Experience Report: Using JA4+ Fingerprints for Malware Detection in Encrypted Traffic

Petr Matoušek
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
matousp@fit.vutbr.cz

Ondřej Ryšavý
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
rysavy@fit.vutbr.cz

Ivana Burgetová
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
burgetova@fit.vutbr.cz

***Abstract*—Detection of malware communications is limited due to encryption. Malware control, updates, and distribution are encapsulated in TLS tunnels, making it difficult to distinguish between malicious and benign transmissions. One way, how to detect malware communication, is to analyze the TLS handshake and obtain so-called JA4+ fingerprints. This report analyses the effectiveness of JA4+ fingerprints for malware detection, focusing specifically on the JA4, JA4S, and JA4X fingerprints and their accuracy. It examines the process of creating malware fingerprints and explores the uniqueness of these fingerprints across different malware families and their ability to distinguish between malicious and benign applications. By examining the overlap and uniqueness, the study evaluates the effectiveness of using JA4+ fingerprints to detect malware in encrypted communications.**

***Index Terms*—malware detection, TLS fingerprinting, JA4+, network monitoring, cyber security**

I. Introduction

Detecting malware in encrypted traffic is a major cyber security challenge. Traditional methods of detecting malicious activity become less effective when communications are encrypted. One way to analyze encrypted communications is through TLS fingerprinting, which identifies and tracks devices or software by analyzing the unique characteristics and patterns of the Transport Layer Security (TLS) handshake protocol, see [1], [2]. The basic idea behind TLS fingerprinting is that when negotiating TLS parameters, each side provides a unique set of security parameters that characterise the application. By obtaining TLS fingerprints of malware samples, we are able to detect malware communication by monitoring only the first packets of the TLS handshakes.

This report analyses the effectiveness of JA4+ fingerprints [3], a collection of the TLS fingerprinting methods, in detecting malware by calculating coverage and uniqueness measures on a dataset consisting of malware communications and encrypted communication instances from mobile and desktop applications. The paper presents preliminary results showing the ability of different types of fingerprints to discriminate between malicious and malware-encrypted communications, and the ability to identify individual malware families.

II. Related Work

Most approaches to malware detection in encrypted traffic use machine learning based on features extracted from the TLS handshake [4], [5], additional protocols such as HTTP and DNS [6], or flow statistics [7], [8]. Other approaches include graph-based methods [9] or neural networks [10], [11].

TLS fingerprinting methods are popular because they are easy to implement and resource-efficient. Anderson et al. [12] conducted a comprehensive study and achieved 99.6% accuracy in detecting malware by combining TLS features such as cipher suites and extensions with flow statistics. Their analysis included 18 malware families with 5,623 samples, examining cipher suites, key lengths, certificate validity, and more. They combined the TLS characteristics with flow metadata and server certificates. While effective, their approach requires more processing than simpler JA4+ fingerprinting.

In later work, Anderson and McGrew [13] used Levenshtein distance to track TLS usage trends, focusing on general patterns rather than specific applications. They analysed only the TLS Client Hello and introduced equivalence classes for destination features such as IP address and server name. Using a weighted Naive Bayes classifier for approximate fingerprint matching, they achieved an F1 score of 0.96-0.99. For malware detection, their precision was 0.63-0.99 and recall 0.64-0.88.

The reliability of TLS fingerprinting for mobile applications was investigated by Matousek et al. [14], where the authors observed the stability and accuracy of TLS fingerprints. Their work highlights the importance of data cleaning, where TLS noise (advertising, tracking and analytics services) covers about 50% of TLS communications with TLS fingerprints overlapping between applications. They also show that for mobile application identification, the best results are achieved by combining JA3 and JA3S fingerprints with SNI.

Althouse et al. [3] propose JA4/S fingerprints, claiming that they are more accurate than JA3, and that specialised fingerprints such as JA4X can identify encrypted malware communications. In this paper, we present experiments with JA4+ fingerprints to evaluate the capabilities of JA4 fingerprints for malware detection in encrypted traffic.

