Passive Inference of User Actions through IoT Gateway Encrypted Traffic Analysis

Pierre-Marie Junges, Jerome Francois, Olivier Festor

Passive Inference of User Actions through IoT Gateway Encrypted Traffic Analysis

Pierre-Marie Junges, Jérôme François, Olivier Festor
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
firstname.lastname@inria.fr

***Abstract*—Internet of Things (IoT) devices become widely used and their control is often provided through a cloud-based web service that interacts with an IoT gateway, in particular for individual users and home automation.**

In this paper, we propose a technique to infer private user information, i.e., actions performed, by considering a vantage point outside the end-user local IoT network. By learning the relationships between the user actions and the traffic sent by the web service to the gateway, we have been able to establish elementary signatures, one for each possible action, which can be then composed to discover compound actions in encrypted traffic.

We evaluated the efficiency of our approach on one IoT gateway interacting with up to 16 IoT devices and showed that a passive attacker can infer user activities with an accuracy above 90%.

***Index Terms*—Internet of Things, fingerprinting, encrypted traffic, security, privacy**

I. Introduction

With the emergence of the Internet of Things (IoT), the use of heterogeneous IoT devices becomes widespread. However, many of them suffer from security issues including the lack of updates or the use of default credentials. As a result, IoT devices are now targets for attackers, and compromised IoT devices can led to the creation of major botnets like Mirai [1] or BrickerBot [2]. In addition to these security concerns, IoT devices in smart homes also present a risk of user privacy leakage [3, 4, 5, 6].

Analyzing the IoT traffic is of paramount importance to evaluate the level of private data a malicious user can infer or to profile malicious actions such as attacks that is now mixed within the IoT traffic. We thus propose a traffic analysis technique dedicated to IoT gateways, more precisely by observing the Internet traffic of the IoT gateway, which interacts with a cloud-based web service. Such a case neither assumes to be able to observe IoT device communications themselves, and so to be in their close vicinity, nor supposes to eavesdrop the end-user commands.

Considering a vantage point outside the end-user network, our objective is to evaluate the actions performed by the end-users that are indirectly exposed by an IoT gateway. Therefore, it may support the privacy assessment of an IoT deployment but also anomaly detection. Indeed, creating normal profiles for user actions can be then leveraged to detect anomalies [6].

Our proposed technique is able to gather information about the actions performed by the user. Due to the use of proprietary and/or encrypted channel, it relies on decomposing the size of encrypted application data, monitored between the IoT gateway and the web service, in elementary sizes representing individual contents, i.e., the user actions. We evaluated our proposed solution on one off-the-shelf IoT gateway and demonstrated that actions performed by a user on the IoT devices can be passively inferred.

The paper is structured as follows. Section II introduces the related work on IoT fingerprinting and network traffic analysis. Section III defines the targeted problem. Section IV describes our proposed technique while Section V presents its evaluation. Section VI concludes our paper.

II. Related works

For security purposes, researchers proposed several fingerprinting solutions to identify IoT devices in a network. A device type identification technique for IoT IP-based devices using header values (e.g., IP addresses, port numbers, protocol) and a random forest classifier to identify an IoT device from a signature is presented in [7]. In [8], a self-learning device type identification technique uses period-related features (e.g., period duration, number of periodic flows) computed from the packet flows. A behavioral fingerprinting based on header and payload based features (e.g., TCP payload length, network protocol observed) is proposed in [9].

Our proposed solution is focused on identifying the actions performed on the IoT devices but some actions can be representative of the use of particular devices. In addition we exclusively rely on the traffic between the IoT gateway and a web service because we considered the IoT devices to be neither accessible nor visible from our vantage point.

IoT devices in smart homes may also lead to user privacy leakage [3, 4, 5, 6]. In [3], from the network traffic generated by Bluetooth Low Energy (BLE) weareable fitness trackers, it is possible to identify a person and its activity. In [4], the authors demonstrate that the network traffic rate from WiFi devices can reveal user activities. Similar inferences have been noticed in [5] using timestamps from wireless X10 motion and contact sensors. In [6], the investigated IoT devices used different wireless protocols (i.e., Wi-Fi, ZigBee and BLE) and a multi-stage privacy attack is able to identify the actions and the states of the IoT devices present in a end-user local network.

