On Data-driven Curation, Learning, and Analysis for Inferring Evolving Internet-of-Things (IoT) Botnets in the Wild

Morteza Safaei Pour[a,*], Antonio Mangino[a], Kurt Friday[a], Matthias Rathbun[a], Elias Bou-Harb[a,*], Farkhund Iqbal[b], Sagar Samtani[c], Jorge Crichigno[d], Nasir Ghani[e]

[a]*Cyber Threat Intelligence Laboratory, College of Engineering & Computer Science, Florida Atlantic University, Florida, USA*
[b]*College of Technological Innovation, Zayed University, Abu Dhabi, UAE*
[c]*Department of Information Systems and Decision Sciences, University of South Florida, Florida, USA*
[d]*Integrated Information Technology, University of South Carolina, Columbia, USA*
[e]*Department of Electrical Engineering and Cyber Florida, University of South Florida, Florida, USA*

Abstract

The insecurity of the Internet-of-Things (IoT) paradigm continues to wreak havoc in consumer and critical infrastructures. The highly heterogeneous nature of IoT devices and their widespread deployments have given rise to several key security and measurement-based challenges, including the lack of IoT-centric data that can be collected, analyzed and correlated. To this end, this paper explores macroscopic, passive empirical data to shed light on this evolving threat phenomena. This not only aims at classifying and inferring Internet-scale compromised IoT devices by solely observing such one-way network traffic~~,~~ but also endeavors to uncover, report and thoroughly analyze orchestrated "in the wild" IoT botnets. Initially, to prepare the effective utilization of such data, a novel probabilistic model is designed and developed to cleanse such traffic from noise samples (i.e., misconfiguration traffic). Subsequently, several shallow and deep learning models are evaluated to ultimately design and develop a multi-window convolution neural network trained on active and passive measurements to accurately identify compromised IoT devices. Consequently, to infer orchestrated and unsolicited activities that have been generated by well-coordinated IoT botnets, hierarchical agglomerative clustering is employed by scrutinizing a set of innovative and efficient network feature sets. By analyzing 3.6 TB of recent darknet traffic, the proposed approach uncovered a momentous 440,000 compromised IoT devices and generated evidence-based artifacts related to 350 IoT botnets. Moreover, by conducting thorough analysis of such inferred campaigns, we reveal their scanning behaviors, packet inter-arrival times, employed rates and geo-distributions. Although several campaigns exhibit significant differences in these aspects, some are more distinguishable; by being limited to specific geo-locations or by executing scans on random ports besides their core targets. Further, while some of these detected botnets refer to previously documented campaigns such as `Hide and Seek`, `Hajime` and `Fbot`, other events illustrate evolving threats including those specifically targeting industrial control and corresponding communication services, as well as campaigns which demonstrate cryptojacking capabilities. To motivate empirical (and operational) IoT cyber security initiatives as well as aid in reproducibility of the obtained results, we make the source codes of all the developed methods and techniques available to the research community at large.

Keywords: Data science, Cyber forensics, Internet-of-Things, IoT security, Internet measurements

1. Introduction

With the escalating adoption of the Internet-of-Things (IoT) paradigm in critical infrastructure, smart homes, transportation, and various other realms, an increasing number of devices are becoming directly Internet-facing. Although IoT devices can be deployed behind Network Address Translation (NAT) gateways, a plethora of such devices are directly connected to the Internet and/or employ port-forwarding for proper and simplified provisioning and management [1]. Unfortunately, such devices continue to lack basic security protocols and measures, rendering them easy targets for exploitations and hence recruitment within coordinated IoT botnets [2]. ~~Furthermore,~~ there exist~~s~~ several IoT inherent factors such as their heterogeneous nature and limited processing resources, further complicating the addressing of their security requirements. ~~Additionally,~~ subpar attention is being paid to IoT security aspects by their manufacturers and users, ~~on top of~~ an overwhelming lack of maturity of IoT-specific update procedures for patch management [2].

~~Indeed,~~ IoT security has been ~~a particular~~ area of focus since `Mirai` [3] infected more than 200,000 devices to conduct ~~debilitating~~ Distributed Denial of Service (DDoS) attacks in late 2016. ~~The Mirai~~ attacks demonstrated the sheer capabilities ~~for maliciousness~~ by way of ~~instrumenting~~ exploited IoT devices. ~~Thereafter,~~ botnets consisting of IoT devices have consistently been evolving, incorporating new devices and services. ~~Moreover,~~ the IoT botnet environment has expanded to include several more players who ultimately compete for control over insecure IoT devices by means of newly-exposed vulnerabilities. ~~Very~~ recently, the `Echobot` [4, 5] campaign ~~has been~~ identified operating "in the wild"; derived from the Mirai's source code, Echobot has compromised millions of IoT nodes from more than 10 diverse vendors through exploiting more than 20 unique (software and firmware) IoT-centric vulnerabilities. ~~Indeed,~~ this IoT threat phenomena will ~~undoubtedly~~ continue to render a ~~very~~ dynamic behavior which makes inferring, attributing, and assessing compromised IoT devices and their coordinated illicit activities significantly challenging.

Despite efforts to mitigate the ever-increasing IoT security issues, challenges exist due to the heterogeneity of ~~the~~ IoT devices and the availability of anti-honeypot techniques [6]. ~~Moreover,~~ acquiring IoT-centric empirical data ~~to be curated and analyzed for~~ maliciousness is problematic~~,~~ given the large-scale deployments of such devices in Internet-wide realms. While network telescope (darknet) traffic [7] (i.e., Internet-scale traffic targeting routable yet unused IP addresses) has recurrently proven to be a reliable and an effective source for generating insights related to Internet-wide maliciousness [7], its exploration for addressing IoT security issues is still at its infancy. ~~Broadly,~~ a major challenge related to the inference of IoT maliciousness through the analysis of network telescope traffic is the lack of sound data-driven artifacts which can be analyzed to confirm that the perceived one-way traffic is ~~in fact~~ originating from IoT devices ~~and~~ not from typical machines. In addition, the devised darknet-driven methodologies should be able to accommodate the evolving nature of IoT botnets by ~~taking into consideration~~ their empirical specificities~~,~~ as perceived by the (somehow limited) vantage point of the network telescope.

[*]Corresponding authors. Tel.: +1 561 931 7531
Email addresses: `msafaeipour2017@fau.edu` (Morteza Safaei Pour), `ebouharb@fau.edu` (Elias Bou-Harb)

A ~~specific~~ successful use-case for leveraging darknet data for IoT security has been demonstrated by correlating darknet-inferred probing IP addresses with databases such as `Shodan` [8] or `Censys` [9] to infer Internet-scale exploitations [10, 11]. Both `Shodan` and `Censys` use IP crawlers, active scanning, and banner grabbing to collect and index open ports and available services on billions of Internet-facing IoT devices. While this strategy would provide large-scale valuable information, the set of those identified IoT devices is incomplete due to the scope of the services being limited to only the devices reachable by their scanners. ~~Indeed,~~ such generated probes towards various services and ports are typically filtered by firewalls, while upon infection, IoT malware tend to also block ports and disable common outward facing services (i.e., Telnet, CWMP, ADB, etc.) [3, 12] or modify banner information. When these events occur, the indexing of the IoT devices is significantly impeded.

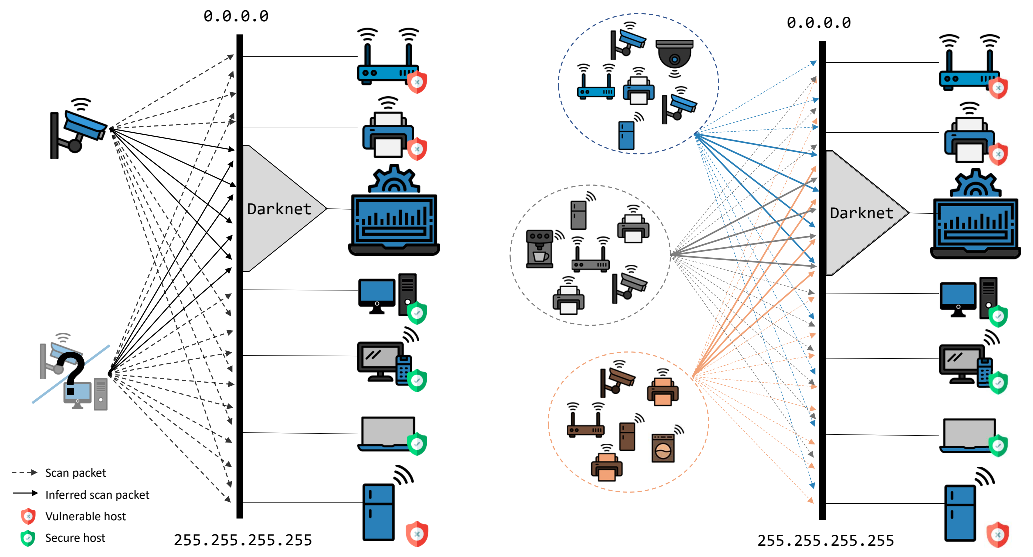
~~Having noted~~ this, it is worthy to note a number of other technical challenges which further hinders leveraging darknet data for inferring IoT maliciousness. ~~Indeed,~~ perceived packets on the network telescope (that have been generated by IoT bots) solely resemble scan activities (i.e., do no include payload information and are unidirectional), which limits the amount of data to analyze and work with. Additionally, only a small portion of IoT-generated unsolicited traffic actually hit the network telescope, which renders time-~~based~~ analysis quite difficult, and ~~certainly~~ complicates extracting effective and robust features to infer orchestration behaviors of compromised IoT devices.

Motivated by the aforementioned limitations coupled with the lack of thorough measurement-based studies which shed light on the insecurity of the IoT paradigm at large, the paper contributes by proposing a multi-threaded, generic methodology by scrutinizing macroscopic darknet data to design, develop and evaluate:

- A novel darknet-specific, formal sanitization model that systematically identifies and filters misconfiguration traffic to permit the storage and processing of network telescope data. The proposed darknet sanitization model does not rely on arbitrary cut-off thresholds, provides likelihood models to distinguish between misconfiguration and other forms of darknet traffic, and is independent from the nature of the traffic sources. The proposed model neatly captures the natural behavior of darknet-targeted misconfiguration traffic.

- An IoT-centric fingerprinting approach rooted in deep learning and active measurement~~s~~ methodologies to infer Internet-scale compromised IoT devices by exclusively operating on network telescope data. The addressed problem ~~herein~~ is illustrated in Figure 1a. Using more than 3 TB of recent darknet data, the outcome of such a devise approach exposes more than 400,000 compromised IoT devices from very well-known vendors. The results also highlight that more than 75% of all the inferred IoT bots do not match the typical `Mirai` signature [3], concurring the evolving nature of this threat phenomena and highlighting the added-value of the proposed methodology.

- An IoT-specific botnet inference methodology based upon effective and lightweight (darknet) data-driven features and hierarchical agglomerative clustering. The addressed problem herein is shown in Figure 1b. The results from instrumenting such an approach uncover more than 300 “in the wild” IoT botnets, where close to 25 of those contain over 1,000 exploited, well-coordinated IoT bots. ~~Moreover,~~ IoT botnet-specific traits are investigated, including scanning modules, probing rates and their geo-distributions. While the results



(a) (b)

Figure 1: Leveraging network telescopes to (a) devise learning techniques for fingerprinting IoT devices; and (b) develop clustering methods for identifying campaigns of orchestrated IoT devices

shed light on previously documented IoT botnets that were found to be still active, the outcome also uncovers new IoT botnets such as those possessing cryptojacking capabilities (which were shown to be coordinated by the same "player" due to the usage of the same key) and those ~~that were~~ inferred to be targeting industrial control systems. To facilitate the reproducibility of the results in addition to motivate passive Internet measurements for IoT security, we make all the developed methods and techniques available to the research and operational communities at large via `https://github.com/COYD-IoT/COYD-IoT`.

The remainder of this paper is organized as follows. ~~The next section~~ reviews the literature related to network telescope research, IoT device fingerprinting and IoT botnet analysis to demonstrate the state-of-the-art contributions of this work. In Section 3, we detail the darknet preprocessing model, the studied machine/deep learning models for fingerprinting compromised IoT devices, in addition to elaborating on the IoT-centric botnet inference methodology. In Section 4, we report and discuss the results derived from executing the proposed approach. ~~Finally,~~ Section 5 summarizes the contributions of this paper and paves the way for future work by addressing a number of limitations.

2. Related Work

In this section, we review three topics central to the contemporary IoT security landscape. The first focuses on network telescopes as a powerful mechanism to capture IoT-specific, illicit network traffic. The second summarizes efforts pertaining to IoT device fingerprinting and the detection of compromised IoT devices. Finally, we enumerate the literature related to

IoT-specific botnet analysis.

