The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle

Omar Alrawi[⋆], Charles Lever[⋆], Kevin Valakuzhy[⋆], Ryan Court[♢],
Kevin Snow[♢], Fabian Monrose[†], Manos Antonakakis[⋆]

[⋆] Georgia Institute of Technology
{alrawi, chazlever, kevinv, manos}@gatech.edu
[♢] Zero Point Dynamics
{rccourt, kevin}@zeropointdynamics.com
[†] University of North Carolina at Chapel Hill
fabian@cs.unc.edu

Abstract

Our current defenses against IoT malware may not be adequate to remediate an IoT malware attack similar to the Mirai botnet. This work seeks to investigate this matter by systematically and empirically studying the lifecycle of IoT malware and comparing it with traditional malware that target desktop and mobile platforms. We present a large-scale measurement of more than 166K Linux-based IoT malware samples collected over a year. We compare our results with prior works by systematizing desktop and mobile malware studies into a novel framework and answering key questions about defense readiness. Based on our findings, we deduce that the required technology to defend against IoT malware is available, but we conclude that there are insufficient efforts in place to deal with a large-scale IoT malware infection breakout.

1 Introduction

The Mirai botnet set a record for the largest distributed denial of service (DDoS) attack and drew the attention of many security professionals [1]. In the aftermath of the attack, many new developments have shaped the IoT malware ecosystem. Therefore, studying the threat lifecycle for IoT malware is vital for securing IoT devices. For example, the Mirai botnet infected devices by using default usernames and passwords, but current IoT malware variants target unpatched vulnerabilities. We seek to study how the emerging IoT malware ecosystem has evolved since Mirai and whether current defenses for traditional malware can protect against it.

To investigate this matter, we need to systematically understand how IoT malware infect systems, deploy payloads, persist on systems, abuse resources, and operate their infrastructure. We guide our analysis by answering two *research questions (RQ)*, namely RQ1: *How is IoT malware different than traditional malware?* and RQ2: *Are current anti-malware techniques effective against IoT malware?* To answer RQ1, we compare the IoT malware lifecycle with traditional malware and highlight the similarities and differences. For RQ2, we qualitatively evaluate how traditional anti-malware techniques work and judge their efficacy based on empirical observations from the IoT malware ecosystem.

Answering RQ1 allows the security community to understand the evolutionary trend of IoT malware and respond accordingly. For example, how do malware adapt to infect and persist on IoT devices? Are there trends that can allow us to better predict the impact of IoT malware on future IoT technologies? Compared to desktop and mobile malware, are IoT malware capabilities bound by the device's resources? How does the IoT malware ecosystem impact different stakeholders? Furthermore, RQ2 allows the security community to gauge if there are sufficient defensive techniques to counter a fast-evolving IoT threat.

To date, there have been several efforts to investigate IoT malware [1]–[8]. However, these efforts either focus on in-depth analysis of a single malware family or rely on small malware corpora collected over short periods. Nonetheless, these efforts provide a fascinating glimpse into the IoT threat landscape and demonstrate the need for additional research. Moreover, current threat frameworks are either too complex with a focus on traditional malware, such as the *MITRE ATT&CK* [9] framework, or study only the infection stage of IoT malware [10], [11]. Our work seeks to address these gaps with a comprehensive evaluation of the IoT malware lifecycle. We guide our study by a principled framework that characterizes various stages of an IoT malware's lifecycle, and we compare our findings with traditional malware.

Our work makes four contributions. First, we propose a novel *analysis framework* that captures the threat lifecycle of IoT malware, which considers the infection vectors, payload properties, persistence methods, capabilities, and *C&C* infrastructure. Second, we use our framework to *systematize* 25 papers that study traditional malware. Third, we *characterize IoT malware* by examining more than 166K samples spanning 6 different system architectures collected over a year. Fourth, we make available the largest and most comprehensive IoT malware corpus to date and include their analysis artifacts, which can be found at https://badthings.info.

