Empirical Study of System Resources Abused by IoT Attackers

Zijing Yin*
Yiwen Xu*
Tsinghua University
Beijing, China

Chijin Zhou
Tsinghua University
ShuiMuYuLin Ltd
Beijing, China

Yu Jiang†
Tsinghua University
Beijing, China

ABSTRACT

IoT devices have been under frequent attacks in recent years, causing severe impacts. Previous research has shown the evolution and features of some specific IoT malware families or stages of IoT attacks through offline sample analysis. However, we still lack a systematic observation of various system resources abused by active attackers and the malicious intentions behind these behaviors. This makes it difficult to design appropriate protection strategies to defend against existing attacks and possible future variants.

In this paper, we fill this gap by analyzing 117,862 valid attack sessions captured by our dedicated high-interaction IoT honeypot, HoneyAsclepius, and further discover the intentions in our designed workflow. HoneyAsclepius enables high capture capability as well as continuous behavior monitoring during active attack sessions in real-time. Through a large-scale deployment, we collected 11,301,239 malicious behaviors originating from 50,594 different attackers. Based on this information, we further separate the behaviors in different attack sessions targeting distinct categories of system resources, estimate the temporal relations and summarize their malicious intentions behind. Inspired by such investigations, we present several key insights about abusive behaviors of the file, network, process, and special capability resources, and further propose practical defense strategies to better protect IoT devices.

KEYWORDS

IoT Attack, System Resource Abuse, Behavior Intention

ACM Reference Format:
Zijing Yin, Yiwen Xu, Chijin Zhou, and Yu Jiang. 2022. Empirical Study of System Resources Abused by IoT Attackers. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3551349.3556901

1 INTRODUCTION

Linux-based IoT devices are widely used in various fields nowadays. Along with it, numerous attacks against these devices have emerged. Some malware attacks use families like Mirai to spread over many IoT devices, construct botnets to achieve impacts like DDoS [28] and crypto mining [30]. In recent years, fileless attacks [12] also increases significantly, posing severe threats to IoT systems. Both of these attackers conduct malicious behaviors by abusing system resources, including files, network, processes, and special system capabilities, to accomplish hostile intentions. Understanding these system resource abuses allows us to have a systematic summary of the malicious behavior characteristics of current prevalent invasions, and intention analysis of these behaviors can provide us with a glimpse into the incentives under attackers' mindsets. Such observations can effectively assist us in designing necessary resistance and defense strategies to protect IoT devices.

In recent years, previous studies like [23] [4] analyze specific IoT malware families and provide valuable conclusions. However, their attention only focuses on certain malware families and presents relatively scattered observations toward the overall IoT attack ecosystem. While some efforts have been made on large-scale studies of IoT malwares across multiple sources, such research concentrates on concrete behaviors at specific stages or comparison over certain features. For example, Alsadi et al. [1] analyzes the infection vectors of IoT malwares and provides the evolution of used exploitations. Donno et al. [15] conducts a comprehensive investigation over the DDoS modules adopted by IoT malwares. Cozzi et al. [11] analyzes the code similarity characteristics and lineage of the malware samples. Although these researchers provide fascinating insights into different aspects of IoT malwares, we still lack an overall understanding of the abuse situation of various system resources across the IoT attack landscape. Meanwhile, the attack behaviors are highly related to outside network environments. For example, attackers often communicate with their specified command-and-control (C&C) server to obtain instructions to decide following actions. Thus malwares might become dormant without such environment during offline dynamic analysis adopted in studies like [9] [2], causing insufficient malicious operation collections.

In this paper, we collected the malicious behaviors of currently prevalent IoT attackers worldwide to explore their system resource abuses and malicious intentions behind. First, we design a high-interaction honeypot, HoneyAsclepius. It can not only send faithful response packets to attackers to maximize capture capability, but also monitor attackers' behaviors to support real-time analysis. In particular, based on the devised Monitor Policy, the honeypot is able to achieve continuous behavior monitoring to active attack sessions on constrained physical IoT devices. It can fully unveil the resource abuses throughout invasion processes. Different from traditional sandboxes like IoTBOX [34] and Limon [26], such procedure can immediately monitor attack behaviors once captured, providing a general real-time analysis solution for heterogeneous IoT devices.

Second, through large-scale honeypot deployments on the real-world network, we successfully captured significant numbers of

*Both authors contributed equally to this research.
†Yu Jiang is the corresponding author.

ASE '22, October 10–14, 2022, Rochester, MI, USA

ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3556901

attacks, collected their malicious behaviors as well as their resource accesses. These honeypots are deployed among ten countries distributed in five continents with three different architectures, i.e., ARM, MIPS, MIPSEL. During a span of 34 days, HoneyAsclepius has received 117,862 valid attack sessions originating from 171 countries. These attacks brought 76,403 malware files, classified with VirusTotal [44] and AVClass [39], and identified 12 different malware families. The attackers' behaviors are simultaneously tracked along with the invasion process and gathered 11,301,239 records of the malicious operations. Based on this information, we systematically summarize the targeted system resources concerned with file, network, process, and special capability, delve into the intentions behind, and further lead to the following insights:

Accesses to file resources are concentrated on a few categories, which should be given higher attention in the future attack detection. All of the valid attack sessions access file system resources after infecting a device. During the experiment, we collected a total of 2,049,011 sensitive file access records. Although these attackers introduced different kinds of malwares and hoped to achieve different impacts, the categories of file resources they accessed were very focused. Attack sessions that conducted operations to the six most frequently used sensitive file resources account for 83.57% of the total amount. Such a high degree of similarity gives us an appropriate oracle for detecting IoT attacks.

From the perspective of invasion and proliferation, the operations to network resources give us notifications about vulnerable devices with fragile services under attacks recently. Based on frequently bound ports by attackers, we can get a glimpse of the fragile network services that are severely under invasion on IoT devices. Some captured IoT attacks proactively bind the ports of vulnerable services to prevent other attackers from invading through them to further monopolize the device. For example, 498 attack sessions bound the 23 port, initially used by *Telnet* service, and thus warded off the following competitors. From the connected ports, we can also have a wide spectrum of the current fragile devices for proliferation. IoT attackers (60.69% in our experiment) usually use the invaded devices to initiate malicious connect behaviors, scanning for new available targets to expand the botnet's size. For instance, 5.68% of the attack sessions tried to connect to the 37215 port, which exactly corresponds to a vulnerable UPnP service on Huawei HG532. Apart from this, we further found that instead of brute-force for infection, more and more attackers switch to exploitations to gain the initial access, which accounts for 60.9% of the attack sessions that have tried to scan for new victims.

Attackers aggressively kill processes responsible for critical functionalities of IoT devices, which seriously hinder their usability, stressing the protection of vital process resources. After the invasion, 8.75% of the attack sessions send *sigkill* signals to running processes to end the execution. Some attackers attempt to defeat their rivals by killing other attackers' malware processes. Meanwhile, some killing operations to process resources can also impact the functionality of targeted devices. For example, a total of 3,076 attack sessions kill network-related process *wpad* (802.1x authentication service) and *odhcpd* (DHCP server), which can affect normal operations of network-attached IoT devices, like routers and modems. This reminds us that the possible impacts of IoT attacks are not limited to relatively mild ones that are mostly mentioned in previous research like DDoS or crypto mining, but can also seriously hinder devices' usability.

An increasing number of intruders abuse special system capabilities to achieve analysis environment evasion, raising alarms for security researchers. We noticed that more than 11.27% of the attackers try to abuse special system capabilities, especially for reconnaissance and camouflage goals. We found that 7,144 attackers used *sys_ptrace* to detect debuggers and then exit for anti-analysis. Meanwhile, 4,103 attack sessions modified their process name to confusing ones like *busybox* to disguise themselves. Apart from that, 2.99% of the attackers used special capabilities to escape from a possible virtual environment to infect the host and escalate the privilege. These observations alert security researchers to mask potential fingerprints of the analysis environment to fully expose IoT attackers' malicious behaviors.

Contribution: We mainly make the following contributions:

- We propose a high-interaction IoT honeypot HoneyAsclepius [1], which can automatically monitor continuous behaviors on active attack sessions with low runtime costs, fully collecting their abuses of system resources while maximizing the capture capability.
- We deployed 60 IoT devices based on HoneyAsclepius around ten countries distributed in five continents with three architectures. During a span of 34 days, we collected 117,862 attack sessions from 171 countries, captured 76,403 malwares, and observed 11,301,239 malicious behaviors.
- We systematically analyze system resource abuses conducted by prevalent IoT attacks with the designed workflow, highlight the intentions behind these malicious operations and present several insights and defense strategies.

