BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2020
http://excel.fit.vutbr.cz

Behavior-Based Network Device Tracking

Michael Adam Polák*

Abstract
With the constantly growing number of devices on private and corporate networks, it is becoming increasingly more important for network administrators to track devices based on their behavior with limited feature availability due to the increasing security risks. This paper analyzes methods used to create device profiles that are subsequently used to identify devices using frequency analysis and the k-Nearest Neighbors algorithm with cosine similarity as the distance metric. Lastly, the results of this method are presented with possible improvements to the existing algorithm.

Keywords: Network Device Identification — Frequency Analysis — k-Nearest Neighbors

Supplementary Material: *N/A*

*xpolak31@fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

With the constantly growing number of devices that are connected to the network, it is becoming increasingly more important for network administrators to be able to identify devices on the network. The root of this necessity lies in the escalating number of attacks that are performed per day. Therefore, methods that identify devices based on their behaviour are being developed to identify and verify whether the behaviour of a device is similar to the already existent user profile, or the device is malicious and the communication needs to be blocked. The goal of this paper is to propose an algorithm that provides administrators an automatized method to track network devices on the networks based on their behavior without the need of tracking any additional information about the devices.

This work explores the current state of the art solutions in Section 2 and proposes new approaches to tracking network devices based on their behavior, with better performance, using frequency analysis to create device profiles and the k-Nearest Neighbors algorithm with cosine similarity as the distance metric. Device profiles are created based on the usernames and source IP addresses, since MAC addresses that would uniquely identify devices are rarely available in a production environment. However, in the case of this paper the MAC addresses are always available and are used to measure the performance of the proposed algorithm (e.g., MAC addresses are used as labels). This combination of features (source IP address and username) to create device profiles has been chosen since it provides a very simple and effective way to distinguish devices in a given time-window. However, this combination of features is not sufficient to identify a specific device, since a user can own multiple devices and the source IP address of a given device may change over time, therefore, behavior based device tracking is necessary.

A further advantage of the proposed approach to device identification is that compared to the methods described in Section 2, it is not reliant on the availability of all of the information, such as visited URLs, and performs well even in the absence of these features. The absence of features is pretty common since network administrators have different goals with what information they need to track and maintain, therefore, robustness and the absence of the necessity to track any additional information is a welcome improvement compared to the existent methods.

2. State of the Art

In the past, methods such as HTTP or TLS fingerprinting have been used to reliably identify devices. However, the constantly increasing network traffic loads result in a significant increase of the size of the databases. Furthermore, it becomes difficult to

maintain these large databases up to date and accurate, since the values of the fingerprints change over time and these values need to be kept for each device and visited website separately [1].

In addition, research that has been done up to this point explores only methods that measure similarity with previously seen behaviour [2]. Kumpost's algorithm achieves an overall accuracy of 78.3%, however, has high spatial and time demands over large data-sets, which renders these methods impractical in an environment with many network devices. Another disadvantage of his approach is that the aforementioned method works only on SSH traffic, where all of the features are always available, which is rarely the case in a production environment.

Another approach to tracking devices has been proposed in the dissertation thesis of Kohno et al. [3] using clock-skew to uniquely identify devices. Clock-skew relies on the uniqueness of crystal frequencies (deviations from the factory frequency are introduced due to manufacturing tolerances) that computers use to measure time, and measure the time measurement deviation from a base device. Kohno et al. proposed to estimate clock-skew by using the slope of the offset points, which is the difference from the beginning of the measurement (e.g., packet transmission) until the observer receives the packet. They have shown that the slope of the upper bound is similar to the slopes of the offsets. This approach, however, often requires synchronized sampling [4] to measure clock-skew, which is an active method to tracking devices.

3. Proposed Approach

The goal of this work is to propose an algorithm that will be accurate at identifying network devices and have low time and spacial requirements. The first Subsection 3.1 describes the proposed approach, and the second Subsection 3.2 describes the data-set on which the experiments are performed.

3.1 Proposed Algorithm

Before applying the algorithm to the data, it is necessary to analyze the data-set in order to gain insight into the behaviour and nature of the features. The analysis revealed that it is possible to determine the duration of the DHCP lease, and the websites visited by the devices. Since the URLs often contain the path, queries and ports which are unique to the specific request, they are removed from the URL and only the domain is extracted to reduce the dimensionality without the loss of information provided by the URL.

Since the data-set contains tens of millions of flows, it becomes computationally difficult to classify each flow by itself. Therefore, it is necessary to aggregate the flows into a profile that will be stored and later compared to the current device behavior. As it is necessary to aggregate multiple flows into one profile, the nature of the data-set (and aggregated profiles) very closely resembles written text.

