Bin there, target that

Analyzing the target selection of IoT vulnerabilities in malware binaries

Al Alsadi, Arwa Abdulkarim; Sameshima, Kaichi; Yoshioka, Katsunari; van Eeten, Michel; Gañán, Carlos H.

DOI
10.1145/3607199.3607241

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2023

Bin there, target that: Analyzing the target selection of IoT vulnerabilities in malware binaries

Arwa Abdulkarim Al Alsadi
Delft University of Technology
Delft, The Netherlands
a.alsadi@tudelft.nl

Kaichi Sameshima
Yokohama National University
Yokohama, Japan
sameshima-kaichi-mx@ynu.jp

Katsunari Yoshioka
Yokohama National University
Yokohama, Japan
yoshioka@ynu.ac.jp

Michel van Eeten
Delft University of Technology
Delft, The Netherlands
M.J.G.vanEeten@tudelft.nl

Carlos H. Gañán
Delft University of Technology
Delft, The Netherlands
C.HernandezGanan@tudelft.nl

ABSTRACT

For years, attackers have exploited vulnerabilities in Internet of Things (IoT) devices. Previous research has examined target selection in cybercrime, but there has been little investigation into the factors that influence target selection in attacks on IoT. This study aims to better understand how attackers choose their targets by analyzing the frequency of specific exploits in 11,893 IoT malware binaries that were distributed between 2018–2021. Our findings indicate that 78% of these binary files did not specifically target IoT vulnerabilities but rather scanned the Internet for devices with weak authentication. To understand the usage of exploits in the remaining 2,629 binaries, we develop a theoretical model from relevant literature to examine the impact of four latent variables, i.e. exposure, vulnerability, exploitability, and patchability. We collect indicators to measure these variables and find that they can explain to a significant extent (R^2=0.38) why some vulnerabilities are more frequently exploited than others. The severity of vulnerabilities does not significantly increase the frequency with which they are targeted, while the presence of Proof-of-Concept exploit code does increase it. We also observe that the availability of a patch reduces the frequency of being targeted, yet that more complex patches are associated with higher frequency. In terms of exposure, more widespread device models are more likely to be targeted by exploits. We end with recommendations to disincentivize attackers from targeting vulnerabilities.

CCS CONCEPTS

• **Security and privacy** → **Malware and its mitigation**; **Vulnerability scanners**; • **Computer systems** → **Embedded systems**.

KEYWORDS

IoT malware; Dynamic Analysis; Exploits; Vulnerabilities; Exposure

RAID '23, October 16–18, 2023, Hong Kong, China

ACM ISBN 979-8-4007-0765-0/23/10.
https://doi.org/10.1145/3607199.3607241

ACM Reference Format:
Arwa Abdulkarim Al Alsadi, Kaichi Sameshima, Katsunari Yoshioka, Michel van Eeten, and Carlos H. Gañán. 2023. Bin there, target that: Analyzing the target selection of IoT vulnerabilities in malware binaries. In *The 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23), October 16–18, 2023, Hong Kong, China.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3607199.3607241

1 INTRODUCTION

The proliferation of Internet of Things (IoT) devices [12] has made them a prime target for cyber attackers looking to exploit devices and build up attack infrastructure [2]. The poor security features of many IoT devices are music to the attackers' ears, as they can easily compromise these devices that were never designed to repel complex attacks. In this abundance of devices, what drives an attacker's decision to target a particular device type?

Previous research on attacker target selection [20, 44] has shown that a *rational choice* approach is often used, with attackers selecting targets that are perceived to be easy to exploit, have known vulnerabilities, and offer a high reward. This has been observed in both physical burglaries and cybercrime, with offenders selecting accessible targets with known vulnerabilities. A new and more realistic attacker model [5] suggests that cyber attackers are not equally likely to exploit all possible vulnerabilities. Instead, attackers will choose to exploit only one vulnerability per software version, include only vulnerabilities with low attack complexity, and be slow at introducing new vulnerabilities into their arsenal. No research has yet tested empirically whether these factors also hold for target selection in IoT malware development. It is unclear whether attackers base their decisions on specific characteristics of the devices, their exposure, or other considerations.

In this paper, we aim to improve our understanding of how attackers choose their targets by studying the vulnerabilities and exploits used in IoT malware that was distributed between 2018–2021. Although 78% of these binaries only scanned the Internet for devices with weak authentication, the remaining 2,629 binaries were targeting specific vulnerabilities. We first develop a theoretical model based on relevant literature to understand the factors that influence the choice of exploits in IoT malware. We then relate these factors to four latent variables (exposure, vulnerability, exploitability, and patchability) and examine the impact on the usage of particular exploits, as measured by the frequency of specific exploit code in binaries observed in the wild.

By understanding these relationships, we aim to provide answers to this overarching question: *How can we explain the attackers' choices of IoT vulnerabilities as targets in malware binaries?*

To answer this question, we gather information about various aspects of IoT vulnerabilities, such as their publication date, the availability of proof-of-concept (PoC) exploit code, and the release dates of patches. Our findings indicate that certain types of vulnerabilities are more frequently exploited, yet this is *not* explained by the severity of the vulnerabilities. We also find that the availability of a patch decreases the exploitation of a particular vulnerability, while the existence of an PoC exploit increases it. Additionally, we find that more complex patches are related with higher exploit frequency. Our results also show that the more widespread a given device model is, the more likely it is to be targeted by exploits.

Our main contributions are as follows:

- We develop a theoretical model that posits that exploit usage is influenced by latent variables such as vulnerability, patchability, exploitability, and exposure, which can be inferred from observable indicators.
- We characterize the targeted IoT vulnerabilities in malware binaries by collecting and analyzing data on vulnerabilities, exploits, patching and device exposure information.
- We estimate a generalized linear model to understand how different factors, such as vulnerability disclosure, patch release dates, patch complexity, and the number of exposed devices on the Internet, influence target selection.
- We provide recommendations to decrease the exploitability of IoT devices in the wild.

2 THEORETICAL MODEL

When analyzing the exploit code malware binaries, it became apparent that some vulnerabilities are more often targeted than others. Some exploit code is only seen once in the wild, while other code has been in use for years. To help explain our main question of how attackers choose IoT vulnerabilities as targets in malware binaries, we will use the exploit frequency in binaries as our dependent variable.

While research in this area is limited, previous studies have investigated various factors and indicators that influence target selection in IoT vulnerabilities. For example, some studies have examined the relationship between attack frequency, vulnerability disclosure, and patching [9], while others studied the relationship between attacks and exposure [10]. We aim to bring together these indicators, and some new ones, in a comprehensive theoretical model.

While we can't always know the true intentions of attackers, we can build on work that assumes attackers maximize their benefit against the lowest possible cost. Allodi et al. [5] formalized this theoretical starting point in their "work-averse attacker" model. They posit that not all vulnerabilities will be attacked equally. The initial fixed costs of exploit development induce attackers to delay implementation and deployment of exploits of vulnerabilities. They found empirical validation for this model in analyzing Symantec's WINE attack signatures.

The key question then becomes: what factors impact the attackers' benefits and costs for exploiting IoT vulnerabilities? Extending the earlier theoretical work, we develop a model that explains the frequency with which IoT vulnerabilities are targeted from four factors that impact the cost and benefits to the attacker. First, the features of the *vulnerabilities* themselves, such as their severity, may make them more attractive to attackers. Second, *patchability*, may reduce the benefits. If a patch is deployed for a vulnerable IoT device, this may reduce the available attack surface that the exploit can successfully compromise. Third, *exploitability*, considers the availability and complexity of the Proof-of-Concept exploit code, which reduces the cost to the attacker. Fourth, *exposure*, considers the size of the install base of devices with the vulnerability. A larger install base increases the benefits to attackers of targeting that vulnerability.

These factors allow us to synthesize indicators from previous studies and study their relative influence. For example, the characteristics of vulnerabilities such as disclosure date, severity and type can be used as indicators to predict whether or how soon vulnerabilities are likely to be selected as a target by attackers [32] and how frequent [9]. Similarly, the availability of Proof-of-Concept (PoC) exploit code is correlated to the likelihood that attackers select the vulnerability for their attacks [21, 32], while the availability and the time of the release of patches [9] as well as the effort needed to deploy them [13] can all impact the attack surface [9].

Moreover, when exposed devices can be discovered via search engines, it increases the likelihood of these devices being targeted for vulnerabilities. [8, 10, 27]. In addition to the indicators from earlier work, we also collect new ones, most notably patch complexity and exploit complexity.

In sum, we developed a theoretical model, shown in Figure 1, that interprets these indicators as signals for the four underlying factors. This allows us to theoretically interpret the relationships. To the best of our knowledge, there is not yet a study that systematically measures the impact of these factors – and all of the associated indicators – on the target selection of vulnerabilities in IoT malware binaries. We now briefly explore each of the four factors in a bit more detail. (In Section 3 we will discuss the data sources we have used to measure the indicators for each factor, as summarized in Table 2.)

Vulnerability. Vulnerability refers to a weakness or flaw to exist in an IoT device that can be exploited by an attacker to gain unauthorized access or cause harm. As each vulnerability has its own characteristics, we want to understand which ones have the most impact on the attackers' choice to select as a target. We used five indicators of vulnerability to measure the impact on IoT exploit frequency in binaries. One of these indicators is the *vulnerability release date* in public, which can provide important information about the number of exploit frequency. In another words, the older a vulnerability is, the higher the potential number of frequency is as it can accumulate over time. We will be using it to measure the lifetime a vulnerability by counting the days since the vulnerability release date until last day of binaries collection.