2 BACKGROUND

2.1 IoT Attacks

Ever since the source code of Mirai, one of the most prominent IoT malware families, was released to the public in 2016, it prompted a surge in more powerful IoT malware variants. They continuously incorporate code from each other and iterate with more functions, enhancing their compromise capabilities. Hence, many IoT malware families, like Gafgyt [29], Tsunami [32] and Hajime [23] have been simultaneously spreading worldwide. Recently, fileless attacks that do not rely on specific malware files have also affected some IoT devices, posing new challenges to device security.

Commonly, the lifecycle of IoT attacks can be refined and divided into four parts: Infection, Reconnaissance, Persistence, and Impact. At each stage, attackers conduct specific operations on the corresponding system resources to achieve their goals. In the first phase, IoT attackers utilize various network characteristics to probe real IoT devices and find an intrusion point. After that, they start reconnaissance by gathering the necessary information, such as system configuration files and network status, to prepare for the subsequent procedures. In the third step, IoT attackers try to install and disguise themselves by various means such as cleaning up log

[1]The artifact and collected data are available at: https://github.com/HoneyAsclepius/HoneyAsclepius.git.

files or hiding processes to achieve persistence on the devices. The forth stage of the lifecycle refers to the ultimate purpose that the attackers hope to achieve. Some attackers exhaust network or computation ability to conduct DDoS attacks, mining cryptocurrency, etc. As we can see, all the malicious operations conducted throughout the IoT attack lifecycle depend on abusing system resources like files or processes to realize the intentions behind them.

2.2 IoT Honeypot

IoT honeypot is a network-attached decoy embedded system with vulnerable entry points to lure IoT attackers, so that defenders can clearly inspect the possible threats and then enhance the protection in advance. There are various IoT honeypot [21] [34] [43] [12], which can be divided into two categories—software honeypot and hardware honeypot. Specifically, the software IoT honeypot refers to the emulated vulnerable IoT system based on the cloud infrastructure. It can provide abnormal traffic monitoring and diverse customized system components to observe IoT attacks comprehensively for its sufficient hardware resources. However, the manually crafting software honeypots cannot systematically learn the behavioral knowledge of IoT devices and fully mimic the real interaction procedure. Thus it usually has deficient capture capability, especially for IoT attackers equipped with in-depth reconnaissance and anti-analysis techniques. On the contrary, hardware IoT honeypots apply actual IoT devices for attackers to interact with. But the insufficient hardware resources in physical devices pose the challenge of tracking and analyzing behaviors conducted by the incoming attackers in real-time. The great divide between software honeypots with their fine analyzing capability, and hardware honeypots with their high-interaction capability, expresses a sharp and clear break with no attempt at its mitigation.

3 EMPIRICAL STUDY WORKFLOW

The general workflow of our research is presented in Figure 1. It is divided into two stages: behavioral information gathering with HoneyAsclepius, and malicious intention processing. In the first stage, we designed a high-interaction honeypot with continuous behavior monitoring mechanism called HoneyAsclepius, and deployed it widely on the public network. During this period, our honeypots automatically recorded a large number of malicious behaviors and abuses of system resources from active attack sessions in real-time. We further processed the recorded information in the second stage, decomposed different attack sessions, classified behaviors targeted at distinct categories of system resources, estimated their temporal relations, and analyzed their intentions behind.

3.1 HoneyAsclepius Design and Deployment

To fully collect the system resources abused by current prevalent attackers, we designed HoneyAsclepius, a high-interaction honeypot with continuous behavior monitoring mechanism. It can record the attackers' operations in real-time with low runtime costs and achieve high capture capability. HoneyAsclepius mainly consists of two parts, Customized Firmware and Frontier, and the overall process is shown in the left part of Figure 1.

3.1.1 Customized Firmware. Based on the features of IoT attacks, we designed a customized IoT firmware to capture the payloads reliably and achieve lightweight malicious behavior monitoring. The main components include the following aspects.

Shell Interceptor. To fully track each attack session, we customized a Shell Interceptor module from two aspects. In IoT honeypots, each login will trigger a new shell process, which identifies the starting point of an attack session. The Shell Interceptor can report such login and logout actions. It refuses another login session from a different source IP when an attacker has already logged in. In this way, each attacker has the opportunity to enjoy the playground within the corresponding attack session without disturbance from other attackers. Also, an attacker who has conducted an invasion before cannot re-access the device after logging out. This limit allows us to collect as many malicious samples from different attackers as possible, increasing the diversity. We set a maximum time of 5 minutes per login session to prevent a single attacker occupying the device too long. This time limit is consistent with previous research like [9], which is sufficient to expose attackers' malicious behaviors.

After logging in, some attackers will use shell to introduce malware files, while in recent years some also conduct fileless-attacks, executing shell commands to achieve their malicious goals directly without any malware files. The Shell Interceptor can also report all the executed commands to the frontier. This provides us with raw interactions between attackers and devices for future research.

Entry Point Backup. Malware files are commonly downloaded through specific entry points for further invasion. However, many malwares tend to conduct self-deletion, path changing, or self-renaming immediately once executed, which hinders the normal malware sample collection process.

Based on this observation, we customized common entry points of malware files like *wget* and *tftp*. During the download process, apart from the original path specified by the attacker, a file with the same content will be backed up into another directory simultaneously, which is completely transparent from the attacker's perspective. This guarantees any downloaded payloads will always be collected in Frontier, improving the robustness of sample capturing.

Monitor Policy. IoT devices are only equipped with limited hardware resources, which causes current behavior analysis mechanisms, such as library hooking unusable on these devices. Therefore, we implement the Monitor Policy to enable the SELinux [40] kernel module automatically record the behaviors conducted by the attackers. Since it is based on Flux Advanced Security Kernel architecture, the overhead is highly reduced by Access Vector Cache (AVC). With the suitable Monitor Policy loaded, the kernel can monitor attackers' abuses of system resources with low overhead in real-time, achieving both high-interaction in hardware-based honeypots and timely analysis in software-based honeypots.

SELinux utilizes security contexts to classify system resources. From the perspective of security contexts, we can have a basic concept of resource functionalities. We included the security contexts of all the system resources as monitor targets (i.e., target security context item in Monitor Policy, referred to as *tcontext*), so that we can have a comprehensive range of monitored behaviors. Meanwhile, we implemented several *typetransition* statements to transit the security context of all the break-in processes and their submodules to *analysis.subj*. With a unified security context, the behaviors that need to be analyzed can be limited to those conducted by processes

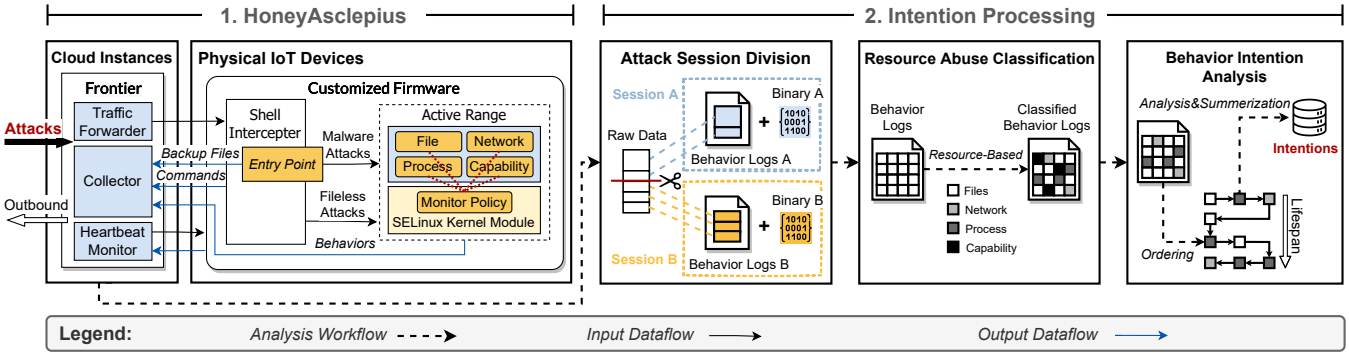

Figure 1: Overall workflow of our study. First, we designed a high-interaction honeypot, HoneyAsclepius, to achieve continuous behavior monitoring as well as high capture capability, and deployed it worldwide. Based on the collected raw data, we further separate different attack sessions, classify their abused resources and summarize the malicious intentions behind.

with *analysis.subj* security context. Therefore, only processes with this security context will be the monitored behavior performer in Monitor Policy (i.e., source security context item in Monitor Policy, referred to as *scontext*). In this way, the SELinux kernel module can be driven to record the system resource accesses performed by the attackers to all the system resources as AVC messages. All the attacks conducted in the monitored active range will be monitored, including both malware attacks and fileless attacks.

