Inferring Software Composition and Credentials of Embedded Devices from Partial Knowledge

Pierre-Marie Junges, Jérome François, Olivier Festor

Inferring Software Composition and Credentials of Embedded Devices from Partial Knowledge

Pierre-Marie Junges, Jérome François, Olivier Festor
University of Lorraine, Inria, LORIA, France
Email: firstname.lastname@loria.fr

***Abstract*—Internet-of-Things (IoT) devices or more generally embedded devices are nowadays commonly deployed in public, personal or work spaces despite suffering from security issues often related to their bad design and/or configuration. For instance, IoT botnets such as Mirai successfully compromised thousands of devices using a bruteforce method on a set of known credentials. Although brute-force attacks against a particular service (*e.g.* SSH, telnet) generate many packets which can be easily detected and mitigated, attackers can easily rely on TCP scans to assess the services present on a device while maintaining a high level of stealthiness. In this paper, we present a method to reconstruct precise information about an IoT device configuration (brand name, usernames, passwords, software components) from partial knowledge such as open ports revealed by a TCP scan. It relies on constituting a knowledge base from a large dataset of publicly accessible firmware serving as training multiple Random Forest (RF) classifiers. Using a dataset of 6935 embedded devices, the HTTP, SSH or DNS software names can be predicted with a precision higher than 80% with a limited knowledge. The correct HTTP, SSH or DNS versions can be inferred in more than 95% of cases after 1.4 *trials* on average. Similarly, our technique also predicts the password of at least one valid user in more than 97% of the cases after 1.15 *trials* on average.**

I. Introduction

Estimated to about 50 billions in 2020 [13], the large growth of Internet of Things (IoT) devices combined with their lack of appropriate protection [27] resulted in an increase of attacks targetting embedded devices. In this paper, and with respect to the literature [8], [9], the terms embedded devices and IoT devices are equivalent and can be used interchangeably.

Awareness on security issues in IoT strongly increased with the recent botnets such as Mirai or BASHLITE that compromised between 100k and 1M devices to then performed massive DDoS (Distributed Denial-of-Service) attacks up to 600 Gbps [3]. As a first step, attackers usually performs an Internet-wide scan to look for exploitable devices and can rely on massive scan tools such as ZMAP [12]. It is known that even industrial systems are exposed in Internet [14]. However, only a list of accessible IP addresses is retrieved assuming one or more given ports. An attacker is also often interested in knowing the type, brand, model and software of the device using a fingerprinting technique to customize a device-specific attack with a better efficiency (*e.g.* using specific default passwords or exploit code). In contrast, a naive approach brute-forcing usernames or passwords or even testing various exploits without particular knowledge about the target will take time and is not stealthy which could be easily detected by intrusion detection systems (IDS).

Although many fingerprinting methods rely on network traffic analysis assuming to be in the vicinity of the device [24], [25], remote probing usually leverages banner grabbing as for example in [11], [20], [33] or with the Shodan search engine[1]. However, banners are not always available or recommended to not disclose sensitive information. For example, web servers commonly do not disclose the software name and version following the recommendation of the reference project OWASP[2] (Open Web Application Security Project). In [33], less than 700,000 packets from 11 millions present a relevant banner to fingerprint the IoT devices. Faking information in banners is also an alternative to defeat attackers [15].

Because information in banners is not reliable, our objective is thus to quantify the level of information an attacker can infer about the configuration of an embedded device (usernames, passwords, software components) from partial knowledge, mostly from the exposed services on devices after performing an Internet-wide scan on selected TCP/UDP port numbers. Except the TCP/UDP scan, the inference is done offline, the attack remains subtle in comparison with brute-force attacks or when trying multiple payloads till success.

This paper introduces a multi-level fingerprinting method based on Random forest (RF) classifiers. A first classification level aims at identifying the device brand while a second level is constituted of brand-specific classifiers. We assume different fingerprinting objectives to assess the exploitable information indirectly exposed be devices which can be exploited by an attacker to 1) improve its attack success rate and 2) reduce the number of interactions with the target (*i.e.,* increase the stealthiness). To evaluate many different configurations on a large set of devices, we decided to rely on a firmware analysis on 6935 embedded devices to extract details about each device default configuration.

Additionally, our work also highlights weak programming practices which have been presented in [27] and in the OWASP[3] top 10 IoT 2018 as a security threat.

The rest of this paper is structured as follows; Section II presents the related work on device fingerprinting and Internet-wide scanning. In Section III, we discuss about the challenges of our approach. Section IV introduces our metrics

[1] www.shodan.io

[2] https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server

[3] https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf

and classification process then our objectives are presented in Section V. Section VI evaluates the results of our experiments. In Section VII, we interpret our results and approach. Finally, Section VIII concludes our paper.

II. Related work

Fingerprinting techniques [5], [6], [19], [24]–[26], [30], [31], [33] have been proposed to identify compromised or newly connected IoT devices in a network. In [25], IoT devices are identified using signatures constructed with 23 features (*e.g.,* packet size, source port) extracted from the network traffic flows and vulnerable devices are isolated thanks to a Software-Defined Networking mitigation solution. Similarly in [30], the authors construct signatures from features and statistics (*e.g.,* protocols present, average ip header and payload length) also extracted from the network flows and used a multi-classification algorithm based on a support vector machine. In [5], the authors propose behavioral features which are less impacted by header spoofing by concatenating a feature set over a sequence of packets. An autonomous technique is introduced in [24] to identify IoT devices using a frequency-based analysis of their network traffic flows which is then used to detect whether or not a device has been compromised. To ensure up-to-date signatures, DEFT [31] is a distributed systems where IoT gateways and a central controller collaborate to update and distribute fingerprints. Inferring information from encrypted IoT communications have been investigated in [19], [26]. In [26], the objective is to retrieve the type and the model of the device while the authors in [19] identify actions performed by the users through an IoT gateway.