Network telescopes and IoT Security. A network telescope (i.e., darknet), is a set of deployed, routable, allocated, yet unused IP addresses deployed in order to passively observe incoming Internet-scale traffic [7]. Since these IP addresses are not associated with any services, traffic targeting them is unsolicited [13] originating from infected devices, victims of Denial of Service (DoS) attacks, or misconfiguration. Network telescopes are reliable sources for investigating large-scale, Internet-wide activities ~~and~~ recent examples of their successful applications include the study of probing activities [14] and DDoS attacks [15, 16]. In the context of assessing the maliciousness of IoT devices through network telescopes, Torabi et al. [10] recently conducted large-scale correlations between passive measurements and IoT-relevant information to investigate and disclose malicious activities associated with more than 26,000 IoT devices, including those within critical infrastructure. Similarly, Shaikh et al. [11] examined nearly 14,000 compromised IoT devices and extracted malicious signatures for further deployment in IoT hosting environments for mitigation purposes. ~~Moreover,~~ by means of applying filters to network telescope data in order to discern Mirai-relevant traffic, Antonakakis et al. [3] were able to gather IoT-related information pertaining to roughly 1.2 million Mirai-infected IP addresses during 7 months, in addition to examining their associated detection-avoidance techniques. Cetin et al. [17] conducted empirical studies focusing on IoT malware cleanup efforts and remediation rates in a medium-sized Internet Service Provider (ISP) leveraging darknet and honeypot sources.
While such contributions are noteworthy, several shortcomings exist. First, such works rely on a specific IoT malware signature (i.e., tcpSeq == dstIP); since IoT bots do not all follow this signature, this prevents the execution of comprehensive identification. Our measurements indeed have revealed that less than 25% of all the inferred IoT bots match this specific signature. Second, the majority of these related works solely depend on databases gathered by Internet scanning services (e.g., Shodan), which miss a large portion of the actual IoT bot population. In contrast, we propose ~~herein~~ an approach consisting of active and passive measurements, coupled with machine/deep learning techniques, which leads to a more comprehensive view of the IoT botnets' populations.

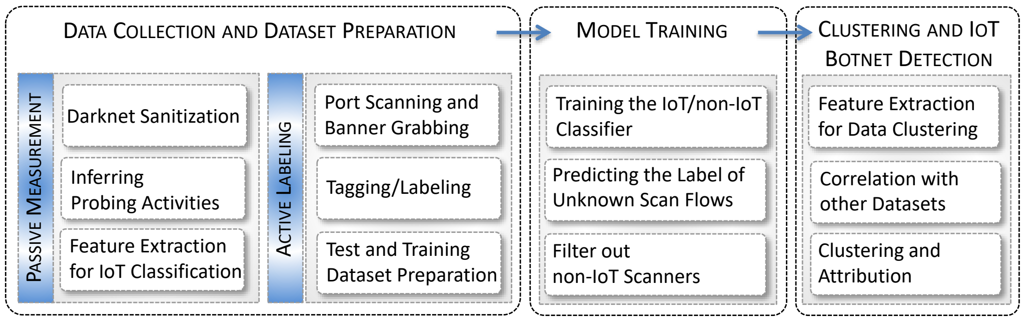
IoT device fingerprinting. Most IoT inference methods in previous works rely on text information in banners, gathered by active measurement or provided by services similar to `Shodan` [8], `Cencys` [18] and ZoomEye [19]. For example, Kumar et al. [20] leveraged predefined text rules from such services to fingerprint the devices through designing ensemble of four supervised classifiers on UPnP and DNS responses, HTTP data banners, and network-layer information. Several research efforts have alternatively elected to attempt IoT device fingerprinting by solely observing network traffic. For instance, Guo et at. [21] postulated that since IoT devices regularly exchange data with servers ran by their manufacturers, IoT device type and vendor can be fingerprinted by observing exchanged flow-level network traffic between devices and the servers. Meidan et al. [22] manually labeled network traffic generated by IoT devices and employed a supervised learning algorithm to classify IoT devices for a given organization's network. Moreover, Miettinen et al. [23] leveraged network traffic generated by IoT devices during their setup process for capturing device-specific traits, and subsequently mapped these signatures to the device type by way of random forest classification. Improving upon anomaly detection premised on device types, Nguyen et al. [24] ~~have~~ recently employed a machine learning algorithm which not only discriminates between the corresponding classes of devices,

but exhibited remarkable accuracy while doing so. In a similar yet refined manner, Thangavelu et al. [25] also developed a machine learning-based distributed device fingerprinting technique from an ISP's perspective to detect the presence of common devices, including unknown new devices. Pinheiro et al. [26] incorporated only the ~~mean, standard deviation of the packet length,~~ and the number of bytes transmitted by each device in one-second windows of encrypted traffic to distinguish between IoT and non-IoT devices. Siby et al. [27] detected devices in a local network by passively intercepting wireless signals and identifying IoT devices by using the encapsulated MAC addresses within investigated flows. In an alternative approach, Acar et al. [28] developed a web script that can identify the presence of IoT devices which running local HTTP servers, while alarming about the feasibility of accessing such IoT devices using DNS rebinding.
A shortcoming of the aforementioned literature ~~works~~ is that their scope is limited to local IoT networks and thus do not present an Internet-wide perspective; hence their proposed techniques are not applicable on one-way scan flows arriving at network telescopes. In contrast, we leverage a set of rules to fingerprint hosts that respond with banners, in addition to devising learning techniques to identify unreachable infected hosts to predict their types using innovative features extracted from sequences of TCP SYN packets arriving at the network telescope.

IoT botnet analysis. Within the context of botnet analysis through tailored honeypots, Pa et al. [29] inferred several malware families by constructing a honeypot to analyze attacks against Telnet services. Furthermore, Guarnizo et al. [30] designed the IoT-centric Scalable high-Interaction Honeypot (SIPHON) which showed an ability to attract a tremendous amount of malicious IoT botnet-generated traffic through a combination of worldwide wormholes and a small number of IoT devices. Moreover, Metongnon and Sadre [31] have recently reported on a large number of exploited IoT protocols, based on the analysis of network traffic from IoT-centric honeypots and network telescopes. Given the copious amounts of IoT hardware in the wild and their accompanying heterogeneity, we have to note that honeypot-based methodologies frequently fail when it comes to mimicking all vulnerabilities for the vast assortment of IoT products and respective firmware which is essential to attributing IoT botnets. Additionally, the vantage points of honeypots are typically quite small, hindering their effectiveness in tracking Internet-scale IoT botnets as well as accurately estimating their population size. From another perspective, Herwig et al. [12] have recently provided a comprehensive investigation related to the `Hajime` IoT botnet using active scanning of Hajime's peer to peer infrastructure and by leveraging a longitudinal collection of root DNS backscatter traffic. Unfortunately, their approach of such a target-specific study is designed to investigate a singular IoT botnet taking advantage of the known botnet infrastructure. However, this cannot be replicated or generalized to study other IoT botnets. In contrast, the proposed work in this paper compliments previous contributions by devising and evaluating a generic, macroscopic approach to infer ongoing IoT botnets based on the orchestration artifacts in their scanning modules.

3. Proposed methodology

This section details the proposed approach as depicted in Figure 2. Its core components include *(i)* data collection and dataset preparation, which introduces the darknet sanitization probabilistic model to filter out misconfiguration traffic along with the inference of Internet-scale probing activities and labeling their sources; *(ii)* the utilization of a deep learning binary classifier for fingerprinting compromised IoT devices; and *(iii)* the feature engineering process coupled with executing hierarchical agglomerative clustering to infer and report on IoT botnets. These steps are subsequently detailed.



Data Collection and Dataset Preparation
Passive Measurement
Darknet Sanitization
Inferring Probing Activities
Feature Extraction for IoT Classification
Active Labeling
Port Scanning and Banner Grabbing
Tagging/Labeling
Test and Training Dataset Preparation
Model Training
Training the IoT/non-IoT Classifier
Predicting the Label of Unknown Scan Flows
Filter out non-IoT Scanners
Clustering and IoT Botnet Detection
Feature Extraction for Data Clustering
Correlation with other Datasets
Clustering and Attribution

Figure 2: The components of the proposed approach

3.1. Network telescope sanitization model

Network telescopes, most commonly known as darknets [7], constitute a set of routable, allocated yet unused IP addresses. Since these addresses do not operate legitimate services, any traffic targeting them is unsolicited. From a deployment perspective, network telescopes are commonly distributed on specific Internet IP subspaces operated by Internet Service Providers (ISPs), educational entities and corporate backbone networks. Darknet IP addresses are, by nature, indistinguishable from other routable addresses, rendering them an effective technique to amalgamate Internet-wide, one-way unsolicited network traffic.

Although network telescope (darknet) data predominantly consists of malicious packets originating from probes, backscattered packets from victims of DDoS attacks, and malware propagation attempts, it also contains misconfiguration traffic. The latter non-malicious packets frequently result from network, routing, hardware, or software faults that were erroneously directed towards a darknet. Such traffic might also be an artifact of improper configurations during darknet deployment. Misconfiguration traffic impedes the proper functioning of cyber threat intelligence algorithms operating on darknet data, which often yields numerous undesirable false positives and false negatives. Additionally, its excessive existence is a sheer waste of valuable storage resources. As a result, given the lack of formalism in addressing this problem, the objective herein is to elaborate on a probabilistic model that is specifically tailored towards the preprocessing of darknet data by way of fingerprinting, and in turn, filtering out embedded misconfiguration traffic.

In a nutshell, the model formulates and computes two metrics, with the aim of capturing the behavioral perspective of misconfiguration flows as they target the darknet space. Regarding the natural tendencies associated with typical network flows, the model initially estimates the rareness of access of the destination. Secondly, to ensure the inclusion of the unique characteristics of the given flow as well, the scope of access is considered, which accounts for the number of distinct darknet IP addresses that a specific remote source has accessed. Subsequently, the joint probability is formulated, computed, and compared. If the probability of the source generating a misconfiguration flow is higher than that of the source being malicious (or unsolicited), then that particular source is deemed to be generating misconfiguration traffic, flagged, and the corresponding flows are filtered out. In the following, we detail the notions of both rareness and scope of access.

Let $D = \{d_1, d_2, d_3, \cdots\}$ represent the set of darknet IP addresses, with D_i being a subset of those accessed by source s_i. First, the model captures how unusual these accessed destinations are. The underlying idea in doing so stems from the fact that misconfigured sources target destinations seldom called upon by others [32]. Thus, the model estimates the distribution of a darknet IP d_i as being accessed by such a source as

$$P_{misc}(d_i) = \frac{n_s(d_i)}{\sum_{\forall d_j \in D} n_s(d_j)}, \tag{1}$$

where $n_s(d_i)$ is the number of sources that have accessed d_i; in contrast, a malicious darknet source will target a given destination at random. Typically, defining a suitable probability distribution to exemplify the randomness of a malicious source taking aim at a specific darknet destination is quite tedious; therefore, a simplistic assumption is often applied to resolve this potential headache. In this context, Durumeric et al. [33] demonstrated that sources probe their darknet targets following a Gaussian distribution. By adopting that assumption, one can model the probability of a darknet destination being accessed by a malicious source as $P_{mal}(d_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$ where σ is the standard deviation, μ is the mean, σ^2 is the variance, and x is the location of the darknet destination following the aforementioned distribution. Recall that not only does the model capture how unusual the accessed destinations are, but it also considers the number of darknet destinations accessed by a particular source, which we subsequently describe. Given a set of D_i darknet destinations accessed by a specific source s_i, the model ultimately measures two probability distributions, namely, $P_{misc}(D_i)$ and $P_{mal}(D_i)$; the former being the probability that D_i has been generated by a misconfigured source and the latter originating from that with a malicious intent towards darknet D_i. For example, if the darknet addresses accessed by s_1 are $D_1 = \{d_{i1}, d_{i2}, d_{i3}\}$, $P(D_1)$ equates to the probability of s_1 accessing the specific combination of addresses $\{d_{i1}, d_{i2}, d_{i3}\}$ given three targeted destinations, multiplied by the probability of s_1 accessing any three destinations. In turn, we can generalize $P(D_1)$ as

$$P(D_i) = P(D_i = \{d_{i1}, d_{i2}, \cdots, d_{in}\} \mid |D_i| = n) \times P(|D_i| = n). \tag{2}$$

