Exploiting DNS for Stealthy User Tracking

Béla Genge, Ioan Pădurean, Dan Macovei

{bgenge, ipadurean, dmacovei}@bitdefender.com

Bitdefender, Romania

1 Introduction

The Domain Name System (DNS) is a fundamental component of the Internet, providing a mapping between human-readable domain names and their corresponding IP addresses. It is used by all devices connected to the Internet, from computers and smartphones to IoT devices. DNS plays a crucial role in enabling users to access websites, services, and applications without needing to remember complex numerical IP addresses.

While DNS performs exceptional functions in making our Internet work, it also plays a critical role in delivering security applications. On the other hand, DNS is considered a "gold mine" by security companies developing security solutions. DNS is used to deliver a wide range of security solutions ranging from spam filtering, parental control, ad-blocking, distributed denial of service mitigation, to load balancing and Internet of Things (IoT) security. Consequently, DNS can be viewed as a double-edged sword: it is essential for the functioning of the Internet, but in the hands of security specialists it can be a gold mine for delivering security applications.

Unfortunately, wether they are aware or not, users are fueling this gold mine by providing valuable information about their online activities. This information is collected by security companies and used to improve their products and services, often without the users' explicit consent or knowledge. This statement is supported by several events from the past in which telecommunication operators (telcos) have been found to collect and analyze user data without proper transparency or consent (see Figure 1).

Nonetheless, the collection of user data is not limited to telcos. It is a common practice among various online service providers, including social media platforms, search engines, and e-commerce websites. These companies often track user behavior across multiple websites and services, creating detailed profiles that can be used for targeted advertising and other purposes.

This whitepaper aims to emphasize the high potential to use DNS for accurate user device tracking. It demonstrates that, despite the recommendations and policies in place such as Organisation for Economic Co-operation and Development (OECD) Privacy Guidelines, European Union's General Data Protection Regulation (GDPR), alongside various country-wide regulations (PIPEDA - Canada, HIPAA - US Healthcare, the California Consumer Privacy Act (CCPA)), user devices are still generating valuable information that can be used for tracking purposes.

It is noteworthy that there are technical privacy measures aimed to limit the tracking of user devices. To this end, we mention: (i) indirectly, randomize network MAC address to limit the time availability of device identifiers; and (ii) the DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT) protocols, which encrypt DNS queries and help protect user privacy. In the former case, the randomized network MAC address is used to prevent tracking based on the device's hardware address. While this raises the bar for tracking, the setting can be diverse, depending on the device type and operating system. For instance, on Android and iOS devices, the randomized MAC address is enabled by default for Wi-Fi connections. However, the same MAC address is used for the same Wi-Fi networks, or in the case of iOS there is a time limitation for the randomized MAC address, after which the value is randomly changed, thus limiting the time window for tracking. Nevertheless, as shown in this whitepaper, even with these privacy measures in place, user devices can still be tracked with high accuracy. In the later case, namely DoH and DoT, these protocols encrypt DNS queries, thus preventing local snooping. Nevertheless, DoH and DoT are not wide spread, at least for for the time being, and therefore, the DNS queries are still sent in clear text. On the other hand, even with DoH and DoT in place, user devices can still be tracked based on the DNS queries, as the DNS queries are still sent to a DNS resolver. If "curious", the DNS resolver can still collect and analyze the DNS queries, thus enabling user tracking.

This whitepaper complements the talk with the same title that was given at **BlackHat USA 2025**. The talk is aimed to be a call for action to take urgent action to protect user privacy and to raise awareness about the potential risks associated with user tracking. It is also a call for action to security researchers, developers, and policymakers to work together to create a more secure and privacy-respecting Internet. This includes advocating for stronger privacy protections, transparency in data collection practices, and

