Pump Up the JARM: Studying the Evolution of Botnets using Active TLS Fingerprinting

Eva Papadogiannaki*
Foundation for Research and Technology-Hellas
epapado@ics.forth.gr

Sotiris Ioannidis†
Technical University of Crete
sotiris@ece.tuc.gr

*Also with Telecommunication Systems Research Institute †Also with Foundation for Research and Technology-Hellas

***Abstract*—The growing adoption of network encryption protocols, like TLS, has altered the scene of monitoring network traffic. With the advent and rapid increase in network encryption mechanisms, typical deep packet inspection systems that monitor network packet payload contents are gradually becoming obsolete, while in the meantime, adversaries abuse the utilization of the TLS protocol to bypass them. In this paper, aiming to understand the botnet ecosystem in the wild, we contact IP addresses known to participate in malicious activities using the JARM tool for active probing. Based on packets acquired from TLS handshakes, server fingerprints are constructed during a time period of 7 months. The fingerprints express servers' responses to a sequence of several ``TLS Client Hello'' messages with different TLS attributes and we investigate if it is feasible to detect suspicious servers and re-identify other similar within blocklists with no prior knowledge of their activities. Based on our study, we can see that fingerprints originating from suspicious servers are repetitive among similarly configured servers. We show that it is important to update fingerprints often or follow a more effective fingerprinting approach, since the overlapping ratio with legitimate servers rises over time.**

I. Introduction

After the SSL/TLS protocol introduction, the progressive shift to encrypted network communications was inevitable. For instance, the 2021 Annual Report of *Let's Encrypt* [1] announced that since 2013, HTTPS page loads have grown from 25% to 84% globally [2]. In 2019, one year after TLS 1.3 been published as an RFC [3], IETF reported that its adoption is rapidly growing with a 30% of Chrome's Internet connections to negotiate TLS 1.3 [4]. The 2021 TLS Telemetry Report published that TLS 1.3 has become the preferred protocol for 63% of the top one million web servers on the Internet [5].

In the meantime, network encryption continues to be abused by malicious actors. Again in the 2021 TLS Telemetry Report [5], we can see that the proportion of phishing sites using HTTPS and valid certificates has risen to 83%. Moreover, Command and Control (CnC) servers take advantage of DNS over HTTPS (DoH) for their communications [6], [7]. Even though network encryption is crucial for the protection of users privacy, it naturally introduces challenges for tools and mechanisms that perform packet inspection and rely heavily on the processing of packet payloads. Such operation is vital for firewalls or intrusion detection and prevention systems [8]–[10]. Typical network intrusion detection systems (NIDS), such as Snort, inspect packet headers and payloads to report malicious or abnormal traffic behavior. In TCP segments that are secured using the TLS protocol though, the only information that makes sense is (i) TLS handshake messages and (ii) TCP/IP headers (the data transmitted in packet payloads is encrypted). So, even popular intrusion detection systems seem to inadequately inspect encrypted connections. The SSL Readme page of Snort, for instance, reports that when inspecting port 443, "only the SSL handshake of each connection will be inspected" [11]. Machine learning techniques are widely used for traffic classification, network analytics and malware detection [12]–[16]. Others focus on the implementation of real-time traffic identification systems for encrypted networks [17]–[19], based on packet metadata, which when properly combined, can offer valuable insights [15].

In a TLS handshake, the first packets sent remain unencrypted and offer valuable information to traffic analysis tools, enabling fingerprinting of devices, operating systems and applications [20]. In this work we generate TLS fingerprints using JARM [21], an open-source tool for active server probing. Our goal is to provide a long-term measurement study of botnets, using CnC server information that is available in public datasets [22], [23]. The contributions of this work are the following:

1) We probe malicious servers using JARM and we construct a database of fingerprints based on the TLS handshake messages, which we wish to share.
2) We present the evolution of IP addresses that participate in botnets using the TLS fingerprints constructed.
3) This measurement study has the longest duration in time (daily measurements for 7 months) when compared to related works.

II. Background

Transport Layer Security (TLS) is an encryption protocol that is widely used to ensure the security and privacy of user communications online [3]. Specifically, TLS is used to encrypt and authenticate the communication channel between two endpoints, and it is widely adopted among others in browsing, messaging, voice over IP (VoIP) calls, emails. TLS allows endpoints to securely communicate over the Internet,

hindering any possible malicious actions like eavesdropping, tampering and forgery.

A communication session between two endpoints starts with the TLS handshake. The TLS handshake mainly kicks-off the decision-making procedure between the two endpoints, about which TLS versions, encryption ciphers and extensions they will actually use during their communication. After this handshake, the two endpoints are able to share data, which is encrypted with the arranged TLS configurations between the two endpoints. The only part of a TLS communication that is in plain sight, is the contents of the TLS handshake messages exchanged. The endpoint that wishes to initiate a communication session with another endpoint sends a ``TLS Client Hello'' message. In the ``TLS Client Hello'', the endpoint includes the TLS versions that it supports, the encryption cipher suites, a string of random bytes and a list of public keys. Then, the other endpoint responds with a ``TLS Server Hello'' message. The ``TLS Server Hello'' message contains the negotiated protocol version, the encryption cipher suit that the server wishes to use, a string of random bytes and a public key for key exchange. Then, the server sends a message with one or more certificates. After the entity, which acts as client, receives those messages, it verifies the legitimacy of the server's certificate with the authority that issued it. After the verification of the server's certificate, the communication between the two endpoints starts[1].