Our results show that IoT malware differs from traditional malware in a few key areas including infection vectors and *C&C* communication. We find signature-based detection lacks support and coverage for many IoT malware variants and that at least 15% of new variants utilize packing to evade detection. Additionally, IoT malware uses various persistent methods to overcome read-only file systems found in IoT devices by reusing vendor-specific tools. We find a large array of capabilities that have been incorporated into IoT malware such as proxy services, device bricking, and information theft. We observe that the current IoT malware ecosystem has not reached its full potential but may become a severe threat due to the sheer number of IoT devices coming online. We conclude with a set of recommendations for different stakeholders including device owners, device vendors, and ISP operators.

2 Background and Related Work

Malware targeting embedded Linux-based systems was first reported in 2008 with the discovery of the Hydra IRC bot [12]. Since then, several other bots have entered the scene with various capabilities. Such bots include psyb0t [13], Chuck Norris [14], Carna [15], Tsunami [16], Aidra [17], Dofloo [18], Gafgyt [19], Elknot [20], XOR.DDoS [21], Wifatch [22], TheMoon [23], LUABot [24], Remaiten [25], NewAidra [26], and Moose [27]. Each family had different purposes such as credential theft [27], cryptocurrency mining [28], device destruction [29], internet-wide scanning [15], and cleaning up infected devices [2], [22]. 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 [30], [31]. The Mirai botnet took down critical DNS infrastructure [32], disconnected over 900K internet subscribers [33], and attacked a large cloud service provider [34]. Soon after the release of Mirai's code, many variants began to surface with enhancement to its infection vector, payload obfuscation, and command-and-control (*C&C*) communication. For example, Satori [35], 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. [36] 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 [2]. More comprehensive studies examine individual components of the IoT malware lifecycle. For example, several works [3], [5], [10], [37] examine IoT malware infection tactics and the payload properties. Other works [6], [38] look at how to detect IoT malware by studying its binary static structural features. De Donno et al. [11] organize IoT malware attack capabilities into a taxonomy while Choi et al. [4] study the role that *C&C* infrastructure plays in the lifecycle of IoT malware.

Additional efforts [39], [40] investigate scanners on the internet to identify if they are infected by IoT malware. Finally, Çetin et al. [41] 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.

3 Framework and Methodology

Next, we describe the data sources, methodology, and the framework that we use for the comparative analysis. We define each component's subcategories and present a summary of our results in Table 1. Appendix A presents an extended analysis of desktop and mobile malware from prior works.

3.1 Comparative Framework

Our framework looks at five components for malware's lifecycle. We study the *infection* vector, the *payload* properties, the *persistence* methods, the *capabilities*, and the *C&C infrastructure*. For each component, we identify techniques discussed in the literature for traditional malware (desktop/mobile) and empirically measure it for IoT malware. The following defines each component:

- **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.

We study 25 papers from prior works to qualitatively derive subcategories under each component, which are in Appendix A. For example, we cite the work of Holz et al. [42] to support the use of drive-by downloads in desktop malware and their distribution networks. Moreover, we use the *MITRE ATT&CK* taxonomy to derive additional subcategories that are not found in prior work but are documented by security companies. Table 1 summarizes the comparative analysis.

Table 1: An overview of the results from our findings comparing desktop, mobile, and IoT malware using the proposed framework.

Components		Summary			
	Categories	Desktop	Mobile	IoT	**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.

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

3.2 Data Sources

We list all the dataset sources for our measurements in Table 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 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 *x*86 or *x*86_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 this 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) [43] and utilize Reduced Instruction Set Computers (RISC) architecture [44], whereas *x86* and *x86_64* are based on Complex Instruction Set Computers (CISC) architecture that are mostly found in servers, desktops, and laptops. We exclude x86, x86_64, and Android malware because (1) they are well covered in prior works [45]–[49], (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 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 engine 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 [50]. First, we collect files with