For both a misconfigured and malicious source, the first term of equation (2) can be modeled as

$$P(D_i = \{d_{i1}, d_{i2}, \cdots\} \mid |D_i|) = \frac{1}{K} \prod_{\forall d_j \in D_i} P(d_i) \tag{3}$$

where K, acting as a normalization constant and solely being used as a means of summing the probabilities to 1, could be defined as $K = \frac{|D|!}{n!(|D|-n)!} \times \frac{1}{|D|^n}$. K is a standard normalization constant often employed in Bayesian probability [34]. Moreover, n encompasses all sources in the data, whereas $|D|$ represents the darknet IP space. Consequently, the likelihood that a source will target a certain number of darknet destinations (i.e., the second term of equation (2)) depends upon whether it is malicious or misconfigured. Characteristically, misconfigured sources access one or few destinations while those with malicious intent target a larger pool. Accordingly, we model such distributions as

$$P_{misc}(|D_i|) = \frac{1}{(e-1)|D_i|!} \tag{4}$$

$$P_{mal}(|D_i|) = \frac{1}{|D|}, \tag{5}$$

where the term $(e-1)$ in equation (4) ensures the distribution's summation equals 1. Equation (4) guarantees a significant decrease in the probability as the number of targeted destinations increases. In contrast, equation (5) captures that of a random number of darknet addresses being accessed by a malicious source. Thereby, via plugging in of equations (4) and (5) into (3), respectively, we can represent the probability of a source being either misconfigured or malicious, given a set of darknet destination addresses, as

$$P_{misc}(D_i) = \frac{1}{K(e-1)|D_i|!} \prod_{\forall d_j \in D_i} P_{misc}(d_i) \tag{6}$$

$$P_{mal}(D_i) = \frac{1}{K|D|} \prod_{\forall d_j \in D_i} P_{mal}(d_i). \tag{7}$$

Equations (6) and (7) provide two distinct likelihood models to distinguish between misconfiguration and malicious, darknet-bound traffic, which enables their simplified and systematic post-processing. Furthermore, as the proposed model generalizes and formalizes the concepts of misconfiguration and malicious darknet traffic, it does not make any assumptions regarding the nature of the sources from which the given types of traffic are originating. Thus, the method deems a source and its corresponding flows as misconfiguration traffic if $\ln P_{misc}(D_i) - \ln P_{mal}(D_i) > 0$. To effectively employ the proposed network telescope sanitization model, we present Algorithm 1, which provides a simplistic yet effective mechanism to flag misconfigured sources.

Algorithm 1 Network Telescope Sanitization Algorithm

```
Input: Darknet Flows, $DarkFlows$
Output: Flag, $MiscFlag$, indicating that the respective flow is originating from a misconfigured source
for $DarkFlows$ do
    $MiscFlag \leftarrow 0$
    $i \leftarrow$ DarkFlows.getUniqueSources()
    Amalgamate $DarkFlows_i$ originating from a specific source $s_i$
    Update $s_i(D_i)$
    Compute $P_{misc}(D_i)$, $P_{mal}(D_i)$
    if $P_{misc}(D_i) > P_{mal}(D_i)$ then
        $MiscFlag \leftarrow 1$
    end if
end for
```

Since the field of Internet measurements for cyber security heavily relies on processing network telescope data [7, 35], we make the model's code available to researchers and security operators at large from https://github.com/COYD-IoT/COYD-IoT/tree/master/Darknet%20Sanitization.

3.2. Data collection and dataset preparation

In this section, we enumerate the methodology used to infer probing activities captured at a network telescope. We also detail the proposed mechanisms for feature engineering and active measurements, in order to fingerprint IoT devices through data-driven learning.

3.2.1. Inferring probing activities

After employing the aforementioned pre-processing steps to sanitize misconfiguration traffic, the aim is to dissect the malicious traffic to extract probing flows as perceived by network telescopes as indicators of exploitation. This is achieved through a Threshold Random Walk (TRW)-based probing detection algorithm [36]. This algorithm searches for subsequent packets from the same source IP address for a duration of 300 seconds. If a threshold is reached prior to the packet's arrival, the given counter is reset. If the threshold has held and the duration has not expired, the counter is incremented. If the counter reaches a threshold of 64 [37], the flow is deemed as a probing event.

3.2.2. Feature extraction for IoT classification

Following the amalgamation of packets into flows, the first t consecutive packets are extracted from each. Given that the majority of the observed scanning traffic are TCP SYN scans, the applicable features would reside in the TCP and IP header fields (i.e., ToS, Total Length, Identification, TTL, Dst IP Address, srcPort, dstPort, TCP SEQ, TCP ACK SEQ, TCP offset, TCP DATA Length, TCP Reserve, TCP Flags, TCP Win, TCP URP, TCP option, Packet Inter-arrival Time). Overall, along with the inter-arrival time of the consecutive packets within a flow, $d = 17$ features are gathered for each packet. In turn, the data samples for each scanner IP address would consist of a $t \times d$ matrix. To elaborate on the model's training procedure, we subsequently detail the labeling process.

3.2.3. Port scanning and banner grabbing

In order to annotate ~~decidedly~~ accurate labels for the training dataset, it was imperative to immediately perform the procedure ~~herein~~ upon detection of a scan activity to circumvent any potential complications due to the dynamic reallocation of the associated device's IP address (through DHCP, for instance). To accomplish this, we utilized the gigabit open-source Internet scanning tool ZMap [38] as well as the high-speed application scanner ZGrab [39]~~,~~ in tandem~~,~~ to provide comprehensive results necessary for guaranteeing the versatility of the classification task. Specifically, ZMap was used to probe 45 ports[1] of the IP addresses (that were previously inferred as probing sources) that were still found to be active. The port list are selected based on reports by ZoomEye [19] during one month analysis of returned banners to cover most of the default ports of various devices in order to maximize the number of captured banners. Furthermore, via ZGrab, we obtained banner fields and application handshakes from various protocols such as HTTP(s), CWMP, TELNET, SMTP(s), IMAP(s), POP3(s), SSH, FTP, SMB, DNP3, MODBUS, BACNET, FOX, Siemens S7 and SSL certificates. Additionally, we designed and developed two custom scanning modules to extract RTSP and SIP banners.

[1]`https://github.com/COYD-IoT/COYD-IoT/blob/master/Port-List.txt`.

3.2.4. Tagging and Labeling

We amalgamated a comprehensive list of keywords related to major Internet-facing IoT devices and vendors. As previously noted, these are typically the devices that are most targeted by IoT botnets. This list consists of devices provided by `Nmap` along with results from ZoomEye Internet Scanner[2] [19] and ZTag, Censys's tagging module[3]. Although it is unrealistic to claim that we cover all IoT products from all vendors, we indeed levarage information from various sources and focused on widely deployed Internet devices. In addition, we implemented a parsing algorithm which extracts useful keywords from banners and SSL certificates such as the combination of letters, digits, "-" and "_" signs, which typically represent device models [40] to enrich our list of devices. We further considered devices running multi-purpose OSs as non-IoT, which were identified using keywords such as "Win64", "Ubuntu", "Microsoft IIS" and "CentOS", etc. while we deemed other specialized devices as IoT where their OS types were indicated as being "embedded", "RouterOS", "FritzOS" etc. For example "TD-W8960N" is a sample keyword in the database related to a TP-LINK router that is marked as IoT. The prepared database consists of 3,286 patterns related to 1,121 vendors. We make this list publicly available at https://github.com/COYD-IoT/COYD-IoT/blob/master/devices.txt. We also filter out benign scanners[4] that have targeted the network telescope based on a Greynoise list[41] and returned information in banners.

3.3. Model training for fingerprinting compromised IoT devices

We propose herein a learning approach for the extraction of embedded features within unsolicited scan flows for the training of a binary classifier which distinguishes between traffic originating from both malicious IoT and non-IoT devices. The underlying methodology is based upon determining similarities in network traffic that are exclusively associated with IoT devices and their corresponding IoT malware in order to fingerprint flows originating from them. Additionally, it is known that IoT products manufactured by the same vendor possess a uniform, low-level architecture such as sharing a similar network card, operating system, etc., and happen to share the same TCP/IP stack information, including but not limited to TTL value and initial TCP window size, thus permitting the fingerprinting of IP addresses that Internet scanning services (i.e., Shodan) may have overlooked or could not identify.

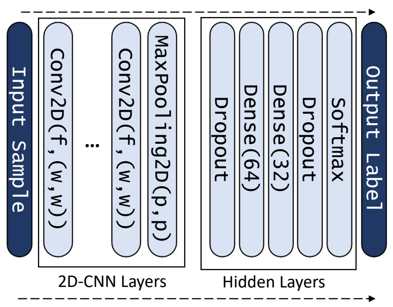
To select a suitable and a sound learning technique, we compare and contrast the performance of five models to permit the classification of compromised IoT devices in order to distinguish them from compromised, multi-purpose hosts. The first three are based on Convolutional Neural Networks (CNN), which are a category of deep learning models. Deep learning has been an emerging branch of machine learning that use multiple layers of feed-forward neural networks, backpropagation, and error correction to ~~automatically~~ learn features ~~(i.e., representations)~~ from a given data input. CNN is a state-of-the-art deep learning algorithm that uses dynamic kernels along a given data input and to ~~automatically~~ extract (i.e., pool) features. To this end, we asses a two-dimensional CNN (2D-CNN), a one-dimensional CNN (1D-CNN) [42] and a multi-window one-dimensional CNN (MW-1D-CNN) [43] in addition to two "shallow" learning methods ~~rooted in~~ Random Forest (RF) models.

[2] https://www.zoomeye.org/component

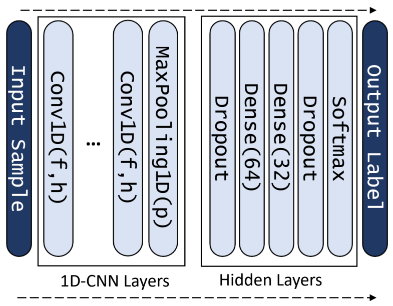
[3] https://github.com/zmap/ztag/tree/master/ztag/annotations

[4] https://github.com/COYD-IoT/COYD-IoT/blob/master/Benign-Scanners.txt

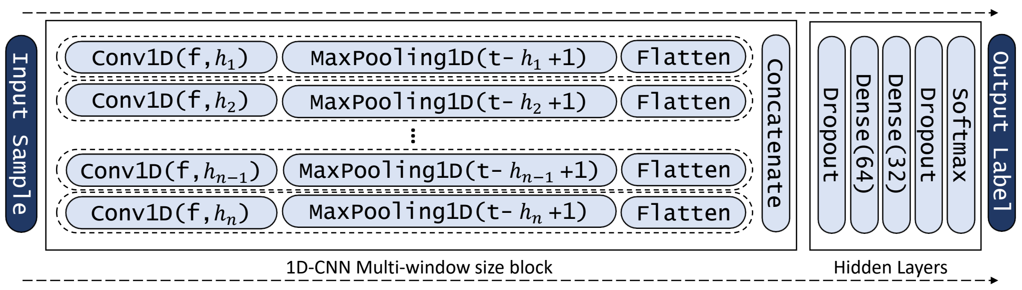
In this context, an input sample consists of a matrix representation $\mathbf{X}$ of a flow with t packets and the number of extracted fields d from a packet is considered, yielding $\mathbf{X} \in \mathbb{R}^{t \times d}$. Namely, the i^{th} packet in a given flow is $\mathbf{x}_i \in \mathbb{R}^d$. Convolution operations are also defined by applying local kernels $\mathbf{w} \in \mathbb{R}^{h \times w}$ on the input to extract spatially local correlations in the data. In terms of the 2D-CNN model, it contains L number of consecutive two dimensional convolutional layers (with k kernels of size $w \times w$) and max pooling, followed by two dense hidden layers of sizes 64 and 32, respectively, and a Softmax classifier at the end (Figure 3a). The 1D-CNN model has a similar architecture to the 2D-CNN, but instead, the convolution kernels have a fixed kernel width equal to the input sample width (i.e., $h \times d$) (Figure 3b). Further, the MW-1D-CNN model mixes the outputs of various kernel heights h to capture the features. In turn, the output of the first layer of the proposed model is given by $c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$, where $\mathbf{x}_{i:i+h-1}$ defines the notation for a sequence of packets $\mathbf{x}_i, \mathbf{x}_{i+1}, ., \mathbf{x}_{i+h-1}$, b representing the bias, and f denoting the non-linear activation function. The filter is applied to each 2D sample instance to produce a feature map $\mathbf{c} = [c_1, \ldots, c_{t-h+1}]$. Subsequently, max pooling is applied over the feature map $\mathbf{c}$, taking the value $\max \mathbf{c}$. We used kernels $\mathbf{w}$ of different window heights h ($h = [2, 4, 6, ..., h_{max}]$) to enable the capture of varying dynamics specific to darknet packet flows (Figure 3c).



(a) 2D-CNN model



(b) 1D-CNN model



(c) 1D-CNN multi-window

Figure 3: CNN models for IoT/non-IoT binary classification

We also devise two RF models, one that has been constructed using raw packet features while the other operates on feature statistics. We define feature statistics as the 5-tuple {*min*, 1-*quantile*, *median*, 3-*quantile*, *max*} of each field in flows of packets, which overall produces 85 features. These statistics can be considered as an estimation of the probability distribution function related to each field of the packet sequence in each flow. Please note that we make available the source code of the developed models, including their specificities from `https://github.com/COYD-`

IoT/COYD-IoT/tree/master/IoT%20classifier%20models.