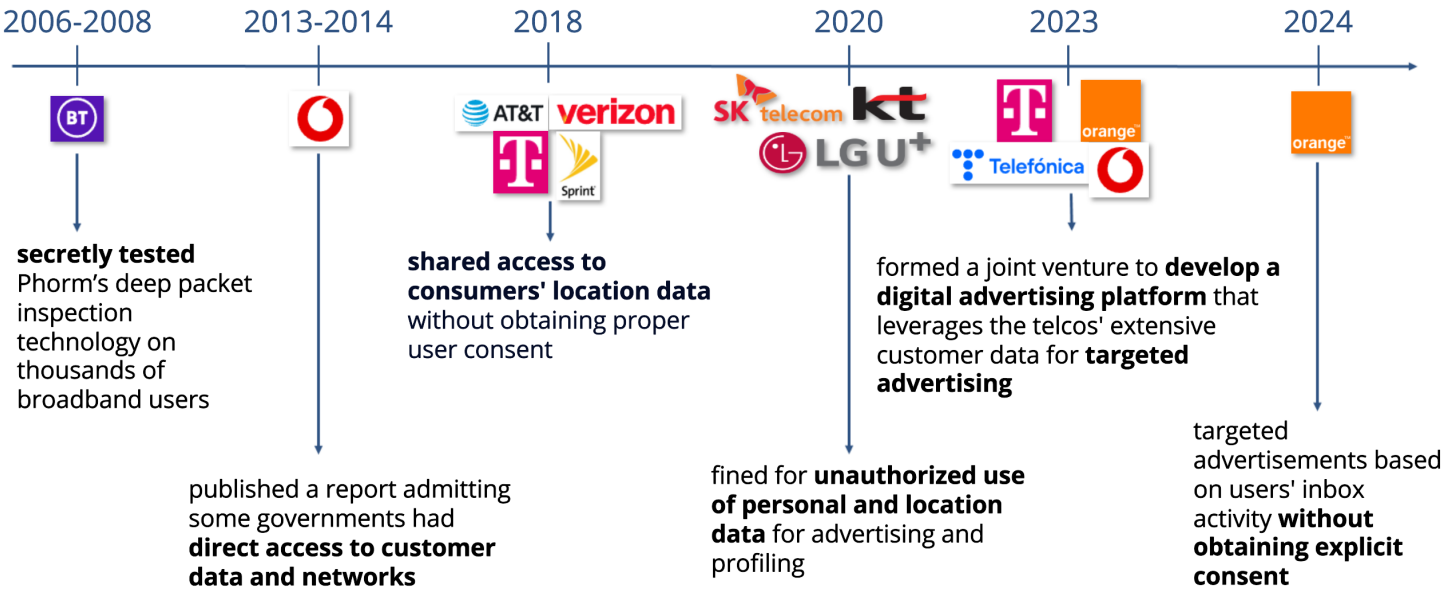

Figure 1: Past events regarding misbehavior in user data collection.

the development of technologies that prioritize user privacy.

The remainder of this paper is structured as follows. Section 2 provides an overview of related work in the field of user tracking through DNS queries. Section 3 describes the methodology used in this research, including data collection, dataset analysis, and user tracking methodology. Section 4 presents the experimental results obtained by applying the user tracking methodology, including tracking accuracy and comparison with machine learning techniques. Section 5 concludes the paper and discusses the implications of the findings.

2 Related work

The topic of user tracking is not new, and there is a significant body of work that explored this area. Especially in the context of DNS, there are several studies that have investigated the potential for user tracking through DNS queries.

We start by mentioning the work of Herrmann et al. [1], which explored the possibility of tracking users based on their DNS traffic patterns. The authors used two months of DNS query data collected from a university campus and student housing to analyze the behavior of user devices and to identify characteristic patterns in their DNS traffic.

We also mention the work of Lai et al. [2], which focused on visualizing and characterizing DNS lookup behaviors through log-mining techniques. The authors analyzed DNS logs from three primary DNS servers in a large university campus network in China, providing insights into user behavior and the overall trend of users' surfing activities. In a similar direction we find the works of [3], [4], and numerous others such as [5] and [6], which also explored the potential for user model and behavior analysis through DNS queries.

In comparison to prior studies, this whitepaper distinguishes itself from several perspectives. First and foremost, it focuses exclusively on smart phones, and, on the two main operating systems: Android and iOS. This is a significant difference, and, nowadays, smart phones are the best candidates for tracking user activity. This is due to the fact that smart phones are used for practically all on-line activities, including: social media, e-commerce, banking, and so on. As a result, the potential for user tracking through DNS queries is outstanding. Second, this research accessed a significant amount of data, from a large pool of devices. Third, the research provides key insights that are particularly relevant for smart phones.

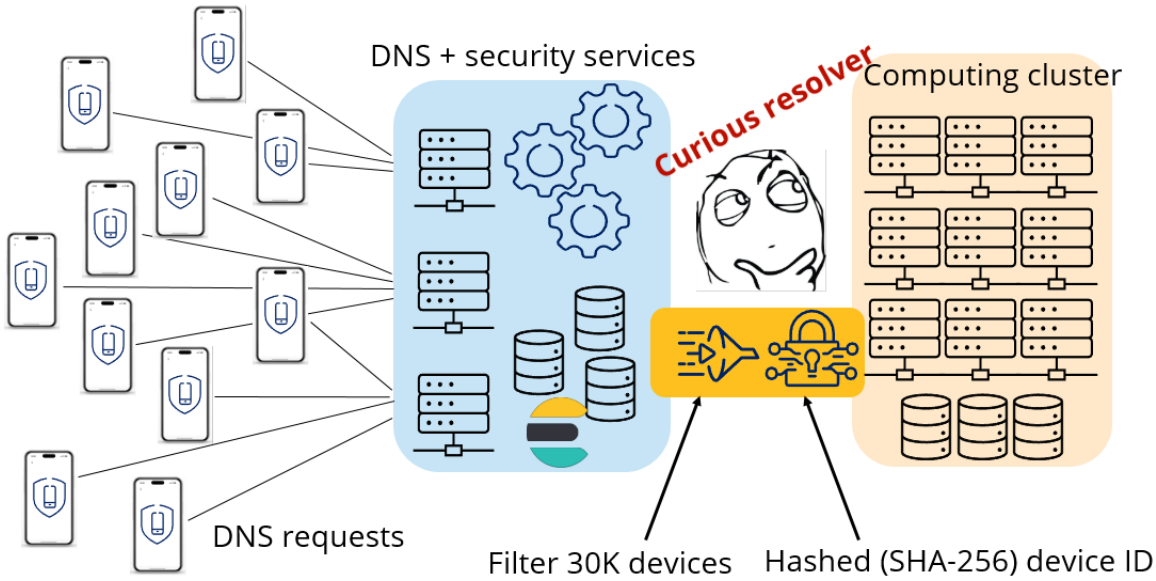

Figure 2: Data collection: the case of the curious DNS resolver.