III. Data Collection and Preliminary Analysis

As already mentioned, we make use of the JARM tool [21], which is an active TLS server fingerprinting tool. Based on the handshake properties of TLS, JARM actively sends 10 consequent "TLS Client Hello" message to a server and collects the "TLS Server Hello" message that come as responses. The 10 consequent "TLS Client Hello" messages that are sent to the target server are specifically generated to force TLS servers to response with unique responses. To be precise, JARM sends "TLS Client Hello" messages with different TLS versions, ciphers and extensions. The "TLS Server Hello" messages, then, contain information with the server's attribute combination of TLS versions, ciphers and extensions. Hashing these 10 server's responses, we receive a JARM fingerprint.

On October 2021, we started collecting publicly available IP addresses from several blocklists. More specifically, we retrieve IP addresses from the Feodo Tracker Botnet C2 IP Blocklist [22] (CC1) and the MalSilo IPv4 feed [23] (CC2)The blocklists with identifiers CC1 and CC2 contain IP addresses, port numbers and activity/botnet name. Since the botnet lists (CC1 and CC2) contain more information, we classify the botnets based on the produced fingerprints. The fingerprints are produced using the combination of the 10 "TLS Server Hello" messages. Figures 1 and 2 show the evolution of the IP addresses and fingerprints that we collect during this study. In the following paragraphs, we set the scene with a preliminary analysis based on the data that we collect and calculate. As occurs, the different command and control servers that exist in the botnet lists that we download and parse, include the activity of the following botnets: (i) `Dridex`, (ii) `Qakbot`, (iii) `Trickbot`, (iv) `Emotet` and (v) `Downloader`. How the number of unique command and control server IP addresses evolve during our study is illustrated in Figure 1. Everyday, we download the fresh command and control server IP addresses from the botnet lists and we calculate the server fingerprints that are produced by JARM. The number of unique fingerprints per botnet is shown in Figure 2.

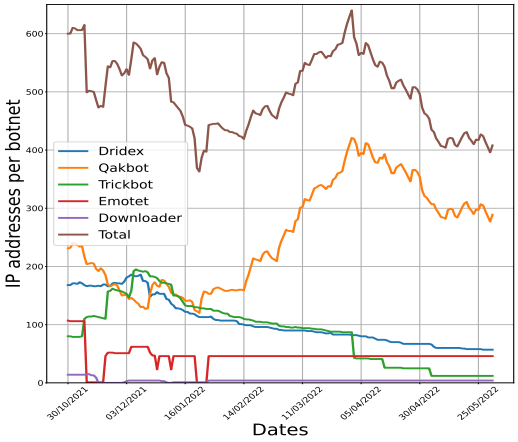

Fig. 1. The number of the unique IP addresses over time, as contained in the lists with the botnet command and control servers that we parse (i.e., CC1, CC2).

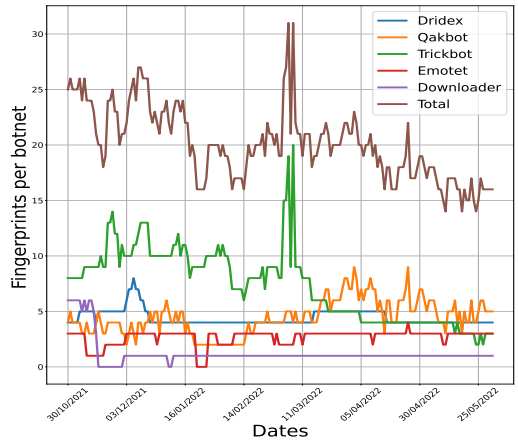

Fig. 2. The number of unique fingerprints over time, as hashed out of the 10 TLS Server Hello responses, when contacting the IP addresses contained in the botnet command and control server lists that we parse (i.e., CC1, CC2).

The number of unique Dridex command and control server IP addresses reach up to 186, while the fingerprints that are produced by actively probing those servers are maximum 8. This means that the majority of the Dridex command

[1]In this work, we utilize the information exchanged in the TLS handshake. More information can be found in [24].

