“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

On Usability of Hash Fingerprinting for Endpoint Application Identification

Jenny Heino
Department of Computing
University of Turku
Turku, Finland
Forcepoint LLC
Helsinki, Finland
jenny.a.heino@utu.fi

Ayush Gupta
Department of Computing
University of Turku
Turku, Finland
ayush.a.gupta@utu.fi

Antti Hakkala
Department of Computing
University of Turku
Turku, Finland
ajahak@utu.fi

Seppo Virtanen
Department of Computing
University of Turku
Turku, Finland
seppo.virtanen@utu.fi

***Abstract*—In network security, a common challenge is the ability to gain information about the communicating endpoints based only on the network traffic. Methods for gaining endpoint awareness on the network level by fingerprinting different network protocol layers have existed for long. A fairly recent addition to these techniques have been different hash finger- printing algorithms, such as JA3 and JA3S, that can be used for identifying the communicating endpoint applications of the network connection. These algorithms pick a suitable set of protocol specific parameters and concatenate their values into a string. An MD5 hash value is calculated from this string, which comprises the fingerprint. In this article we contest the use of the MD5 hash in the fingerprinting process, and propose that the original string of concatenated protocol parameter values should be used instead. We argue that the original string provides more value for the network security landscape.**

***Index Terms*—Computer network management, Firewalls (computing), Middleboxes, Network security**

I. Introduction

The ability to gain awareness of the communicating endpoints on the network level has been a target of research for a long time. Knowing more about the protected endpoint can greatly enhance the protection capabilities of network security solutions, and give crucial information about the protected network to network administrators. The downside is that the ability to gain knowledge about the endpoint purely based on eavesdropped network traffic can give an attacker leverage when selecting a suitable exploit.

As an example, certain values in the TCP handshake can be a good indication of the underlying operating system [1]. In addition, some application layer protocols, such as HTTP, include a specific protocol header field to report the endpoint application. For such protocols, simple string based matching can be enough to identify the endpoint application. Many network protocols, however, do not include any protocol parameters that simply report the endpoint application.

Hash fingerprinting algorithms, such as *JA3* and *JA3S*, have become popular for identifying the endpoint applications from network connections within the last few years. These algorithms gather a list of protocol specific parameter values

“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

from the network traffic, present them in a string form, and calculate an MD5 hash value of the string. Different endpoint applications support different features, and provide supported features in a different order. Because of this, the calculated hash fingerprint can be a good indication of what the endpoint application behind the network traffic is. Implementations of the algorithms have already found their way into several network security solutions.

These algorithms have especially been utilised in the secu- rity community for identifying network connections initiated by malicious endpoint applications. Malware developers often tend to lean towards a quick and dirty implementation, result- ing in a distinguishable network fingerprint. Still, it is rather easy to change the produced fingerprint with small changes to the parameter values, such as changing the order in which the supported features are presented.

The hash algorithms have their limitations, but they certainly have their use in network security. They are a quick way to gain some information on the network traffic, and they can be useful for identifying benign endpoint applications. Due to their nature, they should not be used as the sole ground for identifying an endpoint as benign, as a malicious entity can fake its hash fingerprint to mimic a benign one with some effort. Instead, they can be used as additional metadata for the network connection.

Still, we believe that these algorithms have a weakness which greatly affects their usability. This weakness is the final step of calculating the MD5 hash value from the set of protocol parameter values. Calculating an MD5 hash value means that even a small change in the original string value creates an entirely different fingerprint, losing all information about how close the original values might have been to each other.