3 Methodology

This section describes the methodology used in this research, including data collection, dataset analysis, and user tracking methodology.

3.1 Data collection

To ensure relevance, we played the role of the curious DNS resolver (see Figure 2). At Bitdefender, we operate DNS resolvers that are used by millions of users worldwide. We tapped into one of these resolvers by registering to the DNS resolving events for a subset of approximately 30.000 smart phones. All details regarding device and user identification were anonymized by applying a one-way hash function (i.e., SHA-256) to the device identifiers to make sure that the data is not traceable to a specific device or user. DNS requests were collected over a period of 35 days, after which the data was analyzed to extract the relevant information and to establish the conditions under which user tracking is possible. In the next phase, for processing, we used 4 computing clusters, totaling 80 virtual CPUs and 1.5 TB of RAM.

3.2 A bird's eye view of the dataset

In total, while originally much larger, after filtering and processing (also considering time and resources limitations) the dataset used in this research was reduced to approximately 985 million DNS requests. After excluding devices with only few records, the number of devices used throughout this study was of approximately 28.000 smart phones.

We first analyzed the dataset from a bird's eye view perspective, to get a general understanding of the data. Figure 3 shows the distribution of DNS requests in terms of device operating system, number of requests per device, and the number of unique domains per device. The left-hand side of the figure shows a visible distinction between Android and iOS devices. The number of unique domains per device is relatively the same for the two operating systems. Conversely, the number of requests per device is significantly higher for iOS devices. To better highlight this aspect, we computed the cumulative distribution function (CDF) of the number of requests per device. As shown in the right hand-side of the same figure, the number of requests per device is significantly higher for iOS devices, in certain cases iOS devices tend to generate 10 times more requests than Android devices.

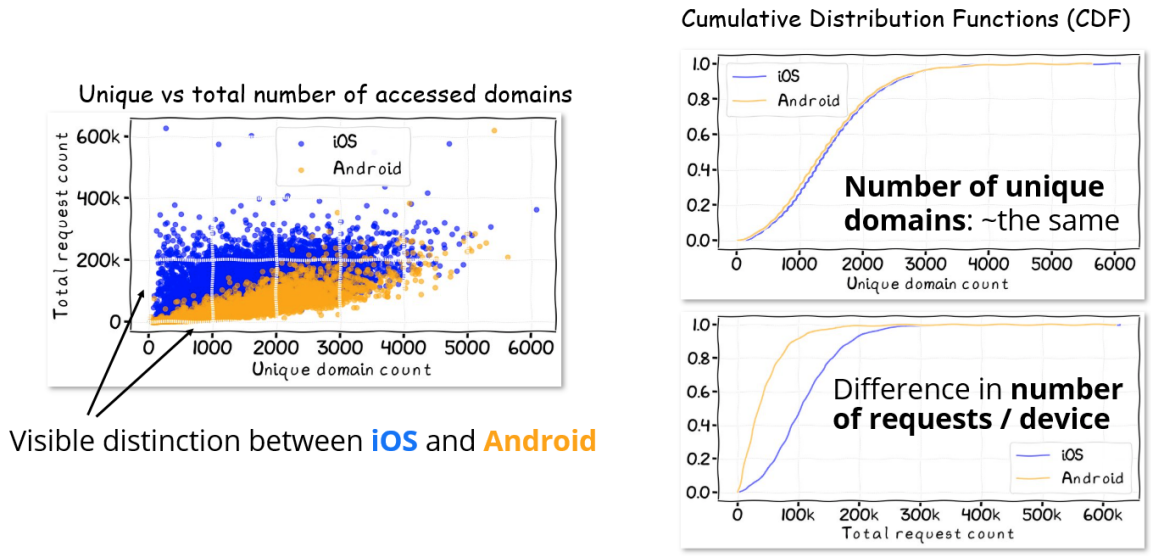

Figure 3: Dataset: a bird's eye view.