For vulnerabilities with CVE-ID, we collected the *severity* as measured by CVSSv3 metric. The CVSSv3 scoring system is used to assign a numerical score to a vulnerability based on its characteristics and the potential impact if exploited. We only collected CVSSv3 since it is the most recent revision of CVSS scoring system for vulnerabilities published 2016 onwards, which corrected the

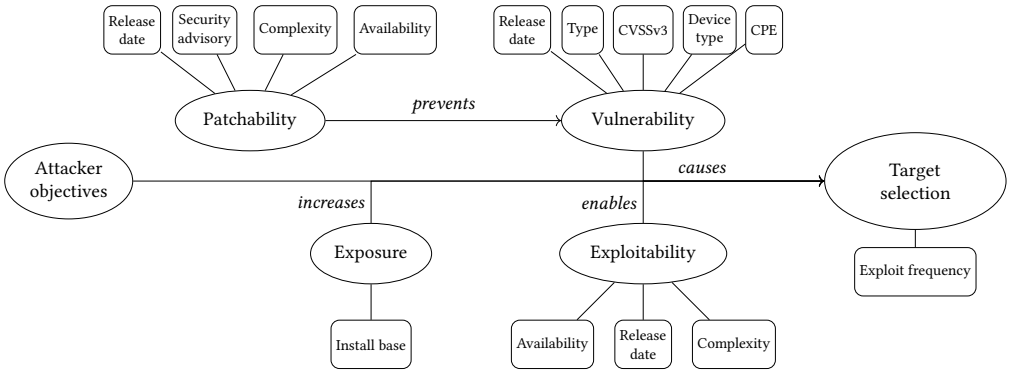

Figure 1: Theoretical Model for Exploit frequency in IoT Binaries.

shortcomings in older versions. The CVSSv3 score ranges from 0 to 10, with higher scores indicating a more severe vulnerability. Additionally, we also counted the number of affected systems using their corresponding *Common Platform Enumeration (CPE)* for each vulnerability. CPE is a naming scheme that structures the information about affected systems, software, and packages. Thus, it can help to identify which systems are affected by a vulnerability and which are not.

Next, we collected data on *device type* that is affected by the vulnerability, as some types of devices are more often targeted than others. We categorized the devices according to their use. Finally, we examined the *vulnerability type* to determine if some vulnerability types such as: Remote Code Execution, Command Injection and Password Overflow are more targeted than others.

Patchability. Patchability refers to the ease or difficulty of patching (fixing) a vulnerability in an IoT device in order to reduce the likelihood of a successful attack. Our hypothesis is that the patching of IoT devices can affect the frequency of exploits.

The availability of patches will lower the chances an attack succeeds, therefore, patchable devices will be targeted less in IoT binaries.

To test this, we use a range of four indicators to measure relevant patching properties. One such indicator is the *patch availability*. We assess this by determining whether patches are available on the vendor's website or on security blogs, which leads to our second indicator, the existence of a vendor's *security advisory*. We hypothesize that the existence of security advisory can lead to more patched devices and less vulnerable devices. This is because if manufacturers or vendors do not have effective public source dedicated to issue security advisories in place, users may be less likely to apply patches to their devices. Then, if patches are available, we evaluate their *complexity*, as this could be a factor in users not updating their devices. We do this by counting the number of steps required to apply the patch listed in the security advisory, as we believe that the more steps involved, the less likely a user is successful or even willing to install the update. Finally, is the *patch release date* provided by vendors, which can serve as a proxy for the amount number of exploit number to measure whether the release time of a patch has an impact on the number of exploits showing in binaries. We use the publication date to measure the patch lifetime by counting the days between the patch publish date and last day of binaries collection.

Exploitability. Assuming that attackers minimize effort, available and complexity of PoC exploit code would increase the odds of exploit code showing up in binaries. We use three simple indicators to measure the exploitability of a particular vulnerability. First, *exploit availability*, where we determine whether the exploit we found in binaries has PoC exploit available in public repositories. If so, we collect information of our second indicator, *exploit complexity*. We crudely approximate this by counting the number of lines of the PoC exploit code, assuming that an exploit with more code is most costly to use successfully by attackers. Lastly, *exploit publish date*, which refers to the publish date where the PoC exploit code was released. We also use the publication date to measure the exploit lifetime by counting the days between the exploit publish date and last day of binaries collection.

Exposure. Exposure refers to the size of the discoverable attack surface that an attacker might target with an exploit. We hypothesize that the larger the *install base* is of devices exposed to Internet-wide scans or public search engines (e.g., Shodan [28]), the higher the chances an attacker would target them.

Table 1: Overview of data collection work process.

Automated	Semi-automated	Manual
Binary collection	Vulnerability release date	Exploit signature generation
Exploit extraction	Patch availability	Mapping exploit to vulnerability
Severity (CVSSv3)	Security advisory	Installed base
CPE	Exploit availability	Device type
	Exploit release date	Vulnerability type
		Patch release date
		Patch complexity
		Exploit complexity

3 METHODOLOGY

Our methodology involves two main steps in order to explain the target selection of IoT vulnerabilities using exploit frequency in binaries as our dependant variable. First, we gather the indicators, including information on the exploit code in the malware binaries, the vulnerabilities they exploit, the patch status of the affected devices, and the level of exposure of the devices (see Table 1 for

Table 2: Description of variables.

Variable	Description	Data sources
Installed base	Number of IoT devices exposed in public	Shodan
Vulnerability type (CI)	A dummy variable that denotes if the vulnerability type is command injection	NVD
Vulnerability type (RCE)	A dummy variable that denotes if the vulnerability type is remote code execution	NVD
Vulnerability type (other)	A dummy variable that denotes for other the vulnerability type	NVD
Device type (router)	A dummy variable that denotes if the device type is router	NVD
Device type (surveillance)	A dummy variable that denotes if the device type is surveillance	NVD
Device type (Other)	A dummy variable that denotes for other device type	NVD
Severity (CVSSv3)	A value between 0 and 10 that measures the impact of exploiting the vulnerability	NVD
CPE	Number of affected systems, software and packages	NVD
Vulnerability lifetime	Number of elapsed days from the vulnerability published date until the last capture day of binaries	NVD
Patching lifetime	Number of elapsed days from the patching release date until the last capture day of binaries	Security advisory
Patch complexity	Number of steps a user needs to patch a device	Security advisory
Patch availability	A dummy variable that denotes whether a patch is available or not	NVD, Security advisory
Security advisory	A dummy variable that denotes whether a vendor has released a security note for a vulnerability or not	Vendors' website
Exploit availability	A dummy variable that denotes whether an exploit code is available or not	Exploit-db, Github, Security reports
Exploit complexity	Number of lines for exploit code	Exploit-db, Github, Security reports
Exploit lifetime	Number of elapsed days from the PoC exploit code publish date until the last capture day of binaries	Exploit-db, Github, Security reports

data collection work process). Second, we use a count regression model to analyze and quantify the significance of each indicator. The output of the model provides insights into the factors that influence the decisions of attackers in selecting what exploit code to include in IoT binaries, allowing us to better understand the threat landscape and make recommendations for improving the security of IoT systems.

3.1 Data collection

In this section, we will discuss the data collection work process we used to measure the indicators for each factor, as summarized in Table 1.

3.1.1 IoT exploits in binaries. In our study, we collected IoT malware samples from an IoT-specific honeypot over a period of four years (2018 – 2021). We used dynamic analysis to extract exploits from these samples and map them to vulnerabilities.

Collecting IoT malware binaries. We obtained 11,893 binaries from September 2018 to September 2021 via IoTPOT [39].

The present version of IoTPOT is a high-interaction honeypot using bare-metal vulnerable IoT devices running Telnet and/or HTTP as Web User Interface. For the entire observation period, the honeypot utilized eight devices (three routers, two WiFi storage devices, an IP camera, a printer, and a satellite decoder) connected to 64 IP addresses in Japan. Additionally, from September 2018 to February 2020, the honeypot utilized seven more devices (five IP cameras, a router, and a WiFi storage device) connected to 32 IP addresses in Japan. Although the devices connected to our high-interaction honeypot may not represent the overall population of IoT devices on the internet, the honeypot can still serve our research objectives and generate insights into the population of IoT exploits used in the wild. Therefore, we believe that the set of binaries collected over a period of fours years captures the essence of exploits targeting vulnerable IoT devices, not only in Japan but globally.

Extracting exploits. We perform dynamic analysis on all the 11,893 collected honeypot binaries. In a sandbox, we executed each of the binaries, which is built as a virtual machine running Linux Debian for the MIPS and ARM architectures. With the exception of DNS resolutions, we run each binary for five minutes in a closed network environment. As malware repeatedly tries to connect to the C&C server in such an environment whereas port scans on the same host are typically not repeated, this isolated environment help us in differentiating between port scans and C&C communications, especially when they are on the same port. In addition, if the malware on a destination port accesses more than 100 destination IP addresses, we conclude that the port is being scanned. Then, we execute all binaries again for five minutes with dummy servers that pretend to be the target on the scanned ports once the sandbox has been cleaned up. With the implementation of PyNetSim, we redirect all connection attempts on the examined port to the dummy servers, which merely establishes TCP sessions with no further response. We found that a significant number of binaries actually begin scanning for and exploiting vulnerabilities immediately after execution, without contacting C&C servers or their intended targets.

