importance \\
Vulnerability_{score} &= CVSS_{score}(level = Critical) + CVSS_{score}(level = High) + CVSS_{score}(level = Medium) + CVSS_{score}(level = Low) \\
CVSS_{score}(level) &= CVSS_{total}(level) * Importance
\end{aligned}
\tag{6.1}
$$

We can assign points based on the importance of each security property. For example, we can assign a value to each pairing method. If devices that pair (connect to the Internet) using WiFi are risky in the network, we can assign a large value to WiFi pairing method that will reflect the risk for devices using WiFi pairing. We can also assign values between zero and 100 to $Importance$ for the *Services* and *CVSS* categories that accounts for the deployment environment. For example, if a device has five or more network services but is in an isolated Virtual Local Area Network (VLAN), then we assign an $Importance$ value of 25%. Additionally, if critical CVSS vulnerabilities are very risky to a network, we can assign higher values to the $Importance$ variable. Similarly, we can calculate the risk score for the other components using the following equations:

$$
\begin{aligned}
Mobile_{base\_score} &= \frac{Data_{score} + Program_{score} + Privilege_{score}}{Data_{total} + Program_{total} + Privilege_{total}} \\
Cloud_{base\_score} &= \frac{Domain_{score} + TLS_{score} + Services_{score}}{Domain_{total} + TLS_{total} + Services_{total}} \\
Network_{base\_score} &= \frac{Protocol_{score} + MITM_{score} + Encryption_{score}}{Protocol_{total} + MITM_{total} + Encryption_{total}}
\end{aligned}
\tag{6.2}
$$

6.3.2 Risk Score Adjustment: Exposure

The longitudinal security evaluation results can adjust the score categories and weights using temporal features. In this instance of the scoring system, we will only consider the

device category since our longitudinal analysis only studied the device component over time. However, we can apply a similar exercise to other components with a longitudinal security evaluation. We define a metric to account for the exposure of vulnerabilities found on the device as E_w. Since the exposure weights impact the vulnerability category of the device score, we adjust only the $Vulnerability_{score}$ variable. We use the device base score equation (Equation 6.1) to incorporate the exposure weights (E_w).

$$Vulnerability_{weight\_score} = E_w * Vulnerability_{score} \tag{6.3}$$

6.3.3 Risk Score Adjustment: Threats

Next, we want to use the IoT malware insights to inform our risk model. IoT malware analysis focuses on IoT device targeting, which accounts for the device and network components and excludes the mobile and cloud components. However, for the network component, we include any network communications that involve the device, like mobile application-to-device (M2D) and device-to-cloud (D2C). The framework in chapter 5 describes five stages of the IoT malware lifecycle: infection, payload, persistence, capability, and communication. We use the analysis from the infection stage to adjust the security base score using an attack weight variable A_w. Similar to the longitudinal analysis, we increase the risk level if a device has a vulnerability that malware actively targets. We adjust the device exposure score equation (Equation 6.3) to incorporate the attack weights (A_w).

$$Vulnerability_{weight\_score} = A_w * E_w * Vulnerability_{score} \tag{6.4}$$

In addition to the device, we adjust the base score equation (Equation 6.2) for the

network component to account for the attack weights.

$$
\begin{aligned}
Network_{weight\_score} &= \frac{Protocol_{weight\_score} + MITM_{weight\_score} + Encryption_{weight\_score}}{Protocol_{total} + MITM_{total} + Encryption_{total}} \\
Protocol_{weight\_score} &= ThirdPartyDNS_{score} + CustomProt_{score} + A_w * UPnP_{score} + A_w * HTTP_{score} + NTPv3_s \\
MITM_{weighted\_score} &= A_w * D2C_{score} + M2C_{score} + A_w * M2D_{score} \\
Encryption_{weighted\_score} &= A_w * ED2C_{score} + EM2C_{score} + A_w * EM2D_{score}
\end{aligned}
\tag{6.5}
$$

We apply the attack weights to clear-text protocols (*UPnP* and *HTTP*) because they offer no security features. An attacker (Type 1) can abuse these protocols without exploitation. The weights applied to the Equation 6.3 and Equation 6.5 can result in the risk score being greater than one. To remediate this, we apply the *Min* function to bound the risk score (x) between zero and one. The Min function is the following:

$$f(x) = min(x, 1) \tag{6.6}$$

6.3.4 Risk Assessment Example

Base Risk Score

We calculate the base risk score for the *VeraLite* device. We will assume a Type 1 attacker (LAN access) for this risk assessment. The *VeraLite* pairs using a wired connection, default configuration, and manual update. The *VeraLite* exposes four network services on the LAN. The *VeraLite* has two critical, one high, two medium, and three low CVSS vulnerabilities. We present each security category's values in Table 6.1. We assign ten points per category and set the *importance* preference for each security property. We multiply the total category by the importance factor to get the score column. We can now use the score values directly into Equation 6.1 to calculate the risk score.

Table 6.1: Example of point assignment for the VeraLite device

Category	Total Points	Importance	Score
Pairing	10	30%	3
Configuration	10	100%	10
Upgrade	10	100%	10
Services	10	75%	7.5
Vulnerable Low	10	30%	3
Vulnerable Medium	10	50%	5
Vulnerable High	10	70%	7
Vulnerable Critical	10	80%	8
Total	80	- -	53.5

$$Device_{base\_score} = \frac{Pairing_{score} + Config_{score} + Upgrade_{score} + Services_{score} + Vulnerability_{score}}{Pairing_{total} + Config_{total} + Upgrade_{total} + Services_{total} + Vulnerability_{total}}$$

$$Device_{base\_score} = \frac{53.5}{80} = 0.67$$

Weighed Risk Score

Next, we assign values to the exposure period (E_w) and active attack (A_w) weights. Note, users can adjust the exposure and active attack weights for specific environments. In this instance, we define the exposure weights based on a flat network as shown in Figure 3.3. The following are the weight assignment for each exposure period: justify the below, clearify type 1

1. Exposure for less than one month should reduce the score by 20% because device update addresses vulnerability and reduces the exposure period.

2. Exposure between one and three months should reduce the score by 10% because device update addresses vulnerability and reduces the exposure period.

3. Exposure between three and six months should increase the score by 10% because delayed device update addresses vulnerability but increases the exposure period.

4. Exposure over six months should increase the score by 30% because device update never addresses the vulnerability, and the exposure period is indefinite.

For the active attack weight, we increase the targeted vulnerability by 50%. We summarize the weight assignment in Table 6.2.

Table 6.2: A summary of assigned weights to exposure and attack variables

	Category Description	Weight	Weight Factor
Exposure	Less than 1 month	-20%	0.80
	Between 1 and 3 months	-10%	0.90
	Between 3 and 6 months	10%	1.1
	More than 6 months	30%	1.3
Attack	Actively Exploitation	50%	1.5

The following is an example of a weighted risk score using the *VeraLite* device:

$$Vulnerability_{weight\_score} = A_w * E_w * Vulnerability_{score}$$

$$Vulnerability_{weight\_score} = 1.5 * 1.3 * Vulnerability_{score}$$

$$Vulnerability_{weight\_score} = 1.5 * 1.3 * (23) = 44.85$$

$$Device_{weight\_score} = \frac{75.35}{80} = 0.94$$

We assign a 1.3 weight factor to E_w since the *VeraLite* device is EOL with an exposure period of over six months. We assign a 1.5 weight factor to A_w because IoT malware is actively targeting a critical CVSS vulnerability on the *VeraLite* device. The weighted score increases the risk level by 40% from 0.67 to 0.94.

6.4 Case Study

We demonstrate how to calculate the risk score for multiple devices from the same vendor. Additionally, we demonstrate the impact on risk scores using longitudinal and threat analysis features.

6.4.1 Multiple Devices, One Vendor

In this case study, we will look at the *Belkin* devices that include the *Belkin Netcam*, *Belkin WeMo Link*, *Belkin WeMo Motion*, *Belkin WeMo Switch*, *Belkin WeMo Crockpot*. We derive

the security property values from Table 3.4, Table 3.6, Table 3.7, and Table 3.8. We assign importance factors based on our deployment environment Figure 3.3.

Table 6.3: The risk scores for the Belkin WeMo Devices.