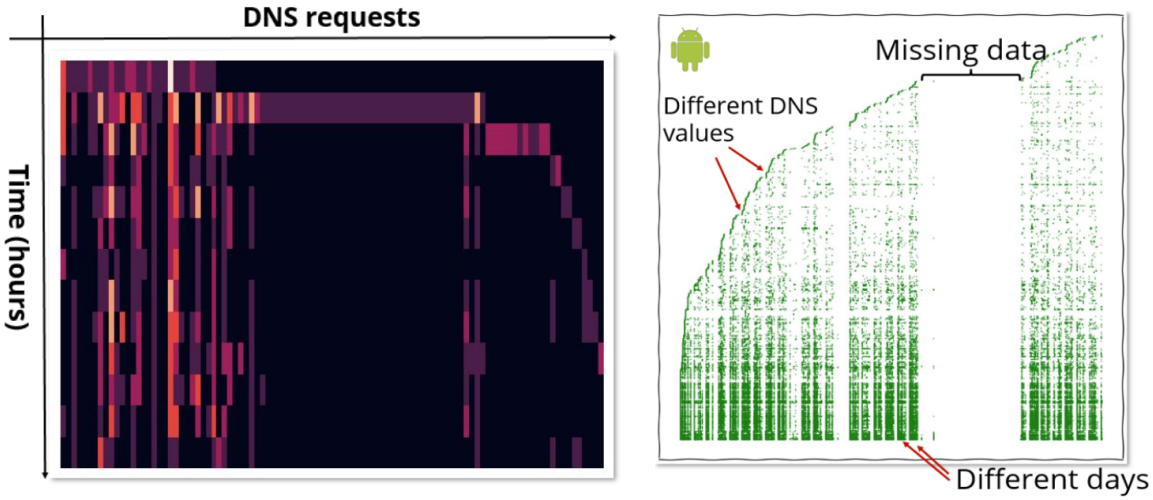

Figure 4: Request pattern examples.

Next, we explored the patterns exposed by DNS request sequences. For instance, we counted the number of distinct domain requests over one hour time periods, and we have color-coded the results. An example is shown in the left hand-side of Figure 4. In this figure, the brighter the color, the more number of requests have been issued for a particular domain. As shown here, certain domains are persisting over time, while others are more sporadic. This is a clear indication that certain domains are more relevant for the user, and are likely accessed more frequently not necessarily in a direct approach, but also indirectly through the installed apps. On the right hand-side of Figure 4, we show an example of request patterns over several weeks for one device. In this case, as we analyze the request patterns we notice the different days of the week, as requests group together. Subsequently, we can also observe that, as time passes (X axis), more domains are accessed (Y axis). At the same time, we also encountered time frames with missing values, which are likely due to the fact that the device was not connected to the Internet, or the user was not using our DNS resolver.

3.3 DNS request transformations

While the previous section provided a general overview of the dataset, we needed to perform additional transformations to ensure proper input for numerical analysis. In fact, early on in this research we realized

the two elephants in the room when it comes to processing DNS requests: we need to deal with categorical values, and we have a huge amount of requests.

To address the first issue, we transformed the categorical values (i.e., domain names) into numerical values. The first challenge that any analysis must tackle is how to work with categorical data. Algorithms best work with numbers, not with strings. The most commonly used techniques are: one-hot encoding, label encoding and feature hashing. One-hot-encoding generates a binary matrix, while label encoding generates an index for each string, in alphabetical order. On the other hand, feature hashing applies a hash function over the string, generating a number for each distinct one. Unfortunately, an analysis of all these encodings showed that they do not apply to the problem at hand. In fact, each one comes with its own drawback ranging from high dimensionality to hash collisions.

To address this issue, domain requests were transformed to sequences of words, namely to *documents*, a solution commonly applied in the field of Natural Language Processing (NLP). By framing the issue to a document comparison problem, we have also identified the tool to compare documents, namely the term frequency-inverse document frequency (TF-IDF). TF represents the rate of the words that appear in the document, while IDF represents the rarity of the term across the documents. More formally, let D be the set of documents, and d be a document from D. The TF-IDF score for a term t in document d is defined as:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t, D) \tag{1}$$

where

$$\text{TF}(t, d) = \frac{f(t, d)}{\sum_{t' \in d} f(t', d)} \tag{2}$$

and

$$\text{IDF}(t, D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right). \tag{3}$$

In terms of applications, TF-IDF works well with strings, quantifies the frequencies of words and highlights those that are rare in most documents. It also scales well because of usage of documents, and the risk of bias is low. So how does one interpret the value of TF-IDF? Lower TF-IDF value maps to higher frequency across all documents or lower frequency in a single document. On the other hand, high TF-IDF value means that a certain word appears repeatedly in a few documents.

Besides having a proper representation of the categorical values, we also needed to address the issue of comparing documents. For this purpose we have chosen cosine similarity, which handles very well high-dimensional data, and compares elements found in documents, and not their frequency. Formally, the cosine similarity between two documents d_1 and d_2 is defined as:

$$\text{cosine\_similarity}(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\| \cdot \|d_2\|} \tag{4}$$

where $d_1 \cdot d_2$ is the dot product of the two documents, and $\|d_1\|$ and $\|d_2\|$ are the norms of the documents, respectively. If we take two documents, one containing the words Black Hat, and a second one repeating the same words three times, by applying cosine similarity we get that the two sentences are identical. However, if we were to apply a different metric such as Euclidean distance, the numbers would yield that the documents are different. Hence, the motivation for choosing cosine similarity as a metric to compare documents.

For the second challenge we had to deal with the astonishing size of the data. Just loading all the data for 28.000 devices in memory takes tens of GB of RAM. To address this issue we identified 3 key parameters:

- The number of devices: for each scenario we selected subsets of devices.
- The number of time windows: we divided the data into time windows, which denotes the time frame for which a device is active. We considered different time windows ranging from 1 hour to 24 hours.