III. Preliminaries of the JA4+ Fingerprints

JA4+ is a collection of TLS fingerprints developed by John Althouse et al. in 2023 [3] intended to replace the JA3 fingerprints [15]. TLS fingerprinting involves extracting specific attributes from the TLS handshake and hashing them to identify applications in encrypted traffic [12].

978-3-903176-66-9 ©2024 IFIP

TLS fingerprinting identifies applications by analyzing the unique security parameters that each client and server negotiate during the TLS handshake. These parameters can create unique fingerprints for specific applications and versions. By comparing observed TLS handshakes with a database of known fingerprints, including those of common applications and malware, we can identify applications in encrypted traffic. However, because some applications share fingerprints, additional attributes such as Server Name Indication (SNI) are required for more accurate identification [14], [16].

A. JA4 and JA4S Fingerprints

JA4 and JA4S fingerprints consist of three parts a_b_c. The JA4 fingerprint includes the following TLS attributes: protocol type (TLS or QUIC), TLS handshake protocol version, sorted list of cipher suites offered by the client, sorted list of extensions, Server Name Indicator (SNI) flag, Application Layer Protocol Negotiation (ALPN), supported versions, and signature algorithms. Some of these values are inserted directly into the JA4 fingerprints, others are sorted or hashed, see Fig.1.

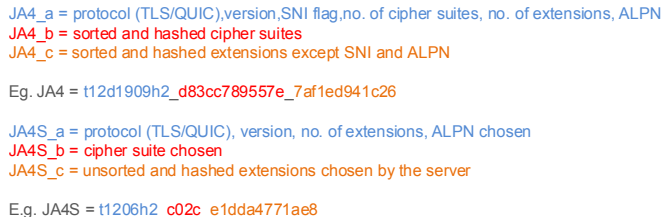


Fig. 1: Format of JA4 and JA4S fingerprints

The JA4S fingerprint represents the server-side TLS configuration and includes TLS attributes similar to the JA4 fingerprint, with the exception of SNI and signature algorithms.

B. JA4X Fingerprints

JA4X fingerprints are used to detect malware by analyzing selected attributes of X.509 certificates sent by the TLS server during the handshake [3]. Unlike JA4 and JA4S, JA4X fingerprinting focuses on how the certificate is generated, rather than its specific values. Malware authors often use the same tool to create spoofed certificates, so these certificates share a common JA4X fingerprint that reflects the generator, regardless of changes to the issuer or subject names.

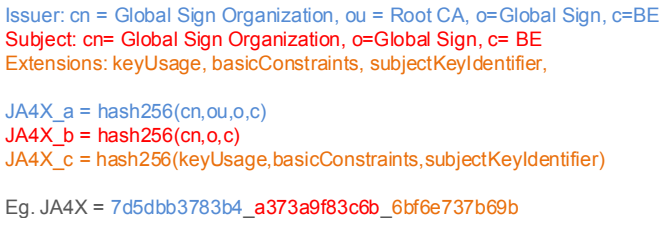


Fig. 2: Example of the JA4X fingerprinting

The JA4X fingerprinting involves observing the format of three certificate attributes: the issuer name, the subject name and a list of certificate extensions. The issuer or subject name is formally a sequence of X.500 objects called the Relative Distinguished Name (RDN), e.g., common name = ”GlobalSign Organization”, organization = ”GlobalSign”, country = ”BE”. The JA4X fingerprint takes the sequence of RDN elements, i.e., common name (cn), organization (o), country (c), without values. The elements (their OIDs) are concatenated and hashed using SHA256. The result is a part of the JA4X fingerprint. The JA4X fingerprint also consists of three parts a_b_c as shown in Fig. 2.

Note that the TLS server typically sends not only its own certificate but also certificates from parent Certificate Authorities (CAs), resulting in multiple JA4X fingerprints per TLS connection. However, in TLS 1.3 and above, certificates are encrypted, so JA4X fingerprints are not available.