Name	Device	Mobile	Cloud	Network
Netcam	0.14	0.46	0.61	0.39
Link	0.21	0.38	0.34	0.46
Motion	0.19	0.38	0.07	0.46
Switch	0.19	0.38	0.45	0.46
Crockpot	0.29	0.38	0.58	0.36

In Table 6.3, we present the risk scores for comparison. Our component-based framework highlights the variation in risk across different components for devices from the same vendor. Surprisingly, the results show that the device risk levels are inconsistent for a single vendor. For example, the cloud component risk score for all five devices ranges from 0.07 to 0.61. The mobile and network component is more consistent, ranging from 0.38 to 0.46 and 0.39 to 0.46, respectively. The mobile components' consistency is because the four products use the same mobile application. We can see a correlation between the mobile and network components since the network protocols enable communication between the device and the mobile application. We hypothesize that a vendor builds their products consistently using a predefined pipeline. However, our results show that each product has varying security issues contributing to the risk score in Table 6.3.

6.4.2 Incorporating Exposure and Attack Weights

This case study demonstrates how temporal and threat analysis can better prioritize insecure devices. To do so, we compare the base risk score of four devices with their weighted exposure and attack risk score. In Table 6.4, we calculate the risk score for four devices; *VeraLite*, *nVidia Shield*, *Apple TV*, and *D-Link Camera*. We calculate the base risk score using Equation 6.1 and the weighed exposure and attack risk score using Equation 6.3. We use the weights defined in Table 6.2.

The *VeraLite* device and network risk score increases due to long vulnerability exposure and active malware attacks that target the vulnerabilities. This device has the highest risk,

Table 6.4: The base risk score and temporal and threat weighed risk score for four devices.

Name	Base		Temporal & Threat	
	Device	Network	Device	Network
VeraLite	0.67	0.54	0.94	0.68
nVidia Shield	0.40	0.14	0.40	0.14
Apple TV	0.12	0.11	0.23	0.11
D-Link Camera	0.38	0.21	0.39	0.32

and we should prioritize its security measure over other devices. Notably, the device has a critical CVSS vulnerability (*CVE-2013-4861*) that IoT malware threats target. The *nVidia Shield* risk score does not change, although the device has medium CVSS vulnerabilities that have more than six months of exposure. The device component had a low CVSS vulnerability that the vendor fixed, which lowered the risk score. However, because the medium CVSS vulnerabilities remain, the exposure weight factor compensates for the low CVSS vulnerability. An internal risk score change did not affect the overall device score. The exposure and attack weights did not impact the *nVidia Shield* network component.

The *Apple TV* device risk score increased by almost 100% but remained the lowest relative to other devices. The device had a high CVSS vulnerability with three months of exposure. The vulnerability was a TCP/IP Initial Sequence Number (ISN) Reuse Weakness, see Figure 4.2. An ISN tracks the order of messages sent between two endpoints. The reuse of ISN vulnerability allows an attacker to predict the next ISN. If the attacker successfully guesses the ISN, they can send spoofed messages to disrupt communication, steal data, or gain unauthorized access. However, because the exposure period was three months, the weight factor for this vulnerability is 0.9, which reduces the risk score. The factor is less than one since the vendor addressed the vulnerability within three months. The risk score for the network component of the *Apple TV* did not change.

The last example is the *D-Link Camera* device. The risk score for the device and network component increased. The device risk score increased because there is a low CVSS vulnerability (see Figure 4.4) that had more than six months of exposure. The device's risk score increased due to the long exposure period. The network risk score also increased,

but the increase is due to the attack weight since the device uses Universal Plug and Play (UPnP), is susceptible to MITM for Mobile-to-Device communication and does not use encryption for Mobile-to-Device communication. A network malware threat (Type 1) can attack the device's communication by intercepting or modifying the traffic between the mobile application and the device. Our risk score framework is an example that practitioners can customize to fit their deployment environment. As we stated before, the risk scoring in this instance may be subjective; however, the framework's purpose is to provide a basis for practitioners to calculate the risk of their IoT deployments and prioritize security measures.

CHAPTER 7
CONCLUSION

Risk assessment frameworks are pivotal in effectively guiding organizations to prioritize their efforts and resources. However, these frameworks' efficacy intrinsically depends on the quality of empirical data. A comprehensive understanding of the vulnerabilities and threats associated with IoT deployments, enriched by context-aware insights, is paramount to the accuracy and reliability of risk assessment outcomes. Consequently, this interdependence necessitates continuously pursuing up-to-date, pertinent knowledge to enable informed decision-making and foster robust IoT security risk assessment. This body of work has shown that comprehensive and iterative vulnerability and threat analysis at scale improves the risk assessment of IoT deployments. In this chapter, we delve into additional facets of IoT vulnerabilities and threat intelligence, emphasizing the criticality of empirical data in shaping and refining risk assessment frameworks. Lastly, we will highlight the broader impact of our research on the academic community and industry.

7.1 Risk Assessment and Empirical Data

This work introduced a systematic security evaluation framework to identify vulnerabilities in IoT deployment. The framework provides a more comprehensive view of IoT deployments by accounting for the device, companion mobile application, cloud backends, and network communication. We then applied the security evaluation framework iteratively for one year and characterized the vulnerabilities that impact IoT devices' lifecycle. We derived insights for vulnerability exposure time and incorporated them in our risk score framework to provide more accurate risk levels for IoT deployments. We then applied empirical threat analysis insights to enhance our risk assessment framework further and improve the accuracy of risk levels. Evidently, The more empirical observations we collect

on IoT deployment vulnerabilities and threats, the more accurate our risk assessment results will be. To that end, we propose future studies to uncover additional facets to improve the risk assessment of IoT deployments.

7.1.1 Understanding Attacks and Abuse on IoT Deployments

Understanding how attackers abuse IoT devices can provide essential insights to inform risk assessment accurately. For example, identifying the most targeted type of IoT devices and how they contribute to the attack chain can help practitioners monitor similarly targeted assets in their environment. We can adjust the risk models to account for highly targeted device types from a risk assessment perspective. Moreover, understanding the attacker's behavior on the device can help identify intentions and potential attack paths that can impact other assets on the same network. For instance, if the attacker installs software on the device or only uses the device to relay traffic may signal different intentions. One looks to persist on the device (long-term infection), while the other looks to use the device as a redirector (proxy). These intentions inform the risk of different attack types, and we can use that to revise the IoT deployment risk models.

Conducting such a study is challenging. The diversity of device types hinders the scalability of many approaches. We must rely on software emulation or simulation of devices to achieve scalability. However, emulation and simulation are low-fidelity approaches that many attackers can detect and evade. Moreover, threats evolve and adapt to detect and evade defensive and measurement systems. We must develop a technique to identify new defensive and evasive tactics and notify us to analyze and understand the new tactics automatically. Finally, the threat intelligence gathered from these measurement systems must be vetted for deceptive behavior. Identifying deceptive behavior is challenging and may require the system to correlate historical events.

7.1.2 A Longitudinal Security Evaluation of IoT Cloud Backends

Our initial security evaluation of IoT deployment found that cloud backends have many security issues. Moreover, devices from the same vendor can use different instances of cloud backends with varying vulnerabilities and exposure. These backends change frequently and may be different for various geographically deployed devices. These inconsistencies can create gaps and additional complexities that IoT operators must consider in their risk assessment. In particular, these backends are critical in facilitating data processing, storage, and device management. Characterizing their security over time can inform the risk assessment models by accounting for external risks to IoT deployments.

There are several challenges to evaluating the security of IoT cloud backends. Every device has unique services and capabilities that are often cloud supported. For example, home assistant devices use machine learning models to respond to voice commands. Other devices may use cloud backends as a management interface to command IoT devices to perform functions like unlocking a door or turning on or off a light. Developing a generalizable approach to account for the diversity in custom backend functionality is challenging. In addition, the use of encryption between device and cloud backends makes it challenging to inspect the connections and evaluate their security. For longitudinal analysis, cloud backend churn must be tracked and cataloged throughout the evaluation period. As devices update or cloud outages happen, the devices often rely on redundant cloud backends that we need to account for in the evaluation. Lastly, legal and ethical considerations are of utmost importance when evaluating the security of cloud backends. Since we do not have ownership or permission to probe the cloud backends actively, we must carefully design our measurement techniques to be passive and non-disruptive.