Again, our work mainly differs by considering exclusively external traffic between the IoT gateway and a web service.

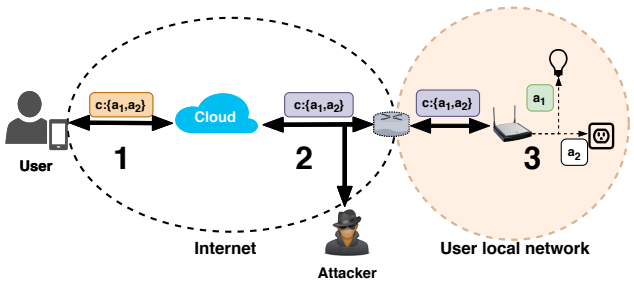


Fig. 1: Attacker model considered in this paper

However, such traffic is often encrypted and so limits the exposure of the IoT devices and their related activities.

From that perspective, a passive fingerprinting technique using clustering algorithms and the sizes of the first few packets of an SSL connection can recognize the corresponding web-application [10]. Similarly in [11, 12], identities of accessed web pages can be also recovered from packet or payload sizes within the encrypted HTTP connections. Fine-grained profiling of user activities is possible by reconstructing the sizes of loaded objects [13].

Finally, to reduce the lack of privacy protection in the IoT world, countermeasures using blockchain [14] or privacy-by-design frameworks [15] have been proposed.

III. Problem definition

Fig. 1 shows an example of the IoT system our work focuses on. The user connects to a mobile application, (1) requests a command c containing the actions a_1 and a_2 to be executed on two IoT devices, (2) the web service sends c to the IoT gateway over an encrypted communication channel and (3) the latter transmits a_1 and a_2 to the intended IoT devices using a wireless protocol (that can be proprietary and/or IoT specific and/or encrypted). Our technique aims at decomposing the encrypted application data, observed during step (2), to deduce information about the IoT devices accessed and related requests during step (3).

A. Motivation

Considering a vantage point within the end-user IoT network, inferring user activities [6] may be relatively straightforward because the network traffic of each individual IoT device is observable. However, this forces the attacker to be in a close vicinity which thus limits the practicability of the attack.

The presence of IoT gateways in the end-user local network makes the user privacy assessment harder because the gateways receive, from a web service, commands that may concern multiple heterogeneous IoT devices at the same time (see Fig. 1) that cannot thus be directly monitored.

However, by communicating with a web service through the Internet, the network traffic of the IoT gateway, often encrypted using secure protocols, can be observed. In this work, we evaluate the level of private user information (mainly user actions requested) exposed by an IoT gateway on the Internet.

B. Challenges and assumptions

Considering our point of observation, our approach raises some challenges:

- **C1 - No individual IoT device signature.** IoT devices may be requested by the users to perform multiple actions and the number of IoT devices might be large. Indeed, assuming the user has m IoT devices with n possible actions for each, the user may request to perform any of these n actions on up to m IoT devices. For example this leads to 19607 combinations, with only $m = 5$ and $n = 7$. As a consequence, it is not possible to learn the traffic generated by every individual combination.
- **C2 - Gateway abstraction.** The IoT gateway receives and processes generic actions. Indeed, even though IoT devices might be completely different (e.g., protocols, brands, models), the IoT gateway receives actions from a single control channel by the web service.
- **C3 - Encryption.** IoT gateway network traffic is encrypted (often using SSL/TLS), so extracting original content from application data is impossible.