3.4. *IoT botnets: features' extraction and campaign inference*

Following the binary classification of IoT-generated scanning activities while filtering out non-IoT sources by employing the developed trained model, we conduct a thorough investigation of the individual scan flows $\mathbf{Flow}_{IP}$, each comprising of at least 500 ($t \geq 500$) sequential packets originating from a particular unsolicited IoT device, and extract the corresponding feature set $\mathbf{F}_{IP}$ = < $\mathbf{Ports}_{IP}$, $\boldsymbol{\pi}_{IP}$, Flag_{IP}, ARR_{IP}>. $\mathbf{Ports}_{IP}$ is the grouping of the targeted transport protocols paired with their associated ports in ascending order (e.g., $\mathbf{Ports}_{IP_x}$ = {TCP:23, TCP:80, TCP:8080}). In turn, $\boldsymbol{\pi}_{IP}$ is the corresponding discrete probability distribution function which represents the frequency of appearance of each of these ports within the given flow of packets (e.g., $\boldsymbol{\pi}_{IP_x} = [0.15, 0.70, 0.15]$). This is relevant due to the fact that IoT devices typically possess a limited supply of resources and as a result, in the midst of conducting illicit scanning activities, are often allocated to different ports and weighted based on the expected return. Flag_{IP} is Boolean, holding a value 1 if the IoT device conducting the scanning has the signature $tcpSeq == dstIP$ and 0 otherwise. This inference provides insights about a `Mirai`-like behavior, possibly indicating a variant or a code-reuse practice. Lastly, the Address Repetition Ratio, or ARR_{IP}, is the ratio of the total number of packets sent by a source IP address over the number of unique destination IP addresses, and is defined as $\text{ARR}_{IP} = \frac{|\boldsymbol{Flow}_{IP}|}{|\{dstIP|dstIP \in \boldsymbol{Flow}_{IP}\}|}$. Such scenarios as an ARR_{IP} greater than one are a consequence of the sending of multiple packets to a particular destination in order to compensate for packet loss and/or the probing of multiple ports at each destination. Note that, each instance of the same probing campaign will exhibit an equivalent ARR_{IP} due to the underlying IoT orchestrated probing machinery.

Minimum number of packets *(t)* for robust feature estimation. To possess a correct and an accurate estimation of the discrete probability distribution $\boldsymbol{\pi}$, we perform statistical analysis to compute a suitable t. By generating a lower bound on the number of packets, we can guarantee a minimum error of 5% within a confidence level of 0.5. Note that at the /8 network telescope, scan packets arrive with a random probability of 1/256, resembling the random sampling procedure. We consider the population of scan packets originated by a compromised IoT device, and adopt a simple random sampling mechanism, as shown in equation (8) [44], to derive a lower bound on the sample size (equivalently, the number of received packets within the network telescope). The method herein is thus used to estimate the minimum sample size necessary to find the lower bound. By leveraging the requirements of a population proportion interval [44], we perform the estimation at a $1-\alpha$ confidence level, margin of error E and a planned proportion estimate p. By selecting more than n_0 samples, we assure that the probability that the actual error to be larger than E is not more than a small value α, i.e., $Pr(|p - P| \geq E) = \alpha$; where $z_{\alpha/2}$ is the $100(1-\alpha/2)$ percentile of the standard normal distribution.

$$n_0 = \frac{z^2_{\alpha/2} p(1-p)}{E^2} \tag{8}$$

Since the product $p(1-p)$ increases as p moves toward 0.5, a conservative estimation of the sample size is obtained by choosing $p = 0.5$, regardless of the actual estimated value of p. Therefore, using a 0.5 planned portion estimate, the sample size needed to achieve a 5% margin of error at 95% confidence level is computed at 385. In this work, we select the number of packets equal to 500 ($\geq$ 385) to significantly minimize the risk of errors in the extracted features, namely $\boldsymbol{\pi}$, to avoid subsequent issues in clustering and campaign detection.

Clustering mechanism. We hierarchically divide the IP addresses of the IoT scanners into separate groups $\boldsymbol{G}_i$ based on the given $\mathbf{Ports}_{IP}$, Flag_{IP} and ARR_{IP} of their feature set $\mathbf{F}_{IP}$. Upon completion, we cluster members of each group $\boldsymbol{G}_i$ to identify those scanning for the same set of ports but with a different probability distribution function $\boldsymbol{\pi}$. This enables us to leverage hierarchical agglomerative clustering [45], which determines the proximity matrix by calculating the distance between every pair of probability distribution functions $\{\boldsymbol{\pi}_{IP}|IP \in \boldsymbol{G}_i\}$ based upon the Jensen-Shannon Divergence (JSD) [46] distance metric. JSD, defined in (9), estimates the distance between two discrete distribution functions, and is the symmetrized version of the well-known Kullback-Leibler Divergence (KLD).

$$JSD(\boldsymbol{\pi}_i \| \boldsymbol{\pi}_j) = \frac{1}{2} KLD(\boldsymbol{\pi}_i \| \boldsymbol{M}) + \frac{1}{2} KLD(\boldsymbol{\pi}_j \| \boldsymbol{M}), \tag{9}$$

where $\boldsymbol{M} = \frac{1}{2}(\boldsymbol{\pi}_i + \boldsymbol{\pi}_j)$. and $KLD(\boldsymbol{P} \| \boldsymbol{Q}) = -\sum_i \boldsymbol{P}(i) \log(\frac{\boldsymbol{Q}(i)}{\boldsymbol{P}(i)})$ for discrete PDF $\boldsymbol{P}$ and $\boldsymbol{Q}$.

We select hierarchical agglomerative clustering due to the fact that based on the statistical analysis, the estimated $\boldsymbol{\pi}$ values are within the specific distance from the actual distribution (cluster centers) with high confidence. Therefore, it can correctly identify centers by merging close samples and executing a bottom-up approach. In addition, other clustering methods (such as k-means) are based on the assumption of equal cluster sizes which is not correct in the context of IoT campaigns while density-based techniques (such as DBSCAN [47]) are only suitable when the density of the data is non-uniform and the clusters can be in arbitrary shape. As noted, hierarchical agglomerative clustering operates in a bottom-up fashion. Each observation forms its own cluster and begins moving up the distance-based hierarchy, subsequently merging with the clusters. To designate appropriate consolidation, we use a distance threshold (of 0.05) in which merging only occurs if the distance between the two given cluster centers falls beneath.

4. Empirical evaluation

The evaluation was executed using 3.6 TB of darknet traffic that was collected for a 24-hour period on Dec. 13th, 2018. This data is provided by the Center for Applied Internet Data Analysis (CAIDA) `/8` network telescope. While this specific dataset *per se* is subject to MOUs and thus cannot be shared as is, interested readers can request access to CAIDA's real-time darknet data through DHS IMPACT [48]. Additionally, while we make available a sample collected at another `/13` darkent IP space available through the `GitHub` repository for experimentation purposes, we note that the developed and open-source methods are generic enough to be applied on any darknet data (within any desired time frame).

4.1. Results of the darknet sanitization model

By executing the proposed model of Section 3.1, the distribution of malicious and misconfiguration traffic with respect to the number of packets were found to be 88.21% and 11.21%, while the distribution of source IP addresses were 26.17% and 73.83%, respectively. Validation of such outcome revealed that close to 90% of the misconfiguration traffic defines packets that hit the `/8` network telescope only once, while the remaining appeared to be malformed packets. Further, it can be observed that even though misconfiguration traffic is relatively low (11.79%), it is responsible for a large proportion of the source IP addresses (73.83%). These findings shed more light on the problematic nature of misconfiguration traffic with regards to Internet measurements via network telescopes and emphasize the effectiveness of the proposed pre-processing model.

Table 1: Distribution of malicious and misconfiguration traffic in the /8 network telescope dataset

	Malicious	**Misconfiguration**
Traffic	88.21%	11.79%
Sources	26.17%	73.83%

In terms of the runtime specifics of the implementation which heavily relied on the Linux-derived libpcap C++ library, running on an Ubuntu 18.04 system with a quad core Intel Core i7-8550 at 1.80GHz processor, and 16GB of RAM, the developed approach processes sets of 8 GB data files each containing close to 67.5 million packets by requiring on average around 636 seconds of execution time while consuming close to 11.6 GB of RAM. We believe this could be considerably improved by switching to SSD storage (since most of the delay was I/O related) and by adopting multithreading.

4.2. Results of dataset preparation

Regarding the data collection and dataset preparation steps of Section 3.2, and by immediately scanning back about 1.7M Internet scanners inferred through the network telescope, about 25.84% of them were found to have at least one open port. Further, amongst total $543,392$ gathered banners, most of them were HTTP (54.11%), FTP (11.10%), SSL Certificate (10.50%), TELNET (10.19%), RTSP (7.00%), and CWMP (2.60%). Having completed this task, we were able to distinguish between 45,184 IoT and 7,763 non-IoT devices to design the training data set. At this juncture, the label and corresponding metadata were incorporated into $t \times d$ training and test data matrices of IoT and non-IoT devices. We shuffled them and then performed normalization by way of the Min-Max method [49]. Subsequently we computed and removed the mean. To evaluate the proposed model, we trained it using a prepared dataset captured in Nov. 2018 and then tested it using our dataset from Dec. 2018. The one month gap between the training and test datasets ensured that there exists no correlation between them for sound evaluation. The test dataset consisted of 34,974 IoT and 7,193 non-IoT sources.

4.3. Evaluating the IoT classification models

The proposed CNN models were implemented in Keras [50]. To address the problem of class imbalance within the training dataset, cost-sensitive learning was applied [51]. The number of epochs was found to be 30 to avoid over-fitting. Further, we performed a random search on subspaces of hyper-parameters as presented in Table 2, leveraging Hyperas [52], and selected the best model (out of 100 trials) with regards to the loss. RF models were implemented and trained using the scikit-learn [53] package. The best model was retrieved based upon random search (using the RandomizedSearchCV method) in the search space as summarized in Table 3. In Tables 2 and 3, parameter ranges are reported with `begin:step:end` format. For evaluating the CNN models, we leverage an NVIDIA GeForce RTX 2070 GPU with 8GB of memory, 2304 CUDA cores and 288 Tensor cores to accommodate for parallelization.

To compare the performance of the different devised models, we rely on standard machine learning metrics such as precision, recall and F-measure for the IoT class. Precision is the ratio

Table 2: Tuned hyperparameters of the selected CNN models

Parameters	Space	2D-CNN	1D-CNN	MW-1D-CNN
Optimizer	SGD, Adam, RMSProp	RMSProp	RMSProp	RMSProp
Num. of kernels (k)	32,64,128	32	128	64
Kernel size ($w \times w$)	(2,2),(3,3)	(2,2)	-	-
Kernel height (h)	2,4,8,16,32,64	-	64	-
Max kernel height (h_{max})	40:10:80	-	-	80
Pool size (p)	2,3	2	3	-
Batch size	128, 256	128	256	256
Activations	Relu, Sigmoid, Tanh	Sigmoid	Tanh	Sigmoid
Dropout	$U(0.1, 0.3)$	0.195	0.296	0.298
learning rate	0.001	0.001	0.001	0.001
Num. CNN layers (L)	1:1:4	4	3	-

Table 3: Tuned hyperparameters of the RF models

Parameters	Space	RF on raw fields	RF on Quantiles
Num. estimators	20:20:100	60	60
Max depth	4:4:20	12	12
Min samples leaf	2:10:102	52	52
Min samples split	$U(2, 10)$	6	4
Bootstrap	True, False	False	False
Criterion	Gini, Entropy	Entropy	Gini

of correctly classified IoT devices over all the instances that have been designated as IoT using the proposed model ($precision = \frac{tp}{tp+fp}$). Recall is the ratio of correctly classified IoT devices over the total number that is actually existing within the test data ($recall = \frac{tp}{tp+fn}$). Recall demonstrates the model's ability to find all relevant cases within a given dataset, whereas precision gives the model's ability to designate only the actual relevant cases as relevant. In order to bring these two metrics together, often F-measure is employed which takes the weighted average of precision and recall, i.e., the harmonic mean ($F\text{-}measure = 2.\frac{precision.recall}{precision+recall}$).

We report the results in Figures 4 and 5. We can note that the AUC-ROC score for the RF model trained on quantiles is slightly higher than that of the other models. Further, both of the figures reveal that the CNN-based models result in higher recall and lower precision scores in contrast to the RF models. The outcome also shows that the multi-window 1D-CNN (MW-1D-CNN) outperforms the 1D-CNN and the 2D-CNN; this is quite expected, since packet fields (unlike image pixels) lack temporal or spatial relationships with one another, and thus moving the kernels over the horizontal dimension would not lead to better learning. Furthermore, the multi-window 1D-CNN can capture varying dynamics given that only a portion of the scan packets actually hit the $/8$ darknet.