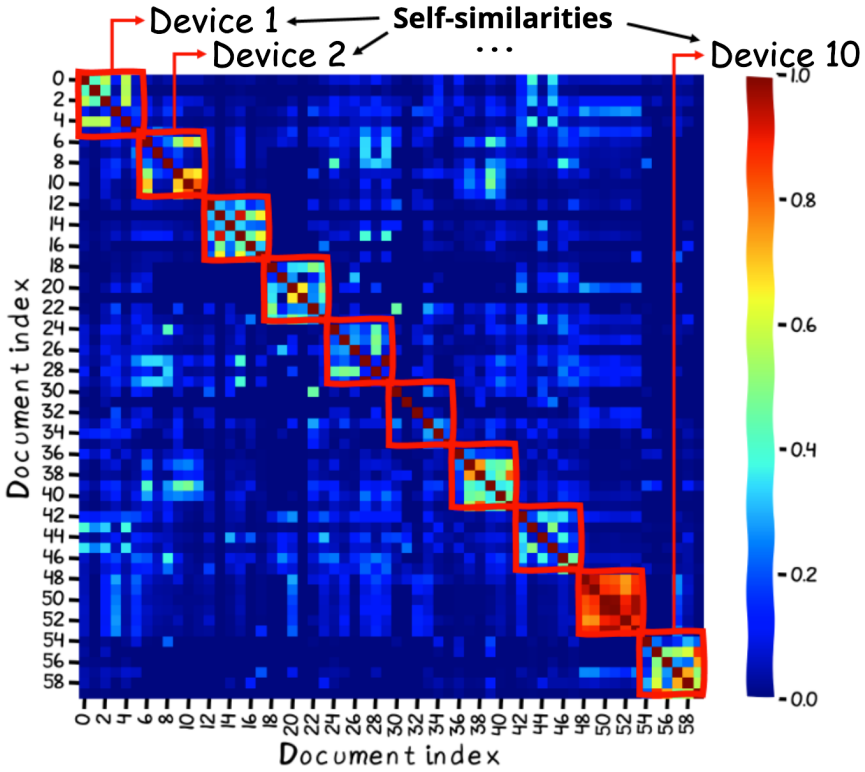

Figure 5: Similarity matrix example.

- The size of each time window: the lower the time window, the more documents we generate.

The combination of these three parameters allowed us to reduce the size of the data to a manageable level. We also utilized different device batches, based on the previous 3 parameters, ranging from 200 to 5000.

3.4 User tracking methodology

The user tracking methodology is based on the assumption that devices generate several DNS traces over time, and that these traces can be used to identify the user. The methodology consists of several steps. First, raw requests are split into time windows, which are then transformed into documents. The documents are then compared using cosine similarity. Lastly, a tracking index is computed.

In order to simplify the analysis, we have refined and formulated the user tracking methodology as computations that are performed over a similarity matrix. As illustrated in Figure 5, the similarity matrix is a square matrix that contains the cosine similarity values between all pairs of documents. The rows and columns of the similarity matrix represent the documents, while the values in the matrix represent the cosine similarity between the documents. The diagonal of the matrix contains the cosine similarity values between each document and itself, which is always 1. The squares around the diagonal represent the similarity values between the documents associated to the same smart phone. As illustrated in the same figure, the colors are brighter around the diagonal, which indicates that the documents are more similar to each other, that is, documents associated to the same smart phone are more similar to each other than to documents associated to different smart phones.

Next, in the preparation of defining the tracking index, we need to define two distinct indexes: (i) the average device self-similarity index; and (ii) the average device cross-similarity index. The average device self-similarity index is computed as the average of the cosine similarity values between all pairs of documents associated to the same smart phone. Formally, let M denote the similarity matrix, and M^{kk} the sub-matrix of M that contains the cosine similarity values between all pairs of documents associated to

the same smart phone k. Let $M^{kk}_{i,j}$ denote the cosine similarity value between documents i and j associated to the same smart phone k. The average device self-similarity index S^s_{kk} is defined as:

$$S^s_{kk} = \frac{1}{n(n-1)/2} \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} M^{kk}_{i,j}, \tag{5}$$

where n is the number of documents associated to the smart phone k. The average device cross-similarity index is computed as the average of the cosine similarity values between all pairs of documents associated to different smart phones. Formally, let M^{kl} $(k \neq l)$ denote the sub-matrix of M that contains the cosine similarity values between all pairs of documents associated to different smart phones k and l. Let $M^{kl}_{i,j}$ denote the cosine similarity value between documents i and j associated to different smart phones k and l. The average device cross-similarity index S^c_{kl} is defined as:

$$S^c_{kl} = \frac{1}{n_k n_l} \sum_{i=0}^{n_k-1} \sum_{j=0}^{n_l-1} M^{kl}_{i,j}, \tag{6}$$

where n_k and n_l are the number of documents associated to the smart phones k and l, respectively.

Now we are ready to define the tracking index. The tracking index T_k for a smart phone k is defined as:

$$T_k = S^s_{kk} - \max_{l \neq k} S^c_{kl}. \tag{7}$$

If the tracking index is positive, the device is more similar with itself than with any other devices, and therefore, it is trackable. If the tracking index is negative, the device is less similar with itself than with any other devices, and therefore, it is not distinguishable from other devices. The tracking index can be interpreted as a measure of how well a device can be tracked based on its DNS requests. A higher tracking index indicates that the device is more trackable, while a lower tracking index indicates that the device is less trackable.

4 Experimental results

This section presents the experimental results obtained by applying the user tracking methodology described in Section 3. The results are presented in terms of the tracking index computed for different time windows and device batches.