Unlike these passive approaches, our technique does not require to be in a close vicinity of the devices (*i.e.,* to observe the network traffic) and aims at assessing specific configuration details (*i.e.,* software components) about an embedded device solely from features that can be obtained from active scanning or probing (*e.g.,* services open). Complementary to these works, fingerprinting at the radio level has also been considered as for example in [22].

Our work can be so qualified as an active fingerprinting technique similarly to [6], [33]. The former [33] matches packet header values (*e.g.,* TTL, application-layer data) to keywords (*e.g.,* device type, brand, model) related to known IoT devices and then construct a neural network to infer the device type, brand or model from the header values. In [6], the identification of wireless devices relies on the bit structure of the responses to malformed or non-standard 802.11 frames. Our approach is similar to [33] because we assume that the devices are probed remotely. However, our fingerprinting approach promotes a very fine grain. Our intent is to predict the configuration of the device: usernames, passwords, software names and versions such as these configuration details could be then used by attackers to perform specific attacks.

Search engines such as Censys[4] or Shodan scan the Internet and extract software components information (*i.e.,* names and versions) from the banners exposed. In [11], the authors assess the performances of Censys at returning the list of devices having vulnerabilities or specific services options. On the other hand, [20] proposes an Internet-wide scanning engine, based on ZMap [12], that is faster at scanning the Internet while accessing up to 29% more devices compared to the existing search engines. The authors also introduce an OS fingerprinting technique based on features extracted from the IP header (*e.g.,* TTL, MSS) and HTTP banner and obtained an accuracy of 51% using the k* algorithm.

Regarding firmware analysis, existing approaches such as [9], [10], [18], [29], [32] discovered vulnerabilities and extracted security-related information (*e.g.,* credentials, software names and versions) which can also be used by attackers to build a knowledge database. In our work, we used the software components and credentials extracted from a firmware analysis to automatically reconstruct precise knowledge from a very small set of information which could be retrieved remotely.

III. Attack scheme

The sections describes how information gathered from firmware analysis can be used by an attacker to enhance the efficiency and the steathiness of an attack.

Fingerprinting tools [16] such as *nmap*[5] or *masscan*[6] infer information about the running services of a device.

On one hand, beside the ports (*i.e.,* services) opened on a device, it is often difficult to retrieve the exact software names and versions used by the running services. However, this information is critical from a security perspective as it would allow an attacker to finely tune its attacks against a particular target. Furthermore, the packets generated by the fingerprinting tools can be detected by IDS or IPS which reduces the future attacks stealthiness.

On the other hand, manufacturers often propose embedded device firmware images on their websites. A firmware analysis can be used to build a database with all the software information and credentials present in the downloaded firmware images. This is the first step illustrated in Fig. 1. In step (2), a classifier can be trained to create a model taking as input partial information which can be easily retrieved later, for example by a probing the device remotely to identify running services (open TCP/UDP ports), and predicting information which cannot be directly observed (brand, software names, versions, default usernames and passwords). Step (4) corresponds to the application of the trained model by reconstructing knowledge of a device from partial information given by an external tool such as a TCP/UDP port scanner or any other tools (information on the public HTTP page of the device for example) in step (3). Obviously, multiple classifiers can be trained depending on the partial information taken as input and the desired knowledge to infer as the output. In this paper, we target different relevant objectives which are detailed in Section V.

[4]www.censys.io

[5]https://nmap.org/

[6]https://github.com/robertdavidgraham/masscan

In step (5), the extracted knowledge is helpful to select a limited set of attack instances (a default password to test or a vulnerability exploit). Ideally, the tool should only select one out of n attack instances which fits the target device. In practice, such a technique can be leveraged to reduce the number of instances to try. As a result, the attacker can improve its attack success rate while minimizing its interactions with the targeted device to increase his or her stealthiness. So, the rationale is that interactions in step (3) are less critical than in step (5). Indeed, systems are unfortunately constantly probed on multiple ports [23] but do not consider it as a severe threat since there is no malicious payload or no real attempts to connect to. Furthermore, we only consider the nine services listed in Tab. I which drastically limits the number of ports to probe.

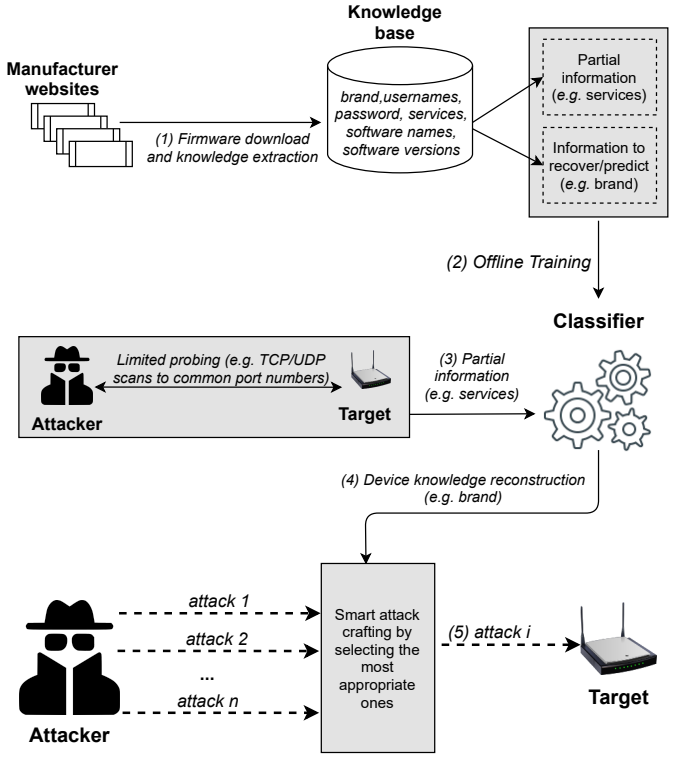


Fig. 1: Attack scheme

Therefore, in this paper, assuming attackers is able to construct such a fingeprinting approach, our objectives are to measure how precisely security-wise information (*i.e.,* software <name, version>, credentials) could be predicted.

