# Exploring Shifting Patterns in Recent IoT Malware

Javier Carrillo-Mondéjar[1], Guillermo Suarez-Tangil[2], Andrei Costin[3,5], Ricardo J. Rodríguez[1,4]
[1]Department of Computer Science and Systems Engineering, Universidad de Zaragoza, Zaragoza, Spain
[2]IMDEA Networks Institute, Madrid, Spain
[3]University of Jyväskylä, Jyväskylä, Finland
[4]Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza, Spain
[5]Binare Oy, Jyväskylä, Finland
jcarrillo@unizar.es
guillermo.suarez-tangil@imdea.org
ancostin@jyu.fi
rjrodriguez@unizar.es

**Abstract**: The rise of malware targeting interconnected infrastructures has surged in recent years, driven largely by the widespread presence of vulnerable legacy IoT devices and inadequately secured networks. Despite the strong interest attackers have in targeting this infrastructure, a significant gap remains in understanding how the landscape has recently evolved. Addressing this knowledge gap is essential to thwarting the proliferation of massive botnets, thereby safeguarding end-users and preventing disruptions in critical infrastructures. This work offers a contemporary analysis of Linux-based malware, specifically tailored to IoT malware operating in 2021-2023. Using automated techniques involving both static and dynamic analysis, we classify malware into related threats. By scrutinizing the most recent dataset of Linux-based malware and comparing it to previous studies, we unveil distinctive insights into emerging trends, offering an unparalleled understanding of the evolving landscape. Although Mirai and Gafgyt remain the most prominent families and present a large number of variants, our results show that (i) there is an increase in the sophistication of malware, (ii) malware authors are adding new exploits to their arsenal, and (iii) malware families that originally attacked Windows systems have been adapted to attack Linux-based devices.
**Keywords: Static Analysis; Dynamic Analysis; Malware IoT; Malware Evolution; Malware lineage**

## 1. INTRODUCTION

The battle against malware has been going on for more than 30 years in the security community. Historically, Windows was the primary target due to its widespread use. However, a significant shift has occurred with the rise of Linux-based operating systems, driven by the growing popularity and the large number of Internet of Things (IoT) devices that use Linux as the underlying operating system.

The global expansion of IoT, which was predicted to reach 50 billion devices by 2020 according to Cisco's (Evans, 2011) and to expand to 100 billion by 2025 according to Huawei (Attia, 2019), brings various benefits in various sectors, including industry, transportation, smart cities, health-care, and daily life. The success and attractive features of IoT have revolutionized the technology, but have also raised serious concerns about the security and privacy of user data.

IoT comprises a multitude of interconnected devices within a network, using various communication protocols, such as MQTT, CoAP, and UPnP, among others. None of these systems are inherently designed with security in mind; rather, they prioritize cost-effectiveness and innovation, often at the expense of security and privacy (Girish et al., 2023). Additionally, the devices themselves have intrinsic limitations, including low storage and computing capacities, along with challenges such as manufacturers' reluctance to provide software/firmware updates (Ibrahim et al., 2023) or the complexity involved in installing updates, even for competent users.

Despite innovative features, many IoT devices exhibit vulnerabilities, ranging from default passwords to outdated software with known security issues (Costin et al., 2016, 2014), which are subsequently abused by increasingly sophisticated IoT malware and botnet families (Cozzi et al., 2018). For instance, Mirai infected more than 600,000 devices in 2016 using default passwords from IoT devices such as IP cameras or routers, causing one of the most severe denial-of-service attacks on the Internet (Antonakakis et al., 2017).

This highlights security challenges within networks of IoT systems, where keeping firmware/software up to date represents a significant hurdle (Carrillo-Mondéjar et al., 2023). Cybercriminals exploit known vulnerabilities to scan networks and launch attacks aimed at taking control of devices (Zhao et al., 2022). Notably, the number of malware attacks targeting interconnected infrastructures has significantly increased in recent years (Al Alsadi et al., 2022; Li et al., 2022). This is mainly due to the widespread presence of vulnerable IoT devices and poorly secured networks (Costin et al., 2014). However, despite attackers' interest in attacking

this type of infrastructure, there is still a significant gap in understanding how the landscape has evolved. It is crucial to address this knowledge gap to prevent the proliferation of massive botnets, protect end-users, and prevent disruptions in critical infrastructures.

In this regard, by examining the latest dataset of Linux-based malware and comparing it to previous studies, we unveil distinctive insights into emerging trends, offering unparalleled insight into the evolving landscape. We show that understanding the evolutionary dynamics of malware is a crucial step in addressing the challenges posed by new computing platforms, such as those in the IoT, online services, and critical infrastructures. By analyzing the latest IoT malware threats through the lens of a novel --- and updated --- dataset of over 45K malware samples we shed light on several threats unknown to the community. We set out to conduct our study using well-established methods in the area of IoT malware analysis (Al Alsadi et al., 2022; Alrawi and others, 2021; Carrillo-Mondéjar et al., 2020; Costin and Zaddach, 2018; Cozzi et al., 2020, 2018). In particular, we analyze malware samples to extract information and identify patterns. Then we run the samples and observe how they interact with the system. We combine the data obtained from both stages, using a distance function to correlate samples that exhibit similar behavior. We examine each of the unknown clusters in detail, using cutting-edge reverse engineering and forensic techniques, along with our extensive expertise as malware analysts. Through our analysis, we provide a deep understanding of the latest trends in malware, including the study of various malware families. Among other findings, we observe for the first time that:

i) the level of sophistication is increasing: the new IoT malware infrastructure is borrowing sophisticated tools from Advanced Persistent Threats (APTs), ii) there is a rapid proliferation of new variants with minimal infrastructure investment, and iii) expertise and tools devised by Windows malware are permeating the IoT ecosystem.