[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

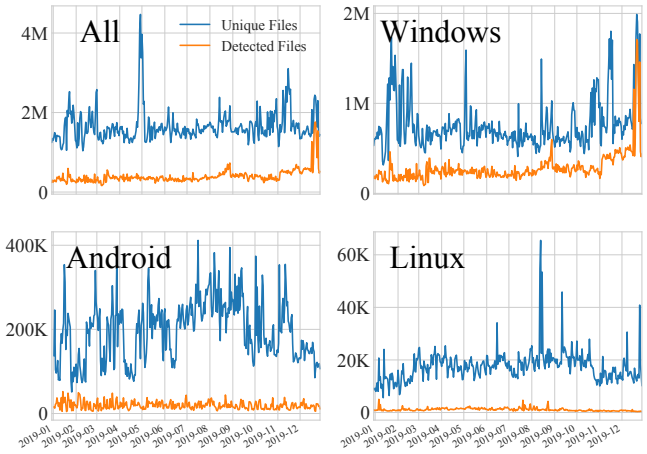


Figure 1: The daily volume of files submitted to VT exceeds 1.5M. Our access is limited to 10K files per day.

one AV detection to stay under the daily access quota (10K/-day, see Figure 1). Second, we filter files with less than five AV detections to suppress false-positives, which are common in VT [50]. 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 [51], 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 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 [52] 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 [53] 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.

Table 2: List of data sources used for empirical measurement.

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

3.3 Analysis Methods

Figure 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 [36], [54]–[56] 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 [57] 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

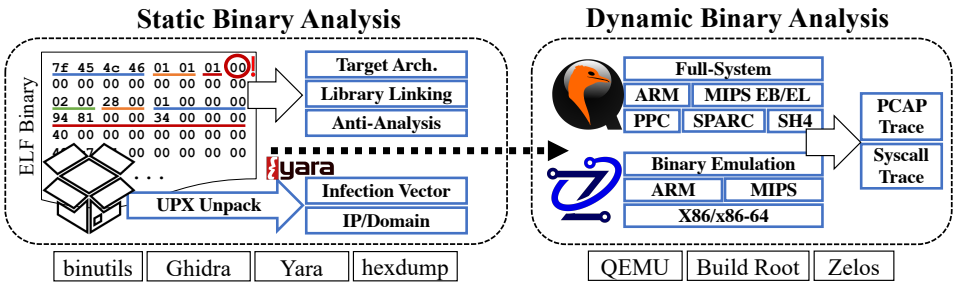


Figure 2: Overview of analysis methodology.

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 [58], 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* [59] in Figure 2. The run time of a sample influences the observed behavior as documented in prior works [60]. 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 publicly available source code from various IoT malware found online [61] 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 [54], [55], [62]. Nevertheless, we report the results in Table 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 [53] 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 [56]. 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.

4 Measurement 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)** to help answer our research questions (RQ1 and RQ2).

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 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 [63]. For dynamic analysis, we use Buildroot [64] and QEMU [65] for full-system analysis and Zelos [59] 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 due to legacy code incompatibility, therefore, we only considered the M68K samples for static analysis.

Table 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.

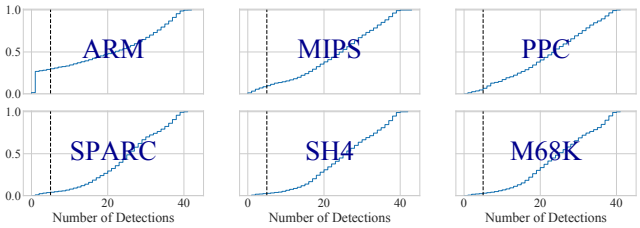


Figure 3: A CDF graph showing the number of AV detection per RISC architectures considered for the IoT malware study and a horizontal dotted line at value of five AV engines.

4.1 Detection and Labeling

In Table 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 [50]. However, Figure 3 suggests that traditional AV engines 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 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. [2] 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 [66], [67].

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.

4.2 Infection Analysis

We observe that IoT malware use remote exploitation and default credentials to infect devices. We present a timeline in Figure 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 [68]–[84]. We observe that the number of exploits

Table 3: An overview of the dataset statistics for the metadata analysis, static analysis, and dynamic analysis by 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	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 4: Top AV labels given to the collected samples by VirusTotal and AVClass.

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	- -	

Table 5: Summary of the top device categories and top vulnerabilities within each category targeted by IoT malware.

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)

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 5 we present results from the Bad Packets LLC [52] honeypot. 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 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 Section 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.