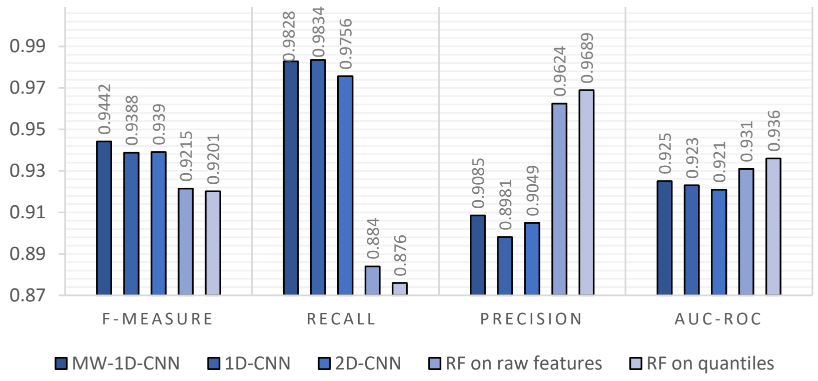


Figure 4: Performance metrics of the devised models

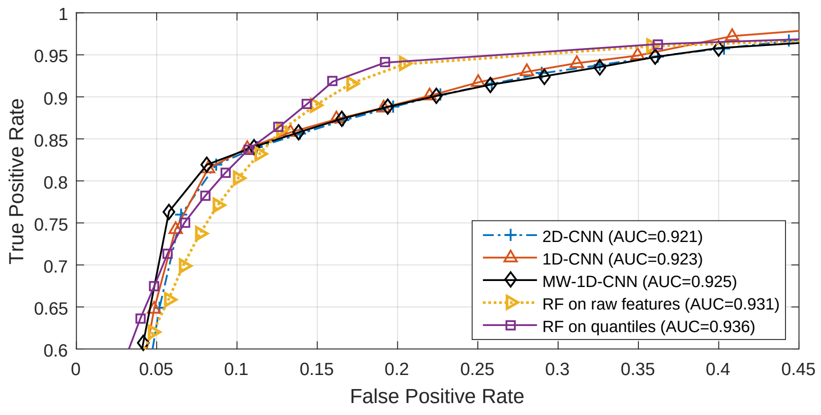


Figure 5: AUC-ROC curves to evaluate the devised models

*4.3.1. **Features' importance***

To shed light on which features were most decisive in the learning process, and given that the RF models performed the highest, we illustrate the features' scores (derived from the RF model on quantiles) in Figure 6. As expected, the distribution of destination ports which typically reveals the scans' intentions plays the most noteworthy role for fingerprinting IoT devices. This is closely followed by other fields such as the total packet length and the total header length, in addition to the TCP/IP stack and OS-related fields such as the TCP window size, option fields and the TTL.

*4.3.2. **Effect of number of packets (t) on the classifiers***

Figures 7 and 8 illustrate the impact of the number of packets within the input sample $\mathbf{X} \in \mathbb{R}^{t \times d}$ on the AUC-ROC and processing time (loading and training data). To quantify the effect, for each value of t, we execute the training process 10 times using parameters taken from the best models, retrieved from Tables 2 and 3. Although it is expected that increasing the number of packets will increase the total amount of information to be processed, subsequently increasing a model's performance, it is not consistently proven. Reviewing the results in Figure 7, when a RF model

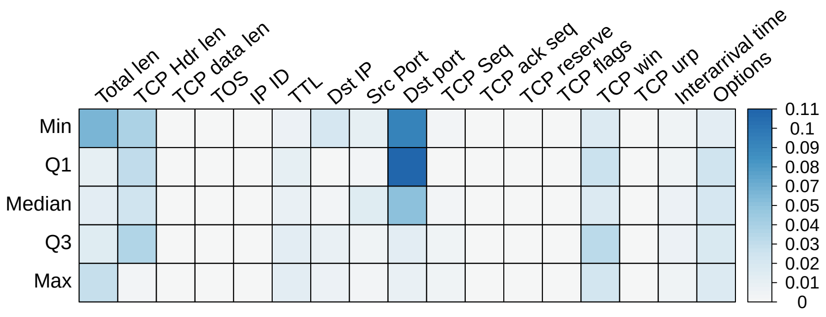


Figure 6: Ranking of features' importance

is trained using raw features, adding an increasing amount of packet data will eventually confuse the model, lowering the AUC-ROC of the RF model. In contrast, when a RF model is trained on quantiles, increasing the amount of input packet data actually lead to an improvement in the AUC-ROC of the model, with diminishing returns. In addition, it can be seen that changing *t* has no significant trending effect on the AUC-ROC of CNN-based models.

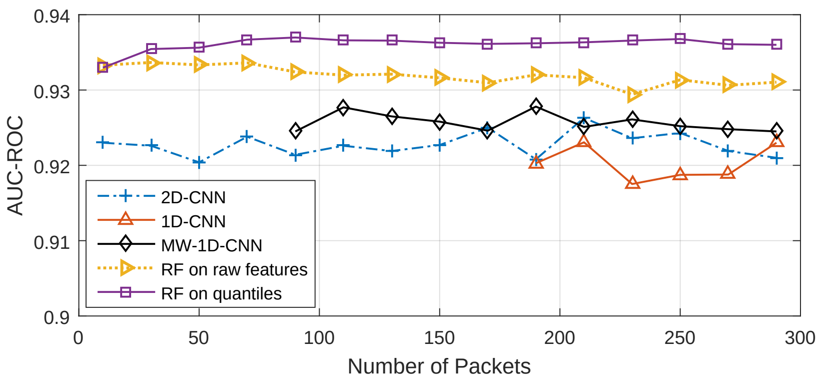


Figure 7: Effect of packet number (*t*) on AUC-ROC

Figure 8 reveals that an increased sample size, containing a larger number of packets, will generally increase the processing time. Most evidently perceived in the MW-1D-CNN model, its high complexity leads to a significant increase in processing time as the sample size is increased. However, an increased sample size leads to a slight decrease in processing time for a quantile-trained RF model. Ultimately, the results depict that maximum AUC is achieved through training an RF model with t = 90. Furthermore, the non-RF models have an acceptable performance and AUC value at *t* = 90. Therefore, to facilitate future implementations and experimentation, *t* = 90 is found to be a suitable choice for efficient and accurate classification.

4.4. *Inferring and characterizing compromised IoT devices and campaigns*

Given the aforementioned classification results, we selected the MW-1D-CNN model since it provided the highest true positive rate while strictly limiting the false positive rate to around 0.08 (Figure 5). We further re-trained the model on recent data from Dec. 2018 to accompany for any evolving dynamics.

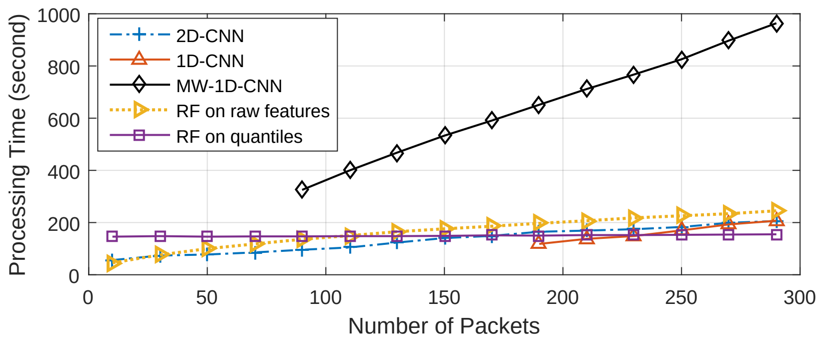


Figure 8: Effect of packet number (t) on processing time

By applying the binary classifier on 24 hours of darknet data of Dec. 13th, it was capable of fingerprinting 441,766 out of the 1,787,718 unique scanners to be originating from compromised IoT devices. Although previous works solely considered those with a `Mirai` signature as IoT-related [3], we inferred that in fact, they make up less than 25% of the IoT scanner population that the proposed model was able to uncover, leaving a whopping 75% to go about their malicious activities without any semblance of an adequate attribution.

Table 4 summarizes the location of these exploited devices, where Brazil (41.93%) was found to be hosting a significant portion, followed by Iran (10.17%), China (5.14%), Russia (3.59%), Egypt (3.36%), India (2.47%) and Turkey (2.32%).

Table 4: Top countries hosting infected IoT devices

Country	(%)	Country	(%)
Brazil	41.93	Greece	1.70
Iran	10.17	Italy	1.60
China	5.14	United States	1.42
Russian	3.59	Indonesia	1.25
Egypt	3.36	Mexico	1.24
India	2.47	Ukraine	1.21
Turkey	2.32	Korea (south)	1.07
Taiwan	2.13	United Kingdom	0.83
Vietnam	1.91	Thailand	0.72
Argentina	1.83	Spain	0.66

Furthermore, the top three ISPs hosting the largest number of compromised IoT devices were Vivo $(134,021)$, TE Data $(11,804)$ and Iran Telecom Co. $(9,912)$.

While the extensive presence of IoT scanners in and of itself gives pause for concern, a relatively significant proportion residing within the telecommunication and ISP sectors is rather expected; conversely, their existence within sectors including but not limited to critical sectors is quite alarming. In Table 5, critical sectors which appear in lists provided by the United States Department of Homeland Security (DHS) and the European Union (EU) are highlighted [54].

Amongst the inferred instances, quite a few were found to be located within that of medical infrastructures (87), government entities (86), manufacturing realms (99), and commercial businesses (38).

Table 5: Top Sectors hosting infected IoT devices

Sector	**Count**
Telecommunications	175,642
Internet Service Provider	82,238
Private Service	1,319
Internet Hosting Services	780
Education	485
Internet Colocation Services	314
Data Services	99
Health	87
Government	86
Manufacturing	74
Finance	57
Lodging	44
Professional Service	38
Transportation	29

Along those lines, the lengthy list of 50 identified vendors reveals a broad range of manufactures and device types that IoT botnets demonstrate preference for exploitation. Amongst them, `MikroTik` (14,090), `Aposonic` (2,222), `Huawei` (732), `Foscam` (594) and `Hikvision` (417) are the topmost five targeted by the tagged compromised devices. Routers (53.64%) and IP Camera/DVR (28.93%) continue to be the most frequently infected devices.

Moreover, the most commonly targeted ports based upon the number of scanning packets generated by the compromised IoT devices are reported in Table 6. The top targeted ports include 23 (%41.9), 80 (%23.9), 8080 (%19.7), 5555 (%4.9), 81 (%3.2), 2323 (%1.7) and 22 (%1.3). Intriguingly, we identified the presence of non-IoT targeted ports such as 2480 (OrientDB), 5984 (CouchDB), 3389 (RDP), 7001 (Oracle), 5900 (VNC) and 2004 (Drupal), as well as that of uncommonly used IoT ports 32764 (router backdoor), 37215 (UPnP in SOHO routers) and 52869 (UPnP in wireless chipsets). Set {23,80,80} is the most prevalent target port combination; 54% of devices actually only scan this port combination.

Inferring and reporting on orchestrated IoT botnets. Among the 441,766 IoT scanners that were detected on Dec. 13th, 2018, those that sent less than 500 packets were filtered out to exclude any of those that began at the end of the day, which can degrade the output of the probability distribution function $\boldsymbol{\pi}$.

Subsequently, the respective features were extracted and the clustering method described in Section 3.4 was executed. In roughly 40,000 scan flows, we witnessed a share of the scanning packets arriving at UDP ports. After analyzing such occurrences, we deduced they resulted from associated bugs or attacks on P2P networks such as BitTorrent; an observation that is also consistent with [55]. As a result, in order to avoid the ill-effects of uncorrelated incidents,

Table 6: TCP port distribution determined by quantifying the number of compromised IoT scan packets received by the each included port. Grey cells highlight unconventional, rarely probed ports and services

Port	Service	(%)	Port	Service	(%)	Port	Service	(%)
23	Telnet	41.912	8181	HTTP-alt	0.114	83	HTTP-alt	0.028
80	HTTP	23.917	88	HTTP-alt	0.057	443	HTTPS	0.025
8080	HTTP-alt	19.784	21	FTP	0.056	3389	RDP	0.023
5555	ADB	4.995	7547	TR-064	0.053	8090	HTTP-alt	0.018
81	HTTP-alt	3.288	8081	HTTP-alt	0.050	8089	HTTP-alt	0.018
2323	Telnet-alt	1.705	8888	HTTP-alt	0.047	139	SMB	0.006
22	SSH	1.391	37215	UPnP	0.045	7001	WebLogic	0.005
9000	MCTP	0.470	2480	OrientDB	0.041	52869	UPnP	0.005
445	SMB	0.315	5984	CouchDB	0.040	8291	Winbox	0.004
5358	Telnet	0.238	82	HTTP-alt	0.029	1433	MS-SQL	0.004
8000	HTTP-alt	0.197	8001	HTTP-alt	0.029	5900	VNC	0.003
2222	SSH	0.165	8088	HTTP-alt	0.028	2004	Drupal	0.003
8443	HTTP	0.121	84	HTTP-alt	0.028	1900	UPnP	0.002
32764	Linksys Vuln.	0.117	85	HTTP-alt	0.028	Other	-	0.596

the identified packets were removed prior to clustering. Regarding the inferred campaigns, the proposed approach detected over 350 orchestrated IoT botnets. Since the size of each IoT probing campaign translates to its given severity, we summarize those botnets possessing more than 300 coordinated IoT bots in Table 7. Interestingly, in solely considering IoT scanners that targeted the set of ports {23, 80, 8080}, we detected 30 distinct botnets with differing distributions, Flag (i.e., Mirai-like signature/behavior), and ARR.

