STAMVS NETWORKS

Google killed JA3 ! Should we be scared ?

2024/07/05 | Eric Leblond

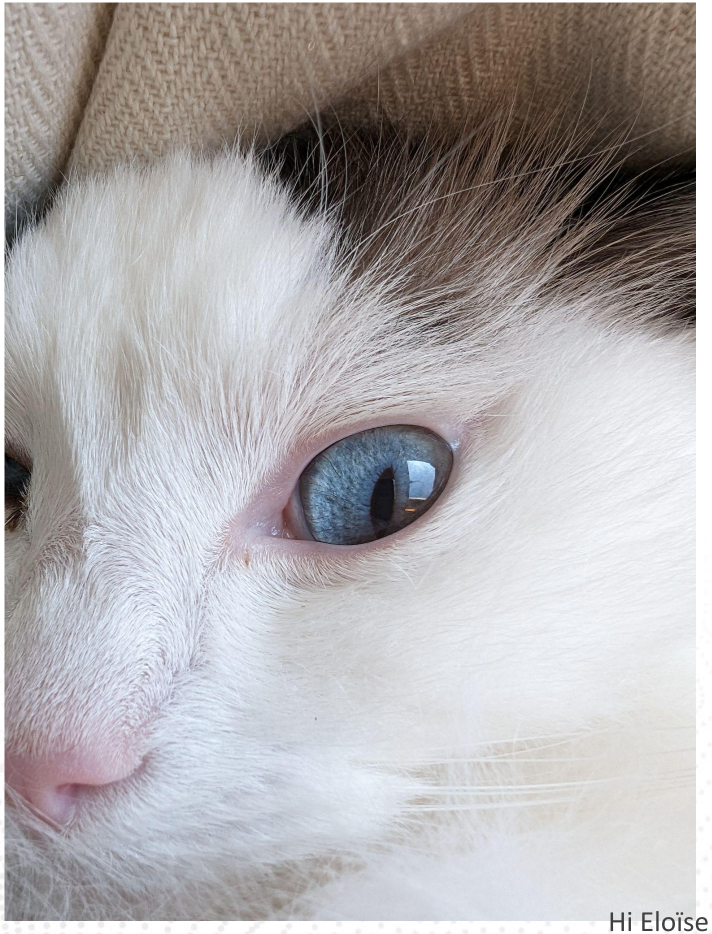
Who am I ?

Eric Leblond

- Co founder & CTO of Stamus Networks
- Member of OISF’s board
- Contributor to Suricata since 2009
- Co-author of “The Security Analyst's Guide to Suricata”

Stamus Networks:

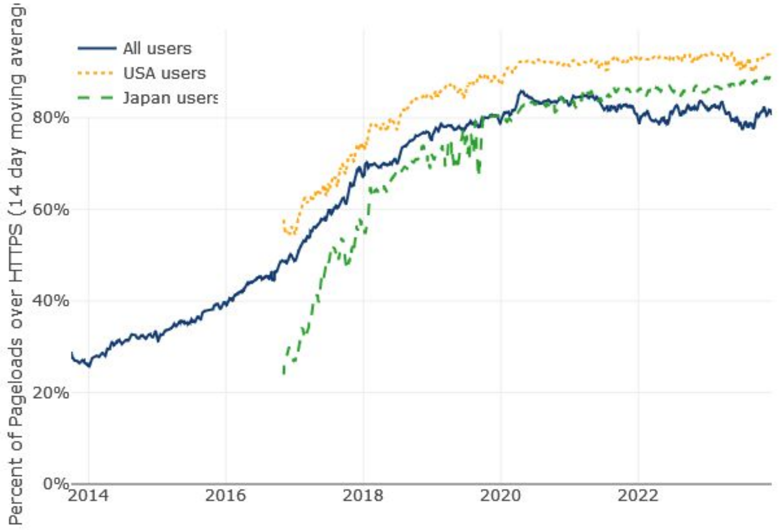
- Editor of a Suricata based NDR solution
- Contributor to Suricata


Hi Eloïse ;-)

Security is about building wall…

Encryption by default has won

- Introduction of SSL in 1995
- Everywhere since 2020
- Thanks Let’s Encrypt !
    - People can change the world
- Privacy is now a thing
    - For the content