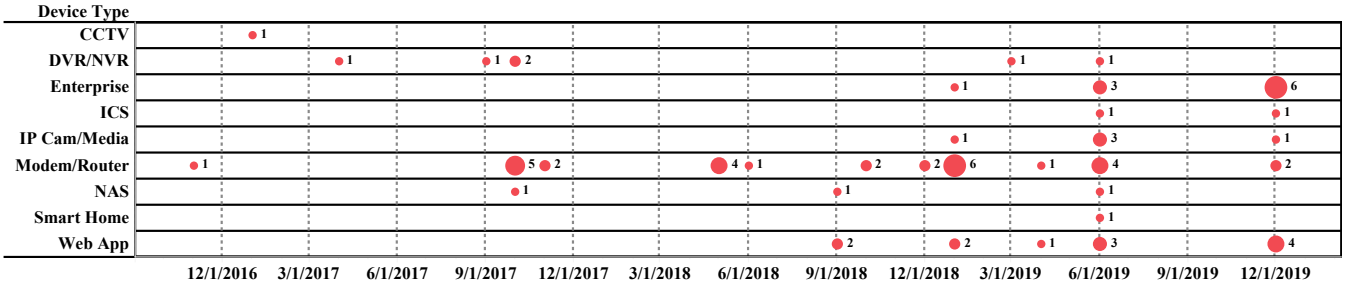


Figure 4: A timeline of exploits for nine device types found in Mirai variants as reported by the security researchers.

Table 6: Categorization of exploits in use by IoT malware. The vast majority of exploits target internet-facing devices via command injection (CMD Inject). There are exceptions such as media devices, hubs, and smart home devices.

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

4.3 Payload Analysis

We observe that IoT malware payloads use packing, environment keying, scripting, and cross-architecture binaries. Table 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.

4.4 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-

Table 7: List of observed scanning and exploit attempts from 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

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 1).* 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" to */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.

4.5 Capability Analysis

Initial variants of IoT malware discussed in Section 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, 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 7 is a summary of the observed scanning and exploitation attempts, which includes a subset of the vulnerabilities found in Table 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 8 presents a list of the DDoS functions found in IoT malware.

Table 8: List of DDoS attacks found through static analysis by correlating function symbol names and public malware 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

In addition to these capabilities, 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. Lastly, we observe IoT malware 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 hardcoding an IP address to a specific mining pool server. We also observe attempts to set

up a proxy service that configures network traffic forwarding on high ports (i.e. *44781* and *57775*).

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.

4.6 C&C Analysis

We observe from dynamic and static analysis that IoT malware can use P2P and centralized infrastructure for *C&C* communication. For example, Hajime [2] 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 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 9 presents the top six malware families based on the infrastructure analysis described in Section 3. 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 5a and Figure 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 9: Top IoT malware labels ranked by client IDs.

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 make IoT botnets less resilient to take downs. IoT malware network activities can be difficult to measure on the internet using DNS since very few samples rely on DNS.

5 In-Depth Case Studies

Motivated by our empirical results in Section 4, we take a closer look at how IoT malware reuses Mirai's code to provide more insightful answers to our research questions.

