The Nonces Project Progress Write-Up

Ihyun Nam

July 2023

1 Motivation

Client nonces sent by the client in a TLS connection are not meant to be repeated, but we observed that some clients use repeated nonces. Our ultimate goal is to diagnose why repeated nonces occur. To do so, we want to create a tool (that ideally everyone can use later) that can identify the client in a TLS connection. As a part of that, we are creating an automated process to identify the OS or the specific device (e.g. IoT) of the client without creating and relying on a database.

2 Literature Review

Three properties commonly used for OS identification are TCP/IP parameters, user-agent, and domain names. According to a recent survey [1] on modern OS identification techniques, most OS identification relies on creating some sort of a database to map certain properties of a device to an OS. Some of the most notable databases that maps TCP/IP parameters to OS are p0f [2], Joy [3], and Zardaxt [4]. However, our empirical results in Section 3 provide concrete evidence that databases are not sufficient to identify client OS. This is an issue to address because these databases are still actively used in research in the field of client OS identification. For example, a recent work by Rexford et al. [5] used p0f to create p40f, a client OS classifier for faster traffic, but p0f proved to be able to classify only 23% of their collected traffic. We believe that while these databases were reliable when they first came out (Zardaxt in the early 2010s, p0f in 2012, and Joy in 2016) they must have grown outdated as popularity in OS shifted and IoT devices emerged.

Similarly, we show in Section 4 through empirical results that a recent database that maps server domains to OS can only classify 20% of our collected fingerprints. This shows that databases can never keep up with the speed of shifts in the client landscape. To the best of our knowledge, no databases are currently updated automatically, which makes the task of maintaining databases strenuous and contributes to the databases growing obsolete after a few years of disuse.

Furthermore, relying on user-agent is well-known to be unreliable since user-agents can be easily edited by a client that wishes to hide its identity.

3 Why database-based OS identification using TCP/IP parameters fails

We identified four most commonly used OS fingerprinting databases that uses TCP/IP parameters to identify client OS: p0f, Joy, Zardaxt, and JA3. We attempted to identify the client OS of 2,000 TLS connections we observed in the Stanford network on 11 November, 2022 during a 24-hour period.

3.1 Overview of results

Our empirical results show that (a) these databases can identify only a very small portion of our TLS clients and (b) even when they do, their results rarely agree.

	Joy	p0f	Zardaxt	JA3
Raw	331	370	2000	115
Percentage(%)	16.55	18.50	100.00	5.75

Table 1: Number of fingerprints each database could classify in the operation system name level

In Figure 1, we see that very little fingerprints exist in the selected databases to begin with. More importantly, in Figure 2 we see that the number of fingerprints that the databases identify as the same OS drastically decreases as more databases are consulted.