In this article we contest the last step of calculating the hash value, and instead recommend using the original string of concatenated protocol parameter values, that is, the *pre-hash string*, for endpoint application identification. To support our claim, we provide example JA3 values from four different endpoint applications: Firefox, Thunderbird, Google Chrome and Microsoft Edge. These endpoint applications were selected due to the fact that Firefox and Thunderbird use the same

“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

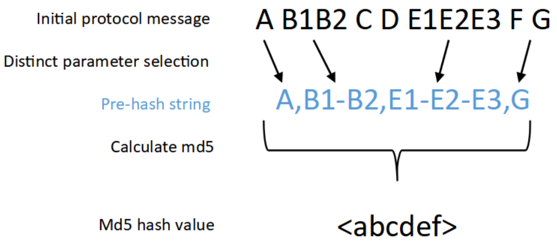


Fig. 1. A generic hash fingerprint structure. A protocol specific set of suitable and distinctive parameters is selected, and the values for these parameters are concatenated into a string. Different parameters are separated with commas, and multiple values for the same parameter are separated with dashes. Finally, an MD5 hash is calculated from this string, comprising the final hash fingerprint.

underlying TLS library, as do Google Chrome and Microsoft Edge. We show that, despite sometimes having a different MD5 hash, the original pre-hash strings are very close to each other when the underlying TLS library is the same.

II. Hash fingerprinting

Hash fingerprinting a network protocol to identify the underlying endpoint application is a relatively new but active field of study. All of the proposed hash fingerprinting algorithms have the same structure. They select a suitable set of significant and distinctive protocol parameters and concatenate the values for these parameters into a string, separating different parameters with commas, and multiple values for a single parameter with dashes. Finally, an MD5 hash value is calculated out of the string. This MD5 hash is then considered the fingerprint for the particular endpoint application regarding the protocol. This process is visualised in Figure 1.

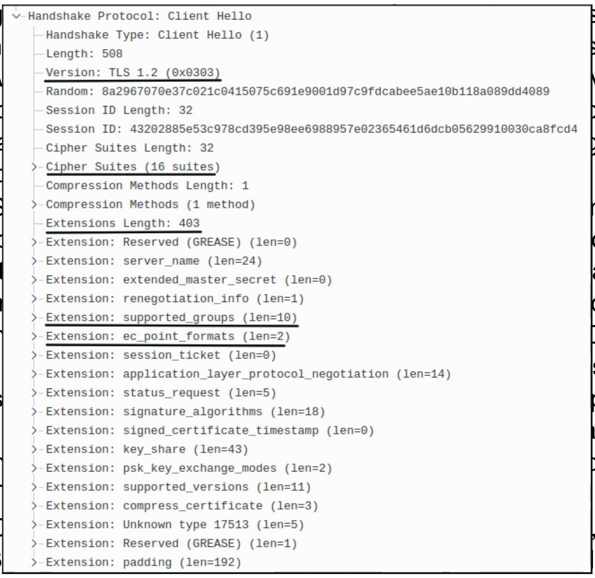
Some protocols, such as HTTP, clearly state the communicating endpoint application in a specific protocol parameter field [2]–[4]. Other protocols do not include any such fields. There are still certain differences in implementations of these protocols that differentiate them from each other. TLS is a good example: it does not include any protocol fields that report the endpoint application, but the protocol has many parameters that report the different features that the endpoint application supports. These include the supported cipher suites, supported named groups for key exchange, and the extensions in the initial hello message [5].

The first publication proposing TLS handshake fingerprinting was the presentation by Lee Brotherston at the 2015 DerbyCon [6]. This presentation paved the way for the first named hash fingerprinting algorithms, JA3 and JA3S. These algorithms for fingerprinting the endpoint application from TLS traffic were invented by the Salesforce employees John Althouse, Jeff Atkinson and Josh Atkins and open-sourced by the company in 2017 [7].