IV. Multi-label classification

In this paper, we employed the Random forest (RF) classification algorithm because it does not require a large dataset in contrast to deep learning models, handles multi-class and multi-label problems (*e.g.,* inferring the usernames of a device) and has shown to be effective in similar problems [7], [28].

A. Knowledge extraction and device representation

Using the firmware analysis framework described in [18] that, given a list of embedded devices manufacturers websites, downloads all the available firmware images, unpacks them, and then extracts the software information (*i.e.,* name, version) related to the services listed in Tab. I using predefined regular expressions on their bytecode or execution output. Our analysis focuses on the protocols DNS, FTP, HTTP, NTP, RPC, SNMP, SSDP/UPnP, SSH and TELNET. We selected these protocols because 1) they are generally accessible through the Internet [33], 2) SSH and TELNET are often used as entry points [4], [21] by attackers, 3) FTP handles the storage of private data (*e.g.,* camera footage) so its compromise often results in user privacy leakage and 4) remaining UDP protocols can be tricked by attackers to perform reflection and/or amplification attacks [1], [2].

TABLE I: Services investigated

services	software
DNS	bind8, bind9, dnsmasq, tinydns, knotd, nsd, maradns, deadwood, pdnsd, posadis, pdns, unbound, yadifad
FTP	ftpd, vsftpd, uftpd, uftp, bftpd, proftpd
HTTP	httpd, nginx, boa, lighttpd, hiawatha, mongoose, thttpd, jjhttpd, minit_httpd, uhttpd, cherokee, jetty, monkey, hfs, navi, shttpd
NTP	ntpd, openntpd, sntp, sntpd, ntimed, ntpsec, chronyd
RPC	rpc.statd, rpcd
SNMP	snmpd, net-snmp, nagios, prometheus, zabbix, icinga, cacti
SSDP/UPnP	ssdp, upnpd, miniupnpd, minissdpd, minidlna, ushare, mediatomb, gmediaserver, gerbera
SSH	dropbear, openssh, apache mina, copssh, teleport, wolfssh
TELNET	telnetd, utelnetd

A firmware image can be used by one or multiple devices so, for the remaining of this section, we used the terms device and firmware image interchangeably.

We defined for each device d, its list of services exposed S_d and Sw_d the list of software used to deploy the S_d which is composed of one or more software sw such as:

- sw_{name} is the software name from Tab. I,
- $sw_{version}$ is the software version.

Additionally, the valid usernames u_d and their associated passwords pwd_{u_d} are extracted from the content of the filenames matching the $*/etc/\{passwd, shadow\}*$ pattern. We defined a username as valid if its password is not locked as indicated by the particular symbols in Unix systems (*e.g.,* $*$, ! or $*LK*$). Thus, each device d is associated with a brand $brand_d$. Because all these features are categorical, one-hot encoding is executed as a pre-process of RF.

B. Classification process

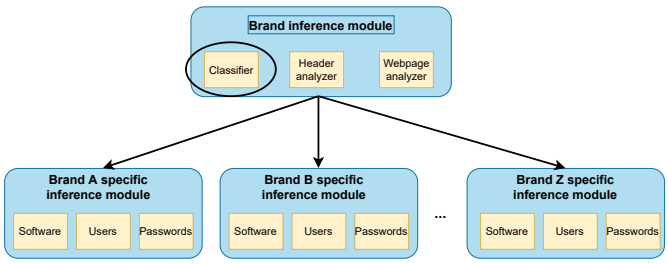


Fig. 2: Classification approach

As shown in Fig. 2, our classification process relies on two inference modules:

1) Main brand inference module: The top level aims at inferring the brand of the device. In this paper, we rely on a RF classifier trained with the features described in Section IV-A. Other techniques are possible such as 1) the network traffic analysis (*i.e.,* header analyzer) which requires access to thousands of real or emulated devices for the training stage or 2) an analysis of the public webpages content hosted by a device which often contain detailed information (*e.g.,* brand, model) about the device but we cannot assume all devices have a webserver.

2) Brand-specific inference module: This module uses the knowledge base restricted to data belonging to the devices from a specific brand to train the classifier and aims at inferring detailed software information (*e.g.,* names, versions) and the credentials. This module also highlights the firmware development process (*e.g.,* updates, software deployed) of a specific manufacturer. For instance, predicting with a 100% precision the version of any software indicates that the manufacturer mostly relies on the same software stack for all its devices. Such a lack of fine customization can be considered as a security concerns as it may lead to having too many irrelevant programs on a device [18].

For the remaining of this paper, the statistics per brand are generated using the brand-specific classifiers whereas the others are from our main classifier.

V. Classifier objectives

Our work aims at quantifying the level of information an attacker can potentially reconstruct from partial knowledge. Below is a list of objectives with their related motivation that an attacker may try to reach. In particular, for O2 and O3, our previous paper [18] demonstrates the frequent use of vulnerable SSH and HTTP servers in embedded devices making these services appealing for the attackers. Each objective can be instantiated in multiple manners depending on the partial information used as input as highlighted in Table II.