Based on preliminary studies and related work described in section II, we make the following assumptions:

- **A1 - Sending actions to the IoT devices.** When the user performs multiple actions on multiple IoT devices *in one command*, we assume these actions are merged into one actions list c.
- **A2 - Incidence of the actions on the packet size.** The larger the list c sent from the user to the web service, the larger the corresponding application data sent from the web service to the IoT gateway. So, actions performed on IoT devices have an incidence on the application data observed (i.e., on the size), even if encrypted.
- **A3 - Command size stability.** When the same action is performed multiple times, its payload size does not change significantly.
- **A4 - Data structures similarity.** In Fig. 1, we can expect some similarities, in their content (actions provided), between the data sent by the web service to the IoT gateway during step (2) and the original data sent by the user during step (1). Such a content would thus impact on the size of the encrypted payload.

IV. Indirect knowledge extraction

Identifying IoT gateway in a network is not the focus of this paper but the reader can refer to techniques from related works [7, 8]. Here, we consider as known the IP address of the IoT gateway.

Our approach follows three main steps to learn the signatures of individual actions:

- From known user actions, extract relevant features from the corresponding network packets sent by the web service to the IoT gateway.
- Signature construction using the features previously extracted.

- Learning of possible variations between our signatures and the observed encrypted application data sizes.

Once learning achieved, user actions can be identified (testing).

Each of these steps is described in details in next subsections.

A. Features extraction

Network traffic is often encrypted (**C3**), so few features are relevant. However, in regards of assumptions **A1**, **A2** and **A3**, we suppose that each individual action contributes to change the application data size. Thanks to **A4**, knowing the data sent by the user to the web service, we can deduce what should contain the data sent by the web service to the IoT gateway. Hence, the main feature used is the encrypted application data size.

Rather than learning all possible combinations of actions (**C1**), we will learn individual encrypted sizes (for each action) that can be composed together, assuming an acceptable approximation error, which has also to be learnt.

The size of each possible action $a_1, a_2, ..., a_n$ can be derived using the following steps: (1) perform the action a_i on one IoT device with the user application, (2) extract the data d_i sent to the web service, (3) find the corresponding packets s_i sent by the cloud-based web service to the IoT gateway and (4) repeat the operation using two IoT devices to get another data structure d_{2i} with its corresponding packets s_{2i}. The rationale behind this process is the existence of additional content (such as timestamps), ac, in the message that is not dependent on the actions or their number. Using two iterations with one and two devices will thus allow to isolate the part solely related to the actions.

In equations (1) and (2), both data structures d_i and d_{2i} are composed of the descriptions of the actions a_i and the additional content ac and we initially consider the size of ac to be constant.

In equation (2), a_i is present two times because d_{2i} is the data structure sent when we requested a_i to be performed on two IoT devices.

$$d_i =< a_i, ac > \tag{1}$$

$$d_{2i} =< a_i, a_i, ac > \tag{2}$$

The corresponding packets are s_i and s_{2i} respectively.

Assuming $|s_i|$ (resp. $|s_{2i}|$), the encrypted application data size of s_i (resp. s_{2i}). Then, it is possible to compute the size of the action a_i (i.e., $|a_i|$):

$$|a_i| = |s_{2i}| - |s_i| \tag{3}$$

because $d_{2i} - d_i =< a_i, a_i, ac > - < a_i, ac >=< a_i >$

The size of the additional content ac (i.e., $|ac|$) is derived as follows:

$$|ac| = |s_i| - |a_i| \tag{4}$$

because $d_i - < a_i >=< a_i, ac > - < a_i >=< ac >$

Our final set of features is composed of $|ac|$ and $|a_i|_{1 \leq i \leq n}$ computed from equations (3) and (4).

B. Learning of the signatures

Once all $|a_i|$ are computed, any size $|s|$ from encrypted application data sent by the web service to an IoT gateway can be rewritten as in equation (5), with $|ac|$ the additional content size, $|a_i|$ the size of the action a_i, $nb\_a_i \in \mathbb{N}$ the number of occurrences of a_i in *s* and $\epsilon \in \mathbb{Z}$, a variation value.

$$|s| = |ac| + \epsilon + \sum_{i=1}^{n} |a_i| \times nb\_a_i \tag{5}$$

C. Learning the variations between the theoretical and observed sizes

We introduced ϵ in (5) because we consider that the encrypted application data size observed may not be exactly equal to the one we can compute using previously inferred $|a_c|$ and $|a_i|_{1 \leq i \leq n}$. The ϵ embeds so both variations of the actions or the additional content.