7.1.3 Understanding Attacks on The IoT Software Supply Chain

Software developers rely on libraries and modules to speed up development. These modules can contain vulnerabilities that may impact IoT firmware. Studying the attacks that

can impact the software supply chain for IoT deployments, like the device firmware and mobile applications, can identify external risks. These potential vulnerabilities can impact many types of IoT devices. Identifying if a particular IoT deployment depends on a highly vulnerable library that can inform IoT operators of high-risk devices and prioritize their security measures. A large-scale study on the software supply chain of IoT firmware and mobile applications can inform risk models to account for latent software vulnerabilities.

There are several challenges to studying the IoT software supply chain. Many IoT devices are closed-source, or their firmware is not publicly available. We must leverage binary analysis and hardware reverse engineering to extract and analyze the firmware software to address this challenge. Moreover, the binary analysis techniques must account for varying system architectures. Additional challenges must be addressed within the binary analysis to identify library and module dependencies accurately. These challenges include inferring compiler optimization, identifying code modules, and matching the modules to third-party libraries. Lastly, this effort must be large-scale to account for the many software libraries and the combination of different software interactions.

7.1.4 An Automated and Iterative Security Evaluation Framework for LE IoT Protocols

Low-energy protocols are widely used in IoT devices. A security evaluation of these protocols can identify potential vulnerabilities that further improve the risk assessment model. For example, Bluetooth, ZigBee, and ZWave are LE protocols that IoT devices use for authentication, sending sensitive data, and controlling peripheral devices. Prior work [21] shows the impact of such vulnerabilities on large IoT deployments. Therefore, to account for various threat models, like type 3, a continuous evaluation of LE protocols can identify more vulnerabilities that impact IoT deployments.

There are several challenges in developing an automated methodology to evaluate LE protocols. The diversity of protocols and variation of protocols require a substantial engineering effort. We require specialized radio hardware to monitor and study these LE

protocols. Additionally, automating the LE monitoring tools is challenging for many devices near. For example, if many devices use the same LE protocol, we must differentiate between signal beacons, which require advanced signal processing techniques. Lastly, we want to democratize the tools for automating security evaluation. To do so, we must utilize affordable hardware, which may have low-fidelity signal processing, and provide support for different protocol versions.

7.2 Broader Impact of Empirical Studies

Beyond the risk assessment application, our empirical studies have a broad impact.

7.2.1 Security Evaluation of Home-based IoT Deployments

We built a state-of-the-art testbed of off-the-shelf home-based IoT devices and evaluated their security. In doing so, we captured the entire evaluation process by recording the full-packet captures from 45 diverse home-based IoT devices. We thoroughly documented our methodology and device configurations, which we make public to researchers and practitioners. We created a dedicated resource known as *YourThings*[1] to host the experiment artifacts. Since then, *YourThings* has garnered the attention of the research community, consumer technology advocacy groups, acclaimed media, and practitioners from the Internet Engineering Task Force (IETF). Our paper, which appears in the proceeding of the 2019 IEEE Security & Privacy, and dataset have accumulated over 300 citations and used in over 20 projects by the academic community. The dataset of 150GB of full-packet network capture traces has been downloaded over 130 times. Consumer technology advocacy groups, such as the Wirecutter, have used our data and methodology to inform consumers about secure IoT devices and provide recommendations. Our work has been featured in acclaimed media such as Newsweek and the New York Times, which disseminate technical material to wider audiences.

[1] https://yourthings.info

Academic Impact

To highlight the impact of the citations beyond the numerical count, Table 7.1 shows the publication societies (conferences and journals) for the works that cite *YourThings*. We can see that IEEE society is the highest, followed by ACM, Springer, and Elsevier. These publication societies are among the most active in the field of computer engineering and computer science. Additionally, eight book series have referenced our work. Our work has had a broad impact on the research community with citations from areas in privacy, system design, network security, energy, human-computer interaction, measurements, access control, digital health, artificial intelligence, cyber forensics, and software testing to name a few. The "Other" category includes dissertations, technical reports, Arxiv papers, and miscellaneous digital publications citations. It is clear from Table Table 7.1 that the impact of our work on academic research is substantial.

Table 7.1: Citation sources by societies.

Publication Societies	**Citations**
IEEE	67
ACM	29
Springer	15
Elsevier	10
Usenix	9
NDSS	2
Others	172
Total	304

Applied Research

To provide more concrete examples of how our work has impacted research, Table 7.2 presents a sample of 15 works that incorporate our dataset into their research. Specifically, these research projects span different applications such as device identification, behavior detection and classification, security and privacy device labels, IoT authentication techniques, device behavior transparency, and surveys on security assessments and device

identification. These publications appear in various top security venues like IEEE S&P (Oakland) and NDSS. Beyond academic works, our dataset has been used by practitioners in the industry. McAfee detection engineers used our dataset to prototype certificate signatures to reduce false positives when identifying IoT device traffic. We worked with a McAfee security engineer to answer questions about our dataset and how they can extract certificates associated with each device's communication. Hamilton Beach, an appliance manufacturer, leveraged our systematic security evaluation methodology to vet cloud platforms they intended to use with their smart-home connected products. Lastly, our dataset is helping researchers define a protocol for sending DNS messages over the Constrained Application Protocol [211].

Table 7.2: A sample of 15 academic research projects using the YourThings dataset.

Paper	**Year**	**Application**
HomeSnitch [212]	2019	Identifying and Controlling IoT Behavior
Storming the Kasa [213]	2019	Reproducibility of Security Evaluation
Hestia [214]	2019	Defining Least Privilage Network Policy
Ask the Experts [215]	2020	Security and Privcy Device Labels
PingPong [216]	2020	Automated Device Signature Identification
ML Traffic Classification [217]	2020	IoT Traffic Classification
Automated Standards [218]	2020	Automation of Security Assessment
IoTFinder [137]	2020	DNS-based Device Identification
IoT ETEI [219]	2021	Device Identification
FIAT [220]	2021	Improved IoT Authentication System
ByteIoT [221]	2021	IoT Device Identification
Standards and Technology [222]	2021	Survey on IoT Security Evaluation
Survey on Device Behavior [223]	2021	Survey on IoT Device Behavior Identification
PinBall [224]	2021	IoT Device Event Identification

Improving Consumer Security

The Wirecutter has incorporated our methodology and evaluation results to recommend secure light bulbs to consumers [225]. We worked with Hamilton Beach, a consumer appliance company, to apply our security assessment methodology and identify secure IoT deployment practices. We worked with Aura, a digital identity management company to recommend techniques for identifying and securing deployed home-based IoT devices running inside their customers' networks.

News Outlets

Our work has been featured in a Newsweek piece about the security and privacy of home-based IoT devices [226]. The article highlights the evolution of home-based IoT products and the risk associated with using these internet-connected devices. The New York Times featured our work on Amazon's Sidewalk project that uses nearby Amazon home-based IoT devices to share internet connections [227]. The article describes the new feature, informs users how to turn it off, and highlights our work to provide a security and privacy perspective. The Verge featured our work in reference to the privacy issues found in the Anker Eufy Security Camera. Eufy claimed the video feed is end-to-end encrypted (E2EE); however, that was not what the case [228].

7.2.2 Large-Scale Analysis of the IoT Malware Lifecycle

To prepare for future IoT attacks, we created a dedicated website called *BadThings*[2] for researchers to study IoT malware threats in-the-wild. The broader impact of our research on the IoT malware lifecycle spans multiple dimensions. It has a far-reaching influence within the cybersecurity community and across various multidisciplinary fields. Our work on the large-scale analysis of IoT malware and the subsequent release of our binary analysis platform, binary files, and analysis artifacts have fostered a collaborative environment among researchers worldwide, enhancing the security posture of IoT devices and networks. Our paper has garnered significant attention in the research community. It has been cited over 45 times in papers appearing in top venues, reflecting its substantial influence on the field. Furthermore, our website was accessed by over 1,200 researchers from ten different countries, including the US, China, Germany, Canada, Japan, Singapore, Hong Kong, Indonesia, the Netherlands, and South Korea. This global engagement underpins the importance and applicability of our research findings and tools in diverse contexts.