- **O1: Infer the brand name of a connected device solely from its open ports or partial software information (SSH, HTTP and DNS software names)**
  Knowing the brand name is useful for attackers because devices built by the same manufacturer often share a lot of commonalities in terms of software composition and so potential vulnerabilities [18]. Also, a few default credentials are brand-specific – Section VI-D.
- **O2: Infer the HTTP software name and version**
  HTTP servers are often available in embedded devices to provide graphical interfaces to the users. Vulnerabilities affecting web servers are unfortunately very frequent (*e.g.,* path traversal, broken authentication[7]). However, many vulnerabilities are specifically tailored for a particular web server. We are thus interested in investigating how the attackers can predict the HTTP software name and version – Section VI-E1.
- **O3: Infer the SSH software name and version**
  SSH protocol can be used to interact with a device is also largely used in IoT devices. Although SSH provides a secure connection, it does not prevent vulnerabilities from affecting SSH server programs. We also evaluate the ability to correctly infer the SSH software name and version. However, compared to HTTP, only few software are used to run an SSH server – Section VI-E2.
- **O4: Infer the DNS software name and version**
  DNS service is the target of major DDoS attacks [3]. Attackers can leverage accessible DNS servers to act as relay in amplification attacks or, as discovered in a recent attack [8], poison the cache of DNS servers. To check for vulnerable servers, an attacker might desire to know the DNS software deployed – Section VI-E3.
- **O5 Infer the default usernames and passwords**
  In contrast to a brute-force technique, the objective here is to connect as quickly as possible with valid credentials while avoiding to be banned after too many erroneous attempts – Sections VI-F and VI-G.

VI. Evaluation

A. Dataset

Our dataset contains 4730 firmware images released between 2009 and 2019 associated to 6935 devices manufactured by one of the nine manufacturers: Asus, D-Link, Edimax, Linksys, NETGEAR, Reolink, Tp-Link, Trendnet and Ubiquiti. A device is defined by a couple <device name, device version>. Some vendors used the same firmware image for multiple devices which explains the one-to-many relationship between the firmwares and devices.

As shown in Tab. III, the distribution of devices per brand is unbalanced. Because Linksys, Reolink, Edimax account for less than 4% hence, we did not consider them in our analysis. Because our study can reveal some sensitive information, especially regarding credentials, we preferred to use the letters A-F to represent each investigated brand rather than using their real name.

B. Method and metrics

Our validation relies on the common k-fold with $k = 4$. The configuration of hyper-parameters have been done accordingly using grid-search.

Precision and recall are the most common metrics to assess a classifier. Because our classification problem is multi-class, adapted versions are used [17]:

- the macro-average computes the arithmetic mean of the metric (precision or recall) computed individually for each class (one-vs-rest);
- the micro-average measures consider the whole number of true positives, true negatives, false positives and false negatives over all classes.

With an imbalance dataset, the macro-average technique is the most drastic because the average consider each class

[7]https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication

[8]https://tsuname.io/

Objective			1	2		3		4		5		
Eval id.			1	2	3	4	5	6	7	8	9	10
Section			VI-D	VI-E1		VI-E2		VI-E3		VI-F		VI-G
Brand $brand_d$			?	•	•	•	•	•	•		•	•
Services S_d			•	•	•	•	•	•	•	•	•	•
Software list sw_d	DNS	name						?	•			
		version							?			
	FTP	name										
		version										
	HTTP	name		?	•							
		version			?							
	NTP	name										
		version										
	RPC	name										
		version										
	SNMP	name										
		version										
	SSDP/UPnP	name										
		version										
	SSH	name				?	•					
		version					?					
	TELNET	name										
		version										
Usernames u_d										?	?	
Passwords pwd_{u_d}												?
Micro-average precision (%)			76.85	89.56	94.40	98.68	80.29	99.23	73.14	88.02	92.05	88.63
Macro-average precision (%)			76.85	75.31	63.75	95.73	61.89	94.47	44.12	40.29	60.97	19.67
Micro-average recall (%)			76.85	87.28	88.81	98.68	74.54	98.95	64.23	79.13	92.50	65.71
Macro-average recall (%)			66.57	68.19	62.97	96.83	51.77	88.99	40.44	32.53	57.91	15.10
Average number of trials (%)			1.35	1.12	1.13	1.01	1.30	1.01	1.44	1.09	1.02	1.46

TABLE II: Summary of the evaluations. ? represents information to predict from partial information symbolized by •. Precision and the number of *trials* are given when applicable.

TABLE III: Brand distribution

Brand	A	B	C	D	E	F	others
Devices	758 (10.9%)	2532 (36.5%)	657 (9.5%)	1565 (22.6%)	833 (12.0%)	336 (4.8%)	254 (3.7%)

has the same weight independently of the number of samples unlike the micro-average. The latter considers the number of samples present in all classes which reflects more properly the performance of a model. Furthermore, the results of classification per brand is often given and boxplot graphs are used to catch the variability of the results among all classes in a precise way.

Additionally, some of our predictions are multi-label. Based on *scikit-learn* implementation[9], one RF estimator is built per label to predict (knowing that due to one hot encoding they are either one or zero). By default, the metrics aforementioned consider a true prediction when the predicted labels exactly match the set of right labels. In many cases, the attacker can do several trials before exploiting a device but this number should remain very low. For example, an attacker can try 2 or 3 wrong passwords before being banned. Hence, we introduced the *trial* metric which is the index of the first correct value present in the predictions set given by RF and having a prediction probability greater than zero. For instance, $trial = 1$ indicates our model returns instantly the correct value whereas $trial = 3$ shows that attackers would need three attempts before finding one valid value. This metric is computed alongside the probability of having at least one true label in the whole set of predicted labels.