Hence, ϵ is simply the expected difference between these two sizes (observed and computed by composing $|a_c|$ and $|a_i|_{1 \leq i \leq n}$).

To automatically learn this value, we can control the user (i.e., perform actions) by requesting m different combinations of actions $A_j = \{nb\_a_1, ..., nb\_a_n\}$ with $j = 1...m$ to be performed and retrieved their corresponding encrypted payload size $|s_j|$. Therefore, each ϵ_j can be derived using (5) and a learning dataset is so built with the m tuples $< |s_j|, \epsilon_j >$. As a result, for each new observed encrypted size $|s_j|$ (when we do not control the actions), we search for the closest size in this dataset to deduce the related ϵ_j, *i.e.* k-Nearest Neighbors classifier (kNN) in one dimension.

D. User action identification

Assuming now the user performs a new command A with $A = \{nb\_a_1, ..., nb\_a_n\}$. The objective is so to automatically infer the value of each nb_a_i with $i = 1...n$, knowing only the global encrypted payload size $|s_c|$ received by the IoT gateway.

Firstly, using the classifier previously trained (kNN), ϵ_c is assigned from $|s_c|$ and equation (5) can be rewritten as follows:

$$|s_c| - |ac| - \epsilon_c = \sum_{i=1}^{n} |a_i| \times nb\_a_i \tag{6}$$

While the left part can be fully computed and considering each $nb\_a_i \in \mathbb{N}$, our problem is similar to the change-making problem [16] and we can use its dynamic programming algorithm to solve equation (6). Finally, a combinations set $C = \{A_1, ..., A_m\}$ such as each $A_j = \{nb\_a_{j1}, ..., nb\_a_{jn}\}$ with $j = 1...m$ satisfies equation (6) is returned. Each combination is a candidate result, so our technique does not guarantee to return a unique set of actions. In the evaluation, we will particularly evaluate how each set is close to the real actions requested and aims at reducing as most as possible the number of sets.

TABLE I: Actions Per Device

Device	Actions
Gateway	None
Smart plug	On, Off, On {1,...,10} mins
Smart lamp holder	On, Off, On {1,...,10} mins, P, P {1,...,10} mins

V. Evaluation

A. Setup

Our evaluation setup is composed of one IoT gateway *G* from a french home automation manufacturer *M* whose the name cannot be disclosed, controlling 12 smart plugs and four smart lamp holders. All the available actions per device are listed in Tab. I. We noted *P*, a personalized pre-configured action by the user (e.g., 50% light intensity) and actions *P* are solely performed by lamp holders.

On one hand, by predicting the actions performed, our method can also deduce the type of IoT devices in certain cases, for instance predicting only *P* actions is representative of a lamp holder. On the other hand, as showed in Tab. I, some actions (e.g., On and Off) are common to both investigated device types.

M proposes a web service (*WS*), among others, to let the user control the IoT devices. We investigated two functionalities offered by the *WS*:

- *Single action*: execute the same action on one or multiple devices of the same type;
- *Scenarios*: perform distinct actions on one or multiple devices of any types

Scenarios are named and created by the user. Once the *scenarios* created, the user is able to execute them.

B. Methodology

We used Mozilla Firefox version 64.0 on a 16.04 Ubuntu, to connect to *WS* and its Network Monitor tool retrieve data prior to encryption for labeling traces. In our experiment setup, we used port mirorring to capture the web service-gateway traffic. In practice, it can be done by an intermediate player such as a network operator or by an attacker relying on a man-in-the-middle attack.

We performed different actions using both investigated functionalities and discovered the identical starting sequence when the gateway receives actions from *WS*. Firstly, *WS* sends one UDP packet with the unencrypted keyword *OPEN* to request the gateway to initiate the TLS 1.2 session during which *WS* transmits the actions requested to be performed. Hence, the user actions to identify are actually encrypted as expected. Shortly after, *G* initiates multiple TLS 1.2 sessions with the web service. These following sessions will be used in section V-C to identify the number of IoT devices, which have performed an action.