Table 7: Orchestrated IoT botnets in the wild

Id	Ports	Flag	ARR	#Bots	π	Crypto Miners	Compromised Devices
1	23, 80, 8080		1	139,858	[0.33 0.33 0.34]	⬡ Ⓜ	MikroTik, Hikvision, Foscam, Vivotek, Huawei, Aposonic, Intelbras, Ubiquiti, Netgear, Mitrastar, Askey, Archer
2	23, 80, 8080		1	55,139	[0.294 0.295 0.411]	⬡ Ⓜ	MikroTik, Hikvision, Intelbras, TP-LINK, D-Link, Huawei, ZTE, Foscam, QNAP, ZyXEL, Cisco, SERCOMM, Vivotek
3	23, 2323	✓	1	36,464	[0.9 0.1]	⬡	Huawei, Aposonic, Foscam, Hikvision, Mikrotik, Cisco, TP-LINK, CIG Shanghai, ZTE, Ubiquiti
4	80	✓	1	12,895	[1.]	⬡ Ⓜ	Huawei, Hikvision, MikroTik, AvTech, ZTE, Foscam, Cisco, Ubiquiti, NUUO
5	5555	✓	1	11,050	[1.]	Ⓜ	Huawei, TP-Link, Hikvision, Aposonic, Foscam , MikroTik, Sagemcom, iGate, VNPT, Trendchip
6	23, 81		1	9,805	[0.495 0.505]	Ⓜ	Aposonic, Foscam, Huawei, Hikvision, ZTE, Lilin, Sagemcom, Netgear

7	23, 80, 8080		2	7,610	[0.171 0.650 0.179]	⬢Ⓜ	MikroTik, TP-LINK, Hikvision, AvTech, Foscam, D-Link
8	23	✓	1	7,200	[1.]	⬢Ⓜ	Huawei, Hikvision, TP-Link, AvTech, TP-LINK, Aposonic, ZEM800, ZTE
9	23, 80, 8080		1	5,971	[0.242 0.244 0.514]	Ⓜ	MikroTik, ZTE, Hikvision, TP-LINK, Foscam
10	23		3	5,491	[1.]		DZS, Foscam, MikroTik, Synology, ZyXEL, Hikvision
11	80, 8080		1	5,162	[0.492 0.508]	⬢Ⓜ	MikroTik, Foscam, Hikvision, Huawei, TP-LINK, Ubiquiti
12	23		1	4,689	[1.]	⬢	D-LINK, Hikvision, Aposonic, MikroTik, Broadcom, ASUS, AVM, Netgear
13	23, 80, 8080		1	4,468	[0.442 0.032 0.526]	⬢Ⓜ	MikroTik, TP-LINK, Hikvision, D-Link
14	23		4	3,911	[1.]		GPON (DZS), Hikvision, Huawei, MikroTik, Dasan, Foscam, Mercusys
15	22, 2222	✓	1	3,783	[0.897 0.103]	⬢Ⓜ	QNAP, Huawei, Hikvision, ASUS, Foscam, SERCOMM, MikroTik, Intelbras, Ubiquiti
16	23, 2323, 5555	✓	1	3,545	[0.249 0.032 0.719]	Ⓜ	ZyXEL, MikroTik, Avtech, Broadcom, Foscam, TP-LINK, Hikvision, D-Link
17	23, 2323	✓	1	2,727	[0.967 0.033]	⬢	Huawei, ZTE, Hikvision, MikroTik, Aposonic, ZEM800, Foscam
18	23		2	2,146	[1.]		TP-LINK, Hikvision
19	23, 32764, 80, 8000, 8080, 8081, 8089, 8090, 81, 8181, 8443, 8888, 9000	✓	1	2,140	[0.034 0.122 0.153 0.02 0.154 0.02 0.019 0.02 0.068 0.123 0.122 0.022 0.121]		NUUO, Foscam, Hikvision, Huawei, AVM, MikroTik, Aposonic
20	23, 8080		1	1,591	[0.48 0.52]	⬢Ⓜ	MikroTik, Hikvision, D-Link, TP-LINK
21	23, 80, 8080		1	1,286	[0.384 0.319 0.298]	⬢Ⓜ	MikroTik, SERCOMM, Foscam
22	80, 8080+rnd		1	1,247	[0.45 0.45]		MikroTik
23	23, 81		1	1,191	[0.095 0.905]		Aposonic, Foscam, Hikvision
24	23, 80, 8080		1	1,083	[0.226 0.5 0.274]	⬢Ⓜ	MikroTik, D-Link, Foscam, Aposonic
25	23, 5358		1	1,059	[0.5 0.5]		Hikvision, Foscam, Intelbras
26	23, 2480, 5555, 5984, 80, 8080+rnd	✓	1	783	[0.126 0.120 0.134 0.121 0.128 0.121]		Foscam, Huawei, Aposonic, Hikvision
27	80, 8080		3	756	[0.814 0.186]	Ⓜ	MikroTik, TP-LINK
28	443, 80, 8000, 8001, 8080, 8081, 8088, 81, 82, 83, 84, 85, 88, 8888	✓	1	723	[0.071 0.071 0.071 0.071 0.071 0.071 0.071 0.072 0.072 0.072 0.071 0.072 0.071 0.072]		Synology, Hikvision
29	23, 2323	✓	1	691	[0.794 0.206]		Huawei, Aposonic, Hikvision
30	23, 9000		1	677	[0.49 0.51]		Sagemcom, SERCOMM, Hikvision, Cisco, Huawei, Aposonic, AVM, Mikrotik, Foscam
31	23		5	642	[1.]		ZTE, Hikvision, Foscam, MikroTik, Netgear
32	80		1	616	[1.]	⬢Ⓜ	MikroTik, Hikvision, Huawei
33	23, 80, 8080		1	544	[0.15 0.3 0.55]	Ⓜ	MikroTik, Huawei, ZTE, Hikvision, Vivotek, Foscam, Aposonic

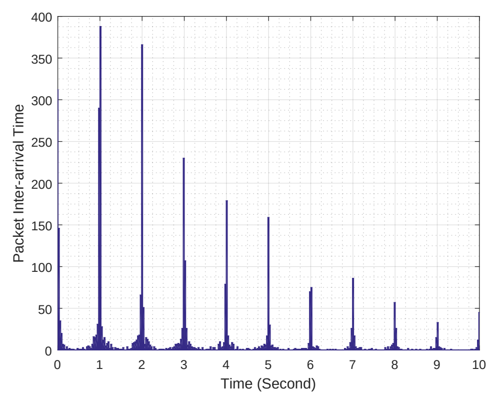
34	23, 81	1	541	[0.291 0.709]		Aposonic, Huawei, Hikvision, Foscam, TP-LINK
35	23, 445, 80, 8080	1	376	[0.3142 0.0587 0.3155 0.3115]	⬡ Ⓜ	MikroTik, Aposonic
36	23, 7547, 80, 8080, 8291	1	340	[0.334 0.002 0.33 0.331 0.002]	⬡ Ⓜ	MikroTik, Hikvision

⬡ `Coinhive` Ⓜ `xmrMiner`

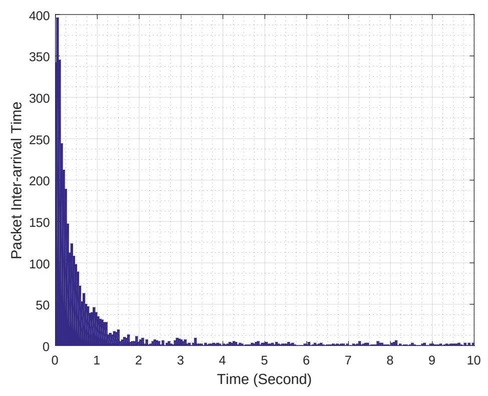
Packet inter-arrival time analysis generated from the inferred botnets. Following the investigation of scan-based behaviors of the inferred compromised IoT devices, we deduced two separate classes of unique scan traits. Class 1 includes devices that present periodic behavior in the time series of their packet Inter-Arrival Time (IAT). For rate limiting purposes, such devices seem to generate a fixed number of packets then wait exactly 1 second to re-confirm their desired scanning rate (in packets per second (pps)). This leads to high peaks in their histograms of packet IATs as seen in Figure 9a. In contrast, the members of the Class 2 do not portray any related periodic behavior when analyzing their packet IAT. Figure 9b portrays the IAT of the packets generated by the IoT devices of Class 2, demonstrating an exponential distribution. To detect the aforementioned behaviors, we first calculate the histogram of packet IATs and then identify the peaks with an auto-correlation coefficient (Figures 9c and 9d). To reveal the population of such inferred classes in the context of the identified probing campaigns, Figure 10 illustrates the fraction of scanning classes in each campaign.

Scan rate analysis of the inferred IoT botnets. Figure 11 presents the distribution of scanning rates extracted from the inferred IoT botnets, as perceived by the network telescope. Campaigns in which their scan rates follow a normal distribution with a single peak and a narrow width (such as #1, #2, #6, #11, #14, #15, #18 and #20 of Table 7) were found to be exhibiting strong rate limiting policies regardless of their identified scanning class. In contrast, the rates of botnets #3, #8 and #17 are distributed over a wider range, showing no artificial rate limiting behaviors. In fact, this inference matches the released source code of the `Mirai` malware (which botnet #3 is attributed to), demonstrating the lack of rate limiting practices. For classes that were discovered to have no artificial rate limiting usages, each individually-compromised device sent a maximized number of scan packets based on their processing power and throughput. By focusing on the scanning rates' distributions of the inferred campaigns coupled with their population of scanning classes (Figure 10), we can deduce some facts about the purity of the clustered campaigns; permitting further scrutiny of such campaigns to determine if they contain a singular or multiple botnets. Note that IAT-related features were not included during the clustering mechanism of Section 3.4. With this in mind, we further examined the identified campaigns looking for Class 1 scan traits (those employing rate limiting practices) while their rates' distributions showing negative outcomes related to following a single normal distribution. The outcome, for instance, showed that although botnet #5 was identified as possessing Class 1 scan traits, its distribution of scanning rate shows 3 distinct peaks. Therefore, we postulate that there may be three unique botnets that were misidentified within a singular cluster. Nevertheless, this is quite expected for campaigns with a single target port (and eventually $\boldsymbol{\pi} = [1]$), similar to campaign #5.

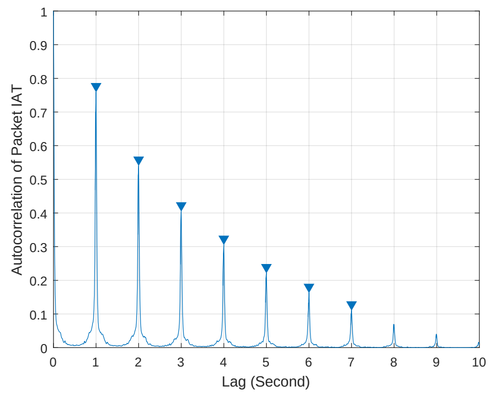
1



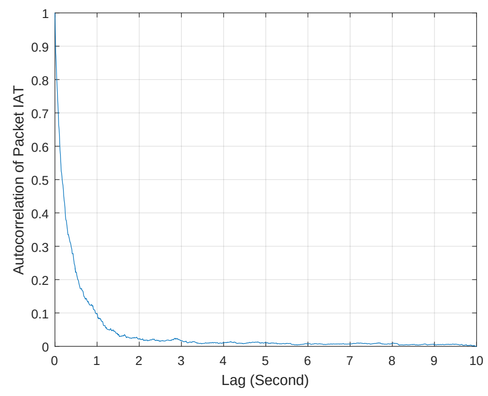
(a) Sample histogram of Class 1. Originated from an IoT device in Campaign 6



(b) Sample histogram of Class 2. Originated from an IoT device in Campaign 4



(c) Autocorrelation of (a) followed by peak detection.