Each recorded message has detailed information about the corresponding behavior and contains two items that need to be noticed: 1) operation class, which is the types of the behavior, and 2) the target security contexts, which represents the categories of the abused resources based on their functionalities. For example, an AVC message with *add_name* as the operation class and *file.initscriptfile* as the target security context represents a behavior which adds a new file to the system initialization script directory like */etc/init.d* to achieve auto-reboot. Such abstract representation can help us summarize the subtle behavioral adjustments of the iterative variants targeting the functionality category of resources to achieve the same goal, helping us better understand the motivation behind this set of behaviors. With this mechanism, HoneyAsclepius can automatically monitor the behaviors and resource abuses of attackers with low overhead, facilitating our subsequent analysis.

3.1.2 Frontier. The Frontier module implements several auxiliary components to achieve traffic forwarding, data collecting, and heartbeat checking. We now illustrate the details below.

Traffic Forwarder. The Frontier module is deployed on virtual cloud instances around the world with public accessible IP addresses to attract attackers from different regions. To achieve high interaction, the attack traffic will be forwarded to IoT devices, and the outbound response will be replied to the original attackers. With this method, we can virtually deploy our honeypots based on physical IoT devices worldwide.

Collector. The Collector aggregates the information uploaded from customized firmware for further analysis. It automatically obtains backup samples from the device. Apart from this, it also saves login and logout attempts along with timestamps to help divide different attack sessions. Once an attacker has logged out and completed invasion, the Traffic Forwarder will be notified to block requests from the same IP to observe more diverse attack attempts. It also receives the behavior logs triggered by Monitor Policy to analyze the operations and resource abuses of attackers.

Heartbeat Monitor. As shown in Figure 1, the Heartbeat Monitor is mainly responsible for resetting the device if necessary. Since the firmware is based on an in-memory filesystem, all the modifications to the system can be reset after reboot. Therefore, the monitor will reboot the device when the following circumstances occur. First, it sends login attempts every minute to the physical device. If no packets responded for three times, the monitor will immediately reset the device. Second, when an attack session ends, the device will proactively notify the monitor to request a reboot. With this mechanism, new attacker will not be affected by the previous invasion and fully expose its malicious intentions in a clean execution environment.

3.1.3 Worldwide Deployment. We deployed HoneyAsclepius worldwide to capture IoT attacks for 34 days (from Oct. 2021 to Nov. 2021). In terms of geolocation, we select ten countries across five continents to place the frontiers of HoneyAsclepius and investigate the development of new or rampant IoT attacks nowadays. Four cloud services are accordingly chosen for frontiers to extend the variety of our IP address arrangement. Further, the frontiers can attract attack traffic at different geographic positions and transit the traffic to the back-end IoT devices. Each frontier in the cloud is furnished with one back-end device. We select three different IoT device models, ASUS AC68U, Xiaomi 4A and Netgear WNDR3800 with distinct CPU architectures, ARM, MIPS, and MIPSEL, to invite various attackers. Therefore, we compiled three versions of customized firmware and flashed them into the corresponding physical devices as the back-end of HoneyAsclepius. Since the design of HoneyAsclepius is independent of any specific architectures, it can be easily adapted to heterogeneous IoT devices. Overall, there are 60 frontiers armed with physical IoT devices distributed in five continents to attract attackers worldwide.

3.2 Intention Processing

We further process the access records collected by the deployed frontiers in this stage. After dividing different attack sessions, we

can then classify the behaviors based their abuses of different resource categories, estimate typical behavior temporal relations and subsequently summarize the malicious intentions behind.

3.2.1 Attack Session Division. We separate the behaviors conducted in different attack sessions based on the login and logout records. The Shell Interceptor records the timestamps of each connection session to the frontier. We extract the raw AVC messages generated in this connection as the attackers' operations in the session. At the same time, the malware samples downloaded during this period are also linked to this session. Thus, we can sort out the mixed raw data into behavior logs and binaries in different attack sessions, facilitating the subsequent analysis.

3.2.2 Resource Abuse Classification. Based on our observation, the system resources abused by attackers can be classified into four categories, file resources, network resources, process resources, and special capability resources. (1) Access to file resources refers to the operations like creating, deleting, and accessing files and directories in the firmware. (2) The abuse of network resources mainly refers to the attacker's binding or connecting to different ports. (3) The access to process resources includes the signals sent by the attacker to the various processes running in the system. (4) Special capability resource access includes other attackers' abuse of special system functions or system calls to conduct privileged operations, such as changing process names and UID. By separating different resource categories, we can analyze the malicious intentions based on the behavior logs targeting the same category of system resources.

3.2.3 Operation Intention Analysis. Furthermore, we evaluate the temporal relations of the resource accesses in the behavior logs and summarize the malicious intentions behind.

First, we enumerate every possible sequential order of each two abstract behaviors and iterate through attack sessions to calculate the frequencies of such order occurs. For two behaviors A and B, we first select the attack sessions that A and B are both performed. Then we count the numbers of attack sessions that the order $A \rightarrow B$ (i.e., behavior A precedes B in conducted timestamps) occurs and $B \rightarrow A$ (i.e., behavior B precedes A in conducted timestamps) occurs. If the former order frequency is more than 1.5 times than the latter, we conclude that the temporal relation — behavior A is commonly performed before B — exists in most circumstances of IoT attacks. After evaluating these temporal relations, we use each behavior as a vertex, and connect each pair of behaviors that exists a temporal relation with a directed edge to form a graph. In this way, we can obtain a complete behavior sequence in temporal order throughout IoT attack lifecycle. Such relations can assist us in estimating malicious intentions behind the behaviors correspond to each stage of the lifespan, and finally derive insightful conclusions.

4 FINDINGS AND IMPLICATIONS

In this section, we systematically analyze the IoT attacks captured by HoneyAsclepius and their malicious behaviors on various aspects of resources, and answer the following research questions:

- What system resources are often abused by IoT attackers?
- What are the malicious intentions behind these accesses?

The first research question focuses on what are the common behaviors of existing attackers and further reveals their possible harm to IoT devices. After further analysis of these behaviours' categories, frequency, and temporal relations, we can explore the incentives under attackers' mindsets and answer the second research question. Understanding these issues can help design better and more effective IoT device defence strategies in the future.

4.1 General Statistics

To delve into the characteristics of IoT attacks, we comprehensively gather vast quantities of run-time information through continuous behavior monitoring. The volume of this information is up to 11.95 GiB. Among them, the login and logout actions recorded by the Shell Interceptor can help us separate different attack sessions. In each session, AVC messages generated by the SELinux module in the firmware record the malicious behaviors of the attacker, containing the security contexts and categories of the abused resources, which is the main data source for the subsequent intention processing. The recorded shell commands and the saved malware samples can assist the researcher in more in-depth analysis in the future and help to implement defense strategies.

Statistically, frontiers of HoneyAsclepius as the bait obtained 117,862 malicious connections from 50,594 attackers distributed in 171 countries. Additionally, we use *ipinfo.io* [25] to acquire geographical information for the attacker of each attack session. Also, we visualize all the attack session between attackers and our honeypots as shown in Figure 2. Meanwhile, the back-end of HoneyAsclepius, i.e., IoT devices with customized firmware, captured 76,403 malicious payloads, including 12 malware families and 1,875 previously unknown variants in only 34 days. During our experiment, HoneyAsclepius collected 11,301,239 malicious behavior records conducted by these attackers.

