Ashrika Agarwal

TLS Fingerprinting using JA3 for Android Application

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master’s Thesis

29 April 2024

PREFACE

The major objective of this thesis is to perform the TLS Fingerprinting of different commonly used Applications on Android operating systems using JA3 method. The purpose of JA3, an SSL/TLS client fingerprinting tool that is open source, is to recognize and categorize SSL/TLS client applications based on their distinctive attributes. During the initial SSL/TLS connection establishment, the tool functions by analyzing the SSL/TLS handshake messages that are transmitted between the client and server.

While working on this Thesis, I got to learn a lot about how TLS fingerprinting can help us to detect various mobile applications running on different Android platforms. Choosing this research topic benefited me both personally and professionally as I got the chance to learn about different new tools and techniques. This thesis involved lot of research and dedicated hours of working time.

Moving further, I want to thank my Head of master's program in Information Technology Ville Jääskeläinen for providing me with this opportunity to appear in this course and in writing this thesis as well. I would also like to thank Sami Sainio, my Master Thesis Supervisor for providing me with the necessary guidance.

Helsinki, April 29, 2024

Ashrika Agarwal

Abstract

Author:	Ashrika Agarwal
Title:	TLS Fingerprinting using JA3 for Android Application
Number of Pages:	54 pages + 3 appendices
Date:	29 April 2024
Degree:	Master of Engineering
Degree Programme:	Information Technology
Professional Major:	Networking and Services / Medical Technology
Supervisors:	Sami Sainio
	Ville Jääskeläinen, Head of Master's program in IT

TLS (Transport Layer Security) fingerprinting has developed as a response to increasing concerns around cybersecurity threats. To detect malicious activity in encrypted traffic between servers and clients without decrypting the data, a mechanism was needed. This has resulted in the creation of the JA3 methodology. Previous studies have used statistical, mathematical, and intelligent computing techniques to analyse network traffic. The focus of this research lies in querying the oddness and identity of what client fingerprints different versions across Android Releases would generate from different applications.

JA3 is a free open source SSL/TLS (Secure Socket Layer/Transport Layer (Definition of) Security) client fingerprinting with the ability to identify and categorize different classes of clients applications created via the unique features within your TLS/SSL packets. It works by inspecting the SSL/TLS handshake messages that a client and server exchange to establish an initial connection. SSL/TLS client app detection JA3 can be used to detect when unusual or suspicious SSL/TLS clients are being used, which in turn suggests a possible security risk. This means that, JA3 is easily identifiable and thus, it improves network security on our network environment. While JA3 has generally proven to be quite efficient at detecting SSL/TLS client applications in all environments, it may be considered too wide for network security. During the

first connection of client-to-server you get the initial handshake message between SSL/TLS, which checks them against the packers of TLS/SSL by examining their unique characteristics to distinguish multiple applications. This is likely more relevant for us (i.e. using TLS client hello packet and fingerprint) for a good reason.A series of widely used applications is made they are executed on different versions of Android, and packets containing them are sniffed out. Finally, fingerprints are generated from these packets. The aim is to enable JA3 be used as a tool to find potential security risks by discovering when strange or suspicious SSL/TLS client applications are used. This process will help organisations to understand their network traffic better and implement measures that anticipate potential security threats.

Keywords: TLS/SSL, Fingerprinting, JA3, Network Packets

Contents

List of Abbreviations

3GPP	3rd Generation Partnership Program
4G	Fourth Generation (mobile network)
TLS	Transport Layer Security
SSL	Secure Socket Layer
JA3	The name JA3 is derived from the first initials of the last names of the three developers: John Althouse, Jeff Atkinson, and Josh Atkins.
ARPANET	Advanced Research Projects Agency Network
DOD	US Department of Defence
POODLE	Padding Oracle On Downgraded Legacy Encryption
MITM	Man In The Middle attack
BEAST	Browser Exploit Against SSL/TLS
CRIME	Compression Ratio Info-leak Made Easy
FREAK	Factoring RSA-EXPORT Keys
DROWN	Decrypting RSA using Obsolete and Weakened Encryption
RC4	Rivest Cipher version 4
ECC	Elliptic Curve Cryptography
AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman
DES	Data Encryption Standard
IDEA	International Data Encryption Algorithm
MAC	Message Authentication Code
HMAC	Hash-based Message Authentication Code
SHA	Secure Hash Algorithm
PSK	Pre-Shared key
CCM	Cipher Block Chaining
HEX	Hexadecimal
HTTPS	Hypertext Transfer Protocol Secure
PCAP	Packet Capture
VPN	Virtual Private Network
macOS	Macintosh Operating System
CSV	Comma Separated Values
JSON	JavaScript Object Notation
Gmail	Google Mail
iOS	iPhone Operating System
JARM	Just Another Relationship Matcher
GREASE	Generate Random Extensions And Sustain Extensibility
CYU	Client Hello fingerprinting by Yossi

1 Introduction

ARPANET was the initial form of the Internet, introduced by the US Department of Defence in 1969. In recent times, the Internet has become a crucial component of our daily lives as almost everything is now connected to it. However, during its early stages, information exchange was uncomplicated and had no security measures. This raised concerns among researchers, as the usage of the Internet was rapidly increasing and expanding into new technologies like e-commerce, emails, and social media. Therefore, they recognized the need to implement security concepts to safeguard innocent Internet users from potential attackers and cyber criminals.

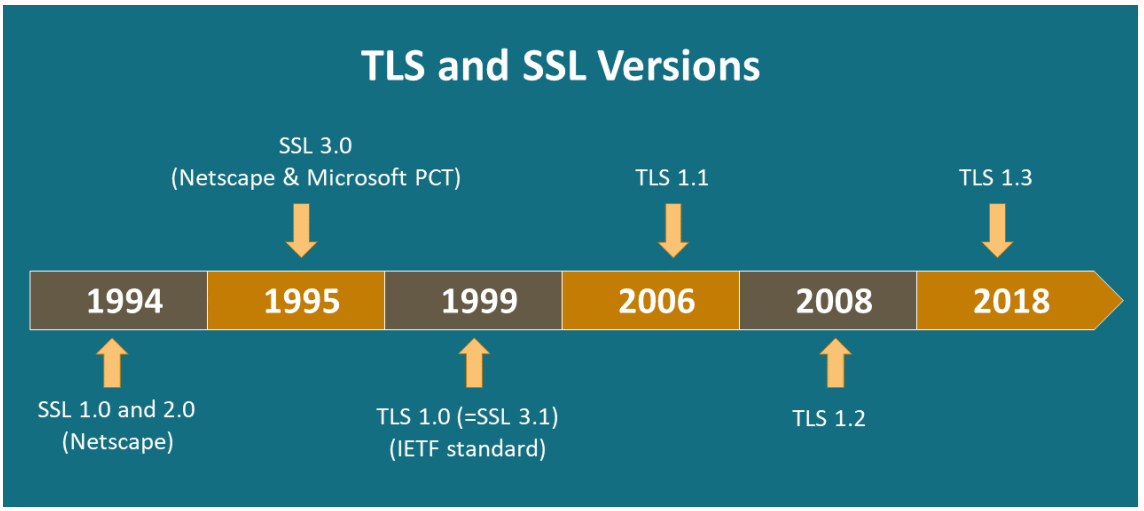

Figure 1 TLS and SSL versions over the years [4]

SSL (Secure Sockets Layer) was developed by Netscape [1] in the mid-1990s as a protocol for securing communication over the internet. The development of SSL was driven by the need to provide a secure way for users to transmit sensitive information, such as credit card numbers and passwords, over the internet. SSL is digital security that only encrypts the data from server end to browser end.When a user connects to an SSL secured site, the web server and your browser create a unique encryption key that is used to encode messages being sent between the two ends. It ensures that any information transfer from

the web server to the browser to a user's device is encrypted and is not intercepted by a hacker. SSL was later replaced by the more secure Transport Layer Security (TLS) protocol. Despite those changes, a lot of systems still refer to the encryption and authentication layer provided by TLS as SSL.

The company that created the first internet web browser, Netscape Communications is also responsible for bringing SSL 1.0 to us (1994). Well, with the exception of SSL 1.0 because it showed how every single security vulnerability operated, so that wasn't released to the public. SSL (Secure Sockets Layer) and its successor TLS (Transport Layer Security) are great security protocol for encryption of communications over internet but it birthed a Mayor problem or a flaw:-

1. Performance Implications: SSL/TLS introduces additional processing and communication overhead that could show up as slower page load times and degraded performance, especially on low power devices like mobiles.
2. Compatibility: Legacy web browsers or software may not work with the most recent SSL/TLS versions-this might present compatibility issues.
3. Managing certificates: SSL/TLS depend on digital certificates for server and client authentication, which is a tedious and complex process that can be overwhelming especially if the organization has to deal with many servers.
4. Cost: The cost of acquiring SSL/TLS certificates can be quite high, especially for organisations that require extended validation (EV) certificates - the certificates with the most identity verification.

5. SSL/TLS vulnerabilities: SSL/TLS, like all security technology, is prone to vulnerabilities and exploits that have been and will continue to be discovered by attackers. It means tight monitoring and updating of SSL/TLS implementations, to avoid becoming vulnerable.

POODLE (Padding Oracle On Downgraded Legacy Encryption) attack was demonstrated by security researchers from Google [2][3]. They demonstrated how the server and client can be forced to fallback to use SSL 1.0 or 2.0 instead of 3.0. Hence in order to avoid this biggest drawback TLS (Transport Layer Security) or SSL 3.1 was finally introduced in 1999.

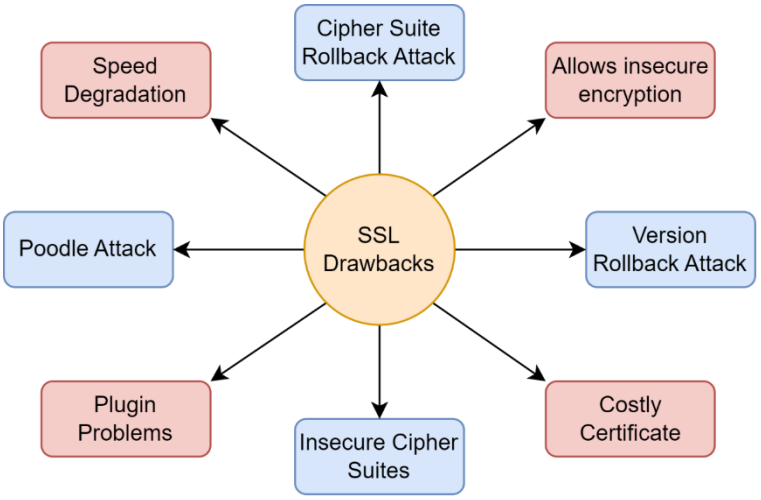

Figure 2 SSL drawbacks

TLS 1.3 is the latest security encryption technique used these days to establish secure and reliable connection between Server and Client. TLS is more secure than SSL. It has stronger encryption algorithms and better cryptographic primitives, making it more resistant to attacks such as man-in-the-middle (MITM) attacks. TLS provides better authentication mechanisms than SSL. It uses digital certificates to authenticate both the server and the client, ensuring that the communication is only between the intended parties. Finally, TLS is more compatible with modern web standards and technologies. As a result, many web browsers and servers have deprecated SSL in favour of TLS.

2 Theoretical Background

2.1 History of SSL

SSL was developed by Netscape organization in 1994. The SSL was inspired by a movie “The Christmas Story”. In this movie a Secret Decoder Ring was used to decode a message by the protagonist. Inspired by this Netscape developed something similar which was later named as Secure Socket Layer [5]. Using this, Netscape implement concept of “Public Key” and “Private Key” which were deployed on individual level.

Netscape also realized that users on the internet can’t be trusted and might lose their keys, so a third-party system was created which will verify the identity of the users on the Internet. Hence, SSL Certificate was developed which will be used to prove the authenticity of an entity on the internet.

Figure 3 Secret Decoder Ring from "The Christmas Story"

Private Key and Public Key served their own purpose based on their name. Private Key was kept by an individual while Public Key as the name suggests, they were meant to distribute to others. If two users were to communicate with each other they will exchange their Public Key, encrypt the data using public Keys and then send the encrypted information. The encrypted information will be decoded using Private Key.

$$Encrypted\ Data = Encrypt(Public\ Key(Sensitive\_Data))$$

$$Sensitive\_Data = Decrypt(Private\ Key(Encrypted\ Data))$$

SSL stands for Secure Socket Layer. SSL works by exchanging Security information between Server and Client which include security encryption techniques that will be used between Server and Client.

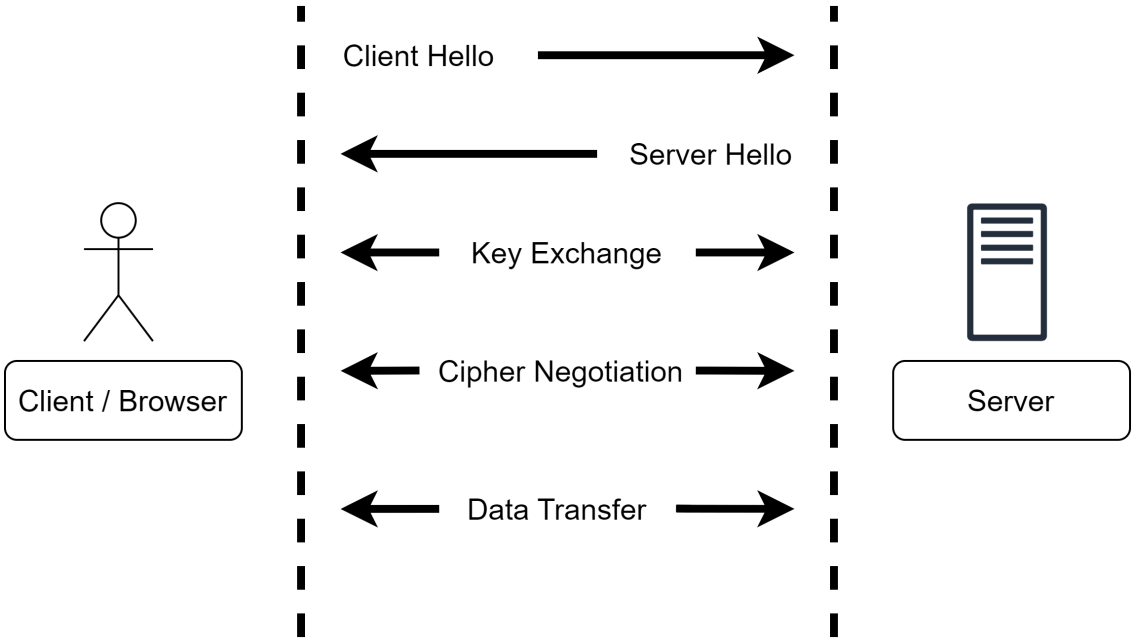

Figure 4 Working of SSL [6]

Working of SSL :-

1. First Client introduces itself to Server with a Client Hello Packet. Client sends data like SSL version to use, cipher suites that it supports and some other session data.

2. The Server selects a particular Cipher from the list of Ciphers suites send by the Client inside the Client Hello packet. Then Server responds back to the Client with “Server Hello” message. The Server Hello message contains SSL version, selected Cipher suite and similar session data that is required to initiate SSL connection.

3. In key exchange process, Server also sends its public key to the Client and its SSL certificate to the Client. The Client verifies Server's SSL certificate from the Certificate Authority service and validates if the Server is authentic of not. If the validation is done successfully then session key is generated by the Client and send to the Server otherwise Client refuses the SSL connection to the Server. The Client encrypts generated Session key with Servers Public Key and sends it to the Server.
4. The Server receives the encrypted Session key and decrypts it with its Private key and sends acknowledgement back to the client.
5. Finally, both Server and Client have session key present with them, and they can start communicating with each other now.

2.2 Attacks on SSL

While SSL provide effective protocols for securing communication over the internet, they are not immune to attacks [7]. Some common attacks on SSL include :-

1. Man-in-the-middle attacks [8]: In this attack, an attacker intercepts communication between a client and server and can view, modify, or inject data into the communication stream.
2. SSL stripping [9]: In this attack, an attacker intercepts SSL traffic and downgrades the secure connection to an insecure connection, making it vulnerable to further attacks.
3. Protocol downgrade attacks [10]: Attackers may exploit vulnerabilities in SSL implementations to downgrade the connection to an older, less secure protocol version.

