Our ability to precisely estimate the average age of IoT malware exploits has been greatly improved with the use of static and dynamic analysis

[1] Dr.Jaggarao Gedela,[2] Dr.Madan Mohan rao Nelluri, [3] Dr.Krishnamoorthy T, [4] Dr.Naganaik.M

[1,2,3,4] Associate Professor, Department of ECE, Balaji Institute of Technology & Science, Warangal-Narsampet Rd, Laknepally, Madhira D, Telangana 506330

ABSTRACT

Many different types of software and hardware make up the IoT, and each of them has its own set of security flaws. Previous studies have indicated that the initial infection attempts might happen within minutes after an IoT device is linked to the Internet. However, there is still a lack of information regarding the development of attack vectors, including which vulnerabilities are being targeted in the field, how the functionality has evolved over time, and how long exploits have been in use. A better grasp of these issues may aid in the creation and rollout of IoT networks with more safety and confidence. By examining 17,720 samples acquired from three distinct sources between 2015 and 2020, we report the first longitudinal research of IoT malware attacks. We use static and dynamic analysis to extract exploits from these binaries, and then analyse them along four dimensions: (1) how infection vectors have changed over time, (2) how long can exploit has been in use, the age of the vulnerability, and the time it takes to exploit it, (3) the functionality of exploits, and (4) the manufacturers and types of IoT devices that have been targeted. Several trends emerge from our descriptive analysis: Internet of Things malware has progressed from relying just on brute force assaults to include a wide variety of device-specific flaws. Exploits, once created, are seldom forgotten. Even the newest binaries exploit (very) outdated flaws. Some vulnerabilities have been known for years, yet new exploits continue to be created for them. When compared to malware that targets other contexts, we discover that the mean time to exploit following vulnerability disclosure is around 29 months.

KEYWORDS

Exploits, vulnerabilities, infection vectors, malware, Internet of Things, static analysis, dynamic analysis

INTRODUCTION

The increased use of IoT devices, such as IP cameras and smart home appliances, not only offers us with novel services but also opens up new entry points for cybercriminals. An alarming number of electronic gadgets have been compromised [4]. While social engineering and user engagement have become more common vectors for attacks on desktop and mobile devices, vulnerabilities remain the primary method of infection for IoT [3]. While our understanding of IoT malware families and their capabilities has grown [11, 61], our understanding of how attackers choose which vulnerabilities to exploit remains mostly unknown. From a dozen or so disclosed in 2010 to over 500 in 2019 [6], the number of vulnerabilities connected to the Internet of Things (IoT) is increasing at the same rate as the total number of newly identified vulnerabilities [41]. Which of these flaws are being exploited? Do the creators of many kinds of malware target the same vulnerabilities? When a new vulnerability is disclosed, how long does it take before people begin exploiting it? How long do they continue to concentrate on a single security hole? For PCs and servers, we have seen that attackers tend to go after the version of the programme that is only one version behind the one that has the most recent patch [1, 51, 58]. However, the IoT ecosystem makes patching more complex, therefore this trend is unlikely to hold [55]. While previous studies have looked at exploit code used by individual malware families at various periods [4, 14, 24], we still don't have a good systemic knowledge of how vulnerabilities are targeted over time in the IoT malware environment as a whole. Alrawi et al[3] .'s concomitant investigation is the most closely similar prior art. The research looked at a large sample of IoT malware binaries gathered in 2019, identifying 25 vulnerabilities seen in the wild in 2019. Some of the findings in this publication are ultimately confirmed by our research. We take it a step further by tracking the development of exploits over the course of five years and across malware families, in addition to the development of the vulnerabilities themselves. We've discovered 63 exploits and have been monitoring them over time, along with the 68 vulnerabilities they attack. As a result, we are able to show that there are trends in the longevity of exploits, vulnerabilities, and time-to-exploit that have not been seen before.

