TELESOFT

FLOWPROBE: TBPS THREAT VISIBILITY

2021

Abstract

There are many challenges faced when operating at CSP or internet backbone level. The traffic flowing across the key data routes between large, interconnected computer networks and core routers of the internet is vast and complex and ensuring a consistent QoS is being delivered to customers requires comprehensive visibility across the entire digital estate.

Data rates on these networks continue to increase day by day, with home broadband speeds increasing from Kbps dial up several years ago, to Gbps fibre optic broadband providing access to millions of individuals and organisations every day and enabling businesses to operate effectively through global interconnectivity.

And due to the constant demand for new smart and IoT devices both in our everyday lives and pan industry, as well as the increasing number of global users of the internet, the requirement for data will continue to increase, with more traffic being generated and lower latency being required. As of 2019, the number of internet users worldwide stood at 4.13 billion, over half the world's population, showing just how far reaching this capability is and how integrated it is in our daily lives.

However, amongst the vast volumes of complex traffic seen today, threat actors continue to operate and protecting assets within your digital estate when operating on internet backbones requires not only comprehensive visibility, but sophisticated utilisation of modern technologies and standards in order to provide the SOC teams with the information they require, in the shortest amount of time.

Without this visibility, CSPs and backbone providers can face poor network performance, often resulting in a poor Quality of Service (QoS) for the end user, as well as increased costs and, in some cases, databases being breached.

Figure [1]: Interconnected devices across a cityscape.

Introduction: Threats in encrypted traffic

Ensuring data is secure and protected has been a focus of information security for many years, ensuring that the Confidentiality, Integrity and Availability (CIA) of the data remains uncompromised. Over the years protocols such as Secure Socket Layer (SSL) and Transport Layer Security (TLS) have been created and implemented on a global scale, underpinning the foundations of the essential protocols being utilised in our daily communications.

In 1996 and 1999 respectively, to offer encryption for the application layer. The main use of TLS today is to encrypt HTTP traffic with which it forms the HTTPS protocol. Today, almost all web traffic is encrypted with TLS and the latest version, TLS v1.3, which was released in August 2018 which included updates to improve the security and speed of the protocol.

While encryption in the TLS protocol is an essential protection against the exposure of our personal information, this also offers those with malicious intent the ability to hide behind the encryption to instigate an attack. This piece describes the problem and presents JA3 fingerprinting as an essential part of your network security arsenal for helping to identify and protect against malware and other threats whilst maintaining the privacy and integrity of network communications.

JA3 and JA3S overview

Most malicious traffic now incorporates encrypted communications to conduct attacks. Emotet and TrickBot are both known to utilise HTTP secure transmissions (TLS) at some stage during their campaign, generally to provide an additional layer of obfuscation during malware delivery and C&C activity, in an attempt to decrease the amount of visibility available to a network IDS.

However, JA3/JA3S hashing can make up for that loss by supplying a metric in the form of a TLS fingerprint, a unique identifier generated during the TLS Handshake between devices, which can support identification of compromised devices, botnets and C&C activity at the network level. By utilising the information generated during TLS negotiations, JA3 can extract information from the Client Hello (JA3) and Server Hello (JA3S) packets which are indicative of the libraries and methods used to establish the connection. These unique fields remain consistent throughout each new connection and by generating a fingerprint, represented in the form of a hash, it can bypass any need to know lists of entries used for C&C IPs or domains that are ever changing through Domain Generation Algorithms (DGA)/Fast Fluxing techniques.