4. BEAST attack [11][12]: This is a vulnerability in SSL 3.0 and TLS 1.0 that allows an attacker to recover plaintext communication by exploiting weaknesses in the protocol's encryption.

5. Heartbleed [13]: A vulnerability in the SSL implementation of OpenSSL that allows both attackers to read data from a server's memory and therefore extract private keys, passwords?

6. POODLE attack [12]: The padding oracle on downgraded legacy encryption (also known as POODLE) is an exploit which takes advantage of the way some browser-websites communicate on private networks using old encryption standards (i.e., SSL 3.0).

7. CRIME attack [12]: An attack against SSL compression, where an attacker recovers plaintext information upon observing the size of compressed data.

8. Lucky13 attack [12]: This is a timing attack that exploits a vulnerability in the CBC (Cipher Block Chaining) mode of encryption used in TLS.

9. FREAK attack [12]: This is vulnerability in SSL that attacker can force client & server to uses lesser secure cipher and then read the traffic.

10. DROWN attack [14]: This is an attack that exploits vulnerabilities in SSLv2 to decrypt TLS traffic.

11. Logjam attack [15]: This is a vulnerability in the Diffie-Hellman key exchange used during SSL/TLS that allows an attacker to downgrade the encryption to weakened cipher.

12. Padding Oracle Attack [16]: The padding oracle attack is an attack on SSL/TLS that makes use of a vulnerability in the encryption padding of some messages, which can then allow for attacker to extract the plain text form encrypted messages.

13. RC4 attack [12]: This is a weakness in the RC4 encryption algorithm used in SSL/TLS that can allow an attacker to decrypt SSL/TLS traffic.

14. Ticketbleed [17]: This vulnerability in the OpenSSL implementation of SSL/TLS however, it is more profound than SSL/TLS exposure to a server memory based attacker.

15. Timeless Timing Attack [12]: This timing attack observes the server response times for each instance of a SSL/TLS cryptographic requests, leaking sensitive information.

Preventing such attacks requires more modern SSL/TLS implementations along with ongoing security patches to support better practices both disabling old SSL/TLS versions and functionality along with the use of weaker encryption and stronger certificate management. Additionally, behaviors such as TLS False Start and TLS Session Resumption can protect you against SSL/TLS attacks.

2.3 Introduction to TLS

TLS is a security layer which keeps debt off communication over the internet [6] Created in the mid 1990s by Netscape, it was created due to the Luxe SSL protocol. Transport Layer Security (TLS) is one of the most important cryptographic protocols designed to make sensitive applications more secure, but which is not branded software_type commercial_open_source proprietary. It secures communication over networks such as the Internet. That's working among applications and application servers to provide systems-to-system service Based on Trust Elsewhere, it offers security at both protocol layers that communicate with users' application data. Security in the case of TLS protocol is maintained in two principal aspects - encryption and authentication. Lastly, over the Internet, data is shared as encrypted,, thus reducing the risk of data being intercepted and read by those who should not have access. It also serves the purpose of authentication, by guaranteeing that communicating parties are who they say they are such as man-in-the-middle attacks. TLS combines symmetric and asymmetric encryption for security. The transmitting data is

encrypted using symmetric encryption and a secure gloves to trade the session key for symmetric encryption is encrypted with asymmetric encryption.

TLS also provides for a selection of cryptographic algorithms such as AES, RSA and Elliptic Curve Cryptography (ECC) thereby supporting the Digital Certificates to enable verification between parties talking.

TLS Version	Description
TLS 1.0	TLS 1.0 which was upgrade of SSL v.3.0 released in January 1999 but it allows connection downgrade to SSL v.3.0.
TLS 1.1	After that, TLS v1.1 was released in April 2006, which was an update of TLS 1.0 version. It added protection against CBC (Cipher Block Chaining) attacks. In March 2020, Google, Apple, Mozilla and Microsoft has announced for deprecation of TLS 1.0 and 1.1 versions.
TLS 1.2	TLS v1.2 was released in 2008 that allows to specification of hash and algorithm used by the client and server. It allows authenticated encryption, which was added more support with extra data modes. TLS 1.2 was able to verify length of data based on cipher suite.
TLS 1.3	TLS v1.3 was released in August 2018 and had major features that differentiate it with its earlier version TLS v1.2 like removal of MD5 and SHA-224 support, require digital signature when earlier configuration used, compulsory use of Perfect forward secrecy in case of public-key based key exchange, handshake messages will now be encrypted after “Server Hello”.

Figure 5 TLS Versions [19]

First, TLS relies on stronger encryption algorithms (RC4, DES, AES, etc.) over Telex's old Fortezza cipher suites. SSL has a single "Alert Message", saying "No Certificate" and TLS provides several different kinds of "Alert Messages". SSL only provides Message Authentication Code (MAC), while TLS implements HMAC, which is a MAC protocol using hashes computed on every handshake message. Furthermore, SSL lacked the naturality to support asymmetric encryption and TLS continues these principles with improvements over SSL such as it is faster, more stable. It also offers quick connectivity and low latency as compared to SSL.

The TLS client packet is made up of several components [6]:

- The first message is the ClientHello message: This goes from the client to the server and contains details of what version of TLS the client supports, a random number, and details of which ciphersuites that it supports.
- Session ID: It may also include a Session ID (if it has established a previous session with the server i.e. to resume a session).
- Encryption: When a client has connected to a server, it will provide a list of cipher suites that it supports; the server will then choose the strongest one both the client and server support.
- Compression Methods: The client also includes a list of compression methods that it supports.
- Extensions: The client can include additional extensions to provide more information about its capabilities and preferences.

Once the server receives the TLS client packet, it responds with a TLS server packet that includes its own random number, the selected cipher suite, and other information required to establish a secure connection. This back-and-forth exchange continues until both the client and server have agreed on a secure protocol for the session.

2.4 TLS Client Hello Packet

TLS Client Hello packet is generated by the Client which include security configuration of the client that will be send to the Server so that TLS session can be initiated.

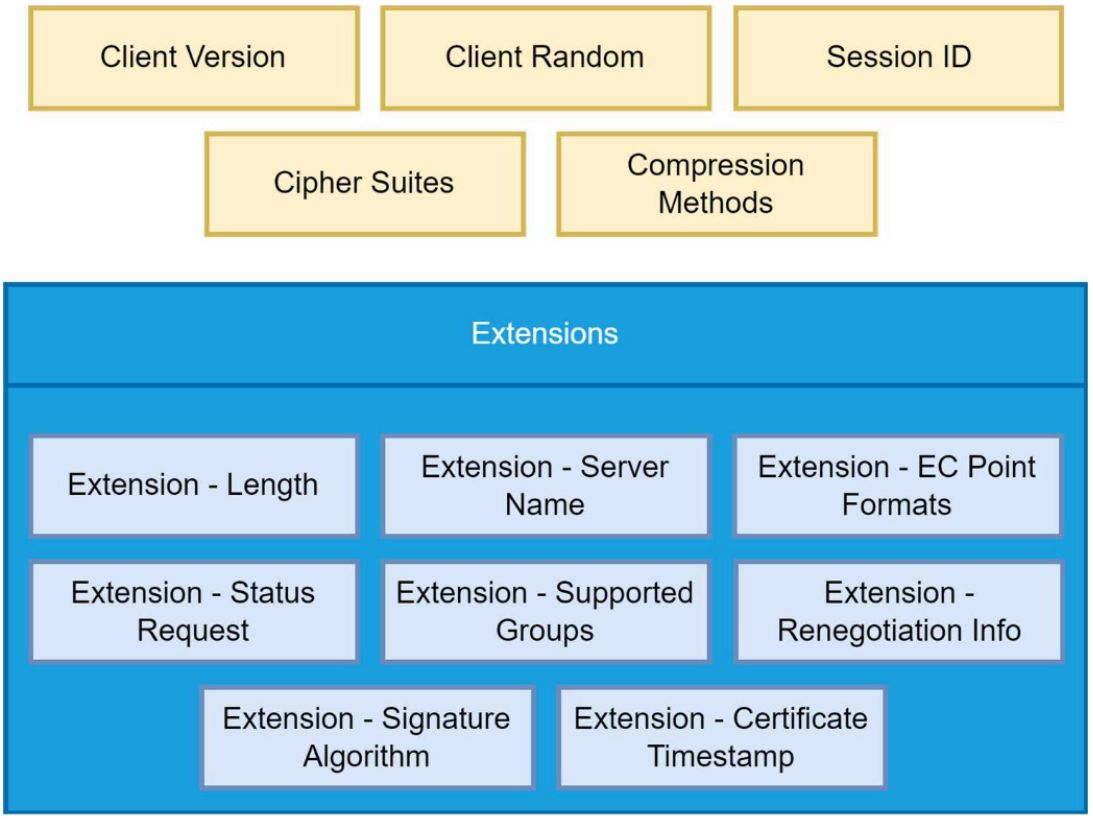

Figure 6 TLS Client Hello Packet

From the above figure, the various elements of TLS Client Hello Packet are as follows :-

- Client Version: This is the version of TLS used by the client. This can be 1.0, 1.1, 1.2 or 1.3.
- Client Random: A random 32 bytes of data generated randomly by the client at the time of TLS negotiation.
- Session ID: If connected to same server previously then Session ID of previous TLS connection is used else set to “00”.
- Cipher Suites: Encryption or Cipher Suites supported by the client for example TLS_SHA256_SHA256, TLS_PSK_WITH_AES_256_CCM, TLS_AES_128_CCM_8_SHA256, TLS_AES_128_CCM_SHA256 etc.
- The Cipher suites are represented by their HEX prefix.

Cipher Suite	Hex Prefix
TLS_SHA256_SHA256	0xC0 0xB4
TLS_PSK_WITH_AES_256_CCM	0xC0 0xA5
TLS_AES_128_CCM_8_SHA256	0x13 0x05
TLS_AES_128_CCM_SHA256	0x13 0x04

Figure 7 Cipher Suites example

- Compression Methods: Compression techniques that are available which will be used for compressing the data before the data is encrypted.
- Extensions
    1. Length: Refers to the length of all the applied or used extensions.
    2. Server Name: Indicates the hostname the client wants to connect to.
    3. EC Point Formats: Indicates the elliptical curve points used in elliptical curve cryptography.
    4. Status Request: Allows the client to directly verify the server's certificate.
    5. Supported Groups: Refers to the named groups that the client and server support for key exchange.
    6. Renegotiation Info: Allows the client and server to initiate a new handshake and establish new cryptographic parameters during an established TLS connection.
    7. Signature Algorithm: Refers to the signature algorithm used to verify the server's digital signature.
    8. Certificate Timestamp: Enables the server to promise to list its certificate in the certificate timestamp log.

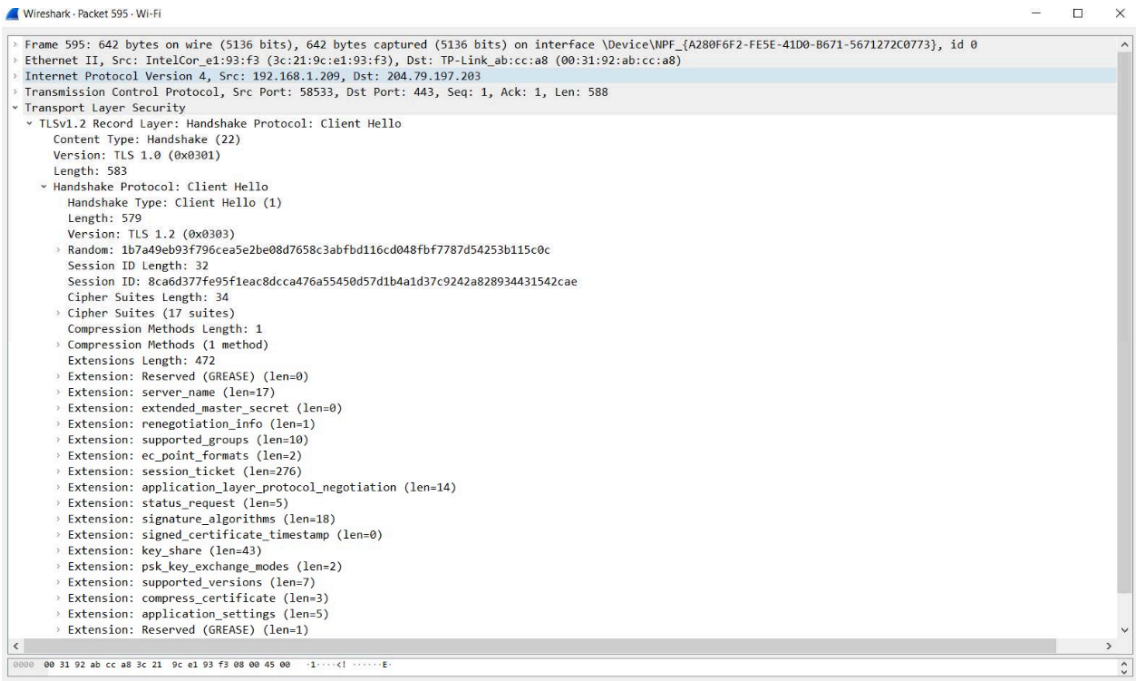

Figure 8 TLS Packet dissection from Wireshark

2.5 Working of TLS

When a client initiates a TLS connection with a server, the following steps take place [6]:-

1. The client sends a "Client Hello" packet to the server, which contains information such as the version of TLS that the client supports, a list of cipher suites that the client prefers, and a random number.
2. The server responds with a "Server Hello" packet, which contains similar information to the client hello packet.
3. The server then sends a digital certificate to the client that lists the public key of the server as well as some identifying information.
4. The client authenticates the server certificate which it receives and the client builds a common secret key for establishing shared link between the client using server public keys and client private key.

5. At this moment client sends "Finished" packet to server, and packets are again encrypted using common secret key. This is a message to the server that client has done its part to key exchange.
6. The server encrypts the "Finished" packet with the session secret key and transmits to the clients.

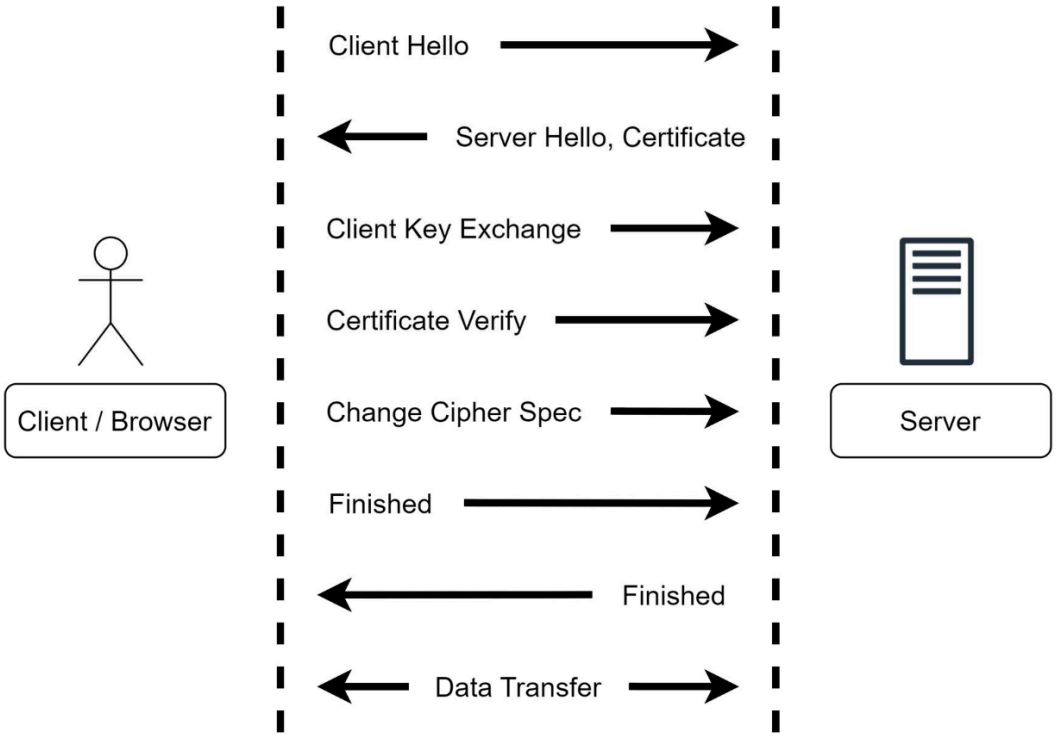

Figure 9 Working of TLS

After the handshake has occurred, then data is exchanged encrypted between client and server. The TLS protocol ensures that every piece of data exchanged between the client and server is encrypted and authenticated, which means it cannot be intercepted or altered by a third party.

