A Near Real-Time Scheme for Collecting and Analyzing IoT Malware Artifacts at Scale

Conference Paper · August 2022

DOI: 10.1145/3538969.3539009

3 authors:

Joseph Khoury
University of Texas at San Antonio
9 PUBLICATIONS **39** CITATIONS
SEE PROFILE

Morteza Safaei Pour
San Diego State University
16 PUBLICATIONS **203** CITATIONS
SEE PROFILE

Elias Bou-Harb
University of Texas at San Antonio
153 PUBLICATIONS **2,761** CITATIONS
SEE PROFILE

A Near Real-Time Scheme for Collecting and Analyzing IoT Malware Artifacts at Scale

Joseph Khoury
The University of Texas at San Antonio
San Antonio, TX, USA
joseph.khoury@utsa.edu

Morteza Safaei Pour
San Diego State University
San Diego, CA, USA
msafaeipour@sdsu.edu

Elias Bou-Harb
The University of Texas at San Antonio
San Antonio, TX, USA
elias.bouharb@utsa.edu

ABSTRACT

The chronic proliferation of Internet of Things (IoT) botnet malware activities coupled with an unprecedented rise in security vulnerabilities convene a new world of opportunities for perpetrators and unveil a new set of hurdles in deriving relevant IoT malware intelligence. Such shortfall within the IoT paradigm exacerbates the capabilities for largely identifying the prevailing IoT malware threats, the origin of the IoT attacks, as well as, the security deficit associated with the IoT paradigm. Previous work has vastly studied IoT malware activities in the wild but has not profiled at a large scale malicious activities to collect in near real-time central IoT artifacts much-needed to understand and eventually elevate the security posture of the IoT ecosystem.

To this end, we propose in this work a near real-time collection scheme to collect and analyze at large IoT malware artifacts essential for understanding the prevalent cyber security risks. We leverage in this work a large network telescope comprising of 16.7 million IPs as one extensive honeypot to examine evidence of malicious IoT probes in the wild. Subsequently, we employ a deception technique to respond to these probes and eventually establish bogus connections to collect IoT malware artifacts. In only *120* hours of near real-time measurements, our proposed scheme collected *80,569,070* interactions originating from *30,190* malware-infected IoT devices. Accordingly, we derive pivotal IoT malware intelligence which includes system commands, file-less attacks evidence, payload URLs, Executable and Linkable Format (ELF) binaries, login credentials, malicious LDAP servers, and unique insights on the abuse of the recent Log4shell security vulnerability in distributing IoT malware binaries.

CCS CONCEPTS

• **Security and privacy → Embedded systems security**; **Network security**.

KEYWORDS

IoT security, IoT malware artifacts, IoT malware intelligence, network telescope, Log4Shell security vulnerability

ARES'22, August 2022, Vienna, Austria

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XX.XXXX/X.X

ACM Reference Format:
Joseph Khoury, Morteza Safaei Pour, and Elias Bou-Harb. 2022. A Near Real-Time Scheme for Collecting and Analyzing IoT Malware Artifacts at Scale. In *Proceedings of ACM Conference (ARES'22).* ACM, Vienna, Austria, 11 pages. https://doi.org/XX.XXXX/X.X

1 INTRODUCTION

By 2025, the total number of connected Internet of Things (IoT) devices worldwide is expected to reach 30.9 billion, generating 79.4 Zettabytes of data [21, 50]. This unprecedented surge is mainly associated with the fast-tracked digital transformation adopted by most organizations that are vastly becoming dependent on connected IoT devices. Although designed to be smart and effective, IoT devices still suffer from numerous vulnerabilities and in-design security flaws; continuously exposing them to various types of malicious scanning, cyber attacks, and malware infections originating from malware-infected IoT devices. That said, we identify a pressing need for collecting and analyzing a multitude of artifacts related to IoT malware activities in the wild to derive valuable IoT malware intelligence and eventually elevate the current security posture of the IoT ecosystem.

Indeed, overlooking the IoT malware activities currently occurring imposes a critical risk to the Internet security, and exposes the cyber space to a new world of cyber threat opportunities and newly surfacing security vulnerabilities. On the other hand, the shortfall of adequate large scale IoT malware artifacts collection scheme introduces a new set of challenges to the IoT security field; impeding by its turn security researchers in identifying the prevailing IoT malware threats, the origin of the IoT threats, and the current deficits pertained to the security posture of the IoT ecosystem. While different research works [13, 29, 32, 48, 57, 58] provided a detailed description and analysis on IoT devices and IoT malware and its inner botnet functionalities, none of these works is accentuating on the large collection of IoT artifacts to support large scale IoT malware evidence identification, acquisition, and analysis, as well as attacks and deficit related to the security of the IoT paradigm.

On the other hand, previous research [18, 36, 44, 51, 52] attempted to collect, identify, and analyze IoT malware activities at large using a significant number of honeypots to collect numerous artifacts (e.g., IoT malware binaries). However, the scalability and the management issues entangled with the passive nature of honeypots restricted large scale IoT malware artifacts collection. In the same vein, some other research works [1, 4, 25, 26] have used a multitude of data perspectives (e.g., Internet-scale network scans [19], Telnet honeypots [15] and network telescope probes [6, 11]) to characterize and analyze cyber threats targeting the IoT paradigm. However, such works lacked in presenting a large scale

enabled scheme; capable of collecting and analyzing the evolving IoT security events and thus profiling the prevalent IoT threats.

To this end, we aim to contribute towards the security of the IoT paradigm by enabling the large collection of IoT malware artifacts in the wild. We mitigate the existing shortfall of conventional IoT honeypots through a near real-time collection scheme that takes advantage of the large vantage size of a network telescope coupled with a deceiving technique to lure malware-infected IoT devices, hence collecting IoT malware artifacts at scale. We accomplish this by leveraging incoming network packets detected on a /8 network telescope and inferred as scanning probes originating from malware-infected IoT devices. For the deception technique, we utilize a flaw in the stateless scanning module of IoT botnet malware (i.e., Mirai botnet [4]) to respond to the scanning probes (i.e., TCP SYN) using crafted deceiving packets (i.e., TCP SYN-ACK) and thus interconnect with malware-infected IoT devices in the wild. The interconnections are then directed to a cluster of high-interaction IoT systems where all the activities are logged. Additionally, we leverage our proposed IoT malware artifact collection scheme to uncover intelligence related to the abuse of the Log4Shell security vulnerability [16] in distributing IoT malware binaries through a dedicated infrastructure.

To this end, we frame our paper's contributions as follows:

- We propose a near real-time scheme for collecting and analyzing IoT malware artifacts by employing a /8 network telescope (i.e., 16.7 million routable and unused IPs), a deception technique, and a cluster of high-interaction systems to interconnect with malware-infected IoT devices.
- We show using a flaw in the stateless scanning module of one of the most dominant IoT botnet malware families, the capabilities of our deception technique in crafting successful deception packets (i.e., TCP SYN-ACK) to respond to IoT malware scanning probes (i.e., TCP SYN) acquired from the network telescope.
- We capture in the span of only *120* hours *80,569,070* interactions originating from *30,190* malware-infected IoT devices in the wild. Our findings comprise of IoT malware intelligence related to system commands, file-less attack evidence, payload URLs, Executable and Linkable Format (ELF) binaries, and log-in credentials. We publish a raw sample of our recorded artifacts on GitHub and Google Drive [42] to assist security researchers in comprehensively profiling the prevailing cyber security threat targeting the IoT paradigm.
- We demonstrate through a real-world cyber security use case the unique capabilities of our proposed scheme in collecting intelligence pertained to the abuse of the recently surfacing Log4Shell security vulnerability in distributing IoT malware binaries compiled with different CPU architectures.