A fingerprint is generated by extracting the decimal value from the following TLS fields and creating a MD5 hash for both the Client and the Server:

- **JA3 (Client Hello)** – TLSVersion,Ciphers,TLSExtensions,EllipticCurves,EllipticCurvePointFormats
- **JA3S (Server Hello)** – TLSVersion,Cipher,TLSExtensions

JA3 hashing process

Source	Source Port	Destination	Destination Port	Host	Server Name	Info
10.9.30.101	64257	45.159.115.191	80	pershel.com		GET /wp-content/Arp/ HTTP/1.1
45.159.115.191	80	10.9.30.101	64257			80 → 64257 [ACK] Seq=1 Ack=77 Win=64240 Len=0
45.159.115.191	80	10.9.30.101	64257			HTTP/1.1 301 Moved Permanently (text/html)
10.9.30.101	64258	45.159.115.191	443			64258 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
10.9.30.101	64257	45.159.115.191	80			64257 → 80 [ACK] Seq=77 Ack=380 Win=63861 Len=0
45.159.115.191	443	10.9.30.101	64258			443 → 64258 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
10.9.30.101	64258	45.159.115.191	443			64258 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
10.9.30.101	64258	45.159.115.191	443		pershel.com	Client Hello
45.159.115.191	443	10.9.30.101	64258			443 → 64258 [ACK] Seq=1 Ack=174 Win=64240 Len=0
45.159.115.191	443	10.9.30.101	64258			Server Hello

Figure [2]: Packet capture showing the TLS Handshake between compromised device (10.9.30.101) and the malware delviery server (45.159.115.191).

During this exchange, several values are shared between the client and the server. These values can be identified when looking deeper into the packet, as shown in Figure 3.

```
> Frame 348: 227 bytes on wire (1816 bits), 227 bytes captured (1816 bits)
> Ethernet II, Src: HewlettP_1c:47:ae (00:08:02:1c:47:ae), Dst: Netgear_b6:93:f1 (20:e5:2a:b6:93:f1)
> Internet Protocol Version 4, Src: 10.9.30.101, Dst: 45.159.115.191
> Transmission Control Protocol, Src Port: 64258, Dst Port: 443, Seq: 1, Ack: 1, Len: 173
v Transport Layer Security
  v TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 168
    v Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 164
        Version: TLS 1.2 (0x0303)
      > Random: 5f74b70b0ec120f6eef615f686b95b26c3eca1cdd7d38f426b95c02da17e3a39
        Session ID Length: 0
        Cipher Suites Length: 42
      > Cipher Suites (21 suites)
        Compression Methods Length: 1
      > Compression Methods (1 method)
        Extensions Length: 81
      v Extension: server_name (len=16)
          Type: server_name (0)
          Length: 16
        > Server Name Indication extension
      > Extension: supported_groups (len=8)
      v Extension: ec_point_formats (len=2)
          Type: ec_point_formats (11)
          Length: 2
          EC point formats Length: 1
        v Elliptic curves point formats (1)
            EC point format: uncompressed (0)
      > Extension: signature_algorithms (len=26)
      > Extension: session_ticket (len=0)
      > Extension: extended_master_secret (len=0)
      > Extension: renegotiation_info (len=1)
```

TLS Version information: 771

Ciphers: 49196-49195-49200-49199-159-158-49188-49187-49192-49191-49162-49161-49172-49171-157-156-61-60-53-47-10

Extensions: 0-10-11-13-35-23-65281

EllipticCurvePointFormats: 0

Figure [3]: Values identified and extracted when creating JA3 hash of a Client Hello. Note: this communication is missing EllipticCurves.

As each of these communications are happening, network monitoring solutions equipped with JA3 fingerprinting capability are able to identify each of these pieces of information. Once each of the values has been identified, they are then grouped together as a string and the entire collection of values are then hashed, resulting in a single unique hash, or fingerprint, for that communication. Note that when utilising JA3, if a value is not present (in this strain of Emotet the EllipticCurves value is missing), then the value will be denoted by a blank space, as shown in Figure 4.

The information extracted from the handshake in Figure 4 is displayed below, along with its' JA3 hash.