2.6 TLS Fingerprinting using JA3

TLS Fingerprinting refers to the identifying of the exact TLS protocol implementation in-use by a client or server whenever a TLS handshake is

tentatively being setup [19]. This attack method includes examining the respective TLS handshake messages with the client and server to find out detail information about how is TLS being implemented for issuant characteristics including cipher suites, protocol version and its extensions as well as other parameters.

While the most common application of TLS fingerprinting today is security, by determining client software that established a TLS connection the method can be used to identify unsecure and outdated TLS implmentations and might be able to detect all sorts of abnormal or even up to newer AV detection mechanisms involving TLS attacks. A common approach to TLS Fingerprinting is to can extract the attributes of a particular TLS implementation and cross-reference these extracted features to check agaist a known database of TLS fingerprints. Some of these are characteristics meant to be seen when TLS implementations issue themselves on the web; while others fingerprint inside browsers, mobile devices, servers and IoT devices [19]. You can infer which TLS implementation is responsible for leaking a lot about the database if you compare the database with the TLS characteristics. The technique can also be abused to identify targets with a vulnerable setup or to keep an eye on user activity. Security Best Practices to Mitigate Risks of TLS Fingerprinting Security best practices associated with TLS fingerprinting include the use of strong encryption algorithms and key lengths, disabling insecure protocols, cipher suites, together with techniques that will obfuscate or modify the present TLS fingerprint [26].

JA3 Fingerprinting is a technique to identify the Software Name/Version of the TLS client that uses the server on an Internet connection. What is TLS formerly known as SSL stands for Transport Layer Security and When you browse the web, many sites use HTTPS to secure your connection using TLS. Kubernetes When one client device set up a TLS connection with a server, it send a handshake message providing several details about tht TLS client software including the cipher suites and version of TLS protocol supported. JA3

Fingerprinting extracts this information from the handshake message to result in creating a signature which identifies the particular client software [20].

JA3 is short for John Althouse Hash, Jeff Atkinson and Josh Atkins (Salesforce) [20][21]. This was unveiled at BSides conference in Delaware, USA from 2017 for the first time. From there, JA3 has taken off and become a de facto way for security researchers, network administrators, and threat intelligence analysts to get a sense of who TLS clients are based off their fingerprint. The methodology was constructed primarily around this insight: that for anytime a new TLS connection is created by the client, there are always unique fields available via the TLS Client Hello packet - as it can be assumed to send one or more times given each server that it is connecting to.

The JA3 fingerprint is computed over hashed values such as [21]:-

- SSL/TLS version
- Supported Cipher List of Client
- List of Supported Elliptic Curves
- If the client supports SSLv2
- When the client does not support TLSv1. 3

This will identify the client software during comparison to a database of known JA3 fingerprints. By analyzing them, one could then use the browser or VPN software or malware to belong to a certain type of client using JA3 fingerprints. While JA3 fingerprints are excellent security tools, they allow down-to-the-behaviour of a specific user fingerprinting as well as server firmware, making them attractive targets of cyber-attacks.As a result, some security experts recommend using techniques to obfuscate or modify the JA3 fingerprint to enhance privacy and security.

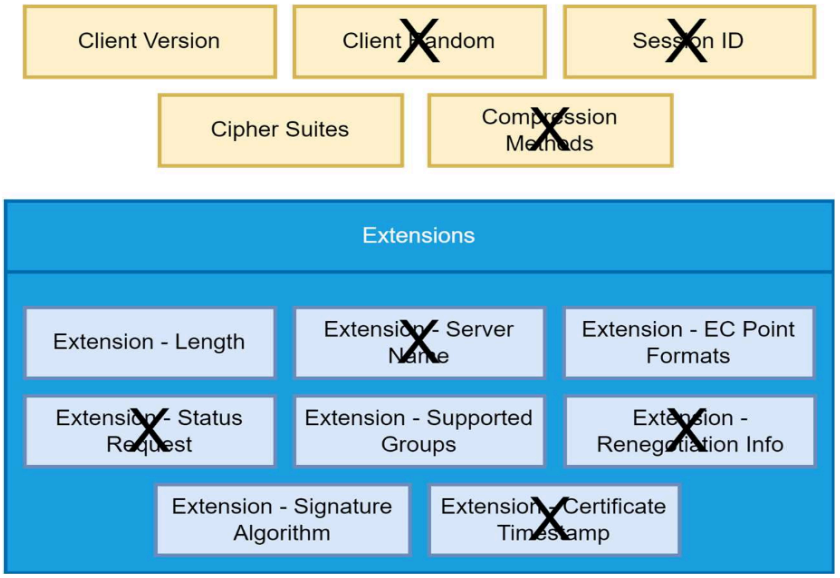

Figure 10 Selection of fields from TLS Client Hello Packet for JA3 fingerprinting

The JA3 will be made by extracting the fields from TLS Client Hello packet in the order of [23]:-

$$Fingerprint = (Version, Ciphers, Extension, EC, EC\_Point\_Formats)$$

These fields are unique and doesn't change with new TLS session. The final JA3 fingerprint will be obtained by taking the hash for these fields as exactly seen in this order.

$$JA3\_Fingerprint = MD5(Version, Ciphers, Extension, EC, EC\_Point\_Formats)$$

It is important to create a MD5 hash of the string in this exact order. The order of the fields should not change because MD5 is very sensitive to minute changes, even a smaller change will result in different hash result.

For example considering the sample result for Microsoft Edge on Android will generate the following values [23]:-

Parameters	Parameter Values
Version	771
Ciphers	4865-4866-4867-49195-49196-52393-49199-49200-52392-49161-49162-49171-49172-156-157-47-53
Extensions	0-23-65281-10-11-35-16-5-13-51-45-43-41
EC	29-23-24
EC_Point_Formats	0

Figure 11 Example of Microsoft Edge TLS Client Hello Packet breakdown

So the resultant JA3 fingerprint will be taking MD5 hash of above values in the correct order as:-

JA3 = MD5(771,4865-4866-4867-49195-49196-52393-49199-49200-52392-49161-49162-49171-49172-156-157-47-53,0-23-65281-10-11-35-16-5-13-51-45-43-41,29-23-24,0) = **c67e9dc27d283f1f89b4ebb4b4670c21.**

Hence, it can be concluded that JA3 fingerprint for Microsoft Edge will be **c67e9dc27d283f1f89b4ebb4b4670c21**.

Similarly other examples are :-

Android Client	JA3 Fingerprint
Discord	9b02ebd3a43b62d825e1ac605b621dc8
Dropbox	59fe159ce36975a15410412d25c2e114
Skype	eaabed81520b23ea8a800b36bd7e359e
Spotify	f79b6bad2ad0641e1921aef10262856b
Google Chrome	cd08e31494f9531f560d64c695473da9
Firefox	6b5e0cfe988c723ee71faf54f8460684

Figure 12 JA3 fingerprint examples for different applications [23]

3 Practical Implementation

3.1 Implementation workflow

In this topic implementation of JA3 to generate the TLS fingerprints will be discussed.

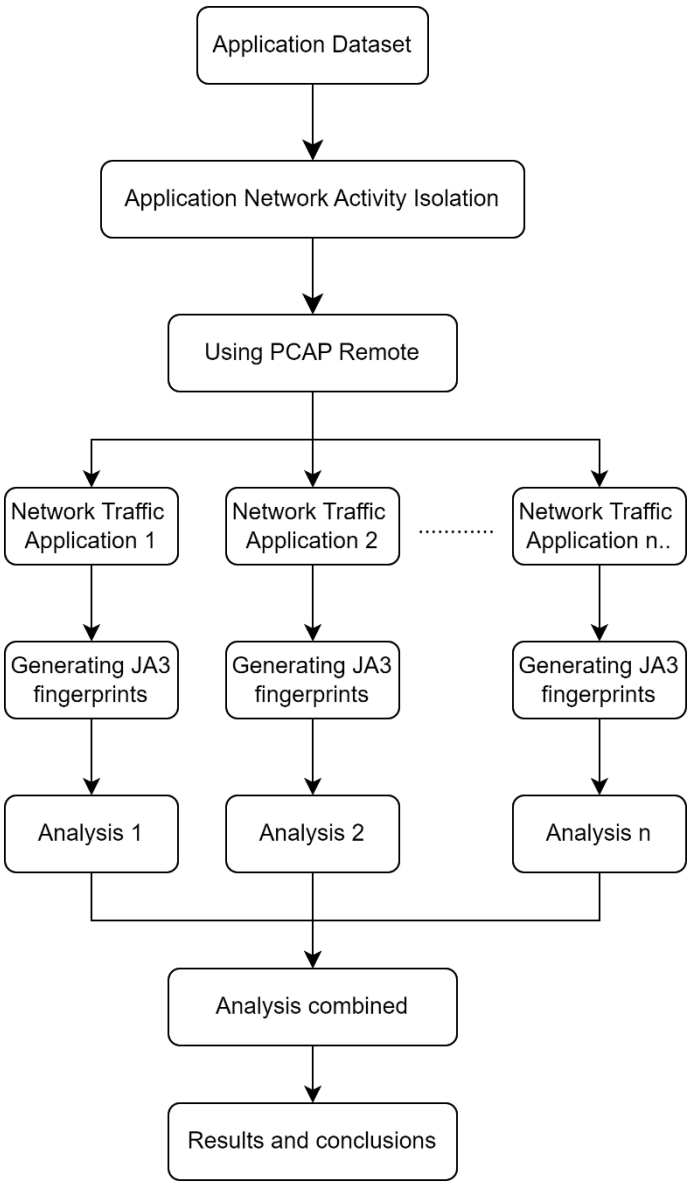

Figure 13 Implementation workflow

To begin the analysis, a group of 12 applications was chosen based on various criteria, such as popularity, type, usage, and industry use. The first task was to isolate the network activity of the chosen application(s), as Android devices generate a lot of background traffic, making it difficult to distinguish between network packets sent or received by a specific application [22]. To address this issue, an additional application named PCAP Remote [23] was utilized to capture the network activity of a specific application. Once the network traffic was generated for a particular application, the TLS client packets were analysed to obtain JA3 fingerprints. The collected data was then analysed. The results from all applications were merged for cross-comparison.

3.2 Dataset generation

The dataset was generated using PCAP Remote android application. The PCAP Remote application uses Android VPN Service [24].

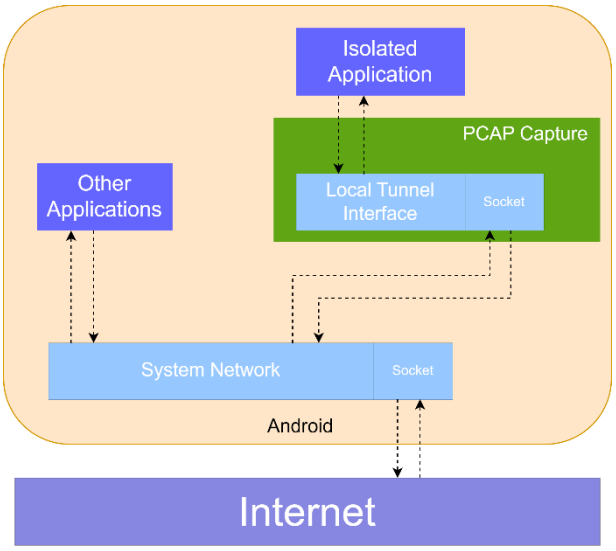

Figure 14 PCAP Remote working with Android VPN Service

This solution creates a virtual interface called the Local Tunnel Interface, which has a dedicated virtual socket. The specific application being analysed is isolated and configured to use PCAP Capture, which directs all of its network

traffic through the Local Tunnel Interface before sending it to the System Network Interface and out to the internet. As packets flow through this pathway, they are captured and stored in the ".pcap" format. Other applications on the device continue to work as usual, utilizing the System Network Interface for their internet communication. In essence, PCAP Remote functions as an intermediary between the isolated application and the System Network Interface.

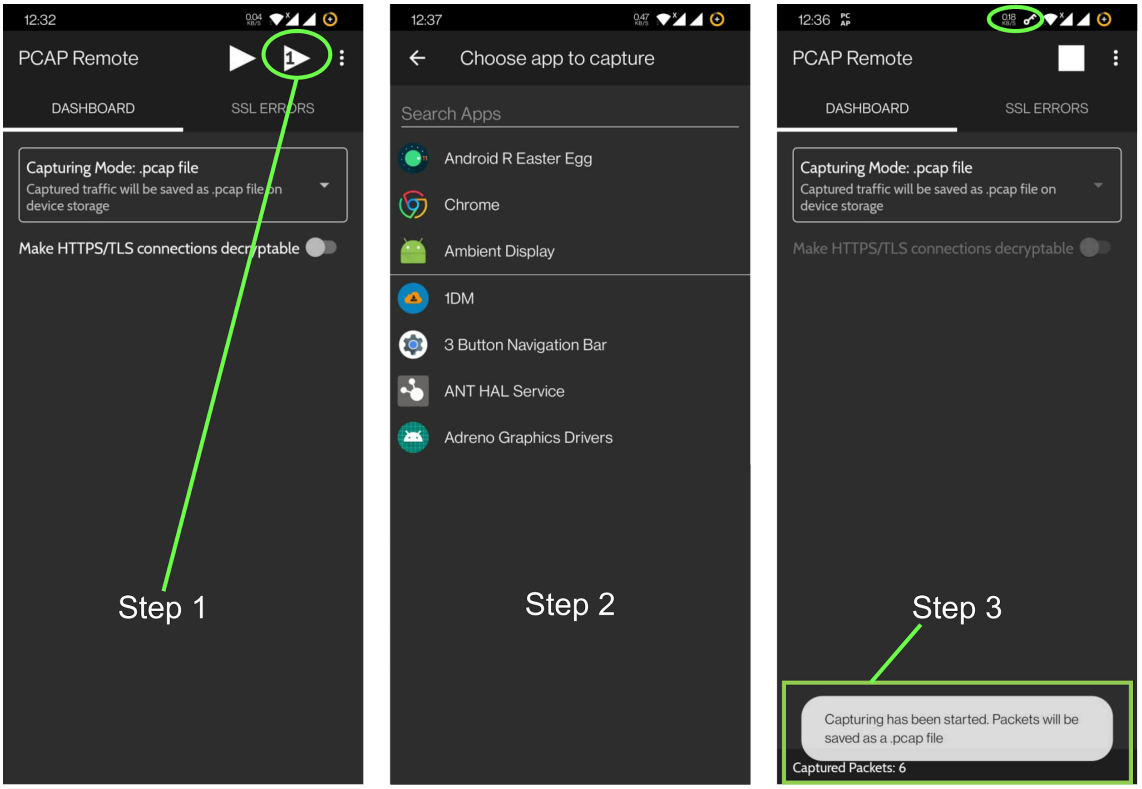

Figure 15 Working of PCAP Remote

The diagram above illustrates how PCAP Remote operates. The first step involves selecting a specific Android application by clicking the button labeled "1". There is another button available, but it captures network data for the entire device, which is not necessary in this scenario. Next, an application is chosen from the list of available applications. Select your Chrome this time, and it starts to capture the network data for that application. Then in the case of stop capture you need to hit on "Stop" button (a white square box which is above right hand corner) and store network capture file any location.

3.3 Viewing Captured Traffic

Wireshark is a free and open source boused network protocol analyser (NPA) software that can capture data packets flowing through a computer netowrk in real time[25]. This free, open-source tool runs on multiple operating systems - Windows, Linux and macOS. Wireshark is primarily used for troubleshooting, network analysis & protocol development etc. It has built-in support for many common network protocols and can decode the payload of these packets to display a clear view on what is happening right now in your network. It also provides built-in filters for narrowing down the search to specific type of packets or traffic out of all captured data[26].