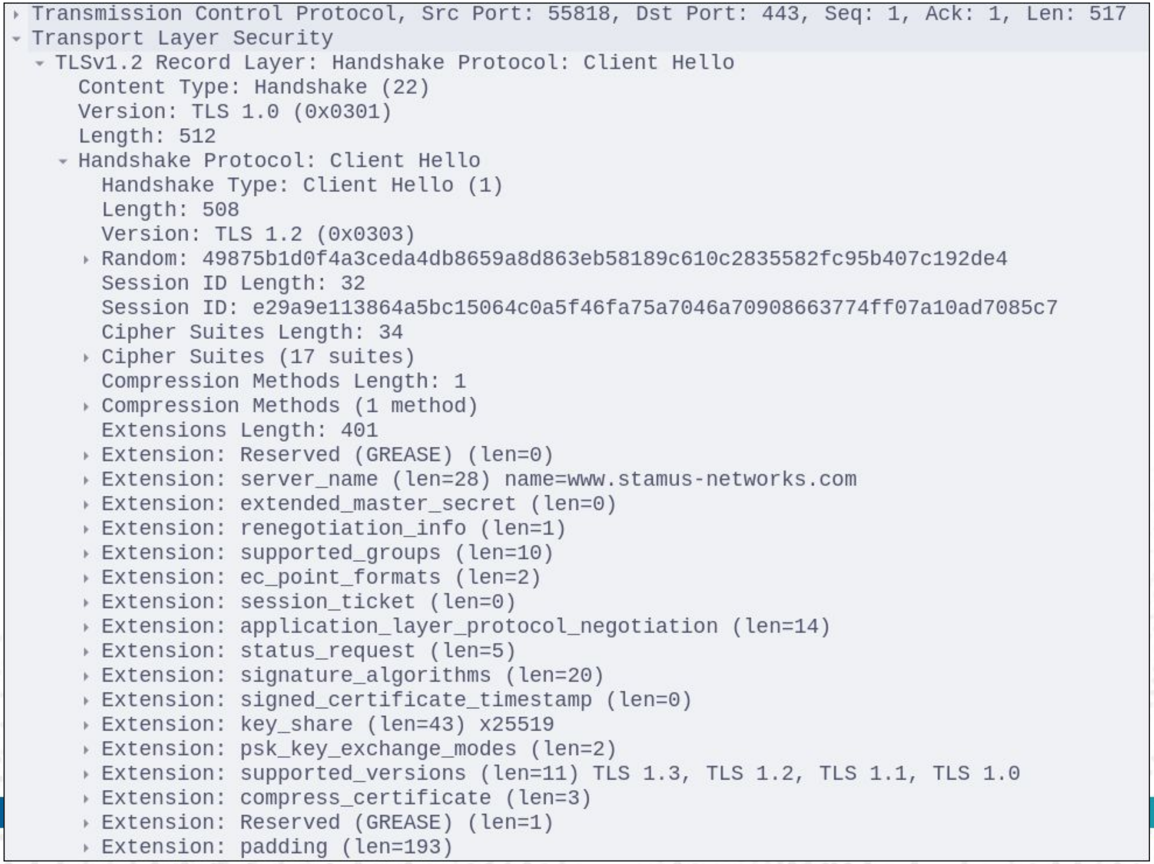


Source:
https://www.eff.org/deeplinks/2023/12/year-review-last-mile-encrypting-web

Privacy 1 - Security 0

- Blindness of all analysis tools relaying on traffic
- What is left is:
  - Decryption devices
  - Analysis of TLS handshake