The rest of the paper is organized as follows. In the subsequent section, we present background information on some technicalities related to our proposed scheme, as well as provide information on the prevailing Log4Shell security vulnerability that we identify its abuse in distributing IoT malware binaries using our proposed scheme. In Section 3, we present a high-level description of our methodology. We detail in Section 4 the setup of our proposed scheme while emphasizing on its unique capabilities. In Section 5, we discuss the collected IoT malware artifacts, along with the derived IoT malware intelligence. In Section 6, we present a real-world IoT security use case that demonstrate the capabilities of our proposed scheme. We dedicate Section 7 to discuss the perceived potential improvements and future directions. In Section 8, we briefly review related literature while emphasizing on the premise of our proposed scheme. Finally, we conclude the paper in Section 9.

2 BACKGROUND INFORMATION

In this section, we provide background information on relevant efforts leveraging the network telescope as a valuable source for acquiring evidence pertained to cyber threats targeting the Internet in general and the IoT paradigm in particular. We also discuss all the gaps and discrepancies in stateless scanners that we mainly consider in our deception technique. Additionally, we present detailed information on the proliferated Lo4Shell security vulnerability that we demonstrate using our proposed scheme its abuse in distributing IoT malware binaries.

2.1 Network telescope

Network telescope encompasses routed, yet unused IP addresses responsible for capturing network traffic on the Internet. It is one extensive method to scrutinize a wide range of intriguing Internet phenomena [55]. Due to the absence of active hosts on the unused network telescope IP space, the traffic is strictly uni-directional and mainly provoked by unsolicited activities associated with malware worms and Internet backscatter radiation [5]. The network telescope is successfully leveraged in numerous works as a distinguished vantage point for investigating Internet probing activities [17], DDoS attacks [7, 39, 40], and Internet worm propagation [38]. The network telescope is also used as one of the main sources of information to characterize potential cyber threats targeting the IoT landscape. For instance, Kumar et al. [30], leverage a passive network telescope of approximately 4.7 million IP addresses to gather insights into whether home network IoT devices are compromised. Additionally, other works [4, 10, 25, 26, 46] utilize the network telescope to collect artifacts related to IoT malware activities, as well as infer IoT campaigns by examining scanning probes.

As stated in CAIDA's technical report [12], the size of the network telescope (i.e., size of monitored address space) plays a vital role in recording single host activities and characterizing visible events on the Internet. To that extent, in this work we leverage CAIDA's /8 network telescope comprised of 16.7 million IP addresses; offering a considerable vantage point for examining scanning probes pertained to malware-infected IoT devices in the wild compared to the limited monitoring and gathering capabilities of conventional IoT honeypots. Subsequently, we respond through our deception technique (detailed in Section 4) to the near real-time scanning probes acquired from the network telescope and eventually interconnect with malware-infected IoT devices in the wild to collect IoT artifacts including malware binaries, malicious commands, and log-in credentials.

2.2 Stateless scanners

Network scanning has always been used as a major source of intelligence gathering for either threat assessment in a defensive context or even for cyber attack preparation in an offensive context. Interesting information can be collected through different network scanning techniques including ping sweep to diagnose alive systems, port scanning to identify listening ports, and Operating System (OS) detection to determine the host's OS [9]. Accordingly, such information can exceedingly be used to coordinate cyber attacks on detected vulnerable systems, similar to the work of Dainotti et al. [17] who present an analysis of a horizontal scan of the entire IPv4 address space attributed to the Sality botnet known to target Session Initiation Protocol (SIP) servers.

Due to its importance, Internet-wide scanners were manifested by numerous and distinct implementations. For instance `Nmap` [37] a popular stateful network scanner is used for network discovery, administration, and security auditing. Leonard et al. [31] propose IRLscanner, a stateless scanner intended for Internet-wide service discovery. Similarly, Durmeric et al. [20] propose `ZMap` a fast Internet-wide stateless scanner that achieves the validation between the scanning requests and responses, by utilizing the source and destination IP addresses and initial sequence number fields of TCP scans.

These high-performance scanners do not record the destination IPs of the sent packets, mainly, to avoid any cost associated with keeping, updating, and deleting records during the scan. However, they incorporate information into the outgoing probe that is preserved on the way back to distinguish replies from backscatter. For example, using a specified 32-bit initial sequence number (ISN) in the initial TCP SYN packet can be reliably used to differentiate the response packet among all incoming traffic contrarily to using an arbitrary ISN 32-bit sequence. This technique also reduces the entire management storage requirement of the scanner to just 4 bytes [24]. Such field encoding can be exploited as an identification scheme. That said, Griffioen and Doerr [24] propose in their work a method to identify and fingerprint distributed scanners based on commonalities in the header fields.

Therefore, only IP/TCP header fields that (i) are not modified by intermediate routers, (ii) entirely preserved in the return packet or modified in a predictable way, such as the usage of the ISN field, or (iii) do not break protocol functionality can be used to embed some state information into the scanning probe. Otherwise, modifications to some other fields might disrupt the proper operation of IPv4 or TCP (e.g., IPv4 version field), or convey meaning in one direction (e.g., TOS, TTL, or IP ID), while other fields, might be altered along the way such as the IPv4 header checksum.

We note, that only the (i) source/destination IP addresses in the IPv4 header, and (ii) the source/destination ports, along with the (iii) sequence number in the TCP header satisfy the aforementioned requirements. However, in the case of a horizontal scan targeting a specific destination port, the destination port field can not be altered. As an example, a scanner has typically a key (k) and a validation algorithm (f) which represent a hashing function. The scanner calculate the $tcp.seq$ using Equation (1) and send the ACK packet, subsequently, the scanners validate every incoming packet using Equation (2).

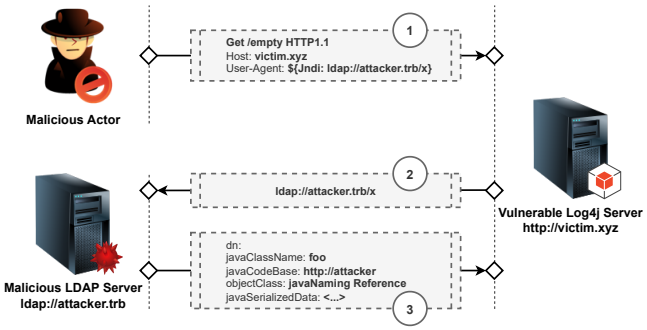

Figure 1: A high-level overview of the three steps used to exploit the Log4Shell security vulnerability and hence gain RCE capabilities.

$$tcp.seq = f(src\_port, dst\_port, src\_ip, dst\_ip, k) \quad (1)$$

$$f(dst\_port, src\_port, dst\_ip, src\_ip, k) = tcp.ack\_seq - 1 \quad (2)$$

Differently, Zmap [20] uses the AES algorithm along with a private key to validate the source port, destination IP, and TCP sequence number fields, while the Mirai botnet [3] and its variants utilize a less resource needed formula to calculate the $tcp.seq$ as in Equation (3) and then validate every incoming packet using Equation (4).

$$tcp.seq = dst\_ip \quad (3)$$

$$tcp.ack\_seq - 1 = src\_ip \quad (4)$$

That said, we extend our proposed IoT malware artifacts collection scheme with a deception technique that takes advantage of the discrepancy found in most stateless scanners to craft deceiving packets. Since Mirai and its variants are the predominant malware families, we employ Equation (3) and (4) associated with Mirai botnet generation and validation formulas to craft deception packets (i.e., TCP SYN-ACK) and hence establish bogus connection with malware-infected IoT devices in the wild.