During the dynamic analysis of the 11,893 binaries, 2,256 of them did not show any scanning (propagating) behavior. There can be different reasons why they did not show propagation: they simply did not have any propagating capabilities; they might have needed a trigger, such as a command from C&C server; or they might not have been executed successfully [2, 6]. Given that we could not detect exploits, we dropped these 2,256 binaries from further analysis. In the end, we had dynamic analysis results for 9,637 samples with propagating behavior. Of this set, 7,008 binaries conducted scans on only Telnet ports, such as 23, 2323, and 2223/TCP, to try brute-forcing with known credentials. Since this general attack vector is not tied to a specific CVE or known vulnerability, we did not include these binaries in the model. For the remaining 2,629 binaries, we found evidence of targeting specific CVE or known vulnerability via 48 different ports and various protocols. All of the detected protocols, such as CPE WAN Management Protocol (CWMP) and Simple Object Access Protocol (SOAP), are HTTP-based in terms of the transport layer, which is consistent with the findings in [7, 22, 40].

Exploit signature generation. For each payload captured from the dynamic analysis, we manually extract HTTP-based request lines and HTTP headers to create a signature of a distinctive substring, such as a destination resource of HTTP requests, which can be used to identify an exploit. We use the URI (Uniform Resource Identifier) syntax or absolute path of the `GET` or `POST` in the request line to generate the unique signature. In Figure 2, we present an example of an exploit payload. As can be seen in this figure, the URI syntax consists of four components (colored in red): scheme, authority, path, query and fragment. The scheme part is followed by a colon and two forward slashes, if specified. Then comes the path name that begins with a single forward slash. Next, the query string preceded by a question mark. Finally, the fragment, which is a set of characters for resource that is subordinate to another.
However, the most common format is: the path name, question mark and the query string. We use this URI syntax during the extraction of each request. In this example, the final exploit signature is `/backupmgt/localJob.php?session=fail`.

```
GET /backupmgt/localJob.php?session=fail;cd+/tmp;wget
+http://212.192.241.72/lolol.sh;curl+O+http://212.192.241.72/lolol.sh; sh+lolol.sh
HTTP/1.1
Connection: close
Accept-Encoding: gzip, deflate
Accept: /
User-Agent: Dark
```

Figure 2: Example exploit payload

Using this signature generation method, we created 59 unique signatures from the captured HTTP requests. To identify the vulnerabilities that each one of these exploit targeted, we search them in public exploit databases such as Exploit-db [41] or Github[47]. From these two sources, we were able to identify 45 out of the 59 signatures. The remaining 14 signatures with no information in the above sources, we use other public sources such as reports from AV vendors and researchers that describe them. For every signatures that were confirmed to be used as an exploit code to target vulnerability, we tag them as *Known exploit*. Otherwise, we would tag them as *Unknown payloads*. However, all the signatures that we generated were found in the public sources meaning that they were all tagged as known exploit.
Apart from using signatures to identify exploits, we also used them to allow us to count how often each exploit appears across the dataset. We call this 'exploit frequency', our dependent variable. It is meant to capture the more prevalent exploits among attackers, hence the vulnerabilities they prefer to target.

3.1.2 Characterizing IoT exploits. After we confirmed that these signatures belonged to known exploits according to the above public sources, we collect information about their corresponding vulnerabilities and their proof of concept (PoC) exploit codes. Next, we characterized these vulnerabilities using the National Vulnerability Database (NVD [33]). We then collected data on the patch information using the vendor's security advisories.

Mapping exploit to vulnerabilities. We manually mapped exploits to their corresponding vulnerabilities using the signatures to search in Exploit-db, Github or Google Search. For example, for each exploit entry in Exploit-db, there is a CVE field we can retrieve the vulnerability information, if available. Otherwise, we used the vulnerability description title in Exploit-db, e.g., "OptiLink ONT1GEW GPON version 2.1.11_X101 RCE" [42]. We applied the same approach for the other two sources. We found that the 59 unique exploits were mapped to 64 vulnerabilities. Next, we checked whether a proof of concept (PoC) exploit code was available for them. If so, we retrieved the date when the PoC exploit code was released and measure its complexity through counting the lines of code.

Exploit coverage. At the heart of our analysis is frequency data: how often is a vulnerability attacked in the wild? For this, we needed to capture binaries in live attacks on our honeypot infrastructure. The number of times that an exploit was present in these captured binaries tells us how often a vulnerability was targeted. This means we can't supplement the data with binaries from an existing repositories of binaries, since those data sources lack the frequency with which the binaries were observed in live attacks. The honeypot infrastructure was running in Japan. This location might impact the observations. That said, we did not observe any binaries targeting specific network ranges. Rather, scanning was traversing randomly through the IPv4 space. To assess the coverage of our dataset with previous work, we compared the overlap of exploits with datasets used in [2] and [6]. Alsadi et al. [2] extracted exploits using both static and dynamic analysis. Therefore, it does not have the limitations of relying exclusively on dynamic analysis to identify exploit code, so it provides a good point of comparison. We observe that our set of exploits is consistent with their datasets, namely UrlHaus (Jul 2020 - Oct 2020), IoTPoT (Sep 2018 - Aug 2020), and the Genealogy (Jan 2015 - Aug 2018). Our results showed that we covered 55.17% (32 out of 58) of the exploits in UrlHaus, all exploits in IoTPoT, and 80% (12 out of 15) in the Genealogy datasets. To compare against Alrawi et al. [6], we looked at the top 25 IoT exploits reported in their study. Out of the 25 exploits, 17 list a CVE-ID. Of these 17, 14 are also in our set, 3 are not. The remaining 8 exploits have no CVE-ID or affected product listed, so we cannot compare them to our dataset. So although we can only conduct a limited comparison, for the vulnerabilities with a CVE-ID, we have a high overlap with their dataset.

Characterizing IoT vulnerabilities. To collect information on the 64 vulnerabilities targeted by the captured exploits, we used the National Vulnerability Database (NVD). It publishes Common Vulnerabilities and Exposures (CVEs), their associated identification number, description, public references, and severity score. In case, a vulnerability is reserved in the NVD, e.g., due to pending CVE status, we used the MITRE website [30] to collect the information. However, 15 of the 64 vulnerabilities were not registered in NVD, so they had no CVE-ID [2, 6]. For those, we used the vulnerability description title in the exploit databases.

Identifying patches. For the identified IoT vulnerabilities, we collect patch information that has been made available by the device manufacturer through security advisories. Security advisories are documents that provide information about vulnerabilities in products, including Internet of Things (IoT) devices. They are typically issued by the manufacturer or vendor of the device in question, or by independent organizations or government agencies that have

discovered a vulnerability. These advisories normally include brief headers that list the vulnerability title, advisory release date, affected products, advisory ID, and any relevant CVE ID. We found them on the website of the manufacturer or vendor, or on the website of the organization or agency issuing the advisory. In this study, we used the security advisory pages of vendors. Specifically, we gathered patch information for each vulnerability, including existence of advisory notes by vendors, patch release dates, availability and patch complexity.

3.1.3 Measuring device exposure. Vulnerabilities are associated with certain device models. To measure the exposure of these targeted IoT device models, we use search engines to identify the number of Internet-connected devices and findable devices of each model. This can help us understand the potential scale of exploiting a certain vulnerability. By searching for the specific model numbers of vulnerable devices, we can approximate the size of the install base for those devices. Shodan is unlikely to provide us with a very accurate absolute number of publicly reachable devices of a certain type, but what we need is a proxy of the install base of each device relative to the other targeted devices.

Estimating device installed base. We use Shodan [28] to determine the quantity of Internet-connected devices for a particular vulnerable model. Shodan allows us to search for specific types of devices, such as servers, routers, and other Internet of Things (IoT) devices, and retrieve information about them. By querying Shodan, we gather information about devices that are publicly available on the Internet, including their metadata and any web interfaces they may have.

To measure the install base of vulnerable IoT devices, we developed a systematic querying approach (see Figure 3) that involves using the vendor or device name as a starting point, then manually checking the search results and inspecting the HTML field to confirm if the search results match the actual targeted device. We also looked for other signatures that contain the model name or the affected software version a given device is running. We then filtered the results based on the device's installed software version, if available.

For example, some banners include a server name that shows the device-specific, and then we search again in Shodan using these signatures. Through experimentation and testing, we found that approximately 57% of the devices can be found using the device name approach, while the remaining cases can be found using the server name approach.

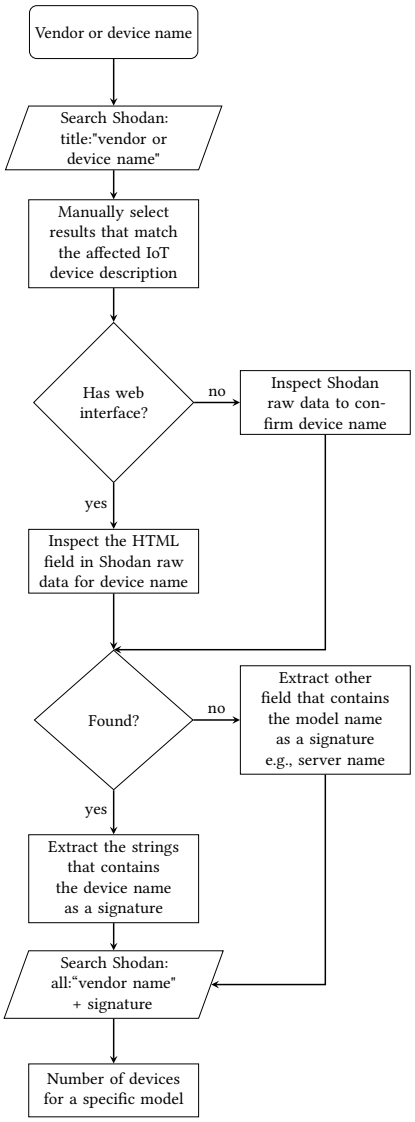

Figure 3: Flowchart to determine the number of publicly accessible IoT devices using Shodan.