Wireshark is an amazing all in one packet analyzer that allows us to capture, visualize and analyze network traffic in a very user friendly way. It lets you capture packets live or from stored captures. You can see the packets either as a Hex Dump, or Packet Details or Combined summary of all intercepted data. In addition, Wireshark supports many protocols even uncommon ones that make it a versatile way for network analysis. It also offers filtering and search facilities to the users so they can select packet or type of traffic from the capture data.

Wireshark is also powerful in its ability to both dissect and decode packets, or interpret the contents of each packet such that you can take as clear a look at network traffic streams.This is helpful for network diagnostics and also in prototyping new networking protocols. Wireshark also supports a collection of statistics tools including Protocol Hierarchy and Endpoint analysis which give you an overview about the traffic patterns on your network. Wireshark is highly flexible and configurable. Output Dissectors: It allows the users to write their own dissectors which interpret new protocols or change how existing ones are displayed. Wireshark also supports different output formats, so you can export what has been captured for whatever other tools or custom facts and figures. Overall, Wireshark is a powerful and flexible tool that is widely used in the network analysis community.

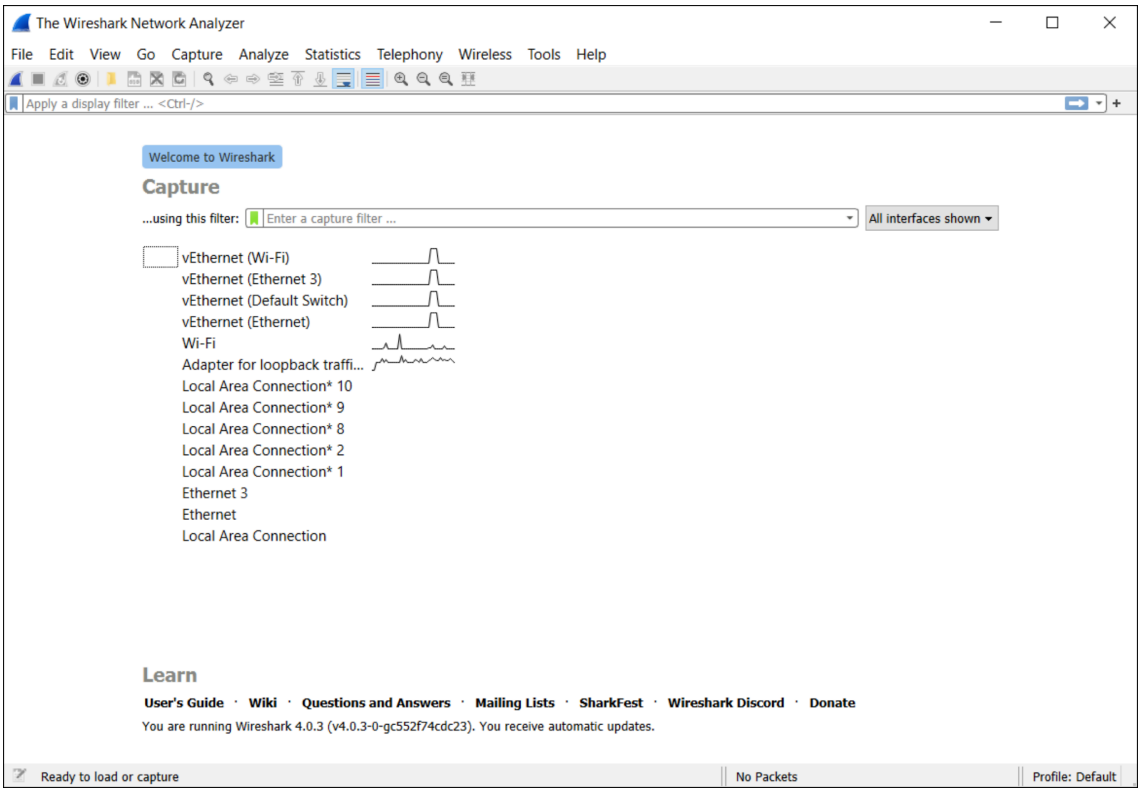

Figure 16 Wireshark

It displays all the network interfaces present on the device from which network packets can be sniffed. Also, different types of network capture files can be opened. Since, the research focuses on TLS packets so after importing a sample pcap file and applying “TLS” as a filter in filter bar, it displays only the TLS packets.

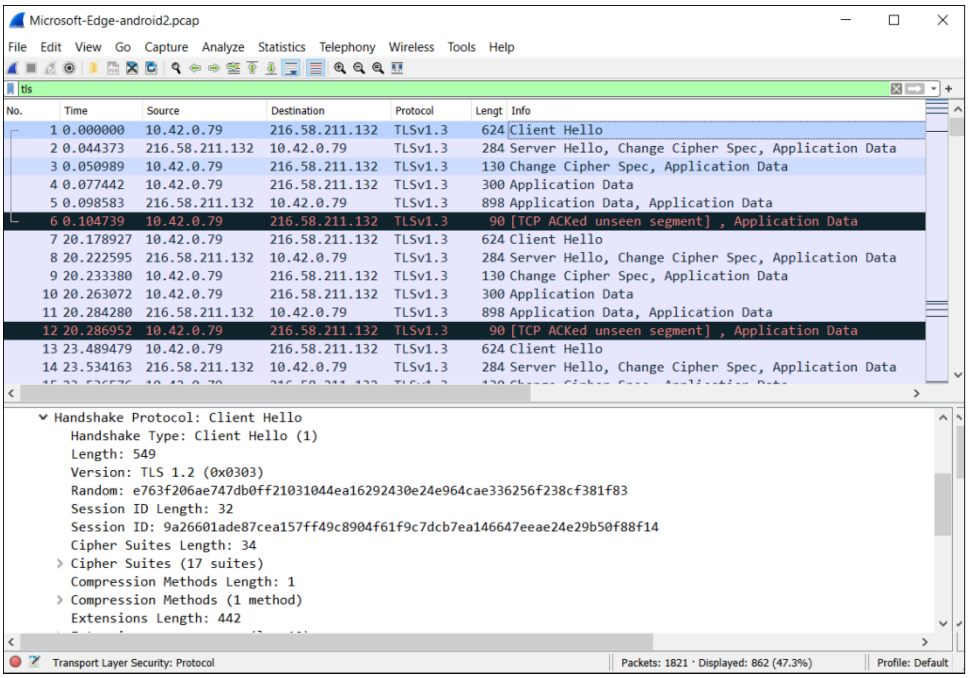

Figure 17 Using filter bar in Wireshark to filter TLS Packets [26]

Generating TLS Fingerprints using JA3

There are two ways to generate TLS fingerprints using JA3: the first method involves using Wireshark, while the second method involves using the JA3 Python library available on GitHub. JA3 fingerprints can be directly viewed in Wireshark, where they are displayed in the description of each TLS packet. They can also be configured to be displayed as a column in the Wireshark packet dissection view. Once configured, the Wireshark packet dissection view can be exported as a CSV file for further analysis.