The insights gained from our research can be applied across multiple fields. For exam-

[2]https://badthings.info

ple, our findings can inform policymakers and regulators to devise more stringent security standards for IoT devices. Additionally, our work can be utilized by IoT device manufacturers to develop more secure products, minimizing the risk of future large-scale IoT malware infection breakouts. The international interest in our research has fostered collaboration and knowledge sharing among researchers from diverse backgrounds. This collaborative environment can help accelerate advancements in IoT security, malware defense, and innovative solutions to emerging threats. The resources we have released serve as valuable tools for educating and training the next generation of cybersecurity professionals, equipping them with the skills and knowledge necessary to tackle the evolving threat landscape in IoT and related fields. By sharing our binary analysis platform, binary files, and analysis artifacts, we have contributed to open science, which promotes transparency, reproducibility, and accessibility in research. This approach helps to accelerate the pace of innovation and discovery in the cybersecurity field and democratizes access to all.

7.3 Closing Remarks

This dissertation aims to combine real-world security evaluation observations and malware threat analysis to quantify the risk and prioritize security measures for IoT deployments. We have shown that a systematic framework can identify more vulnerabilities in different components of an IoT deployment. We conducted a longitudinal security evaluation of IoT devices and showed how vulnerability exposures could increase the risk level of IoT deployments. We conducted a large-scale analysis of the IoT malware lifecycle to inform our risk assessment model with threat data. We demonstrated how the risk assessment model could prioritize high-risk devices based on vulnerabilities, exposure, and threats. Finally, we highlight the broader impact of our empirical studies and results on the academic community, industry, and consumer advocacy groups.

REFERENCES

[1] M. Antonakakis *et al.*, "Understanding the mirai botnet," in *Proc. of the 26th USENIX Security*, Vancouver, BC, Canada, Aug. 2017.

[2] M. Barnes, *Alexa, are you listening?* https://labs.mwrinfosecurity.com/blog/alexa-are-you-listening, 2017.

[3] Clinton, Ike and Cook, Lance and Banik, Shankar, *A Survey of Various Methods for Analyzing the Amazon Echo*, https://vanderpot.com/Clinton_Cook_Paper.pdf, 2016.

[4] B. Ur, J. Jung, and S. Schechter, "The current state of access control for smart devices in homes," in *Workshop on Home Usable Privacy and Security (HUPS)*, 2013.

[5] C. Wuesst, *How my TV got infected with ransomware and what you can learn from it*, https://www.symantec.com/connect/blogs/how-my-tv-got-infected-ransomware-and-what-you-can-learn-it, 2015.

[6] A. Chapman, *Hacking into Internet Connected Light Bulbs*, https://www.contextis.com/blog/hacking-into-internet-connected-light-bulbs, 2014.

[7] B. Rodrigues, *ARRIS Cable Modem has a Backdoor in the Backdoor*, https://w00tsec.blogspot.com/2015/11/arris-cable-modem-has-backdoor-in.html, 2015.

[8] J. Max, *Backdooring the Frontdoor Hacking a "perfectly secure" smart lock.* https://media.defcon.org/DEFCON24/DEFCON24presentations/DEFCON-24-Jmaxxz-Backdooring-the-Frontdoor.pdf, 2016.

[9] Y. Tian *et al.*, "Smartauth: User-centered authorization for the internet of things," in *Proc. of the 26th USENIX Security*, Vancouver, BC, Canada, Aug. 2017.

[10] J. Obermaier and M. Hutle, "Analyzing the security and privacy of cloud-based video surveillance systems," in *Proc. of the 2nd ACM IoTPTS*, 2016.

[11] S. P. Kavalaris and E. Serrelis, "Security issues of contemporary multimedia implementations: The case of sonos and sonosnet," in *The International Conference in Information Security and Digital Forensics*, 2014.

[12] A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, and S. Antipolis, "A large-scale analysis of the security of embedded firmwares.," in *Proc. of the 23rd USENIX Security*, San Diego, CA, Aug. 2014.

[13] D. Lodge, *Steal your Wi-Fi key from your doorbell? IoT WTF!* https://www.pentestpartners.com/security-blog/steal-your-wi-fi-key-from-your-doorbell-iot-wtf/, 2016.

[14] L. Franceschi-Bicchierai, *Hackers Make the First-Ever Ransomware for Smart Thermostats*, https://motherboard.vice.com/en_us/article/aekj9j/internet-of-things-ransomware-smart-thermostat, 2016.

[15] C. O'Flynn, *A Lightbulb Worm?* http://colinoflynn.com/wp-content/uploads/2016/08/us-16-OFlynn-A-Lightbulb-Worm-wp.pdf, 2016.

[16] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proc. of the 37th S&P Oakland*, San Jose, CA, May 2016.

[17] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging iot application frameworks.," in *Proc. of the 25th USENIX Security*, Austin, TX, Aug. 2016.

[18] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, *Smart Nest Thermostat: A Smart Spy in Your Home*, https://www.blackhat.com/docs/us-14/materials/us-14-Jin-Smart-Nest-Thermostat-A-Smart-Spy-In-Your-Home-WP.pdf, 2014.

[19] S. Morgenroth, *How I Hacked my Smart TV from My Bed via a Command Injection*, https://www.netsparker.com/blog/web-security/hacking-smart-tv-command-injection/, 2017.

[20] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Security implications of permission models in smart-home application frameworks," in *Proc. of the 38th S&P Oakland*, San Jose, CA, May 2017.

[21] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *Proc. of the 38th S&P Oakland*, San Jose, CA, May 2017.

[22] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *Proc. of the 24th ACM CCS*, Dallas, TX, Oct. 2017.

[23] Q. Wang, W. U. Hassan, A. Bates, and C. Gunter, "Fear and logging in the internet of things," in *Proc. of the 2018 NDSS*, San Diego, CA, Feb. 2018.

[24] D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," in *Proc. of the 17th ACM CCS*, Chicago, Illinois, Oct. 2010.

[25] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "Pscout: Analyzing the android permission specification," in *Proc. of the 19th ACM CCS*, Raleigh, NC, Oct. 2012.

[26] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in android applications," in *Proc. of the 20th ACM CCS*, Berlin, Germany, Oct. 2013.

[27] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of google play," in *Proc. of the 2014 ACM SIGMETRICS*, 2014.

[28] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-phones attacking smart-homes," in *Proc. of the 9th ACM WiSec*, 2016.

[29] S. Demetriou *et al.*, "Hanguard: Sdn-driven protection of smart home wifi devices from malicious mobile apps," in *Proc. of the 10th ACM WiSec*, 2017.

[30] J. Chen *et al.*, "Iotfuzzer: Discovering memory corruptions in iot through app-based fuzzing," in *Proc. of the 2018 NDSS*, San Diego, CA, Feb. 2018.

[31] C. Nandi and M. D. Ernst, "Automatic trigger generation for rule-based smart homes," in *Proc. ACM PLAS*, 2016.

[32] A. Blaich and A. Hay, *Hello Barbie Initial Security Analysis*, https://static1.squarespace.com/static/543effd8e4b095fba39dfe59/t/56a66d424bf1187ad34383b2/1453747529070/HelloBarbieSecurityAnalysis.pdf, 2016.

[33] J. Wilson, R. S. Wahby, H. Corrigan-Gibbs, D. Boneh, P. Levis, and K. Winstein, "Trust but verify: Auditing the secure internet of things," in *Proc. of the 15th MobiSys*, 2017.

[34] M. Surbatovich, J. Aljuraidan, L. Bauer, A. Das, and L. Jia, "Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of ifttt recipes," in *Proc. of the 26th International World Wide Web Conference (WWW)*, Perth, Australia, 2017.

[35] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Decentralized action integrity for trigger-action iot platforms," in *Proc. of the 2018 NDSS*, San Diego, CA, Feb. 2018.

[36] US-CERT/NIST, *CVE-2011-3389*, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-3389, 2011.

[37] D. Garcia, *Upnp mapping*, 2011.

[38] N. J. AlFardan and K. G. Paterson, "Lucky thirteen: Breaking the tls and dtls record protocols," in *Proc. of the 34th S&P Oakland*, San Francisco, CA, May 2013.

[39] M. Ryan, *Bluetooth Smart:The Good, The Bad, The Ugly... and The Fix*, https://lacklustre.net/bluetooth/bluetooth_smart_good_bad_ugly_fix-mikeryan-blackhat_2013.pdf, 2013.

[40] B. Fouladi, *Honey, I'm Home!! Hacking Z-Wave Home Automation Systems*, https://cybergibbons.com/wp-content/uploads/2014/11/honeyimhome-131001042426-phpapp01.pdf, 2013.