One of the most common methods used to analyze text and compare similarity of texts is using frequency analysis in combination with a similarity measure, such as Jaccart or cosine similarity. Due to the resemblance of the aggregated flows to text, it enables us to create profiles using frequency analysis for these devices. Therefore, the exported data is separated into 5-minute segments and profiles are created based on the usernames and source IP addresses, since MAC addresses that would uniquely identify devices are rarely available in a production environment. They are, however, used as labels for the purposes of this paper. This combination of features (source IP address and username) to create device profiles has been chosen, since it provides a very simple and effective way to distinguish devices in a given time-window. However, this combination of features is not sufficient to identify a specific device, since a user can own multiple devices and the source IP address of a given device may change over time.

Since some of the data might contain duplicates of the same value but in a different context (such as source and destination IP addresses), these features are analyzed separately and then appended to the vector containing frequencies of all of the features (an example of flow aggregation based on source IP address of a device can be seen Table 1 and Table 2). This approach enables the creation of user profiles that are memory efficient and accurate for the given 5-minute window. Afterwards, it is possible to compare the similarity of these vectors in any of the following 5-minute time-frames.

Flow 1	Flow 2	Flow 3	Flow 4
User 1	User 1	User 1	User 1
src IP A	src IP A	src IP A	src IP A
dst IP B	dst IP B	dst IP C	dst IP D
port 1	port 1	port 1	port 2
google.com	google.com	-	facebook.com
TLS 1	TLS 1	TLS 2	-
-	HTTP f 1	-	-

Table 1. Example of a 5-minute aggregated profile for a device based on its source IP address.

The behaviour of users and devices can change over time, therefore, the previously created profiles need to be updated over time and the older profiles

Term	Frequency
src IP A	4
dst IP B	2
dst IP C	1
dst IP D	1
port 1	3
port 2	1
google.com	2
facebook.com	1
TLS 1	2
TLS 2	1
HTTP f 1	1
-	5

Table 2. Example of frequency analysis including only the features used in classification. The column frequency represents the vector used in similarity comparison.

need to be replaced by newer ones. Device users tend to change the context of their work approximately every hour, therefore, the number of 5-minute profiles that are stored in the memory is limited to 12. If there are 12 profiles, each time a device is classified as a specific device, the oldest profile is deleted and the current one is added into the profile pool. In the case that there are less than 12 profiles, the newest profile is directly added to the profile pool.

Since each of the devices has multiple profiles created for it, it is possible to use the k-Nearest Neighbors algorithm with cosine similarity as the distance metric (defined by Equation 1 [5]) to find the nearest behaviour profile for a given device. The value k has been chosen to be 3, since less than 1% of devices have communicated in only one 5-minute time-frame during the initialization phase. The initialization phase consists of profile creation in the first hour of the exported data using the frequency analysis method described earlier in this section. This approach, however, does not account for the new devices that appeared later than the first hour of the export. Therefore, flows that contain a previously unseen username have a profile created for that device. In the case when 2 of the 3 highest similarity values are below a threshold (set to 0.9, value selection is described in Section 4), the device is considered to be a new, previously unseen device.

$$similarity = cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

Device profiles are stored in memory in the form of cascading hash tables to improve searching efficiency. The following Figure 1 describes the system how device profiles are stored.

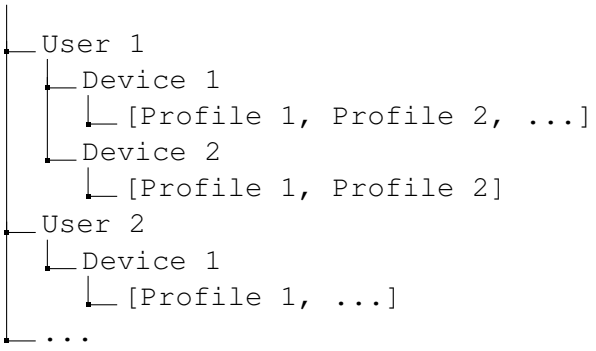

Figure 1. Hierarchy describing the structure of profiles stored in memory.

3.2 Data-set Description

Data-sets that are used for experiments of the proposed algorithm are provided by Cisco Systems and contain live data, therefore, the data-set itself cannot be published. The data has been collected using a combination of NetFlow, DNS queries and by using HTTP and TLS fingerprinting methods. However, features such as URLs, HTTP and TLS fingerprints are not always available, therefore, it is necessary to propose an algorithm that is robust and can overcome the sparsity of available data. The following features are available: username, source IP address, destination IP address, port, URL, HTTP fingerprint, TLS fingerprint, and the MAC address. The subsequent Table 3 describes basic information regarding the used data-set.