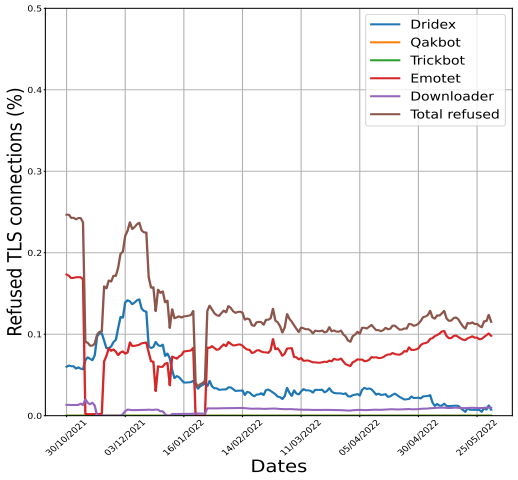

Fig. 3. The number of refused TLS connections from IP addresses contained in the botnet command and control server lists (i.e., CC1, CC2).

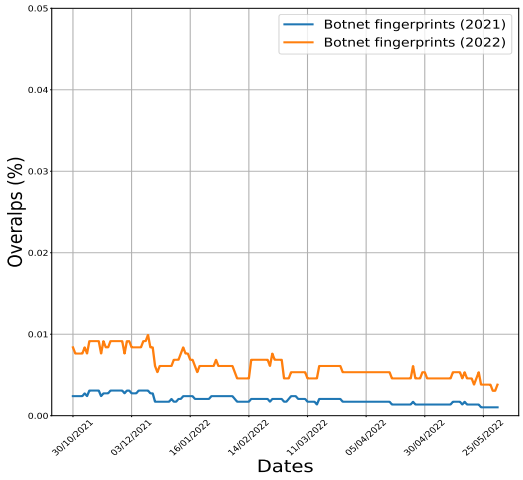

Fig. 4. TLS server fingerprints (from servers found in CC1, CC2) that overlap with servers found in the top 10K domains ([25]) in 2021 and 2022.

and control servers mostly have same TLS configurations. Likewise, the number of unique QakBot command and control server IP addresses reach up to 421, while the fingerprints that are produced by actively probing those servers are maximum 9. These number show that the fingerprints produced by the servers of this botnet are significantly uniform. For TrickBot, we encounter a maximum of 195 distinct IP addresses in a single day, resulting to 20 fingerprints. For the Emotet botnet, we encounter a maximum of 107 unique IP addresses in a single day, resulting to 4 fingerprints. The highest fingerprint diversity is introduced by the `Downloader` botnet, where for only 15 distinct IP addresses we get 6 fingerprints. Finally, our findings make us confident that maintaining a database with these fingerprints can eventually help mitigate cyber attacks related to the examined botnets.

A. *Refused TLS connections*

Refused TLS connections from servers are expected. Figure 3 shows the number of servers that refused all TLS connections that we started by not responding to the "TLS Client Hello" messages that we sent. From the figure, we see that there are times that contacted servers could refuse an incoming TLS connection. Specifically, in Figure 3 we plot the number of the refused TLS connections from known command and control servers (found in CC1, CC2). Servers from the Qakbot and Trickbot botnets mostly accept our connections and respond to the "TLS Client Hello" messages. Downloader servers also respond with a high ratio. Dridex and Emotet are the botnets that refuse incoming connections more frequently.

B. *Fingerprints of benign servers*

We actively contacted the top-10K benign servers from the The Majestic Million list [25] one day in October 2021 and one day in October 2022. Results presented in Figure 4 show the daily overlaps of the botnet servers with the benign servers of the top-10K from the Majestic list [25][2]. The TLS server fingerprints that are produced using the IP addresses found in the botnet lists result to less than 1% overlapping fingerprints. One year later, we observe a rise in the overlaps. The overlaps of botnet server are less than 5%, but higher when compared to the overlaps from 2021. The unique fingerprints of the legitimate servers (top-10K Majestic) are 2915 in 2021, while in 2022 the unique fingerprints of the legitimate servers (top-10K Majestic) are 1311. The overlaps of malicious server fingerprints with legitimate server fingerprints are 371 in 2021 (13%) and 534 in 2022 (40%). Based on our measurements, we can see that in a year, the variation in TLS configurations of legitimate servers is reduced (i.e., less unique fingerprints). Furthermore, based on Figure 4, it is clear that as time passes, the calculated fingerprints of malicious servers are becoming less effective, when not updated.

IV. Analysis

In this section, we analyse our findings based on IP addresses that we have been contacting for 7 months, and share our insights.

A. *Botnet ports*

For each established connection, the port number used can sometimes signify the underlying activity. Aiming to shed some light in the port selection by known botnets, Table I contains the top-5 ports that we encounter in the blocklists together with the IP addresses. It is really interesting to notice that except for the well known ports (e.g., 443, 8080), the different botnets operate using a diverse list of port numbers. Such ports, along with the fingerprints produced, can offer better accuracy results in the performance of a network monitoring tool.

[2]The lists of domains were downloaded in October 2021 and 2022, with their fingerprints calculated in October 2021 and 2022, respectively.

TABLE I
MOST POPULAR PORT NUMBERS PER BOTNET (TOP-5 IN CC1, CC2).