[41] N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt, "On the security of rc4 in tls and wpa," in *Proc. of the 22th USENIX Security*, Washington, DC, Aug. 2013.

[42] J. Selvi, *Bypassing HTTP Strict Transport Security*, https://www.blackhat.com/docs/eu-14/materials/eu-14-Selvi-Bypassing-HTTP-Strict-Transport-Security-wp.pdf, 2014.

[43] B. Möller, T. Duong, and K. Kotowicz, "This POODLE Bites: Exploiting The SSL 3.0 Fallback," Google, Tech. Rep., 2014.

[44] US-CERT/NIST, *CVE-2015-0204*, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-0204, 2015.

[45] US-CERT/NIST, *CVE-2012-4929*, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-4929, 2015.

[46] B. Beurdouche *et al.*, "A messy state of the union: Taming the composite state machines of tls," in *Proc. of the 36th S&P Oakland*, San Jose, CA, May 2015.

[47] D. Adrian *et al.*, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proc. of the 22nd ACM CCS*, Denver, Colorado, Oct. 2015.

[48] T. Zillner and S. Strobl, *Zigbee Exploited: The good, the bad and the ugly*, https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf, 2015.

[49] N. Aviram *et al.*, "DROWN: Breaking TLS using SSLv2," in *Proc. of the 25th USENIX Security*, Austin, TX, Aug. 2016.

[50] S. Jasek, *GATTacking Bluetooth Smart devices*, http://gattack.io/whitepaper.pdf, 2016.

[51] P. Kintis, Y. Nadji, D. Dagon, M. Farrell, and M. Antonakakis, "Understanding the privacy implications of ecs," in *Proc. of the DIMVA*, Donostia-San Sebastián, ES, Jul. 2016.

[52] N. Apthorpe, D. Reisman, and N. Feamster, "Closing the blinds: Four strategies for protecting smart home privacy from network observers," in *ConPro*, 2017.

[53] D. Wood, N. Apthorpe, and N. Feamster, "Cleartext data transmissions in consumer iot medical devices," in *IoT S&P*, 2017.

[54] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine, "Survivable key compromise in software update systems," in *Proc. of the 24th ACM CCS*, Dallas, TX, Oct. 2017.

[55] Y. Acar, M. Backes, S. Bugiel, S. Fahl, P. McDaniel, and M. Smith, "Sok: Lessons learned from android security research for appified software platforms," in *Proc. of the 37th S&P Oakland*, San Jose, CA, May 2016.

[56] C. Zuo, W. Wang, Z. Lin, and R. Wang, "Automatic forgery of cryptographically consistent messages to identify security vulnerabilities in mobile services.," in *Proc. of the 2016 NDSS*, San Diego, CA, Feb. 2016.

[57] *About - IFTTT*, https://ifttt.com/about, 2018.

[58] *Work Super Smart - Automate.io*, https://automate.io, 2018.

[59] *Cloud Business App Integration*, https://cloudwork.com, 2018.

[60] M. Riley, A. Sharpe, and J. Robertson, *Equifax Suffered a Hack Almost Five Months Earlier Than the Date It Disclosed*, https://www.bloomberg.com/news/articles/2017-09-18/equifax-is-said-to-suffer-a-hack-earlier-than-the-date-disclosed, 2017.

[61] G. De Vynck, *Orbitz Hack May Have Compromised 880,000 Credit Cards*, https://www.bloomberg.com/news/articles/2018-03-20/expedia-s-orbitz-hack-may-have-compromised-880-000-credit-cards, 2018.

[62] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," in *DAT*, 2016.

[63] J. Novet, *Amazon scrambles to fix cloud networking issue affecting companies like Atlassian, Twilio*, https://www.cnbc.com/2018/03/02/amazon-cloud-networking-outage-affecting-atlassian-twilio-slack.html, 2018.

[64] N. Garun, *Yahoo says all 3 billion user accounts were impacted by 2013 security breach*, https://www.theverge.com/2017/10/3/16414306/yahoo-security-data-breach-3-billion-verizon, 2017.

[65] S. Moss, *Major ddos attack on dyn disrupts aws, twitter, spotify and more*, https://www.theverge.com/2017/10/3/16414306/yahoo-security-data-breach-3-billion-verizon, 2016.

[66] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and A. J. Halderman, "A search engine backed by internet-wide scanning," in *Proc. of the 22nd ACM CCS*, Denver, Colorado, Oct. 2015.

[67] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proc. of the 17th S&P Oakland*, Oakland, CA, May 1996.

[68] *AWS IoT Core*, https://aws.amazon.com/iot-core/, 2018.

[69] *IoT Hub Connect, monitor, and manage billions of IoT assets*, https://cloud.google.com/solutions/iot/, 2018.

[70] *GOOGLE CLOUD IOT: Intelligent IoT platform that unlocks business insights from your global device network*, https://cloud.google.com/solutions/iot/, 2018.

[71] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari, *Client subnet in dns queries*, http://www.ietf.org/rfc/rfc7871.txt, 2016.

[72] A. Bellissimo, J. Burgess, and K. Fu, "Secure software updates: Disappointments and new challenges.," in *HotSec*, 2006.

[73] CERT/CC, *Vulnerability note vu#361684*, https://www.kb.cert.org/vuls/id/361684, 2015.

[74] GNUcitizen, *Hacking the interwebs*, http://www.gnucitizen.org/blog/hacking-the-interwebs, 2008.

[75] HD Moore, "Security Flaws in Universal Plug and Play," Tech. Rep., 2013.

[76] Bluetooth SIG, *Bluetooth Low Energy - Bluetooth Technology Website*, https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy, 2016.

[77] Alliance, Zigbee and others, *Zigbee Specification*, 2006.

[78] Z-Wave Alliance, *About Z-Wave Technology*, http://z-wavealliance.org/about_z-wave_technology, 2016.

[79] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *Proc. of the 11th ACM Symposium on Information, Computer and Communications Security (ASI-ACCS)*, Xi'an, China, Jun. 2016.

[80] *Zigbee "insecure rejoin" faq*, https://support.smartthings.com/hc/en-us/articles/208201243-ZigBee-Insecure-Rejoin-FAQ, 2018.

[81] Z-Wave Alliance, *Z-Wave Transport-Encapsulation Command Class Specification*, http://zwavepublic.com/sites/default/files/command_class_specs_2017A/SDS13783-5Z-WaveTransport-EncapsulationCommandClassSpecification.pdf, 2017.

[82] Zigbee Alliance, *Zigbee: Securing the Wireless IoT*, http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/, 2015.

[83] J. Clark and P. C. van Oorschot, "Sok: Ssl and https: Revisiting past challenges and evaluating certificate trust model enhancements," in *Proc. of the 34th S&P Oakland*, San Francisco, CA, May 2013.

[84] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," *IEEE Security & Privacy*, 2003.

[85] H. Zuzana, *Malicious campaign targets south korean users with backdoor-laced torrents*, https://web.archive.org/web/20190822042548/https://www.welivesecurity.com/2019/07/08/south-korean-users-backdoor-torrents/, Online; accessed 25 January 2020, 2019.

[86] C. Daniel, *Interesting information about ssh scans*, https://web.archive.org/web/20160430170921/https://dcid.me/blog/2006/03/interesting-information-about-ssh-scans/, Online; accessed 25 January 2020, 2006.

[87] G. McDonald, L. O. Murchu, S. Doherty, and E. Chien, *Stuxnet 0.5: The missing link*, https://web.archive.org/web/20200208170135/https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/stuxnet-missing-link-13-en.pdf, Online; accessed 25 January 2020, 2013.

[88] N. Provos, P. Mavrommatis, M. Rajab, and F. Monrose, "All your iframes point to us," Aug. 2008.

[89] T. Holz, M. Steiner, F. Dahl, E. Biersack, F. C. Freiling, *et al.*, "Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[90] S. Shin, R. Lin, and G. Gu, "Cross-analysis of botnet victims: New insights and implications," in *Proc. of the 14th RAID*, Menlo Park, California, Sep. 2011.

[91] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraş, "The dropper effect: Insights into malware distribution with downloader graph analytics," in *Proc. of the 22nd ACM CCS*, Denver, Colorado, Oct. 2015.

[92] B. Stone-Gross *et al.*, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proc. of the 16th ACM CCS*, Chicago, Illinois, Nov. 2009.