Id	Desktop Malware	Id	Mobile Malware
0	agenttesla	33	3d-skin-editor-for-minecraft
1	asyncrat	34	3d-skin-editor-for-minecraft-2
2	azorult	35	alaa-win-play
3	bazarbackdoor	36	auto-click-repeater
4	darkcomet	37	bank-bingo
5	dridex	38	bitcoin-master
6	emotet	39	cashzine
7	formbook	40	colo-chess
8	gozi-ifsb	41	crazy-magic-ball
9	hawkeye	42	daily-step
10	hawkeye-reborn	43	get-rich-scanner
11	icedid	44	happy-2048
12	lokibot	45	hexa-pop-link-2048
13	masslogger	46	jelly-connect
14	matiex	47	letter-link
15	metasploit	48	macaron-boom
16	modiloader	49	macaron-match
17	nanocore	50	mega-coin-dozer
18	netwire	51	mega-wins-slot
19	njrat	52	picpro
20	pony	53	play-box
21	qakbot	54	royal-dice-party
22	qnodeservice	55	smart-walk
23	raccoon	56	star-quiz
24	remcos	57	tiler-master
25	revengerat	58	track-your-sleep
26	smokeloader	59	treasure-scanner
27	sodinokibi	60	vfly
28	trickbot	61	vibetik
29	upatre		
30	wannacry		
31	yunsip		
32	zloader		

TABLE I: List of malware families in tested dataset

IV. Testing Dataset

The dataset for the experiments was created using malware analysis sandbox for desktop malware and an Android Virtual Device (AVD) emulator for mobile malware. These tools collected network traces of isolated malware communications. To test the effectiveness of JA4+ fingerprints in identifying malware communications, the raw dataset was processed to extract TLS-related features, including JA4+ hash values. All annotated datasets are available on GitHub[1].

1) Generating Desktop Malware Fingerprints: The malware communication dataset was generated using the Tria.ge malware analysis sandbox. This framework analyses custom

[1]See https://github.com/matousp/malware-analysis [Sept 2024].

Id	Application	Id	Application	Id	Application
0	amazon-music	21	tor	42	netflix
1	amplibraryagent	22	trello	43	packeta
2	appletv	23	whatsapp	44	reddit
3	chrome	24	zoom	45	regiojet
4	firefox	25	accuweather	46	seznam
5	fournet	26	alza	47	shazam
6	gamebar	27	cestovne-poriadky	48	signal
7	hp	28	discord	49	snapchat
8	itunes	29	disneyplus	50	spotify
9	maps	30	duolingo	51	tiktok
10	messenger	31	facebook	52	tmobile
11	ms-teams	32	foodora	53	tor
12	msedge	33	gmail	54	twitter
13	msmpeng	34	idos	55	uc-browser
14	omencmdcenter	35	instagram	56	viber
15	primevideo	36	linkedin	57	whatsup
16	runtimebroker	37	mapycz	58	wolt
17	sky-go	38	messenger	59	youtube
18	skype	39	moje-vut		
19	spotify	40	mujvlak		
20	telegram	41	navlak		

TABLE II: Desktop (0-24) and mobile (25-59) applications

malware samples and provides results from previously uploaded samples, categorising each positive sample by malware family. We used the Tria.ge public API to collect samples from 33 different malware families (see Table I, left column), with 50 samples requested per family. Each sample analysis report, in JSON format, included indicators of compromise such as domain names, IP addresses, and URLs. Communication traces of malware execution were collected, but these traces included all host communication. To isolate malware communication, the traces were filtered using reported IP addresses. The final dataset, organised by the malware family, contains communication traces in PCAP files annotated with malware family names as labels for network connections.

2) Generating Mobile Malware Fingerprints: For the mobile malware analysis, we focused on dangerous and infected apps reported by sources such as McAfee and Doctor Web. We obtained APKs with infected apps, uploaded them to an Android Virtual Device (AVD) for testing, and captured network communications in PCAP files. We extracted TLS communications and filtered out unrelated TLS sessions based on SNI. The remaining TLS connections were labelled with the malware name. In total, we analysed 31 different mobile malware applications, as shown in Table I, right column.