2.3 Log4Shell security vulnerability

In this Section we discuss one of the very recent security vulnerabilities named Log4Shell that we detected its abuse in distributing IoT malware binaries. First, we provide background information on the Apache Log4j Utility. Second, we present information on the Log4Shell security vulnerability.

2.3.1 Apache Log4j utility. Log4j is an open-source, Java-based logging library developed by the Apache Software Foundation (ASF) as part of the Apache Logging Services. The Log4j utility can run across all major platforms (e.g., Windows, Linux, and macOS) and enables the logging of various data within an application, which is beneficial for debugging and a plethora of other purposes. This utility was first released in January 8, 2001, and has since been superseded by Log4j 2 in order to address the shortcomings of its

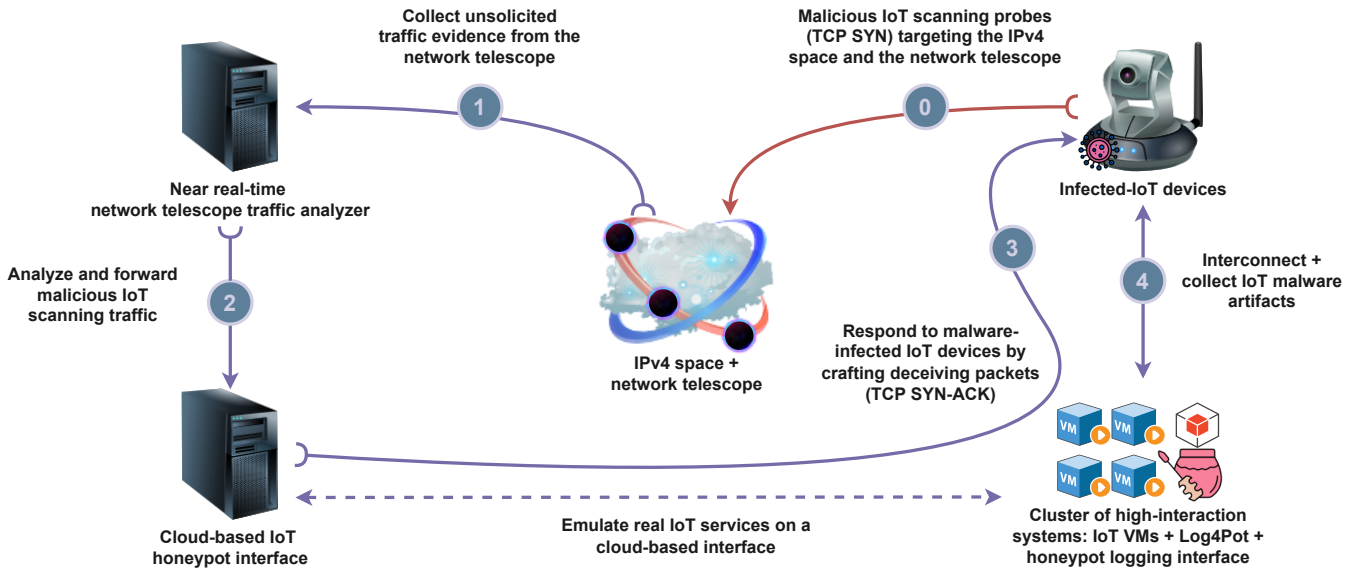

Figure 2: A chronological overview of our proposed near real-time IoT malware artifacts collection scheme using network telescope data. Using a large network telescope, a deception technique, and a cluster of high-interaction systems this scheme is capable of interconnecting with malware-infected IoT devices in the wild and hence collect and analyze IoT malware artifacts at scale.

predecessor, in addition to allowing users to define and configure custom components [35].

The Log4j utility is used by millions of machines spanning the globe running an assortment of online services [41]. The primary reason it is generally preferred over basic log formatting is its ability to perform lookups, such as those associated with context maps, system properties, and the Java Naming and Directory Interface (JNDI) [34]. Of these types of lookups, the JNDI API is leveraged to obtain naming and directory services from multiple available service providers, including Domain Name Service (DNS), Lightweight Directory Access Protocol (LDAP), and Remote Method Invocation (RMI), to name a few.

2.3.2 Log4Shell security vulnerability. Mainly referred to Log4Shell; the word 'Shell' is used due to the shell capability offered by this vulnerability. The Log4Shell vulnerability is tracked under CVE-2021–44228 [16] and is a zero-day, Remote Code Execution (RCE) vulnerability. In particular, an adversary can exploit a server running Log4j to execute any malicious software. Any Java application using a version of Log4j prior to 2.14.1 has this vulnerability.

The Log4Shell vulnerability comes into play due to the lack of necessary input sanitation of URLs not being performed by LDAP and JNDI requests, which enables adversaries to execute arbitrary Java code on a vulnerable server. An example of a generic malicious string that adversaries are aspiring to be logged is shown in Figure 1. Initially, at *step 1*, a vulnerable Log4J server comes across a JNDI request embedded in the User-Agent field of an HTTP request to obtain an exploit. At *step 2*, the server executes by designing the malicious JNDI request and contacts the malicious LDAP server. Finally, at *step 3*, the contacted malicious entity will distribute a malicious piece of code to ultimately gain RCE capabilities. Such capabilities may also lead to distributing malicious payloads including a ransomware and IoT botnet malware. Recently, several article [8, 23, 56] reported on the malicious abuse of the Log4Shell security vulnerability.

3 METHODOLOGY

Towards the aim of studying the current cyber security deficits targeting the IoT paradigm, this work focuses on collecting much-needed IoT malware artifacts in the wild using a large network telescope coupled with a deception technique and a cluster of high-interaction systems. Indeed, the IoT security field is confronting a series of challenges mainly associated with monitoring the prevailing IoT malware threats, the origin of the threats, as well as attack and deficit attribution related to the IoT paradigm. The goal is to address the existing gaps and shortfalls by proposing an IoT malware artifact collection scheme to extract valuable IoT malware intelligence to assist security researchers in understanding the prevailing security threats in the IoT paradigm. Figure 2 depict an overview of our proposed scheme. We provide below a high-level description of our methodology:

- We employ in our proposed scheme one large network telescope comprising of 16.7 million IP addresses, one server responsible for analyzing near real-time network telescope traffic, one cloud-based IoT honeypot interface comprised of a virtualized pool of IoT devices, one Log4Shell-related honeypot, and one honeypot logging interface. Using this scheme we enable the collection of artifacts related to IoT malware activities in the wild.
- We leverage our proposed scheme to (i) acquire and analyze evidence from the network telescope associated with IoT

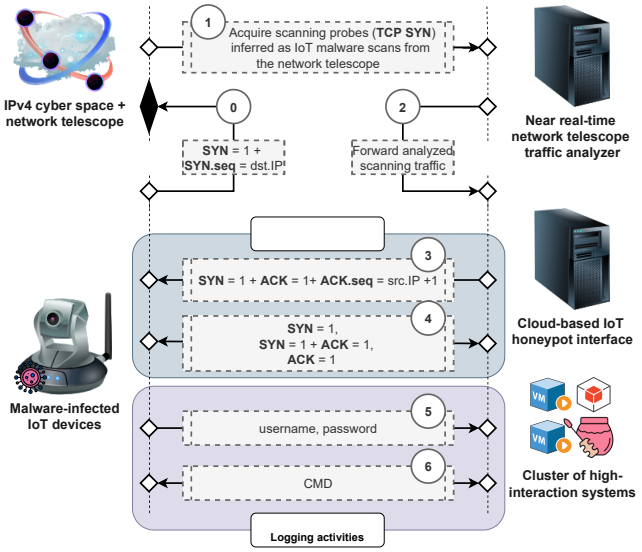

Figure 3: A step-wise detailed guide of our proposed IoT malware artifacts collection scheme.

malware scanning probes, (ii) respond to the scanning probes using our proposed deception technique, (iii) interconnect with malware-infected IoT devices in the wild to collect much-needed IoT malware artifacts.