TLS Handshake Analysis: Client Hello

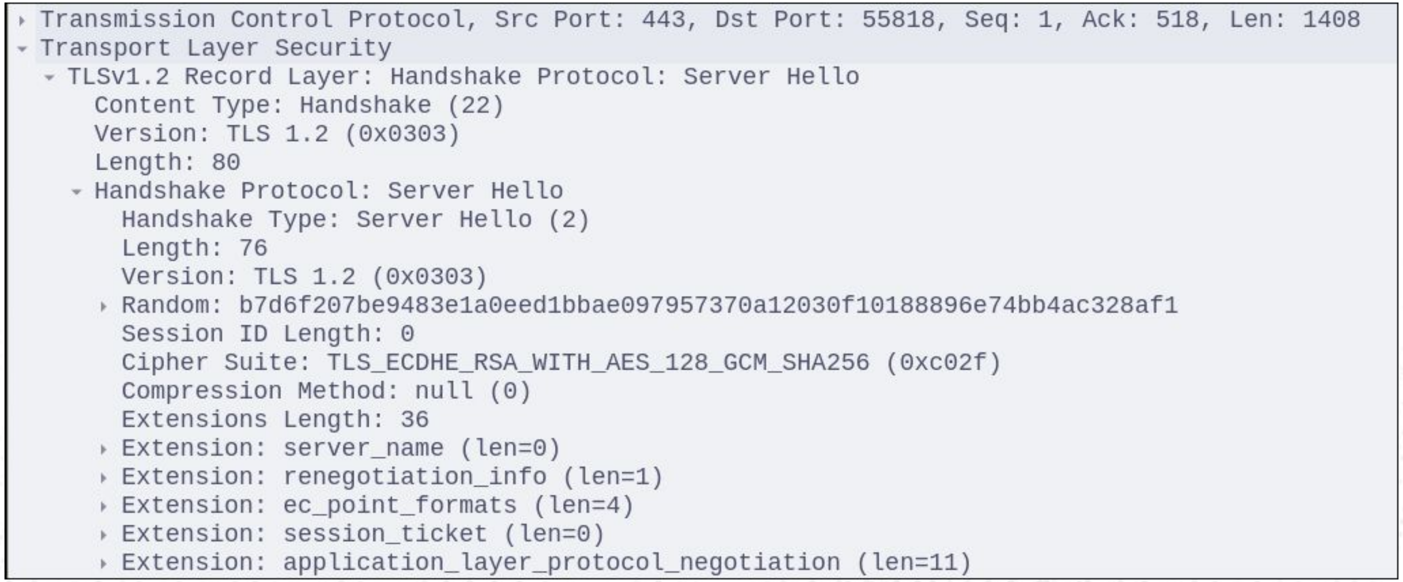


```
▸ Transmission Control Protocol, Src Port: 55818, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 512
    ▾ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 508
        Version: TLS 1.2 (0x0303)
      ▸ Random: 49875b1d0f4a3ceda4db8659a8d863eb58189c610c2835582fc95b407c192de4
        Session ID Length: 32
        Session ID: e29a9e113864a5bc15064c0a5f46fa75a7046a70908663774ff07a10ad7085c7
        Cipher Suites Length: 34
      ▸ Cipher Suites (17 suites)
        Compression Methods Length: 1
      ▸ Compression Methods (1 method)
        Extensions Length: 401
      ▸ Extension: Reserved (GREASE) (len=0)
      ▸ Extension: server_name (len=28) name=www.stamus-networks.com
      ▸ Extension: extended_master_secret (len=0)
      ▸ Extension: renegotiation_info (len=1)
      ▸ Extension: supported_groups (len=10)
      ▸ Extension: ec_point_formats (len=2)
      ▸ Extension: session_ticket (len=0)
      ▸ Extension: application_layer_protocol_negotiation (len=14)
      ▸ Extension: status_request (len=5)
      ▸ Extension: signature_algorithms (len=20)
      ▸ Extension: signed_certificate_timestamp (len=0)
      ▸ Extension: key_share (len=43) x25519
      ▸ Extension: psk_key_exchange_modes (len=2)
      ▸ Extension: supported_versions (len=11) TLS 1.3, TLS 1.2, TLS 1.1, TLS 1.0
      ▸ Extension: compress_certificate (len=3)
      ▸ Extension: Reserved (GREASE) (len=1)
      ▸ Extension: padding (len=193)
```

Thanks Wireshark

STAMUS NETWORKS

TLS Handshake Analysis: Server Hello



```
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 55818, Seq: 1, Ack: 518, Len: 1408
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 80
    ▾ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 76
        Version: TLS 1.2 (0x0303)
      ▸ Random: b7d6f207be9483e1a0eed1bbae097957370a12030f10188896e74bb4ac328af1
        Session ID Length: 0
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        Compression Method: null (0)
        Extensions Length: 36
      ▸ Extension: server_name (len=0)
      ▸ Extension: renegotiation_info (len=1)
      ▸ Extension: ec_point_formats (len=4)
      ▸ Extension: session_ticket (len=0)
      ▸ Extension: application_layer_protocol_negotiation (len=11)
```

How to get information about client ?