3.2 Data processing

To prepare for analysis and modelling, we clean up the data, transforming it in a way to make it suitable for the analysis. We used the CVE-ID or the vulnerability name as the unit of analysis to count the frequency for each exploit. The relationship between exploits and vulnerabilities is often one-to-one, but not always. It was one-to-one for 36 out of 49 observations. Where it is not one-to-one, this creates an issue of creating duplicate data in the observations for the model.

We encountered three scenarios that were not one-to-one. First, sometimes we encountered multiple exploits targeting the same vulnerability. If we would count each of these combinations as a separate observation, we would get observations where the vulnerability properties are copied, hence feeding the model with artificially duplicate data while the model assumes these are all independent datapoints. This is obviously not allowed. Therefore, for this first scenario, we merge the frequency counts for each exploit and then connect those with the one vulnerability. We investigated this further and found that in all these cases, the different exploit codes were coming from the same PoC. So they were already linked, if not basically the same exploit to begin with. So merging their counts makes sense. This scenario holds for six of our observations.

The second scenario is where we have one exploit targeting multiple vulnerabilities. Here we merge the vulnerabilities into one observation. Since they have different publish dates, we picked the oldest date amongst them to cover the whole life cycle of such case.

This scenario holds for four of our 49 observations. Since multiple vulnerabilities also have different CVSS3 base scores, we use the highest severity score.

The third scenario is where we found multiple exploits targeting multiple vulnerabilities. This case is basically a mix of previous two scenarios: some exploits stem from the same PoC and that PoC actually targets multiple vulnerabilities. In those cases, we merged both the exploit frequency counts as well as the vulnerabilities. Again, we picked the oldest vulnerability publish date and the highest CVSS3 score for the observation. This scenario holds for three of our 49 observations.

A final bit of data processing concerns the size of the install base. Since the size of the observed install bases spans multiple orders of magnitude, we log-transformed the indicator to approximate a normal distribution that the model can work with.

3.3 Modeling approach

We used a general linear model (GLM) as it is a commonly used statistical tool for studying the relationship between a dependent outcome variable and one or more independent explanatory variables. In the context of this study, where the dependent variable is a count of the number of times a certain exploit code was observed in a binary, using a GLM allows us to investigate the influence of multiple factors on this outcome. The dependent variable (exploit frequency) will be a non-negative integer. Previous research suggests that the Poisson regression model is a suitable option for modeling this type of data (i.e., discrete, non-negative, and sporadic) [45]. However, a major limitation of the Poisson regression model is that the variance of the dependent variable must be equal to its mean. This is not our case as the variance is more than hundred times larger than the mean. When the mean and variance of the data are not equal, the estimated Poisson model coefficients tend to be underestimated and biased. This limitation can be addressed by using the negative binomial distribution, which is well-suited for describing discrete, non-negative events, and does not have the requirement that the mean must be equal to the variance. We estimate the negative binomial dispersion parameter using maximum likelihood, which is the technique that is most often used, and usually provides a slightly better estimate and smaller standard deviation than other commonly used estimators. The density function of the negative binomial model is given by:

$$f(y|x,\beta) = \frac{\Gamma(y+r)}{\Gamma(y+1)\Gamma(r)} \left(\frac{1}{1+\frac{1}{r}\cdot x^{\top}\beta}\right)^{y} \left(\frac{1}{1+\frac{1}{r}\cdot x^{\top}\beta}\right)^{r}$$

whereby in our case, y is the count of exploit frequency, x is the vector of explanatory variables, β is the vector of model coefficients, and r is a dispersion parameter for the negative binomial family. The function Γ is the gamma function, which is defined as $\Gamma(n) = (n-1)!$ for positive integers n and has a more general definition for other values of n.

The j_{th} explanatory variable can be tested for its significance in explaining the frequency of an exploit by examining the respective coefficient β_j. However, it is necessary to check if the variable is already represented by other factors, as redundant variables can result from high collinearity with another variable. The variance inflation factor (VIF) can be used to quantify this redundancy and is defined as:

$$VIF_j = \frac{1}{1-R_j^2}$$

where R_j^2 is the R^2 in a model containing x_j as the explanatory variable only. O'Brien proposed [37] to indicate multicollinearity $VIF(\beta_j) > 5$ or $VIF(\beta_j) > 10$. In this paper, we chose a threshold of 5 as VIF values exceeding this level indicate the presence of multicollinearity issues. [17, 23, 29]. There are some approaches for designing a measurement for the fitness of the GLMs. The R^2 can only be computed in a linear regression model, so most coefficients of determination work with the likelihood function. McFadden's R^2 is chosen for the present analysis and defined by:

$$R^2 = 1 - \frac{LL_{full}}{LL_{null}}$$

where LL_{full} is the log-likelihood of the full model and LL_{null} is the log-likelihood of the null model (a model with no explanatory variables).

4 DESCRIPTIVE FINDINGS

We present the factors and their indicators, which we hypothesized will explain the vulnerability target selection in IoT malware binaries. Table 3 presents a summary of descriptive statistics for the indicators.

4.1 Distribution of exploit frequency in binaries

The exploit frequency distribution, as shown in Figure 4, ranges from a minimum of 1 to a maximum of 2,177 times the same exploit was seen in the 2,629 binaries. This results in a highly skewed distribution. The average frequency is approximately 169, while the median is 23. We encountered 12 exploits (24.5%) appearing only once in the binaries. All but one of these 12 exploits targeted a unique vulnerability that was not targeted ever again in another binary.

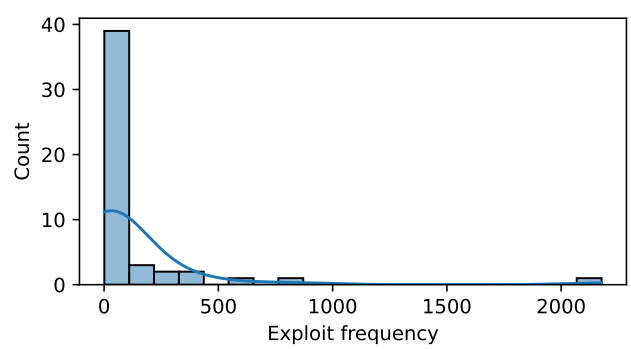

Figure 4: Distribution of exploits frequency.

Figure 5 presents the time series of the exploit frequency of the top five most frequently exploited vulnerabilities. The vulnerability with the highest frequency (2,177 occurrences) was CVE-2017-17215 [35]. Despite the release of a patch around the time of the vulnerability's discovery, it continued to be targeted in binaries. The second highest was CVE-2014-8361 [34], which was targeted by three different exploits that all came from the same PoC exploit code. This exploit was released eight months after the publication of the vulnerability in October 2014. The devices affected by

Table 3: Descriptive statistics per indicator.

Indicator	Stats / Values	Freqs (%)
Installed_Base [numeric]	Mean (sd) : 37098.1 (217960.2) min < med < max: 2 < 485 < 1527377 IQR (CV) : 2819 (5.9)	≤ 35: 13 (26.5%) ≤ 572: 26 (51.0%) ≤5108: 37 (75.5%)
Vulnerability_Published [logical]	1. FALSE 2. TRUE	15 (30.6%) 34 (69.4%)
Vulnerability_type [factor]	1. A Password Overflow Issue 2. Backdoor 3. Command Injection 4. RCE	1 (2.0%) 2 (4.1%) 19 (38.8%) 27 (55.1%)
Device_Type [factor]	1. NAS 2. Router 3. Surveillance 4. TV 5. Web app 6. Web server	3 (6.1%) 28 (57.1%) 7 (14.3%) 1 (2.0%) 7 (14.3%) 3 (6.1%)
Severity_CVSS3 [numeric]	Mean (sd) : 6.1 (4.5) min < med < max: 0 < 8.8 < 10 IQR (CV) : 9.8 (0.7)	0.0 : 17 (34.7%) 7.2 : 2 (4.1%) 7.5 : 1 (2.0%) 8.8 : 7 (14.3%) 9.8 : 21 (42.9%) 10.0 : 1 (2.0%)
CPE [numeric]	Mean (sd) : 1.9 (4.4) min < med < max: 0 < 1 < 27 IQR (CV) : 1 (2.3)	0 : 16 (32.7%) 1 : 21 (42.9%) 2 : 5 (10.2%) 3 : 3 (6.1%) 5 : 1 (2.0%) 6 : 1 (2.0%) 16 : 1 (2.0%) 27 : 1 (2.0%)
Vulnerability Lifetime [numeric]	Mean (sd) : 1373 (971.6) min < med < max: 49 < 1222 < 4614 IQR (CV) : 1217 (0.7)	≤ 660: 13 (26.5%) ≤1222: 26 (51.0%) ≤1877: 37 (75.5%)
Patch_Availability [logical]	1. FALSE 2. TRUE	25 (51.0%) 24 (49.0%)
Security_Advisory [logical]	1. FALSE 2. TRUE	24 (49.0%) 25 (51.0%)
Patch_Complexity [numeric]	Mean (sd) : 2.1 (3.7) min < med < max: 0 < 1 < 13 IQR (CV) : 3 (1.8)	≤ 1: 35 (71.4%) ≤ 4: 41 (83.7%) ≤ 8: 48 (98.0%)
Patch_Lifetime [numeric]	Mean (sd) : 1010.9 (983.3) min < med < max: 17 < 907 < 3045 IQR (CV) : 753 (0.7)	≤ 66: 3 (12.0%) ≤554: 6 (24.0%) ≤907: 13 (52.0%)
Exploit_Availability [logical]	1. FALSE 2. TRUE	1 (2.0%) 48 (97.9%)
Exploit_Complexity [numeric]	Mean (sd) : 76.4 (67.7) min < med < max: 0 < 69 < 317 IQR (CV) : 81 (0.9)	≤ 21: 13 (26.5%) ≤ 69: 26 (51.0%) ≤101: 37 (75.5%)
Exploit_Lifetime [numeric]	Mean (sd) : 1308 (983.3) min < med < max: -27 < 1165 < 4412 IQR (CV) : 1161 (0.7)	≤ 556: 13 (26.5%) ≤1165: 26 (51.0%) ≤1717: 37 (75.5%)

CVE-2014-8361 have reached their End-of-Life, meaning that the vendor will no longer provide software upgrades or address any new vulnerabilities discovered in them.