In summary, our paper introduces a state-of-the-art analysis of Linux-based malware in the context of IoT, specifically focusing on malware prevalent in the years 2021-2023. In particular, the key contributions of our work are:

1. We deploy an automated system dedicated to examining malware in IoT systems and specially engineered to examine the most recent malware samples seen in the wild at the time of writing. Our system employs a combination of well-established static and dynamic analysis, designed to establish connections between unknown samples and recognized threats.
2. Leveraging our system, we validate recent and unidentified IoT malware samples, characterizing their behaviors. Subsequently, we present an extensive analysis of the most recent trends based on an in-depth study of several malware families.
3. We elucidate the impact of architecture on malware propagation, providing valuable insights into the dynamics influencing the recent spread of malicious activities. Our study deliberately separates Linux-based IoT malware exploration from the predominant IoT architectures associated with malware in the wild.

The rest of the paper is organized as follows. Section 2 provides an overview of related work in the field of IoT malware analysis. Section 3 details our methodology, covering data collection, static and dynamic analysis. In Section 4, we present our analysis results, covering key aspects in the evolution of IoT malware based on static similarity (including a characterization for obfuscation and static-linking), behavioral similarity (including a characterization for dynamic-linking), correlation of malware family, and exploits used in the wild. Section 5 delves into the discussions, highlighting key findings and addressing the limitations of this work. Finally, Section 6 concludes the paper, summarizing our contributions and outlining potential avenues for future research.

## 2. RELATED WORK

Cozzi et al. (2018) design and implement a comprehensive analysis sandbox for Linux-based malware. Testing this sandbox involves using 10,548 malware samples collected between November 2016 and November 2017, covering more than ten different architectures.

Alrawi et al. (2021) introduce an analysis framework for the life-cycle of IoT malware, taking into account five components: infection vectors, payload properties, persistence methods, capabilities, and C&C infrastructure. Then, IoT malware is characterized through extensive measurements involving over 166,000 Linux-based IoT malware samples collected in 2019 across six different architectures. The results are compared with traditional studies on desktop and mobile malware, revealing Mirai as the predominant family for IoT malware and a variety of malware families for desktop and mobile systems.

Al Alsadi et al. (2022) aims to track the evolution of exploit code in IoT malware using static and dynamic analyses. It identifies 63 unique exploit codes from 17,720 binaries belonging to 26 IoT malware families and conducts multidimensional analysis. The analyzed binaries cover a period from 2015 to 2020. Findings show an average exploit lifespan of 23 months, with an impressive time-to-exploit of 29 months.
Carrillo-Mondéjar et al. (2020) use static and dynamic analyses with similarity detection in order to create an automated system to classify malware into related threats. The dataset used in this paper was collected by the authors in (Cozzi et al., 2018) between 2016 and 2017. Their findings show diverse sophistication levels in Linux-based malware, ranging from brute-force to botnet attacks. In contrast to architectures like x86_64, Linux-based malware is less complex. Therefore, static analysis techniques are effective due to the simplicity of the disassembly code.
Cozzi et al. (2020) employs binary code similarity to methodically reconstruct the lineage of IoT malware families, tracing their relationships, evolution, and variants. Their methodology is applied to a dataset comprising over 93,000 samples submitted to a prominent database (VirusTotal) over a 3.5-year period (from January 2015 to August 2018). Despite the simplicity of IoT malware, their findings indicate that antivirus tools struggle to detect minor modifications within binaries, resulting in mislabeling and missed detections.
Costin and Zaddach (2018) presents an extensive survey, analysis, and cross-validation of more than 60 IoT malware families and 48 unique vulnerabilities. The study covers malware families between the years 2008 and 2018. Alongside this, they provide an open-source malware analysis framework and a comprehensive dataset covering all IoT malware. Their findings reveal that the mean scores for vulnerabilities in IoT malware families are relatively modest, averaging 6.9 for CVSSv2 and 7.5 for CVSSv3. Importantly, preventive measures based on public knowledge could have been implemented, on average, at least 90 days before the submission of the initial malware samples for analysis.

All datasets that have been studied in previous research are before 2020. **Our work builds on these efforts to offer a contemporary view of the malware landscape.** Thus, our data collection spans over three years between early 2021 to late (December) 2023.

## 3. METHODOLOGY

In this section, we explain our methodology. First, we show the dataset we used. Then we present the static analysis that we conducted on each of the malware samples. Finally, we discuss the dynamic analysis and the sample correlation based on our corpus of execution traces.

### 3.1 Dataset

To better understand the evolution of malware targeting IoT devices and networks, we collect a set of malware samples from VirusShare (Corvus Forensics, 2023) between 2021 and 2023. VirusShare is a malware repository that archives real malware samples for different operating systems and architectures. We collected all torrent files uploaded to the repository during this time, which adds up to approximately 84 torrent files, each containing around 65,536 malware samples. This dataset provides us with recent and comprehensive information on the evolution of malware targeting IoT devices, allowing us to compare it with previous studies. We focus on detecting malware targeting devices running Linux-type operating systems and use ELF (Executable and Linkable Format) binary file format to filter samples. Our dataset comprises 45,871 malware samples targeting various CPU architectures. Table 1 provides a breakdown of all the files, distributed by the architectures they target. For each malware sample in our dataset, we obtain its VirusTotal report.
Additionally, we use AVClass (Sebastián and Caballero, 2020) to label malware samples. This tool is widely accepted and used by the community for labelling malicious samples (Alrawi and others, 2021; Wei et al., 2017; Zhu et al., 2020). AVClass uses the antivirus labels from VirusTotal reports to label a given sample. In total, AVClass provides labels for 44,204 malware samples from 248 different families.