To calculate the JA3 fingerprint, the following values are collected from the Client Hello message: Client Hello ver- sion, supported cipher suites, the list of extensions, supported

“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.



```
v-Handshake Protocol: Client Hello
   Handshake Type: Client Hello (1)
   Length: 508
   Version: TLS 1.2 (0x0303)
   Random: 8a2967070e37c021c0415075c691e9001d97c9fdcabee5ae10b118a089dd4089
   Session ID Length: 32
   Session ID: 43202885e53c978cd395e98ee6988957e02365461d6dcb05629910030ca8fcd4
   Cipher Suites Length: 32
 > Cipher Suites (16 suites)
   Compression Methods Length: 1
 > Compression Methods (1 method)
   Extensions Length: 403
 > Extension: Reserved (GREASE) (len=0)
 > Extension: server_name (len=24)
 > Extension: extended_master_secret (len=0)
 > Extension: renegotiation_info (len=1)
 > Extension: supported_groups (len=10)
 > Extension: ec_point_formats (len=2)
 > Extension: session_ticket (len=0)
 > Extension: application_layer_protocol_negotiation (len=14)
 > Extension: status_request (len=5)
 > Extension: signature_algorithms (len=18)
 > Extension: signed_certificate_timestamp (len=0)
 > Extension: key_share (len=43)
 > Extension: psk_key_exchange_modes (len=2)
 > Extension: supported_versions (len=11)
 > Extension: compress_certificate (len=3)
 > Extension: Unknown type 17513 (len=5)
 > Extension: Reserved (GREASE) (len=1)
 > Extension: padding (len=192)
```

Fig. 2. A screen capture from Wireshark, demonstrating the protocol values used for calculating a JA3 fingerprint. JA3 fingerprint is the hash fingerprint for a TLS client. It is calculated from the Client Hello message. For calculating the JA3 fingerprint, the following protocol parameters are selected: Client Hello version, supported Cipher Suites, list of Extensions, supported groups and supported point formats. The values for these parameters are concatenated into a string, and the final JA3 fingerprint is generated from this string by calculating its MD5 hash value.

groups, and supported elliptic curve point formats. These values are concatenated into a string, and an MD5 hash value is calculated of the string. This hash value comprises the final JA3
fing ssage
gen shark
netv value
calc ocol
para ge to
calc

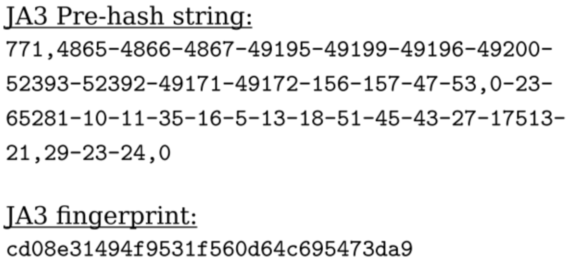
S nting
algo other
prot algo-
rithi d by
Ben Caleb
Yu) shed
has ped
by tema
Ban chael
R. T

C , the
JA3 arity.
Many network security solutions have implemented these algorithms [13], and especially the JA3 fingerprint has become a common tool for many security analysts and incident responders. There are public databases of JA3 hashes, such as the JA3er database [14] and the SSL Blacklist by abuse.ch [15].

The popularity of JA3 over the other algorithms can be attributed not only to its status as the pioneer of the hash fingerprinting algorithms, but also to the popularity of the

“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.



JA3 Pre-hash string:

```
771,4865-4866-4867-49195-49199-49196-49200-
52393-52392-49171-49172-156-157-47-53,0-23-
65281-10-11-35-16-5-13-18-51-45-43-27-17513-
21,29-23-24,0
```

JA3 fingerprint:

```
cd08e31494f9531f560d64c695473da9
```

Fig. 3. JA3 pre-hash string and the final JA3 fingerpint calculated from the Client Hello visible in Figure 2. The parameter values are presented in decimal format, so for example the original version value 0x0303, which indicates TLS 1.2, is presented as 771. The Client Hello message that this JA3 fingerprint was calculated from was produced by the Google Chrome browser version 95.0.4638.69 on Windows operating system.

TLS protocol. RDP, DHCP and SMB are all protocols used mainly in internal networks. QUIC, in turn, has experienced vast changes since the CYU fingerprint was developed, and the CYU algorithm does not apply to the IETF standardized version of QUIC from RFC 9000 [16].