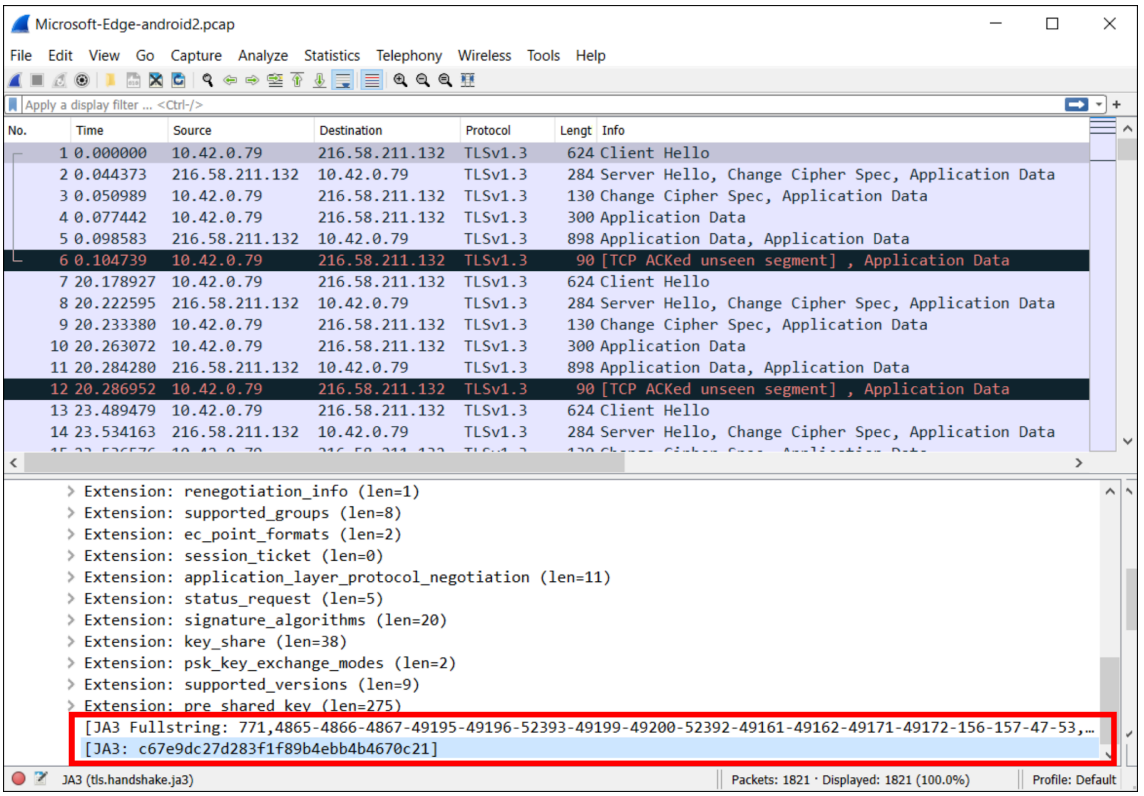

Figure 18 JA3 Fingerprint in Wireshark

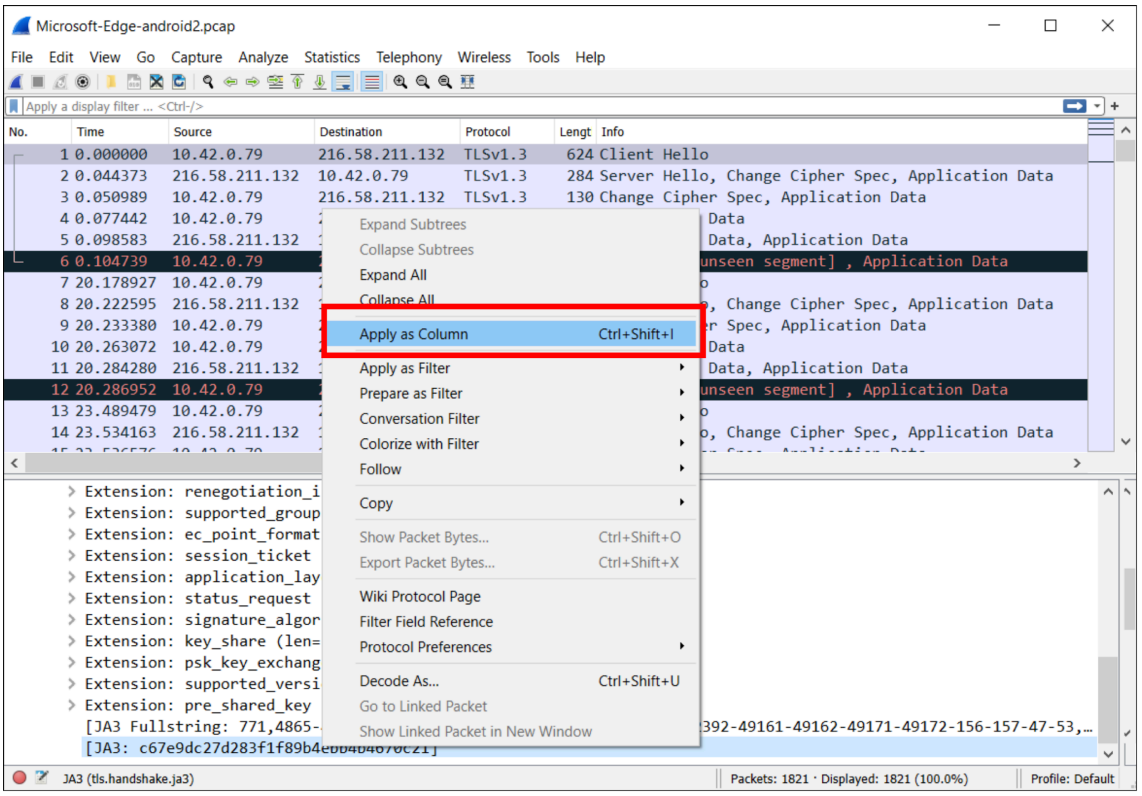

Figure 19 Applying "JA3" & "JA3 Fullstring" to display as column

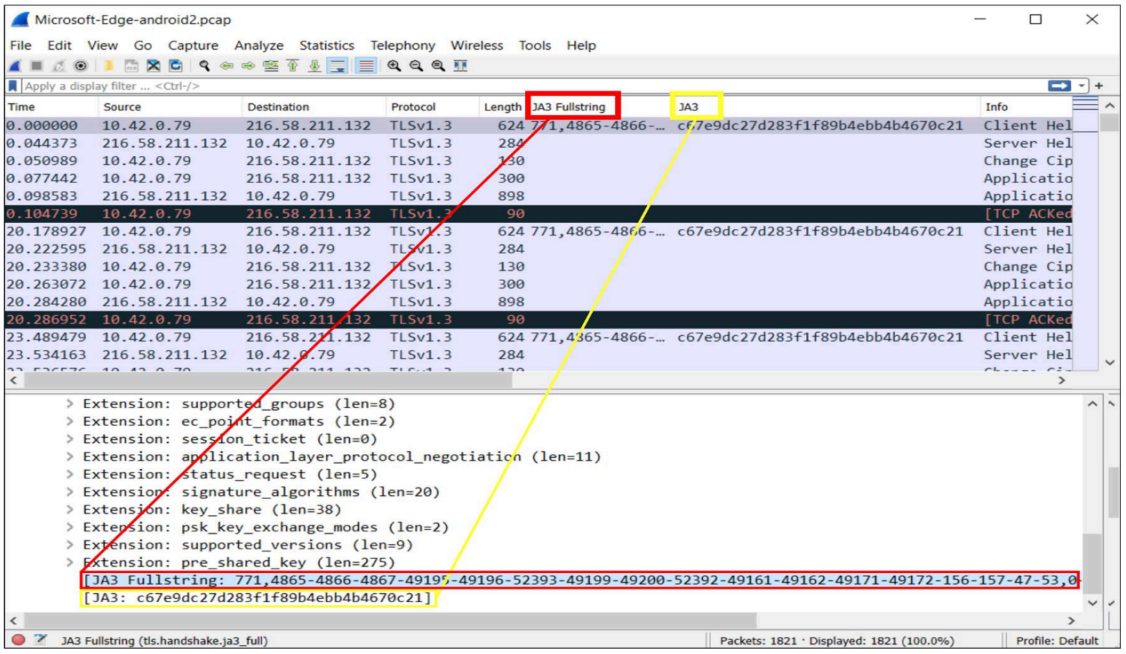

Figure 20 JA3 & JA3 Fullstring applied as column

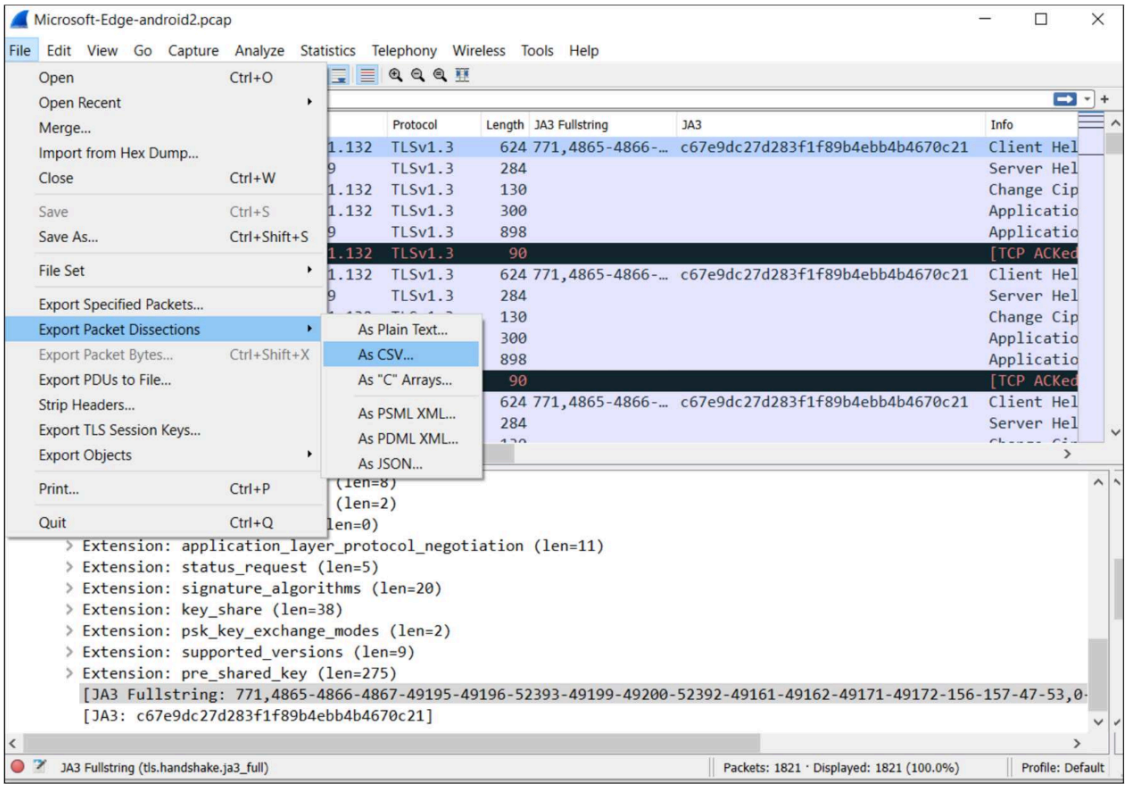

Figure 21 Exporting packets dissections as CSV

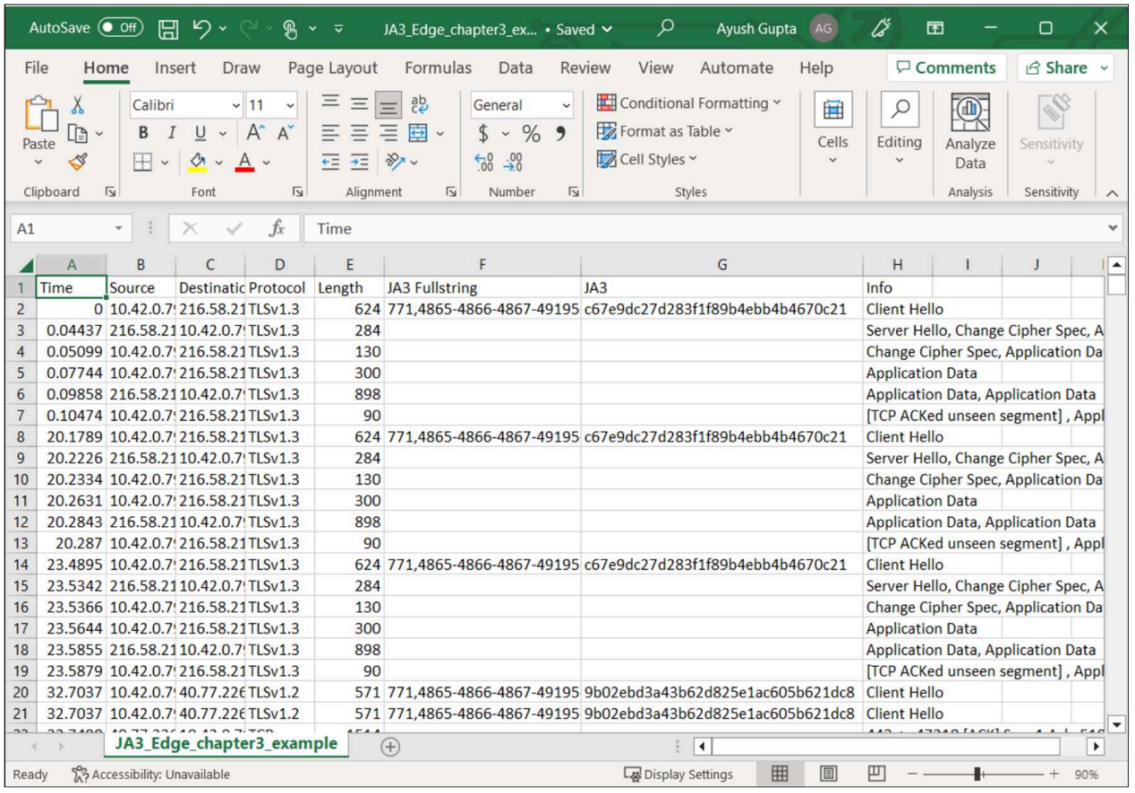

Figure 22 Visualizing exported CSV data of JA3 in excel

JA3 can be generated using the JA3 Python library available on GitHub or by installing the "pyja3" module using Python-pip. The output will be in JSON format, and it can be saved directly to a text file using the pipe operator.

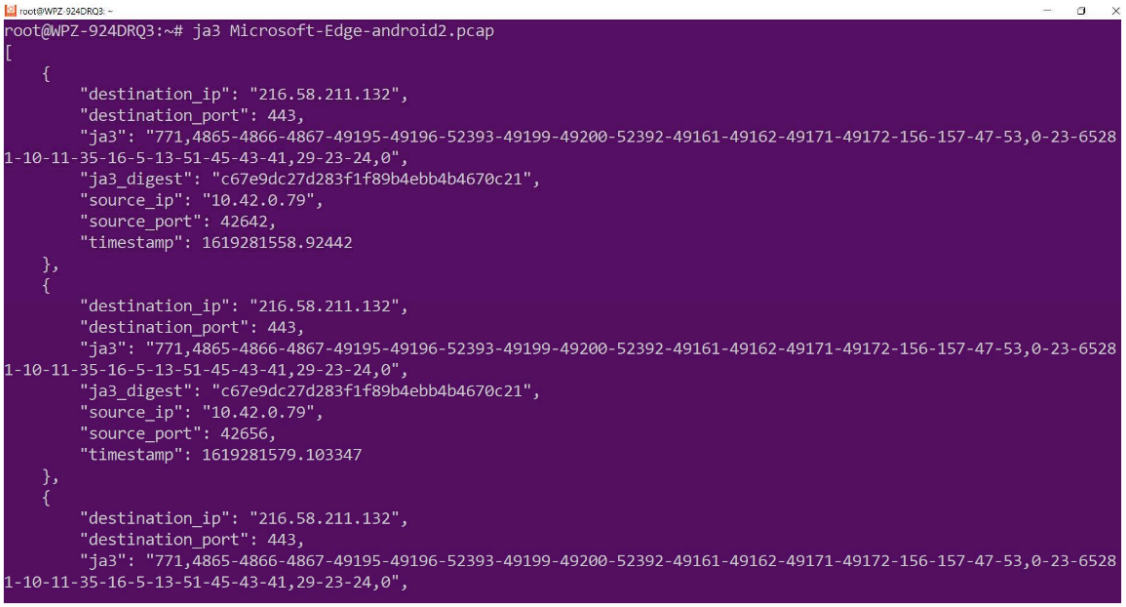

Figure 23 Generating JA3 fingerprints using pyja3 library from python-pip

4 Results

The subsequent sections present the results derived from JA3 of diverse applications operating on Android.

4.1 Applications used for Analysis

Several applications were taken into account for the analysis. Applications considered are as follows :-

1. Epassi
2. Facebook
3. Foli
4. Gmail
5. Google Chrome
6. Google Maps
7. Hsl
8. Linkedin
9. Microsoft Edge
10. ResQ
11. Wolt
12. Youtube

The aforementioned applications selected for analysis are frequently utilized in different scenarios. Based on their usage, they are the most prevalent in their respective categories, and therefore, have been chosen for analysis.

4.2 JA3 results from OnePlus 7 with Android 11

First device OnePlus 7 with Android 11 was used which generated 23 unique fingerprints for all the applications. Microsoft Edge produced the most unique fingerprints (9), while Youtube only produced one. The fingerprints "cd08e31494f9531f560d64c695473da9" and "9b02ebd3a43b62d825e1ac605b621dc8" were the most generated among all the applications. Some applications produced unique fingerprints that were not shared with others. Although Google Chrome and Microsoft Edge are both web browsers, they did not generate any overlapping fingerprints. However, Gmail and Google Chrome had common fingerprints, indicating the use of the same TLS library. The hash "b32309a26951912be7dba376398abc3b" was only generated by Youtube, but further analysis [27][28] suggested that it was produced by web browsers such as Google Chrome and Mozilla Firefox on Windows operating systems, implying that Youtube on Android 11 may be using the same libraries as these softwares.

4.3 JA3 results from Samsung Galaxy A71 with Android 12

Samsung Galaxy A71 with Android 12 was used as the second device, which generated 26 unique fingerprints. Like Android 11, Microsoft Edge produced the highest number of fingerprints (9) compared to the other applications. Most of the applications, including Foli, Gmail, Google Chrome, Hsl, LinkedIn, Maps, Wolt, and YouTube, generated the fingerprint "cd08e31494f9531f560d64c695473da9". YouTube only generated two fingerprints: "598872011444709307b861ae817a4b60" and "cd08e31494f9531f560d64c695473da9". Microsoft Edge and Google Chrome did not share any common fingerprints. The fingerprints generated by Resqclub and Wolt were almost identical.

4.4 JA3 results from Samsung Galaxy S22 with Android 13

The third device used in the study was the Samsung Galaxy S22 running on Android 13. A total of 21 unique fingerprints were identified. The fingerprint "cd08e31494f9531f560d64c695473da9" was the most commonly generated by the majority of the applications (7 times), followed by "9b02ebd3a43b62d825e1ac605b621dc8" which was produced by six applications. It is worth noting that Google Chrome only generated two unique fingerprints, while Microsoft Edge had the highest number of unique fingerprints (6). The Hsl and Foli applications generated the second-highest number of fingerprints. Facebook, Gmail, Linkedin, Google Maps, and Youtube all produced the same number of fingerprints (two). Similar to the previous devices, no fingerprints overlapped between Google Chrome and Microsoft Edge. Youtube was observed to have generated the same fingerprints as Foli. Facebook was the only application that did not share any fingerprints with other applications.

4.5 Application Wise Comparison on Android 11, 12 and 13

The following presents a comparison of the results obtained from the applications running on the three selected operating systems.

1. Epassi

   Across all Android versions, the Epassi application produced four distinct fingerprints. However, for Android 12, a new fingerprint "c67e9dc27d283f1f89b4ebb4b4670c21" was generated, which was not observed in other Android versions. On the other hand, Android 11 and Android 13 had identical fingerprints for the Epassi application.

   Table 1 Epassi JA3 fingerprints

JA3	Android 11	Android 12	Android 13

f79b6bad2ad0641e1921aef10262856b	X	X	X
c67e9dc27d283f1f89b4ebb4b4670c21		X	
d75e7289c86c15b305ac36097bfa0487	X	X	X
9b02ebd3a43b62d825e1ac605b621dc8	X	X	X

2. Facebook

Facebook created a total of six fingerprints across all Android versions. Specifically, it generated five fingerprints on Android 11, four on Android 12, and only two on Android 13. While there are numerous fingerprints on Android 11 and Android 12, there are only two overlapping fingerprints ("0bb1258a19c600139c138bdd9e28894b" and "00c2fab61a044c6d81b8257059977a7f") that are common to all Android versions.

Table 2 Facebook JA3 fingerprints

JA3	Android 11	Android 12	Android 13
9b02ebd3a43b62d825e1ac605b621dc8		X	
0bb1258a19c600139c138bdd9e28894b	X	X	X
61e4063abe913214d3e1a979a7666d73	X	X	
00c2fab61a044c6d81b8257059977a7f	X	X	X
cd08e31494f9531f560d64c695473da9	X		
c3b68ab6f37555233153ad0f8be35f04	X		

3. Foli

The Foli application created a total of five unique fingerprints across all three Android versions. These fingerprints were mostly identical in all three versions, with only one exception: the fingerprint "598872011444709307b861ae817a4b60" was not generated in Android 11, but was generated in Android 12 and Android 13.

Table 3 Foli JA3 fingerprints

JA3	Android 11	Android 12	Android 13
598872011444709307b861ae817a4b60		X	X
9b02ebd3a43b62d825e1ac605b621dc8	X	X	X
cd08e31494f9531f560d64c695473da9	X	X	X
6c0f0a346dcd84cb4b97a0d9382c53fd	X	X	X
e1d8b04eeb8ef3954ec4f49267a783ef	X	X	X

4. Gmail

Gmail produced total of five fingerprints out of which all of them are present in Android 12 but with some overlap with other versions. Only two were observed in Android 13 while three are observed in Android 11. Only one fingerprint “cd08e31494f9531f560d64c695473da9” is common in all the Android versions. Fingerprints observed in Android 13 are all generated in Android 12 as well.

Table 4 Gmail JA3 fingerprints

JA3	Android 11	Android 12	Android 13
6ec2896feff5746955f700c0023f5804	X	X	

598872011444709307b861ae817a4b60	X	X	
312c0b788e784899a8cc07c960430c3a		X	X
cd08e31494f9531f560d64c695473da9	X	X	X
893d25297c894da36fbdee5cea98b01c		X	

5. Google Chrome

Google chrome has generated five fingerprints. There is only one fingerprint that is common across all Android versions. Two unique fingerprints, namely "598872011444709307b861ae817a4b60" and "e1d8b04eeb8ef3954ec4f49267a783ef", were only generated in Android 11. One unique fingerprint, namely "74ad8ec6876e2e3366bfd566581ca7e8", was only generated in Android 12. The fingerprints generated in Android 13 are exactly the same as those in Android 12.

Table 5 Google Chrome JA3 fingerprints

JA3	Android 11	Android 12	Android 13
598872011444709307b861ae817a4b60	X		
74ad8ec6876e2e3366bfd566581ca7e8		X	
0d69ff451640d67ee8b5122752834766		X	X
cd08e31494f9531f560d64c695473da9	X	X	X
e1d8b04eeb8ef3954ec4f49267a783ef	X		

6. Google Maps

Google Maps generated a total of five fingerprints, with only two of them being produced in Android 13. Two fingerprints were generated in Android 12 that were not observed in either Android 11 or Android 13. The fingerprints "893d25297c894da36fbdee5cea98b01c" and "cd08e31494f9531f560d64c695473da9" were found to be common across all Android versions. Android 12 generated more fingerprints, indicating that it uses more TLS libraries than Android 11 and 13. The fingerprint "598872011444709307b861ae817a4b60" was common in Android 11 and Android 12 but not generated by Android 13.

Table 6 Google Maps JA3 fingerprints

JA3	Android 11	Android 12	Android 13
6ec2896feff5746955f700c0023f5804		X	
ee26b1f1aec16d6098768e2c67388ace		X	
893d25297c894da36fbdee5cea98b01c	X	X	X
cd08e31494f9531f560d64c695473da9	X	X	X
598872011444709307b861ae817a4b60	X	X	

7. Hsl

The Hsl app created a total of seven unique fingerprints, all of which were observed in Android 12, with some overlapping in Android 11 and Android 13. While some fingerprints were common in all three Android versions, only one fingerprint ("598872011444709307b861ae817a4b60") was unique to Android 12. The fingerprints from Android 11 have nearly complete overlap with those from Android 13, except for "84cd49a759d15947552e61b46fa5f2ec".