Thereafter, to preliminary understand these behaviors, we thoroughly analyze and understand various types of system resources that attackers tend to access for their malicious intentions. These resources can be classified into four categories: 1) file resources, 2) network resources, 3) process resources, 4) special capability resources. The abuse frequencies and the temporal relations of behaviors that target these resources are shown in Figure 3. As can be seen from the figure, file resource abuses are most commonly observed in the early stage of the attack lifespan. Actions such as

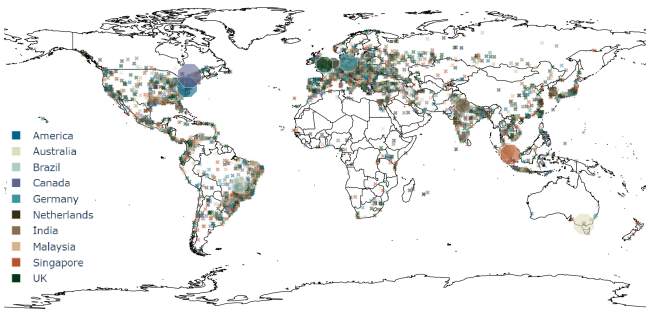

Figure 2: Attack sessions that HoneyAsclepius captured around the world. Each octagon indicates a HoneyAsclepius honeypot, whose color corresponds to the deployed location presented in the legend. Each cross symbol indicates the origin of an attack session, and its color represents the country of the targeted honeypot presented in the legend.

self-deleting (point 1) and detecting environmental information (point 2, 5) were conducted in more than 4,000 attack sessions. The abuse of network resources, on the other hand, is concentrated in the middle and late stages of the attacks. We also observed the payload sharing among different malware families presented in previous studies [27] [10], which leads to similar abuse frequencies on ports like 8080, 81 and 8443 (point 22, 23, 24). Process resources are often abused right before certain network resource accesses. Attackers may try to kill processes like *telnetd*, *dropbear* (point 31, 32) and then bind their ports (point 11, 12) to achieve device monopolization, preventing other attackers from infecting through these services. The abuse of special capability resources is reflected throughout the attack lifespan, achieving different purposes, from anti-analysis conducted in the early stage (point 41, 42) to privilege escalation in the late stage of the attacks (point 46).

Based on previous studies of the IoT malware lifecycle [2] [33], we can evaluate the purpose of each behavior in conjunction with its temporal order, so that the intentions behind the behavior can be better analyzed. In the following sub-sections, we further explore and share intention-related insights by measuring each class of abused resources accessed during the lifecycle of attack sessions.

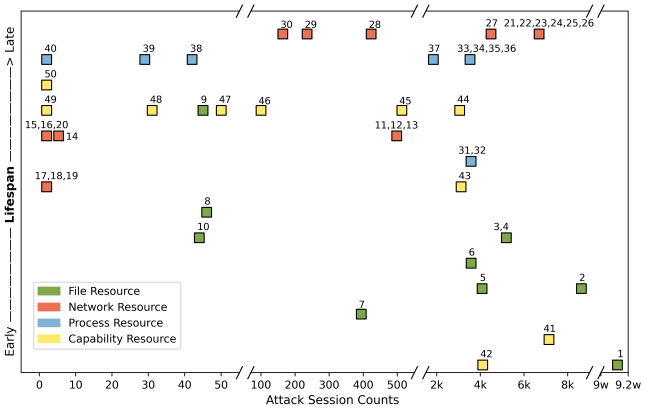

Figure 3: The abuse frequencies of system resources and temporal relations of behaviors target these resources. The horizontal axis indicates the number of attack sessions in which the corresponding behaviors occurred. The vertical axis indicates the temporal relations of the behaviors. The points are labeled that correspond to the specific behaviors presented in Table 1, Table 2, Table 3, Table 4 and Table 5. Behaviors represented by more upward points indicate that it were performed relatively late in an attack session.

4.2 File Resources

Attackers tend to concentrate their access to only a few file resources due to their frequent code reuse and relatively similar malicious purposes. Based on the observed behaviors, the most frequently accessed sensitive file categories are shown in Table 1. We systematically analyzed the possible intentions behind these operations, as listed below.

Gather Information. Many attackers read configuration files to gather information about the invaded devices to achieve reconnaissance. As shown in Table 1, 8,631 attack sessions read network information files, including router tables, interface configurations, socket information, etc. Intranet-related files can help attackers probe the accessible devices to proliferate, and socket information contains current bound ports and corresponding processes, which can guide attackers to prepare for subsequent process killing or further monopolizing attempts. Meanwhile, attackers also read files like */etc/cpuinfo* to check if the invaded system is emulated based on virtual machines. Such mechanisms can help attackers evade the virtual analysis environment, hiding malicious intentions from researchers. The architecture information also assists the attackers in selecting suitable malware binaries for further invasion.

Introduce malware files. Malware-based attacks need entry points to download malware files into the device. The most commonly used tools in IoT devices are *uclient-fetch* (a lightweight *wget* implementation in IoT firmware) and *atftp* (a mini *tftp* client in IoT firmware). 3,572 attack sessions utilize these tools to download many binaries of different architectures and execute each of them to ensure successful infection. Meanwhile, some crafty attackers adopt a smarter strategy based on the gathered architecture information, downloading only the suitable binaries into the device to avoid occupying too much storage and bringing noticeable influence. However, because of the heterogeneity of IoT devices, malware introduced with these mechanisms may still be inexecutable on the victim devices due to incompatible libraries or distinct cross-compilers. Therefore, we also observed 3 attack sessions attempting to compile the binary on the device with tools like *gcc*. Since most firmware is not installed with a compiler by default, we believe that such a mechanism is only suitable for high-end IoT devices with relatively sufficient computational and storage resources.

Disguise and Camouflage. Attackers usually adopt various techniques to achieve disguise and camouflage. For example, many attackers (91,208 attack sessions) will delete the original binary immediately once executed. 5,315 attack sessions create multiple binary copies into other covert file paths. This can help IoT attackers decentralize their malicious intentions into several processes, and even achieve automatic startup when a part of processes are killed. These created submodules are often stored in misleading paths, such as system configuration path (5,186 attack sessions) like */etc* and system binary paths (5,050 attack sessions) like */bin, /usr/bin, /sbin*. In this way, these malwares can be mixed up with the default system files in the same directory, making them unnoticeable to the user. Some of the variants, e.g., *Ganiw*, even directly replace normal system binaries (such as *ps, netstat*) with malwares to better disguise malicious processes and connection behaviors.

Achieve Persistence. Attackers achieve persistence on IoT devices through many file resource accesses. First, 46 attack sessions abuse the system startup procedure, appending scripts to */etc/rc.local*. This can make malwares automatically executed along with system boots. Meanwhile, 394 attack sessions cover their tracks and delete system logs recorded in paths like */tmp* to avoid being noticed. 45 attack sessions even try to corrupt system's security mechanisms, deleting file resources such as SELinux policies to achieve long-term residency on the IoT device. It is worth noting that our designed Monitor Policy denies any abuse of core components, including SELinux policies. Therefore such attackers cannot disrupt the behavior monitoring mechanism of HoneyAsclepius.

Table 1: Top abused sensitive file resources. The first column indicates the behavior labels. The second column indicates the behavior types, and the third column presents the target file categories of the behaviors. The forth column indicates the numbers of attack sessions that conducted the behaviors.

No.	Operations	Resource Categories	Counts
1	Delete File	Malware Samples	91,208
2	Read File Content	Network Information in */proc/net*	8,631
3	Create New File	Configuration Directories in */etc*	5,186
4	Create New File	System Binary Directories like */bin*	5,050
5	Read File Content	CPU Information File */proc/cpuinfo*	4,078
6	Execute File	Download Applications like *wget, tftp*	3,572
7	Delete File	Temporary Files Including Logs like *wtmp*	394
8	Modify File	Initialization Scripts in */etc/rc.local*	46
9	Delete File	SELinux-related Files like *policy.31*	45
10	Delete File	Download Applications like *wget, tftp*	44

Monopolize the device. Since most of the IoT devices are only equipped with limited hardware resources, the simultaneous existence of multiple malwares can seriously affect their performance, thus hindering the effectiveness of achieving malicious impacts like DDoS or crypto mining. Therefore, many attackers abuse file resources to monopolize the device, ensuring that only the malwares deployed by themselves can survive on the victim device. For example, 44 attacks try to delete common entry points like *atftp* to avoid other rivals introducing new malware files. Since many infections in the IoT domain apply the brute-force method initially, 20 attack sessions even attempt to edit password files like */etc/shadow*, changing the default login password to a more complex one to "protect" the device from competitive attackers.