CONTENT THAT IS RELEVANT

Few studies have investigated the security of previously deployed IoT devices and their vulnerabilities, despite the fact that most IoT security research has focused on creating appropriate security measures for resource-constrained devices. In order to suggest more effective countermeasures, Feng et al. [21] researched IoT vulnerabilities utilising several

sources in the open, including public vul notability and exploit databases, forums, mailing groups, and blogs. Lebowski and Piotrowski [6] developed a vulnerability categorization based on the CVE of IoT systems using comparable data sources but employing machine learning. Alawi et al. [2] conducted the first empirical study of the security measures and vulnerabilities inherent in commercially available IoT devices by ignoring open data and instead focusing on a sample of home-based IoT installations. While these early research efforts concentrated on preventative measures, more recent studies have investigated assaults by studying IoT malware [3, 5, 10, 14, 16]. The majority of these research either utilise honeypots to detect IoT malware (like Hotpots [43]), get samples from Virus Total [50], or rely on publicly available threat intelligence data (e.g., Cyberbooks [15]). IoT malware has been studied before; for instance, Hamulate and Razali [23] looked at the CVEs that have received the most attention in the media. They proved that IoT malware focuses on exploitable vulnerabilities that may be used to infiltrate a device without the intervention of the user. To better understand the code repetition and development of various IoT malware families, Alawi et al. [3] recently evaluated a collection of 166,000 IoT mal ware samples gathered in 2019. The authors, like us, utilised static and dynamic analytics to compare and contrast the various malware strains. Our research builds on their early efforts in four ways, allowing us to define the development of various exploit types over a longer time frame. In comparison to the 25 vulnerabilities studied by Alawi et al. [3], we provide a much more in-depth understanding of the targeted vulnerabilities by analysing 68 vulnerabilities (excluding hard-coded credentials) as found in the binaries; (2) we covered binaries from a much larger time frame, 2015 to 2020; (3) we extracted exploits via a combination of static analysis and dynamic analysis. They were unable to pinpoint when exactly the first information about the vulnerabilities they found appeared in the industry. Neither do they see this in their own data, nor do they track it over time (like the period during which binary exploits are used). Our research provides the groundwork for knowing why and how attackers keep using the same exploits over and over again, even knowing that they have been patched.

METHODOLOGY

By monitoring the changes in exploit code used by IoT malware, we hope to learn more about the vulnerabilities and devices that are being targeted over time. Both static and dynamic analysis of binaries may be used to locate exploit code. Manual static analysis, or reverse engineering, is labour-intensive but more thorough and trustworthy than automated static analysis but is vulnerable to code obfuscation and packing. Auto-mated dynamic analysis, on the other hand, is scalable and can handle packing, but it has the problem of not being as thorough in its coverage of ex-plaits. The strengths of both static analyses performed by humans and dynamic analysis performed by computers are combined in our technique. After getting these first findings, we supplement them with a search for specific exploits in a binary repository that spans the three years prior to when our binaries were gathered. In Figure 1, we provide an overarching summary of our approach. Table 1: Number of collected samples per dataset, collection time period, and included malware families.

Dataset	# Binaries	# Shell Scripts	Malware Families (Avclass2)	# Vhash	# Singletons	Collection Period
URLhaus	2,292	6	Mirai, Gafgyt, Tsunami, Xorddos, Hajime, Mozi	108	6	Jul 2020 - Oct 2020
Honeypot	5,855	2,815	Mirai, Gafgyt, Tsunami, Xorddos, Hajime, Mozi, Berbew, Generica, Silex	107	59	Sep 2018 - Aug 2020
Genealogy	6,752	0	Mirai, Gafgyt, Tsunami, Hajime, Generica, Casur, Dofloff, Dnsamp, Dofloo, Ciboc, Goram, Iotreaper, Luabot, Minerd, Pnscan, Qysly, Rakos, Rampo, Remaiten, Satori, Shishiga and Zlorg	277	144	Jan 2015 - Aug 2018

Data Collection

Weaponized Code for Internet-of-Things Devices

To generate exploit signatures, we first gather samples from the present day from two distinct sources (Uraeus and a honeypot), and then compare them to an older dataset (Genealogy) that spans a longer time period. Malware distribution links are collected in Uraeus, a central database [56]. By mining this repository, we were able to compile a collection of fresh binaries for use in our dynamic and static testing. We obtained a daily file from July 2020 through October 2020 that included the URLs of all recorded binaries as well as other relevant information like file type. Since our research is limited to Internet of Things malware, we only collected URLs for "Executable and Linkable Format" (ELF) files. During that four-month time span, we used a script to retrieve those files regularly. We grabbed a total of 2,298 binaries for various platforms and CPU architectures, such as Renesas SH, Motorola 68000, SPARC, Intel 80386, ARM, PowerPC, MIPS, ARC Cores Tangent-A5, and AMD x86-64.