3) Non-malware Applications: For comparison, we used an annotated dataset of common desktop and mobile applications. The dataset contained 25 desktop and 35 mobile applications, see Table II. For further analysis, we extracted TLS connections, computed JA4+ fingerprints and added annotations.

V. Fingerprints Analysis

This study focuses on properties that capture the ability to identify the malware families based on JA4 fingerprints:

- The *uniqueness property* assumes that the fingerprints should be able to distinguish between two different malware families. Two borderline cases can be identified. Firstly, malware families can share the same fingerprint, making them indistinguishable. Alternatively, no two malware families can share the same fingerprint, making each family distinguishable.
- The *stability property* requires that the fingerprints of a malware family should remain sufficiently similar between two observations to be recognisable. Malware fingerprints evolve as instances of malware within the same family change.

We first analysed the uniqueness of malware families by comparing their identified fingerprints with those of other families, assessing the ability to identify each family based on TLS fingerprints alone. We then compared malware fingerprints with those of benign applications to determine whether TLS fingerprinting can effectively distinguish between malicious and benign encrypted communications.

A. Inter-family Fingerprinting

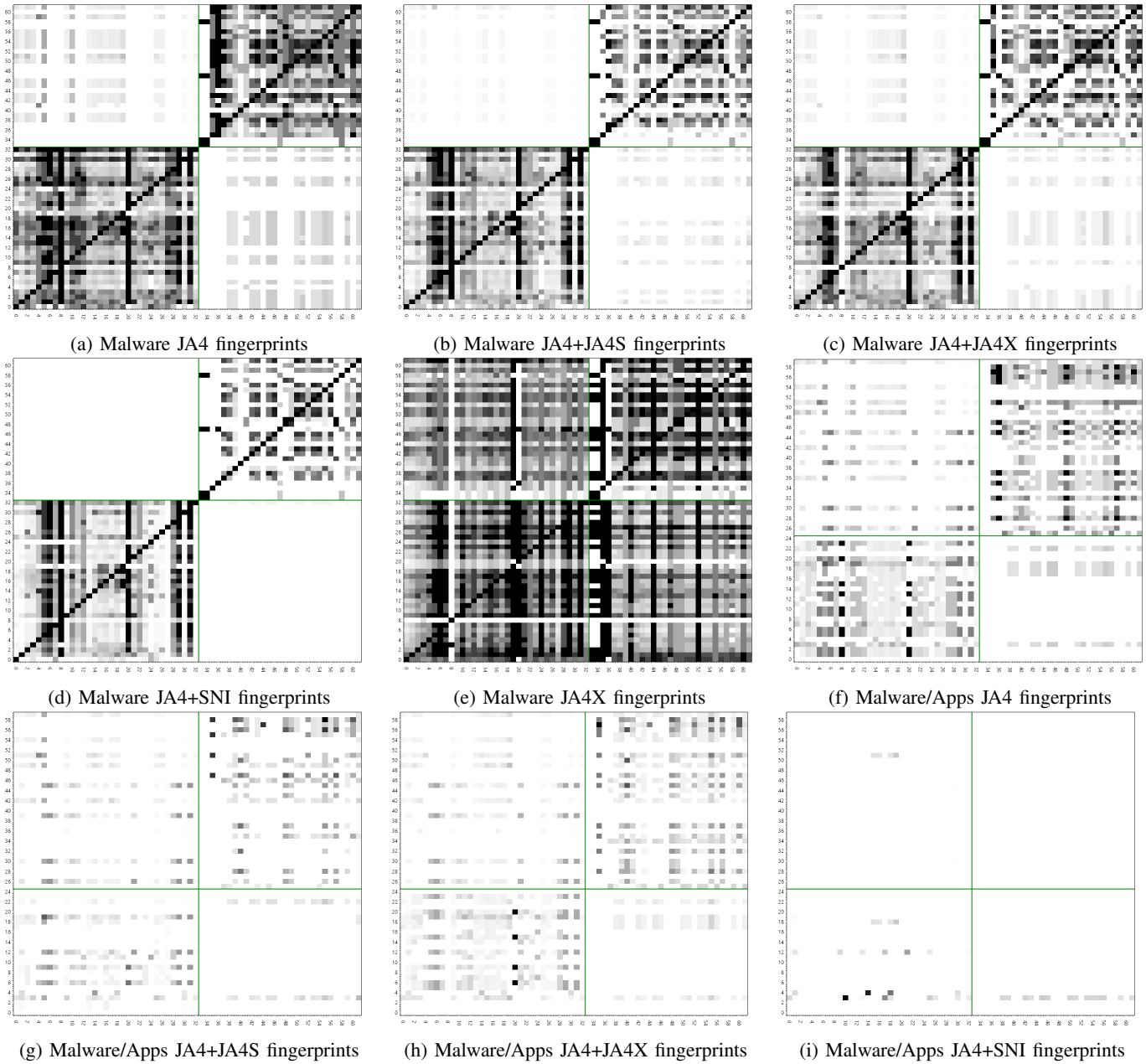
For each malware family, we collected fingerprints and identified any overlap between them, analyzing JA4 fingerprints and their combinations such as JA4+JA4S, JA4+JA4X and JA4+SNI. Figures 3(a-e) show the relationships between malware families based on shared fingerprints, with columns and rows representing different families and darker colours indicating a higher ratio of shared fingerprints. The lower left quadrant represents desktop malware relationships, the upper right shows mobile malware, and the other quadrants show cross-platform relationships.