- We derive central IoT malware intelligence which include system commands, file-less attacks evidence, payloads URLs, ELF binaries, log-in credentials. Additionally, we collect information associated with the abuse of the Log4Shell security vulnerability in distributing IoT malware binaries.
- **We abide by all the privacy and ethical considerations:** all the collected evidence including ELF binaries, IP addresses, URLs are well preserved at our side and are not being used to cause any harm. If the paper gets accepted, we will anonymize all the sensitive evidence reported throughout the paper.

In the next Section, we thoroughly discuss all the key elements of our proposed scheme along with all the inner details that we adopted to interconnect with a large number of malware-infected IoT devices; enabling the collection of numerous IoT malware artifacts during a limited period of time.

4 IOT MALWARE ARTIFACTS COLLECTION SCHEME SETUP

In this section, we elaborate on three key elements of our proposed scheme that include (i) collecting and analyzing network telescope packets associated with IoT scanning probes (ii) implementing a reactive deception technique to lure malware-infected IoT devices in the wild, and lastly (iii) building a cluster of high-interaction systems to interconnect with malware-infected IoT devices and eventually log their interactions. Throughout this section we refer to Figure 3 to describe all the detailed steps of our proposed scheme.

4.1 Collect and analyze network telescope packets

Malware-infected IoT devices are regularly sending scanning probes over all the IPv4 space including the network telescope to target vulnerable IoT devices in the wild. Figure 3 illustrates at *step 0* a scanning packet originating from a malware-infected IoT device and targeting the network telescope. In our proposed scheme, we leverage CAIDA's /8 network telescope as a large vantage point to acquire evidence pertained to these malignant probing scans.

At *step 1*, we acquire from the network telescope near real-time network flows related to unsolicited traffic, then, we analyze these flows to include only scanning probes associated with malware-infected IoT devices.

First, we accomplished this task by employing the Threshold Random Walk (TRW) based probing detection algorithm [28] using a threshold of 64 [47] to vet a particular network flow as a probing event. Second, since we are primarily interested in this paper in interconnecting with malware-infected IoT devices, we solely focus on the probing events targeting ports that are commonly scanned by IoT malware. For instance, Mirai and its different variants mostly scan port 23, 2323 (i.e., SSH) and port 22 (i.e., Telnet) [24].

To this point, the network telescope traffic analyzer is (i) processing in near real-time 150 GB of network flows per hour, (ii) applying the TRW algorithm to identify probing activities, (iii) selecting probing activities with specifically targeted ports, (iv) performing summary statistics for each scanning flow [22], and finally at *step 2*, (v) forwarding the analyzed scanning traffic to the cloud-based IoT honeypot interface.

4.2 Deceive malware-infected IoT devices

In its turn, the cloud-based IoT honeypot interface intercepts the already selected scanning probes and promptly respond at *step 3* by crafting deceiving packets using a Python library called Scapy [49] to deceive and ultimately establish bogus connection with the corresponding devices primarily initiating the scans.

We craft the response packets using a deception technique that builds upon the 'Mirai' signature (i.e., $dst.IP == TCP.seq$). In essence, we craft our deceiving packets (i.e., TCP SYN-ACK) by embedding a TCP ACK sequence number that equates to the IP address of our cloud-based IoT honeypot interface and subsequently send the packets to the intended malware-infected IoT devices.

We elaborate more on our deception technique using the following example: Initially, when we identify a scan from $IP_{infected\_device}$ inferred as 'Mirai' targeting port 23, using an arbitrary source port x, we accordingly craft and send a TCP SYN-ACK packet with the following TCP header fields $TCP.ack\_seq = (IP_{cloud} + 1)$, $TCP.src\_port = 23$, and $TCP.dst\_port = x$. In the case of a sequence of scanning packets having an arbitrary set of source port throughout, we respond to these packets by utilizing one of these ports. Whenever an infected device does not validate the incoming packets or does not use a validation algorithm that we already encode in the TCP SYN-ACK packets (in this case Mirai's validation algorithm which is $src.IP == TCP.ack\_seq - 1$), the malware on the infected device identifies our IP cloud interface as a potential vulnerable target. Therefore the malware starts sending full

TCP handshakes, as well as connections to Telnet or SSH services; illustrated by Figure 3, *step 4*.

At this stage, the cloud-based honeypot interface associated with the cluster of high-interaction systems is set to interact with malware-infected IoT devices in the wild. More details on the high-interaction systems setup are presented in the sequel.

4.3 Build a high-interaction setup

Our cloud-based IoT honeypot interface is defined by a cluster of high-interaction systems comprising of a virtualized pool of IoT devices along with an auxiliary Log4Shell related honeypot, and a honeypot logging interface. The main goal is to emulate IoT-related services (i.e., Telnet, SSH) and provide a vulnerable playground for the Log4Shell security vulnerability to enable the collection of numerous near real-time IoT malware activities originated from the wild. Figure 3 illustrates at *steps 5* and *6* the occurring interconnections, as well as the logging of the IoT malware activities.

That said, we employ a `Cowrie` honeypot [15] in a high-interaction setting (i.e., proxy) to utilize the Telnet and SSH proxies which can assist in providing a fully functional environment. The high-interaction setting of `Cowrie` empowers a dynamic pool of IoT Virtual Machines (VMs) along with a back-end system responsible for continuously running the IoT VMs. The back-end system reliably manages and assigns IoT VMs to any incoming connection preventing any duplicate assignment or resource miss-use. The virtualized IoT images in the back-end are emulated using a toolkit dedicated to managing virtualized platforms called `libvirt` [33] and an open-source generic machine emulator called `QEMU` [54].

Additionally, we employ `OpenWrt` [43] to emulate Linux-based operating systems that offers a realistic file-system with package management. By utilizing `OpenWrt` we provide malware-infected IoT devices the capability of manipulating files and packages which is important in showcasing the legit malicious activities that the IoT malware adapts during an interaction. We focus in this work on a single CPU architecture by employing the ARM-based `OpenWrt` images. Accordingly, we set up the IoT images will all the needed packages and settings including both Telnet and SSH services, `busybox` utility, network settings, firewall configurations, as well as predefined log-in systems that accept any combination of username and password. For security reasons, we limit all VMs' services to only Telnet and SSH to prevent any unwanted activities.

The cloud-based IoT honeypot interface is hosted on a Digital Ocean's droplet (i.e., 16 GB memory and 8 core Intel CPU) and the pool of 10 IoT images are emulated on a physical machine (i.e., 62 GB RAM, 16 core Intel Xeon W-2145 3.70GHz) which are recycled every one hour to prevent any changes that could be made during the malicious interactions. We also integrate into our cloud-based IoT honeypot interface a Log4Shell related honeypot called `Log4pot` [45] dedicated to (i) listen on multiple ports (i.e., 8080, 80, 443) to record possible Log4Shell related events, (ii) detect exploitation in the request lines and headers, and (iii) log all the events and payload URLs. In the following Section, we report on our findings and discuss the derived IoT malware intelligence.