### 3.2 Static Analysis

We statically analyze the malware binaries to extract metadata from them. During this phase, we collect information such as the architecture and endianness used by the binary, the type of linking it employs (static or dynamic), as well as other details such as cyclomatic complexity, entropy, file size, and text strings. This analysis allows us to obtain information from dynamic link libraries. This information will be used to choose

the virtualized environment that best suits the analysis we carry out in subsequent stages. This static analysis allows us to better understand the behavior of malware during dynamic analysis.
Additionally, during this phase we also check whether the binary is packed or not. To do so, we first use the entropy of the binary to identify if it is packed, and then apply a set of rules to detect the specific packer. For detection we employ the cross-platform tool Detect It Easy (DIE), which contains pre-defined signatures to identify a large number of packers and compilers. We also found a few samples packed with Ultimate Packer eXecutable (UPX), and tried to unpack them using the UPX tool itself.
Finally, we look for exploits targeting IoT devices. To do this, we collect exploits from various sources and create specific YARA rules (Alvarez, 2013) for each of them. We also create rules for the vulnerabilities mentioned in (Al Alsadi et al., 2022; Alrawi and others, 2021; Costin and Zaddach, 2018), as well as other exploits targeting IoT devices found on the Internet, mainly from exploit.db. In total, we produce YARA rules for 132 exploits. YARA rules focus on searching for a Uniform Resource Identifier (URI) present in exploits targeting HTTP-based protocol interfaces. We check each sample to identify any exploits embedded in them through static detection. We then use the static information collected in this phase to assist in the dynamic analysis process.

**Table 1 Dataset distribution**

| Arch | Total | Not packed | Packed | Static Linking | Dynamic Linking |
|---|---|---|---|---|---|
| ARM | 15,603 | 13,146 (15.75%) | 2,457 (15.75%) | 13,190 (84.54%) | 2,369 (15.18%) |
| MIPS | 11,437 | 7,900 (69.07%) | 3,537 (30.93%) | 9,741 (85.17%) | 1,666 (14.57%) |
| Intel 80386 | 5,416 | 4,679 (86.39%) | 737 (13.61%) | 5,030 (92.87%) | 371 (6.85%) |
| x86-64 | 3,662 | 3,268 (89.24%) | 394 (10.76%) | 2,059 (56.23%) | 1,565 (42.74%) |
| PowerPC | 2,806 | 2,166 (77.19%) | 640 (22.81%) | 2,775 (98.90%) | 25 (0.89%) |
| Renesas SH | 2,567 | 2,567 (100%) | 0 | 2,538 (98.87%) | 24 (0.93%) |
| Motorola m68k | 2,218 | 2,218 (100%) | 0 | 2,064 (93.06%) | 150 (6.76%) |
| SPARC | 1,510 | 1,510 (100%) | 0 | 1,494 (98.94%) | 14 (0.93%) |
| AARCH64 | 525 | 514 (97.90%) | 11 (2.10%) | 37 (7.05%) | 487 (92.76%) |
| Others | 127 | 127 (100%) | 0 | 43 (33.86%) | 80 (62.99%) |
| | **45871** | 38,095 (83.05%) | 7,776 (16.95%) | 38,971 (84.96%) | 6,751 (14.72%) |

### 3.3 Dynamic Analysis

We develop a framework to analyze each of the malware samples in our dataset. This framework leverages a virtualized environment that we implemented for the main architectures. We equip this environment with home-made tools to speed up the execution of the malware sample and collect information related to the system calls made during its execution. To create the virtual machines, we use Buildroot (Petazzoni and Electrons, 2012), which helps us automate the process of creating an embedded Linux system. For the architectures that have Buildroot support, we create different virtual machines using various runtime libraries (i.e., uClibc, glibc, and musl) to ensure that our system supports multiple dynamic linking libraries.
We use the results of the static analysis (e.g., architecture, endianness) to determine which virtual machine is best suited to run the malware sample, thus increasing the probability of success. Overall, we create virtual machines for nine different architectures, including ARM, MIPS, MIPSel, Intel 80385, x86-64, SPARC, PowerPC, Renesas SH, and Motorola m68k. However, we only have virtual machines with uclibc as a dynamic linking library for Motorola m68k and SPARC architectures. For the remaining architectures, we run samples linked dynamically using glibc, uclibc, or musl. To emulate these architectures, we use QEMU as it supports with most of the architectures found in our dataset. To monitor the system calls made at runtime by the malware, we utilize the *strace* system utility and run each sample for 60 seconds in our virtualized environment, since code coverage tends to stabilize within the first minute or two of execution (Küchler et al., 2021). At the end of this stage we obtain execution traces of the samples that were executed successfully. These execution traces consist of the sequence of system calls made by the malware sample, as well as each of the arguments to that system call.

### 3.4 Phylogenetic Analysis

Our dynamic analysis framework allows us to link samples based on the behavior they exhibit during execution in our sandboxes. We use the execution traces we collect to connect samples that have a similarity greater

than 80%. To determine this similarity, we extract N-grams of size 4 from the set of executed system calls and use the Jaccard index to calculate the similarity between each pair of samples of a particular architecture. We define two samples as similar when their Jaccard index is greater than 80%. Next, we use clustering to represent the similarities between the samples. We construct a graph to examine the phylogenetic relationship (evolutionary connection) between various malware samples and families. Each node in the graph represents a sample, while the edges represent the presence of similarity between them.