Botnet name	Popular ports (descending order)	Total ports
Dridex	443, 7443, 4664, 10172, 6225	45
QakBot	443, 995, 2222, 993, 1194	26
TrickBot	443, 447, 449	3
Emotet	8080, 443, 80, 7080, 4001	10
Downloader	6602, 1973, 13786, 29795, 46187	11

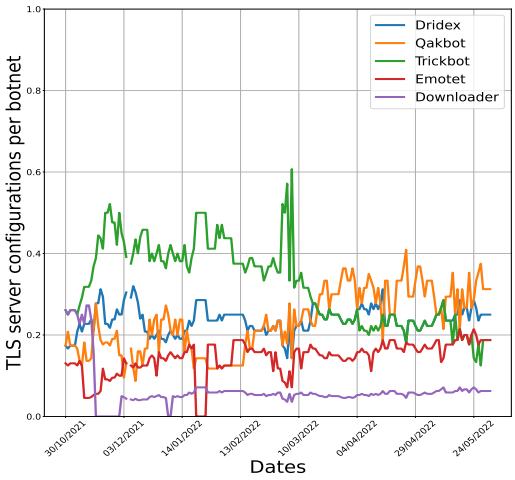

Fig. 5. The number of unique TLS server configurations from IP addresses contained in the botnet command and control server lists (i.e., CC1, CC2).

B. TLS Server Configurations

Each fingerprint that is calculated by the JARM tool for a single server is comprised of 62 Bytes in total. The first 30 Bytes present information about the configurations (i.e., TLS version and ciphers) of the server contacted and the remaining 32 Bytes contain information about the extensions. This means that for two different fingerprints that share the same first 30 Bytes, they also share the same configuration of TLS versions and supported ciphers. We believe that it would be very interesting and insightful to examine if there are trends between the servers that participate in the same botnet activity. Then, studying if these trends are distinct between different botnets, we might be able not only to tell if a server looks malicious, but to also indicate the botnet that the server is part of. Figure 5 shows that concerning the different botnets that we study, the TLS configurations of the servers are always less than the actual server fingerprints. This means that the servers that we probed in order to calculate their fingerprints based on the TLS Server Hello message sent, share very similar TLS configurations. To explore the uniqueness of the fingerprints produced, we check for overlapping fingerprints (length of 30Bytes) between the different servers that act as part of a certain botnet family. Table II shows the fingerprints that are common among the different botnets. Fingerprint `2ad2ad0002ad2ad0002ad2ad2ad2ad` is the most common and is produced by servers that are known to participate in Dridex, TrickBot and Emotet botnets. The

TABLE II
OVERLAPPING FINGERPRINTS BETWEEN DIFFERENT BOTNETS.

Botnet 1	Botnet 2	Overlapping Fingerprint
Dridex	TrickBot	`2ad2ad0002ad2ad0002ad2ad2ad2ad`
Dridex	Emotet	`2ad2ad0002ad2ad0002ad2ad2ad2ad`
QakBot	TrickBot	`20d08b20d21d20d20c42d08b20b41d`
QakBot	TrickBot	`2ad2ad16d2ad2ad22c2ad2ad2ad2ad`
QakBot	Downloader	`04d02d00004d04d04c04d02d04d04d`
QakBot	Downloader	`20d08b20d21d20d20c42d08b20b41d`
TrickBot	Emotet	`2ad2ad0002ad2ad0002ad2ad2ad2ad`

TABLE III
BOTNET FINGERPRINTS FOUND IN BLOCKLISTS BL1 AND BL2.

Botnet	Fingerprints	In CI-Badguys	In Blocklist.de
Dridex	9	9	8
QakBot	36	22	25
TrickBot	47	14	14
Emotet	7	7	7
Downloader	7	6	7

total number of unique fingerprints that overlap between different botnets is only 4. Removing the 4 *most common fingerprints* that represent the most common TLS configurations among servers, significantly decreases the number of fingerprint overlaps with benign servers that are included in the Majestic Million domains (causing a 20% decrease in numbers of overlaps shown in Fig. 4). This could mean that the malicious servers with the specific fingerprint imitate a popular TLS configuration profile used by normal and benign TLS servers (a technique that censorship circumvention tools also follow [26]).

Aiming to search for the botnets' fingerprints that are calculated, we collect also the IP addresses from two other blocklists: the CINS Score CI-Badguys list [27] (BL1) and the blocklists.de list [28] (BL2). Each blocklist is downloaded everyday and we calculate the fingerprints for each IP address. Altogether, we can find many fingerprints obtained from botnets that exist inside the set of fingerprints calculated for the two blocklists (i.e., BL1, BL2). The aggregated numbers of those fingerprints are shown in Table III. This makes us confident that it is possible to distinguish servers of different botnets in a vast list of IP addresses without any prior knowledge, based only on the fingerprints calculated by JARM.