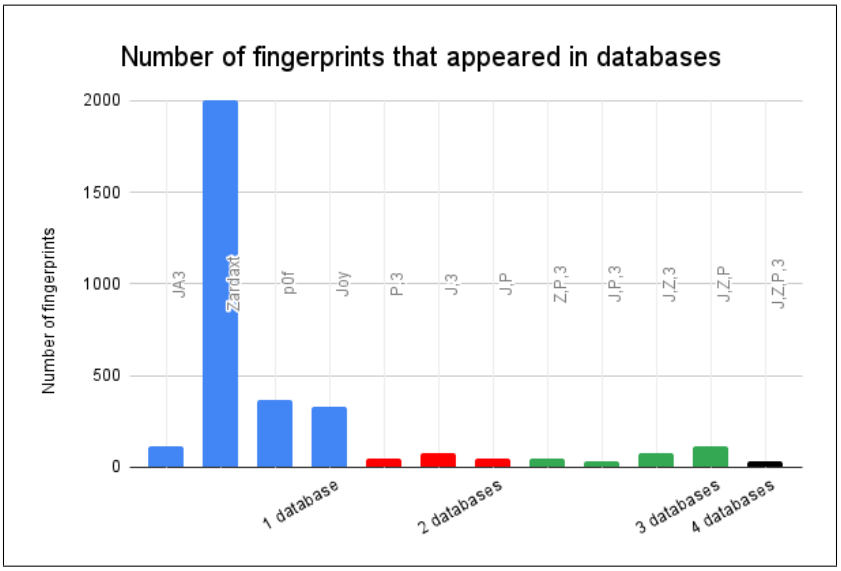

Figure 1

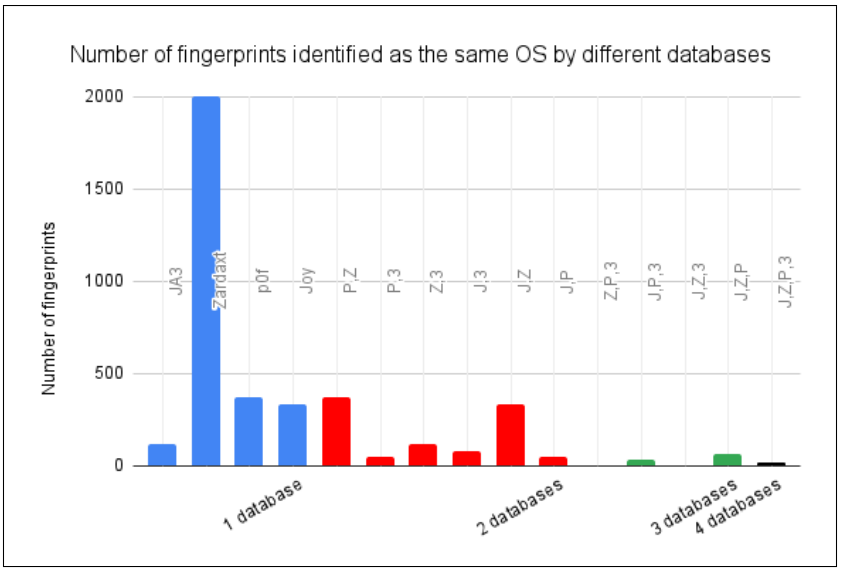

Figure 2

Note: Zardaxt returns the most likely OS (among Linux, Windows, iOS, and Android) based on their scoring system out of 16.75. Therefore, it's able to give a label to 100% of the fingerprints. However, the average score was 6.65, so the results are not too accurate.

3.2 Fingerprints identified by 4/4 databases

For this initial write-up, we only consider the operating system name (e.g. Windows) as opposed to the operating system version (e.g. Windows 10.0) to compare the results of identification by different databases. The four databases could identify 33 fingerprints, which is **1.65%** of our fingerprints. The four databases agreed on the result of 18 fingerprints, or 0.9% of our fingerprints. Also, these were all classified as Windows because Windows is only OS whose fingerprints were collected in all four databases.

3.3 Fingerprints identified by 3/4 databases

'Appearance' is the number of fingerprints that appear in all three databases and 'matching' is the number of fingerprints that all three databases identified as the same OS.

	Joy, Zardaxt, p0f	Joy, Zardaxt, JA3	Joy, p0f, JA3	Zardaxt, p0f, JA3
Raw	117	76	33	46
% of total	5.85	3.80	1.65	2.30

Table 2: Number of fingerprints that appeared in three databases

	Joy, Zardaxt, p0f	Joy, Zardaxt, JA3	Joy, p0f, JA3	Zardaxt, p0f, JA3
Raw	61	0	33	1
% of total	3.05	0	1.65	0.05

Table 3: Number of fingerprints that three databases identified as the same OS

3.4 Fingerprints identified by 2/4 databases

To begin with, Joy only had matching data for Mac OS and Windows. In the entire database, it only contains Mac OS, Windows, and Linux. p0f only matching data is Windows and Linux.

	Joy, p0f	Joy, Zardaxt	Joy, JA3	p0f, Zardaxt	p0f, JA3	Zardaxt, JA3
Raw	46	331	76	370	46	115
% of total	2.30	16.55	3.80	18.50	2.30	5.75

Table 4: Number of fingerprints that appeared in two databases

4 Why database-based OS identification using SNI fails

Using SNI is one of the popular methods to identify cliehnt OS, along with using TCP/IP parameters. However, according to our literature review, current methods all rely on some database, such as by creating a 'SNI to OS' mapping from observed traffic. In particular, a 2018 paper [6] collected 100 domain names and labeled each with the most likely OS of a client connecting to the domain. However, **only 20.55% of the unique client IPs in our TLS handshake database ever connected to a domain in the database**. This proves our point that maintaining a manual database is impractical and there needs to be a new database-free method.