JA3 string:

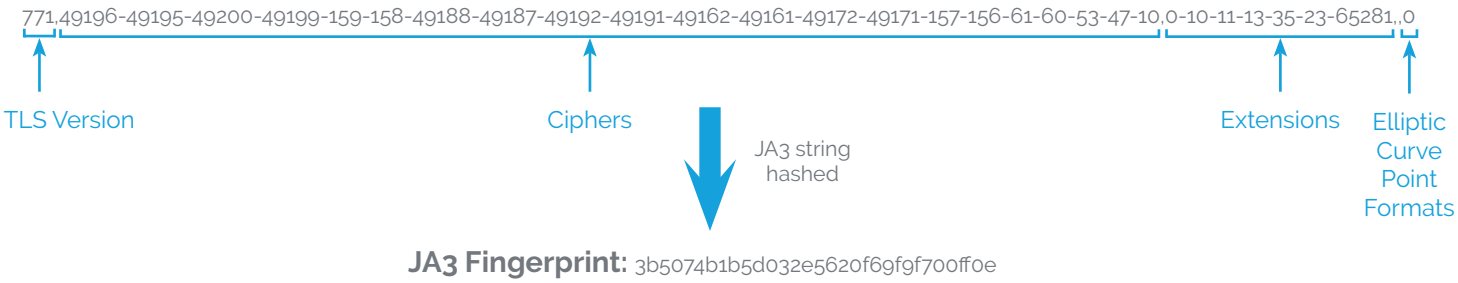

Figure [4]: Process of hashing values obtained previously with resulting JA3 fingerprint

Having generated this hash, the JA3 fingerprint of the Client Hello is now known for that system.

In addition to JA3 fingerprinting, JA3S fingerprinting can also be utilised. Following a similar process as JA3, JA3S looks for specific values within the Server Hello negotiation, which can be extracted to form the JA3S string, which are then hashed to provide the JA3S fingerprint. However, the JA3S fingerprint is not always required.

The JA3 fingerprinting technique when used in isolation, i.e. in absence of JA3S, has been reported to be a very effective method of understanding and grouping network activity. Additionally, by utilising JA3 fingerprinting in support of signature-based detection methods, an organisation can ensure that JA3 hashes of known or newly identified threats can be shared widely throughout certain devices across network (such as IDS), enabling detection of malicious activity in line with the JA3 fingerprints being generated by the passive monitoring capability.

With the JA3 fingerprint identified for the Client Hello, this unique hash can be utilised as a valuable pivot point for SOC analysts and forensic investigations, as well as grouping activity according to an entity set, such as a specific threat actor.

JA3 fingerprints can be shared across the wider threat intelligence community, enabling more users and organisations to utilise the JA3 fingerprint from known databases to support their security solutions in order to further identify malicious activity.