Table 7 Hsl JA3 fingerprints

JA3	Android 11	Android 12	Android 13
f3e8b92336e2ffa7b31fcfb9a4bbb617		X	X
f79b6bad2ad0641e1921aef10262856b	X	X	X
598872011444709307b861ae817a4b60		X	
84cd49a759d15947552e61b46fa5f2ec	X	X	
9b02ebd3a43b62d825e1ac605b621dc8	X	X	X
e1d8b04eeb8ef3954ec4f49267a783ef	X	X	X
cd08e31494f9531f560d64c695473da9	X	X	X

8. Linkedin

For Linkedin application, Android 11 dominates over Android 12 and Android 13. Two fingerprints “9b02ebd3a43b62d825e1ac605b621dc8” and “6ec2896feff5746955f700c0023f5804” are unique to Android 11. Out of all the five unique fingerprints observed, only two fingerprints “cd08e31494f9531f560d64c695473da9” and “e1d8b04eeb8ef3954ec4f49267a783ef” are found to be common in all the Android versions in consideration. Only one overlap is observed in Android 11 and 12 for “598872011444709307b861ae817a4b60”. In this case Android 11 is using more distinct TLS libraries as compared to Android 12 and 13.

Table 8 Linkedin JA3 fingerprints

JA3	Android 11	Android 12	Android 13
9b02ebd3a43b62d825e1ac605b621dc8	X		
cd08e31494f9531f560d64c695473da9	X	X	X

6ec2896feff5746955f700c0023f5804	X		
e1d8b04eeb8ef3954ec4f49267a783ef	X	X	X
598872011444709307b861ae817a4b60	X	X	

9. Microsoft Edge

Microsoft Edge application generated 10 fingerprints which is the highest number among all the applications. Android 11 and 12 have the same fingerprints with some similarity with Android 13. One fingerprint, "f3e8b92336e2ffa7b31fcfb9a4bbb617" is exclusively generated in Android 13. Four fingerprints are unique to Android 11 and 12 and are not observed in Android 13. This implies that Microsoft Edge is likely using the same TLS libraries in both Android 11 and Android 12.

Table 9 Microsoft Edge JA3 fingerprints

JA3	Android 11	Android 12	Android 13
0acf40799ecbd83506d6caeb38514671	X	X	X
f3e8b92336e2ffa7b31fcfb9a4bbb617			X
c67e9dc27d283f1f89b4ebb4b4670c21	X	X	
9b02ebd3a43b62d825e1ac605b621dc8	X	X	X
295b8fa5db7cfec6cf325aae9e1c60b6	X	X	
de4389c26478c9cf885d0b33fd1307cf	X	X	X
04dd53033ee41ead468844c613c3baf2	X	X	
a5804cc8972df3ab8b7ccde6b7a9e0f4	X	X	
f79b6bad2ad0641e1921aef10262856b	X	X	X
24c42d3e77f06f2a8f47f4ebbbfdfaf4	X	X	X

10. ResQ

The ResQ app created a total of five unique fingerprints across all Android versions, and all of them were observed for Android 11. Only one fingerprint, "a5804cc8972df3ab8b7ccde6b7a9e0f4", was unique to Android 11. There was a complete overlap in the fingerprints between Android 12 and Android 13, and they were also observed in Android 11. This indicates that the ResQ app is using the same TLS libraries for all the Android versions being considered.

Table 10 Resq JA3 fingerprints

JA3	Android 11	Android 12	Android 13
c67e9dc27d283f1f89b4ebb4b4670c21	X	X	X
f79b6bad2ad0641e1921aef10262856b	X	X	X
a5804cc8972df3ab8b7ccde6b7a9e0f4	X		
9c815150ea821166faecf80757d8826a	X	X	X
9b02ebd3a43b62d825e1ac605b621dc8	X	X	X

11. Wolt

The Wolt application created a total of six fingerprints for all Android versions. Five of these fingerprints were generated in Android 12. There is complete overlap between the fingerprints observed in Android 11 and Android 13. Two fingerprints, "cd08e31494f9531f560d64c695473da9" and "c67e9dc27d283f1f89b4ebb4b4670c21," are unique only to Android 12. Since Wolt displays identical behaviour in Android 11 and Android 13, it can be concluded that it is using the same TLS libraries for both versions.

Table 11 Wolt JA3 fingerprints

JA3	Android 11	Android 12	Android 13
f79b6bad2ad0641e1921aef10262856b	X	X	X
cd08e31494f9531f560d64c695473da9		X	
9b02ebd3a43b62d825e1ac605b621dc8	X	X	X
c67e9dc27d283f1f89b4ebb4b4670c21		X	
a5804cc8972df3ab8b7ccde6b7a9e0f4	X	X	X
84cd49a759d15947552e61b46fa5f2ec	X		X

12. Youtube

Youtube produced only three unique fingerprints, with one fingerprint unique to Android 11, and no overlapping fingerprints between all three Android versions. Android 12 and 13 had the same behavior. It can be inferred that Youtube is using different TLS libraries on Android 11, while using the same TLS libraries on Android 12 and 13.

Table 12 Youtube JA3 fingerprints

JA3	Android 11	Android 12	Android 13
b32309a26951912be7dba376398abc3b	X		
598872011444709307b861ae817a4b60		X	X
cd08e31494f9531f560d64c695473da9		X	X

5 Detailed analysis using JA3 for Android Applications

5.1 Security implications and concerns

Security-wise, there are pros and cons to employing TLS fingerprinting via JA3 for Android application analysis. When theoretically applied this way, visibility methods have the power to supercharge both threat detection and network analysis - as well as a number of possible vulnerabilities and threats. Privacy violation is a source of concernThe problem with TLS fingerprinting lies in the fact that it leaks information regarding what an individual may transmit through encrypted connections. While stewardship of JA3 fingerprints does not provide access to these sorts, bad actors could use them for profiling the user then connect their target with some manner of breach or abuse such as obtaining credentials during authentication and accessing a bank account. Also, TLS fingerprinting might be used for reconnaissance tasks to identify vulnerable or outdated apps. In this scenario, attackers can use the data to target specific cyberattack classes and exploitation techniques. Analyzing Traffic based on JA3 Fingerprints for such reveals interesting details such as Device IDs, OS Versions, and network info. This raises privacy concerns. TLS fingerprinting can also be used to break some encryption and anonymity tools. Hackers could correlate JA3 fingerprints with legitimate applications so detection evasion was hard to detect. Although it also provides valuable benefits to security researchers and threat analysts, JA3 fingerprinting needs proper protections and a whitehat approach in order not endanger user privacy & safety.

Android apps were found to be doing TLS fingerprinting via JA3 and people freaked out. Data and app monitoring of the users was a worry. Sent the JA3 fingerprints exposes latent user information. These fingerprints expose apps, services and devices that are alive in it as well a network with which they communicate. This leads to a risk of having personal or sensitive data revealed. For example, the JA3 analysis could show- visited websites, protocols used or content of encrypted app-messages. This must be addressed by using more of

the consideration. There are very big issues on privacy implications in user activity tracking. In other words, that we are being surveilled and profiled online without our permission. Fingerprinting for tracking monitoring the behavior of apps even erodes privacy by recording an extensive log of digital engagements. These records could be abused in the interest of advertising, surveillance or even discrimination. By collecting the fingerprints, you could even link these data points together and create a very detailed user profile that would be another massive risk to privacy. JA3 TLS fingerprinting is useful for security research and detection of threats. But it takes full-on privacy protection, such as de-identified data, minimal data collection and user permission. To guarantee the respect and protection of individual privacy rights. Transparency in gathering and using JA3 fingerprints is also key. Trust and accountability cannot be established, unless transparency about privacy policies- How they are conducted. and practices -What is followed on location data-collection-is effectively communicated between their silos of Users & stakeholders. This way, they may fairly cater to privacy issues while striking a fine balance between improving security and respecting the right of users when utilizing TLS fingerprinting for Android apps. The purpose of TLS fingerprinting is to distinguish between the JA3 on Android gadgets. However, what is important to check the performance impact of it. The fact that TLS fingerprinting has a cost does not mean it will block any form of detection, rather depending on how and where in your infrastructure you are using this technique there could be an economic impact for better or worse from the perspective of decision-making. Complexity of generating JA3 fingerprints is one consideration. One example is the analysis of TLS handshake data to decode details about cryptography. This process can be more heavy in CPU and memory. This means higher CPU usage and longer response time of the device. The task of observing network data to retrieve and analyze TLS handshakes as they happen has a tendency to intensify pressure on system components particularly the likes of networking equipment and storage. This load can add strain on the network which could delay communication and hinder apps that rely more heavily on real-time data or immediate user input. For instance, storing and processing JA3 fingerprints have more device storage than the remaining available space. Furthermore,

since TLS fingerprinting requires a conversation between the device and external servers to obtain or update its JA3 databases - amounts of which may increase over time as attackers seek to adjust their toolkits - systems might end up using more data communications resources and slowing networks. Testing for TLS fingerprinting with JA3 on Android devices is more extensive. The impact that the speed of a device and user experience has on system resources is also very important. Everything comes down to benchmarking (lots of it, spanning models and networks and different usage scenarios). This data enable the deployment of Android TLS fingerprinting solutions to make informed choices regarding security gains and performance costs.

5.2 Performance analysis

It is important to measure how much TLS fingerprinting with JA3 slows down Android devices in order not use it. By analyzing TLS handshake packets, we can create JA3 fingerprints for a lot of websites to discover the activity and sign up with proper access. - Doing fingerprinting research on outbound Ja3 comes at additional cost in terms of device resources which could slow stuff down. However, the complexity behind producing JA3 fingerprints is something to look at too. Describing the packets involved in a TLS handshake requires CPU time and memory, both of which can max out your CPU causing devices to be unresponsive. It entails a closer examination of the handshake packets parts. Moreover, the algorithms needed to convert packet data for JA3 fingerprints are CPU-heavy and can be resource-intensive on a device. Short sentences. Long, intricate ones.Consistent word count. In addition, it is necessary for continuous monitoring of network traffic in order to capture and analyze TLS handshake on the fly. The process might be using an exorbitant amount of system resources, such as your network interfaces or a large portion of available storage. End Result - Slower round trip over the network: With only a single RTT being higher, this will make any apps more dependent on fast data transfer or low

latency interactions harder to maintain. Furthermore, the storage and retrieval of JA3 fingerprints and their metadata may require additional device memory. This may alter how the storage performance is by crowding out there storage potential TLS Fingerprinting additional load on devices Typically, devices must frequently communicate with external servers to gather fingerprint data. This could slow things down. It could also use more data.This is why testing TLS fingerprinting on Android should be executed with caution across multiple devices, networks and usage scenarios. However, if you are an English teacher (such as me), it might be a good question to look at exactly how much slower things get and decide where the security gain is worth the speed cost. Before we start using TLS fingerprinting on Android phones, there are many trade-offs to consider very carefully. Using JA3 to do TLS fingerprinting consumes different amount of performance across device model and Android version. This gives an overview of resource consumption, and system behavior changes. Of course, performance can vary depending on hardware specifics, software optimizations and system setting changes. There is however something of note about the computing power and memory size on these devices. That, in turn, impacts how effectively it process the TLS handshake parsing and JA3 fingerprint generation. You may not notice any slowdown using TLS fingerprinting on newer phones with the fast processors and abundant of memory. That said, not all handsets will gain speed (some older or cheaper ones come with less grunt) Android versions also matter. Newer Android has improved handling against TLS fingerprinting and it is not slowing as previously. This technology is not optimized for older Android and it may hangs your device. Performance can also be influenced by other components on the device, like network and storage speeds or background apps among others. This means the fingerprinting method that work in some phones may require calibration on others. Abstract Comparing the performance of TLS fingerprinting on different Android devices and versions is very important. This is valuable information to help solve for any performance problems, design the best resource utilization and ensure that your app performs consistently across the Android ecosystem. By measuring the performance impact on different device configurations and software environments, stakeholders can make informed

decisions about deploying and optimizing TLS fingerprinting solutions for Android apps.

5.3 Retaliatory retaliation

Stakeholders can use the overhead measurements to guide decisions on if and how much performance of an Android app will suffer when deploying as well as tuning TLS fingerprinting solutions in specific device configurations and software environments. To better protect the security and privacy of Android apps, it is necessary to consider potential ways against efficient TLS fingerprinting. One is to transcend the state of JA3 fingerprints or muddy / change them using Traffic mingling methods, provide some randomness in TLS fingerprint spheres like foliages cipher suites and extensions as well as negotiation behaviors on clients that affectly shuffles for an adversary usages looking at their communications feeds Another proper way really clear profile points becomes more robust with deploying encryption tools together anonymisation aid. Network-level defenses Deploy network-based security tools like Intrusion Detection Prevention System (IDPS) Real-time detection of malicious TLS fingerprinting attempts Monitors live-network traffic for patterns or anomalies that may indicate the presence of a file backend, thereby enabling IDPS to connect earlier and protect against potential threats. In addition, adopting a security themed approach with the incorporation of multiple security controls; network segmentation; application firewalls and endpoint securities can proffer holistic solutions in respect to TLS fingerprinting and other personalised cyber threats that target supportive patterns. Exploit less, making it easier for other organizations to strengthen their overall security posture and defend against emerging threats.Overall, if required and they offer countermeasures tailored to specific TLS fingerprint characteristics which, can help reduce its effectiveness and increase the security resilience of Android applications.

A discussion of methods for manipulating or exploiting JA3 fingerprints is necessary to enhance privacy and security with respect to TLS fingerprints. One method JA3 introduces changes to the TLS fingerprint parameters used for fingerprint generation. Through changes to features such as supported cipher suites, TLS extensions, and client behaviours, applications can confuse their fingerprints, making it difficult for adversaries to accurately identify and profile them For example, applications can select cipher suites dynamically or randomly. or you can negotiate TLS parameters based on user-specific parameters, creating a separate JA3 fingerprint for each connection. In addition, the neighbourhood of the neighbours can be affected by the addition of coastal or randomization, and the new technology is used in the implementation of the TLS strikes. Ja3 Fingerprint Points are generated by systematically disturbing fingerprint parameters and monitoring resulting fingerprints, applications can generate authentic JA3 profiles, making it more difficult for adversaries to target network activity to specific applications or devices some accurate -Size design, by implementing work, can help hide the underlying TLS handshake information from outside observers, thus reducing the risk of fingerprint detection and analysis so the decrease. By strategically integrating these spoofing techniques, applications can enhance their privacy and security by making it more difficult for adversaries to view, profile, or transmit their network connections based on JA3 fingerprints it will apply to the s By taking advantage of effectiveness, applications can strengthen their defences against fingerprinting TLS and protect user privacy in an increasingly interconnected digital landscape and in the maintenance.

5.4 Real-World applications

Looking at actual instances this can be used for, we find a mixed bag of use cases in the world of cybersecurity issues: This makes network monitoring an area where TLS fingerprinting could be highly valuable. By examining JA3 fingerprints in network traffic, security teams can discover the applications and

services running on their networks that could potentially reveal security vulnerability exposures, software permissions exposure or malfeasance. Allowing security systems to recognize JA3 fingerprints as resources with previously associated illicit software or compromise indicators allow potential threats, for example command & control executions and connections, malware downloads, data exfiltration attempts etc., to be detected in advance of their active status... In addition, TLS fingerprints are a powerful digital forensics and incident response tool. Forensic analysts use JA3 fingerprints to build historical timelines of web traffic, correlate security incidents and attribute malicious activity back to an application or device. The identification of TLS fingerprints will in turn be identified within the world itself through JA3 from active threat detection to forensics audit all combining it versatility and efficacy as cybersecurity tool, contextually varying on selection between analysis based or just straight usage via JA3 With strategic fingerprinting an organizations overall cyber defense can solidify with confidence against new threats that eventually reiterate.

The key here is that TLS fingerprints can be very effectively integrated into JA3 to provide unparalleled visibility and add results to threat detection operations, like when a network traffic anomaly detect pervasive strange related experience awareness this can flag anomalies or suspicious patterns based on detected characters in the occurrence of XD default behavior output string matching typical activities with malicious activities after joining forces for immense value performing security teams odd man out correct sequences. Moreover, the TLS fingerprinting would also enhance threat intelligence and sharing. Using JA3 fingerprints, security analysts can construct indicators of compromise (IoC) from such inputs as open reporting, honeypots reports and incident reports in order to recognize new threats or shift changes by APT groups more timely; additionally instead of a purely defensive stance after the recent spike increasing amount attack modalities available for use with TLS fingerprinting also aid in identifying possibly vulnerable/outdated software running on an orgs' network. As the comparison results can be connected with JA3 fingerprints that have been mapped to identified vulnerabilities or software versions, security

teams are able to determine areas for remediation and identify suspected ongoing attacks. Moreover, due to TLS fingerprinting is useful also for forensic investigation and incident response as well. Forensic analysts can reconstruct the timeline of events that proceeds a security incident or triggers malicious behaviour by an application/device, obtain indications to be used as evidence in prosecution actions for perpetration legal infractions or aid detection against emerging threats bypass their defences and effectively contain security incidents.