5 IOT MALWARE ARTIFACTS

Our proposed scheme enabled through the interim of only *120* hours to empirically measure and collect central IoT malware activities. We recorded in January 2022, *80,569,070* interactions belonging to *30,190* malware-infected IoT devices in the wild. At first, all the logged interactions are dumped in a typical Cowrie Honeypot *.log* file that we analyzed using different python scripts that we developed to extract insightful intelligence pertained to IoT malware activities in the wild. That said, we thoroughly discuss in this Section all the artifacts that we investigated which include system command artifacts, file-less attacks evidence, IoT malware binaries, CC communications, log-in credentials. We also publish on GitHub and Google Drive [42] a raw sample of our recorded IoT malware activities as log files for the security community to investigate and understand the current security posture of the IoT paradigm.

5.1 System commands and file-less attacks

A vast majority of IoT devices run on a Linux-based OS, thus most attacks and attempts targeting Linux-based IoT devices apply Linux-related commands to interact and manipulate such devices. During our investigation, we identified 2, 650, 522 interactions including at least one BusyBox multi-call binary *'/bin/busybox'* command which provide many common UNIX utilities into a single small executable. During our analysis, we identify reconnaissance and foothold establishment-related activities that are known to Mirai and Hajime botnet malware and their variants. The identified activities are manifested as follows: (i) new connection initiated mostly using Telnet service, (ii) multiple log-in attempts using a set of username/password credential pairs (e.g., root/admin), (iii) Once a successful username/password combination is met, a series of shell scripts are sent which include 'enable', 'system', 'shell', and 'sh', and finally, (vi) a BusyBox command followed by the 'Mirai' word *'/bin/busybox Mirai'* or other sequence of characters (i.e., we elaborate on this below). The main purpose of sending the *'/bin/busybox'* command at the end is to test whether a Linux shell has actually been started based on the BusyBox response.

On the other hand, we detected different patterns of system commands using also the BusyBox utility. Although no clear indication to what IoT botnet malware, or Mirai variants they belong to, we observed a very significant number of hexadecimal data structures used as a sequence of 6 or 8 bytes succeeding the BusyBox command as follows: /bin/busybox echo -e '$\backslash x52\backslash x4C\backslash x59\backslash x51\backslash x51\backslash x4F$'. By converting the hexadecimal strings to ASCII, we realized that the echoed data structures translate to arbitrary upper-case English letter alphabets (e.g., RLYQQO) indicating a type of IoT botnet malware employing this tactic in all its interactions activities. While the first two interaction steps are similar to the Mirai botnet malware activities, the rest of the commands are mostly file-less attacks exhibited using UNIX commands (e.g., wget, rm, cat, dd, chmod, cp, hive-passwd). By examining the recorded artifacts we identified different types of file-less attacks [18] that we report on their behaviors and intent as follows:

- Damaging system data by copying, removing or altering specific files, as well as changing permissions. Our proposed scheme recorded numerous interactions using (264,448) *dd*,

Table 1: Top 10 identified artifacts related to log-in credentials used by malware-infected IoT devices to establish Telnet and SSH connections.

#	User/Password	Freq.	#	User/Password	Freq.
1	root/admin	358	**2**	admin/1234	293
3	root/root	272	**4**	root/aquario	259
5	root/	206	**6**	root/xc3511	205
7	root/123456	182	**8**	root/888888	177
9	ubnt/ubnt	176	**10**	root/vizxv	175

(240,431) *cp*, (228,733) *chmod*, and (228,354) *rm* UNIX commands.

- Stealing specific data by reading passwords or content from specific files. We recorded 1, 396, 049 interactions using the *cat* command.
- Retrieving malicious payloads or launching network attacks by contacting specific IP addresses. Our proposed scheme was able to record 667, 891 and 223, 847 interactions using the *wget* and *curl* commands respectively.
- We also examined a low number of other UNIX commands used to retrieve system information using *ps*, change the password of the system using the *hive-passwd*, and kill specific processes using the *kill* command.

5.2 IoT botnet malware artifacts

Through our analysis, we recognized the presence of more than 20 IoT botnet malware variants each of which utilizing a specific string/name in their BusyBox interactions to facilitate their identification. For instance, we identified 31, 583 interactions utilizing a randomized 5 upper-cased characters, a technique used by Hajime malware. In addition, we examined 702 interactions pertained to 'TFTP', 390 interactions related to 'V3G4', 224 associated with 'Demons', 173 to 'WICKED', to name a few. This shows a snapshot of the current security posture of the IoT ecosystem where there is a significant number of IoT malware abusing and battling over vulnerable IoT devices in the wild. Moreover, we recorded 129, 466 URLs from which only 1992 are unique URLs used to install ELF binaries pertained to IoT malware compiled to serve distinct CPU architectures. Furthermore, we recorded different username/password combinations that are currently used by malware-infected IoT devices to log into Telnet and SSH services. Table 1 present a list of the top 10 log-in credentials used.

6 REAL-WORLD CYBER SECURITY USE CASE: LOG4SHELL IOT MALWARE DISTRIBUTION

In this Section we employ our proposed scheme to identify and investigate a real-world cyber security event currently occurring in the wild and affecting the security posture of the IoT paradigm. By extending our proposed scheme with a Log4Pot interface [45], we were able to collect additional intelligence to examine the abuse of the Log4Shell security vulnerability in the IoT paradigm. Using the collected interactions, we identified artifacts related to (i) mutated JNDI lookups and exploits, (ii) malicious LDAP servers, payload URLs, and more interestingly, (iii) dedicated Log4Shell-IoT attack infrastructure employed to distribute IoT botnet malware binaries.

6.1 Log4Shell recorded interactions and obfuscation tactics

Our artifacts analysis indicates that 261, 536 events were recorded out of which 346 are Log4Shell exploits while the remaining 261, 190 events are considered Internet noise which include arbitrary malformed HTTP post and get requests. After further investigation of the identified Log4Shell exploits, we detected only 40/346 distinct exploits originating from 11 unique IP sources.

Although the number of exploits is low, yet, valuable information can be derived from the recorded artifacts. Initially, the first step in the Log4Shell vulnerability is executed by embedding a JNDI lookup request in the *User-Agent* header field of the HTTP protocol; previously discussed in Section 2. Nonetheless, we discovered that in almost each exploits (i.e., a total of 346 exploits) the JNDI request was embedded in multiple HTTP header fields and not only embedded in the *User-Agent* field. By examining the 346 Log4Shell exploits we determine that 76 out of the 100 existing HTTP headers were indeed used to embed the JNDI request, such headers include *Origin*, *Warning*, *Cookie*, *Authorization*, and many other fields. We observe that most of the recorded JNDI requests used distinct obfuscation methods to bypass available detection techniques (e.g., firewalls). Table 2 showcases the nested string obfuscation technique used on both the 'JNDI' interface and the 'LDAP' protocol words. Such findings provide security researchers insights into the various mutations tactics applied by malicious actors to alter the original exploits, and hence bypass existing detection techniques. In other words, investigators can now have a better understanding of the current attacks ingenuity used to propel in infecting more vulnerable systems in the wild.