[9]https://scikit-learn.org/stable/modules/multiclass.html

C. Overview of the results

Table II gives an overview of all performed experiments. To ease the reading and the understanding, each column refers to one objective (O1-O5) and a unique evaluation identifier (Eval id) is added to be used in the following sections discussing deeply the results achieved. Besides, this table indicates the features used for training (•) and the output (?) of the prediction model.

D. Identification of brands (Eval. 1)

As highlighted in Section V, our first objective is to predict the brand $brand_d$ solely from its services S_d. It is also the first level of our classification scheme.

Our classifier achieves a macro-average precision and recall of 76.85% and 66.57% respectively. Fig. 3 shows a high variability of the classification performance depending on the brand. Brands **A** and **E** underperformed with an average precision and recall lower than 0.65. They cannot thus be identified by our technique. However, brands **B**, **C** and **F** are predicted with an average precision greater than 89.6% but a recall above the 57.8%. The model does not so identify other brands as **B**, **C** or **F** but misses many samples of those brands especially for **C** and **F**. On the contrary, our classifier misses

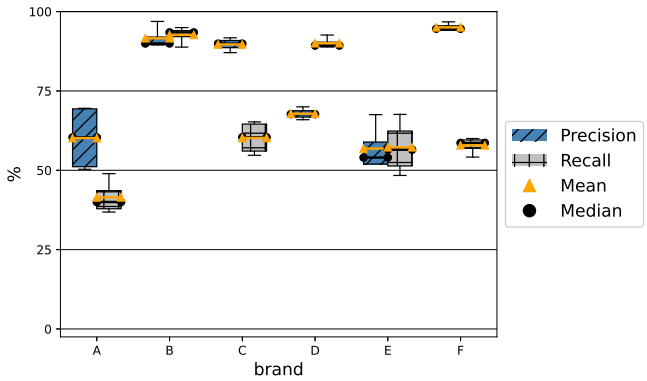


Fig. 3: Brand inference from services present

very few instances of the brand **D** (*i.e.,* average recall equal to 90.02%).

We can conclude that the set of running services is brand-specific to a certain extent and often not precise enough for the entire studied brands set. However, other options exist such as accessing the device webpage, the telnet or ssh prompt as also introduced in Section II. In the following sections, the brand is assumed to be known to train one classifier per brand.

E. Software inference

In the next experiments, our objectives are to 1) infer the software name and 2) detect its related version by considering HTTP (O2), SSH (O3) and DNS (O4).

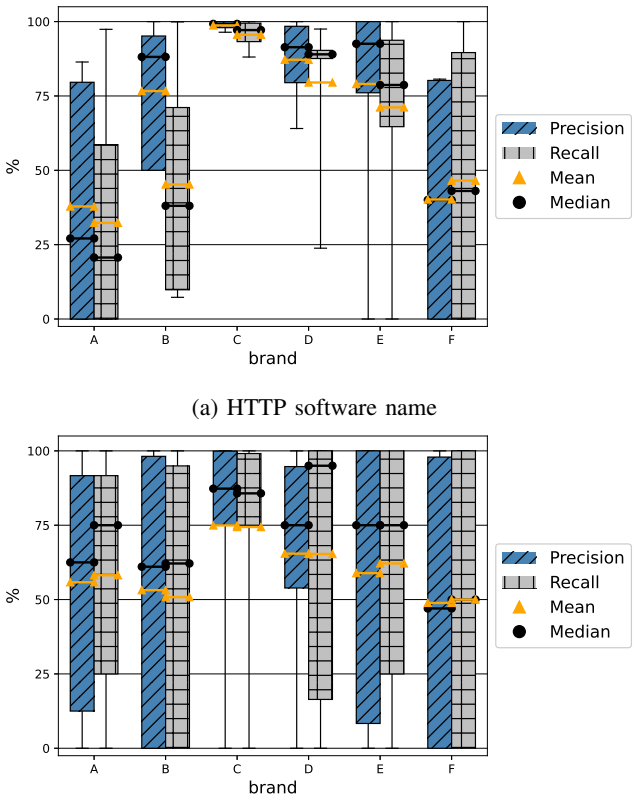
1) HTTP: Predicting the HTTP software names from the services present S_d and the brand $brand_d$ (Eval. 2) achieves a high micro-average precision and recall (89.56% and 87.28% respectively). The macro-average precision is a bit lower (75.31%). So, under-represented software (*e.g.,* shttpd, jjhttpd) are often misclassified but the identification works well for major software names (*e.g.,* lighttpd, uhttpd).

Indeed, Fig. 4a highlights that the HTTP software names of brands **C**, **D** and **E** can be fingerprinted with a high accuracy.

Considering the *trial* metric, any brand-specific classifer contains the correct HTTP software in more than 98.17% and the attacker would require only 1.16 *trials* in average to find the right version. In practice, this means that an attacker only needs to test a few malicious payloads to compromise the web server if an exploitable vulnerability exists.