The data shows significant overlap in fingerprints between malware families on both the desktop and mobile platforms, but minimal overlap between these two platforms. JA4 fingerprints alone can only uniquely identify a few families (see Fig. 3a). Adding JA4S (Fig. 3b) or JA4X improves the results slightly. The best improvement comes from adding SNI (Fig. 3d), but many fingerprints are still shared between different malware families. Fig. 3e illustrates the JA4X fingerprint, which is designed to detect malware using information from certificates. The figure shows that many malware families share the same JA4X fingerprints, confirming the authors' hypothesis. However, it also shows that without additional information it is difficult to distinguish between different malware families using this method alone.

B. Malware to Application Discrimination

We also explored the use of fingerprints to distinguish between malicious and benign application communication. We computed fingerprints for desktop and mobile applications and analysed the overlap with malware fingerprints. The results are shown in Figures 3(f-i), where the columns represent 62 malware families (0-61), see Table III, and the rows represent 25 desktop and 35 mobile applications (0-59), see Table IV.

In the JA4 fingerprint analysis (Figure 3f), many desktop malware families share fingerprints with desktop applications, and mobile malware shares fingerprints with mobile applications, with limited overlap between platforms. Combining client and server fingerprints (Figure 3g) or using JA4 with JA4X hashes (Figure 3h) improves results. Adding SNI to the JA4 fingerprint significantly reduces collisions (Figure 3i).



(a) Malware JA4 fingerprints (b) Malware JA4+JA4S fingerprints (c) Malware JA4+JA4X fingerprints

(d) Malware JA4+SNI fingerprints (e) Malware JA4X fingerprints (f) Malware/Apps JA4 fingerprints

(g) Malware/Apps JA4+JA4S fingerprints (h) Malware/Apps JA4+JA4X fingerprints (i) Malware/Apps JA4+SNI fingerprints

Fig. 3: Fingerprint collision matrix

C. Statistical Analysis

Next, we performed a statistical analysis of the calculated fingerprints to determine the following three characteristics:

- *Uniqueness:* represents the percentage of fingerprints that are unique to only a single application/malware, i.e., these fingerprints are not shared between applications.
- *Coverage:* is the percentage of applications/malware that has at least one unique fingerprint.
- *Efficiency:* is a measure that provides an average number of applications/malware assigned to fingerprints. Let $i = 1 \dots n$ be the number of unique fingerprints and let the function $f(i)$ map each fingerprint to an application. Then the efficiency is $E = \frac{\sum_{i=1}^{n} f(i)}{n}$. The best efficiency is represented by a value of 1, which means that all fingerprints uniquely identify the applications.