Since prior research has demonstrated that x86-based malware is prevalent in IoT devices [37], we have included it in our analysis. Despite the fact that 6 of the 2,298 files were shell scripts, we carefully checked and confirmed that they just performed binary downloads and did not use any propagation tactics. Using the Hotpots [43] honeypot, we were able to collect 5,855 MIPS binaries from September 2018 through August 2020. Hotpots combines a low-interaction honeypot with a high-interaction honeypot. The low-

interaction honeypot acts as a proxy for a variety of network services, including Telnet, HTTP front-ends, CPE WAN Management Protocol (CWMP), a backdoor of Natis routers, and the remote access setup service of a number of IP cameras. Four bare-metal Internet of Things devices are used in the high-interaction honeypot (a router, an IP camera, and two WIfi storage devices). More than a hundred and thirty Japanese IP addresses are now linked to the honeypot. In addition, we were able to collect a dataset consisting of 2,815 files recorded by Hotpots but not of the ELF binary format. 2.608 contained the functionality of downloaders utilising wet, curl, etc. when run as shell scripts in a secure environment. Among the remaining 207 files, 10 were found to be Python scripts, 2 were found to be Perl scripts, and the rest 195 were found to be ASCII texts and not script files. As a follow-up, we personally evaluated these 10 Python programmes and 2 Perl scripts and ran them in a sandbox, where we discovered that just one of the Python scripts has vulnerabilities. Since this is the case, we will confine the rest of our investigation to the binary samples.

CONSUME THE LANDSCAPE

The results of our research on exploits and vulnerabilities in IoT malware are shown below. The results from all three datasets are summarised in Table 3. We discovered a total of 64 infection vectors, the majority of which involve brute-forcing hard-coded credentials, and 63 distinct exploits that aim to attack 68 vulnerabilities. Table 3 shows the frequency with which each vulnerability was found in each dataset in the last column. The vulnerabilities, exploits, and device makers are all identified. The table excludes two sets of Uraeus binaries because they did not include vulnerabilities. Twenty-seven of the 108 (or 25%) binaries only had brute-force credentials hard-coded. A

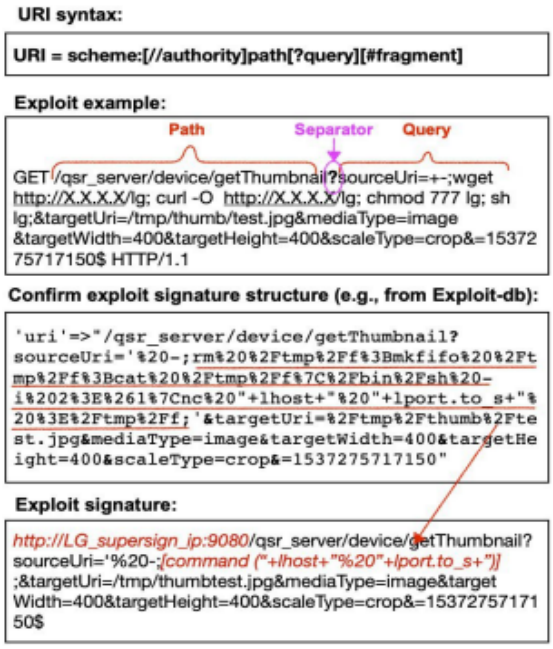

Figure 2: Example of a signature we generated for an exploit against CVE-2018-17173 [20, 40].