(d) Autocorrelation of (b) followed by peak detection.

Figure 9: Sample of packet IAT's histograms for Class 1 (a) and Class 2 (b) scanning practices. (c) and (d) respectively show the auto-correlation of (a) and (b) and the detected peaks.

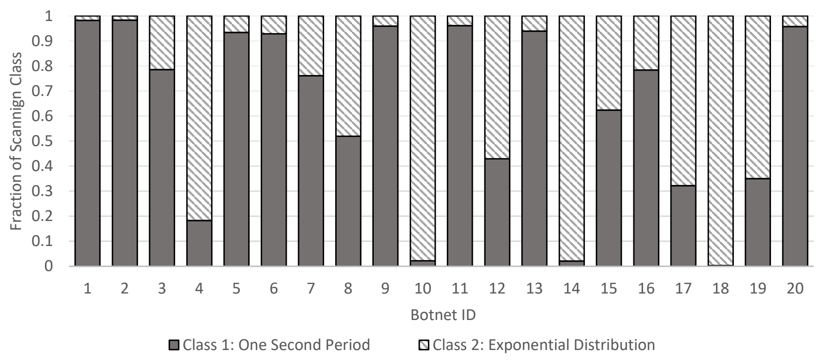


Figure 10: Fraction of the derived scanning practices within the top populated campaigns (as detailed in Table 7).

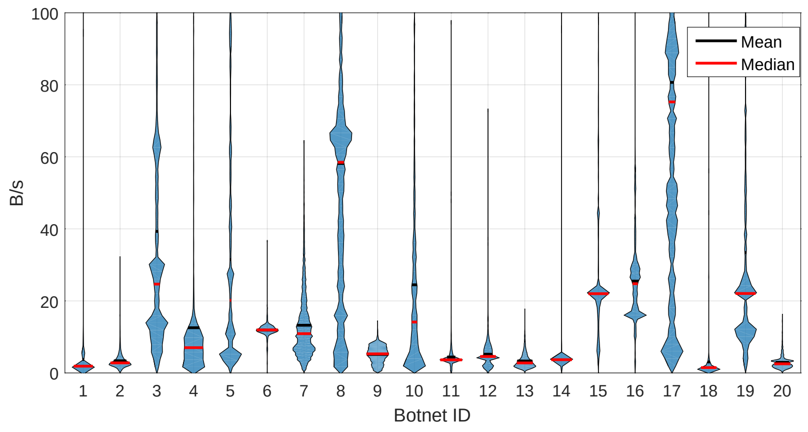


Figure 11: A Violin illustration of the top 20 populated campaigns' scanning rated as observed on the /8 network telescope

Geo-distributions of the inferred IoT botnets. By examining the geo-distribution characteristics of the identified botnets, we observe significant differences. Figure 12 depicts the geo-distribution of the most populated campaigns generating scan events from multiple continents. Indeed, geo-distribution characteristics are likely a direct result of the popularity differences related to the adopted device types and manufacturers, which is known to be unique to each region [20]. Since botnets leverage definitive attack vectors, they are typically customized to target specific vendors; coherently, the vendor's popularity will also attract botnets, which is reflected in the concentration of such popular devices/bots in certain geo-locations. For further analysis, Figure 13 displays the cumulative distribution of campaigns over Autonomous Systems (AS). Despite a few cases with highly similar distributions (e.g., #2 and #9), other specially less populated botnets are discovered to have a larger difference between distributions, as shown in Figure 13b. Furthermore, amidst the inferred campaigns, there exists campaigns whose geo-distributions do not comply with that of the global distribution of infections (Table 4). For instance, with respect to campaigns #7, #13, #24 and #27, over 98% of infected IoT devices are located in Iran. Campaign #30 has upwards of 50% and 10% of compromised IoT devices located in USA and UK, respectively. Further, campaign #28 shows a 40% infection rate in North America, and a 21% infection rate in Europe. Spread across multiple geographic regions, these campaigns contradict the global distribution of infections.

IoT botnets with cryptojacking capability. Aside from the dominant monetization method for IoT botnets of performing DDoS attacks, cryptojacking has become a noteworthy technique [56, 57]. In essence, compromised routers have become responsible for injecting JavaScript crypto-currency miners into the HTTP pages requested by devices on their network [58]. JavaScript miners such as `Coinhive` [59] and `xmrMiner` [60] strive for Monero altcoin in particular. To this end, we examined the responses to HTTP requests derived from the IoT scanners, tagging those that contain the `xmrMiner` or `Coinhive` JavaScript modules, and exporting their corresponding keys. By doing so, we discovered 1,134 `xmrMiner` and 923 `Coinhive` instances with 23 and 30 distinct keys, respectively. The campaigns designated as containing members with cryptojacking capabilities are highlighted in Table 7. The relation of crypto-mining keys

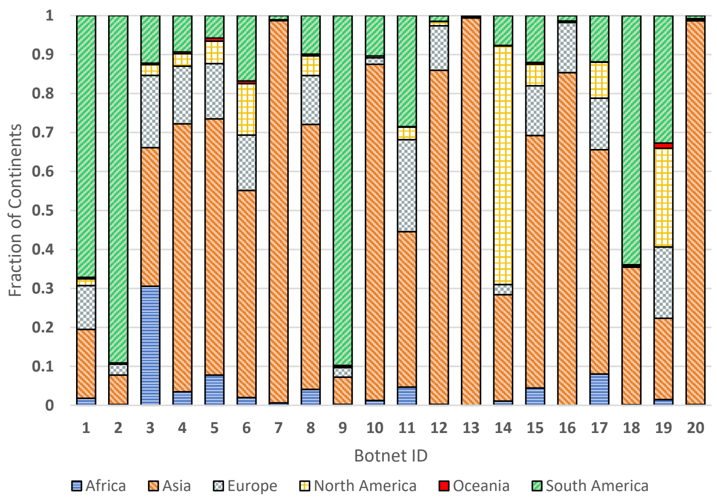
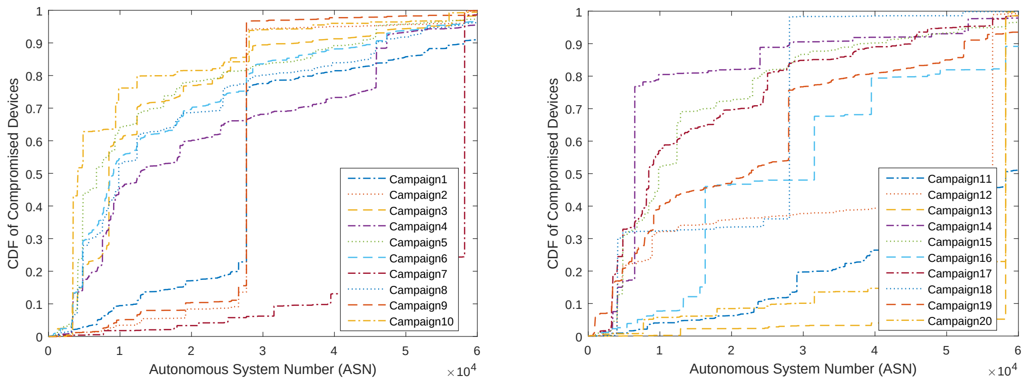


Figure 12: Geo-distribution of the top 20 populated campaigns over different continents



(a) Campaigns 1-10

(b) Campaigns 11-20

Figure 13: Cumulative distribution of the top 20 populated campaigns (of Table 7) over different Autonomous Systems

which appeared in each campaign is illustrated using a 3-partite graph as in Figure 14. We analyzed the graph to find the number of components to uncover any further relations between the campaigns. Interestingly, the graph is found to be a connected graph which demonstrates that there exists salient connections between all the campaigns involved in cryptojacking activities.

In addition, we uncovered large campaigns maintaining crypto miner instances with and without the presence of Mirai-like signatures. Moreover, 943 out of 1,134 devices, belonging to a total of 18 campaigns (#1, #2, #4, #5, #7, #8, #11, #13, #15, #16, #19, #20 #21, #24, #32, #33, #35 and #36), share the same xmrMiner-related key "4983e34ef01b4b579725b3a228e59e79" (red edges in Figure 14). In other words, large portions of immense IoT campaigns could be reported to be attributed to the same "player". On top of that, upon exploring the key within `Censys`, 54,743 `Mikrotiks` were shown to possess it. In total, these campaigns equate to approximately 250,000 compromised IoT devices; a significant 54% of all the identified compromised devices.

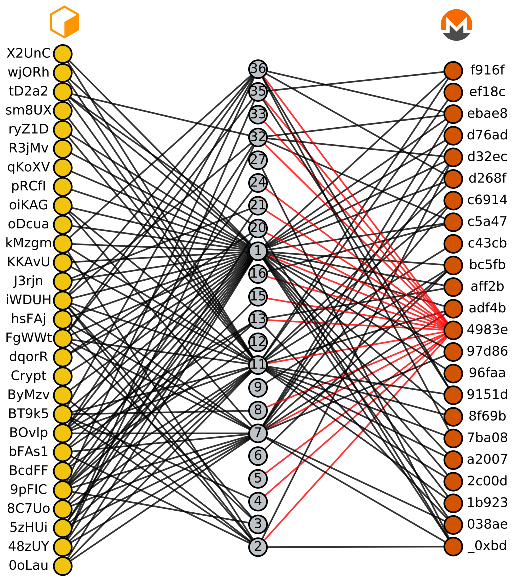


Figure 14: A 3-partite graph of discovered cryptomining-related keys within each campaign. Each key is represented by its first 5 characters. Red threads highlight the reuse of the same key "4983e34ef01b4b579725b3a228e59e79" in 18 different campaigns.

A closer look at other campaigns of interest. Campaign #3 with 36,464 bots was inferred to be targeting ports 23 and 2323 with a proportion of 9:1, which is the same as instructed within the `Mirai` released code. Another interesting observation is related to botnet #26 (of Table 7) where packets to random TCP and UDP ports were sent in addition to targeting the defined set of ports of {23, 2480, 5555, 5984, 80, 8080}. Additionally, this campaign targeted port 2480 (OrientDB) and 5984 (CouchDB), as well as other common IoT-related ports including 23, 5555 (ADB) and 8080. Upon further analysis, this behavior could be attributable to the infamous `Hide and Seek` botnet [61].

In the context of port 32764 which is related to a backdoor vulnerability [62], the proposed IoT botnet clustering approach revealed a campaign of substantial size (#19 in Table 7), consisting of 2,140 active IoT scanners with the following signature <{23, 32764, 80, 8000, 8080, 8081, 8089, 8090, 81, 8181, 8443, 8888, 9000}, Flag=1, ARR=1>. We did not come across any previously reported botnet families that scan such ports. As a result, we postulated that this campaign is either new or specific ports have been recently added to the target list of a previously known IoT botnet. Another aspect is that this is the only large campaign that exploited a relatively significant number of `NUUO` products, which is a common indicator of the `Reaper` IoT botnet. The `JenX` botnet [63], which scans ports 37215 and 52869, was also disclosed. Moreover, a botnet with <{2004, 80, 8080, 81}, Flag=0, ARR=2 > was also discovered and consisted of 35 coordinated IoT scanners, all of which compromised QNAP NAS. This campaign strongly resembles that of the `Muhstik` botnet [64], with the exception of the substitution of port 7001 with 81 in the target port set.

With the prevalence of IoT botnets, port 5555 (Android debug bridge) has become a popular target port. We found 23 IoT botnets that port 5555 is among their target port set. Based on the reports on ADB miner [65] and the similarity of its scanning module to `Mirai`, we can attribute the inferred large IoT botnet (#5 in Table 7) to `Mirai` or its variant `Fbot` [66]. Additionally, we found `xmrMiner` instances with the same previously noted key (of Figure 14 in the latter campaign and in campaign #16. Based off the set of target ports pertaining to campaign #25 (port 23, 5358), it seems to be highly likely attributed to that of the `Hajime` [12] [67] IoT botnet. In total, this campaign possessed 1,059 active IoT scanners (made up of IP cameras/DVRs).

A note on Industrial Control Systems (ICS). We also inferred an IoT botnet of 25 bots with the signature <{102, 8888, 993}, Flag=0, ARR=1>, probing Siemens S7 (heavily used in SCADA systems), IEC 61850 and ICCP (both are mostly used in utility/electric substations) on port 102. To provide additional insights, we also actively scanned each of the identified compromised IoT devices for ICS open ports on TCP and UDP 102 (S7), 502 (MODBUS), 20000 (DNP3), 47808 (BACNET) and 1911 (FOX) and found 100, 101, 465, 70 and 85 devices with open ports, respectively. We note that we have also inferred close to 40 devices having simultaneously all the above-mentioned ICS ports open, which we thought are related to ICS honeypots. Nevertheless, the appearance of compromised IoT devices within ICS setups is alarming.