[93] L. Lu, V. Yegneswaran, P. Porras, and W. Lee, "Blade: An attack-agnostic approach for preventing drive-by malware infections," in *Proc. of the 17th ACM CCS*, Chicago, Illinois, Oct. 2010.

[94] L. Invernizzi *et al.*, "Nazca: Detecting malware distribution in large-scale networks," in *Proc. of the 2014 NDSS*, San Diego, CA, Feb. 2014.

[95] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proc. of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2006.

[96] P. Kotzias, L. Bilge, P.-A. Vervier, and J. Caballero, "Mind your own business: A longitudinal study of threats and vulnerabilities in enterprises," in *Proc. of the 2019 NDSS*, San Diego, CA, Feb. 2019.

[97] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna, "Polymorphic worm detection using structural information of executables," in *Proc. of the 8th RAID*, Seattle, Washington, Sep. 2005.

[98] P. Barford and V. Yegneswaran, "An inside look at botnets," *Malware Detection*, 2007.

[99] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging," in *Proceedings of the 1st Usenix Workshop on Hot Topics in Understanding Botnets*, 2007.

[100] D. Dagon, G. Gu, C. P. Lee, and W. Lee, "A taxonomy of botnet structures," in *Proc. of the 23nd ACSAC*, 2007.

[101] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Proc. of the 15th NDSS*, San Diego, CA, Feb. 2008.

[102] M. POLYCHRONAKIS, "Ghost turns zombie: Exploring the life cycle of web-based malware," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[103] C. Kanich *et al.*, "Spamalytics: An empirical analysis of spam marketing conversion," in *Proc. of the 15th ACM CCS*, Alexandria, VA, Oct. 2008.

[104] C. Y. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song, "Insights from the inside: A view of botnet management from infiltration," in *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, 2010.

[105] M. Lindorfer, C. Kolbitsch, and P. M. Comparetti, "Detecting environment-sensitive malware," in *Proc. of the 14th RAID*, Menlo Park, California, Sep. 2011.

[106] C. Rossow, C. Dietrich, and H. Bos, "Large-scale analysis of malware downloaders," in *Proc. of the DIMVA*, Jul. 2012.

[107] C. Gañán, O. Cetin, and M. van Eeten, "An empirical analysis of zeus c&c lifetime," in *Proc. of the 10th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, Singapore, Apr. 2015.

[108] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proc. of the 33rd S&P Oakland*, San Francisco, CA, May 2012.

[109] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee, "The core of the matter: Analyzing malicious traffic in cellular carriers," in *Proc. of the 20th NDSS*, San Diego, CA, Feb. 2013.

[110] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. Van Der Veen, and C. Platzer, "Andrubis–1,000,000 apps later: A view on current Android malware behaviors," in *Proceedings of the 3rd workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, 2014.

[111] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, "The evolution of android malware and android analysis techniques," *ACM Computing Surveys (CSUR)*, 2017.

[112] K. Stevens and D. Jackson, *Zeus banking trojan report*, https://web.archive.org/web/20191222124154/https://www.secureworks.com/research/zeus, Online; accessed 25 January 2020, 2010.

[113] M. Cruz, *Security 101: The rise of fileless threats that abuse powershell*, https://web.archive.org/save/https://www.trendmicro.com/vinfo/mx/security/news/security-technology/security-101-the-rise-of-fileless-threats-that-abuse-powershell, Online; accessed 25 January 2020, 2017.

[114] M. T. Paracha, D. J. Dubois, N. Vallina-Rodriguez, and D. Choffnes, "Iotls: Understanding tls usage in consumer iot devices," in *Proc. of the 21st ACM SIGCOMM Conference on Internet Measurement (IMC)*, Virtual, Nov. 2021.

[115] H. Zhang, A. Anilkumar, M. Fredrikson, and Y. Agarwal, "Capture: Centralized library management for heterogeneous iot devices," in *Proc. of the 30th USENIX Security*, Aug. 2021.

[116] T. Sasaki, A. Fujita, C. H. Ganán, M. van Eeten, K. Yoshioka, and T. Matsumoto, "Exposed infrastructures: Discovery, attacks and remediation of insecure ics remote management devices," in *Proc. of the 43nd S&P Oakland*, May 2022.

[117] IADMIN, *Hydra IRC bot, the 25 minute overview of the kit*, https://web.archive.org/web/20190617034526/http://insecurety.net/hydra-irc-bot-the-25-minute-overview-of-the-kit/, Online; accessed 25 January 2020, 2018.

[118] nenolod, *Network Bluepill - stealth router-based botnet has been DDoSing dronebl for the last couple of weeks*, https://web.archive.org/web/20191223213657/https://www.dronebl.org/blog/8, Online; accessed 25 January 2020, 2009.

[119] P. Čeleda, R. Krejčí, J. Vykopal, and M. Drašar, "Embedded malware - an analysis of the chuck norris botnet," in *European Conference on Computer Network Defense*, 2010.

[120] Carna Bot, *Internet Census 2012*, https://web.archive.org/web/20191226230924/http://census2012.sourceforge.net/paper.html, Online; accessed 25 January 2020, 2012.

[121] unixfreaxjp, *Another story of Unix Trojan: Tsunami/Kaiten.c (IRC/Bot) w/ Flooder, Backdoor at a hacked xBSD*, https://web.archive.org/web/20191022131906/https://blog.malwaremustdie.org/2013/05/story-of-unix-trojan-tsunami-ircbot-w.html, Online; accessed 25 January 2020, 2013.

[122] Symantec, *Linux.Lightaidra*, https://www.symantec.com/security-center/writeup/2014-120115-3009-99, Online; accessed 25 January 2020, 2014.

[123] I. Zeifman, R. Atias, and O. Gayer, *Lax Security Opens the Door for Mass-Scale Abuse of SOHO Routers*, https://web.archive.org/web/20191028220814/https://www.imperva.com/blog/ddos-botnet-soho-router/, Online; accessed 25 January 2020, 2015.

[124] Trend Micro, *Bash Vulnerability (Shellshock) Exploit Emerges in the Wild, Leads to BASHLITE Malware*, https://web.archive.org/web/20181129100545/https://blog.trendmicro.com/trendlabs-security-intelligence/bash-vulnerability-shellshock-

exploit-emerges-in-the-wild-leads-to-flooder/, Online; accessed 25 January 2020, 2014.

[125] unixfreaxjp, *MMD-0021-2014 - Linux/Elknot: China's ELF DDoS+backdoor*, https://web.archive.org/web/20190620160643/http://blog.malwaremustdie.org/2014/05/linux-reversing-is-fun-toying-with-elf.html, Online; accessed 25 January 2020, 2014.

[126] unixfreaxjp, *MMD-0028-2014 - Linux/XOR.DDoS : Fuzzy reversing a new China ELF*, https://web.archive.org/web/20200111215513/https://blog.malwaremustdie.org/2014/09/mmd-0028-2014-fuzzy-reversing-new-china.html, Online; accessed 25 January 2020, 2014.

[127] The White Team, *linux.wifatch*, https://gitlab.com/rav7teif/linux.wifatch, Online; accessed 25 January 2020, 2014.

[128] Johannes, *Linksys Worm ("TheMoon") Captured*, https://web.archive.org/web/20190506033506/https://isc.sans.edu/forums/diary/Linksys+Worm+TheMoon+Captured/17630, Online; accessed 25 January 2020, 2014.

[129] unixfreaxjp, *MMD-0057-2016 - Linux/LuaBot - IoT botnet as service*, https://web.archive.org/web/20191001035222/https://blog.malwaremustdie.org/2016/09/mmd-0057-2016-new-elf-botnet-linuxluabot.html, Online; accessed 25 January 2020, 2016.

[130] M. Malik and M.-E. M. Léveillé, *Meet Remaiten – a Linux bot on steroids targeting routers and potentially other IoT devices*, https://web.archive.org/web/20190921144358/https://www.welivesecurity.com/2016/03/30/meet-remaiten-a-linux-bot-on-steroids-targeting-routers-and-potentially-other-iot-devices/, Online; accessed 25 January 2020, 2016.

[131] unixfreaxjp, *MMD-0059-2016 - Linux/IRCTelnet (new Aidra) - A DDoS botnet aims IoT w/ IPv6 ready*, https://web.archive.org/web/20191001035221/https://blog.malwaremustdie.org/2016/10/mmd-0059-2016-linuxirctelnet-new-ddos.html, Online; accessed 25 January 2020, 2016.