The second set of 11 binaries (10%) only comprised routines to accept orders from a command-and-control (C2) server and perform assaults; they did not have any infection vc tor in the code. UDP flood, SYN flood, ACK flood, TCP flood, UDP flood, VSE flood, DNS flood, GRE IP flood, GRE Ethernet flood, and HTTP flood are only some of the attack vectors that have been identified to be implemented using these commands [29]. Tsunami, Ordos, Hajime, and Singletons were the owners of these binary files. For the remaining 65 percent, or 70 binaries, we discovered 256 exploits aimed at online vulnerabilities, namely those relying on HTTP GET and POST requests. Based on the vulnerability description in NVD or Exploit-DB, we divided the flaws into six categories in Table 3: remote code execution (RCE), backdoors, command injection (CIA), buffer overflow, web application firewall (WAF) bypass, and brute force. Re mote Code Execution (RCE) was the infection vector for more than half of the vulnerabilities (55.62%). Similarly, Remote Code Execution (RCE) is the most commonly exploited vulnerability type in both the Uraeus and honeypot datasets (55.9% and 53.65%, respectively). CIA was the most popular infection vector, accounting for 56.25 percent of all infections across all three datasets that exploited the same vulnerabilities.

Utilize Your Life Expectancy as Much as Possible

Over time, both the quantity of IoT vulnerabilities and the frequency with which they are exploited have become steadily higher. Following the

procedure outlined in Section 2.3, we combed through the Genealogy dataset looking for exploit signature matches (2015–2018). There was a match in the Genealogy dataset for 17 attack signatures (representing 16 vulnerabilities) out of a total of 64. This limited presence of the vulnerabilities in earlier binaries is shown in Figure 3 as one possible explanation. After August 2018, when the Genealogy dataset's gathering period ended, 32 vulnerabilities (47%) were disclosed to the public. Although there is progress, the Genealogy dataset is still missing 15 attacks that target already patched vulnerabilities. Assuming the Genealogy dataset is typical of the time period in question, this means that programmers of newer malware are choosing vulnerabilities that were revealed many years ago. Out of a total of 6,752 binaries, we identified matches in 5,421 samples, or 80%. As the repository's creators admit, there are a lot of packed and end coded samples in this dataset, which may explain why none of them match the remaining 20%. This restriction is discussed further in Section 6. Using the larger amount of time, we were able to collect from the Genealogy dataset, we looked at the exploits' lifetimes, or the amount of time between the first and final time an exploit was spotted. We also look at time-to-exploit, or how long it takes from when a vulnerability is disclosed to when an exploitable binary is first seen in the wild. We gathered Virus Total's "first seen" date for all binaries.

Table 2: Number of hits (occurrence), exploits, and vulneraryabilities per year

Year	# Occurrences	# Exploits	# Vulnerabilities
2017	46	10	8
2018	727	15	15
2019	376	26	27
2020	1,855	58	63

Exploit lifetimes are shown in Figure 2 by plotting the dates of vulnerability disclosure (black X) and exploit code publishing against the number of times the exploit was seen in binaries (coloured dots) (red circle). CVE-2013-7471 is only one example of a CVE whose ID was captured years before the official publication date of 2019-06-11. We found that the publication date does not match the CVE ID in four other CVEs (CVE-2020-1956, CVE-2018-20841, and CVE-2019-2725). This might be the reason why the publication date of the exploit sometimes predates the publication date of the corresponding vulnerability. Twenty exploits were released before the official vulnerability disclosure date, albeit these dates were often close together and may represent mistakes in the underlying data rather than the actual chronology of events. Malware families' development is also seen in Figure 2. In order to brute force, all binaries introduced between 2015 and 2016 rely only on information stored in a text file. Out of a total of 5,421, 4,091 were binaries. The fact that the Mirai code was made public in November 2016 [22] may have helped to solidify this. To the IoT malware world, Mirai was a giant step forward since it was the first botnet to successfully gather millions of infected devices. Consequently, it appears that additional versions and families followed suit with Mirai's reliance on brute force hard-coded credentials. Ever since then, brute force has been there in binaries right up until the most current information became available.