4.1 Tracking accuracy using the detailed methodology

The tracking accuracy is defined as the percentage of devices that can be tracked based on their DNS requests. The tracking accuracy is computed as the number of devices with a positive tracking index divided by the total number of devices. First, we evaluated the tracking accuracy for different time windows ranging from 1 hour up to 72 hours, and batches of 250 devices taken from the entire set of 28.000 smart phones. The average tracking accuracy results have been summarized in Figure 6. As shown in the figure, computations have been performed for iOS and Android platforms separately, and considering a mix of both platforms. The results show that the tracking accuracy is slightly higher for Android devices compared to iOS devices. This is due to the fact that iOS devices generate more DNS requests, with more distinct domain names, which increases the size of the time window needed to capture all relevant requests. Consequently, the tracking accuracy is higher for larger time windows, as more requests are captured.

Nevertheless, as shown in Figure 6, the tracking accuracy reaches a shocking value that exceeds 80% after only 2 hours of smart phone activity. This accuracy increases over 90% after capturing 12 hours of activity and tops at 96% after 24 hours of activity. This means that, after 24 hours of activity, 96% of the devices can be tracked based on their DNS requests.

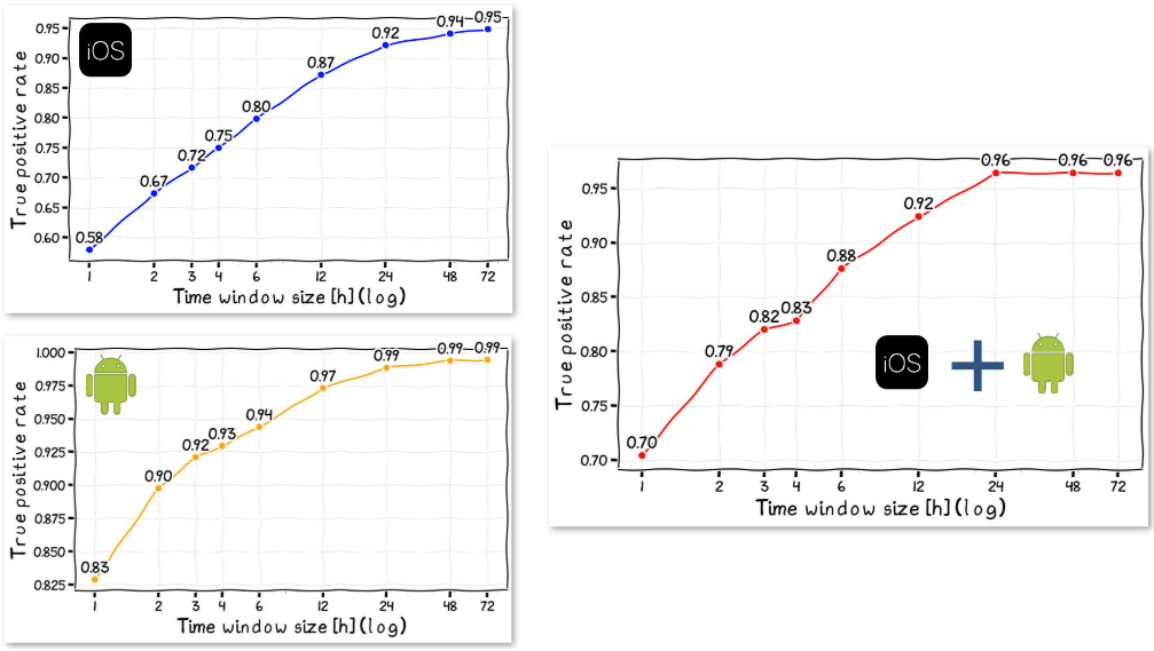

Figure 6: Tracking index accuracy for 250 devices.