[132] O. Bilodeau and T. Dupuy, "Dissectinglinux/moose," eset, Tech. Rep., 2017.

[133] L. ARSENE, *Hold My Beer Mirai – Spinoff Named 'LiquorBot' Incorporates Cryptomining*, https://web.archive.org/web/20200108154200/https://labs.bitdefender.com/2020/01/hold-my-beer-mirai-spinoff-named-liquorbot-incorporates-cryptomining/, Online; accessed 25 January 2020, 2020.

[134] radware, *"BrickerBot" Results In PDoS Attack*, https://web.archive.org/web/20191226230924/http://census2012.sourceforge.net/paper.html, Online; accessed 25 January 2020, 2017.

[135] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin, "Measurement and analysis of hajime, a peer-to-peer iot botnet.," in *Proc. of the 2019 NDSS*, San Diego, CA, Feb. 2019.

[136] O. **Alrawi**, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *Proc. of the 40th S&P Oakland*, May 2019.

[137] R. Perdisci, T. Papastergiou, O. **Alrawi**, and M. Antonakakis, "Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis," in *Proc. of the 5th EuroS&P*, Sep. 2020.

[138] S. Hilton, *Dyn Analysis Summary Of Friday October 21 Attack*, https://web.archive.org/web/20191211172341/https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/, Online; accessed 25 January 2020.

[139] B. Krebs, *New Mirai Worm Knocks 900K Germans Offline*, https://krebsonsecurity.com/2016/11/new-mirai-worm-knocks-900k-germans-offline/, Online; accessed 25 January 2020, 2016.

[140] L. Constantin, *Armies of hacked IoT devices launch unprecedented DDoS attacks*, https://www.csoonline.com/article/3124344/armies-of-hacked-iot-devices-launch-unprecedented-ddos-attacks.html, Online; accessed 25 January 2020, 2016.

[141] Check Point Research, *Huawei Home Routers in Botnet Recruitment*, https://web.archive.org/web/20200106091208/https://research.checkpoint.com/2017/good-zero-day-skiddie/, Online; accessed 25 January 2020, 2017.

[142] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," in *Proc. of the 39th S&P Oakland*, San Francisco, CA, May 2018.

[143] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: Analysing the rise of iot compromises," in *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*, 2015.

[144] P.-A. Vervier and Y. Shen, "Before toasters rise up: A view into the emerging iot threat landscape," in *Proc. of the 21th RAID*, Crete, Greece, Sep. 2018.

[145] J. Choi, A. Anwar, H. Alasmary, J. Spaulding, D. Nyang, and A. Mohaisen, "Iot malware ecosystem in the wild: A glimpse into analysis and exposures," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019.

[146] F. Dang *et al.*, "Understanding fileless attacks on linux-based iot devices with honeycloud," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019.

[147] Y. Park, D. Reeves, V. Mulukutla, and B. Sundaravel, "Fast malware classification by automated behavioral graph matching," in *Proceedings of the 6th Annual Workshop on Cyber Security and Information Intelligence Research*, 2010.

[148] H. Alasmary *et al.*, "Analyzing and detecting emerging internet of things malware: A graph-based approach," *IEEE Internet of Things Journal*, 2019.

[149] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Ddos-capable iot malwares: Comparative analysis and mirai investigation," *Security and Communication Networks*, 2018.

[150] J. Choi *et al.*, "Honor among thieves: Towards understanding the dynamics and interdependencies in iot botnets," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, 2019.

[151] P. Richter and A. Berger, "Scanning the scanners: Sensing the internet from a massively distributed network telescope," in *Proc. of the 19th ACM SIGCOMM Conference on Internet Measurement (IMC)*, Amsterdam, Netherlands, Nov. 2019.

[152] S. Torabi, E. Bou-Harb, C. Assi, M. Galluscio, A. Boukhtouta, and M. Debbabi, "Inferring, characterizing, and investigating internet-scale malicious iot device activities: A network telescope perspective," in *Proc. of the International Conference on Dependable Systems and Networks (DSN)*, 2018.

[153] O. Çetin *et al.*, "Cleaning up the internet of evil things: Real-world evidence on isp and consumer efforts to remove mirai.," in *Proc. of the 2019 NDSS*, San Diego, CA, Feb. 2019.

[154] tenable, *Nessus Professional*, http://info.tenable.com/rs/934-XQB-568/images/NessusPro__DS__EN_v8.pdf, 2005.

[155] MITRE, *About CVE*, http://cve.mitre.org/about/index.html, 1999.

[156] FIRST, *Common Vulnerability Scoring System SIG*, https://www.first.org/cvss/, 2005.

[157] A. Abraham, *Mobile Security Framework (MobSF)*, https://github.com/MobSF/Mobile-Security-Framework-MobSF/blob/master/README.md, 2016.

[158] Linkedin, *QARK - Quick Android Review Kit*, https://github.com/linkedin/qark/blob/master/README.md, 2016.

[159] *Kryptowire EMM+S*, http://www.kryptowire.com/enterprise.php, 2011.

[160] ntop, *High-Speed Web-based Traffic Analysis and Flow Collection*, https://www.ntop.org/products/traffic-analysis/ntop/, 1998.

[161] G. Combs, *About Wireshark*, https://www.wireshark.org, 1998.

[162] D. Roethlisberger, *SSLsplit - transparent SSL/TLS interception*, https://www.roe.ch/SSLsplit, 2009.

[163] *RASPBERRY PI ZERO*, https://www.raspberrypi.org/products/raspberry-pi-zero/, 2018.

[164] *THE JUNE OVEN*, https://juneoven.com/the-oven, 2018.

[165] *About Let's Encrypt*, https://letsencrypt.org/about/, 2018.

[166] *Unlocking the potential of the internet of things*, http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world, 2015.

[167] A. Mirian *et al.*, "An internet-wide view of ics devices," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, IEEE, 2016.

[168] D. Uhrıcek, *Lisa–multiplatform linux sandbox for analyzing iot malware*, https://excel.fit.vutbr.cz/submissions/2019/058/58.pdf, 2020.

[169] *Detux: The multiplatform linux sandbox*, https://github.com/detuxsandbox/detux, 2016.

[170] S. Yonamine, Y. Taenaka, and Y. Kadobayashi, "Tamer: A sandbox for facilitating and automating iot malware analysis with techniques to elicit malicious behavior.," in *ICISSP*, 2022.

[171] H.-V. Le and Q.-D. Ngo, "V-sandbox for dynamic analysis iot botnet," *IEEE Access*, vol. 8, 2020.

[172] A. Costin and J. Zaddach, "Iot malware: Comprehensive survey, analysis framework and case studies," *BlackHat USA*, 2018.

[173] O. **Alrawi**, *BadThings - Linux-based IoT Malware*.

[174] M. Clive, "2017 Embedded Markets Study: Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments," EETimes/Embedded.com, Tech. Rep., 2017.

[175] EE—Times Embedded, "2019 Embedded Markets Study Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments," EETimes/Embedded.com, Tech. Rep., 2019.

[176] O. **Alrawi**, C. Zuo, R. Duan, R. Pai Kasturi, Z. Lin, and B. Saltaformaggio, "The betrayal at cloud city: An empirical analysis of cloud-based mobile backends," in *Proc. of the 28th USENIX Security*, Aug. 2019.

[177] S. Zhu *et al.*, "Measuring and modeling the label dynamics of online anti-malware engines," in *Proc. of the 29th USENIX Security*, Aug. 2020.

[178] A. Kountouras *et al.*, "Enabling network security through active DNS datasets," in *Proc. of the 19th RAID*, Evry, France, Sep. 2016.

[179] B. P. LLC, *Bad Packets - We provide cyber threat intelligence on emerging threats, IoT botnets, and network abuse*, https://badpackets.net/, Online; accessed 25 January 2020.

[180] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proc. of the 2019 NDSS*, San Diego, CA, Feb. 2019.

[181] U. Bayer, C. Kruegel, and E. Kirda, "TTAnalyze: A Tool for Analyzing Malware," in *15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR).*, 2006.

[182] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in *Proc. of the DIMVA*, Jul. 2008.

[183] C. Lever, P. Kotzias, D. Balzarotti, J. Caballero, and M. Antonakakis, "A lustrum of malware network communication: Evolution and insights," in *Proc. of the 38th S&P Oakland*, San Jose, CA, May 2017.