6.2 Log4Shell artifacts and Log4Shell-IoT attack infrastructure

Furthermore, our analysis unveiled pivotal insights on the malicious LDAP servers used, payload URLs and binaries transmitted, along with unique intelligence pertained to the Log4Shell attack infrastructure employed to distribute IoT botnet malware binaries. Particularly, we recorded 8 distinct malicious LDAP servers embedded in the JNDI requests to transmit payload URLs and other malicious payloads (i.e., Shell code, Perl code, and binaries). Table 3 discloses 6/8 of the malicious LDAP servers that were used to

Table 2: Log4Shell obfuscation tactics such as nested strings applied on 'JNDI' and 'LDAP' words to mutate the original exploit and bypass existing detection techniques.

Original	Obfuscation tactics
jndi:	${env:NaN:-**j**}**ndi**${env:NaN:-:}
	${env:BARFOO:-**j**}**ndi**${env:BARFOO:-:}
	${upper:**j**}${lower:**n**}${lower:**d**}${upper:**i**}${date:'**:**'}
ldap:	${env:NaN:-**l**}**dap**${env:NaN:-:}
	${env:BARFOO:-**l**}**dap**${env:BARFOO:-:}
	${lower:**l**}${lower:**d**}${date:'**a**'}${test:by:d2lab:-**p**}${lower:**:**}

Table 3: Intelligence related to LDAP servers used in the Log4Shell exploit to distribute malicious payloads and URLs including IoT ELF binaries compiled for numerous CPU architectures. Note: defanged URLs are used to prevent inadvertently clicking malicious links.

#	Malicious LDAP Servers	Last seen (Date)	Payload URLs	Payload Type	Succ. Payload
1	ldap://13.78.223.142:1389	2022-01-22	hxxp://192.3.1.12/8UsA1.sh	Shell code	1
			hxxp://192.3.1.12/AB4g5/Josho.{m68k, ppc, x86, arm7, arm6, arm5, arm4, mpsl, mips}	ELF (Mirai)	8
			hxxp://51.161.64.198/install.sh	Shell code	1
			hxxp://51.161.64.198/httpd.{mpsl, arm6, arm4, sparc, x86, arm5, ppc, mips}	ELF (Tsunami)	8
2	ldap://66.71.248.138:1389	2022-01-20	hxxp://82.165.155.100/sal	Perl code (Shellbot)	1
3	ldap://165.22.33.143:1389	2022-01-21	hxxp://158.69.33.162/sshd	HTML code	1
4	ldap://141.98.10.141:2420	2022-01-21	hxxp://158.101.118.236/setup	ELF (Mirai)	1
5	ldap://150.136.111.68:1389	2022-01-24	hxxp://158.101.118.236/setup	ELF (Mirai)	1
6	ldap://5.181.80.103:1389	2022-01-29	hxxp://51.161.64.197/8UsA.sh	Shell code	1
			hxxp://107.174.24.16/AB4g5/Josho.{arm5, ppc, sh4, arm4, x86, mips, mpsl, arm6}	ELF (Mirai)	7

distribute at least one payload. We documented in total 30 payloads where 25 are ELF binaries. As a note, we use in Table 3 defanged URLs to prevent inadvertently clicking malicious links, additionally we emphasize on the *last seen* column in Table 3 that presents the last dates of the payload URLs.

To that extent, we note that the first and sixth malicious LDAP servers listed in Table 3 were primarily used to circulate malicious payload URLs associated with the distribution of IoT botnet malware binaries. For instance, the first malicious LDAP server was last seen on 2022-01-22 and was used twice to circulate distinct payload URLs. Initially, in both attempts (i.e., first and sixth), a ShellCode was installed followed by a series of 'Josho' and 'httpd' ELFs compiled for different CPU architectures including MIPS, ARM, MPSL, x86, m68k, PowerPC (PPC), and SPARC. The total number of successful 'Josho' and 'httpd' ELF payloads installed was 16 excluding 'Josho.arm4' which was not successfully installed. Similarly, the sixth malicious LDAP server was used to install only 'Josho' ELFs with several architectures, yet, distinct from the ones installed using the first LDAP server. This observation shows that there exist different LDAP servers hosting the same ELF names under the same directory name 'AB4g5', but with different contents which raise some doubts on the common ownership of these malicious LDAP servers. After further investigations using VirusTotal [53], Intezer Analyze [2], and Joe Sandbox [27] we can infer that the 'Josho' ELFs are variants of Mirai botnet malware, while the 'httpd' ELFs are attributed to Tsunami malware; both are botnet malware targeting IoT devices in the wild. Table 4 presents a sample of the collected ELF binaries while providing insights on the targeted CPU architectures, as well as the corresponding IoT botnet malware family. We can notice in Table 3 that the fourth and fifth LDAP servers are circulating the same URLs and distributing the same ELF which are also both related to Mirai botnet malware per our analysis.

Table 4: A small sample of the captured ELF binaries associated with IoT malware families (i.e., Mirai and Tsunami) compiled in distinct CPU architectures.

ELF Binaries (md5)	CPU arch.	Malware
3589e536d48793638924927ccf3188ff	ARM7	Mirai
0af3b329065b92b6099ea3e0d9b375d1	MIPS	Mirai
309dac88018d182c095a8b894a6cf272	MPSL	Mirai
4d0f5c6ffc911477212d4ace7b601dae	SPARC	Tsunami
325f326232680d70d21d122369014774	PPC	Tsunami
058e5c820388ef9e327bdc10590af9e0	x86	Tsunami

Interestingly, we discovered an intriguing infrastructure that we call 'Log4Shell-IoT attack infrastructure' exclusively used to distribute IoT botnet malware compiled for different CPU architectures. Accordingly, Figure 4 presents intelligence on the discovered attack infrastructure that we derived from the artifacts provided in

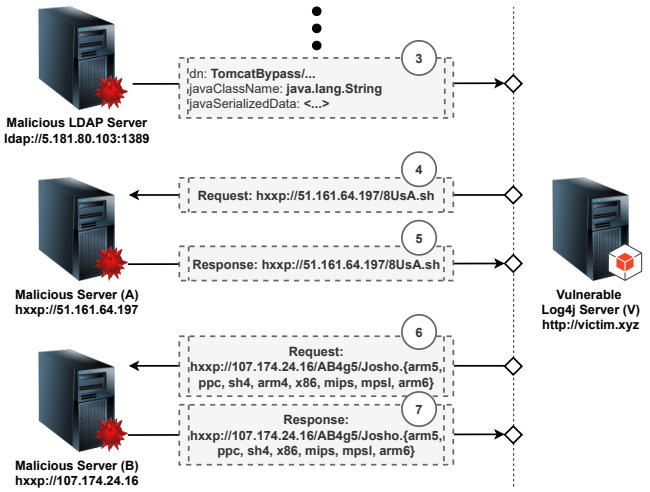

Figure 4: A discovered Log4Shell attack infrastructure that we call 'Log4Shell-IoT attack infrastructure' encompassing one LDAP server and two other servers maliciously employed to distribute ELF binaries related to IoT botnet malware.

Table 5: Number of deception packets sent to targeted ports and number of successful responses received for Mirai and non-Mirai inferred scanners.