## 4. RESULTS

This section presents the result of the statistical analysis of the metadata of the existing malware samples in our dataset.

### 4.1 Overview

As we pointed out in Section 3.1, our original data set consists of 45,871 ELF malware samples spread across more than 10 CPU architectures. Table 1 shows a breakdown of the number of samples per architecture. We observe that the architectures with the most samples are ARM 32 bits and MIPS, which includes both little and big endianness. Between both architectures, they represent approximately 59% of the total set of samples existing in the dataset.

### 4.2 Obfuscation

We also study the number of samples that are obfuscated (packed) and the number of samples linked statically and dynamically. We see obfuscation in 7,776 (16.95%) of our samples, many of them using UPX. When unpacking these samples, we observe that 38.87% of the samples have been obfuscated with a custom packer that prevented automatic unpacking (e.g., by adversarially modifying the headers of the binary). As a result, we unpack 67.13% of the samples.

**Takeaway**. We have observed that unlike other platforms like Windows or Android, malware developers on IoT devices do not use code packers or obfuscators to make their malicious code difficult to detect (Dong et al., 2018; O'Kane et al., 2011). This is because security measures for IoT are not widely practiced. The primary reason for this is the lack of security solutions that are specifically designed for IoT devices and networks. Previous studies (Al Alsadi et al., 2022; Alrawi and others, 2021; Cozzi et al., 2018) reported similar type (UPX) and percentages of packed samples in their datasets, suggesting that the malware ecosystem has not evolved in complexity when it comes to the use of packers.

### 4.3 Static analysis

We see that most of the malicious software samples (about 85%) in our collection are statically linked. This is a common practice for malware that targets IoT devices because there are different architectures and library versions on these platforms. By statically linking the binaries, the malware creators can ensure that the software will run correctly on the infected systems. Additionally, we found that 69.82% and 74.21% of statically and dynamically linked binaries, respectively, have had their symbols removed.

**Takeaway.** Symbol removal is a technique used by malware creators to obfuscate their code. Stripping symbols makes it harder for security analysts and tools to analyze and understand the functionality of the malicious software. Despite leveraging vanilla packers (UPX), our finding suggests a deliberate effort by malware authors to hinder reverse engineering and increase the complexity of detection.

### 4.4 Dynamic analysis

Our dynamic analysis framework analyzed 35,378 (77.12%) of the samples observed between 2021 and 2023. By analyzing 35,378 execution traces, we study the following three dimensions: interaction between processes, evasion mechanisms, and profiling behaviors.

**Process interaction.** We see that around 29% execute a single process and around 66% create three or more processes, with 16,283 being the maximum number of processes created by a sample related to the exploitation of CVE-2015-1325. We also see that 4,566 malware samples attempted to execute some system commands. The most used shell commands were *sh*, *iptables*, *killall*, *rm*, *chmod*, and *wget*.

**Evasion.** We found that IoT malware tries to hide its identity, some by disguising itself with the name of legitimate processes such as *sshd* (39.49%), *busybox* (6.32%), or services ([*kworker*] (4.25%), [*watchdog0*] (4.19%)). Some others choose a random name for cloaking. We noticed that a few samples were using certain

access paths crafted so that they can detect virtual environments (225 samples). We also observed that 169 samples try to "trace" themselves to prevent the sample from being debugged or to detect if a debugger is attached.

**Profiling.** We observed that the malware gathers system data, likely for communication with C2C or its operations. Specifically, we see 21,711 access paths found under the */proc* directory, 611 access paths under the */sys* directory, and 8025 access paths related to the system configuration under */etc*.

**Takeaway.** Our findings on the use of evasion indicate that some IoT malware authors are concerned about their samples being analyzed dynamically, but we did not find this figure as prevalent as the obfuscation used to evade static analysis. This speaks to the challenges the CIT community faces to emulate the full gamut of architectures, which may devoid malware developers from the need to evade dynamic analysis as we discuss in Section 5.

### 4.5 Evolutionary connection

We generate similarity graphs for the ARM, MIPS/MIPSel, Intel 80386, x86-64, PowerPC, Renesas SH, Motorola m68k, and SPARC architectures. Due to space restrictions, we only show the similarity graphs for ARM, MIPS, x86, and x86-64.