It is worth to note that these hash fingerprinting methods identify the underlying protocol implementation, which often is not enough to uniquely identify the endpoint application itself. A good example is the TLS library developed by Google, BoringSSL, which is a fork from the OpenSSL library [17]. The BoringSSL library is used by Google Chrome and other Chromium based web browsers, such as Microsoft Edge [18]. It is also used by the Chromium Embedded Framework, which in turn is used by many other popular desktop applications, including Spotify [19]. Because of this, a JA3 fingerprint produced by the Google Chrome web browser can have a collision with a JA3 fingerprint produced by Spotify.

III. Spoofing and randomization

All of the passive traffic identification methods referenced here are easy to evade with enough labour. As long as a malicious entity is able to modify the relevant parts of the traffic, it can make it look like a benign endpoint application. The hash fingerprinting algorithms are no exception to this rule.

It is already common that malicious tools fake the User-Agent header value in HTTP traffic. Mimicking a benign hash fingerprint is, however, not as easy as faking the User-Agent header value. For a malicious tool to mask the hash fingerprint value as a benign one, it either needs to use the same underlying protocol implementation as the benign application it mimics, or implement the same features as the benign application. Both of these alternatives can be too much work for a malware developer, who often seeks to find the best profit with as little work as necessary.

Instead of mimicking a benign endpoint applicat

"© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works."

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

presenting only a random subset of the actually supported cipher suites, or presenting all supported cipher suites but in a random order, it is easy for the malicious entity to prevent detection by a JA3 fingerprint. This has already been seen in the wild by Akamai researchers, who reported that bots are randomizing their TLS parameters to evade detection by JA3 fingerprints [20].

Because of this, the endpoint application identification made based on the hash fingerprint should be taken with a grain of salt. As an example, if a malicious tool uses the BoringSSL library developed by Google, it may falsely be identified as the Google Chrome browser based on its JA3 fingerprint and be let through. And vice versa: if the same tool is identified to be malicious and its traffic is thus flagged as malicious based on its JA3 fingerprint, the result may be that legitimate Google Chrome traffic gets flagged as malicious.

IV. Improving security efficacy with hash fingerprints

Even though they are not a silver bullet, the hash fingerprints can still give valuable metadata about the network connection to a network administrator. In most cases they give a good indication about what the endpoint application is, thus giving a better understanding of the network. In addition, the hash fingerprints can be a great tool for adding endpoint context to a network security solution.

A network security solution capable of deep packet inspection may not be aware of the communicating endpoint application. This might cause issues such as false positive and false negative identifications, as different endpoint applications have different vulnerabilities. Traffic that may be benign to one endpoint application can trigger a vulnerability in another. Using the hash fingerprints, a network security solution can make an educated guess about the endpoint application, which it can utilise when making a deep packet inspection based traffic termination decision. This can, as an example, make it possible to terminate a network connection only if the receiving endpoint application is vulnerable to a specific attack.

As mentioned previously, public databases already exist for JA3 fingerprints, the most extensive being the JA3er database, which anyone can contribute to. The advantage of a public database which anyone can contribute to is that it can easily become very extensive. The disadvantage is that it is also easy to contaminate the database as it is difficult to verify the accuracy of a submission.

As opposed to using a public database, it is also possible to gather a private database of trusted hash fingerprints from a protected network. Using a private database removes the risk of an outsider contaminating the database, but it requires additional work to populate and verify the content of the database. Still, if implemented properly, a private database can give an accurate representation of the trusted network and be a good additional tool for validating network traffic.

"© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works."

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

V. Weakness

As stated, the hash fingerprints, especially the JA3 and JA3S fingerprints, have already managed to establish their ground in the network security landscape. They definitely have their value when making a quick assessment of the traffic. But we argue that they have a weakness which greatly reduces their usability, which is the final step of calculating the MD5 hash of the string of protocol parameter values.