Total flows	64,806,407
Length of export [h]	16
Total users	7,385
Total devices	10,737
Total unique domains	53,226
Total unique TLS fingerprints	1,105,546
Total unique HTTP fingerprints	0
Availability of URL [%]	0.159
Availability of TLS fingerprint [%]	0.170
Availability of HTTP fingerprint [%]	0.0

Table 3. Basic information regarding the tested data-set.

4. Experimental Results

Due to the collected data from the 16-hour period missing values, it is first necessary to determine how do the available or unavailable features influence the accuracy of the proposed algorithm. Therefore, the first experiment evaluates the accuracy of the algorithm when the first hour is used for profile creation, and the second hour for scoring the accuracy without the

detection of new devices, where k-NN is used with k equal to 3. As it is always available, the source IP address is used to aggregate data for the profile in a given 5-minute period. Subsequently, the MAC addresses are used as the labels to evaluate the accuracy of the prediction. The following Table 4 describes the accuracy of the algorithm. From the table it is possible to come to the conclusion that even features that are often not available, such as TLS or HTTP fingerprints, improve the accuracy of the algorithm significantly.

Features	Accuracy [%]
source IP, destination IP, TLS fingerprint, HTTP fingerprint port, domain	93.65
source IP, destination IP, HTTP fingerprint port, domain	93.61
source IP, destination IP, TLS fingerprint port, domain	92.90
source IP, destination IP, port, domain	13.22
source IP, destination IP, port	0.0

Table 4. Results of the experiments evaluating the accuracy, depending on the used features.

The following step was to verify that the cosine similarity that is used as the distance metric creates a gap in similarity values between devices with the same and different MAC address. The following Figure 2 describes the distribution of values between the same and different devices. The device pairs that have been used for the comparison of similarity have been chosen at random to get the best representation of the distribution.

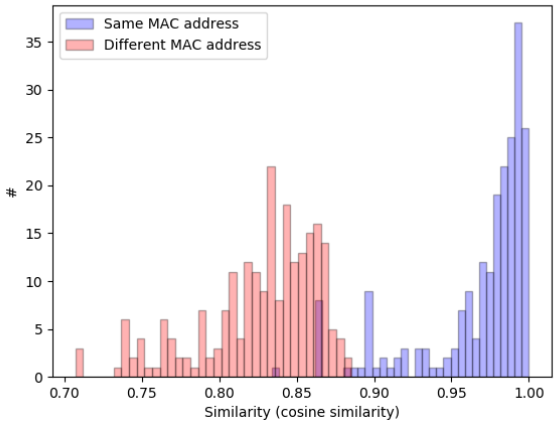

Figure 2. Histogram describing the distribution of cosine similarities between device profiles with the same MAC addresses (blue) and different MAC addresses (red).

The mean for the similarity of devices with the same MAC addresses is 0.98 and for devices with different MAC addresses the mean is 0.83. From this graph it is possible to conclude that there is a gap between the values of similarity between devices with the same and different MAC addresses, therefore, cosine similarity is the appropriate distance metric to be used in device identification. The following Figure 3 describes the distribution of device profiles, and visualizes how the profiles create clusters. Due to the device profile clustering, k-NN is an appropriate approach towards solving the problem.

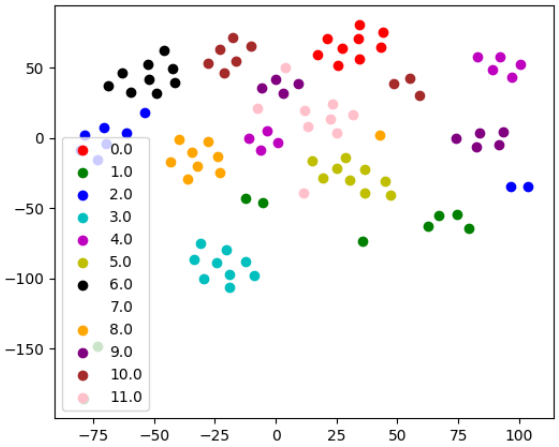

Figure 3. t-Distributed Stochastic Neighbor Embedding (t-SNE) used to visualize how device profiles are clustered. Each of the number labels represents one device and each point is a device profile.

Then, the effect of new device detection on accuracy needs to be evaluated. During this experiment the first hour has been used to create the initial profiles, as described in Section 3, and the effect of detection of new devices over time is evaluated. A device is classified as a new device, when all of the nearest neighbor distances are bellow the 0.9 threshold, chosen based on the histogram in Figure 2. The Table 5 describes the accuracy of the base algorithm. The following Table 6 then describes the accuracy during the given hour, without the replacement of old profiles.

Hour	Accuracy	Precision	Recall	F1
2	0.91304	0.88956	0.92931	0.89126
3	0.87210	0.84849	0.90754	0.84961
8	0.75068	0.72366	0.84124	0.72577

Table 5. Experimental results for the base algorithm without device detection or profile updates, and its accuracy over time.