Indeed, the presence of the fingerprints that are calculated by the servers that participate in each botnet exists in the list of fingerprints that are produced by servers in the two blocklists. Figures 6 and 7 illustrate the proportion of fingerprints extracted from IP addresses that are contained in the botnet command and control server lists that were found between the fingerprints extracted from IP addresses contained in the blocklists. In Figures 6 and 7, we search the botnet fingerprints of full length against the fingerprints of the blocklists. For instance, on the 11th of November 2021 we extracted 372 fingerprints from servers found in Blocklist.de and 348 fingerprints from servers found in CI-Badguys list. Out of the total Blocklist.de fingerprints, 3 of them matched those of Dridex, 2 of them matched those of QakBot, 7 of them matched those of TrickBot, 1 of them matched those of Emotet and 3 of them matched those of Downloader. Out of the total

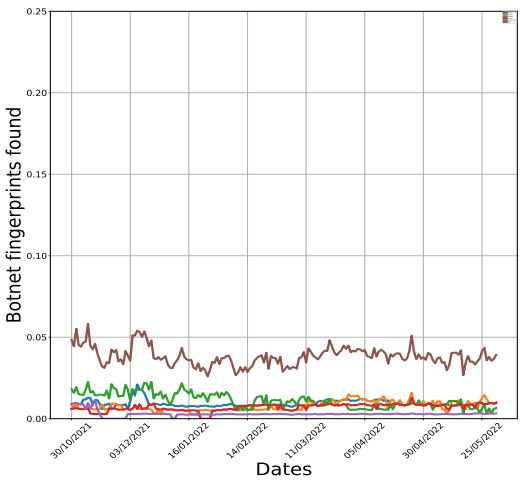

Fig. 6. The proportion of botnet fingerprints found into the list of fingerprints calculated from IP addresses contained in BL1.

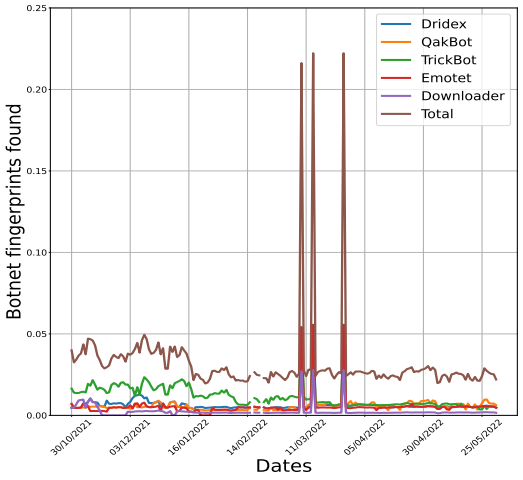

Fig. 7. The proportion of botnet fingerprints found into the list of fingerprints calculated from IP addresses contained in BL2.

CI-Badguys fingerprints, 4 of them matched those of Dridex, 2 of them matched those of QakBot, 5 of them matched those of TrickBot, 1 of them matched those of Emotet and 4 of them matched those of Downloader.

The first 30 bytes of each fingerprint calculated by JARM provides information about the TLS versions and ciphers that a server supports. Servers that share the same 30 first bytes in their fingerprints, also share the same TLS configuration (i.e., TLS versions and ciphers). As expected, searching for the 30 first bytes of the initial fingerprints results to higher number of botnet occurrences into the blocklists. Furthermore, extensions shared via the TLS Server Hello message could add turbulence in fingerprints, since the application of popular fingerprinting circumvention techniques, like extension randomization, are common from servers with malicious activities (or censorship circumvention tools [26]). The existence of malicious servers that are similarly configured is highly expected. Thus, searching for these configurations could assist in identification of malicious infrastructure. On the 11th of November 2021 we extracted 372 fingerprints from servers found in Blocklist.de and 348 fingerprints from servers found in CI-Badguys list. Out of the total Blocklist.de fingerprints, 14 of them matched those of Dridex, 3 of them matched those of QakBot, 45 of them matched those of TrickBot, 1 of them matched those of Emotet and 4 of them matched those of Downloader. Out of the total CI-Badguys fingerprints, 17 of them matched those of Dridex, 6 of them matched those of QakBot, 32 of them matched those of TrickBot, 1 of them matched those of Emotet and 9 of them matched those of Downloader. These numbers show a significant rise when we study the coverage of the examined botnets into the two blocklists, a trend that remains during the whole period of our analysis, with a peak of 30% total coverage on the 13th of December 2021 in both blocklists.

Diving into more depth, we further analyzed the data collected during the 7-month period. More specifically, for each IP address that we contact, we process the server responses to each TLS Client Hello that is sent. Dridex, QakBot, TrickBot and Emotet botnets select TLS 1.2 and 1.3, while Downloader also accepts TLS 1.1. Besides the TLS version chosen by the botnet servers, we also analyzed their preferred cipher suite.

Table IV presents the codes of the cipher suites that each botnet selects in most cases. In Table V we present a dictionary for these codes, along with their ranking by *ciphersuite.info* [29]. *Weak* ciphers are considered old and they should be avoided, while *insecure* ciphers can be broken with minimum effort. Dridex and QakBot select cipher suites marked as weak. In some cases, QakBot selects also an insecure cipher suite. TrickBot, Emotet and Downloader select a combination of weak and secure ciphers.