5.5 Future research directions

Providing potential areas for further research and development of TLS fingerprinting using JA3 for Android applications opens up exciting possibilities for enhancing cybersecurity practices and meeting emerging challenge's role. One area ripe for exploration is refining and extending JA3 fingerprinting mechanisms to accommodate the growing variety of Android applications and network protocols As Android applications continue to diversify in functionality and complexity, it is imperative that they develop robust and comprehensive JA3 fingerprint methods that can accurately profiling a wider range of applications and communication patterns of the analysis of fingerprint extraction And other methods to explore, such as machine learning algorithms or deep packet inspection techniques , holds the promise of improving TLS fingerprinting accuracy and performance on Android devices. In addition, investigating the impact of Android platform updates and security improvements on TLS fingerprinting methods represents a fertile area for research. By analysing the variation in TLS handshakes between Android versions and devices, researchers can better understand the implications of the JA3 fingerprint and develop strategies to adapt to security scenarios a in the ongoing process. Furthermore, by integrating TLS fingerprinting with other cybersecurity tools and techniques, such as threat notification systems, intrusion detection systems, and scanning malware scanning systems, can be

provided enhanced effectiveness of JA3 based safeguards actions may enhances. Furthermore, exploring the ethical and legal implications of TLS fingerprinting in Android applications warrants further research. By examining issues such as user consent, data privacy, and regulatory compliance, researchers can help develop ethical guidelines and best practices for the responsible use of TLS fingerprinting techniques role in cybersecurity practices Overall, the analysis of TLS fingerprinting using JA3 for Android applications -The proposal for guidelines offers interesting opportunities to update the state of the art in cybersecurity has developed and developed the future of threat detection and mitigation strategies. The use of JA3 for Android applications to highlight emerging features or technologies that may affect the TLS fingerprinting field provides valuable insights into the future direction of cybersecurity practices. One notable trend is the proliferation of IoT (Internet of Things) devices and the rise of mobile and IoT technologies. As IoT devices become increasingly popular among consumers and enterprises, there is an increasing need to extend TLS fingerprinting techniques to include a wider range of connected devices and protocols to enhance threat detection and security management in this IoT ecosystem such as MQTT (Message Queuing Telemetry Transport) and CoAP (Command Processing Protocol). It includes investigating the feasibility of using JA3 fingerprinting in IoT communication systems. Additionally, the advent of 5G networks and edge computing brings new opportunities and challenges for TLS fingerprinting on Android devices. The proliferation of high speed and low density communications and the shift to decentralized computing infrastructure require the development of small, efficient TLS fingerprinting protocols that can operate in resource limited edge environments. There are opportunities and threats to cryptography. While quantum computing holds the promise of overriding existing cryptographic algorithms used in TLS, it also offers opportunities if quantum resistant encryption schemes and secure communication protocols are to be developed it can withstand quantum attacks. Research on the implications of quantum computing on TLS fingerprints and the development of quantum resistant fingerprints represent important areas for future research and development Furthermore, there is a strong emphasis on privacy creating technologies

fences and architecture placed on the fingers of nations, such as blockchain and decentralized identification systems. And the implications for security Using techniques such as ignorant authentication and decentralized authentication methods, researchers can find alternatives for TLS fingerprinting on Android devices privacy and security have increased Nevertheless, emerging trends and technologies highlight the dynamic nature of cybersecurity TLS highlights the need for new trends and changes in fingerprinting practices.

5.6 Ethical considerations

Discussion of the ethical implications of using TLS fingerprinting techniques, especially in relation to user consent, data privacy, and regulatory compliance, is necessary to ensure the use of this technology act responsibly and transparently in cybersecurity practices the main concern relates to the issue of consent use and awareness. Because TLS fingerprinting monitors and analyses network traffic, often without users' explicit consent or knowledge, it risks violating individual privacy rights and freedoms Users may not realize that their online activities are being reviewed and profiled based on JA3 fingerprints, raising questions about the ethical implications of such analysis. Furthermore, TLS fingerprinting techniques can inadvertently collect and analyse sensitive personal information transmitted over encrypted communications, increasing privacy concerns Also, TLS fingerprinting the use of network monitoring and threat detection purposes raises ethical dilemmas about balancing security and privacy although TLS fingerprinting in detecting security threats and in the reduction Can help, hear if malware infections or data breaches, but also has the potential to violate individual privacy rights and civil liberties Furthermore, legal TLS fingerprinting practices may vary across jurisdictions, posing challenges for organizations the implementation of these techniques is required in accordance with applicable laws and regulations security standards and other information TLS in different contexts may affect the ethical considerations of the

use of fingerprints. It is therefore important for organizations to carefully consider the ethical implications of TLS fingerprinting techniques, including obtaining explicit user consent, providing measures to enhance privacy, and ensure compliance with the appropriate legal framework for reputational or legal responses In addition to risk mitigation, engage in public consultation and collaboration with stakeholders, including those who including users, regulators, and advocacy groups, can help ensure that TLS fingerprinting practices conform to ethical principles, respect user privacy, and contribute to responsible cybersecurity practices get progress. It is therefore imperative that the serious ethical dilemmas faced are examined with regard to JA3 use in Android application monitoring and analysis, as well as suggestsious actions for combing through ethics-related consideration when doing cybersecurity practices. One of the primary ethical concerns is user privacy implications. Android dynamically loads API from Applications, while JA3 also cannot fingerprint applications via app-loaded APIs and cannot profile sine Android libraries are native to the; as a result they can(indirectly) paint an accurate picture about user online activities without proper insight or notification of users; which leads one into questioning the ethical standards behind such research. Moreover, unnecessarily relying on JA3 fingerprinting for threat detection and security management may have the opposite of its intended effect - it can trade in bias unintentionally by targeting wrongfully biased or incorrect application data which will inevitably lead to testing improperly done or further tarnish any individual through inconsistent profiling with an inherently incomplete mechanism. Using JA3 to survey android applications and analyze them can even encounter issues with legality/legal obligations, as organizations navigate the legal-ethical quagmire of data protection laws for surveillance systems against erring on the privacy side. Greetings, makes sure you does not break any laws and respect user privacy so I wanted to share this quick announcement regarding best practices for monitoring and analysing Android apps with the JA3 algorithm All actions undertaken when exploring these metrics should be transparent, accountable, obtain users' consent in order to justify such exploration of their data (exercise #1: project-specific transparency), employ due care settings where special attention is given keeping your

research within ethical bounds (covers situations like taking on board ethics challenges by doing an annual review) fwiw. Addressing the ethical dilemmas and education challenges endemic to JA3 implementation in cybersecurity practices is also consistent with fostering trust, but it means holding commercial interests more accountable while respecting user privacy.

5.7 Compared to other methods

Comparing TLS fingerprinting using JA3 to other traffic inspection methods or fingerprints such as DPI (Deep Packet Inspection) or JA3S provides valuable insights into robustness, limitations, and transmission are generally used in cybersecurity scenarios At the level of application, the content of network packets is managed. While DPI provides detailed visibility into network traffic and application behaviour, it is often resource intensive and may raise privacy concerns due to its malicious nature Unlike TLS fingerprint extraction using JA3 cryptographic parameters and determine application characteristics, examine TLS handshakes focused packets, providing a lightweight and privacy-protecting alternative to DPI. However, JA3 may be less effective in situations where protocols other than TLS are used to encrypt traffic or when applications use encryption evasion techniques Furthermore, JA3S, an extension of JA3 that includes server side fingerprinting , the specific server used and programs Identify includes provide additional context and granularity Although JA3S enhances the richness of fingerprint data, it can introduce additional complications and costs, especially in areas where there are different server architectures or dynamic server configurations. By comparing and contrasting these options, organizations can make informed decisions on the most appropriate approach to their specific cybersecurity needs, they can deliver effortless, efficient operations, and a balance of privacy security for Overall, comparing TLS fingerprinting using JA3 with other traffic inspection methods provides valuable insights towards evolving cybersecurity techniques

issues and reports best practices for monitoring network infrastructure and detecting threats.

To perform an Android usage analysis, an overview and assessment of the possible advantages and disadvantages of TLS fingerprinting using JA3, DPI, including JA3S, etc. must be conducted to establish which of the methods are the most efficient and proper for cybersecurity practice and analyze TLS hand wrapped packets without sending encrypted traffic but identifying and profiling Android apps by their communication protocols and in so doing reducing risk and the use of resources. Furthermore, JA3 also provides sufficient information concerning application-specific behavior and network usage, which is beneficial for managing threats and security-related visualizations without compromising user privacy. However, TLS fingerprinting using JA3 has several significant limitations: first, when the traffic is encrypted by protocols not encapsulated by TLS and if the apps use custom encryption or obfuscation. In addition, since Android apps and network protocols are too diverse to cope with JA3 may be limited in its effectiveness_int hat it can produce false positives or misclassifications. DPI looks at packet content on the application layer, provides more granularity in terms of network traffic performance and app behavior. DPI can detect the specific actions of Android apps, which is well-suited for effective threat analysis. But DPI is resource hungry and privacy invasive. Moreover, DPI may encounter issues in deciphering encrypted traffic and this demands extractions methods which might breach user privacy as well legal and ethical boundaries. JA3S complements TLS fingerprinting, too by the server-side matching of JA3 fingerprints to better make sense in an Android application context: When making use of inconsistent server architectures or continually changing configurations at a large organization, as new information comes forth surrounding specific servers that appear vulnerable via one means (perhaps missing patches), having more detailed forensic-level data can help defenders conduct risk assessment and threat hunting operations.Therefore, organizations can to determine which approach is best for their specific cybersecurity needs and objectives Of the techniques disclosure trade off between data utility efficiency privacy security Compliance in addition In networks, analysis of these

native synergies And possible integrations throughout Android provide opportunities improved overall network performance Enhancing measurement technology covering fields detection mitigation Threats.

5.8 Integration of security systems

Exploring how TLS fingerprinting using JA3 can be integrated into existing security systems or tools to provide better threat detection and prevention on Android devices offers promising intrusion opportunities strengthen security defences and protect against emerging threats. By integrating JA3 fingerprinting into existing security systems, such as intrusion detection systems (IDS), network security appliances, or endpoint protection platforms, organizations can use JA3 as a lightweight and flying option effectively identifying and classifying Android applications based on their network behaviour. By adding JA3 fingerprints to threat and intelligent feed signature databases, security devices can also proactively detect and prevent malicious activity, such as command and connection execution, malware infection, or attempts to data will be extracted, in real time using JA3 fingerprinting and machine learning algorithms together or behavioural analytics techniques to enable dynamic optimization of ongoing attack engineering processes to improve the accuracy and performance of threat detection well Additionally, enforce JA3 fingerprint protection policies and obtain Android applications based on those detected JA3 fingerprints Mobile to instrument can be deployed in device management (MDM) solutions or application control frameworks. By using JA3 as the foundation of existing security frameworks and tools, organizations can increase the ability to detect and prevent security threats on Android devices, reducing the risk of a breach data over or cyber attacks, and secure critical information and assets but, including resource limitations, scalability and compatibility with legacy systems, it is important to consider the potential limitations and challenges of JA3 integrating fingerprints into existing security systems. By addressing these challenges through careful design, implementation, and

optimization, organizations can realize the potential of TLS fingerprinting with JA3 the overall role increases threat detection and prevention. Discussing the feasibility and benefits of adding TLS fingerprinting to a complete security system using JA3 is essential to understanding the potential impact and effectiveness on real world computers security practices over under TLS fingerprinting provides a minimal privacy protecting mechanism to identify and intercept Android applications based on their network behaviour Risk in Ksha frameworks. An attractive option to enhance detection and prevention capabilities Through JA3 fingerprinting integrated with existing security systems, organizations can leverage its unique capabilities to increase visibility, accuracy and responsiveness in detecting and mitigating security threats on Android devices. Additionally, TLS fingerprinting using JA3 can support other security mechanisms and technologies to provide additional context and granularity for threat analysis and decision making, such as intrusion detection systems (IDS), network monitoring tools, endpoint protection platforms plus the scalability and use of TLS fingerprinting for large scale security architectures makes it well suited, This enables organizations to effectively manage and secure a variety of Android devices and applications but must overcome the concepts and challenges of practicality are also handled when TLS fingerprinting using JA3 is added to a full security policy. These include resource constraints, such as computing costs and network bandwidth, connectivity and existing systems and protocols, and regulatory compliance requirements, such as data privacy and legal considerations. Additionally, when implementing TLS fingerprinting techniques in advanced security systems, organizations should consider the potential impact on user experience and system performance, to ensure the security measures will not compromise functionality or performance by carefully testing the feasibility and benefits of adding TLS fingerprints to full security architectures Through implementation, organizations enhance their cybersecurity posture, enforcing the ability to detect and prevent threats strongly, effectively protecting Android devices from emerging security threats.

5.9 Exploring the dangers of TLS Fingerprinting and analysing traffic

Exploring the dangers of TLS fingerprinting and analysing traffic is key in today's digital landscape, with growing privacy and security concerns especially methods like JA3 enable TLS fingerprinting to detect and describe Android applications based connections encrypted system although the method serves legitimate security purposes, however , because the privacy and security of the user poses a serious risk. One of the biggest concerns is that it can expose sensitive user data sent over encrypted connections, because TLS fingerprints can reveal insights into application usage patterns and behaviour without user consent Now also, the increasing use of TLS fingerprinting methods user privacy, consent, and data security This increases ethical challenges available. Users may not be aware that their online activities are being monitored and analysed, leading to concerns about intrusive reviews and tracking practices Furthermore, TLS fingerprinting carries the risk of false positives and misclassifications occur, which can lead to improper investigations or loss of reputation among innocent individuals or organizations. The ways in which our research can be used for TLS fingerprinting and traffic analysis should be carefully considered, according to the risks posed by them as well as their benefits with regards to user privacy (or lack thereof), security...and ethical considerations.

Educating Android users about practices pertaining to risk reduction on research is a key part of empowering individuals in order to safeguard their privacy and security as they are increasingly interconnected within the digital sphere. After all, with everyone having a smartphone and living life on the web these days it is important for Android users to know ways not only how to utilize TLS but also make sure they are protected from things like Fingerprint traffic analysis. The only weapon users got is the level of awareness about threats to their privacy and strategies they can implement to safeguard data by updated educational programs i.e a combination of sedulous campaigns on systems, educative materials or enlightening sessions that are customized expressly towards Android user. Why having secure password & authentication methods

are the critical to keep your information safe. How sharing too much of your data become a risk online. Users should also be informed about the surveillance tools (e.g.TLS fingerprinting, traffic analysis) that can potentially compromise their privacy rights and how malicious actors might exploit them Users of emerging digital research strategies must prioritize proactive measures designed to actively Address ways to preventatively control risks at baseline (eg: regular security updates for software, responsible use of public WiFi networks or downloading applications from untrusted marketplaces). We believe an informed Android user is essential to a sound and safe mobile ecosystem for all, so we will continue striving towards different methods of informal education. Through the collective efforts of industry partners, government agencies and advocacy groups, we can work to build an informed and privacy-conscious society, where individuals have the confidence and resources to safely navigate the digital landscape and obligation.

6 Conclusions

The main objective of the thesis was to examine the TLS packets of multiple mobile applications using the JA3. The research revealed that many applications utilize identical TLS libraries resulting in the same JA3 hashes [29][30]. However, some applications use different libraries even across different Android operating systems. It is crucial to include other popular browsers such as Opera or Safari in future studies since web browsers produce more hashes per operating system. Additionally, since Apple devices are widely used, it is recommended to consider "iOS" on Apple smartphones for future research work.

The fingerprint “cd08e31494f9531f560d64c695473da9” is observed to appear in all the results for majority of the applications on all the Android versions. After conducting a deeper investigation, it was discovered that the hash in question is also produced by a malicious software called "Wave Browser_6xz4wdjd_.exe" on Windows OS when analysed on the website "joesandbox.com" [31], but other sources suggests that it is also generated by “nginx-ssl” [32]. So different research exists for this fingerprint hence correct conclusion cannot be deduced for this fingerprint. One more observation can be deduced is that for all the Android operating systems Google Chrome and Microsoft Edge didn’t share any common fingerprints. Also, applications with same parent developers like Google Chrome, Maps and Gmail, they share some common fingerprints.