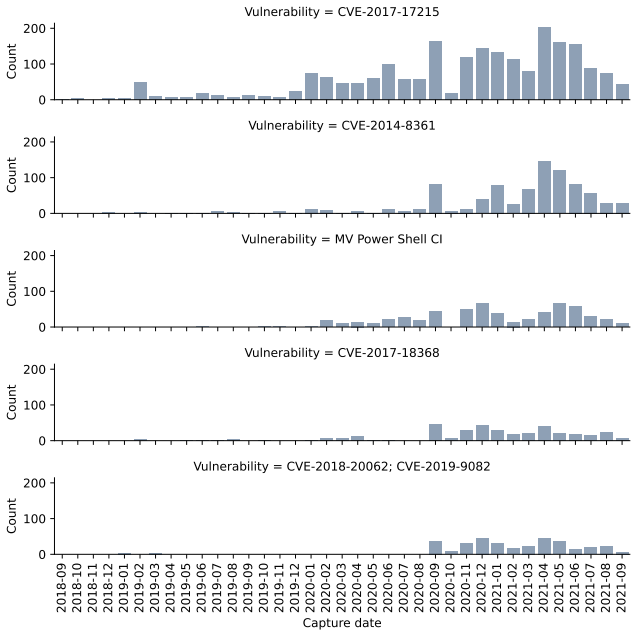

Figure 5: Top five targeted vulnerabilities based on their monthly exploit frequency in binaries.

The next vulnerability in terms of exploit frequency is MV Power Shell CI, which was published in August 2015. Unlike the previous two, it did not have a patch available and its PoC exploit was not released until a year and a half after its publication. It took another 46 months for the first exploit to show up in a binary. The fourth most exploited vulnerability in binaries was CVE-2017-18368. Its PoC exploit was made public six months before the vulnerability was disclosed, but a patch is still not available to the public. The final case in the top five is the combined CVE-2018-20062 and CVE-2019-90082, which were exploited in binaries using the same exploit a few times after their first disclosure in November 2018, but stopped being seen in binaries until after the PoC exploit was released in April 2020. From then on, these vulnerabilities were exploited in binaries for several months. To contrast, in some other cases (such as CVE-2013-7471 and Xiaogmai Backdoor) PoC exploits had been available for years, but attacks only started to appear more frequently in binaries after the vulnerabilities were publicly disclosed.

4.2 Distribution of vulnerability features

We found all targeted vulnerabilities were published in 2013 onwards except for one that was published in 2009 (CVE-2009-0545). However, the 2009 vulnerability was found to share the same PoC exploit with a vulnerability that was published 10 years later, CVE-2019-12725. At the other extreme, the most recent vulnerability, CVE-2021-38647, was published in August 2021 and was seen in binaries a month after that. Most targeted vulnerabilities were published in 2018 and 2021.

Looking at the vulnerability type in Table 3, Remote Code Execution (RCE) represents the majority (55%) in the IoT binaries,

followed by Command Injection around (39%). The other vulnerability types, Backdoor and Password Overflow, both represent 4% and 2%, respectively. That being said, CVE-2017-17215, which is the vulnerability with the highest exploit frequency of 2,177 times and was seen continuously in binaries used command injection for exploitation.

In terms of the six device types we encountered, routers are the primary type attackers target, which covers 28 out of the 49 observations. Surveillance and Web application were each seen in seven cases, NAS and Web servers supporting IoT devices in only three. The one remaining device type is TV, which was only seen once. This result ties well with previous studies wherein TV is the least device type attackers are targeting [2, 6].

Also, Table 3 shows that the maximum CVSSv3 score is 10 out of 10 (critical), which was seen in one case: CVE-2019-7256. Remarkably, this vulnerability was exploited only once in binaries. Overall, the average score base is 6.1 ("Medium") and the median is 8.8 ("High").

Next indicator is CPE. The average affected versions is 1.9. This means that 85.7% of the total cases had two or less versions affected for a given vulnerability. On the other hand, the maximum CPE number is 27. These 27 affected systems belonged to one vulnerability (CVE-2020-9054), of which the corresponding exploit seen only once in our malware binaries.

Finally, the vulnerability lifetime. The shortest exploit window we observed since the publication of vulnerability was 49 days for CVE-2021-38647 that was published in August 2021, less than two months before the end of our binary collection. On the contrary, the largest is 4,614 days for a vulnerability that was published in 2009 (CVE-2009-0545). However, the average exploit window for is 1,373 days.

4.3 Distribution of patching features

Although patching is critical to mitigate any security flaw in a vulnerable device, Table 3 shows that only 49% of vendors have actually released patches. That means almost half of the IoT vulnerable devices we observed remained vulnerable, unless users found another way to secure them in the absence of a patch.

The percentage is a bit higher (51%) when we measure if a vendor has issued a security advisory. In one case a vendor did provide a security advisory, but the patch was not available.

Another indicator is patch complexity. When patching only needs from a user to upgrade a firmware, we count this as one step unless it states otherwise in the security notes. For example, the maximum number of steps to patch was seen in D-Link DSL-2750B device, the user needs 13 steps, whereas for CVE-2016-6277, users needs to perform six steps according to the instructions on their security advisory. In average, a user needs around 2 steps to patch a vulnerable device, and one step in median.

In regards to the patch lifetime, the patches for the targeted vulnerabilities were available and released for around 1,010 days, on average. The vulnerability with the maximum patch lifetime of 3,045 was Netgear DGN1000 RCE, and CVE-2021-38647 had the minimum window of 17 days for patching.

4.4 Distribution of exploit features

We assume the availability of public PoC exploit can influence the exploit frequency in binaries. Table 3 shows that all exploits have PoC exploit available in public except for one exploit. It was for a vulnerability we mentioned earlier that was reserved by NVD (CVE-2019-7405), and it was only seen twice in our binaries. So while PoC was available almost always, and this might influence whether a vulnerability is targeted at all, it cannot help us explain how often a vulnerability is targeted.

We approximate complexity via the number of lines of PoC exploit. The average PoC exploit is around 76 lines, with the minimum being one line and the maximum 317 lines. The minimum was seen in exploitation of Vacron NVR RCE vulnerability. Only fewer cases were seen when the lines of PoC code getting higher than the average. In addition, the vulnerability with the highest frequency of being targeted had an exploit complexity of less than the average (25).

Lastly, the exploit lifetime. The maximum time a PoC exploit was available is 4,412 that was targeting CVE-2009-0545; CVE-2019-12725, yet only seen once in binaries. On the other hands, the most recent PoC exploit that was targeting the most recent vulnerability (CVE-2021-38647) for 20 times, it was available 27 days *after* the last day of binary collection. On average, PoC exploits were available for 1,308 days.

4.5 Distribution of exposure features

The installed base for the targeted devices of the 49 observations can bee seen in Table 3. The mean and median install base for the targeted devices are 37,098 and 485, respectively. However, the standard deviation is 217,960. This explains how spread out the data are from the mean. It can be explained when looking at the maximum number of installed base of 1,527,377 (outlier) while the minimum number of exposed devices in public is two. It is worth discussing these interesting findings of the highest install base device. It was for Huawei HG532 router that is affected by CVE-2017-17215, which is the vulnerability with the highest exploit frequency and were targeted continuously in binaries. Yet, the least number of exposed devices was ZTE ZXV10 H108L gateway, which was seen targeted five times in binaries. An interesting finding in the understanding of the install base in regards to device types, we found that the largest top four exposed devices were routers. On the other hand, the smallest top four install base were all web application except for one, which was a router.

5 QUANTIFYING THE FACTORS THAT DRIVE IOT VULNERABILITY EXPLOITATION

To quantify to what extent some factors explain why some IoT vulnerabilities are more frequently targeted than others, we create a statistical model based on our theoretical model. By doing so, we can develop a more comprehensive understanding of the factors that drive IoT vulnerability exploitation.

5.1 Indicators selection

The first step is to select indicators for inclusion in the model. This selection process is based on the principle of non-redundancy, whereby highly correlated indicators are removed from the model.

This reduced set of indicators is then used as explanatory factors in the GLM, and the significance of each indicator is assessed in order to further refine the model. The process of checking for redundancy is performed using the variance inflation factor (VIF), and indicators with VIF values greater than 5 are iteratively omitted. This leaves us with ten indicators. Based on the the guideline from O'Brien et al. [37] we exclude any of the indicators whose generalized variance inflation factor (GVIF) exceeds five (see Table 4).