Fingerprint type	Total	Unique	Covered malware	Efficiency
JA4	119	43.9%	27.4%	4.15
JA4S	260	49.0%	33.9%	4.04
JA4+JA4S	552	59.0%	53.2%	2.52
JA4+JA4S+SNI	3053	87.0%	79.0%	1.36

TABLE III: Efficiency of TLS fingerprints on malware

Table III shows the uniqueness of different fingerprint types and their combinations on our malware dataset and the percentage of malware families they uniquely identify. TLS connections to analytics, tracking, or advertising servers (noise) have been filtered out. For JA4 fingerprints, we used as much data as possible (12,722 connections), including incom-

plete data. For other fingerprints, only complete connections (11,141) were used. The uniqueness of JA4+JA4S+SNI is about 87% and they cover almost 80% of the malware families.

Fingerprint type	Total	Unique	Covered malware	Covered apps	Efficiency
JA4	160	46.3%	21.0%	28.6%	4.17
JA4S	293	33.8%	27.4%	22.2%	4.62
JA4+JA4S	775	64.4%	50.0%	61.9%	2.30
JA4+JA4S+SNI	4032	89.3%	77.4%	95.2%	1.29

TABLE IV: Efficiency of TLS fingerprints on malware-apps

Table IV shows how the ability to identify malware families changes when desktop and mobile application fingerprints are combined. As some fingerprints are shared between malware and desktop applications, the malware coverage drops to 27.4% for JA4S, to 50.0% for JA4S+JA4S, and to 77.4% for JA4+JA4S+SNI. This is mainly caused by Android malware apps with similar JA4S fingerprints as for benign Android apps, see also Fig. 3f-3i. However, if we add SNI, the malware coverage is 77.4%. The coverage of benign applications is 95.2% and the the overall efficiency is 1.29, which is better than for malware alone in Table III.

Fingerprint type	Total	Shared
JA4	160	21.25%
JA4S	293	25.94%
JA4+JA4S	775	9.94%
JA4+JA4S+SNI	1032	1.02%
JA4X1	59	32.20%
JA4X1+JA4X2	78	29.5%
JA4X1+JA4X2+JA4X3	80	12.5%

TABLE V: Overlap of malware and application fingerprints

Finally, Table V presents the percentage of fingerprints shared by malware and application connections. Using a combination of JA4+JA4S+SNI, there is only 1.02% overlap between malware and benign applications which helps to distinguish these two groups of encrypted traffic. We can also see that JA4X fingerprints have a large overlap for the direct Certification Authority (JA4X1) and its closest parent (JA4X2). However, only 27% of the tested TLS connections contained JA4X1 fingerprints, 15.7% had JA4X2 fingerprints, and 11.95% had JA4X3 fingerprints. This shows that the combination of JA4+JA4S+SNI is more useful and effective in detecting malware than JA4X hashes.

VI. Conclusion

This report investigates the effectiveness of JA4+ fingerprints in malware detection. The experimental results show that the combination of JA4, JA4S, and SNI gives the best results for malware detection. These combined fingerprints are highly unique, covering approximately 80% of known malware families.

Given a dataset of malware samples, we can analyse malware communications and extract JA4+ fingerprints and additional TLS characteristics. The resulting database of malware JA4+ fingerprints can be used by network monitoring devices to detect malware communication in real-time.

Based on our experiments, we also demonstrated that JA4X fingerprints are of limited use, as they are present in less than a third of TLS connections, and the same JA4X fingerprints are shared between malware and benign applications.