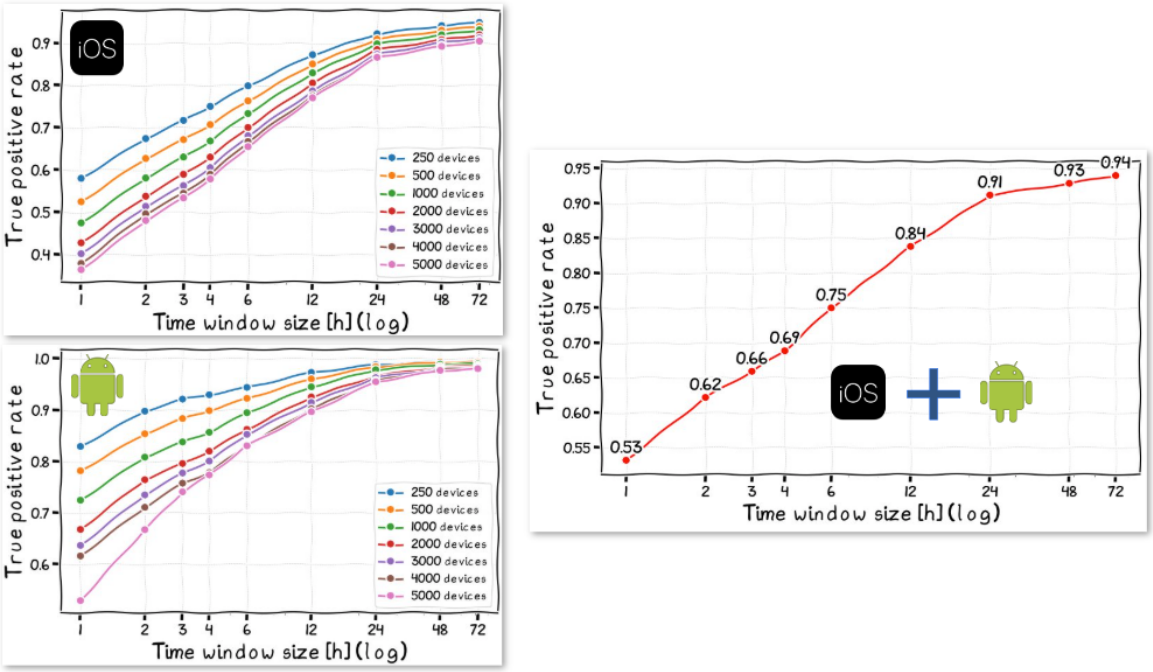

Figure 7: Tracking index accuracy for various device batches, up to 5000 devices.

Next, we evaluated the tracking accuracy for different device batches, ranging from 250 to 5000 smart phones. The results are shown in Figure 7. Here, on the left-hand side, we show the tracking accuracy for iOS and Android devices separately. Results for a mix of both platforms are shown on the right-hand side. As shown in the figure, the tracking accuracy decreases with the number of devices in the batch. For the mixed case, the tracking accuracy for two hours of activity is around 62% for 5000 devices. Nevertheless, this value increases above 90% after 24 hours of activity.

The results show that the tracking accuracy is still high even for larger device batches, which indicates that the user tracking methodology is robust and can be applied to large-scale environments comprising of thousands of smart phones.

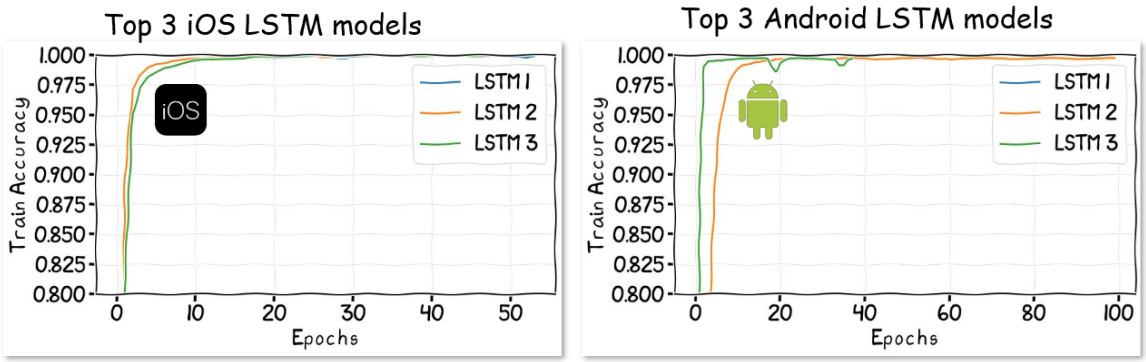

Figure 8: Tracking index accuracy for various LSTM models.

4.2 Tracking accuracy with machine learning

In addition to the detailed methodology, we also explored the use of machine learning techniques to improve tracking accuracy. While in the scientific literature we find a plethora of machine learning algorithms that can be applied to the problem at hand, we decided to focus on a single algorithm, namely LSTM (Long Short-Term Memory). The reason for this is that LSTM is a type of recurrent neural network that is designed to handle sequential data, such as DNS requests. LSTMs have also been successfully applied in speech recognition, time series, robot control, and many more fields.

However, choosing the best architecture of LSTM is a distinct problem on its own. There are many hyperparameters to tune, including the number of layers, the number of units in each layer, the learning rate, and the batch size. Additionally, the choice of optimizer and the initialization of weights can also have a significant impact on the performance of the model.

Besides these aspects, we also had to deal with the size of our dataset. We ended up consuming a lot of time in finding not only the optimal LSTM model, but the model that could fit into our available memory (which was not small, but 400 GB RAM). As it turns out we are unable to make data related to 5000 devices and time windows of 3h fit into memory. We had to reduce the data size, that is, reduce the number of documents, by grouping them into larger time windows. Once we identified the data structure that could fit in memory, we dived into the problem of finding the optimal model. After many trials and errors, we have found a configuration that worked the best: one LSTM layer with 128 cells would fit the needs of this research.

The results of the LSTM model are shown in Figure 8. The left-hand side of the figure shows the tracking accuracy for iOS devices, while the right-hand side shows the tracking accuracy for Android devices. As shown in the figure, the tracking accuracy is comparable, and, with the increasing number of training epochs the tracking accuracy increases and it slightly surpasses the accuracy obtained with the methodology detailed in the previous sections. In fact, top accuracy reaches 96% for both platforms.

While the results are promising, it is important to note that the LSTM model requires a significant amount of computational resources and time to train. Training only 48 architectures over 2500 devices consumed 280 GB of RAM for iOS and 190 GB for Android. Subsequently, we used 2 computing clusters, 64 virtual CPUs, and a total of 8 days for training 48 models for each OS. So, there are significant drawbacks to using machine learning for user tracking, which are owed to the data size, resources required for training, and for optimizing model parameters.