TABLE IV
MOST SELECTED CIPHER SUITES PER BOTNET (CC1, CC2).

Botnet name	**Cipher suites (codes)**
Dridex	c013, 00c0, 0016, 0084, c012
QakBot	000a, 0005, c012, 006b
TrickBot	c030, c014
Emotet	c030, c012, cca8, 009e, 0016, 1302, c02f
Downloader	c012, c013, 0035, 1302

V. RELATED WORK

Researchers started proposing techniques to gain insight on the nature of the underlying network traffic [30]. Machine learning is widely used to test the feasibility of encrypted traffic analysis, using network packet metadata and not payload contents [14], [15], [31]–[36]. Other works focus on real-time traffic inspection based on network metadata patterns [17], [37]–[39]. Other works that are able to process encrypted traffic perform searchable encryption, enabling middleboxes to gain insight from the exchanged traffic nature [40]–[42].

TABLE V
CIPHER SUITES DICTIONARY AND CHARACTERIZATION [29].

Code	Cipher suite name	Marked as
c013	TLS ECDHE RSA WITH AES 128 CBC SHA	Weak
00c0	TLS RSA WITH CAMELLIA 256 CBC SHA256	Weak
0016	TLS DHE RSA WITH 3DES EDE CBC SHA	Weak
0084	TLS RSA WITH CAMELLIA 256 CBC SHA	Weak
c012	TLS ECDHE RSA WITH 3DES EDE CBC SHA	Weak
000a	TLS RSA WITH 3DES EDE CBC SHA	Weak
0005	TLS RSA WITH RC4 128 MD5	Insecure
006b	TLS DHE RSA WITH AES 256 CBC SHA256	Weak
c030	TLS ECDHE RSA WITH AES 256 GCM SHA384	Secure
c014	TLS ECDHE RSA WITH AES 256 CBC SHA	Weak
cca8	ECDHE RSA WITH CHACHA20 POLY1305 SHA256	Secure
009e	TLS DHE RSA WITH AES 128 GCM SHA256	Secure
1302	TLS AES 256 GCM SHA384	Secure
c02f	TLS ECDHE RSA WITH AES 128 GCM SHA256	Secure
0035	TLS RSA WITH AES 256 CBC SHA	Weak

TLS fingerprinting is a common technique that assists in the extraction of meaningful observations from TLS handshake messages, since limited content can be revealed from encrypted data packets. TLS fingerprinting has been used for studying the TLS deployment [43] and the TLS usage in consumer IoT devices [20] and Android applications [44]. In [45], authors publish a knowledge database consisting of TLS fingerprints (passively constructed) from enterprise deployments and malware traffic, together with an analysis of trends concerning the utilization of TLS by applications and malware. Yet, this knowledge database has not been updated since 2019. TLS fingerprinting has been also used in identifying known censorship circumvention tools, like Tor [26]. JA3 is a method that enables TLS client profiling, while JA3S enables server side TLS fingerprinting [46]. S ince JA3 and JA3S hashes (or JA3/JA3S fingerprints) can be easily distributed, they promote the easy cyberthreat intelligence exchange. In fact, support to JA3S has been added in Suricata [9]. Another library that enables TLS handshake fingerprinting is fingeprinTLS [47], [48]. In a nutshell, the produced fingerprints are a combination of TLS version, accepted ciphers, extensions, elliptic curves and elliptic curve formats.

Besides these passive techniques for TLS fingerprinting, there is JARM [21]. JARM enables *active* TLS server fingerprinting. In our work, we actively contact servers with suspicious activity, with IP addresses found in publicly available lists (the lists get updated on a daily basis and we download the updated lists everyday), using JARM. In an article posted online on October 2020 [49], JARM creators explain how it works and how it can be used to identify malicious servers. In the article, creators made public 1 fingerprint per botnet. The botnets that were examined are TrickBot, AsyncRAT, Metasploit, Cobalt Strike and Merlin C2. As authors state, they contact IP addresses on port 443. In this work, we utilize JARM to contact IP addresses of servers with malicious activity (their malicious activity is known and published in several popular blocklists). Except for port 443, we also contact these IP addresses on other ports as well, based on the botnet lists records that we retrieve (more information for the lists that we examine can be found in Section III). The botnets investigated are Dridex, QakBot, TrickBot, Emotet and Downloader (TrickBot and Emotet are two of the most frequent malware variants delivering ransomware [5]). We search for the produced fingerprints against two different blocklists (BL1 and BL2) and we are able to identify the five botnets within them (based on the fingerprints produced). We provide an analysis of the fingerprints produced by these botnet servers within a time-span of 7 months (October 2021 – May 2022).