5. Concluding Remarks

With the continuous adoption of the IoT paradigm in critical infrastructure and consumer sectors, their security and privacy concerns are becoming quite serious, leading to devastating consequences. This work compliments current IoT-centric research by offering a macroscopic, generic and passive methodology to infer Internet-scale compromised IoT devices and to report on ongoing IoT botnets. The work initially introduces a novel darknet-specific sanitization model that contributes to the field of Internet measurements at large. Subsequently, by devising a binary classifier based upon a CNN in conjunction with active measurements, the proposed work is capable of fingerprinting compromised IoT devices by solely operating on darknet traffic. Consequently, by automating the generation of signatures related to the ports being probed coupled with their distribution in addition to other simplistic yet effective features, the proposed approach provides the capability to infer ongoing orchestrated botnets. The results demonstrate the significant security issue with the IoT paradigm by exposing more than 400,000 exploited IoT devices during only a 24-hour period, some of which have been deployed in critical sectors such medical and manufacturing. Additionally, the outcome provides evidence-based indicators

related to ongoing IoT botnets such as those of `Mirai`, `Hide and Seek`, and `Reaper`, to name a few. More interestingly, the results demonstrate evolving IoT botnets with cryptojacking capabilities, where many of those seem to be attributed to the same mastermind by exposing the same employed key.
Future work will address a number of current limitations. Examples include addressing the misidentification of two distinct IoT botnets (as one larger campaign) by including packet IAT related features, and improving the tagging/labeling procedure. We will also be examining IoT-specific malware samples and devising formal methodologies between the traffic they generate from one side and the corresponding darknet traffic from the other side, to fortify the attribution evidence. Moreover, extracting a number of features from IoT malware binaries will empower the attribution of each malware sample to their respective campaigns, enabling agile IoT botnet inference and characterization for mitigation and remediation purposes.

Acknowledgments

The authors would like to express their sincere gratitude in advance to the anonymous reviewers and editors for their constructive feedback. This work was supported by a grant from the U.S. National Science Foundation (NSF) (Office of Advanced Cyberinfrastructure (OAC) #1907821).

[1] L. Da Xu, W. He, S. Li, Internet of things in industries: A survey, IEEE Transactions on industrial informatics 10 (4) (2014) 2233–2243.

[2] E. Bertino, N. Islam, Botnets and internet of things security, Computer (2) (2017) 76–79.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al., Understanding the mirai botnet.

[4] R. Nigam, New mirai variant adds 8 new exploits, targets additional iot devices, https://unit42.paloaltonetworks.com/new-mirai-variant-adds-8-new-exploits-targets-additional-iot-devices/ (2019).

[5] L. Cashdollar, Latest echobot: 26 infection vectors, https://blogs.akamai.com/sitr/2019/06/latest-echobot-26-infection-vectors.html (2019).

[6] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, J. Schönfelder, A survey on honeypot software and data analysis, arXiv preprint arXiv:1608.06249.

[7] C. Fachkha, M. Debbabi, Darknet as a source of cyber intelligence: Survey, taxonomy, and characterization., IEEE Communications Surveys and Tutorials 18 (2) (2016) 1197–1227.

[8] Shodan, The search engine for internet of things, http://shodan.io (2019).

[9] C. Team, Internet-wide scan data repository, Retrieved (2017) 2017.

[10] S. Torabi, E. Bou-Harb, C. Assi, M. Galluscio, A. Boukhtouta, M. Debbabi, Inferring, characterizing, and investigating internet-scale malicious iot device activities: A network telescope perspective, in: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2018, pp. 562–573.

[11] F. Shaikh, E. Bou-Harb, N. Neshenko, A. P. Wright, N. Ghani, Internet of malicious things: Correlating active and passive measurements for inferring and characterizing internet-scale unsolicited iot devices, IEEE Communications Magazine 56 (9) (2018) 170–177.

[12] S. Herwig, K. Harvey, G. Hughey, R. Roberts, D. Levin, Measurement and analysis of hajime, a peer-to-peer iot botnet.

[13] E. Bou-Harb, M. Debbabi, C. Assi, A novel cyber security capability: Inferring internet-scale infections by correlating malware and probing activities, Computer Networks 94 (2016) 327–343.

[14] A. Dainotti, A. King, K. Claffy, F. Papale, A. Pescapé, Analysis of a /0 stealth scan from a botnet, IEEE/ACM Transactions on Networking (TON) 23 (2) (2015) 341–354.

[15] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, S. Savage, Inferring internet denial-of-service activity, ACM Transactions on Computer Systems (TOCS) 24 (2) (2006) 115–139.

[16] C. Fachkha, E. Bou-Harb, M. Debbabi, On the inference and prediction of DDoS campaigns, Wireless Communications and Mobile Computing 15 (6) (2015) 1066–1078.

[17] O. Cetin, C. Ganán, L. Altena, T. Kasama, D. Inoue, K. Tamiya, Y. Tie, K. Yoshioka, M. van Eeten, Cleaning up the internet of evil things: Real-world evidence on isp and consumer efforts to remove mirai.

[18] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, J. A. Halderman, A search engine backed by Internet-wide scanning, in: 22nd ACM Conference on Computer and Communications Security, 2015.

[19] ZoomEye, http://www.zoomeye.org/ (2019).
[20] All things considered: An analysis of iot devices on home networks, in: 28th USENIX Security Symposium (USENIX Security 19), USENIX Association, Santa Clara, CA, 2019.
URL https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-deepak
[21] H. Guo, J. S. Heidemann, Detecting iot devices in the internet (extended), 2018.
[22] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, Y. Elovici, Profiliot: A machine learning approach for iot device identification based on network traffic analysis.
[23] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, S. Tarkoma, Iot sentinel: Automated device-type identification for security enforcement in iot, in: Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on, IEEE, 2017, pp. 2177–2184.
[24] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, A.-R. Sadeghi, D\" iot: A crowdsourced self-learning approach for detecting compromised iot devices, arXiv preprint arXiv:1804.07474.
[25] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, M. Gurusamy, Deft: A distributed iot fingerprinting technique, IEEE Internet of Things Journal.
[26] A. J. Pinheiro, J. d. M. Bezerra, C. A. Burgardt, D. R. Campelo, Identifying iot devices and events based on packet length from encrypted traffic, Computer Communications 144 (2019) 8–17.
[27] S. Siby, R. R. Maiti, N. O. Tippenhauer, Iotscanner: Detecting privacy threats in iot neighborhoods, in: Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security, IoTPTS '17, ACM, New York, NY, USA, 2017, pp. 23–30. doi:10.1145/3055245.3055253.
URL http://doi.acm.org/10.1145/3055245.3055253
[28] G. Acar, D. Y. Huang, F. Li, A. Narayanan, N. Feamster, Web-based attacks to discover and control local iot devices, in: Proceedings of the 2018 Workshop on IoT Security and Privacy, ACM, 2018, pp. 29–35.
[29] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, C. Rossow, Iotpot: A novel honeypot for revealing current iot threats, Journal of Information Processing 24 (3) (2016) 522–533.
[30] J. Guarnizo, A. Tambe, S. S. Bunia, M. Ochoa, N. Tippenhauer, A. Shabtai, Y. Elovici, Siphon: Towards scalable high-interaction physical honeypots, arXiv preprint arXiv:1701.02446.
[31] L. Metongnon, R. Sadre, Beyond telnet: Prevalence of iot protocols in telescope and honeypot measurements, in: Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity, ACM, 2018, pp. 21–26.
[32] M. Ford, J. Stevens, J. Ronan, Initial results from an ipv6 darknet13, in: International Conference on Internet Surveillance and Protection, IEEE, 2006, pp. 13–13.
[33] Z. Durumeric, M. Bailey, J. A. Halderman, An internet-wide view of internet-wide scanning., in: USENIX Security Symposium, 2014, pp. 65–78.
[34] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Rubin, Bayesian data analysis, Vol. 2, Taylor & Francis, 2014.
[35] N. Koroniotis, N. Moustafa, E. Sitnikova, Forensics and deep learning mechanisms for botnets in internet of things: A survey of challenges and solutions, IEEE Access 7 (2019) 61764–61785.
[36] J. Jung, V. Paxson, A. W. Berger, H. Balakrishnan, Fast portscan detection using sequential hypothesis testing, in: Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on, IEEE, 2004, pp. 211–225.
[37] C. Rossow, Amplification hell: Revisiting network protocols for ddos abuse., in: NDSS, 2014.
[38] Z. Durumeric, E. Wustrow, J. A. Halderman, Zmap: Fast internet-wide scanning and its security applications., in: USENIX Security Symposium, Vol. 8, 2013, pp. 47–53.
[39] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, J. A. Halderman, A search engine backed by internet-wide scanning, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 542–553.
[40] X. Feng, Q. Li, H. Wang, L. Sun, Acquisitional rule-based engine for discovering internet-of-things devices, in: 27th {USENIX} Security Symposium ({USENIX} Security 18), 2018, pp. 327–341.
[41] GreyNoise, https://viz.greynoise.io/ (2019).
[42] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of Machine Learning Research 12 (Aug) (2011) 2493–2537.
[43] Y. Cheng, F. Wang, P. Zhang, J. Hu, Risk prediction with electronic health records: A deep learning approach, in: Proceedings of the 2016 SIAM International Conference on Data Mining, SIAM, 2016, pp. 432–440.
[44] W. G. Cochran, Sampling techniques, John Wiley & Sons, 2007.
[45] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Annals of Data Science 2 (2) (2015) 165–193.
[46] S.-H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, City 1 (2) (2007) 1.
[47] K. Khan, S. U. Rehman, K. Aziz, S. Fong, S. Sarasvady, Dbscan: Past, present and future, in: The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014), IEEE, 2014, pp. 232–238.
[48] I. M. F. POLICY, A. O. C.-R. . TRUST, https://impactcybertrust.org/ (2019).
[49] S. García, J. Luengo, F. Herrera, Data preprocessing in data mining, Springer, 2015.

[50] F. Chollet, et al., Keras, https://keras.io (2015).

[51] N. Thai-Nghe, Z. Gantner, L. Schmidt-Thieme, Cost-sensitive learning methods for imbalanced data, in: Neural Networks (IJCNN), The 2010 International Joint Conference on, IEEE, 2010, pp. 1–8.

[52] M. Pumperla, https://github.com/maxpumperla/hyperas (2019).

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[54] M. Husák, N. Neshenko, M. S. Pour, E. Bou-Harb, P. Čeleda, Assessing internet-wide cyber situational awareness of critical sectors, in: Proceedings of the 13th International Conference on Availability, Reliability and Security, ACM, 2018, p. 29.

[55] K. Benson, Leveraging internet background radiation for opportunistic network analysis, Ph.D. thesis, UC San Diego (2016).

[56] S. I. S. T. R. (ISTR), Cryptojacking: A modern cash cow, Tech. rep.
URL https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-cryptojacking-modern-cash-cow-en.pdf

[57] H. Tuttle, Cryptojacking, Risk Management 65 (7) (2018) 22–27.

[58] A. Zimba, Z. Wang, M. Mulenga, Cryptojacking injection: A paradigm shift to cryptocurrency-based web-centric internet attacks, Journal of Organizational Computing and Electronic Commerce 29 (1) (2019) 40–59.

[59] Coinhive, https://coinhive.com/, [Online; accessed 01-March-2019] (2018).

[60] xmrminer, Monero web miner, https://xmrminer.cc/ (2019).

[61] Y. Rootkiter, Hns botnet recent activities, https://blog.netlab.360.com/hns-botnet-recent-activities-en/, [Online; accessed 01-March-2019] (2018).

[62] S. L. Thomas, Backdoor detection systems for embedded devices, Ph.D. thesis, University of Birmingham (2018).

[63] I. X.-F. Exchange, Jenx botnet, https://exchange.xforce.ibmcloud.com/collection/JenX-Botnet-c47476c5e6fafd7df487cecd1110a761, [accessed 01-March-2019] (2018).

[64] Y. Rootkiter, Botnet muhstik is actively exploiting drupal cve-2018-7600 in a worm style, https://blog.netlab.360.com/botnet-muhstik-is-actively-exploiting-drupal-cve-2018-7600-in-a-worm-style-en/, [Online; accessed 01-March-2019] (2018).

[65] 360Netlab, Adb.miner: More information, https://blog.netlab.360.com/adb-miner-more-information-en/, [Online; accessed 01-March-2019] (2019).

[66] 360Netlab, Fbot, a satori related botnet using block-chain dns system, https://tinyurl.com/yavvhf4v, [Online; accessed 01-March-2019] (2019).

[67] radware, Hajime botnet friend or foe?, https://security.radware.com/ddos-threats-attacks/hajime-iot-botnet/, [Online; accessed 01-March-2019] (2018).