C. Reduction

In general, the user can also monitor in real-time its IoT devices status (e.g., actions log, sensor triggered) using the web service. However, the IoT end-devices are not directly connected to Internet and this information is thus provided through the IoT gateway. After receiving the actions intended for nb_{IoT} devices, *G* initiates nb_s new TLS 1.2 sessions with *WS*, those ones are actually extra information that will allow us to reduce the number of solutions (i.e., C). We noticed that the new opened TLS sessions are generally close to each other in terms of inter-arrival time (i.e., time between two Client Hello packets). We assume they represent the feedback of the IoT devices in regards to the user request. From empirical observation, we inferred in (7), a relationship between nb_s, the number of TLS sessions following the reception of a request, and nb_{IoT} the number of IoT devices concerned by this request.

$$nb_{IoT} = nb_s - 1 \tag{7}$$

Due to the encryption and the proprietary protocol, we cannot confirm but assume that one session (feedback) is created per IoT device while we discard the one corresponding to the global request itself. To solve equation (7), nb_s is the number of TLS sessions that are within 2.5 seconds to each other. The cardinality of the combinations set C can thus be reduced by removing the combinations A_j, with $A_j = \{nb\_a_{j1}, ..., nb\_a_{jn}\}$ with a different number of devices, *i.e.* $\sum_{i=1}^{n} nb\_a_{ji} \neq nb_{IoT}$). The reduced combinations set is denoted as C'.

D. Learning sizes

We applied our proposed solution (section IV-A) to deduce the encrypted sizes of the actions (i.e., all $|a_i|$). We thus execute all different actions using the WS, collect the generated traffic between the web service and the gateway and learn those sizes. They are reported in Table II. As introduced in section IV-B, an additional content that is encrypted as well can be derived (i.e., $|a_c| = 216$ bytes).

E. Identifying actions

Using the four lamp holders and the 12 smart plugs, 307 different combinations of actions have been performed represented by $A_j, 1 \leq j \leq 307$ with $A_j = \{nb\_a_{j1}, ..., nb\_a_{j7}\}$. Actually, the number of devices used varies between 1 and 16, and for each case, 20 random combinations have been generated except with a single device as only seven distinct actions can be achieved in that case. That is why we obtained 307 different combinations to be tested.

Fig. 2 shows the mean number of each device type. As observed, our random generation of actions follows the distribution of usable devices: four lamp holders and 12 smart plugs. Hence, with height devices, six actions are performed on smart plugs and two on lamp holders on average.

Equation (6) is refined according to variables previously learned in Section V-D:

$$\begin{aligned} |s| = \epsilon + 216 + 221 \times nb\_a_1 + 238 \times nb\_a_2 \\ +240 \times nb\_a_3 + 489 \times nb\_a_4 + 490 \times nb\_a_5 \\ +491 \times nb\_a_6 + 492 \times nb\_a_7 \end{aligned} \tag{8}$$

TABLE II: Actions Sizes And Device Types

Device type	Action	Size (bytes)
lamp holder	P	$\|a_1\| = 221$
smart plug and lamp holder	On	$\|a_2\| = 238$
smart plug and lamp holder	Off	$\|a_3\| = 240$
smart plug and lamp holder	On 1 min	$\|a_4\| = 489$
smart plug and lamp holder	On {2,...,10} mins	$\|a_5\| = 490$
lamp holder	P 1 min	$\|a_6\| = 491$
lamp holder	P {2,...,10} mins	$\|a_7\| = 492$

To solve (8) and so determine the values nb_a_i, we evaluated the variation ϵ from the total encrypted size $|s|$ using kNN as explained in section IV-B. In our experiment, kNN is configured with $k = 2$. Using our dataset, a four-fold cross validation is applied to determine ϵ.