Considering the correct software name found, it is added to the classification features and the inference of its version is performed in Eval 3. On one hand, the micro-average precision and recall are very high (94.4% 88.81% respectively). Thus, our classifiers correctly predict the versions with a minimum number of errors. On the other hand, the macro averages are around 60%. That is why precision and recall vary in the large range of values in Fig. 4b

It is worth mentioning that our dataset contains multiple firmwares for the same device because of updates made by manufacturers. The low macro-averages are actually due to changes in the software versions over the years but the large



(a) HTTP software name

(b) HTTP software version

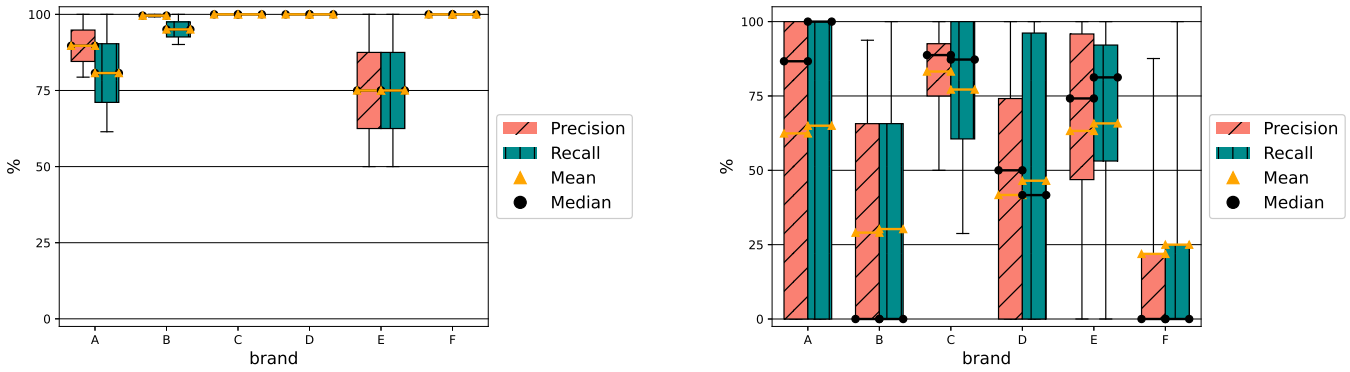
Fig. 4: HTTP software components inference

majority of them is deployed with similar versions (*e.g.,* lighttpd v1.4.39 or uhttpd 1.0.0).

We showed that attackers are able to correctly infer the HTTP software names and versions for the large majority of devices and brands. Nonetheless, the use of different versions (*i.e.,* often due to updates) for the same device has a significant impact. However, in 99.44% of cases, the right version is identified in the candidate set of RF and the average number of *trials* is 1.13.

2) SSH: The SSH software distribution is highly unbalanced with: *dropbear* (90.08%) and *openSSH* (9.92%). Fig. 5a shows that in majority of cases, the precision and recall is always high when the goal is to predict the SSH software name from the services present S_d and brand information $brand_d$ (Eval 4.). Thanks to a manual investigation, the performance of brand **A** is related to the missed predictions of *openSSH* present in two devices whereas, for brand **E**, it is due to the nearly equal distribution of *dropbear* and *openSSH*.

Considering now the SSH software name is found, the next experiment assesses the performance of predicting its version (Eval 5). The average precision and recall for all brands except **C** is below 65% as shown in Fig. 5b whereas the micro-average precision and recall is greater than 74% in table II. This phenomenom clearly reflects that the large majority of the devices use similar versions and so uncommon versions



(a) SSH software name

(b) SSH software version

Fig. 5: SSH software components inference

TABLE IV: Number of *trials* to identify SSH software version

Brand	A	B	C	D	E	F
Avg. *trials*	1.07	1.37	1.09	1.26	1.20	1.18
Std. *trials*	0.32	0.66	0.28	0.44	0.62	0.50

are concentrated on a few devices.

Regarding brand **C**, eight different versions are used in the respective devices. All versions have a precision above the 50% which indicates that the SSH software version in use depends on the services running on the device, and so indirectly on the type of the device.

By computing the *trial* metric, we observed that our classifiers return the correct prediction in more than 98.37% of the returned predictions set thus. Tab. IV indicates that attackers can find the expected SSH software version within 1.37 predictions in average allowing them to maintain a high level of stealthiness. Moreover, the standard deviation of the number of *trials* illustrates that each classifier tends to return one major version directly.

3) DNS: Our dataset contains four DNS software: *dnsmasq* (96.16%), *BusyBox's dns* (3.77%), *maradns* (0.02%) and *nsd* (0.05%).

A naive model always predicting *dnsmasq* would yield to a 96.16% precision. Nonetheless, predicting the DNS software name from the brand $brand_d$ and the running services S_d results to a micro-average precision and recall of about 99% (Eval. 6 in Table II).

Even though attacks against DNS software, notably *dnsmasq*, are often related to configuration parameters (*e.g.,* CVE-2020-25687, CVE-2017-14495) which would require us to analyze the *dnsmasq* configuration file to prove the presence or absence of vulnerability, non-configuration related attacks exist (*e.g.,* CVE-2020-25686, CVE-2017-14492). Therefore, if an attacker can correctly identify the *dnsmasq* version, it can be critical.

To predict the version of the DNS software, we assumed the services presents S_d and DNS software and brand name $brand_d$ are known (Eval. 7). Results per brand are given in Fig. 6. Because of the large number of versions (*i.e.,* up to 47),

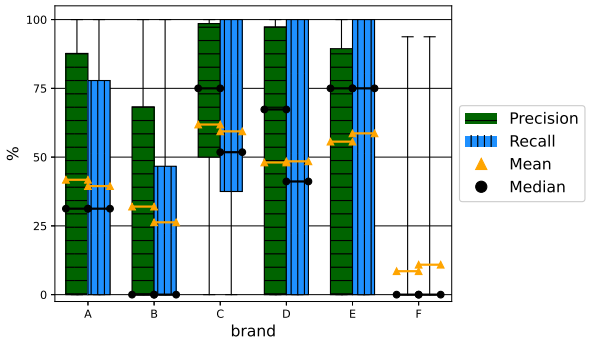


Fig. 6: DNS software version inference

TABLE V: Number of *trials* to identify DNS software version

Brand	A	B	C	D	E	F
Avg. *trials*	1.20	1.44	1.09	1.44	1.18	1.94
Std. *trials*	0.58	0.75	0.29	0.87	0.55	1.46

the precision and recall are spread out for all brands. On one hand, Brand **B** and **F** lead to the worst results with median recall and precision equal to 0. However, as highlighted in Table II, the micro-average precision and recall 73.14 and 64.23 due to two major versions (*e.g.,* 2.47 and 2.66) being correctly predicted. From a classification point of view, this could be considered as biased but in practice a device has a high chance to have one of these two versions, simplifying the attacker strategy.