Hash values are, indeed, a quick and efficient tool for validating that two entities have the same content. They are often used in the information security field when identifying different malicious entities from legitimate ones. As an example, a reputation check for a file based on its hash value is a very common way for antivirus software to check whether a file has been identified as malicious before. Despite their popularity, file reputation checks based on MD5 and SHA1 hashes are prone to collision attacks and public tools exist for creating a collision, such as [21].

For large entities, such as files, calculating a hash value makes a lot more sense than comparing the likeness of the entities byte by byte. The hash values are short and easy to share, and the algorithms for calculating the values are fast. The advantage gained from calculating a hash value in such cases is evident. But when the original value is rather short, such as it is with the protocol hash fingerprinting algorithms, the advantage gained from calculating a hash value starts to lose its effect, and it can even turn into a disadvantage.

When calculating a hash value of an entity, even a change of one byte in the original content entirely changes the output of the hash algorithm. This is how the hash algorithms are designed to work. A good hashing algorithm is designed to be non-invertible and have correlation freeness, meaning that even a small change in the original value should produce an entirely different output [22].

But when considering the original pre-hash string for the protocol hash fingerprinting algorithms, the information lost when calculating an MD5 hash out of the pre-hash string outweighs the advantage gained from calculating the MD5 hash. The pre-hash string is relatively short, thus storing and sharing an MD5 hash value instead gives little added value. In addition, gaining an entirely different fingerprint after a small change in the original value, such as one added cipher suite in the list of supported cipher suites in a JA3 fingerprint, loses all information about the original values being so close to each other. With the protocol hash fingerprinting values, this is a clear disadvantage: one added cipher suite, when all other values and their ordering remains the same, indicates that the underlying implementation is otherwise the same, but support for one new cipher suite has been added.

One advantage of using an MD5 sum instead of the pre-hash string is that it is faster for a human to pick up a difference between two values when the value is entirely different, as it is with an MD5 hash value. Programmatically, there is no notable difference in the effectiveness of comparing the pre- or post-hash value. It can be argued that the ability of a human

“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

to quickly distinguish two values should not be the basis for choosing the hash value instead of the more informative pre- hash string.

VI. Using the pre-hash string as the fingerprint instead

Based on the disadvantages of using the MD5 hash for the protocol hash fingerprinting algorithms, we propose that the pre-hash string of these algorithms should be used as the fingerprint instead. We claim that the information that can be gained from the pre-hash string greatly outweighs the value gained from using the MD5 hash value.

The pre-hash string consists of a list of the protocol parame- ter values that the endpoint application supports or has chosen to use. In addition to providing an indication of the underlying software component, this value also gives quick visibility into many other useful properties related to the traffic itself, such as the used protocol version and other supported features. This information can give valuable context for a network security solution or an incident responder. A quick look into this value can reveal features in the traffic that can immediately be utilised for flagging the connection as suspicious.

In addition, using this pre-hash string of the protocol param- eter values as the fingerprint makes it possible to identify that two different fingerprints are approximately equal. This can make it possible to identify that a certain connection belongs to the same endpoint application as another connection, even when the fingerprint itself is different. This can happen for example when a new software version is released: the release brings support for a new feature, thus changing the fingerprint, but otherwise the values and their ordering remains the same. This also makes it possible to identify an endpoint application which randomizes its protocol values.

VII. A comparison of the JA3 hash and the pre-hash string for four endpoint applications

To give a lightweight proof-of-concept to support our claim, we compared the JA3 hash and pre-hash string of four different endpoint applications on Windows 10 operating system. The endpoint applications we selected for our proof of concept were Firefox web browser (version 94.0.1.0), Thunderbird email client (version 91.2.1.0), Google Chrome web browser (version 95.0.4638.69) and Microsoft Edge web browser (version 95.0.1020.44). We selected these particular endpoint applications, because the underlying TLS libraries for Firefox and Thunderbird are the same [23], as are the underlying TLS libraries for Google Chrome and Microsoft Edge. We selected the latest two distinctive JA3 values produced by each endpoint application on our test machine at a given point in time, and analysed them. These values can be seen in Table I.