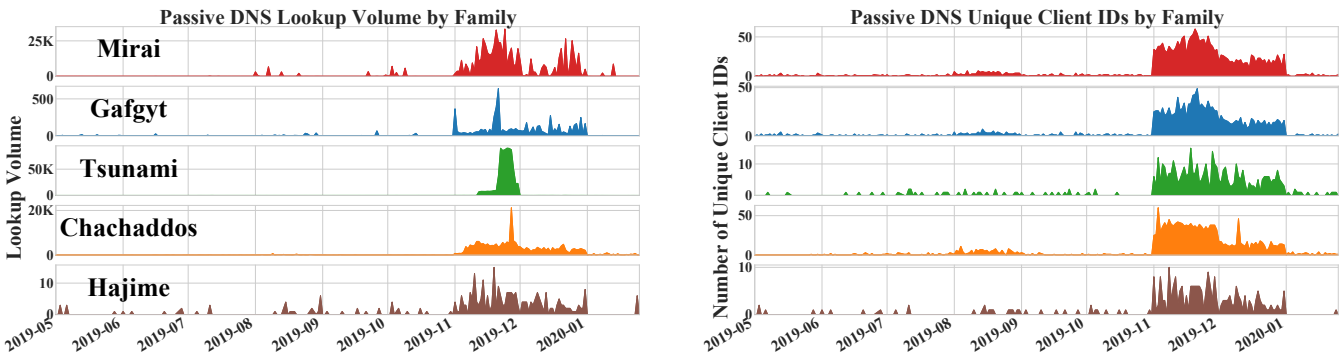
5.1 Code Reuse and Evolution

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 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 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 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