4.3 Dataset reduction

Towards the end of our research, we observed that devices tend to generate repetitive DNS requests. This prompted the question of whether it is necessary to retain all requests, or if maintaining their statistical distribution would be enough for accurate tracking. To investigate this, we performed a random under-sampling of DNS requests and evaluated the resulting impact on tracking accuracy. Remarkably, the

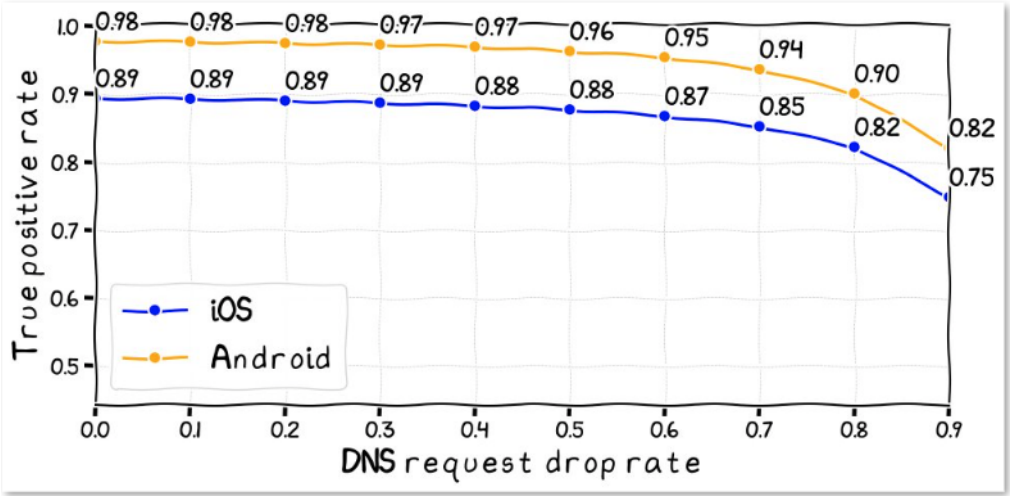

Figure 9: Tracking index accuracy with under-sampling.

results indicated that removing up to 20% of requests did not affect accuracy, and even with a reduction of approximately 80%, devices remained relatively easy to identify. Results are shown in Figure 9.

Given these findings, we conclude that it is possible to reduce the size of the dataset by approximately 80% without significantly impacting the tracking accuracy. This reduction can lead to substantial savings in terms of storage and computational resources, making the user tracking methodology more efficient and scalable.

5 Conclusions

This whitepaper accompanies the talk with the same title given at BlackHat USA 2025. It is aimed as a call for action to raise awareness about the potential risks associated with user tracking through DNS requests. The research presented in this paper demonstrates that user devices can be tracked with high accuracy based on their DNS requests, even when privacy measures such as randomized MAC addresses and DNS-over-HTTPS (DoH) are in place. The results show that the tracking accuracy can exceed 90% after only 24 hours of activity, and that the user tracking methodology is robust and can be applied to large-scale environments comprising of thousands of smart phones. The findings of this research should be taken seriously by all stakeholders involved in the development and deployment of smart home devices, as well as by security researchers and policymakers. The research highlights the need for a more secure and privacy-respecting Internet, and it calls for urgent action to protect user privacy.

Acknowledgements

Special thanks to our colleagues from the team of machine learning engineers from Bitdefender for their support and collaboration throughout this research. We extend our sincere gratitude to our BlackHat speaker coach, Dr. Marina Krotofil, for her invaluable guidance on delivering an impactful talk, her assistance with refining our slides, and her confidence in the significance of our research!

References

[1] D. Herrmann, C. Banse, and H. Federrath, "Behavior-based tracking: Exploiting characteristic patterns in dns traffic," *Computers & Security*, vol. 39, pp. 17–33, 2013. 27th IFIP International Information Security Conference.

[2] Q. Lai, C. Zhou, H. Ma, Z. Wu, and S. Chen, "Visualizing and characterizing dns lookup behaviors via log-mining," *Neurocomputing*, vol. 169, pp. 100–109, 2015. Learning for Visual Semantic Understanding in Big Data ESANN 2014 Industrial Data Processing and Analysis.

[3] K. Schomp, M. Rabinovich, and M. Allman, "Towards a model of dns client behavior," in *Passive and Active Measurement* (T. Karagiannis and X. Dimitropoulos, eds.), (Cham), pp. 263–275, Springer International Publishing, 2016.

[4] M. Panza, D. Madariaga, and J. Bustos-Jimenez, "Extracting human behavior patterns from dns traffic," vol. 77, pp. 407–420, 2022.

[5] J. Qu, X. Ma, and W. Liu, "Who is dns serving for? a human-software perspective of modeling dns services," *Knowledge-Based Systems*, vol. 263, p. 110279, 2023.

[6] Z. Sun, T. Guo, S. Luo, Y. Zhuang, Y. Ma, Y. Chen, and X. Wang, "Dns request log analysis of universities in shanghai: A cdn service provider's perspective," *Information*, vol. 13, no. 11, 2022.