Hour	Accuracy	Precision	Recall	F1
2	0.92779	0.89310	0.93256	0.89548
3	0.90626	0.85449	0.91313	0.85642
8	0.82190	0.73724	0.84763	0.73786

Table 6. Experimental results with new device detection without profile updates, and its accuracy over time.

Table 6 illustrates that the detection of new devices using the methods described in Section 3 provides a significant improvement in accuracy of the device identification algorithm.

The last experiment evaluates the accuracy of the proposed method when the algorithm is exchanging the profiles for the newer ones after each classification. This experiment also uses the concept of learning new devices during the classification phase as described in the previous experiment. The first hour is used for the initial profile creation and the following hours of the export are evaluated. The following Table 7 describes the achieved results. Lastly, Figure 4 compares the achieved accuracy of the aforementioned algorithms.

Hour	Accuracy	Precision	Recall	F1
2	0.92855	0.89815	0.93714	0.90031
3	0.90696	0.86116	0.91963	0.86295
4	0.88689	0.81799	0.89923	0.82030
5	0.87956	0.79107	0.88519	0.79327
6	0.86123	0.76723	0.87079	0.76893
7	0.84820	0.75270	0.86247	0.75451
8	0.83831	0.74328	0.85536	0.74492
9	0.83319	0.74001	0.85298	0.74167

Table 7. Accuracy and further metrics of the final proposed algorithm with the addition of the sliding window of profiles.

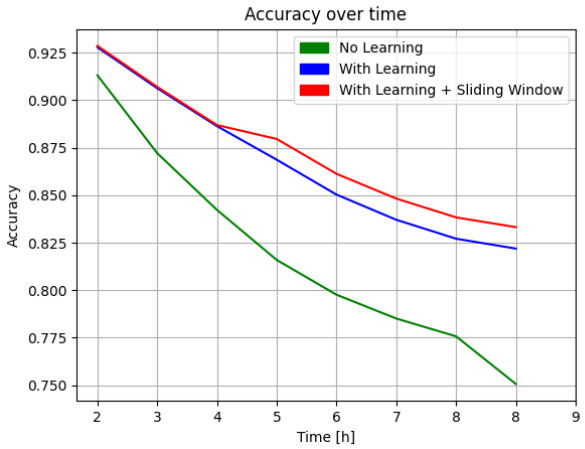

Figure 4. Comparison of all of the mentioned algorithms and their achieved accuracy over time.

Overall the average accuracy of the proposed algorithm is 89.26% over the tested 8 hour period. The decrease in accuracy over time is caused by incorrectly classified profiles, which cause a skew towards incorrect device classifications over the time period. However, the decrease in accuracy appears to level out at hour 9, which implies a consistent error rate in the future predictions.

5. Conclusion

This paper describes the implementation of the network device tracking algorithm using frequency analysis and the k-Nearest Neighbors algorithm with cosine similarity as the distance metric. The steps taken during the development of the algorithm are explained, along with the improvements over the previous version of the algorithm. Generally, the algorithm has proven to be effective over the tested 8 hour period, achieving the average tracking accuracy of 89.26%. The proposed algorithm provides a significant improvement compared to the currently used tracking methods, such as the method proposed by Kumpost achieving the accuracy at 78.3%. Furthermore, the proposed approach does not require any active querying, as clock-skew does, and proves to be an effective approach to passively track devices.

The only downfall of the algorithm is the limited ability to detect new devices based on their similarity. Therefore, algorithms such as outlier detection should be explored further to improve the accuracy of the proposed approach.

Acknowledgements

I would like to sincerely thank my supervisor Ing. Libor Polčák, Ph.D. for his valuable advice and guidance. Furthermore, I would like to thank Mgr. Jan Kohout and Cisco Systems for their advice and patience throughout the development of the algorithm.

References

[1] Blake Anderson. Tls fingerprinting in the real world, April 2019. Online; visited 3.4. 2020. Available at: https://blogs.cisco.com/security/tls-fingerprinting-in-the-real-world.

[2] Marek Kumpošt. *Context information and user profiling*. Masaryk University, Faculty of Informatics. Supervisor Vašek Matyáš, Brno, 2009. PhD thesis.

[3] Tadayoshi Kohno, Andre Broido, and K.C. Claffy. *Remote Physical Device Fingerprinting*. IEEE, 2005. ISSN 1545-5971.

[4] Steven J. Murdoch. Hot or not: Revealing hidden services by their clock skew. *Computer and Communications Security*, pages 27–36, 2006.

[5] Jun Ye. Cosine similarity measures for intuitionistic fuzzy sets and their applications. *Mathematical and Computer Modelling*, pages 91–97, 2011. ISSN: 0895-7177.