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

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

Figure 5: DNS measurement of domains for the top IoT malware family clusters as seen from our pDNS dataset.

```
// 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 6: Original Mirai's buggy code (top) and evolved fixed code (bottom).

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 [61] 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.

5.2 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 4.6. 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 4.2 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.

6 Summary and Discussion

Recall, RQ1 seeks to identify the similarities and differences between desktop, mobile, and IoT malware, while RQ2 seeks to qualitatively assess current defensive techniques against IoT malware.

6.1 RQ1: 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 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 [42] 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 Section 5, 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).

6.2 RQ2: 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 [41]. 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 interactions, 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. [41]. 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.

7 Conclusion

This work provides a large-scale empirical measurement of the current IoT malware threat landscape. By analyzing over 166K Linux-based IoT malware, we uncovered important insights that compare and contrast traditional desktop and mobile malware to IoT malware. We find that IoT malware evolution follows a similar lifecycle trend to traditional malware by using exploits for infection, packing its payload to avoid detection, using specialized capabilities based on device resources, and leveraging P2P and centralized infrastructure for *C&C* call-back. We speculate that IoT malware will be a much more serious threat because of the number of new IoT devices that come online and the unrealized potential of IoT malware development. Based on our findings, we believe that the required technology to defend against IoT malware is available. However, we do not think there are sufficient preparation efforts to proactively deal with a large-scale breakout. In effort to support this ongoing research in the IoT malware space, we release the largest IoT malware corpora to date and make our tools, analysis artifacts, and results available at: https://badthings.info.

8 Acknowledgment

We thank the anonymous reviewers for their help in improving this work. We thank Bad Packets LLC for sharing their data. This work is supported in part by the US Department of Defense grant no. FA8750-17-C-0016. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Department of Defense (DoD).

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *Proceedings of the 26th USENIX Security Symposium (Security)*, Vancouver, BC, Canada, Aug. 2017.

[2] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin, "Measurement and analysis of hajime, a peer-to-peer iot botnet," in *Proceedings of the 2019 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, 2019.

[3] 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.

[4] J. Choi, A. Abusnaina, A. Anwar, A. Wang, S. Chen, D. Nyang, and A. Mohaisen, "Honor among thieves: Towards understanding the dynamics and interdependencies in iot botnets," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, 2019.

[5] P.-A. Vervier and Y. Shen, "Before toasters rise up: A view into the emerging iot threat landscape," in *Proceedings of the 21st International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Crete, Greece, Sep. 2018.

[6] H. Alasmary, A. Khormali, A. Anwar, J. Park, J. Choi, A. Abusnaina, A. Awad, D. Nyang, and A. Mohaisen, "Analyzing and detecting emerging internet of things malware: A graph-based approach," *IEEE Internet of Things Journal*, 2019.

[7] H. Alasmary, A. Anwar, J. Park, J. Choi, D. Nyang, and A. Mohaisen, "Graph-based comparison of iot and android malware," in *International Conference on Computational Social Networks*, 2018.

[8] A. Anwar, H. Alasmary, J. Park, A. Wang, S. Chen, and D. Mohaisen, "Statically dissecting internet of things malware: Analysis, characterization, and detection," in *International Conference on Information and Communications Security*, 2020.

[9] The MITRE Corporation, *MITRE ATT&CK*, https://attack.mitre.org/, Online; accessed 25 January 2020.

[10] F. Dang, Z. Li, Y. Liu, E. Zhai, Q. A. Chen, T. Xu, Y. Chen, and J. Yang, "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.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] O. Bilodeau and T. Dupuy, "Dissectinglinux/moose," eset, Tech. Rep., 2017. [Online]. Available: https://web.archive.org/web/20191228043619/https://www.welivesecurity.com/wp-content/uploads/2015/05/Dissecting-LinuxMoose.pdf.

[28] 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.

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

[30] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *Proceedings of the 40th Symposium on Security and Privacy (Oakland)*, San Francisco, CA, May 2019.

[31] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," in *Proceedings of the 39th Symposium on Security and Privacy (Oakland)*, San Francisco, CA, May 2018.

[37] 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.

[38] 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.

[39] P. Richter and A. Berger, "Scanning the scanners: Sensing the internet from a massively distributed network telescope," in *Proceedings of the 19th Internet Measurement Conference (IMC)*, 2019.

[40] 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 *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2018.

[41] O. Çetin, C. Ganán, L. Altena, T. Kasama, D. Inoue, K. Tamiya, Y. Tie, K. Yoshioka, and M. van Eeten, "Cleaning up the internet of evil things: Real-world evidence on isp and consumer efforts to remove mirai," in *Proceedings of the 2019 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2019.

[42] 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.

[43] M. Clive, "2017 Embedded Markets Study: Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments," EETimes/Embedded.com, Tech. Rep., 2017. [Online]. Available: https://web.archive.org/web/20191010030447/https://m.eet.com/media/1246048/2017-embedded-market-study.pdf.

[44] EE|Times Embedded, "2019 Embedded Markets Study Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments," EETimes/Embedded.com, Tech. Rep., 2019. [Online]. Available: http://web-old.archive.org/web/20200731183039/https://www.embedded.com/wp-content/uploads/2019/11/EETimes_Embedded_2019_Embedded_Markets_Study.pdf.