5 Our new method: database-free client identification

We want a method that produces more reliable results that let us be more confident in identification for more clients. Such a method may not give high confidence for all clients, but it should outperform the current state-of-the-art. To do so, we want to build a tool that uses machine learning to intelligently classify the OS of a client based on the server domain names it connects to. The key to creating a sustainable tool is to avoid creating and relying on a hard-coded database.

5.1 Methodology

We make use of machine learning, Bayesian optimization, Kmeans clustering, and some concepts borrowed from bipartite graphs. The breakdown of our methodology is as follows.

1. Perform **Bayesian optimization** on the client IP and SNI fields of TLS handshakes to determine the optimal numbers of client clusters and domain clusters to form. We define *optimum* as follows.

    - The metric of success for optimization is the number of client clusters that is *strongly associated* with only a small number of (1 or 2) domain clusters. A *strong association* between a client cluster C and a domain cluster D is when the weight of the edge between C and D is at least one standard deviation higher than all other weights of the edges between C and other domain clusters. The graph and weights are defined below.

2. Perform **KMeans clustering** on clients and domain names separately using the optimal number of clusters determined by Bayesian optimization.

3. Form a **bipartite graph** with the client clusters as a set of vertices and the domain clusters as another set of vertices. An edge between a client cluster C and domain cluster D suggests that the clients in C connects to domains in D. The weight of an edge is proportional to the number of times the clients in C connected to domains in D, and inversely proportional to how widely the domains of D appear in other client clusters. That is, for an edge between a client cluster C and a domain cluster D to be weighted high, (1) D has to appear exclusively in C and (2) D has to appear frequently in C. (1) is needed because common services like 'outlook.com' may appear many times across all operating systems. (2) is needed because, for example, a client connecting to 'weather.apple.com' once might be an Android phone checking weather on Apple Weather for some reason. More specifically, weight is computed as

$$\text{weight} = \frac{\text{frequency}}{\text{non-exclusivity}}$$

    where frequency and non-exclusivity are defined as:

$$\text{frequency} = \frac{\sum_{c \in C} \text{number of domains in } D \text{ that a client } c \text{ in } C \text{ connected to}}{\text{total number of clients in } C}$$

$$\text{non-exclusivity} = \frac{\sum_{d \in D} \text{sum of weights of edges between } d \text{ and all other clients in clusters other than } C}{\text{total number of devices in } D}$$

4. For each C, identify the domain cluster D with the highest weight. Predict the OS of clients in C by either (a) establishing some ground truth for the domains in D or (b) connecting to the domains in D and identifying what they are.

5.2 Justification for methodology

Here's a few things we checked that convince us that our method will work. By 'distinct host' we mean distinct TCP fingerprints. To be considered a unique host, the fingerprints need to have exact matching **TCP header length, window scaling, maximum segment size, flags, window size, TCP options, and IP time to live**. These are the fields that commonly differ by OS.

- How many unique IPs per host?
  - Mean: 49.9
  - **Median: 2**
  - Min: 1 (most common)
  - Max: 84978
- How many unique hosts per IP?
  - Mean: 2.73
  - **Median: 2**
  - Min: 1 (most common)
  - Max: 629
- How many connections did a unique IP make during one day?
  - Mean: 501.5
  - **Median: 7**
  - Min: 1
  - Max: 22856793