Scanners	Target Ports	Sent (all)	Succ. Resp.	Ratio (%)
non-Mirai	23	96,865	20,809	21.48%
	8080	66,076	14,243	21.55%
	80	65,377	12,438	19.02%
	2323	17,317	3,349	19.33%
	Other	320,104	32,729	10.22%
Mirai	23	41,442	7,076	17.07%
	8080	9912	239	2.41%
	80	10,565	247	2.33%
	2323	21,298	4,284	20.11%
	Other	109,576	5,750	5.24%

the sixth row of Table 3. Figure 4 is a sequel to Figure 1 presented in Section 2 showcasing the second stage of the attack (i.e., *step 3* and beyond). Initially, at *step 3*, the vulnerable Log4j server (V) receives a serialized Java object, next at *steps 4* and *5*, the server (V) requests and receives a shellcode from server (A). Subsequently, the shellcode received from server (A) includes commands (i.e., wget and curl) to request from server (B) ELF binaries pertained to IoT malware and compiled to serve distinct CPU architectures, the last 2 aforementioned steps are conferred in Figure 4 at *steps 6* and *7* respectively.

7 POTENTIAL IMPROVEMENTS AND FUTURE DIRECTIONS

The primary focus of this paper is to propose a near real-time scheme to collect much-needed IoT malware artifacts to profile the current cyber security posture of the IoT ecosystem at scale, and hence derive central IoT malware intelligence. Nonetheless, we perceive potential improvements associated with our proposed scheme favorable to empower the interactions with a higher number of malware-infected IoT devices thus enabling the collection of further IoT malware artifacts. Table 5 presents information on the response success rate of the top four targeted ports for both Mirai and non-Mirai scanners. On average, the limited response rate from inferred Mirai scanners on ports 23 and 2323 is mainly due to the possibility of being deployed behind the NAT, not to mention the limitations of adopting one CPU architecture (i.e., ARM) which can affect the response rate. Additionally, the limited percentage on ports 80 and 8080 is expected since most IoT devices do not offer HTTP-related services.

As future work, we plan to incorporate additional IoT CPU architectures and IoT services, as well as consider possibly new IoT malware signatures to extend our deception technique. The latter step can be achieved by examining our own collected ELF binaries which position our approach as a self-sustained scheme to adapt to the different IoT botnet malware mutations and variations.

8 RELATED LITERATURE

Due to the immense proliferation of IoT devices coupled with the escalated malignant activities in the cyber space, much literature tackled the need for characterizing and examining the pervasive IoT botnet malware activities (e.g., botnet malware, scanning campaigns, DDoS attacks) at scale using a myriad of vantage points. Although we briefly review the literature in this section, yet, we would reiterate the unique capabilities enabled by our proposed near real-time scheme in collecting central IoT malware artifacts. This scheme empowered by a /8 network telescope offer a hefty look into the current IoT botnet malware activities currently occurring in the wild.

Initially, the interest in characterizing botnet dates back to 2005 when Cooke et al. [14] demonstrated the use of honeypot in tracking and examining Internet-scale botnets. Later in 2015, Pa et al. [44] presented the very first virtualized sandbox environment to collect artifacts related to malware-infected IoT devices. Subsequently, a plethora of contributions was dedicated to understand the cyber security posture of the IoT ecosystem at scale using a multitude of data point collection. For instance, Antonakakis et al. [4] are the first to examine the Mirai botnet source code, along with its malicious activities by performing empirical measurements using different data sources including network telescope, active scanning, Telnet honeypots, and DNS data (i.e., passive and active). Similarly, Herwig et al. [26] leveraged active scanning along with a longitudinal collection of root DNS backscatter traffic to analyze the Hajime botnet. Subsequently, Griffioen et al. [25] reported evidence on a prevailing battle among distinct IoT malware botnet using numerous honeypot deployments. Additionally, Dang et al. [18] examined file-less attacks on Linux-based IoT devices using 4 hardware IoT honeypots and 108 specifically designed software honeypots. Despite the use of multiple vantage points, the IoT security field perceives a shortage in the provision of large scale IoT malware intelligence necessary for identifying security threats in the IoT paradigm and thus elevate its overall security posture. We believe this contribution is a setting stone towards a more sound and reliable schemes dedicated for the continuous collection and analysis of IoT artifacts much-needed to address the many security gaps in the IoT paradigm.

9 CONCLUSION

In this paper, we tackled the many concerns related to the IoT security field by introducing a near real-time IoT malware artifacts collection scheme to understand the on-going IoT malware threats, the origin of the IoT malware attacks, along with the deficits of the IoT paradigm security. We enabled the collection of valuable IoT malware artifacts by employing a /8 network telescope comprising of 16.7 million IP addresses coupled with a deception technique to interconnect with malware-infected IoT devices in the wild. During our empirical measurement, we recorded *80,569,070* interactions and detected *30,190* malware-infected IoT devices. Accordingly, we derived pivotal IoT malware intelligence which include system commands, file-less attacks artifacts, payload URLs, ELF binaries, log-in credentials, malicious LDAP servers, and using our proposed scheme we were able to investigate and construct evidence associated with the abuse of the Log4Shell security vulnerability in distributing IoT malware binaries. Finally, we released to the security community a raw sample of our collected IoT malware artifacts on GitHub and Google Drive [42].

REFERENCES

[1] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Kevin Snow, Fabian Monrose, Manos Antonakakis, et al. 2021. The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle. In *30th USENIX Security Symposium (USENIX Security 21)*.

[2] analyze.intezer.com. 2022. Intezer Analyze. Retrieved 2022-01-28 from https://analyze.intezer.com

[3] Anna-senpai. 2016. Mirai Source code release. Retrieved 2022-01-28 from https://hackforums.net/showthread.php?tid=5420472

[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*. 1093–1110.

[5] Michael Bailey, Evan Cooke, Farnam Jahanian, Andrew Myrick, and Sushant Sinha. 2006. Practical darknet measurement. In *2006 40th Annual Conference on Information Sciences and Systems*. IEEE, 1496–1501.

[6] Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, David Watson, et al. 2005. The internet motion sensor-a distributed blackhole monitoring system.. In *NDSS*. Citeseer.

[7] Michael Bailey, Evan Cooke, Farnam Jahanian, and David Watson. 2005. The blaster worm: Then and now. *IEEE Security & privacy* 3, 4 (2005), 26–31.

[8] blog.netlab.360.com. 2021. Ten families of malicious samples are spreading using the Log4j2 vulnerability Now. Retrieved 2022-01-28 from https://blog.netlab.360.com/ten-families-of-malicious-samples-are-spreading-using-the-log4j2-vulnerability-now/

[9] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. 2013. Cyber scanning: a comprehensive survey. *Ieee communications surveys & tutorials* 16, 3 (2013), 1496–1519.

[10] Olivier Cabana, Amr M Youssef, Mourad Debbabi, Bernard Lebel, Marthe Kassouf, Ribal Atallah, and Basile L Agba. 2021. Threat Intelligence Generation Using Network Telescope Data for Industrial Control Systems. *IEEE Transactions on Information Forensics and Security* 16 (2021), 3355–3370.

[11] caida.org. 2022. CAIDA Network Telescope Data. Retrieved 2022-01-28 from https://www.caida.org/projects/network_telescope/

[12] caida.org. 2022. Network Telescopes: Technical Report. Retrieved 2022-01-28 from https://www.caida.org/catalog/papers/2004_tr_2004_04/tr-2004-04.pdf

[13] Hyunji Chung, Jungheum Park, and Sangjin Lee. 2017. Digital forensic approaches for Amazon Alexa ecosystem. *Digital investigation* 22 (2017), S15–S25.

[14] Evan Cooke, Farnam Jahanian, and Danny McPherson. 2005. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. *SRUTI* 5 (2005), 6–6.