[45] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2008.

[46] P. Kotzias, L. Bilge, P.-A. Vervier, and J. Caballero, "Mind your own business: A longitudinal study of threats and vulnerabilities in enterprises," in *Proceedings of the 2019 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2019.

[47] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proceedings of the 33rd Symposium on Security and Privacy (Oakland)*, San Francisco, CA, May 2012.

[48] 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.

[49] O. Alrawi, C. Zuo, R. Duan, R. P. Kasturi, Z. Lin, and B. Saltaformaggio, "The betrayal at cloud city: An empirical analysis of cloud-based mobile backends," in *Proceedings of the 28th USENIX Security Symposium (Security)*, Santa Clara, CA, Aug. 2019.

[50] S. Zhu, J. Shi, L. Yang, B. Qin, Z. Zhang, L. Song, and G. Wang, "Measuring and modeling the label dynamics of online anti-malware engines," in *Proceedings of the 29th USENIX Security Symposium (Security)*, Boston, MA, Aug. 2020.

[51] A. Kountouras, P. Kintis, C. Lever, Y. Chen, Y. Nadji, D. Dagon, M. Antonakakis, and R. Joffe, "Enabling network security through active DNS datasets," in *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Evry, France, Sep. 2016.

[52] 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.

[53] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings of the 2019 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2019.

[54] 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.

[55] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in *Proceedings of the 5th Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2008.

[56] C. Lever, P. Kotzias, D. Balzarotti, J. Caballero, and M. Antonakakis, "A lustrum of malware network communication: Evolution and insights," in *Proceedings of the 38th Symposium on Security and Privacy (Oakland)*, San Jose, CA, May 2017.

[57] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," in *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Evry, France, Sep. 2016.

[58] National Institute of Standards and Technology (NIST), *NATIONAL VULNERABILITY DATABASE*, https://nvd.nist.gov/, Online; accessed 25 January 2020.

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

[60] M. Lindorfer, C. Kolbitsch, and P. M. Comparetti, "Detecting environment-sensitive malware," in *Proceedings of the 14th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Menlo Park, CA, Sep. 2011.

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

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

[63] 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.

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

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

[66] 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.

[67] A. Mohaisen and O. Alrawi, "Av-meter: An evaluation of antivirus scans and labels," in *Proceedings of the 11th Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2014.

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

[69] 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.

[70] *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.

[71] *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.

[72] 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.

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

[74] *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.

[75] *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.

[76] *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.

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

[78] *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.

[79] *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.

[80] *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.

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

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

[83] 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.

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

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

[86] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna, "Polymorphic worm detection using structural information of executables," in *Proceedings of the 8th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Seattle, Washington, Sep. 2005.

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

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

[89] 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.

[90] D. Dagon, G. Gu, C. P. Lee, and W. Lee, "A taxonomy of botnet structures," in *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC)*, 2007.

[91] 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.

[92] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage, "Spamalytics: An empirical analysis of spam marketing conversion," in *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, VA, Oct. 2008.

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

[94] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, Chicago, Illinois, Nov. 2009.

[95] L. Lu, V. Yegneswaran, P. Porras, and W. Lee, "Blade: An attack-agnostic approach for preventing drive-by malware infections," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, Chicago, Illinois, Oct. 2010.

[96] 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.

[97] S. Shin, R. Lin, and G. Gu, "Cross-analysis of botnet victims: New insights and implications," in *Proceedings of the 14th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Menlo Park, CA, Sep. 2011.

[98] C. Rossow, C. Dietrich, and H. Bos, "Large-scale analysis of malware downloaders," in *Proceedings of the 9th Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2012.

[99] L. Invernizzi, S. Miskovic, R. Torres, C. Kruegel, S. Saha, G. Vigna, S.-J. Lee, and M. Mellia, "Nazca: Detecting malware distribution in large-scale networks," in *Proceedings of the 2014 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2014.

[100] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraş, "The dropper effect: Insights into malware distribution with downloader graph analytics," in *Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS)*, Denver, Colorado, Oct. 2015.

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

[102] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee, "The core of the matter: Analyzing malicious traffic in cellular carriers," in *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2013.

[103] 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.

[104] 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.

[105] 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.

[106] 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.

[107] 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.

[108] 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.