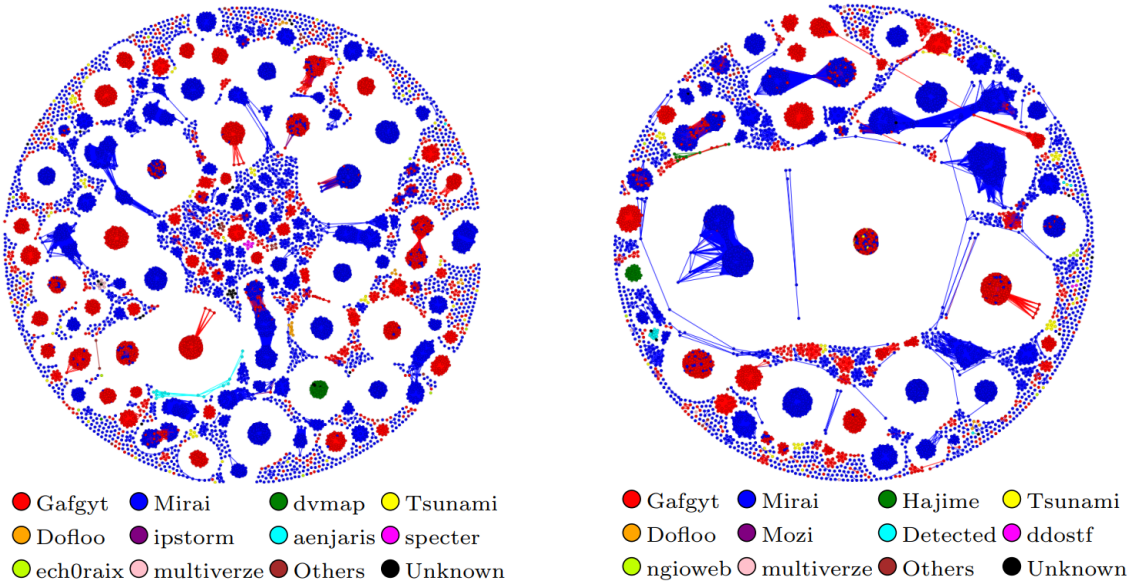
**MIPS and ARM.** Figure 1 shows the similarity graph for the MIPS and ARM architectures. The colors represent the labels of the families with the most presence in each of the architectures. Mirai and Gafgyt are the most prominent malware families. Interestingly, we see different clusters for the Mirai and Gafgyt families, showing evolutionary traits stemming from variations of the same family. This indicates that antivirus engines using generic signatures for Gafgyt and Mirai families may fail to detect new variants.

We see new families like Aenjaris, Specter, or Mutiliverze. Aenjaris is a ransomware family that was created using leaked Babuk source code, while Specter is a new botnet that specifically targets IoT devices. We have also observed a new family called Multiverze, which appears in both MIPS, ARM, x86, and x86-64 architectures. However, upon close inspection, we have determined that Multiverze is a generic label that encompasses a range of behaviors, from backdoors to cryptomining malware. These types of "bag of malware samples" severely challenge existing data-driven identification and detection mechanisms due to their morphic behavioral traits.

**Intel 80385 and x86-64.** Figure 2 shows the clustering of the samples belonging to the Intel 80385 and x86-64 architectures. These architectures are commonly used in personal computers and servers, and thus, consist of a larger number of different families. However, significant differences can be observed between them. Looking at Intel 80385 in Figure 2 we see, apart from the usual suspects, Xorddos and Setag. These widely known families continue to operate.

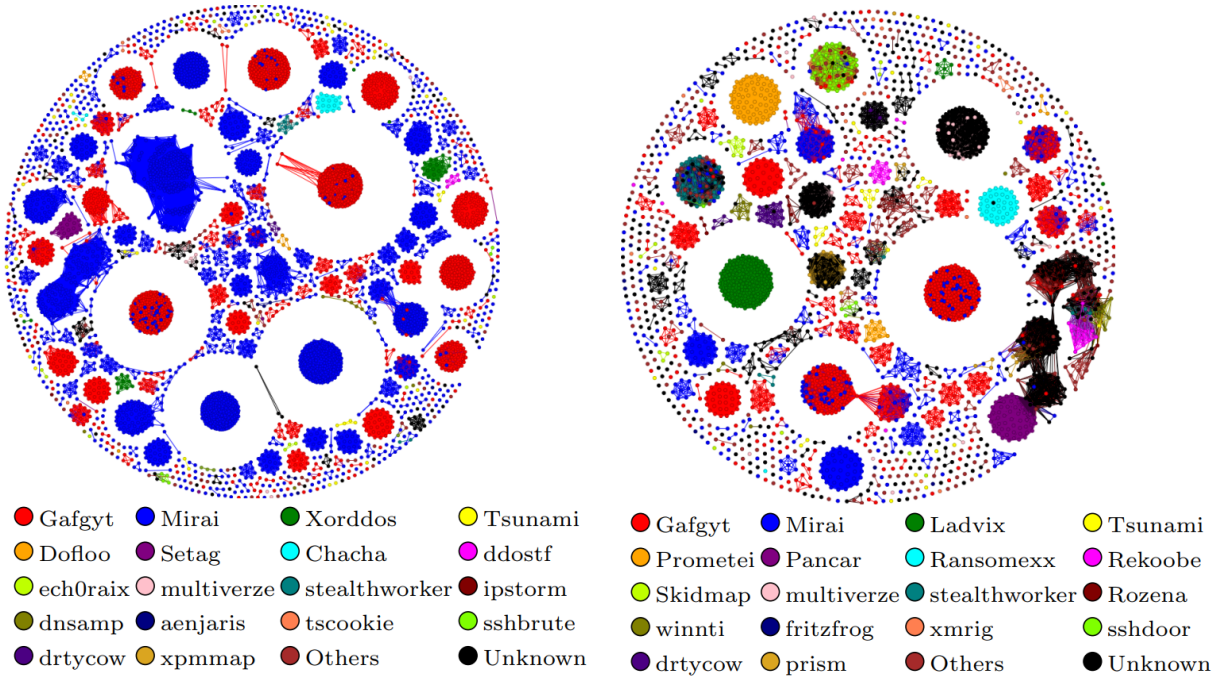
Looking at x86-64 in Figure 2 we observe the following differences:

- Although both Mirai and Gafgyt families are still present, there are not as many clusters of both families anymore. This suggests that not all variants of Mirai and Gafgyt generate specific versions for this architecture.
- We also observe several families with significant representation, such as Ladvix, Prometei, Pancar, Ransomexx, and Rekoobe. These families have different functionalities, ranging from a file infector to a Trojan that is based on the publicly available Tiny SHell source code, which is used by APT31.
- We have identified several cryptomining families such as Prometei, Skidmap, and Xmrig. Prometei, for instance, is a botnet that was initially discovered on Windows and now targets Linux-based systems as well.
- We also see the Rozena family, a sophisticated type of malware written in Python where the interpreter is embedded in the binary itself.
- Another prominent variant we came across is called Prism, which is an open-source backdoor.
- Finally, we have seen several examples of the Multiverze family, which is a generic name given to many unidentified samples as discussed earlier. After analyzing a few samples from the Multiverze cluster and some unlabeled samples, we determined that it is a downloader most likely engaging in Pay-Per-Install (PPI) services.




a) MIPS and MIPSel b) ARM 32 bits

**Figure 1: Graph Similarity**




a) Intel 80385 b) x86-64

**Figure 2: Graph similarity**

**Other architectures.** Clustering on the rest of the architectures shows that variants of Mirai, Gafgyt, and some Tsunami samples are also prevalent on Motorola m68K, Renesas SH, and PowerPC. It is worth noting that these architectures have striking similarities to those depicted in Figure 1.

**Takeaway.** While we see that Mirai and Gafgyt continue to be prevalent families (Carrillo-Mondéjar et al., 2020), we see an explosion of variants. We note that a significant evolution of popular IoT malware can jeopardize existing defenses. We also see new families in the landscape that have the potential (sophisticated open-sourced projects) to become as insidious as Mirai.

### 4.6 Exploits used in the wild

Our analysis reveals the use of 89 different exploits. We map each exploit with the samples we have collected in the wild. In total, we found that 13,061 (28.47%) samples with actionable exploits. Figure 3 displays a graph,

where the nodes represent malware samples and the vulnerability they exploit. Edges represent the reuse of exploits across samples. The size of the node represents the severity of the vulnerability. The different colors represent the corresponding malware families. All the samples found are distributed mainly in 5 families, with the Mirai and Gafgyt families being the ones that stand out from the rest. Some exploits are very central to the samples of one single family (e.g., Gafgyt), while other malware families carry an arsenal of exploits scattered across samples.

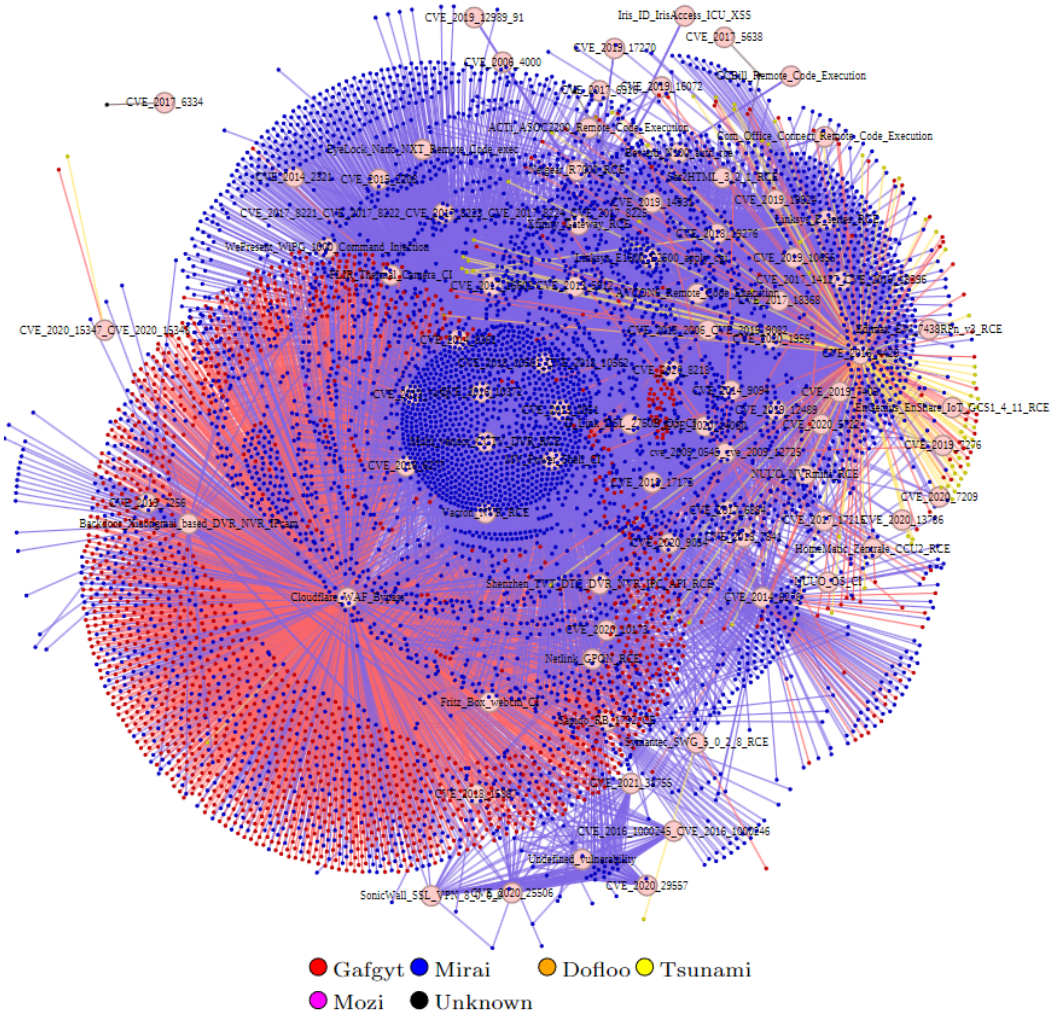



**Figure 3: Exploits**

**Takeaway.** After comparing our results with those presented in previous studies (Al Alsadi et al., 2022; Alrawi and others, 2021), we have noticed a slight increase in the number of exploits found, which were 25 and 58 respectively. This indicates that malware creators are including new exploits in their arsenals, that are already publicly available as proof of concept, to improve their chances of infecting new devices and achieving success. However, some other malware families persistently stick to a handful of exploits.