[15] CowrieHoneypot. 2022. Cowrie Honeypot. Retrieved 2022-01-28 from https://github.com/cowrie/cowrie

[16] cve.mitre.org. 2021. CVE-2021-44228. Retrieved 2022-01-28 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228

[17] Alberto Dainotti, Alistair King, Kimberly Claffy, Ferdinando Papale, and Antonio Pescapé. 2014. Analysis of a "/0" stealth scan from a botnet. *IEEE/ACM Transactions on Networking* 23, 2 (2014), 341–354.

[18] 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*. 482–493.

[19] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J Alex Halderman. 2015. A search engine backed by Internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 542–553.

[20] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2013. ZMap: Fast Internet-wide scanning and its security applications. In *22nd USENIX Security Symposium (USENIX Security 13)*. 605–620.

[21] Nick G. 2022. How Many IoT Devices Are There in 2022? [All You Need To Know]. Retrieved 2022-01-28 from https://techjury.net/blog/how-many-iot-devices-are-there/#gref

[22] Vincent Ghiëtte, Norbert Blenn, and Christian Doerr. 2016. Remote identification of port scan toolchains. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 1–5.

[23] Jonathan Greig. 2022. Log4j: Mirai botnet found targeting ZyXEL networking devices. Retrieved 2022-01-28 from https://www.zdnet.com/article/log4j-mirai-ddos-botnet-targeting-zyxel-networking-devices/

[24] Harm Griffioen and Christian Doerr. 2020. Discovering Collaboration: Unveiling Slow, Distributed Scanners based on Common Header Field Patterns. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9.

[25] Harm Griffioen and Christian Doerr. 2020. Examining Mirai's Battle over the Internet of Things. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 743–756.

[26] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. 2019. Measurement and analysis of Hajime, a peer-to-peer IoT botnet. In *Network and Distributed Systems Security (NDSS) Symposium*.

[27] joesandbox.com. 2022. Joe Sandbox Cloud. Retrieved 2022-01-28 from https://www.joesandbox.com/#linux

[28] Jaeyeon Jung, Vern Paxson, Arthur W Berger, and Hari Balakrishnan. 2004. Fast portscan detection using sequential hypothesis testing. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*. IEEE, 211–225.

[29] Georgios Kambourakis, Constantinos Kolias, and Angelos Stavrou. 2017. The mirai botnet and the iot zombie armies. In *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*. IEEE, 267–272.

[30] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. 2019. All things considered: an analysis of IoT devices on home networks. In *28th USENIX Security Symposium (USENIX Security 19)*. 1169–1185.

[31] Derek Leonard and Dmitri Loguinov. 2010. Demystifying service discovery: implementing an internet-wide scanner. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 109–122.

[32] Shancang Li, Kim-Kwang Raymond Choo, Qindong Sun, William J Buchanan, and Jiuxin Cao. 2019. IoT forensics: Amazon echo as a use case. *IEEE Internet of Things Journal* 6, 4 (2019), 6487–6497.

[33] libvirt.org. 2022. libvirt. Retrieved 2022-01-28 from https://libvirt.org/

[34] logging.apache.org. 2022. *Lookups*. https://logging.apache.org/log4j/2.x/manual/lookups.html.

[35] logging.apache.org. 2022. *Welcome to Log4j 2!* https://logging.apache.org/log4j/2.x/manual/index.html.

[36] Tongbo Luo, Zhaoyan Xu, Xing Jin, Yanhui Jia, and Xin Ouyang. 2017. Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices. *Black Hat* (2017), 1–11.

[37] Gordon Fyodor Lyon. 2008. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure. Com LLC (US).

[38] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. 2003. Inside the slammer worm. *IEEE Security & Privacy* 1, 4 (2003), 33–39.

[39] David Moore, Colleen Shannon, Douglas J Brown, Geoffrey M Voelker, and Stefan Savage. 2006. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems (TOCS)* 24, 2 (2006), 115–139.

[40] David Moore, Colleen Shannon, and K Claffy. 2002. Code-Red: a case study on the spread and victims of an Internet worm. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. 273–284.

[41] ncsc.gov.uk. 2021. Log4j vulnerability - what everyone needs to know. Retrieved 2022-01-28 from https://www.ncsc.gov.uk/information/log4j-vulnerability-what-everyone-needs-to-know

[42] Authors of ARES 22 Current Submission. 2022. IoT Forensic Artifacts 2022. Retrieved 2022-01-28 from https://github.com/IoT-Forensics/IoT-Forensic-Artifacts-2022

[43] openwrt.org. 2022. Linux Distribution for Embedded Devices. Retrieved 2022-01-28 from https://openwrt.org/

[44] 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)*.

[45] Thomas Patzke. 2022. Log4Pot. Retrieved 2022-01-28 from https://github.com/thomaspatzke/Log4Pot

[46] Morteza Safaei Pour, Antonio Mangino, Kurt Friday, Matthias Rathbun, Elias Bou-Harb, Farkhund Iqbal, Sagar Samtani, Jorge Crichigno, and Nasir Ghani. 2020. On data-driven curation, learning, and analysis for inferring evolving internet-of-Things (IoT) botnets in the wild. *Computers & Security* 91 (2020), 101707.

[47] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse.. In *NDSS*.

[48] Asanka Sayakkara, Nhien-An Le-Khac, and Mark Scanlon. 2019. Leveraging electromagnetic side-channel analysis for the investigation of IoT devices. *Digital Investigation* 29 (2019), S94–S103.

[49] scapy.readthedocs.io. 2022. Scapy. Retrieved 2022-01-28 from https://scapy.readthedocs.io/en/latest/

[50] statista.com. 2020. Internet of Things (IoT) and non-IoT active device connections worldwide from 2010 to 2025. Retrieved 2022-01-28 from https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

[51] Amit Tambe, Yan Lin Aung, Ragav Sridharan, Martín Ochoa, Nils Ole Tippenhauer, Asaf Shabtai, and Yuval Elovici. 2019. Detection of threats to IoT devices using scalable VPN-forwarded honeypots. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*. 85–96.

[52] Alexander Vetterl and Richard Clayton. 2019. Honware: A virtual honeypot framework for capturing CPE and IoT zero days. In *2019 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 1–13.

[53] virustotal.com. 2022. Virus Total. Retrieved 2022-01-28 from https://www.virustotal.com/gui/home/upload

[54] wiki.qemu.org. 2022. Qemu. Retrieved 2022-01-28 from https://wiki.qemu.org/Main_Page

[55] Eric Wustrow, Manish Karir, Michael Bailey, Farnam Jahanian, and Geoff Huston. 2010. Internet background radiation revisited. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 62–74.
[56] zdnet.com. 2022. Log4j Flaw. Retrieved 2022-01-28 from https://www.zdnet.com/article/the-log4j-flaw-hasnt-led-to-massive-hacking-attacks-but-that-doesnt-mean-the-threat-is-over/
[57] Xiaolu Zhang, Kim-Kwang Raymond Choo, and Nicole Lang Beebe. 2019. How do I share my IoT forensic experience with the broader community? An automated knowledge sharing IoT forensic platform. *IEEE Internet of Things Journal* 6, 4 (2019), 6850–6861.
[58] Xiaolu Zhang, Oren Upton, Nicole Lang Beebe, and Kim-Kwang Raymond Choo. 2020. IoT botnet forensics: A comprehensive digital forensic case study on Mirai botnet servers. *Forensic Science International: Digital Investigation* 32 (2020), 300926.