DISCUSSION

Once Mirai began trying to infect systems by brute-forcing default (or weak) credentials, the number of vulnerabilities available to it quickly multiplied. Our results paint a more frightening picture than the previous study [3, which detected 25 exploits]. We found 68 new exploits and 68 new vulnerabilities that were specifically targeted, and we showed that the development of exploits is increasing momentum. Exploits and targeted vulnerabilities have nearly quadrupled every year since 2017. Our research also sheds light on the perpetrators' mindset and methodology. Approximately half of all vulnerabilities are exploited for at least two years, whereas the other half follow a pattern of brief usage followed by abandonment. The latter may indicate a habit of making mistakes when learning. The assaults will continue if the exploit code is effective at enlisting bots. As an assault drags on, the exploit code is more likely to be shared amongst different families and groups. Then, for years, the vulnerability is continuously targeted by attackers. The fact that attackers deliberately choose their weak spots is another another intriguing discovery. Unlike those who create malware for desktop OSes or server software, those who create malware for IoT devices prefer to exploit outdated flaws. Researchers have discovered that the latter group targets the version of software that is only one patch release behind in order to exploit the most recent vulnerabilities [58]. The time-to-exploit, or the time between the publication of a vulnerability and the first observation of a binary attacking that vulnerability, can be as little as one day (e.g., "Exploit Wednesday" following on Microsoft's "Patch Tuesday") or as long as a few months for a few high-profile attacks like Wannacry and Not-Petya [18, 63]. Given that the vast majority of exposed machines are still on the previous-to-last software version, this approach makes logical.

CONCLUSION

In this work, we conducted the first longitudinal measurement research to use numerous

perspectives in order to examine the dynamics of the IoT malware ecosystem. We used static analysis, dynamic analysis, and signature matching to identify and remove 63 unique vulnerabilities from 17,720 binaries representing 26 distinct families of Internet of Things malware. With our findings, we can see that the ecosystem has evolved from relying just on brute-force techniques to include many other types of vulnerabilities tailored to various devices. Since its beginning in 2016, the Mirai family has shown the greatest innovation and evolution. This was initially seen in Mirai, where most vulnerabilities were first spotted. The complexity of IoT malware as a whole rose with the number of IoT devices and protocols that were targeted. Rapid change is occurring: exploits and targeted vulnerabilities have increased annually since 2017. Exploits, once created, are seldom forgotten. Many are still accessible in the most up-to-date binaries. Our adventures have a lifetime of almost 5 years, although only lasting an average of 38 months. Vulnerabilities of any age are fair game for attackers. Although exploits have widely varying time to exploit windows, the average period between the disclosure of a vulnerability and the first appearance of an exploit in a binary is 29 months. That's not at all like the malware patterns we see aimed at computers and servers. Assuming this new approach to attacking the Internet of Things is sound, our data reveals that the devices being attacked are seldom, if ever, updated. The time available to exploit a flaw thus decreases slowly over time. Attackers care more about the device's instal base and how simple it is to construct exploit vectors than the vulnerability's age. Once created, they will be used for a long time. It is obvious from our research that attackers are exploiting the many holes in the IoT ecosystem, including its lack of patching and the wide variety of devices and manufacturers, which is now thought to number

more than 14,000 separate businesses [30]. The number of potential victims is high since each device

its own unique routes and dead ends on the road to protecting itself from malware. Several customers, ISPs, and companies in the manufacturing sector are directly affected by our results.

REFERENCES

[1] Luca Allodi. 2017. Economic Factors of Vulnerability Trade and Exploitation. In Proceedings of the ACM SIGSAC Conf. on Computer and Communications Security.

[2] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In Proceedings of the IEEE Symposium on Security and Privacy (S&P). https://doi.org/10.1109/SP.2019.00013

[3] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Ryan Court, Kevin Snow, Fabian Monrose, and Manos Antonakakis. 2021. The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle. In 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, 3505–3522.

[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the Mirai Botnet. In Proceedings of the USENIX Security Symposium. 1093–1110.

[5] Afsah Anwar, Jinchun Choi, Abdulrahman Alabduljabbar, Hisham Alasmary, Jeffrey Spaulding, An Wang, Songqing Chen, DaeHun Nyang, Amro Awad, and David Mohaisen. 2021. Understanding Internet of Things Malware by Analyzing Endpoints in their Static Artifacts. arXiv preprint arXiv:2103.14217 (2021).

[6] Grzegorz J Blinowski and Paweł Piotrowski. 2020. CVE Based Classification of Vulnerable IoT Systems. In Proceedings of the International Conference on Dependability and Complex Systems (DepCoS). 82–93.