Acknowledgment

This work is supported by the project “Flow-based Encrypted Traffic Analysis (FETA)”, 2022-2025, no. VJ02010024, funded by the Ministry of the Interior of the Czech Republic, and the internal BUT project ”Smart Information Technology for a Resilient Society”, 2023-2025, no. FIT-S-23-8209. The authors would like to thank the student Kristián Kičinka, who participated in the generation of mobile fingerprints.

References

[1] T. Dierks and E. Rescorla, “*The Transport Layer Security (TLS) Protocol Version 1.2*,” IETF RFC 5246, August 2008.

[2] E. Rescorla, “*The Transport Layer Security (TLS) Protocol Version 1.3*,” IETF RFC 8446, August 2018.

[3] J. Althouse. (2023) JA4+ Network Fingerprinting. [Online]. Available: https://blog.foxio.io/ja4+-network-fingerprinting

[4] O. Barut, M. Grohotolski, C. DiLeo, Y. Luo, P. Li, and T. Zhang, “Machine Learning Based Malware Detection on Encrypted Traffic: A Comprehensive Performance Study,” in *Proceedings of the 7th International Conference on Networking, Systems and Security*, ser. NSysS ’20. New York, NY, USA: ACM, 2020, p. 45–55.

[5] A. Mohaisen, O. Alrawi, J. Park, J. Kim, D. Nyang, and M. Mohaisen, “Network-based Analysis and Classification of Malware using Behavioral Artifacts Ordering,” *EAI Endorsed Transactions on Security and Safety*, vol. 5, no. 16, 12 2018.

[6] I. Lee, H. Roh, and W. Lee, “Poster Abstract: Encrypted Malware Traffic Detection Using Incremental Learning,” in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 1348–1349.

[7] B. Anderson and D. McGrew, “Identifying encrypted malware traffic with contextual flow data,” in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, ser. AISec ’16. New York, NY, USA: ACM, 2016, p. 35–46.

[8] O. Roques, “Detecting Malware in TLS Traffic,” Master’s thesis, IMPERIAL COLLEGE LONDON, 2019.

[9] Z. Fu, M. Liu, Y. Qin, J. Zhang, Y. Zou, Q. Yin, Q. Li, and H. Duan, “Encrypted Malware Traffic Detection via Graph-based Network Analysis,” in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, ser. RAID ’22. New York, NY, USA: ACM, 2022, p. 495–509.

[10] P. Prasse, L. Machlica, T. Pevný, J. Havelka, and T. Scheffer, “Malware detection by analysing encrypted network traffic with neural networks,” in *Machine Learning and Knowledge Discovery in Databases*, M. Ceci, J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, Eds. Cham: Springer International Publishing, 2017, pp. 73–88.

[11] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, “Identification of Encrypted Traffic Through Attention Mechanism Based Long Short Term Memory,” *IEEE Transactions on Big Data*, vol. 8, no. 01, pp. 241–252, feb 2022.

[12] B. Anderson, S. Paul, and D. McGrew, “Deciphering malware’s use of TLS (without decryption),” *Journal of Computer Virology and Hacking Techniques*, 2018.

[13] B. Anderson and D. McGrew, “TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behavior,” in *Proc. of the Internet Measurement Conference*, 2019, pp. 379–392.

[14] P. Matoušek, I. Burgetová, O. Ryšavý, and M. Victor, “On Reliability of JA3 Hashes for Fingerprinting Mobile Applications,” in *Digital Forensics and Cyber Crime, ICDF2C 2020*, ser. LNICST, vol. 351. Springer, 2021, pp. 1–22.

[15] J. Althouse. (2019) TLS Fingerprinting with JA3 and JA3S. [Online]. Available: https://medium.com/salesforce-engineering/tls-fingerprinting-with-ja3-and-ja3s-247362855967

[16] B. Anderson and D. A. McGrew, “Accurate TLS fingerprinting using destination context and knowledge bases,” *CoRR*, vol. abs/2009.01939, 2020. [Online]. Available: https://arxiv.org/abs/2009.01939