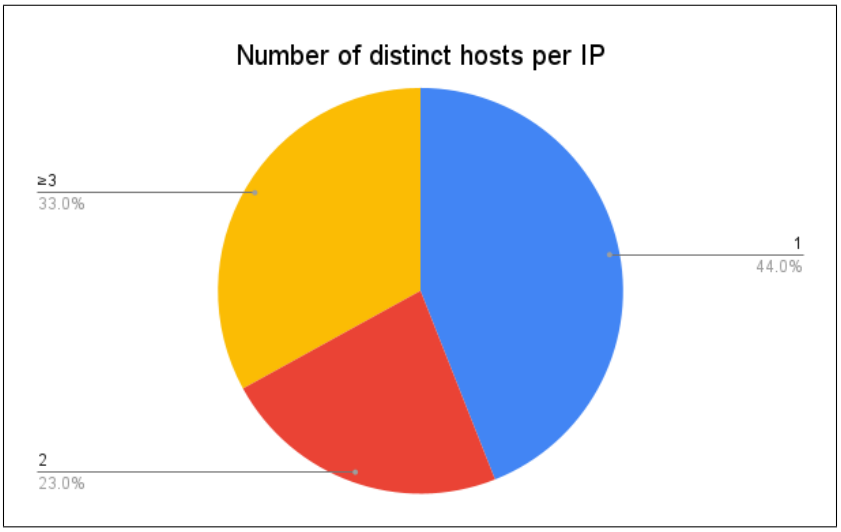

Figure 3

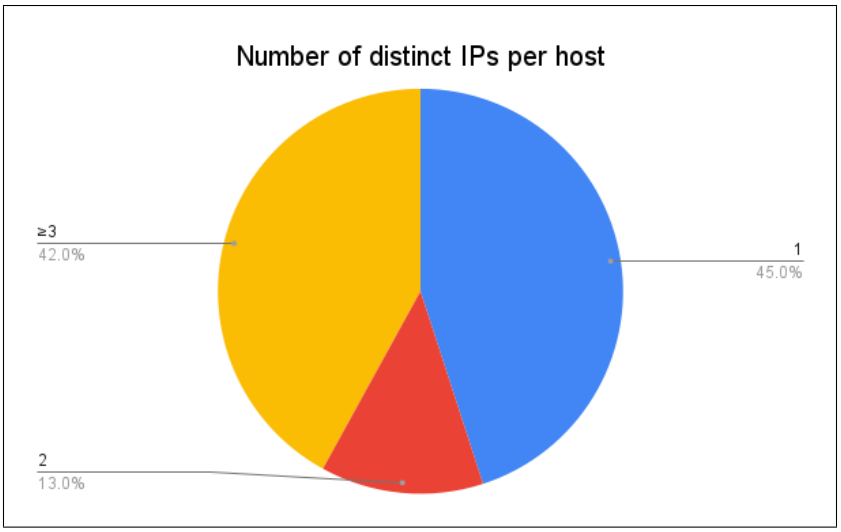

Figure 4

These statistics show that there is a rough bijection between a unique IP address and a unique host. In fact, 44% of distinct client IPs we observed were occupied by 1 distinct host, and more than 23% by 2 hosts. Also, more than 45% of distinct hosts ever only use 1 client IP, and 13% use 2 IPs. Therefore, when we identify the OS of a client IP, we can be fairly confident that it represents one distinct device. If there are conflicting evidence for multiple operating systems, we can further study how many distinct hosts are represented by the IP and identify an OS for each. Furthermore, 60% of distinct IP address makes more than 5 TLS connections and 50% makes more than 7 connections (possibly with overlapping server domains) so there are multiple server domains we can study for each host.

5.3 Results

To be populated.

6 To-do list

Here are some things we still need to figure out.

- Establishing ground truth for performance tests in the future.
- Have empirical results on how many distinct SNIs a unique client IP needs to connect to in order to have give us enough confidence in its OS identification.

References

[1] M. Latoviaka, M. HusÃ¡k, P. Velan, T. JirsÃk, and P. Aeleda, "Passive operating system fingerprinting revisited: Evaluation and current challenges," *Computer Networks*, vol. 229, p. 109782, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912862300227X

[2] Aug 2022. [Online]. Available: https://www.kali.org/tools/p0f/

[3] Cisco, "Joy." [Online]. Available: https://github.com/cisco/joy/blob/master/fingerprinting/resources/fingerprint_db.json.gz

[4] NikolaiT, "Zardaxt." [Online]. Available: https://github.com/NikolaiT/zardaxt

[5] S. Bai, H. Kim, and J. Rexford, "Passive os fingerprinting on commodity switches," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022, pp. 264–268.

[6] M. Lastovicka, T. Jirsik, P. Celeda, S. Spacek, and D. Filakovsky, "Passive os fingerprinting methods in the jungle of wireless networks," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9.