[7] Brennen Bouwmeester, Elsa Rodríguez, Carlos Gañán, Michel van Eeten, and Simon Parkin. 2021. "The Thing Doesn't Have a Name": Learning from Emergent Real-World Interventions in Smart Home Security. In Proceedings of the USENIX Symposium on Usable Privacy and Security (SOUPS). USENIX Association.

[8] Orçun Çetin, Carlos Gañán, Lisette Altena, Takahiro Kasama, Daisuke Inoue, Kazuki Tamiya, Ying Tie, Katsunari Yoshioka, and Michel van Eeten. 2019. Cleaning Up the Internet of Evil Things: Real-World Evidence on ISP and Consumer Efforts to Remove Mirai. (2019). https://doi.org/10.14722/ndss.2019.23438

[9] Orçun Çetin, Carlos Ganán, Lisette Altena, Samaneh Tajalizadehkhoob, and Michel Van Eeten. 2019. Tell Me You Fixed It: Evaluating Vulnerability Notifications via Quarantine Networks. In 2019 IEEE European Symposium on Security.and Privacy (EuroS&P). IEEE, 326–339.

[10] Jinchun Choi, Afsah Anwar, Hisham Alasmary, Jeffrey Spaulding, DaeHun Nyang, and Aziz Mohaisen. 2019. IoT Malware Ecosystem in the Wild: A Glimpse into Analysis and Exposures. In Proceedings of the ACM/IEEE Symposium on Edge Computing (SEC). 413–418.

[11] Andrei Costin and Jonas Zaddach. 2018. IoT Malware: Comprehensive Survey, Analysis Framework and Case Studies. BlackHat USA (2018).

[12] Emanuele Cozzi. 2021. The Tangled Genealogy of IoT Malware Dataset. https: //github.com/eurecom-s3/tangled_iot/tree/master/dataset

[13] Emanuele Cozzi, Sophia Antipolis, France Pierre-Antoine Vervier France Matteo Dell, France Yun Shen, Leyla Bilge, Davide Balzarotti, Pierre-Antoine Vervier, Matteo Dell, and Yun Shen. 2020. The Tangled Genealogy of IoT Malware. In Proceedings of the Annual Computer Security Applications Conference (ACSAC).

[14] Emanuele Cozzi, Mariano Graziano, Yanick Fratantonio, and Davide Balzarotti. 2018. Understanding Linux Malware. In Proceedings of the IEEE Symposium on Security and Privacy (S&P), Vol. 2018-May. 161–175.

[15] CyberIOCs. 2020. CyberIOCs Daily malware pack. https://freeiocs.cyberiocs.pro

[16] Fan Dang, Zhenhua Li, Yunhao Liu, Ennan Zhai, Qi Alfred Chen, Tianyin Xu, Yan Chen, and Jingyu Yang. 2019.

Understanding Fileless Attacks on Linux-based IoT Devices with Honeycloud. In Proceedings of the Annual International Conference on Mobile Systems, Applications, and Services (MobiSys). 482–493.

[17] Antoine d'Estalenx and Carlos Gañán. 2021. NURSE: eNd-UseR IoT malware detection tool for Smart homEs. In Proceedings of the 11th International Conference on the Internet of Things (IoT '21). 1–8.

[18] Michel Edkrantz, Staffan Truvé, and Alan Said. 2015. Predicting Vulnerability Exploits in the Wild. In Proceedings on the IEEE International Conference on Cyber Security and Cloud Computing. 513–514.

.

[19] Exploit-db. 2009. Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers. Retrieved June 15, 2021 from https://www.exploit-db.com/

[20] Alejandro Fanjul. 2018. LG SuperSign EZ CMS 2.5. Retrieved May 01, 2021 from https://www.exploit-db.com/exploits/45448

[21] Xuan Feng, Xiao Liao, X Wang, Qiang Li, Kai Yang, Hong Zhu, and Limin Sun. 2019. Understanding and Securing Device Vulnerabilities through Automated Bug Report Analysis. In Proceedings of the USENIX Security Symposium.

[22] Jerry Gamblin. 2016. Mirai Source Code. https://github.com/jgamblin/MiraiSource-Code