Looking first at the values for Firefox and Thunderbird, we see that all MD5 hashes are entirely different in each instance. However, taking a deeper look into the pre-hash strings, there is actually very little difference in these values. In all cases, the supported cipher suites are exactly the

“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

TABLE I

JA3 VALUES FOR FIREFOX, THUNDERBIRD, GOOGLE CHROME AND MICROSOFT EDGE ON WINDOWS OPERATING SYSTEM

Endpoint Application	Pre-hash string	MD5 Hash
Firefox	771,4865-4867-4866-49195-49199-52393-52392-49196-49200-49162-49161-49171-49172-156-157-47-53-10,0-23-65281-10-11-16-5-34-51-43-13-45-28-41,29-23-24-25-256-257,0	1af5d1fe5c1c9bdba4ec723ac5cab44f
Firefox	771,4865-4867-4866-49195-49199-52393-52392-49196-49200-49162-49161-49171-49172-156-157-47-53-10,0-23-65281-10-11-16-5-34-51-42-43-13-45-28-41,29-23-24-25-256-257,0	a0262d81f08838bbb1877a10e3fd70f1
Thunderbird	771,4865-4867-4866-49195-49199-52393-52392-49196-49200-49162-49161-49171-49172-156-157-47-53-10,0-23-65281-10-11-35-5-34-51-43-13-45-28-21,29-23-24-25-256-257,0	490dba4384bdcf3fb9f1682374dd4afc
Thunderbird	771,4865-4867-4866-49195-49199-52393-52392-49196-49200-49162-49161-49171-49172-156-157-47-53-10,0-23-65281-10-11-35-16-5-34-51-43-13-45-28-21,29-23-24-25-256-257,0	6b5e0cfe988c723ee71faf54f8460684
Chrome	771,4865-4866-4867-49195-49199-49196-49200-52393-52392-49171-49172-156-157-47-53,0-23-65281-10-11-35-16-5-13-18-51-45-43-27-17513-41,29-23-24,0	598872011444709307b861ae817a4b60
Chrome	771,4865-4866-4867-49195-49199-49196-49200-52393-52392-49171-49172-156-157-47-53,0-23-65281-10-11-35-16-5-13-18-51-45-43-27-17513-21,29-23-24,0	cd08e31494f9531f560d64c695473da9
Microsoft Edge	771,4865-4866-4867-49195-49199-49196-49200-52393-52392-49171-49172-156-157-47-53,0-23-65281-10-11-35-16-5-13-18-51-45-43-27-17513-21,29-23-24,0	cd08e31494f9531f560d64c695473da9
Microsoft Edge	771,4865-4866-4867-49195-49199-49196-49200-52393-52392-49171-49172-156-157-47-53,0-23-65281-10-11-35-16-5-13-18-51-45-43-27-17513,29-23-24,0	e1d8b04eeb8ef3954ec4f49267a783ef

JA3 values for Firefox web browser version 94.0.1.0, Thunderbird email client verion 91.2.1.0, Google Chrome web browser version 95.0.4638.69 and Microsoft Edge web browser version 95.0.1020.44 produced on Windows 10 Operating System. The values were selected based on being the two latest distinctive JA3 values produced by the endpoint application in question in our test environment at the moment of the test.

same: ”4865-4867-4866-49195-49199-52393-52392-49196-49200-49162-49161-49171-49172-156-157-47-53-10”. The supported groups are also exactly the same: ”29-23-24-25-256-257”. And in all instances the supported elliptic curve point formats are empty.

The only difference between the four JA3 values for Firefox and Thunderbird is in the extensions. And even there the difference is quite small. In each instance, the extensions begin with the same 5 extensions in the same order: ”0-23-65281-10-11”. In addition there are other common sub-strings in all values: each one has the three extensions ”5-34-51” in this order, and each one has the four extensions ”43-13-45-28” in this order. The differences are in extensions 16, 35, 42, 41 and 21, which appear in some values but not all.