Table 4: Variance inflation factors

	GVIF	Df	GVIF^(1/(2*Df))
Installed_Base	1.19	1	1.09
Vulnerability_Lifetime	1.15	1	1.07
Severity_CVSSv3	1.63	1	1.28
Vulnerability_type	1.53	2	1.11
Device_Type	1.38	2	1.08
Patch_Complexity	1.56	1	1.25
Patch_Availability	3.46	1	1.86
Security_Advisory	3.01	1	1.73
Exploit_Complexity	1.43	1	1.20
Exploit_Availability	1.32	1	1.15

5.2 Model fit

We explore the model fit by adding variables to the model using a stepwise forward procedure. The models are fitted using the log link function, and the Type III test is used to evaluate the significance of the variables [43]. This is because the results of the test indicate the significance of the complete variable, which may not be the same as the significance of the individual estimates of the different categories for categorical variables. Even if the estimates of some categories do not significantly differ from zero, the variable can still be significant to the model because the categories can differ significantly from each other.

We add variables to the model step-wise in the order they are presented in table Table 5. First, we start having the intercept only model (Model0) and then adding *exposure* variables, followed by *vulnerability*, *patching* and *exploit* variables respectively. This lead to the five different models as shown in Table 5.

To evaluate the goodness of fit, we employed selection criteria based on established practices in the literature [1, 14, 16, 24, 25]. First, we used the Akaike Information Criterion (AIC) to compare model scores and determine the best-fitting model for the data [8], which also tends to perform better with relatively smaller sample sizes. Second, the Bayesian Information Criterion (BIC) places a stronger penalty on model complexity, favoring simpler models. This criterion is especially useful for identifying the most parsimonious model. A lower AIC or BIC value indicates a better fit [14, 16, 24]. Next, we calculated the Log-likelihood [18] to compare different models and assess their relative fit to the data. Then, the F statistic [18] to evaluate and compare the variance explained by the model against the unexplained variance, providing a measure of the improvement in model fit compared to a null model. Unlike AIC or BIC value, The higher the log-likelihood and the F statistic value, the better a model fits a dataset. Furthermore, we used the

Table 5: GLM regression results.

	Model0	Model1	Model2	Model3	Model4
(Intercept)	4.831^{***}	2.475^{***}	0.194	0.369	−1.881
	(0.389)	(0.735)	(1.833)	(1.161)	(1.276)
Exposure					
Install Base		0.186^{*}	0.148	0.202^{**}	0.225^{***}
		(0.091)	(0.112)	(0.073)	(0.063)
Vulnerability					
Vulnerability Lifetime		0.0006+	0.0006	0.0005^{*}	0.0007^{***}
		(0.0003)	(0.0004)	(0.0002)	(0.0002)
Severity (CVSSv3)			0.065	0.005	−0.028
			(0.091)	(0.062)	(0.052)
Command Injection			2.469	1.399	2.783^{**}
			(1.710)	(1.066)	(0.999)
Remote Code Execution			1.440	−0.242	0.794
			(1.669)	(1.040)	(0.944)
Router			0.094	1.137^{*}	1.334^{**}
			(0.855)	(0.560)	(0.480)
Surveillance			0.703	1.771^{*}	1.480^{*}
			(1.232)	(0.810)	(0.705)
Patching					
Complexity				0.294^{***}	0.262^{***}
				(0.068)	(0.059)
Availability				−1.122	−0.886
				(0.772)	(0.684)
Security Advisory				0.029	0.108
				(0.712)	(0.642)
Exploit					
Complexity					-0.008^{*}
					(0.003)
Availability					1.382
					(0.974)
Num.Obs.	49	49	49	49	49
R^2 McFadden	-	0.151	0.204	0.334	0.377
AIC	830.6	709.7	675.6	573.6	542.5
BIC	832.5	715.3	690.8	594.4	567.1
Log.Lik.	−414.280	−351.836	−329.814	−275.805	−258.274
F	-	3.968	1.448	5.265	6.257
RMSE	337.69	299.13	268.71	499.84	479.02

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Root Mean Square Error (RMSE), a widely used statistical metric for comparing regression models [14]. RMSE offers a measure of overall accuracy for the model's predictions, aiding in the quantification of average prediction error. Even though our focus is to explain not to predict, we only reported the RMSE scores for completeness.

Looking at the scores provided by these criteria, Model 4 had the best fit among the models, with a lower AIC, BIC, and log likelihood, and a higher F statistic and lower RMSE. Even though the prediction score of RMSE in model 4 doesn't reflect the lowest, we used it to only report the observed value. Additionally, several of the variables in Model 4 were statistically significant, including the "Install Base", "Vulnerability Lifetime", "Vulnerability type Command Injection", "Device Type Router", "Device Type Surveillance", "Patch Complexity", and "Exploit Complexity" variables. This suggests that these factors play a significant role in determining exploit frequency in IoT binaries.

Looking at McFadden's R^2 value, we can get insights about how well a model explains the variance in the data. Note that in a GLM with a negative binomial distribution, the R^2 value is often lower than in other types of regression, such as linear regression, because the negative binomial distribution is often a better fit for overdispersed data (data with higher variance than expected under the

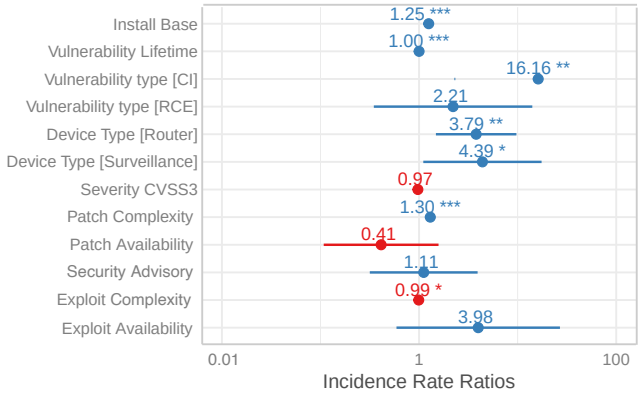

Figure 6: Forest plot of the GLM exponentiated coefficients

model). This means that even though the model may have a good fit to the data, there will still be some remaining variation that is not explained by the model, resulting in a lower R^2 value.

Additionally, our focus is on explanatory power rather than prediction, as the negative binomial model is better suited for modeling the underlying processes that generate the data. Therefore, a lower R^2 value does not necessarily indicate a poor model fit, but rather reflect the inherent variability in the data and the focus on explaining the underlying processes rather than making precise predictions.

5.3 Interpretation of the variable effects

The coefficients of the GLM model represent the relationship between the latent variables (vulnerability, patching, exploit, and exposure) as measured by the corresponding indicators and the frequency of exploits. If the coefficient for a particular indicator of vulnerability is positive, this suggests that an increase in that indicator is associated with an increase in exploit frequency in binaries. On the other hand, if the coefficient is negative, this suggests that an increase in that indicator is associated with a decrease in exploit frequency in binaries. The coefficients can be used to understand how the different latent variables and their indicators influence the target selection of vulnerabilities in IoT binaries.

All the coefficients are positive except for *Patch Availability*, *Severity (CVSSv3)* and *Exploit Complexity*. The exponentiated coefficient and the 95% confidence interval are plotted in Figure 6. The forest plot shows the exploits frequency is affected in a different manner by several factors. The variable with the largest impact is the vulnerability type, with a 16.16-fold increase in the exploits frequency when the vulnerability is a command injection compared to other vulnerability types. The install base also has a significant effect, with a 1.25-fold increase in the frequency of exploits per unit increase in the installed base. For example, if the installed base is 1,000, a unit increase in the installed base (i.e, 10,000) would correspond to a 25% increase in the frequency of exploits when all other variables in the model are held constant. Other significant factors include the device type, with a 3.79-fold increase in the frequency of exploits when the device is a router compared to other types of devices, and patch complexity, with a 1.3-fold increase in the exploits frequency per unit increase in patch complexity. This confirms that vulnerability severity (CVSS) rating has no impact, which was studied in previous work in [13].

On the other hand, some variables have a negative impact on the frequency of exploits. These include patch availability, with a 0.41-fold decrease in the frequency of exploits when patches are available compared to when they are not, and exploit complexity, with a 0.99-fold decrease in the frequency of exploits per unit increase in exploit complexity.

To understand the particular impact of a change in the explanatory variables, we compute a set of contrasts at the mean. In Table 6, the contrast estimates show the difference in the mean of the dependent variable (exploits) between each group and a reference group. For example, the contrast estimate for "Vulnerability type: Command Injection - Other" indicates that, compared to a hypothetical exploit with average characteristics, the number of exploits is 287 higher for vulnerabilities of type "Command Injection" compared to vulnerabilities of other types.

Table 6: Contrasts at the mean

Variable	Contrast	Estimate	Std.Error
Install_Base	+1	9.43	7.26
Vulnerability_Lifetime	+1	0.03	0.02
Vulnerability_type	CI - Other	287.00	197.00
Vulnerability_type	RCE - Other	22.90	24.10
Device_Type	Router - Other	30.80	23.70
Device_Type	Surveillance - Other	37.40	32.10
Severity_CVSSv3	+1	−1.17	2.36
Patch_Complexity	+1	11.00	8.33
Patch_Availability	TRUE - FALSE	−24.60	28.00
Security_Advisory	TRUE - FALSE	4.30	26.70
Exploit_Complexity	+1	−0.33	0.26
Exploit_Availability	TRUE - FALSE	31.30	27.20

Looking at Table 6, we can see that some of the contrast estimates have negative values. This indicates that, on average, the number of exploits is lower for the group being compared to the reference group. For example, the contrast estimate for "Patch Availability: TRUE - FALSE" shows that, on average, the number of exploits is 24.60 lower when a patch is available to remediate the vulnerability targeted by a specific exploit. Similarly, the contrast estimate for "Exploit Complexity: +1" indicates that, on average, the number of exploits is 0.33 lower for exploits with higher complexity compared to those with lower exploit complexity.