Assuming the real action and inferred combination sets $A = \{nb\_a_1, \ldots, nb\_a_7\}$ and $C = \{A_1, ..., A_m\}$ respectively, we introduce several metrics to consider different perspectives of the results assuming either when the reduction step is used, $A \in C'$, or not, $A \in C$:

- $Pred\_nb$ is the ratio of tests whose the predicted number of devices nb_{IoT} corresponds to the real one, i.e., $nb_{IoT} = \sum_{i=1}^{7} nb\_a_i$
- $P(A \in C')$ or $P(A \in C)$ assess if the real set of actions that have been requested has been actually identified as a candidate decomposition after equation solving with or without the reduction step respectively.
- $card(C)$ and $card(C')$: the previous metric measures the ability of our technique to find a set of decompositions, C or C', that contains the real actions set but it has to be compared to the total number of decompositions, $|C|$ or $|C'|$, as the goal is to have those sets as smaller as possible to keep as lower as possible the uncertainty about the right decomposition.
- $P(A \in C'|Pred\_nb)$ measures the probability to identify the right decomposition A (always among others) given that the number of devices has been correctly determined.

Tab. III summarizes the obtained results, while using 4-fold cross validation as previously mentioned. As observed, finding the performed actions A in the reduced set C' (91.2%) is more effective in case the right number of devices has been identified (99.2%). At most, we can find A with a precision of 98.4% which also means that ϵ has been correctly assign to 98.4% of our 307 combinations. When using our reduction technique, the precision decreases to 91.2%. This is a side effect because the reduction limits the selected decompositions in C', including possibly the real one. However the main benefit is to have a lower cardinality, $|C'| < |C|$.

More precisely, in Fig. 3a, the average cardinality of C and C' are shown. C can contain up to 813 combinations whereas C' has a maximum of only 107 combinations. The reduction efficiency is measured at 79.15% and we observed, on average, eight combinations in C' instead of 36 in C. For example, with 10 devices we obtain, on average, seven combinations in C' and 24 in C. Now, assuming these 10 devices were smart plugs and without using the proposed technique, we would have to learn the traffic signature of 1048576 combinations.

TABLE III: Overall evaluation results

Events	Precision
P($A \in C$)	98.4
P($A \in C'$)	91.2
P($Pred\_nb$)	91.8
P($A \in C'|Pred\_nb$)	99.2

Although those metrics provide different view of obtained results, they clearly show that identifying automatically a unique and exact set of devices is rather difficult. Hence, a last metric is introduced in equation (9) to assess the similarity between an inferred combination and the real one.

$$dist(p,q) = \frac{|\sum_{p_i \in p} p_i - \sum_{q_i \in q} q_i| + \sum_{i=1}^{n} |p_i - q_i|}{2} \quad (9)$$

$dist(p, q)$ corresponds to the mean between the differences in number of actions (real and predicted) in total and the individual ones, i.e., for each nb_a_i.

For instance, considering the combinations $p = \{1, 0, 1\}$, the performed one, and $q = \{0, 1, 1\}$. We can notice that q has one misplaced action leading to $dist(p, q) = 1$. Assuming now $p = \{1, 1, 1\}$ and $q = \{3, 2, 1\}$, there are three misplaced actions in q, two for the first action and one for the second so, $dist(p, q) = 3$.

In Fig. 3b, the combinations in C' are generally closer to real ones compared to the ones in C. For instance, assuming 11 devices, the average distance is around five and seven with C' and C respectively. In that case, the combinations in C' have six correctly identified actions against four with C. We also observe that the larger the number of actions performed, the larger the average distance. This problem is mainly due to similar sizes between some a_i as shown in Table II.

VI. Conclusion

In this paper, we introduced a method to automatically infer the actions requested by a user by solely observing the resulting traffic sent from the web service to the IoT gateway. Our approach consists in finding a correlation between the user inputs (actions performed) and the observed encrypted application data sizes received by the IoT gateways. Our evaluation highlights the ability of our technique to predict a set of action lists that contain the real actions with a high accuracy (up to 98.4%) with a limited incertitude compared to all possible combinations of actions.

Hence, even though the network traffic is encrypted and the IoT end-devices not directly observable, the presence of an IoT gateway does not prevent an intermediate entity to retrieve fine-grained information about the user activities.