Appendices

A Detailed Comparative Analysis

This section provides an extended analysis of malware threats for desktop and mobile platforms to compare with IoT malware. Table 10 summarizes our extended systematization of 25 prior studies on traditional malware.

A.1 Infection Comparison

Desktop Infection Vectors. In Table 10, we see desktop malware pioneered many of the infection techniques. Moore et al. [85] 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 [104]–[106]. Desktop malware rely more on infection vectors like drive-by download and phishing. Provos et al [93] present an extensive study on drive-by downloads, and several prior works measure [42], [93], [97], [100] and propose defenses [94], [95], [99] for them.

For phishing, Abu Rajab et al. [87] present a multi-dimensional measurement into botnets. Their work documents how botnets leverage phishing emails for spreading. Holz et al. [42] and Kotzias et al. [46] 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 (pirated software).

Mobile Infection Vectors. Similar to our study, Zhou et al. [47] look at Android mobile malware and characterize the

Table 10: A comparison between desktop, mobile, and IoT malware using the proposed framework.

Components		Summary			Desktop																				Mobile				IoT	
	Categories	Desktop	Mobile	IoT	Moore03 [85]	Kruegel05 [86]	AbuRa06 [87]	Barfo07 [88]	AbuRa07 [89]	Dagon07 [90]	Holz08 [45]	Polyc08 [91]	Kanic08 [92]	Holz08 [42]	Provo08 [93]	Stone09 [94]	Lu10 [95]	Cho10 [96]	Lindo11 [60]	Shin11 [97]	Rosso12 [98]	Inver14 [99]	Kwon15 [100]	Ganan15 [101]	Kotzi19 [46]	Zhou12 [47]	Lever13 [102]	Lindo14 [48]	Tam17 [103]		
Infection	Remote Exploit	✓		✓	✓			✓				✓							✓					✓					✓	
	Repackaging	✓*	✓																						✓					Sec. 4.2
	Drive-by	✓	✓										✓	✓	✓	✓			✓		✓	✓			✓					
	Phishing	✓	✓				✓						✓											✓	✓					
	Default Cred.	✓*		✓																									✓	
	Rem. Media	✓*	✓																								✓			
Payload	Packing	✓	✓	✓		✓		✓					✓								✓	✓				✓		✓	✓	
	Env. Keying	✓	✓	✓														✓								✓			✓	Sec. 4.3
	Scripting	✓*		✓																									✓	
	Cross-Arch/Plat.	✓*	✓	✓																							✓		✓	
Persist.	Firmware	✓		✓											✓														✓	
	OS - Kernel	✓	✓	+							✓		✓		✓										✓					Sec. 4.4
	OS - User	✓	✓	+										✓													✓	✓		
Capability	Priv. Escalation	✓	✓	✓							✓		✓												✓					
	Defense Evasion	✓	✓	✓			✓	✓						✓				✓		✓					✓		✓	✓	✓	
	Info. Theft	✓	✓	✓			✓	✓			✓				✓								✓		✓		✓		✓	Sec. 4.5
	Scanning	✓		✓	✓		✓	✓			✓								✓					✓					✓	
	DDoS	✓		✓			✓	✓							✓														✓	
	Destruction	✓	✓	✓																				✓			✓		✓	
	Resource Abuse	✓	✓	✓			✓				✓	✓			✓		✓							✓	✓		✓		✓	
C&C	Peer-2-Peer	✓		✓					✓			✓	✓							✓									✓	Sec. 4.6
	Email/SMS	✓	✓								✓																✓			
	Centralized	✓	✓	✓			✓	✓		✓	✓	✓		✓	✓		✓			✓	✓	✓	✓			✓	✓	✓		✓	

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

infection techniques. Their work shows that many Android malware use repackaging, drive-by download, and phishing to propagate as shown in Table 10. Lindorfer et al. [48] 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 users to visit a link to get infected.

A.2 Payload Comparison

Desktop Payload Properties. In Table 10, we see that all the payload categories apply to desktop malware. Kruegel et al. [86] predicted the rise of polymorphic payloads and proposed a way to detect them offline. Later, Barford et al. [88] 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. [42] 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. [98] 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 have to adopt the use of polymorphism to successfully evade detection.

On the defense side, Invernizzi et al. [99] 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 [60], [107] and scripting [108] 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 [48]. 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. [47] 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 [48] 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.

A.3 Persistence Comparison

Desktop Malware Persistence. Table 10 shows that desktop malware use all levels of persistence. Provos et al. [93] and Polychronakis et al. [91] identify bots that persist through user-space and kernel modules, respectively. Additionally, Stone-Gross et al. [94] 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. Desktop 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 [47], [48], [103] 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 provide 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.

A.4 Capability Comparison

Desktop Malware Capability. In Table 10 we find that desktop malware exhibit all of the listed capabilities. Moore et al. [85] document the capabilities in the Slammer worm, which other botnets also borrow [46], [87], [88], [91], [97]. Several works [92], [94], [96], [101] 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 [46] and DDoS attacks [87] for hire.

Another aspect of desktop malware capabilities is the fact that it can escalate privileges [91] by exploitation or key-logging, and they can evade detection by disabling security tools [60], [98]. 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 systems. 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 10 shows that mobile malware has the same abusive capabilities as desktop malware with the exception of scanning and DDoS attacks. Zhou et al. [47] 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. [48] 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.

A.5 Command & Control Comparison

Desktop Malware C&C. Table 10 shows that desktop malware use all of the listed methods for *C&C* communication. Polychronakis et al. [91] show that desktop malware rely on email for *C&C* call-back. Moreover, Kanich et al. [92] and Holz et al. [42] study the Storm botnet P2P network to analyze the spam campaigns and estimate the botnet size. They identify a complex layered infrastructure of 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. [98] 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) [94], [99], multi-tier centralized topology [96], fast-flux [45], and bullet-proof or hacked [101] 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. [45] note content delivery networks (CDNs) and round-robin DNS (RRDNS) provide resilience to legitimate internet applications, which desktop malware mimics by using fast-flux services.

Mobile Malware C&C. Lindorfer et al. [48] 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 [47], [48], [102] identify that mobile malware does not use the same sophistication for *C&C* call-back found in desktop malware. Furthermore, Lever et al. [102] 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.