## 5. DISCUSSION

### 5.1 Findings

**Strategies.** IoT devices typically have limited resources and diverse architectures, making them attractive targets. Despite not seeing a radical change in the technology used to pack malware, IoT malware authors are adopting strategies, such as statically linking stripped binaries, to make detection more difficult while ensuring their malicious software runs seamlessly on different IoT devices. This insight is valuable for IoT security strategies as it emphasizes the need for customized defenses against such threats.

**Evasion.** The limited need for mechanisms to evade dynamic analysis suggests potential challenges within the CIT community in accurately emulating a comprehensive range of architectures. This could potentially reduce

the imperative for malware developers to actively avoid dynamic analysis. We found this to be a valuable new insight, which calls for important future work in the research community.

**Evolution.** Malware targeting IoT devices is constantly changing. Most samples in the IoT landscape are variants of the well-known Mirai and Gafgyt malware families. However, in recent years other families of diverse natures have appeared. Interestingly, some of these malware families initially attacked Windows systems but have expanded to Linux-based systems. This insight can help forensic analysts and inspire new research into developing effective mitigations for Windows systems in the IoT field. On top of that, we have also seen new malware families that rely heavily on sophisticated code, publicly available on the Internet, emerging from an Advanced Persistent Threat (APT31), for which we foresee a prominent emergence. We advocate for greater scrutiny of open repositories and underground forums that may host IoT malware.

**Prioritization.** We noticed an increase in the number of exploits found in previous studies. Naturally, malware authors are adding new exploits to their toolbox. On the other hand, some malware families seem to continue using the same old exploits. This insight can provide a valuable prioritization strategy to CIT when designing mitigations. That is, fixing one single exploit will effectively thwart many malware attacks (most in Gafgyt).

### 5.2 Limitations

Our analysis platform may have some limitations that may influence the study. First, the analyses of the interaction with the system are carried out in user mode through the *strace* tool, so it can be detected by malware. However, only 0.48% of the dynamically executed samples use *ptrace* to attempt to debug themselves and could therefore detect that the process is being debugged since there cannot be two debuggers attached to the same process. Execution traces could also be affected if the malware detects that it is running in a virtualized environment, so it may change its behavior relative to a real environment. However, we have only identified 0.64% of the samples that try to access system paths that allow them to detect whether it is a virtual machine or not. On the other hand, for the extraction of exploits, we focus exclusively on the clear text strings that appear in the binaries and, therefore, there may be more exploits than we report since these may be encrypted or encoded within the binary and decrypted before use.

## 6. CONCLUSIONS

In this work, we presented a thorough examination of recent malware trends targeting IoT platforms between 2021-2023. Through data analysis, we systematically characterized malware across diverse threats using static and dynamic features. This approach helped us identify new malware samples and their connections to previously known threats, following one of the main objectives of this work.

Our methodology extended beyond individual samples, allowing the extraction of knowledge over large groups of connected samples. This approach was instrumental in investigating recent unlabeled malware samples found in the wild. To expand our analysis, we used state-of-the-art reverse engineering techniques and our own malware analysis pipeline to study unknown clusters and provide nuanced insights into the type of exploits they leverage. Our analysis delved into the latest trends, revealing novel insights including: i) an increase in sophistication as the emerging IoT malware infrastructure incorporates advanced tools borrowed from APT31, ii) the dissemination of knowledge and tools originally designed for Windows malware in the IoT ecosystem, and iii) a rapid proliferation of novel variants driving future work in the area as discuss next.

**Future work:** The minimal need for evasive mechanisms in dynamic analysis highlights upcoming challenges for the CIT community in effectively simulating diverse architectures. This emerging trend may alleviate pressure on malware developers to actively evade dynamic analysis. Our discovery presents a novel perspective, underscoring the need for significant future efforts within the research community to address this evolving dynamic.

## ACKNOWLEDGEMENTS

This research was supported in part by TED2021-132900A-I00 and by TED2021-131115A-I00, funded by MCIN/AEI/10.13039/501100011033, by the Recovery, Transformation and Resilience Plan funds, financed by the European Union (Next Generation), by the Spanish National Cybersecurity Institute (INCIBE) under *Proyectos Estratégicos de Ciberseguridad -- CIBERSEGURIDAD EINA UNIZAR*, and by the University, Industry and Innovation Department of the Aragonese Government under *Programa de Proyectos Estratégicos de Grupos de Investigación* (DisCo research group, ref. T21-23R). G. Suarez-Tangil was appointed as 2019 Ramon y Cajal fellow (RYC-2020-029401-I) funded by MCIN/AEI/10.13039/501100011033 and ESF Investing in your future.