In future work, such an information will be leveraged to create normal activity profiles and detect deviations afterwards in order to help in detecting anomalies and attacks.

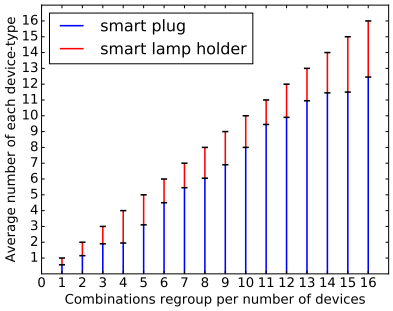
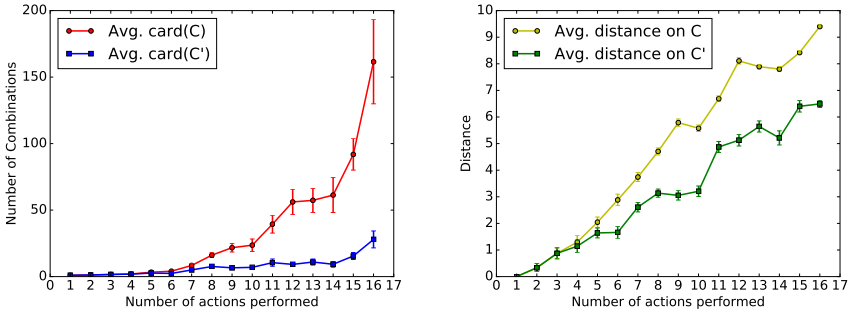


Fig. 2: Device distribution in the dataset



(a) Avg. number of combinations found in C and C' per number of actions performed. Confidence intervals set to 68%

(b) Avg. distance metric computed in C and C' per number of actions performed. Confidence intervals set to 68%

Fig. 3: Evaluation of the reduction step

Acknowledgments This work has been partially supported by the project SecureIoT, funded from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 779899.

References

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

[2] Radware, "Brickerbot results in pdos attack," 2017, available at https://security.radware.com/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service.

[3] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Uncovering privacy leakage in ble network traffic of wearable fitness trackers," in *17th International Workshop on Mobile Computing Systems and Applications (HotMobile)*. ACM, 2016.

[4] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic," *CoRR*, vol. abs/1708.05044, 2017. [Online]. Available: http://arxiv.org/abs/1708.05044

[5] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Protecting your daily in-home activity information from a wireless snooping attack," in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 202–211.

[6] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and A. S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" 2018.

[7] 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.

[8] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, and A. Sadeghi, "Dïot: A crowdsourced self-learning approach for detecting compromised iot devices," *CoRR*, vol. abs/1804.07474, 2018. [Online]. Available: http://arxiv.org/abs/1804.07474

[9] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of iot devices," *CoRR*, vol. abs/1804.03852, 2018. [Online]. Available: http://arxiv.org/abs/1804.03852

[10] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *Passive and Active Network Measurement (PAM)*. Springer, 2007.

[11] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections," in *Conference on Computer and Communications Security (CCS)*. ACM, 2006.

[12] W. M. Shbair, T. Cholez, J. François, and I. Chrisment, "A Multi-Level Framework to Identify HTTPS Services," in *Network Operations and Management Symposium*. Istanbul, Turkey: IEEE/IFIP, 2016.

[13] P.-O. Brissaud, J. Francois, I. Chrisment, T. Cholez, and O. Bettan, "Passive Monitoring of HTTPS Service Use," in *14th International Conference on Network and Service Management (CNSM)*, Rome, Italy, 2018.

[14] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017.

[15] C. Perera, C. McCormick, A. K. Bandara, B. A. Price, and B. Nuseibeh, "Privacy-by-design framework for assessing internet of things applications and platforms," in *6th International Conference on the Internet of Things (IoT)*. ACM, 2016.

[16] J. W. Wright, "The change-making problem," *J. ACM*, vol. 22, no. 1, pp. 125–128, Jan. 1975. [Online]. Available: http://doi.acm.org/10.1145/321864.321874