And whilst this process has been utilised against Emotet in this example, it is essential to note that this process can be implemented against all types of network communications. Additionally, as this process can be run passively, it can be utilised to very good effect within multi-Tbps networks, monitoring all of the traffic and fingerprinting all communications in order to identify the malicious activity.

```
[**] [1:15003002:1] Emotet Download [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64252 -> 3.23.235.182:80
[**] [1:15003001:1] .doc Download [**] [Classification: (null)] [Priority: 3] {TCP} 3.23.235.182:80 -> 10.9.30.101:64252
[**] [1:15003002:1] Emotet Download [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64257 -> 45.159.115.191:80
[**] [1:15003005:1] TLS Emotet JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64258 -> 45.159.115.191:443
[**] [1:15003006:1] TLS Emotet JA3S hash [**] [Classification: (null)] [Priority: 3] {TCP} 45.159.115.191:443 -> 10.9.30.101:64258
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64263 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64263 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64263 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64263 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64263 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64263 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64263 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64314 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64314 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64314 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64314 -> 80.87.201.221:7080
[**] [1:15003003:1] Emotet POST-infection traffic [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64348 -> 80.87.201.221:7080
```

```
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64273 -> 91.200.101.192:443
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64270 -> 91.200.101.192:443
[**] [1:15003008:1] TLS TrickBot JA3S hash [**] [Classification: (null)] [Priority: 3] {TCP} 91.200.101.192:443 -> 10.9.30.101:64270
[**] [1:15003004:1] IP Checker Site [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64271 -> 176.58.123.25:443
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64286 -> 185.234.72.147:447
[**] [1:15003008:1] TLS TrickBot JA3S hash [**] [Classification: (null)] [Priority: 3] {TCP} 185.234.72.147:447 -> 10.9.30.101:64286
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64296 -> 91.200.101.192:443
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64297 -> 91.200.101.192:443
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64310 -> 91.200.101.192:443
[**] [1:15003008:1] TLS TrickBot JA3S hash [**] [Classification: (null)] [Priority: 3] {TCP} 91.200.101.192:443 -> 10.9.30.101:64310
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64312 -> 91.200.101.192:443
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64353 -> 91.200.101.192:443
[**] [1:15003008:1] TLS TrickBot JA3S hash [**] [Classification: (null)] [Priority: 3] {TCP} 91.200.101.192:443 -> 10.9.30.101:64353
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64362 -> 91.200.101.192:443
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64332 -> 91.200.101.192:443
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64330 -> 91.200.101.192:443
[**] [1:15003008:1] TLS TrickBot JA3S hash [**] [Classification: (null)] [Priority: 3] {TCP} 91.200.101.192:443 -> 10.9.30.101:64330
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64344 -> 91.200.101.192:443
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64339 -> 91.200.101.192:443
[**] [1:15003008:1] TLS TrickBot JA3S hash [**] [Classification: (null)] [Priority: 3] {TCP} 91.200.101.192:443 -> 10.9.30.101:64339
[**] [1:15003007:1] TLS TrickBot JA3 hash [**] [Classification: (null)] [Priority: 3] {TCP} 10.9.30.101:64345 -> 91.200.101.192:443
```

Figure [5]: IDS alert output identifying Emotet and TrickBot activity against derived JA3 fingerprint.

Summary

No matter the protocol used, some form of communications activity will inevitably take place amongst compromised systems within a network, whether that be through sending of directions/orders from botmasters to the botnet, or exfiltration of data from the compromised host to a data silo/exfiltration server.

Having visibility over this traffic is a starting point, but having the additional ability to assign a fingerprint to every single one of the communications happening within a multi-Tbps networks enables SOC teams and threat analysts to gain an advantage in the ongoing cyber war; a legitimate and accurate means of identifying specific values from within a communication and to cross reference against a database of known threats. The FlowProbe, accelerated through latest generation FPGA technology, identifies this traffic and assigns the JA3 fingerprint to every flow at Tbps.

As the amount of encrypted traffic continues to increase, so too do the variations of malware, with ever more distinctive traffic patterns being generated in a bid to blend in with web traffic and prevent payload analysis. JA3 fingerprinting provides an opportunity to passively monitor and identify all network communications, grouping malicious activity utilising entity sets and providing initial indication as to the suspected threat actor, providing valuable insight for SOC analysts when identifying the most pressing threats to respond to first.

TELESOFT

HEADQUARTERS
Telesoft Technologies Ltd
Observatory House, Stour Park
Blandford DT11 9LQ UK
+44 (0)1258 480880
+44 (0)1258 486598
sales@telesoft-technologies.com

LONDON
Telesoft Technologies Ltd
The Shard
Floor 25
32 London Bridge Street
London SE1 9SG
sales@telesoft-technologies.com

AMERICAS
Telesoft Technologies Inc
430 National Business Parkway
Suite 480, Annapolis Junction
MD 20701
USA
salesusa@telesoft-technologies.com

ASIA
Telesoft Technologies Ltd
Tapasya Corp Heights
Ground Floor, Sector 126
Noida, 201301
+91 120 612 7725
salesindia@telesoft-technologies.com

TELESOFT

telesoft-technologies.com

FLOWPROBE: TBPS THREAT VISIBILITY