(Part of) This work was supported by the European Commission under the Horizon Europe Programme, as part of the project LAZARUS (https://lazarus-he.eu/) (Grant Agreement no. 101070303). The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

## REFERENCES

Al Alsadi, A.A., Sameshima, K., Bleier, J., Yoshioka, K., Lindorfer, M., van Eeten, M., Gañán, C.H., 2022. No Spring Chicken: Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis, in: Proceedings of the ACM on Asia Conference on Computer and Communications Security (ASIACCS). Association for Computing Machinery.

Alrawi, O., others, 2021. The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle, in: 30th USENIX Security Symposium. USENIX Association.

Alvarez, V., 2013. YARA - The pattern matching swiss knife for malware researchers [WWW Document]. The pattern matching swiss knife for malware researchers. URL http://virustotal.github.io/yara/ (accessed 1.27.24).

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y., 2017. Understanding the Mirai Botnet, in: 26th USENIX Security Symposium (USENIX Security 17). USENIX Association, Vancouver, BC, pp. 1093–1110.

Attia, T.M., 2019. Challenges and opportunities in the future applications of IoT technology.

Carrillo-Mondéjar, J., Martínez, J.L., Suarez-Tangil, G., 2020. Characterizing Linux-based malware: Findings and recent trends. Future Generation Computer Systems 110, 267–281.

Carrillo-Mondéjar, J., Turtiainen, H., Costin, A., Martínez, J.L., Suarez-Tangil, G., 2023. HALE-IoT: Hardening Legacy Internet of Things Devices by Retrofitting Defensive Firmware Modifications and Implants. IEEE Internet of Things Journal 10, 8371–8394. https://doi.org/10.1109/JIOT.2022.3224649

Corvus Forensics, 2023. VirusShare.com - Because Sharing is Caring.

Costin, A., Zaddach, J., 2018. IoT malware: Comprehensive survey, analysis framework and case studies. BlackHat USA.

Costin, A., Zaddach, J., Francillon, A., Balzarotti, D., 2014. A {Large-scale} analysis of the security of embedded firmwares, in: 23rd USENIX Security Symposium.

Costin, A., Zarras, A., Francillon, A., 2016. Automated dynamic firmware analysis at scale: a case study on embedded web interfaces, in: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security.

Cozzi, E., Graziano, M., Fratantonio, Y., Balzarotti, D., 2018. Understanding Linux Malware, in: IEEE Symposium on Security and Privacy (SP).

Cozzi, E., Vervier, P.-A., Dell'Amico, M., Shen, Y., Bilge, L., Balzarotti, D., 2020. The Tangled Genealogy of IoT Malware, in: Annual Computer Security Applications Conference (ACSAC). Association for Computing Machinery.

Dong, S., Li, M., Diao, W., Liu, X., Liu, J., Li, Z., Xu, F., Chen, K., Wang, X., Zhang, K., 2018. Understanding Android Obfuscation Techniques: A Large-Scale Investigation in the Wild, in: Beyah, R., Chang, B., Li, Y., Zhu, S. (Eds.), Security and Privacy in Communication Networks. Springer International Publishing, Cham, pp. 172–192.

Evans, D., 2011. How the next evolution of the internet is changing everything. Cisco Whitepapers.

Girish, A., Hu, T., Prakash, V., Dubois, D.J., Matic, S., Huang, D.Y., Egelman, S., Reardon, J., Tapiador, J., Choffnes, D., others, 2023. In the Room Where It Happens: Characterizing Local Communication and Threats in Smart Homes, in: Proceedings of the 2023 ACM on Internet Measurement Conference. pp. 437–456.

Ibrahim, M., Continella, A., Bianchi, A., 2023. AoT - Attack on Things: A security analysis of IoT firmware updates, in: 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P). pp. 1047–1064. https://doi.org/10.1109/EuroSP57164.2023.00065

Küchler, A., Mantovani, A., Han, Y., Bilge, L., Balzarotti, D., 2021. Does Every Second Count? Time-based Evolution of Malware Behavior in Sandboxes., in: NDSS.

Li, R., Li, Q., Huang, Y., Zhang, W., Zhu, P., Jiang, Y., 2022. IoTEnsemble: Detection of Botnet Attacks on Internet of Things, in: Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part II. Springer-Verlag, Berlin, Heidelberg, pp. 569–588. https://doi.org/10.1007/978-3-031-17146-8_28

O'Kane, P., Sezer, S., McLaughlin, K., 2011. Obfuscation: The Hidden Malware. IEEE Security & Privacy 9, 41–47. https://doi.org/10.1109/MSP.2011.98

Petazzoni, T., Electrons, F., 2012. Buildroot: a nice, simple and efficient embedded Linux build system, in: Embedded Linux System Conference.
Sebastián, S., Caballero, J., 2020. AVclass2: Massive Malware Tag Extraction from AV Labels, in: Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC). Association for Computing Machinery.
Wei, F., Li, Y., Roy, S., Ou, X., Zhou, W., 2017. Deep ground truth analysis of current android malware, in: 14th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA). Springer.
Zhao, B., Ji, S., Lee, W.-H., Lin, C., Weng, H., Wu, J., Zhou, P., Fang, L., Beyah, R., 2022. A Large-Scale Empirical Study on the Vulnerability of Deployed IoT Devices. IEEE Transactions on Dependable and Secure Computing 19, 1826–1840. https://doi.org/10.1109/TDSC.2020.3037908
Zhu, S., Shi, J., Yang, L., Qin, B., Zhang, Z., Song, L., Wang, G., 2020. Measuring and modeling the label dynamics of online {Anti-Malware} engines, in: 29th USENIX Security Symposium.