Taking a look at the values for Google Chrome and Microsoft Edge, we can make very similar remarks.

One of the values is exactly the same for Google Chrome and Microsoft Edge: this is for each browser. Again, the cipher suites are exactly than the one for Firefox and Thunderbird. The supported

“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

groups string is also identical for each value: ”29-23-24” - which, again, is different than the one produced by Firefox and Thunderbird. Similar to Firefox and Thunderbird, the supported elliptic curve point formats are empty.

Taking a look at the extensions, the similarities are even greater than between the values for Firefox and Thunderbird. The values are identical except for the very last extension. One instance has the extension 41 as the last extension, another one has 21 as the last extension, and the last one does not have either of them.

There are greater differences between the extensions when comparing the extensions for Firefox and Thunderbird to the extensions for Google Chrome and Microsoft Edge. As an example, the extensions 18, 27 and 17513 are present in all values for Google Chrome and Microsoft Edge, but in none of the values for Firefox and Thunderbird. Similarly, the extensions 28 and 34 are present in all values for Firefox and Thunderbird, but in none of the values for Google Chrome and Microsoft Edge.

The purpose of this lightweight proof-of-concept is to demonstrate with a real life example how significant the benefits of using the pre-hash string instead of the MD5 hash value can be. We can quickly see which pre-hash strings belong to the same TLS libraries, even when the MD5 hashes are entirely different. It is clear that the sample set is very small, which is why further research is needed to better

“© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

validate the claim.

VIII. CONCLUSION

Moving towards utilising the pre-hash string of the hash fingerprinting algorithms can pave an easier way forward for incident responders and future research. Collecting and sharing the pre-hash strings instead of the MD5 hashes provides a greatly extended view into the properties of the network connection. Instead of focusing on the exact match of an MD5 sum, network security solutions can move towards checking if the matches are *close* to each other. We tested the suitability of our proposition with a proof-of-concept experiment, where we saw that the pre-hash string can indeed improve the capability of identifying endpoint applications. Even though the MD5 hashes were entirely different, the pre-hash strings could be used for identifying the endpoint application. The pre-hash string has, in addition, a lot of potential for many different machine learning applications. Extensive use of the pre-hash string instead of the MD5 hash value could enable for easy-to-apply machine learning methods.

ACKNOWLEDGMENT

Jenny Heino wishes to thank her colleagues at Forcepoint NGFW for inspiring discussions regarding hash fingerprinting algorithms and emerging network security topics in the ever changing network security landscape.

REFERENCES

[1] E. Hjelmvik, Passive OS Fingerprinting. [Online]. Accessed 6th July 2021. Available: https://www.netresec.com/?page=Blog\&month=2011-11\&post=Passive-OS-Fingerprinting (URL)

[2] Mozilla Foundation, Firefox user agent string reference. [Online]. Accessed 11th April 2022. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent/Firefox (URL)

[3] Microsoft Corporation, Detecting Microsoft Edge from your website. [Online]. Accessed 11th April 2022. Available: https://docs.microsoft.com/en-us/microsoft-edge/web-platform/user-agent-guidance (URL)

[4] Google, User Agent Strings - Google Chrome. [Online]. Accessed 11th April 2022. Available: https://developer.chrome.com/docs/multidevice/user-agent/ (URL)

[5] Eric Rescorla and Tim Dierks, The Transport Layer Security (TLS) Protocol Version 1.2. RFC-5246. [Online]. Accessed 11th April 2022. Available: https://www.rfc-editor.org/rfc/rfc5246 (URL)

[6] L. Brotherston, “Stealthier attacks and smarter defending with TLS fingerprinting,” *DerbyCon V (2015), Louisville, Kentucky, USA*. (conference presentation)