Finding 1: 83.57% of the attack sessions concentrated on the top six file resources to achieve certain malicious intentions. Giving higher weights to these file resource accesses can enhance the effectiveness of IoT attack detection.

4.3 Network Resources

Network is treated as the basic functionality of most IoT devices. Different network port resources often correspond to specific network services. They are regarded as a vulnerable pathway for attacks or channels for information transmission. For the sake of some malicious intentions, this class of resources is seriously manipulated by IoT attackers based on our observation. Through examining bound and connected ports illustrated in Table 2 and Table 3, statistical results reflect the critical intentions and fresh insights behind.

Bound Network Ports. We quantitatively present ports that captured IoT attackers bound after their infections, as shown in Table 2. The purposes of binding ports mainly lie either in the communication with partners or the resistance to rivals. Specifically, after IoT attackers have a firm footing in the devices, they need to receive a series of commands or exchange key information between C&C server or peer bots based on a predetermined bound port. For example, *Mozi* botnet utilized its extended DHT protocol [31] to build a P2P network. In order to join their DHT network, new bots should bind the specified 6881 port to communicate with peers.

Table 2: Top abused sensitive network resources with bind operations. The first column indicates the behavior labels. The second column indicates the behavior types, and the third column presents the target ports of the behaviors. The forth column indicates the numbers of attack sessions that conducted the behaviors.

No.	Operation	Port	Count
11	Bind	23	498
12	Bind	22	481
13	Bind	80	477
14	Bind	8000	4
15	Bind	6881	2
16	Bind	8080	2
17	Bind	38273	2
18	Bind	34561	1
19	Bind	14737	1
20	Bind	443	1

Unlike this situation, some IoT attackers (a total of 507 attack sessions) attempt to kill processes of some fragile services and bind corresponding ports to fend off the following competitors. For example, remote control services like SSH (22), Telnet (23), HTTP(80) with default, factory-set weak passwords, or corny exploitation is almost the useless security barrier against any attackers. We come across that IoT malware families, like *Gafgyt*, proactively bind and listen to these ports but never accept any sessions to maintain the monopoly and ward off adversaries after taking over the device in preference. Apart from these easy-to-copy infection methods, complicated or novel exploitations based on other network ports acquire more technical skills and raise the infection threshold, thus lowering the chance of sharing devices with other attackers. Therefore, the commonly bound ports represent the most vulnerable services prone to be attacked.

In addition, a few attackers also bind some unusual local ports to ensure a single instance of itself running on the device. For example, some *Mirai* malware variants, like *Satori* or *Hito*, define and set their SINGLE_INSTANCE_PORT macro to 38273 or 34561 port to maintain single running instance. If malwares fail in binding, it may request the process termination by connecting to that port or wait for a while to kill the running process.

Connected Network Ports. Many IoT attackers launch external communications. The purpose of these connection behaviors can be discovered and concluded based on their target network ports, as exhibited in Table 3.

For further proliferation, IoT attackers either brute-force or exploit vulnerable services under specific ports to gain remote control of target devices. Connect behaviors in 4,670 attack sessions target at ports like 23, 2323, 22, which are commonly used for services like *Telnet* or *SSH*. IoT attackers can attempt to connect them to gain access to IoT devices, which is a basic way for infection. Apart from it, we observe that occasional connections to some rare ports (6,636 attack sessions) are closely associated with vulnerable services in certain IoT devices. For example, *Satori*, a new Mirai branch, exploits CVE-2017–17215 [14] vulnerability of Huawei HG532 routers on the TCP port 37215. In the vein of *Satori*, *Omni* compromises

Table 3: Top abused sensitive network resources with connect operations. The first column shows the behavior labels. The second column indicates the behavior types, and the third column presents the target ports of the behaviors. The forth column indicates the numbers of attack sessions that conducted the behaviors.

No.	Operation	Port	Count
21	Connect	37215	6,690
22	Connect	8080	6,653
23	Connect	81	6,457
24	Connect	8443	6,412
25	Connect	5555	6,401
26	Connect	52869	6,383
27	Connect	23	4,486
28	Connect	80	423
29	Connect	2323	235
30	Connect	22	164

Table 4: Top abused sensitive process resources. The first column indicates the behavior labels. The second column indicates behavior types, and the third column presents the target processes of the behaviors. The forth column indicates the numbers of attack sessions that conducted the behaviors.

No.	Operation	Target Process	Counts
31	sigkill	dropbear	3,569
32	sigkill	telnetd	3,567
33	sigkill	netifd	3,522
34	sigkill	logread	3,095
35	sigkill	odhcpd	3,075
36	sigkill	urngd	3,069
37	sigkill	wpad	1,840
38	sigkill	rcsysntpd	42
39	sigkill	dnsmasq	29
40	sigkill	logd	2

Netgear R7000 and R64000 devices via the CVE-2016-6277 [13] exploitation on port 8443. Obviously, by understanding connect behaviors targeted at special port resources, we can identify the scope of the vulnerable devices in danger of IoT attacks.

Furthermore, IoT attackers also establish malformed connections to launch DoS attacks. These steep and massive malicious connections attempt to disrupt the normal interaction of a targeted server. Theoretically, DoS attacks can be directed at any service on any device. However, to achieve significant damage, such attack traffic is typically aimed at some vital open services, like web applications with the HTTP(80) port.

Apart from the above illustration, we also note that in terms of intrusion methods, variants with vulnerability exploitation roughly equate to ones with password cracking nowadays, which is significantly different from the predominant brute-force based infection during the initial outbreak of IoT attacks.

Finding 2: 60.9% of the attack sessions with network behaviors switch to exploitations targeting diverse ports. These abused network resources reveal vulnerable services on fragile devices under attacks. It prompts defenders to shield these network services, like modifying default ports.

4.4 Process Resources

The malicious operations of IoT attackers to process resources are concerned with the immediate termination of particular system service processes. As featured in the Table 4, these killed processes include services responsible for devices' basic functionalities.

After infection, IoT attackers use their own rules to kill specific processes based on process names or identifiers. Through continuous behavior analysis, we find that these rules for killing processes are basically devised for network functions. For example, 3,569 attack sessions destroy the *telnetd* or the *dropbear* process to deny the remote access to the already occupied device. Moreover, a few attack sessions (3,111 attack sessions) terminate the *urngd* or *sysntpd* process to disrupt the random number generator and affect time synchronization by NTP protocol, which may damage the TLS-encrypted connections of IoT devices. Apart from this, a number of attackers try to cover their tracks by erasing logs and further disabling logging services to retain access for a longer time. More importantly, by reverse engineering malware samples, we observed that some attackers use a white-listing mechanism to only keep a few necessary processes, such as the init process with PID 1, while other attackers utilize black-listing rules to terminate certain processes, such as numeric named processes or processes of known malware adversaries. Some strategies cause IoT attackers (3,076 attack sessions) to kill processes responsible for critical functionalities aggressively, like *odhcpd* (DHCP server) or *wpad* (802.1x authentication service), seriously affecting the basic usability of network-related IoT devices like routers and modems. This warns us that the impact of many IoT attacks is not confined to the limited implications such as DDoS or crypto mining mentioned in the previous studies, but also has the potential to severely influence the device usability.

Finding 3: Processes responsible for critical functionalities are seriously damaged in 29.84% of the attack sessions with process resource abuses, degrading the usability of devices. Additional daemon programs for basic processes are required to ensure devices' essential functions.

4.5 Special Capability Resources

Special system capabilities are widely abused by attackers. They try to conduct privileged operations, such as special system calls and restriction override attempts to further compromise the device. We systematically analyzed such malicious behaviors, and the most prevalent ones are shown in Table 5. Specifically, these operations are mainly used for the following purposes.

Table 5: Top abused sensitive special capabilities. The first column indicates the behavior labels. The second column indicates the behavior types, and the third column presents operation descriptions. The forth column indicates the numbers of attack sessions that conducted the behaviors.