Recently, Sosnowski et al. proposed a new TLS fingerprinting technique [50]. Authors use a binary classifier to investigate if a server is a command and control server, with better precision and recall results when compared to JARM. In addition, authors study *weekly* snapshots in a period of 7 months. In DissecTLS [51], authors follow a more exhaustive fingerprinting approach to calculate more effective fingerprints, which outperform five popular TLS scanners (including JARM). They perform a measurement study using the same public datasets that we use, and they repeat the measurements nine times in a period of nine weeks. In our paper, we study *daily* snapshots of blocklisted server IP addresses in a period of 7 months and we present the evolution of their JARM fingerprints in detail. The goal of our work is to study the evolution of known command and control servers obtained by public blocklists. When compared to other measurement studies that perform TLS fingerprinting, our work produces a significantly larger dataset, which we aim to share upon request.

VI. CONCLUSION

In this paper, we explore how the examined botnets evolve in time and if the fingerprint overlapping with legitimate servers is low enough for this solution to be viable and practical. By actively contacting IP addresses of known command and control servers using JARM, we create a database of TLS server fingerprints grouped by botnet. We show that keeping an outdated list of fingerprints can cause false positives when characterizing malicious servers on the Internet. Investigating the existence of those fingerprints into blocklists of maliciously acting IP addresses, we are able to (re-)identify the same TLS server configurations that could indicate specific botnet activity, based on our knowledge base. Finally, we wish to share all of our TLS fingerprints database with security researchers and students upon request (daily data collection for 7 months).

ACKNOWLEDGEMENTS

This work was supported by the projects SENTINEL and AI4HealthSec funded by the European Commission under Grant Agreements No. 101021659 and 883273. This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which maybe made of the information contained therein.

REFERENCES

[1] "Let's Encrypt," https://letsencrypt.org, 2020, Accessed: 2020-10-29.
[2] "2021 Internet Security Research Group Annual Report," https://www.abetterinternet.org/documents/2021-ISRG-Annual-Report.pdf, 2021, Accessed: 2022-07-15.
[3] "The Transport Layer Security (TLS) Protocol Version 1.3," https://tools.ietf.org/html/rfc8446, 2018, Accessed: 2020-10-29.

[4] "TLS 1.3: One Year Later," https://www.ietf.org/blog/tls13-adoption/, 2019, Accessed: 2020-10-29.

[5] "The 2021 TLS Telemetry Report," https://www.f5.com/labs/articles/threat-intelligence/the-2021-tls-telemetry-report, 2021, Accessed: 2022-07-15.

[6] "Spamhaus Botnet Threat Update: Q4-2021," https://www.spamhaus.org/news/article/817/spamhaus-botnet-threat-update-q4-2021, 2021, Accessed: 2022-07-15.

[7] N. P. Hoang, M. Polychronakis, and P. Gill, "Measuring the accessibility of domain name encryption and its impact on internet filtering," in *International Conference on Passive and Active Network Measurement*. Springer, 2022, pp. 518–536.

[8] "The Snort IDS/IPS," Available: https://www.snort.org/, 2020, Accessed on Oct. 8, 2020.

[9] "Suricata Open Source IDS / IPS / NSM engine ," Available: https://www.suricata-ids.org/, 2020, Accessed on Oct. 8, 2020.

[10] "The Zeek Network Security Monitor," Available: https://www.zeek.org/, 2020, Accessed on Oct. 8, 2020.

[11] "Snort: README.ssl," Available: https://www.snort.org/faq/readme-ssl, 2020, Accessed on Oct. 28, 2020.

[12] K. Tam, S. J. Khan, A. Fattori, and L. Cavallaro, "Copperdroid: automatic reconstruction of android malware behaviors." in *Proceedings of The Network and Distributed System Security Symposium (NDSS)*, 2015.

[13] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, pp. 1–14, 2017.

[14] N. Rosner, I. B. Kadron, L. Bang, and T. Bultan, "Profit: Detecting and quantifying side channels in networked applications." in *Proceedings of The Network and Distributed System Security Symposium (NDSS)*, 2019.

[15] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1723–1732.

[16] J. Liang, W. Guo, T. Luo, H. Vasant, G. Wang, and X. Xing, "Fare: Enabling fine-grained attack categorization under low-quality labeled data," in *Proceedings of The Network and Distributed System Security Symposium (NDSS)*, 2021.

[17] E. Papadogiannaki, C. Halevidis, P. Akritidis, and L. Koromilas, "Otter: A scalable high-resolution encrypted traffic identification engine," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 315–334.

[18] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing android encrypted network traffic to identify user actions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, 2016.

[19] J. Bushart and C. Rossow, "Padding ain't enough: Assessing the privacy guarantees of encrypted dns," in *10th USENIX Workshop on Free and Open Communications on the Internet*, 2020.

[20] M. T. Paracha, D. J. Dubois, N. Vallina-Rodriguez, and D. Choffnes, "Iotls: understanding tls usage in consumer iot devices," in *Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference*, 2021, pp. 165–178.