There are some drawbacks for JA3 which need to be considered [33]. Firstly, it relies on the assumption that every client will send the same set of extensions in the ClientHello message. However, some clients may choose to omit certain extensions or send them in a different order. This can result in different JA3

hashes being generated for the same application, making it less reliable for fingerprinting. It is due to the fact that MD5 hash significantly depends upon input hence even if a single character in an input changes, final value of whole generated md5 hash also varies. Second, the contents of these ServerHello messages (which can differ between servers) are not considered in JA3. Two servers running the same application with different configurations may produce a ServerHello messages that are compared to discover the JA3 hash differences The third limitation is that with only JA3, we cannot infer what the traffic itself contains. While this implies that it is incapable of detecting attacks leveraging the context in which data was obfuscated such as data exfiltration, malware communication. Given the results, it is clear that fingerprints produced for some apps have overlap with those of other applications hence its becoming difficult to distinguish. Hence need a better way to address this which requires more analysis and work in the area. New techniques such as JARM and CYU have been built in order to increase JA3 precision while also drastically reducing fingerprint collisions across different applications, which can be combined with the use of JA3. JARM is another JA3 like technique that gives us information on Applications and version based on their SSL/TLS handshake [34]. However, JARM uses more than only the Client Hello packet by combining information also from other handshake packets such as Server Key Exchange with respects to SSLLabs definition [35]. This is handy when attempting to fingerprint more accurately, as the data in such packets may differ between various versions of an app or even separate servers sharing one and same app. JARM is able to determine if a TLS connection uses well-known malware-related extensions or the GREASE extension which can be used for detecting SSL/TLS stripping attacks. This is a good method to detect bad traffic and recognize compromised systems. Similar to JA3, JARM is a passive fingerprinting approach that does not require decryption of the TLS traffic and hence can be used for network security monitoring/ threat intelligence purposes.

The cursor was tracked using CYU fingerprinting [36][37], a Client-side Identification, to extract information about the user's browser and device. This is done through the JavaScript code which allows it to gather information on your

browser type, screen resolution, installed plugins, timezone offset and other variables device specific details. This is then used to create an identifier or "browser fingerprint" for the given device. Tracking user behaviorIdentifying returning visitorsDetecting fraud activitiesThe CYU fingerprint method can be applied for many purposes. However, as other fingerprinting methods do CYU fingerprinting has been also criticized for a possible problems with privacy and security of user. Overall, JA3, JARM, and CYU all have their strengths and weaknesses, and their effectiveness depends on the specific use case and the nature of the TLS clients being analysed.

References

1. SSL Guide – the complete guide to SSL/TLS certificates [Internet]. About SSL. digicert; [cited 2023Mar1]. Available from: https://aboutssl.org/ssl-guide/

2. Qinglin G, Dressman M. Microsoft Security Advisory 3009008 [Internet]. Microsoft Security Advisories. Microsoft; 2022 [cited 2023Mar3]. Available from: https://learn.microsoft.com/en-us/security-updates/securityadvisories/2015/3009008

3. Kiprin B. How to Prevent SSL POODLE Attack [Internet]. CRASHTEST SECURITY. VERACODE; 2021 [cited 2023Mar3]. Available from: https://crashtest-security.com/prevent-ssl-poodle-attack/

4. THE EVOLUTION OF SSL AND TLS [Internet]. Digicert Blog. digicert; 2015 [cited 2023Mar3]. Available from: https://www.digicert.com/blog/evolution-of-ssl

5. Track the evolution of the SSL/TLS marketplace from its inception [Internet]. Netcraft; [cited 2023Mar3]. Available from: https://www.netcraft.com/internet-data-mining/ssl-survey/

6. Prodromou A. TLS Security 5: Establishing a TLS Connection [Internet]. THE ACUNETIX BLOG - WEB SECURITY ZONE. ACUNETIX ; 2019. Available from: https://www.acunetix.com/blog/articles/establishing-tls-ssl-connection-part-5/

7. Satapathy A, Livingston J. A comprehensive survey on SSL/ TLS and their vulnerabilities. International Journal of Computer Applications. 2016Nov17;153(5).

8. Alkazimi A. TLS fallback signaling cipher suite value (SCSV) for preventing protocol downgrade attacks. 2015;

9. Adjei HAS, Shunhua T, Agordzo GK, Li Y, Peprah G, Gyarteng ES. SSL stripping technique (DHCP snooping and ARP spoofing inspection). 2022 24th International Conference on Advanced Communication Technology (ICACT). 2021Mar5;

10. Alashwali ES, Rasmussen K. What's in a downgrade? A taxonomy of downgrade attacks in the TLS protocol and application protocols using

TLS. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. 2018Dec29;:468–87.

11. Luedtke D. Secure Sockets Layer Transport Layer Security BEAST Attack. Munich: University of the German Federal Armed Forces; 2012.

12. Satapathy A, Livingston J. A comprehensive survey on SSL/ TLS and their vulnerabilities. International Journal of Computer Applications. 2016Nov;153(5):31–8.

13. Kyatam S, Alhayajneh A, Hayajneh T. Heartbleed attacks implementation and vulnerability. 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT). 2017Aug;

14. Aviram N, Schinzel S, Somorovsky J, Heninger N, Dankel M, Steube J, et al. DROWN: Breaking TLS using SSLv2 [Internet]. USENIX; 2016 [cited 2023Mar5]. Available from: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/aviram

15. Kerner SM. Logjam SSL/TLS Vulnerability Exposes Cryptographic Weakness [Internet]. eWEEK; 2015 [cited 2023Mar5]. Available from: https://www.eweek.com/security/logjam-ssl-tls-vulnerability-exposes-cryptographic-weakness/

16. Merget R, Somorovsky J, Aviram N, Young C, Fliegenschmidt J, Schwenk J, et al. Scalable scanning and automatic classification of TLS padding Oracle Vulnerabilities: Proceedings of the 28th USENIX Conference on Security Symposium [Internet]. Guide Proceedings. USENIX; 2019 [cited 2023Mar5]. Available from: https://dl.acm.org/doi/10.5555/3361338.3361410

17. Stricot-Tarboton S, Chaisiri S, Ko RKL. Taxonomy of man-in-the-middle attacks on HTTPS. 2016 IEEE Trustcom/BigDataSE/ISPA. 2016Aug;

18. SSL vs. TLS – What are differences? [Internet]. SSL2BUY - Information Technology Journal. SSL2BUY; [cited 2023Mar5]. Available from: https://www.ssl2buy.com/wiki/ssl-vs-tls

19. Gancheva Z, Sattler P, Wüstrich L. TLS Fingerprinting Techniques [Internet]. Technical University of Munich, Germany; 2020 [cited 2023Mar5]. Available from: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2020-04-1/NET-2020-04-1_04.pdf

20. Althouse J, Atkinson J, Atkins J. TLS Fingerprinting with JA3 and JA3S [Internet]. Salesforce Engineering Blog. Salesforce; 2015 [cited

2023Mar6]. Available from: https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967

21. Althouse J, Atkinson J, Atkins J. Open Sourcing JA3 [Internet]. Salesforce Engineering Blog. Salesforce; 2015 [cited 2023Mar6]. Available from: https://engineering.salesforce.com/open-sourcing-ja3-92c9e53c3c41/

22. Design for seamlessness [Internet]. Android Developers. Google Developers; [cited 2023Mar12]. Available from: https://developer.android.com/guide/practices/app-design/seamlessness

23. Egorov A. Receiving captured packets in Wireshark from PCAP Remote running on Android emulator [Internet]. Andrey Egorov | Dev blog & Portfolio. Andrey Egorov; 2019 [cited 2023Mar6]. Available from: https://egorovandreyrm.com/

24. VpnService [Internet]. Android Developers. Google Developers; [cited 2023Mar6]. Available from: https://developer.android.com/reference/android/net/VpnService

25. Wireshark · go deep [Internet]. Wireshark. Wireshark Foundation; [cited 2023Mar6]. Available from: https://www.wireshark.org/

26. What is wireshark and how to use it: Cybersecurity: Comptia [Internet]. CompTIA; [cited 2023Mar6]. Available from: https://www.comptia.org/content/articles/what-is-wireshark-and-how-to-use-it

27. Randomize the TLS fingerprint [Internet]. Get-Set, FETCH; 2022. Available from: https://getsetfetch.org/blog/tls-fingerprint.html

28. O'Hanlon E. Seem to be getting wrong hashes for JA3? [Internet]. Suricata - Open Information Security Foundation. Redmine; 2021 [cited 2023Mar7]. Available from: https://redmine.openinfosecfoundation.org/issues/4435

29. Matoušek P, Burgetová I, Ryšavý O, Victor M. On reliability of JA3 hashes for fingerprinting mobile applications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. 2021Feb7;:1–22.

30. Novák P, Oujezský V. Detection of malicious network traffic behavior using JA3 fingerprints. Proceedings II of the 28st Conference STUDENT EEICT 2022: Selected papers. 2022;:194–7.

31. Windows Analysis Report Wave Browser_6xz4wdjd_.exe [Internet]. JOE Sandbox Cloud. Joe Security LLC ; [cited 2023Mar6]. Available from: https://www.joesandbox.com/analysis/470994/0/html

32. Golovanov M. JA3 on guard against bots [Internet]. WAF.NINJA. WAF.NINJA; 2022 [cited 2023Mar7]. Available from: https://waf.ninja/ja3-on-guard-against-bots/

33. Perez G. JA3 Fingerprinting: Functionality, Pitfalls, and Future Outlook [Internet]. ENTERPRISE SECURITY - SECURITYTRAILS BLOG. Security Trails; 2021 [cited 2023Mar7]. Available from: https://securitytrails.com/blog/ja3-fingerprinting

34. Easily Identify Malicious Servers on the Internet with JARM [Internet]. Salesforce Engineering Blog. Salesforce ; 2017 [cited 2023Mar7]. Available from: https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a/

35. Sosnowski M, Zirngibl J, Sattler P, Carle G, Grohnfeldt C, Russo M, et al. Active TLS stack fingerprinting: Characterizing TLS Server deployments at scale [Internet]. Technical University of Munich. Technical University of Munich - Department of Informatics; 2022 [cited 2023Mar7]. Available from: https://doi.org/10.48550/arXiv.2206.13230

36. The state of TLS fingerprinting: What's Working, What Isn't, and What's Next [Internet]. Fastly Security Research Team; 2022 [cited 2023Mar7]. Available from: https://www.fastly.com/blog/the-state-of-tls-fingerprinting-whats-working-what-isnt-and-whats-next Yu C. GQUIC Protocol Analysis and Fingerprinting in Zeek [Internet]. Salesforce Engineering Blog. Salesforce; 2019 [cited 2023Mar7]. Available from: https://engineering.salesforce.com/gquic-protocol-analysis-and-fingerprinting-in-zeek-a4178855d75f/

JA3 results from OnePlus 7 with Android 11

JA3	Chrome	Edge	Epassi	Facebook	Foli	Gmail	Hsl	Linkedin	Maps	Resqclub	Wolt	Youtube
c3b68ab6f37555233153ad0f8be35f04				X								
295b8fa5db7cfec6cf325aae9e1c60b6		X										
598872011444709307b861ae817a4b60	X					X		X	X			
de4389c26478c9cf885d0b33fd1307cf		X										
9b02ebd3a43b62d825e1ac605b621dc8		X	X		X		X	X		X	X	
84cd49a759d15947552e61b46fa5f2ec							X				X	
cd08e31494f9531f560d64c695473da9	X			X	X	X	X	X	X			
61e4063abe913214d3e1a979a7666d73				X								
893d25297c894da36fbdee5cea98b01c									X			
0bb1258a19c600139c138bdd9e28894b				X								
6c0f0a346dcd84cb4b97a0d9382c53fd					X							
e1d8b04eeb8ef3954ec4f49267a783ef	X				X		X	X				
6ec2896feff5746955f700c0023f5804						X		X				
00c2fab61a044c6d81b8257059977a7f				X								
9c815150ea821166faecf80757d8826a										X		
04dd53033ee41ead468844c613c3baf2		X										
f79b6bad2ad0641e1921aef10262856b		X	X				X			X	X	
d75e7289c86c15b305ac36097bfa0487			X									
24c42d3e77f06f2a8f47f4ebbbfdfaf4		X										

b32309a26951912be7dba376398abc3b												X
c67e9dc27d283f1f89b4ebb4b4670c21		X								X		
a5804cc8972df3ab8b7ccde6b7a9e0f4		X								X	X	
0acf40799ecbd83506d6caeb38514671		X										

JA3 results from Samsung Galaxy A71 with Android 12

JA3	Chrome	Edge	Epassi	Facebook	Foli	Gmail	Hsl	Linkedin	Maps	Resqclub	Wolt	Youtube
ee26b1f1aec16d6098768e2c67388ace									X			
84cd49a759d15947552e61b46fa5f2ec							X					
de4389c26478c9cf885d0b33fd1307cf		X										
6ec2896feff5746955f700c0023f5804						X			X			
74ad8ec6876e2e3366bfd566581ca7e8	X											
9b02ebd3a43b62d825e1ac605b621dc8		X	X	X	X		X			X	X	
f3e8b92336e2ffa7b31fcfb9a4bbb617							X					
e1d8b04eeb8ef3954ec4f49267a783ef					X		X	X				
598872011444709307b861ae817a4b60					X	X	X	X	X			X
cd08e31494f9531f560d64c695473da9	X				X	X	X	X	X		X	X
61e4063abe913214d3e1a979a7666d73				X								
0d69ff451640d67ee8b5122752834766	X											
d75e7289c86c15b305ac36097bfa0487			X									
c67e9dc27d283f1f89b4ebb4b4670c21		X	X							X	X	
893d25297c894da36fbdee5cea98b01c						X			X			
312c0b788e784899a8cc07c960430c3a						X						
24c42d3e77f06f2a8f47f4ebbbfdfaf4		X										
295b8fa5db7cfec6cf325aae9e1c60b6		X										
9c815150ea821166faecf80757d8826a										X		

a5804cc8972df3ab8b7ccde6b7a9e0f4		X									X	
0bb1258a19c600139c138bdd9e28894b				X								
04dd53033ee41ead468844c613c3baf2		X										
f79b6bad2ad0641e1921aef10262856b		X	X				X			X	X	
0acf40799ecbd83506d6caeb38514671		X										
6c0f0a346dcd84cb4b97a0d9382c53fd					X							
00c2fab61a044c6d81b8257059977a7f				X								

JA3 results from Samsung Galaxy S22 with Android 13

JA3	Chrome	Edge	Epassi	Facebook	Foli	Gmail	Hsl	Linkedin	Maps	Resqclub	Wolt	Youtube
6c0f0a346dcd84cb4b97a0d9382c53fd					X							
9b02ebd3a43b62d825e1ac605b621dc8		X	X		X		X			X	X	
9c815150ea821166faecf80757d8826a										X		
d75e7289c86c15b305ac36097bfa0487			X									
cd08e31494f9531f560d64c695473da9	X				X	X	X	X	X			X
c67e9dc27d283f1f89b4ebb4b4670c21										X		
598872011444709307b861ae817a4b60					X							X
893d25297c894da36fbdee5cea98b01c									X			
00c2fab61a044c6d81b8257059977a7f				X								
84cd49a759d15947552e61b46fa5f2ec											X	
de4389c26478c9cf885d0b33fd1307cf		X										
f3e8b92336e2ffa7b31fcfb9a4bbb617		X					X					
f79b6bad2ad0641e1921aef10262856b		X	X				X			X	X	
0bb1258a19c600139c138bdd9e28894b				X								
0acf40799ecbd83506d6caeb38514671		X										
0d69ff451640d67ee8b5122752834766	X											
a5804cc8972df3ab8b7ccde6b7a9e0f4											X	
e1d8b04eeb8ef3954ec4f49267a783ef					X		X	X				
74ad8ec6876e2e3366bfd566581ca7e8												

24c42d3e77f06f2a8f47f4ebbbfdfaf4		X										
312c0b788e784899a8cc07c960430c3a						X						