On the other hand, considering the other brands (**A**, **C**, **D** and **E**), the micro-average precision and recall are both higher than 75%.

Regarding the *trials* values in Tab. V, each brand-specific classifier contains the correct version in more than 95% of its predictions set. However, the important standard deviation, especially for brand **B**, **D** and **F** confirm the difficulty of our classifier to find uncommon versions. Nonetheless, on average, the correction version is returned within the first 1.38 predictions.

F. Usernames inference

To compromise devices, attackers might try to access through default passwords and usernames, especially when there is no vulnerability to exploit. In the following experiments, we are particularly interested to know if an attacker can guess a valid couple <username, password> with a very few attempts.

We predicted the usernames u_d from the services S_d without (Eval 8.) or with (Eval. 9) the brand information $brand_d$. Overall, at least one valid username is found in more than 99.78% of the returned predictions set, and is returned within the 1.02 *trials* if $brand_d$ is known or 1.09 otherwise. Furthermore, naively predicting that the *root* username is usable in all the devices would not suffice because it is only present in 47.67% of our studied devices. So our technique is more reliable and most of the time one valid username will be found in a single *trial*.

G. Password inference

Obviously, a login process requires a couple <username, password> so, this multi-label classification (Eval 10.) returns the predicted passwords set pwd_{ud} related to the users u_d present in a device thus, from an attacker perspective, it is not necessary to correctly predict the entire set of passwords which explains why we used the *trial* metric in this section.

As stated in IV-A, the extracted passwords may be encrypted meaning that attackers may not directly log into the device. However, breaking it can be done offline if a weak hash algorithm was used but still requires time hence, it is important to identify at least one correct password as earlier as possible.

We presented in Tab. VI, the *trial* results per brand for the *root* and the users u_d passwords. Firstly, we noticed that brand **F** has no useable *root* passwords so, no values were computed. Similarly, brand **B** classifier contains the correct *root* password in only 55.06% of the returned prediction sets which is explained by the use of another username instead of *root*. Furthermore, comparing the results of *root* and the usernames u_d shows that brands **A** and **E** also use an additional user to handle the role of administrator thus, in the case of **B** the username is brand-specific.

Considering the passwords associated to the usernames u_d investigated in Section VI-F, the results in Tab. VI highlight that at least one usable password can be inferred in all brands in a reliable manner (in worst case with an average *trial* value of 1.28).

Our technique showed that the use of a well documented credentials dictionary may suffice to compromise a large number of devices hence, it is not surprising to see botnets such as Mirai [21] using this strategy.

TABLE VI: Number of *trials* to identify usernames u_d and *root* passwords

Brand	username(s)	Password found	Avg. *trial*	Std. *trial*
A	$root$	73.12	1.19	0.52
B	$root$	55.06	1.05	0.21
C	$root$	88.12	1.17	0.39
D	$root$	86.85	1.08	0.31
E	$root$	75.32	1.23	0.66
F	$root$	N.A.	N.A.	N.A.
A	all(u_d)	97.67	1.28	0.81
B	all(u_d)	99.19	1.03	0.19
C	all(u_d)	98.89	1.24	0.60
D	all(u_d)	99.73	1.10	0.48
E	all(u_d)	99.27	1.27	0.79
F	all(u_d)	97.05	1.00	0.00

VII. Discussion

Based on the described results and the summary of the experiments in Table II, the accuracy of our fingerprinting technique mainly depends on the brand of the device. From a pure data analysis point of view, the results are mixed. However, the context of our classification approach is attacker-driven, meaning that having false positives or negatives in our prediction results is not critical because attackers are smart and will focus on the devices that can be easily distinguished. Reminding that the first step is to scan thousands or even millions hosts in the Internet, fingerprinting does not need to be accurate over all of them. Similarly, attackers are more effective at testing a few default passwords on millions of devices rather than brute-forcing complex passwords on thousands of them. Moreover, an attacker could do several attempts without being suspected or blacklisted until the number of trials is low. This is actually the case for all our experiments. This highly differs from common classification use-cases where the first trial must be accurate, for example to detect the anomalies. As a summary, our results unfortunately show that an attacker can easily retrieve valid information from open services/ports even if protocol banners are obfuscated. However, running the services on dynamic port numbers or faking the presence of services on well known ports could be used as countermeasures to our technique because of the noise added to the TCP/UDP ports scan.

VIII. Conclusion

This paper introduced a fingerprinting technique dedicated to embedded devices by training a Random forest model from data extracted from a large-scale firmware analysis on 6935 embedded devices between 2009 and 2019. We assessed multiple scenarios where attackers search embedded devices through the Internet via TCP/UDP scans to infer precise information such as the expected HTTP or SSH software versions to then, perform specific attacks. Our objective was to give a comprehensive overview of information which can be easily recovered from TCP/UDP scans without actually trying to establish real connections to grab the service banners.