No.	Operation	Description	Count
41	sys_ptrace	Attach debugger	7,144
42	rename_process	Modify the process name	4,103
43	kill	Gain signal permissions to other processes	3,113
44	dac_override	Bypass DAC restrictions	3,046
45	dac_read_search	Bypass restrictions of read and search	513
46	setuid	Change UID	100
47	compute_av	Calculate the Access Vector	50
48	setfscreate	Override the default file context	31
49	sys_nice	Change the process priority	2
50	sys_time	Change system time	2

Detect Analysis Environment. Evasion from analysis environments is often employed by attackers to prevent their behaviors from being exposed to security researchers. Among our captured samples, 7,144 attack sessions invoked *sys_ptrace* to detect whether it is currently being debugged. Since one process can only be attached to one debugger, malware samples attempt to debug themselves before subsequent malicious operations are conducted. If the attempts are failed, it means that the process is currently being debugged by another program. This illustrates the limitations of some current dynamic analysis techniques to an extent. More obscure analysis mechanisms need to be used to avoid such evasion from hindering offline sample analysis.

Escalate Privilege. Some attackers abuse special capabilities to achieve privilege escalation. For example, 3,046 attackers attempt to bypass the directory access control mechanism (DAC), and 513 attack sessions try to bypass the read and search limit on sensitive files. Since such permissions are too powerful and may cause arbitrary reading and writing on the filesystem, it is disabled by default by the system of IoT devices. Only binaries with special attributes can have the permissions after execution. However, some virtual environments (e.g., the early version of docker) may execute with such properties by default [20], and in these cases, attackers can exploit this permission to achieve container escape. This illustrates that some IoT attackers have the intention to escape from an isolation analysis environment if possible. It warns the research community always to pay attention to the virtual machine's configuration when analyzing malwares, preventing the samples from invading the host. Meanwhile, 100 attack sessions invoke system calls like *setuid* to try manipulating processes' UIDs to gain higher permissions. 31 attacks, on the other hand, attempted to manipulate the security contexts of their created files with *setfscreate* to bypass SELinux restrictions. Also, a small proportion of attackers (2 attack sessions) tried modifying the priority of other processes utilizing the *sys_nice* capability to acquire more resources from the kernel.

Mask the footprint. Some special capabilities may help attackers to mask their footprints and make their existence more invisible to users. For example, 4,103 attack sessions try to forge their process name with system calls like *prctl*. The top faked process names include *dropbear*, *busybox* and even empty process name, which can deceive normal users when checking their device status. A few samples (2 attack sessions) try to modify the hardware system clock to set obstacles for log analysis. System logs with completely messed timestamps will make it more difficult for users to analyze the recent system records of the device.

Finding 4: 53.77% of attack sessions with special capability abuses adopt evasive techniques. It indicates us to conceal analysis procedures more carefully or strike back by camouflaging devices into analysis-like environments to deactivate such attacks.

4.6 Defense Strategy Outlook

Through in-depth analysis of the malicious intentions, we obtain several important conclusions about the attackers' abuses of file, network, process, and special system capability resources. These findings shed light on defense mechanisms for IoT devices. Now we discuss the implications for researchers and software engineers, as well as possible safeguard solutions based on these findings.

Using existing kernel modules like FUSE [16] to monitor the operations of a few specific directories can effectively help to detect attackers with low overhead. According to Finding 1, file resource abuses are concentrated on specific targets. 83.57% of the attack sessions have conducted operations on the same six file directories. This reveals that we can monitor the access operations in real-time to these several paths targetedly to detect possible attackers with kernel modules like FUSE. When the monitor target is limited to only a few specific paths, the resource consumption of such module is relatively low. Overall, combined with Finding 1, we can use this method to detect possible attackers directly on devices with acceptable runtime costs.

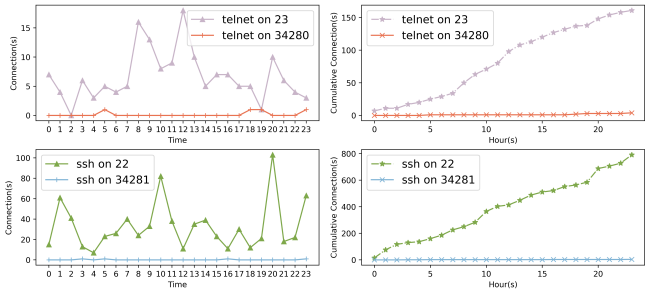

Figure 4: Attack connection times of *SSH* and *Telnet* services with different ports. The left graphs present the connection times per hour, while the right graphs indicate the cumulative times of connections.

Changing default ports of vulnerable services can circumvent the proliferation attempts of IoT attackers and prevent the device from being infected through the network. Finding 2 reveals the malicious intentions to scan, proliferate, and monopolize devices through bound and connected ports, which in turn unveils vulnerable services that IoT attackers often target. Based on this, we can avoid such attempts by adjusting the ports of the vulnerable services identified in this insight. For example, as shown in Figure 4, a total of 791 attack connections targeting the *SSH*

service on port 22 during the day. When we change its port to a relatively random one (34281 port), the number of attacks drops to 4. The same trend can also be seen in the attacks targeting *Telnet* service. Based on this observation, IoT firmware developers can prevent many invasions without much effort by simply designating another ports for services that are prone to be attacked.

Additional daemon programs should be implemented to ensure the normal execution of processes responsible for critical functionalities to protect IoT devices' usability. Finding 3 confirms a situation worthy of attention: Some IoT invaders will not only cause relatively mild impacts, but also prey on critical system processes, severely affecting the basic functionalities of the device. Therefore, daemon programs should be customized for IoT devices' essential services, resuming critical processes at once when interrupted to ensure usability. For network-related devices, for example, the daemons should always keep the normal working of processes like *odhcpd* (DHCP Server) and *wpad* (WiFi-related process), which have been affected in 3,076 attack sessions during our experiment. Additional demons that guard these fundamental processes can thus ensure that the functionalities of IoT devices are still guaranteed during these attacks.

Evasive techniques adopted by invaders can conversely be leveraged by us through deliberately faking an analysis environment to deactivate such IoT attacks. Finding 4 elaborates the attackers using special system capabilities to achieve malicious intentions like reconnaissance. Malwares often terminate themselves as soon as they discover the existence of an analysis environment. This makes security research more difficult, but it also opens up the possibility to take advantage of this feature in reverse to defend against them. For example, up to 7,144 attack sessions utilize *sys_ptrace* capability to detect the possible existence of a debugger and immediately exit if such analysis environment is found. Attacks like these can be directly defended if we fake attaching a debugger to the program. Likewise, many other anti-analysis methods can also be used in reverse like this to deactivate such evasive IoT attacks.

4.7 Summary of Implications

The above conclusions give some important implications for IoT firmware developers. We suggest that software engineers follow the process shown in Figure 5 to reinforce the security of the firmware at both pre-release and run-time stages.

Pre-release Stage. In addition to the usual software testing process, we recommend that developers pay attention to the following two points before releasing the firmware. (1) Identify the common network services present on the device and modify the default port. According to Finding 3, attacks often scan for ports used by common vulnerable services to perform proliferation. Such an attack can be effectively circumvented by changing the default port according to our experiment shown in Figure 4. (2) Add additional fingerprint files of common analysis environments to deactivate evasive malwares. Based on Finding 4, developers can utilize IoT malwares' reconnaissance process in reverse. By adding a forged fingerprint of the analysis environment to the device, evasive IoT malwares will limit their malicious behaviors or even exit after detecting such characteristics, deactivating such attacks with very low overhead.

Run-time Stage. We offer the following two suggestions for software engineers to better protect IoT firmwares at run-time. (1) When designing defense frameworks, pay attention to monitoring several file paths that are often abused. From Finding 1, we can observe that malwares have a high concentration of access to file resources. Therefore, when designing a defense mechanism, developers can also take advantage of this by focusing on a few directories that are often abused by attackers, thus achieving an effective defense with low run-time costs. (2) At firmware run-time, it is necessary for developers to design daemons to keep critical services running properly. Process resource abused by attackers shown in Finding 2 reminds us of the great harm that IoT malwares can have on devices' availability. Therefore, it also motivates software engineers to design suitable daemons to guard the proper functioning of the devices' core services.

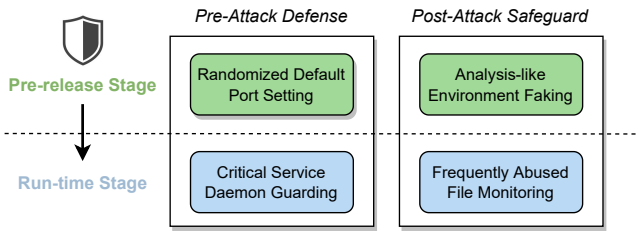

Figure 5: Implications for IoT security reinforcements at the pre-release and run-time stages.