6 DISCUSSION

Our results have shown that several factors drive the exploitation of IoT vulnerabilities, including the exposure of the IoT devices and their vulnerability, the existence and complexity of patches and exploits, and the type of affected devices. These factors are aligned with traditional criminology theories, where rational attackers are interested in maximizing success while minimizing their effort and the chances of being caught. This suggest that attackers are limited by their resources when deciding which vulnerabilities to exploit, and they tend to focus on exploiting vulnerabilities that have a high impact and low complexity, as it is more cost-effective for them. This is in contrast to the commonly held belief that attackers will

exploit any vulnerability they can find. The evidence from actual exploits confirms this theory indicates that attackers are not keen on additional exploitation work, and rather tend to stick to the same tactics as long as they are successful against a significant number of IoT devices. This aligns with data from underground markets [5], and it suggests that attackers are work-averse in the sense that they are not inclined to change their approach if it is still effective. From this perspective, IoT malware is likely to target vulnerabilities with a high potential for financial gain and a low risk of detection. This means that IoT malware is likely to focus on vulnerabilities that can be easily exploited, have a large install base, and provide significant benefits to attackers.

Attackers target vulnerabilities in commonly used devices, such as routers and security cameras, allowing them to conduct distributed denial-of-service attacks. These vulnerabilities provide attackers with a large number of potential targets and can be exploited with relatively little effort. Additionally, the potential for disruption makes these vulnerabilities highly lucrative for attackers. On the other hand, IoT malware is less likely to target vulnerabilities with low potential for financial gain. For example, vulnerabilities in niche devices or those that only allow for limited access may not be worth the effort for attackers, as they may not provide significant benefits. Similarly, vulnerabilities that are difficult to exploit may also be less attractive to attackers, as they may not be able to exploit them successfully without being detected.

Our findings suggest that the CVSS score of a vulnerability did not significantly impact the exploit frequency. This means attackers may not necessarily prioritize exploiting vulnerabilities with higher CVSS scores over those with lower scores. This could be because attackers may have different priorities and objectives than those used to determine the CVSS scores. For example, a vulnerability with a lower CVSS score may be more attractive to an attacker if it is in a commonly used IoT device or if it has the potential for significant disruption. Additionally, we also find that the availability of patching did not significantly reduce the frequency of exploits. This suggests that even when a patch is available, attackers may still exploit the vulnerability successfully. This could be because not all users may apply the patch, or because attackers may find a way to bypass the patch. This highlights the importance of not only patching vulnerabilities but also of continuous monitoring, to detect and respond to any attempted exploitation.

Based on the results of our analysis, we recommend the following measures to decrease the usage of exploits and consequently improve the security of IoT devices:

- Prioritizing efforts on device types that are targeted more frequently. Large manufacturers with large install bases for their products should leverage their scale to improve defenses, as they are more likely to be targeted.
- Investing in better updating mechanisms that take into account patching and patch complexity to disincentivize IoT malware developers.
- Relying solely on CVSS scores on prioritizing vulnerability can lead to the risk of being targeted by cyberattacks. Instead, taking into consideration other characteristics of vulnerability such as vulnerability type and device types can help mitigating attacks.
- Providing a mechanism for consumers to obtain information on vulnerabilities, patches, and device support status, as well as considering the adoption of frameworks for secure software development and security labeling, can improve overall security and help consumers make informed decisions about the devices they use as most of the targeted devices are consumer-grade IoT.
- Requiring secure passwords, encrypted protocols for device access can help prevent unauthorized access and reduce the risk of vulnerabilities. Removing these feasibility will force attackers to develop more sophisticated, and costly, exploits.
- Disabling or uninstalling unused services and debug hooks, modes, and interfaces can help reduce the device's attack surface. The most frequent vulnerability types we observed in the wild are linked to services that are not necessary for the proper functioning of the devices. Hence, removing these would directly reduce the attack surface.

7 LIMITATIONS

Due to the fact that we used a variety of external sources, it was inevitable to inherit their limitations as well. First, the exploit coverage. It is hard to estimate how representative our dataset is for exploits frequency in IoT malware. During the dynamic analysis of binaries collected over the last four years, we were able to capture 59 unique exploits during. This number could be limited due to some IoT malware techniques used by malware authors such as anti-analysis and obfuscation to evade detection. This issue in dynamic analysis techniques has been reported before in [2, 6]. We did corroborate that our set of exploits is very similar to those reported in two prior studies that used both static and dynamic analysis in finding targeted vulnerabilities [2, 6]. However, to cover a wider range of exploits in IoT binaries, we recommend collecting data from diverse and comprehensive sources and conducting longitudinal studies over an extended period. This will enable the capturing of the evolving nature of IoT exploits. It is also important to involve cross-referencing and validating exploit information using multiple sources, such as Virustotal [46], to ensure a more representative set of IoT exploits.

Moreover, during the phase of collecting information on vulnerabilities, we had to look for different sources to measure publication date for the 15 vulnerabilities with no CVE-ID. We used sources we believe a given vulnerability was first discovered and use the date of the publication as the disclose date. This issue of vulnerability disclosure date was in some published CVEs. Their publication date didn't match the year in the CVE-ID as some CVEs where published years before the assigned date.

Second, in terms of device exposure, we used Shodan to measure the size of install base of IoT devices in public. Even though it is open for public, navigating through to find the correct device has certain limitations [8, 10]. It's not a single specific query that can be carried out for all IoT devices. To maximize our chances of finding the correct IoT devices, we manually checked the raw data produced by the banner grabbing for all the devices in order to find the most matching device name, services or software version to build our query syntax. Despite manually verifying the available information for all checked devices, the software version data is not

always accessible due to reliance on Shodan's collected banners. Therefore, we can't be sure that those devices are running the vulnerable software version. They might not be vulnerable. That said, this same limitation is faced by an attacker who is estimating the potential victim population before deciding which vulnerability to target. Only banner would be available to estimate the install base of a vulnerable device, until the attacker can run actual exploit code against potential victims. In this sense, this noisy data would still provide a good predictor, since we are trying to infer the reasoning of the attacker.

Finally, the patch availability. We couldn't find information for half of the cases even though we search their vendors website for any updates. Although patching information can be found easier for published vulnerabilities in the NVD, some of these CVEs are not updated to include their security advisory in the resources. For those case, we did our best to look on the internet, as well as for the vulnerabilities that didn't have CVE-ID.

8 RELATED WORK

Attackers select specific IoT vulnerabilities to target in IoT binaries, yet only few studies that focus on the attacker decision process [15]. Some studies looked at the relationship between vulnerabilities and exploits. Householder et al. [19] looked into 'when and how many vulnerabilities get associated public exploits' using CVE-IDs.

As a result of 6 years study of 75,807 vulnerabilities, only 3,164 of them had public exploits within 91 days, on average. On the other hand, Al Alsadi et al. [2] focused on answering which vulnerabilities are targeted in the wild and for how long? They analyzed 17,720 samples statically and dynamically using three different datasets from 2015 to 2020. They found only 63 unique exploits. On average, they found the time it takes for an attacker to exploit a vulnerability after it's been disclosed is around 29 months. Similarly, Alrawi et al.[6] used both static and dynamic analyses of 166k samples collected in 2019 to find list of targeted IoT vulnerabilities found in [2], which is also aligned with our findings. In addition, Nayak et al. [36], found only 15% of known vulnerabilities are exploited in the wild using around 300 million reports of intrusion-protection collected from more than six million hosts. This finding of attackers targeting only a few vulnerabilities out of thousands in the wild was also addressed in [2–4, 6, 48].

While these previous studies focused on vulnerabilities and exploits, others have looked at their relationship with respect to patching in IoT and non-IoT devices. Nakajima et al. [31] found some vendors don't work to improve their patch release delay over the years of study; and vulnerabilities with sever impact had no prioritize over the others. Yet, other factors can also impact the duration of patch deployment. For example, Nappa et al. [32] found that hosts belonging to security analysts and applications with an automated updating mechanism have significantly lower median times to patch, and Kotzias et al. [26] found enterprise hosts are patching faster than consumers of 90% of vulnerable systems take more than six months on average to patch, while others found that attackers prefer to spend their effort on known vulnerable devices that remained unpatched [3, 38, 48] or on ones that needs less effort to compromise [15].

Finally, some authors looked at the relationship between attacks and Internet-exposed devices. Bada and Pete [10] they analyzed forums that discuss the IoT search engine "Shodan" using CrimeBB dataset from the Cambridge Cybercrime Centre. They investigated the main use cases of Shodan for attackers. They found that Shodan is actively used by attackers to collect information about vulnerable devices. Others reported that using Shodan and other search engines for exposed vulnerable devices made them susceptible for being used by attackers [8, 27], yet these search engines were used in research to help security defenders automate their detection for vulnerable devices [8, 11].

9 CONCLUSION

We found that several factors have an impact on the frequency of exploits. The type of vulnerability had the largest impact, with a 16.16-fold increase in the frequency of exploits when the vulnerability is a command injection compared to other types of vulnerabilities. The installed base and the type of device were also found to have a significant effect, as well as patch complexity. On the other hand, variables such as patch availability and exploit complexity were found to have a negative impact on the frequency of exploits.

Our findings have important implications for cybersecurity efforts, as they can inform the design of more robust countermeasures and controls against IoT attacks and help protect against the growing threat of IoT botnets targeting connected devices. Additionally, our study highlights the need for proactive efforts to address vulnerabilities in IoT devices, as the proliferation of these devices is likely to continue and create new opportunities for attackers to exploit.