Overall, information about software infrequently used are logically harder to predict but in general an attacker will be able to reconstruct the correct information after a very few number of attempts and possibly selecting rapidly the right malicious payload or even a correct couple <username, password> to use. Therefore, relying on obfuscation by removing information from banners seems obsolete and a

possible defeating technique would be to open random ports (without a real service). Highlighted in the recent botnets [21] attacks, our experiments inferring usernames and passwords confirmed that brute-force techniques based on a well formed dictionary can indeed unlock the large majority of the devices.

In our experimentations, all the programs present and able to run a service are considered active by default, that might not reflect the reality. In future work, we plan to improve our firmware analysis to identify the services started during the device boot sequence. With more precise information, the classification accuracy would be higher. Also, we will compare the performance of our model to an Internet-wide scan database such as Censys or Shodan.

Acknowledgments This work has been partially supported by the project SPARTA, funded from the European Union's Horizon 2020 industrial leadership programme under grant agreement no. 830892.

References

[1] "Cyber threat intelligence report," available at https://www.netscout.com/threatreport.

[2] "Alert (ta14-017a) - udp-based amplification attacks," US Department of Homeland Security CISA, Tech. Rep., 12 2019, available at https://us-cert.cisa.gov/ncas/alerts/TA14-017A.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium*, 2017.

[4] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[5] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of iot devices," in *Workshop on Attacks and Solutions in Hardware Security (ASHES)*. ACM, 2018.

[6] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, "Active behavioral fingerprinting of wireless devices," in *Proceedings of the First ACM Conference on Wireless Network Security*, ser. WiSec '08. New York, NY, USA: ACM, 2008, pp. 56–61. [Online]. Available: http://doi.acm.org/10.1145/1352533.1352543

[7] P.-O. Brissaud, J. François, I. Chrisment, T. Cholez, and O. Bettan, "Transparent and Service-Agnostic Monitoring of Encrypted Web Traffic," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 842–856, Sep. 2019. [Online]. Available: https://hal.inria.fr/hal-02316644

[8] D. Chen, M. Egele, M. Woo, and D. Brumley, "Towards automated dynamic analysis for linux-based embedded firmware," 01 2016.

[9] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 95–110.

[10] A. Costin, A. Zarras, and A. Francillon, "Automated dynamic firmware analysis at scale: a case study on embedded web interfaces," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 437–448.

[11] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2015.

[12] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *Security Symposium*. Washington, D.C.: USENIX, 2013.

[13] D. Evans, "The internet of things, how the next evolution of the internet is changing everything," *Cisco white paper*, Apr 2011, available at https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[14] J. François, A. Lahmadi, V. Giannini, D. Cupif, F. Beck, and B. Wallrich, "Optimizing Internet Scanning for Assessing Industrial Systems Exposure," in *International Workshop on TRaffic Analysis and Characterization (TRAC)*, Paphos, Cyprus, 2016.

[15] D. Fraunholz and H. D. Schotten, "Defending web servers with feints, distraction and obfuscation," in *International Conference on Computing, Networking and Communications (ICNC)*, 2018.

[16] V. Ghiette, H. Griffioen, and C. Doerr, "Fingerprinting tooling used for SSH compromisation attempts," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. Chaoyang District, Beijing: USENIX Association, Sep. 2019, pp. 61–71. [Online]. Available: https://www.usenix.org/conference/raid2019/presentation/ghiette

[17] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," 2020.

[18] P.-M. Junges, J. François, and O. Festor, "Software-based Analysis of the Security By Design in Embedded Devices," in *IFIP/IEEE International Symposium on Integrated Network Management*, Bordeaux, France, 2021.

[19] P.-M. Junges, J. Franois, and O. Festor, "Passive inference of user actions through iot gateway encrypted traffic analysis," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019.

[20] H. Kim, T. Kim, and D. Jang, "An intelligent improvement of internet-wide scan engine for fast discovery of vulnerable iot devices," *Symmetry*, vol. 10, no. 5, 2018.

[21] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the iot: Mirai and other botnets," vol. 50, pp. 80–84, 01 2017.

[22] M. Kse, S. Taciolu, and Z. Telatar, "Rf fingerprinting of iot devices based on transient energy spectrum," *IEEE Access*, vol. 7, 2019.

[23] S. Lagraa and J. Francois, "Knowledge Discovery of Port Scans from Darknet," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM) - AnNet workshop*, Lisbonne, Portugal, May 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01636215

[24] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Audi: Toward autonomous iot device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[25] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017.

[26] N. Msadek, R. Soua, and T. Engel, "Iot device fingerprinting: Machine learning based encrypted traffic analysis," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2019.

[27] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[28] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A multi-level framework to identify https services," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 240–248.

[29] Y. Shoshitaishvili, R. Wang, C. Hauser, C. Kruegel, and G. Vigna, "Firmalice - automatic detection of authentication bypass vulnerabilities in binary firmware," in *Network and Distributed System Security Symposium (NDSS)*, 2015.

[30] Y. Song, Q. Huang, J. Yang, M. Fan, A. Hu, and Y. Jiang, "Iot device fingerprinting for relieving pressure in the access control," in *Proceedings of the ACM Turing Celebration Conference - China*, ser. ACM TURC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3321408.3326671

[31] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "Deft: A distributed iot fingerprinting technique," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 940–952, 2019.

[32] V. Visoottiviseth, P. Jutadhammakorn, N. Pongchanchai, and P. Kosolyudhthasarn, "Firmaster: Analysis tool for home router firmware," in *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2018, pp. 1–6.

[33] K. Yang, Q. Li, and L. Sun, "Towards automatic fingerprinting of iot devices in the cyberspace," *Computer Networks*, vol. 148, pp. 318 – 327, 2019.