5 THREATS TO VALIDITY

We deployed HoneyAsclepius worldwide and systematically analyzed abused system resources and malicious intentions behind. There still remain several concerns derived from the components.

Scalability of Honeypot Deployment. Some previous research like [12] tend to purchase from ISP to get Internet access to physical IoT devices with about 2 times costs compared with VPS instances, which seriously limited their deployment scalability. While with our designed HoneyAsclepius, physical devices can be virtually deployed based on the Traffic Forwarder module with public IPs obtained from multiple cloud infrastructure vendors, achieving large-scale deployment with 60 devices distributed among ten countries in five continents. In future research, we will further expand the scale of our deployment and collect more diverse attack behaviors.

Possible IP Range Bypass From Future Attackers. The cyber arms race between hackers and the defenders fuels the evolution of IoT attacks. We have noticed malware variants of some families like *Mirai* deliberately avoiding hard nuts or unattractive ones, such as General Electric Company or United States Postal Service, when setting up the target IPs for their attacks. Future IoT attackers may skip the IP range of cloud services to avoid being captured by honeypots. Some public cloud providers, like Google Cloud or AWS, generously release their concrete IP addresses available on the Internet [6] [3]. These IP features facilitate IoT attackers to evade the cloud-based IoT honeypots in the future, reducing the efficiency of capturing malwares. Therefore, we need to diversify the number of cloud providers for honeypot deployments.

Ambiguity of Malware Classifications. We found inconsistencies in the classification of collected IoT malware variants when

using AVClass and VirusTotal. Specifically, 0.88% variants are labelled with distinct names of similar IoT malware families. Based on our manual analysis, the great code similarity and architectural diversity frustrate several classification signatures, leading to the misclassified variants. In this situation, aggregating the analysis results from multiple tools can harvest more fruits. Although the classification of samples is not the key point of this paper, we still suggest that the further optimization of malware classification during pre-processing are required to gather more accurate labels by fingerprinting or clustering malwares based on different aspects, like functions, ELF headers or CFG graphs.

Possible Limitation From OS Selection. HoneyAsclepius' customized firmware is implemented based on Linux operating system with SELinux kernel module to enable continuous behaviour monitoring, which might impose limitations on validity. Though Linux is widely applied on IoT devices, very low-end devices still cannot run such an operating system because of their severely constrained hardware resources and, therefore, cannot be adapted to HoneyAsclepius. This also makes our empirical study mainly focuses on the abuses of system resources by attackers targeting at Linux-based IoT devices. Our study does not cover other malicious operations against embedded devices with operating systems like RTOS.

6 RELATED WORK

IoT Honeypots. Researchers and industrial practitioners have designed multiple honeypots to trap and collect malware samples. These research honeypots are used to learn countermeasures against attackers. Honeyd [37], as a low-interaction honeypot, configures virtual hosts on the network to run arbitrary services and provide traffic threat detection and assessment. Cowrie [8] and Kippo [36] are medium-interaction honeypots that can record brute force attacks and shell interactions through *SSH* or *Telnet*. Apart from this, there are few honeypots designed for capturing IoT malwares. IoTPOT [34] equipped with IoTBOX captures Telnet-based IoT malwares based on low-interaction virtual IoT devices with various architectures, and conducts simple network traffic statistical analysis. On the contrary, SIPHON [21] leverages a few physical devices exposed under cloud services by traffic forwarding to achieve high-interaction. It additionally deployed a storage unit to record raw network traffic and parse the recorded *pcaps* files to obtain the basic features of each TCP connection. IoTCandyJar establishes an intelligent-interaction honeypot for IoT attacks. This work focuses on imitating the interaction of different IoT devices from distinct vendors by machine learning.

Unlike the above-mentioned research, our study implements HoneyAsclepius with a high-interaction back-end that concentrates on automatically monitoring malicious behaviors to various resources. It provides a comprehensive understanding of *system resource abuses across the IoT attack landscape*. Meanwhile, it exposes malicious behaviors related to outside network environments, achieving *timely analysis* of malware samples that might become dormant when executed offline.

Surveys on Malwares. With profit-driven businesses, malware developers actively make efforts on various platforms [5]. Early malwares were designed for the Windows platform, and research like [35] [38] [41] [18] [24] [19] mainly focuses on the their features. Since 2004, some researchers like Guo et al. [22] have already predicted the risks of malwares targeted at embedded devices like smartphones and alarmed the public about the damage from privacy violations, identity theft to DDoS attacks. Felt et al. [17] study incentives behind 46 mobile malwares in the wild and discuss some predictable incentives. Vidas [42] analyzes the Android security model to determine whether and where the vulnerability exists and proposes six mitigations for the identified problems. As illustrated in Introduction (Section 1), there are also some empirical studies [4] [9] [12] [7] focusing on IoT attacks. In contrast to these studies, our paper focuses on comprehensively unveiling and analyzing the *system resource abuses* across IoT attack landscape and exploring the *malicious intentions* behind their operations.

7 CONCLUSION

In this paper, we systematically summarize the system resources abused by prevalent IoT attacks, analyze their malicious intentions, and further present insights and defense strategies. To achieve this, we implement and deploy high-interaction honeypots, HoneyAsclepius, which have a high capture capability and a continuous behavior monitoring mechanism to immediately record malicious behaviors and resource abuses in active attack sessions. Through large-scale experiments worldwide, we collected 117,862 valid attack sessions and gathered 11,301,239 behaviors. This provides the basis for our subsequent intention processing. We further separate behaviors in different attack sessions, estimate their temporal relations, and finally lead to findings of resource abuses of distinct categories as well as malicious intentions in IoT attacks.

We expect our provided data and proposed insights can help realize better detection and defense for IoT devices. (1) For file resources, malicious behaviors are mostly focused on a few specific directories, which helps us to detect IoT attackers by monitoring a small number of file path accesses. (2) In network resource abuses, ports bound and connected by attackers hint at current vulnerable services and devices, and users can circumvent attackers' proliferation by modifying default ports of fragile services. (3) Meanwhile, misuse of process resources reminds us the impact of IoT attacks is not limited mild ones like DDoS or crypto mining mentioned most in previous research, but can also severely influence the devices' basic functionalities, requiring us to implement daemon programs to ensure ctheir usability. (4) Attackers' abusing special capabilities for anti-analysis is a wake-up call to researchers. In reverse, we can take advantage of such attempts, deactivating these attacks with fake analysis environment.

ACKNOWLEDGMENTS

This research is sponsored in part by the NSFC Program (No. 62022046, 92167101, U1911401, 62021002, 62192730), National Key Research and Development Project (No. 2019YFB1706203, No. 2021QY0604).

REFERENCES

[1] Arwa Abdulkarim Al Alsadi, Kaichi Sameshima, Jakob Bleier, Katsunari Yoshioka, Martina Lindorfer, Michel van Eeten, and Carlos H. Gañán. 2021. *No Spring Chicken : Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis*. Vol. 1. Association for Computing Machinery. https://doi.org/10.1145/3488932.3517408

[2] 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. https://www.usenix.org/conference/usenixsecurity21/presentation/alrawi-circle

[3] Amazon. 2022. AWS IP address ranges - AWS General Reference. https://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html. (Accessed on 05/06/2022).

[4] 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

[5] AV-ATLAS. 2022. Malware & PUA. https://portal.av-atlas.org/malware/statistics. (Accessed on 05/06/2022).

[6] Google Cloud. 2022. The instance ip address range of Google Cloud. https://www.gstatic.com/ipranges/cloud.json. (Accessed on 05/06/2022).

[7] Andrei Costin. 2018. IoT Malware : Comprehensive Survey , Analysis Framework and Case Studies. Black Hat USA.

[8] Cowrie. 2022. Welcome to Cowrie's documentation! — cowrie 2.3.0 documentation. https://cowrie.readthedocs.io/en/latest/. (Accessed on 05/04/2022).