[21] "JARM: An active Transport Layer Security (TLS) server fingerprinting tool," https://github.com/salesforce/jarm, 2020, Accessed: 2022-1-23.

[22] "The Feodo Tracker Botnet C2 IP Blocklist," https://feodotracker.abuse.ch/downloads/ipblocklist.csv, 2021, Accessed: 2022-1-03.

[23] "The MalSilo IPv4 feed," https://malsilo.gitlab.io/feeds/dumps/ip_list.txt, 2021, Accessed: 2022-1-03.

[24] "The Illustrated TLS 1.3 Connection," https://tls13.xargs.org, 2022, Accessed: 2023-3-30.

[25] "The Majestic Million," https://downloads.majestic.com/majestic_million.csv, 2021, Accessed: 2022-1-03.

[26] S. Frolov and E. Wustrow, "The use of tls in censorship circumvention." in *Proceedings of The Network and Distributed System Security Symposium (NDSS)*, 2019.

[27] "The CINS Score CI-Badguys list ," https://cinsscore.com/list/ci-badguys.txt, 2021, Accessed: 2022-1-03.

[28] "blocklist.de," https://lists.blocklist.de/lists/, 2021, Accessed: 2022-1-03.

[29] "TLS Ciphersuite Search," https://ciphersuite.info, 2022, Accessed: 2022-10-31.

[30] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Comput. Surv.*, vol. 54, no. 6, 2021.

[31] B. Anderson, A. Chi, S. Dunlop, and D. McGrew, "Limitless http in an https world: Inferring the semantics of the https protocol without decryption," in *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, 2019, pp. 267–278.

[32] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[33] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for https and quic," in *INFOCOM*. IEEE, 2018, pp. 1331–1339.

[34] S. Xu, S. Sen, and Z. M. Mao, "Csi: inferring mobile abr video adaptation behavior under https and quic," in *Proceedings of the 15th European Conference on Computer Systems*, 2020, pp. 1–16.

[35] V. Ghiëtte, H. Griffioen, and C. Doerr, "Fingerprinting tooling used for ssh compromisation attempts," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses*, 2019, pp. 61–71.

[36] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[37] E. Papadogiannaki, D. Deyannis, and S. Ioannidis, "Head (er) hunter: Fast intrusion detection using packet metadata signatures," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2020, pp. 1–6.

[38] E. Papadogiannaki and S. Ioannidis, "Acceleration of intrusion detection in encrypted network traffic using heterogeneous hardware," *Sensors*, vol. 21, no. 4, p. 1140, 2021.

[39] E. Papadogiannaki, G. Tsirantonakis, and S. Ioannidis, "Network intrusion detection in encrypted traffic," in *2022 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2022, pp. 1–8.

[40] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: Deep packet inspection over encrypted traffic," *ACM SIGCOMM Computer communication review*, vol. 45, no. 4, pp. 213–226, 2015.

[41] D. Naylor, R. Li, C. Gkantsidis, T. Karagiannis, and P. Steenkiste, "And then there were more: Secure communication for more than two parties," in *Proceedings of the International Conference on emerging Networking EXperiments and Technologies*. ACM, 2017, pp. 88–100.

[42] S. Canard, A. Diop, N. Kheir, M. Paindavoine, and M. Sabt, "Blindids: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 561–574.

[43] P. Kotzias, A. Razaghpanah, J. Amann, K. G. Paterson, N. Vallina-Rodriguez, and J. Caballero, "Coming of age: A longitudinal study of tls deployment," in *Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference*, 2018, pp. 415–428.

[44] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying tls usage in android apps," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, 2017, pp. 350–362.

[45] B. Anderson and D. McGrew, "Tls beyond the browser: Combining end host and network data to understand application behavior," in *Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference*, 2019, pp. 379–392.

[46] "JA3: A method for profiling SSL/TLS Clients," https://github.com/salesforce/ja3, 2018, Accessed: 2022-1-23.

[47] "FingerPrinTLS," https://github.com/LeeBrotherston/tls-fingerprinting/tree/master/fingerprintls, 2016, Accessed: 2022-1-23.

[48] X. Li, B. A. Azad, A. Rahmati, and N. Nikiforakis, "Good bot, bad bot: Characterizing automated browsing activity," in *2021 IEEE symposium on security and privacy (sp)*, 2021, p. 17.

[49] "Easily Identify Malicious Servers on the Internet with JARM," https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a/, 2020, Accessed: 2022-7-10.

[50] M. Sosnowski, J. Zirngibl, P. Sattler, G. Carle, C. Grohnfeldt, M. Russo, and D. Sgandurra, "Active tls stack fingerprinting: Characterizing tls server deployments at scale."

[51] M. Sosnowski, J. Zirngibl, P. Sattler, and G. Carle, "Dissectls: A scalable active scanner for tls server configurations, capabilities, and tls fingerprinting," in *Passive and Active Measurement: 24th International Conference, Proceedings*. Springer, 2023, pp. 110–126.