[7] J. Althouse, J. Atkinson and J. Atkins, Open Sourcing JA3. [Online]. Accessed 22nd September 2021. Available: https://engineering.salesforce.com/open-sourcing-ja3-92c9e53c3c41 (URL)

[8] B. Reardon, Open Sourcing HASSH. [Online]. Accessed 15th September 2021. Available: https://engineering.salesforce.com/open-sourcing-hassh-abed3ae5044c (URL)

[9] C. Yu, GQUIC Protocol Analysis and Fingerprinting in Zeek. [Online]. Accessed 11th April 2022. Available: https://engineering.salesforce.com/gquic-protocol-analysis-and-fingerprinting-in-zeek-a4178855d75f
(URL)

[10] A. Karimishiraz, RDP Fingerprinting. [Online]. Accessed 11th April
2022. Available: https://medium.com/@0x4d31/rdp-client-fingerprinting- 9e7ac219f7f4 (URL)

[11] T. Coffeen, “DHCPv6 Fingerprinting and BYOD,” in *NANOG 59, Chandler, Arizona, USA*, 2013. [Online]. Accessed 6th October 2021. Available: https://archive.nanog.org/meetings/abstract?id=2206 (Conference presentation)

[12] M. R. Torres, SMBFP SMB Fingerprinting Zeek package. [Online]. Accessed 11th April 2022. Available: https://github.com/micrictor/smbfp (URL)

"© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works."

J. Heino, A. Gupta, A. Hakkala and S. Virtanen, "On Usability of Hash Fingerprinting for Endpoint Application Identification," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 38-43, doi: 10.1109/CSR54599.2022.9850305.

[13] J. Althouse, J. Atkinson and J. Atkins, JA3 - A method for profiling SSL/TLS Clients. [Online]. Accessed 11th April 2022. Available: https://github.com/salesforce/ja3 (URL)

[14] JA3er.com, SSL Fingerprint JA3. [Online]. Accessed 11th April 2022. Available: https://ja3er.com/ (URL)

[15] Abuse.ch, SSLBL | Malicious JA3 Fingerprints. [Online]. Accessed 11th April 2022. Available: https://sslbl.abuse.ch/ja3-fingerprints/ (URL)

[16] J. Iyengar and M. Thomson, QUIC: A UDP-Based Multiplexed and Secure Transport. RFC-9000. [Online]. Accessed 11th April 2022. Avail- able: https://datatracker.ietf.org/doc/html/rfc9000 (URL)

[17] Google, BoringSSL. [Online]. Accessed 6th July 2021. Available: https://boringssl.googlesource.com/boringssl/ (URL)

[18] Microsoft, Download the new Microsoft Edge based on Chromium. [Online]. Accessed 11th April 2022. Available: https://support.microsoft.com/en-us/microsoft-edge/download-the-new-microsoft-edge-based-on-chromium-0f4a3dd7-55df-60f5-739f-00010dba52cf (URL)

[19] Spotify AB, Open Source @ Spotify.com: Chromium Embedded Framework (CEF). [Online]. Accessed 11th April 2022. Available: https://www.spotify.com/us/opensource/ (URL)

[20] M. Zioni, E. Shuster, S. Shavit, Y. Daya, Bots Tampering With TLS to Avoid Detection. [Online]. Accessed 15th September 2021. Avail- able: https://www.akamai.com/blog/security/bots-tampering-with-tls-to- avoid-detection (URL)

[21] P. Selinger, MD5 Collision Demo. [Online]. Accessed 17th July 2021. Available: https://www.mscs.dal.ca/~selinger/md5collision/ (URL)

[22] S. Al-Kuwari, J. H. Davenport, R. J. Bradford, "Cryptographic Hash Functions: Recent Design Trends and Security Notions," *IACR Cryptology ePrint Archive, Report 2011/565*

[23] Mozilla Corporation, Network Security Services (NSS) – Firefox Source Docs documentation. [Online]. Accessed 11th April 2022. Available: https://firefox-source-docs.mozilla.org/security/nss/index.html (URL)