[9] Emanuele Cozzi, Mariano Graziano, Yanick Fratantonio, and Davide Balzarotti. 2018. Understanding Linux Malware. In *2018 IEEE Symposium on Security and Privacy (SP)*. 161–175. https://doi.org/10.1109/SP.2018.00054

[10] Emanuele Cozzi, Pierre-Antoine Vervier, Matteo Dell'Amico, Yun Shen, Leyla Bilge, and Davide Balzarotti. 2020. The Tangled Genealogy of IoT Malware. In *Annual Computer Security Applications Conference* (Austin, USA) *(ACSAC '20)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3427228.3427256

[11] Emanuele Cozzi, Pierre Antoine Vervier, Matteo Dell'amico, Yun Shen, Leyla Bilge, and Davide Balzarotti. 2020. The Tangled Genealogy of IoT Malware. *ACM International Conference Proceeding Series* (2020), 1–16. https://doi.org/10.1145/3427228.3427256

[12] 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 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 482–493. https://doi.org/10.1145/3307334.3326083

[13] National Vulnerability Database. 2016. NVD - CVE-2016-6277. https://nvd.nist.gov/vuln/detail/CVE-2016-6277. (Accessed on 05/06/2022).

[14] National Vulnerability Database. 2017. NVD - cve-2017-17215. https://nvd.nist.gov/vuln/detail/cve-2017-17215. (Accessed on 05/06/2022).

[15] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi. 2018. DDoS-Capable IoT Malwares: Comparative Analysis and Mirai Investigation. *Security and Communication Networks* 2018 (2018). https://doi.org/10.1155/2018/7178164

[16] Paul R. Eggert and D. Stott Parker. 1993. File Systems in User Space. In *USENIX Winter 1993 Conference (USENIX Winter 1993 Conference)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/usenix-winter-1993-conference/file-systems-user-space

[17] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. 2011. A Survey of Mobile Malware in the Wild. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices* (Chicago, Illinois, USA) *(SPSM '11)*. Association for Computing Machinery, New York, NY, USA, 3–14. https://doi.org/10.1145/2046614.2046618

[18] Carlos Gañán, Orcun Cetin, and Michel van Eeten. 2015. An Empirical Analysis of ZeuS C&C Lifetime. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security* (Singapore, Republic of Singapore) *(ASIA CCS '15)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/2714576.2714579

[19] Gazet and Alexandre. 2010. Comparative analysis of various ransomware virii. *Journal in Computer Virology* 2010 (2010). https://doi.org/10.1007/s11416-008-0092-2

[20] Aaron Grattafiori. 2016. Understanding and Hardening Linux Containers. *NCC Group Whitepaper* (2016), 1–123. https://research.nccgroup.com/wp-content/uploads/2020/07/ncc_group_understanding_hardening_linux_containers-1-1.pdf

[21] Juan David Guarnizo, Amit Tambe, Suman Sankar Bhunia, Martin Ochoa, Nils Ole Tippenhauer, Asaf Shabtai, and Yuval Elovici. 2017. SIPHON: Towards Scalable High-Interaction Physical Honeypots. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security* (Abu Dhabi, United Arab Emirates) *(CPSS '17)*. Association for Computing Machinery, New York, NY, USA, 57–68. https://doi.org/10.1145/3055186.3055192

[22] Chuanxiong Guo, Helen J. Wang, and Wenwu Zhu. 2007. Smart-Phone Attacks and Defenses.

[23] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. 2019. Measurement and Analysis of Hajime, a Peer-to-peer IoT Botnet. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/measurement-and-analysis-of-hajime-a-peer-to-peer-iot-botnet/

[24] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C. Snoeren, and Damon McCoy. 2018. Tracking Ransomware End-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*. 618–631. https://doi.org/10.1109/SP.2018.00047

[25] IPinfo.io. 2022. Comprehensive IP address data, IP geolocation API and database. https://ipinfo.io/. (Accessed on 05/04/2022).

[26] Monnappa KA. 2015. Automating Linux Malware Analysis Using Limon Sandbox. In *Black Hat Europe 2015*.

[27] Raphaël Khoury, Benjamin Vignau, Sylvain Hallé, Abdelwahab Hamou-Lhadj, and Asma Razgallah. 2021. An Analysis of the Use of CVEs by IoT Malware. In *Foundations and Practice of Security*, Gabriela Nicolescu, Assia Tria, José M. Fernandez, Jean-Yves Marion, and Joaquin Garcia-Alfaro (Eds.). Springer International Publishing, Cham, 47–62.

[28] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and Other Botnets. *Computer* 50, 7 (2017), 80–84. https://doi.org/10.1109/MC.2017.201

[29] Artur Marzano, David Alexander, Osvaldo Fonseca, Elverton Fazzion, Cristine Hoepers, Klaus Steding-Jessen, Marcelo H. P. C. Chaves, Ítalo Cunha, Dorgival Guedes, and Wagner Meira. 2018. The Evolution of Bashlite and Mirai IoT Botnets. In *2018 IEEE Symposium on Computers and Communications (ISCC)*. 00813–00818. https://doi.org/10.1109/ISCC.2018.8538636

[30] Trend Micro. 2018. Miner Malware Targets IoT, Offered in the Underground. https://www.trendmicro.com/en_us/research/18/e/cryptocurrency-mining-malware-targeting-iot-being-offered-in-the-underground.html. (Accessed on 05/06/2022).

[31] 360 Netlab. 2019. Mozi, Another Botnet Using DHT. https://blog.netlab.360.com/mozi-another-botnet-using-dht/. (Accessed on 05/06/2022).

[32] Palo Alto Networks. 2017. New IoT/Linux Malware Targets DVRs, Forms Botnet. https://unit42.paloaltonetworks.com/unit42-new-iotlinux-malware-targets-dvrs-forms-botnet/. (Accessed on 05/06/2022).

[33] Palo Alto Networks. 2020. 8 Stages of the IoT Attack Lifecycle - Palo Alto Networks. https://www.paloaltonetworks.com/resources/8-stages-of-the-iot-attack-lifecycle. (Accessed on 08/28/2022).

[34] 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

[35] Phillip Porras. 2009. Inside Risks Reflections on Conficker. *Commun. ACM* 52, 10 (oct 2009), 23–24. https://doi.org/10.1145/1562764.1562777

[36] The Honeynet Project. 2022. desaster/kippo: Kippo - SSH Honeypot. https://github.com/desaster/kippo. (Accessed on 05/04/2022).

[37] Niels Provos. 2004. A Virtual Honeypot Framework. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13* (San Diego, CA) *(SSYM'04)*. USENIX Association, USA, 1.

[38] Darrel Rendell. 2019. Understanding the evolution of malware. *Computer Fraud & Security* 2019, 1 (2019), 17–19. https://doi.org/10.1016/S1361-3723(19)30010-7

[39] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. AVclass: A Tool for Massive Malware Labeling. In *Research in Attacks, Intrusions, and Defenses*, Fabian Monrose, Marc Dacier, Gregory Blanc, and Joaquin Garcia-Alfaro (Eds.). Springer International Publishing, Cham, 230–253.

[40] Smalley and Stephen. 2005. configuring the SELinux Policy. https://www.nsa.gov/portals/75/images/resources/everyone/digital-media-center/publications/research-papers/configuring-selinux-policy-report.pdf. (Accessed on 05/06/2022).

[41] Sam Stover, David Dittrich, John Hernandez, and Sven Dietrich. 2007. Analysis of the Storm and Nugache Trojans: P2P Is Here. *login Usenix Mag.* 32, 6 (2007). https://www.usenix.org/publications/login/december-2007-volume-32-number-6/analysis-storm-and-nugache-trojans-p2p-here

[42] Timothy Vidas, Daniel Votipka, and Nicolas Christin. 2011. All Your Droid Are Belong to Us: A Survey of Current Android Attacks. In *5th USENIX Workshop on Offensive Technologies (WOOT 11)*. USENIX Association, San Francisco, CA. https://www.usenix.org/conference/woot11/all-your-droid-are-belong-us-survey-current-android-attacks

[43] Meng Wang, Javier Santillan, and Fernando Kuipers. 2018. ThingPot: an interactive Internet-of-Things honeypot. https://doi.org/10.48550/ARXIV.1807.04114

[44] Shuofei Zhu, Jianjun Shi, Limin Yang, Boqin Qin, Ziyi Zhang, Linhai Song, and Gang Wang. 2020. Measuring and Modeling the Label Dynamics of Online Anti-Malware Engines. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2361–2378. https://www.usenix.org/conference/usenixsecurity20/presentation/zhu