ACKNOWLEDGMENTS

This work is partly supported by the Dutch Research Council (NWO) under the RAPID project (Grant No. CS.007), by the Ministry of Internal Affairs and Communications (Commissioned Research JPJ000254), National Institute of Information and Communications Technology (Commissioned Research No.05201), and JSPS KAKENHI (21KK0178). This research is also partly funded by King Abdulaziz City for Science and Technology (KACST).

REFERENCES

[1] SHANJIDA AKHTER. 2015. *Generalized Linear Modeling for Cottage Insurance Data.* Master's thesis.

[2] Arwa Abdulkarim Al Alsadi, Kaichi Sameshima, Jakob Bleier, Katsunari Yoshioka, Martina Lindorfer, Michel van Eeten, and Carlos H Gañán. 2022. No Spring Chicken: Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis. (2022), 309–321.

[3] Global Cyber Alliance. 2022. *GCA Internet Integrity Papers: IoT Policy and Attack Report II.* Technical Report. Global Cyber Alliance. https://www.globalcyberalliance.org/reports_publications/iot-policy-and-attack-report-ii/.

[4] Luca Allodi and Fabio Massacci. 2014. Comparing Vulnerability Severity and Exploits Using Case-Control Studies. *ACM Trans. Inf. Syst. Secur.* 17, 1, Article 1 (aug 2014), 20 pages. https://doi.org/10.1145/2630069

[5] Luca Allodi, Fabio Massacci, and Julian Williams. 2022. The work-averse cyber-attacker model: theory and evidence from two million attack signatures. *Risk Analysis* 42, 8 (2022), 1623–1642.

[6] 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.

[7] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher,

Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1093–1110. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[8] Marc Arnaert, Yoann Bertrand, and Karima Boudaoud. 2016. Modeling Vulnerable Internet of Things on SHODAN and CENSYS : An Ontology for Cyber Security. In *International Conference on Emerging Security Information, Systems and Technologies*.

[9] Ashish Arora, Anand Nandkumar, and Rahul Telang. 2006. Does information security attack frequency increase with vulnerability disclosure? An empirical analysis. *Information Systems Frontiers* 8 (01 2006), 350–362. https://doi.org/10.1007/s10796-006-9012-5

[10] Maria Bada and Ildiko Pete. 2020. An exploration of the cybercrime ecosystem around Shodan. In *2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. 1–8. https://doi.org/10.1109/IOTSMS52051.2020.9340224

[11] Genge Bela and Calin Enachescu. 2015. ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services. *Security and Communication Networks* 9 (04 2015). https://doi.org/10.1002/sec.1262

[12] Laura J Bowman. 2022. Statista. *Journal of Business & Finance Librarianship* (2022), 1–6.

[13] Mehran Bozorgi, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. 2010. Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Washington, DC, USA) *(KDD '10)*. Association for Computing Machinery, New York, NY, USA, 105–114.

[14] Peter Bruce and Andrew Bruce. 2017. *Practical Statistics for Data Scientists*. O'Reilly Media.

[15] Louis Anthony Tony Cox. 2008. Some limitations of "Risk = Threat x Vulnerability x Consequence" for risk analysis of terrorist attacks. *Risk analysis : an official publication of the Society for Risk Analysis* 28, 6 (December 2008), 1749—1761. https://doi.org/10.1111/j.1539-6924.2008.01142.x

[16] Henry De and Graft Acquah. 2010. Comparison of Akaike information criterion (AIC) and Bayesian information criterion (BIC) in selection of an asymmetric price relationship. *Journal of Development and Agricultural Economics* 2 (02 2010), 1–6.

[17] James Gareth, Witten Daniela, Hastie Trevor, and Tibshirani Robert. 2013. *An introduction to statistical learning: with applications in R*. Spinger.

[18] David W. Hosmer, Trina A. Hosmer, Saskia le Cessie, and Stanley Lemeshow. 1997. A comparison of goodness-of-fit tests for the logistic regression model. *Statistics in medicine* 16 9 (1997), 965–80.

[19] Allen D Householder, Jeff Chrabaszcz, Trent Novelly, David Warren, and Jonathan M Spring. 2020. Historical analysis of exploit availability timelines. In *13th USENIX Workshop on Cyber Security Experimentation and Test (CSET 20)*.

[20] Alice Hutchings and Richard Clayton. 2017. Configuring Zeus: A case study of online crime target selection and knowledge transmission. In *2017 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 33–40.

[21] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. 2021. Exploit Prediction Scoring System (EPSS). *Digital Threats* 2, 3, Article 20 (jul 2021), 17 pages.

[22] John P. John, Alexander Moshchuk, Steven D. Gribble, and Arvind Krishnamurthy. 2009. Studying Spamming Botnets Using Botlab. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation* (Boston, Massachusetts) *(NSDI'09)*. USENIX Association, USA, 291–306.

[23] Ron Johnston, Kelvyn Jones, and David Manley. 2018. Confounding and collinearity in regression analysis: a cautionary tale and an alternative procedure, illustrated by studies of British voting behaviour. *Quality & quantity* 52 (2018), 1957–1976.

[24] David R. Anderson Kenneth P. Burnham. 2002. *Model Selection and Inference: A Practical Information-Theoretic Approach*.

[25] Maciej Korczynski, Maarten Wullink, Samaneh Tajalizadehkhoob, Giovane C. M. Moura, Arman Noroozian, Drew Bagley, and Cristian Hesselman. 2018. Cybercrime After the Sunrise: A Statistical Analysis of DNS Abuse in New GTLDs. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (Incheon, Republic of Korea) *(ASIACCS '18)*. Association for Computing Machinery, New York, NY, USA, 609–623. https://doi.org/10.1145/3196494.3196548

[26] Platon Kotzias, Leyla Bilge, Pierre-Antoine Vervier, and Juan Caballero. 2019. Mind Your Own Business: A Longitudinal Study of Threats and Vulnerabilities in Enterprises. https://doi.org/10.14722/ndss.2019.23522

[27] Eireann P Leverett. 2011. Quantitatively assessing and visualising industrial system attack surfaces. *University of Cambridge, Darwin College* 7 (2011), 21.

[28] John Matherly. 2022. Shodan Search Engine. https://www.shodan.io/dashboard.

[29] Scott Menard. 2002. *Applied logistic regression analysis*. Number 106. Sage.

[30] MITRE. 2022. CVE - Common Vulnerabilities and Exposures (CVE). https://cve.mitre.org/index.html.

[31] Asuka Nakajima, Takuya Watanabe, Eitaro Shioji, Mitsuaki Akiyama, and Maverick Woo. 2019. A Pilot Study on Consumer IoT Device Vulnerability Disclosure and Patch Release in Japan and the United States. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. 485–492.

[32] Antonio Nappa, Richard Johnson, Leyla Bilge, Juan Caballero, and Tudor Dumitras. 2015. The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. In *2015 IEEE Symposium on Security and Privacy*. 692–708. https://doi.org/10.1109/SP.2015.48

[33] National Institute of Standards and Technology (NIST). 2021. National Vulnerability Database. https://nvd.nist.gov/

[34] National Institute of Standards and Technology (NIST). 2022. CVE-2014-8361. https://nvd.nist.gov/vuln/detail/CVE-2014-8361

[35] National Institute of Standards and Technology (NIST). 2022. CVE-2017-17215. https://nvd.nist.gov/vuln/detail/cve-2017-17215.

[36] Kartik Nayak, Daniel Marino, Petros Efstathopoulos, and Tudor Dumitraş. 2014. Some vulnerabilities are different than others: Studying vulnerabilities and attack surfaces in the wild. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8688 LNCS. Springer Verlag, 426–446. https://doi.org/10.1007/978-3-319-11379-1{_}21

[37] Robert M O'Brien. 2017. Dropping highly collinear variables from a model: why it typically is not a good idea. *Social Science Quarterly* 98, 1 (2017), 360–375.

[38] Hamed Okhravi. 2008. Evaluation of patch management strategies. *International Journal of Computational Intelligence: Theory and Practice* 3 (12 2008).

[39] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: Analysing the Rise of IoT Compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/woot15/workshop-program/presentation/pa

[40] Roberto Perdisci, Wenke Lee, and Nick Feamster. 2010. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation* (San Jose, California) *(NSDI'10)*. USENIX Association, USA, 26.

[41] Offensive Security. 2022. Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers. https://www.exploit-db.com/.

[42] Offensive Security. 2023. OptiLink ONT1GEW GPON 2.1.11_X101 Build 1127.190306 - Remote Code Execution. https://www.exploit-db.com/exploits/49955.

[43] Walter W Stroup, George A Milliken, Elizabeth A Claassen, and Russell D Wolfinger. 2018. *SAS for mixed models: introduction and basic applications*. SAS Institute.

[44] ST Tajalizadehkhoob, Hadi Asghari, Carlos Gañán, and MJG Van Eeten. 2014. Why them? Extracting intelligence about target selection from Zeus financial malware. In *Proceedings of the 13th Annual Workshop on the Economics of Information Security, WEIS 2014, State College (USA), June 23-24, 2014*. WEIS.

[45] Jay M Ver Hoef and Peter L Boveng. 2007. Quasi-Poisson vs. negative binomial regression: how should we model overdispersed count data? *Ecology* 88, 11 (2007), 2766–2772.

[46] VirusTotal. 2020. VirusTotal Reference API | Files. https://developers.virustotal.com/v3.0/reference#files

[47] Chris Wanstrath, P. J. Hyett, Tom Preston-Werner, and Scott Chacon. 2022. Github Database. https://github.com/.

[48] Suzanne Widup, Marc Spitler, David Hylender, and Gabriel Bassett. 2018. 2018 Verizon Data Breach Investigations Report.