[184] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," in *Proc. of the 19th RAID*, Evry, France, Sep. 2016.

[185] National Institute of Standards and Technology, *NATIONAL VULNERABILITY DATABASE*, https://nvd.nist.gov, 2019.

[186] Zeropoint Dynamics, *Zelos: A comprehensive binary emulation and instrumentation platform*, https://github.com/zeropointdynamics/zelos, Online; accessed 30 September 2020.

[187] threatland, *TL-BOTS/TL.MIRAI*, https://github.com/threatland/TL-BOTS/tree/master/TL.MIRAI, Online; accessed 25 January 2020.

[188] A. Mohaisen, O. **Alrawi**, and M. Mohaisen, "Amal: High-fidelity, behavior-based automated malware analysis and classification," *Elsevier Computers & Security*, 2015.

[189] M. F. Oberhumer, L. Molnár, and J. F. Reiser, *UPX: the Ultimate Packer for eXecutables*, https://www.unicorn-engine.org/, Online; accessed 25 January 2020.

[190] Buildroot, *Buildroot: Making Embedded Linux Easy*, https://buildroot.org/, Online; accessed 25 January 2020.

[191] QEMU, *QEMU: the FAST! processor emulator*, https://www.qemu.org/, Online; accessed 25 January 2020.

[192] A. Mohaisen, O. Alrawi, M. Larson, and D. McPherson, "Towards a methodical evaluation of antivirus scans and labels," in *International Workshop on Information Security Applications*, 2013.

[193] A. Mohaisen and O. **Alrawi**, "Av-meter: An evaluation of antivirus scans and labels," in *Proc. of the DIMVA*, London, UK, Jun. 2014.

[194] W. Largent, *New vpnfilter malware targets at least 500k networking devices worldwide*, http://blog.talosintelligence.com/2018/05/VPNFilter.html, May 2018.

[195] *New mirai variant targets enterprise wireless presentation & display systems*, https://unit42.paloaltonetworks.com/new-mirai-variant-targets-enterprise-wireless-presentation-display-systems/, Mar. 2019.

[196] *Muhstik botnet exploits the latest weblogic vulnerability for cryptomining and ddos attacks*, https://unit42.paloaltonetworks.com/muhstik-botnet-exploits-the-latest-weblogic-vulnerability-for-cryptomining-and-ddos-attacks/, Apr. 2019.

[197] *New mirai variant adds 8 new exploits, targets additional iot devices*, https://unit42.paloaltonetworks.com/new-mirai-variant-adds-8-new-exploits-targets-additional-iot-devices/, Jun. 2019.

[198] *Hide 'n seek botnet updates arsenal with exploits against nexus repository manager & thinkphp*, https://unit42.paloaltonetworks.com/hide-n-seek-botnet-updates-arsenal-with-exploits-against-nexus-repository-manager-thinkphp/, Jun. 2019.

[199] *Mirai variant echobot resurfaces with 13 previously unexploited vulnerabilities*, https://unit42.paloaltonetworks.com/mirai-variant-echobot-resurfaces-with-13-previously-unexploited-vulnerabilities/, Dec. 2019.

[200] L. Cashdollar, *Latest echobot: 26 infection vectors*, https://blogs.akamai.com/sitr/2019/06/latest-echobot-26-infection-vectors.html, Jun. 2019.

[201] J. v. D. Wiel, V. Diaz, Y. Namestnikov, and K. Zykov, *Hajime, the mysterious evolving botnet*, https://securelist.com/hajime-the-mysterious-evolving-botnet/78160/, Apr. 2017.

[202] A. Team, *Realtek sdk exploits on the rise from egypt*, https://www.netscout.com/blog/asert/realtek-sdk-exploits-rise-egypt, May 2019.

[203] lennarthaagsma, *Recent vulnerability in eir d1000 router used to spread updated version of mirai ddos bot*, https://blog.fox-it.com/2016/11/28/recent-vulnerability-in-eir-d1000-router-used-to-spread-updated-version-of-mirai-ddos-bot/, Nov. 2016.

[204] *Huawei router exploit involved in satori and brickerbot given away for free on christmas*, https://blog.newskysecurity.com/huawei-router-exploit-involved-in-satori-and-brickerbot-given-away-for-free-on-christmas-by-ac52fe5e4516, Apr. 2018.

[205] *Cve-2018–10561 dasan gpon exploit weaponized in omni and muhstik botnets*, https://blog.newskysecurity.com/cve-2018-10561-dasan-gpon-exploit-weaponized-in-omni-and-muhstik-botnets-ad7b1f89cff3, May 2018.

[206] R. J. Yang and Kenny, *A wicked family of bots*, https://www.fortinet.com/blog/threat-research/a-wicked-family-of-bots.html, May 2018.

[207] https://blog.netlab.360.com/iot_reaper-a-rappid-spreading-new-iot-botnet-en/, Oct. 2017.

[208] *Early warning: A new mirai variant is spreading quickly on port 23 and 2323*, https://blog.netlab.360.com/early-warning-a-new-mirai-variant-is-spreading-quickly-on-port-23-and-2323-en/, Jun. 2018.

[209] *Multi-exploit iot/linux botnets mirai and gafgyt target apache struts*, https://unit42.paloaltonetworks.com/unit42-multi-exploit-iotlinux-botnets-mirai-gafgyt-target-apache-struts-sonicwall/, Sep. 2018.

[210] *Xbash combines botnet, ransomware, coinmining in worm that targets linux and windows*, https://unit42.paloaltonetworks.com/unit42-xbash-combines-botnet-ransomware-coinmining-worm-targets-linux-windows/, Sep. 2018.

[211] M. S. Lenders, C. Amsüss, C. Gündoğan, T. C. Schmidt, and M. Wählisch, "DNS over CoAP (DoC)," RFC Editor, RFC Draft, Jul. 2022.

[212] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "Homesnitch: Behavior transparency and control for smart home iot devices," in

Proceedings of the 12th conference on security and privacy in wireless and mobile networks, 2019, pp. 128–138.

[213] A. Halterman, "Storming the kasa? security analysis of tp-link kasa smart home devices," *Creat. Compon*, 2019.

[214] S. Goutam, W. Enck, and B. Reaves, "Hestia: Simple least privilege network policies for smart homes," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, 2019, pp. 215–220.

[215] P. Emami-Naeini, Y. Agarwal, L. F. Cranor, and H. Hibshi, "Ask the experts: What should be on an iot privacy and security label?" In *2020 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2020, pp. 447–464.

[216] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home devices," in *Proc. of the 2020 NDSS*, San Diego, CA, Feb. 2020.

[217] V. Melnyk, P. Haleta, and N. Golphamid, "Machine learning based network traffic classification approach for internet of things devices," *Theoretical and Applied Cybersecurity*, vol. 2, no. 1, 2020.

[218] A. N. d. P. T. Gurgo *et al.*, "Automated standard based security assessment for iot," 2020.

[219] F. Yin, L. Yang, Y. Wang, and J. Dai, "Iot etei: End-to-end iot device identification method," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, IEEE, 2021, pp. 1–8.

[220] Y. Xiao and M. Varvello, "Fiat: Frictionless authentication of iot traffic," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 483–484.

[221] C. Duan, H. Gao, G. Song, J. Yang, and Z. Wang, "Byteiot: A practical iot device identification system based on packet length distribution," *IEEE Transactions on Network and Service Management*, 2021.

[222] B. Delinchant and J. Ferrari, "Standards and technologies from building sector, iot, and open-source trends," in *Towards Energy Smart Homes*, Springer, 2021, pp. 49–111.

[223] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, and G. M. Pérez, "A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1048–1077, 2021.

[224] C. Duan, S. Zhang, J. Yang, Z. Wang, Y. Yang, and J. Li, "Pinball: Universal and robust signature extraction for smart home devices," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2021, pp. 1–9.

[225] *How wirecutter vets the security and privacy of smart home devices*, Sep. 2020.

[226] A. Piore, *We're surrounded by billions of internet-connected devices. can we trust them?* Oct. 2019.

[227] *Amazon sidewalk will share your internet with strangers. it's not as scary as it sounds.* Jun. 2021.

[228] *Anker's eufy lied to us about the security of its security cameras*, Dec. 2022.