Building JA3

- Client identification algorithm
  - Developed by John Althouse, Jeff Atkinson, Josh Atkins
  - Created around June 2017
  - Use the first message sent by the client to build a fingerprint
  - A simple concatenation of the fields
- Building Algorithm: fields separated via comma, array by dash

SSLVersion,Cipher,SSLExtension,EllipticCurve,EllipticCurvePointFormat

- Example:

769,47-53-5-10-49161-49162-49171-49172-50-56-19-4,0-10-11,23-24-25,0

JA3 was successful for a few years

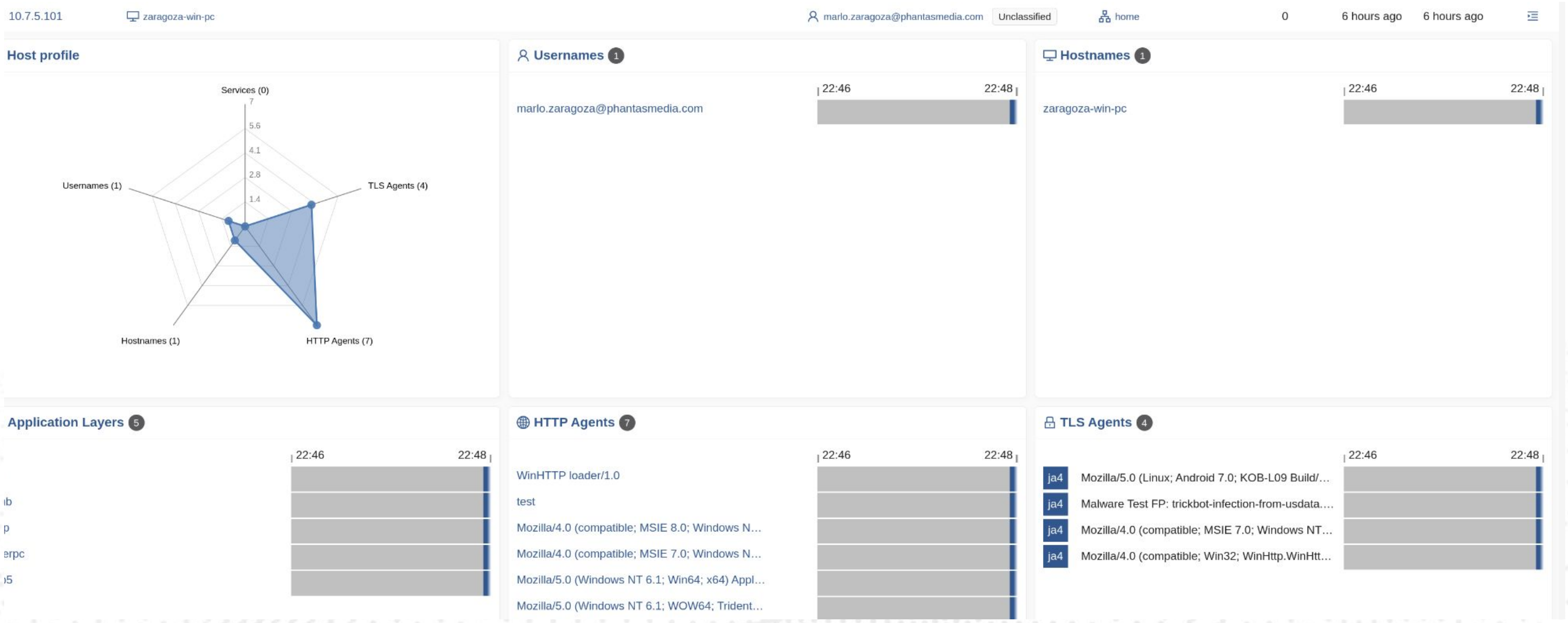
- Identification of implementation:
  - Browser with version
  - Some malwares
- JA3 to agent databases
- De facto standard for fingerprinting
  - Without decryption
  - Can be used anywhere and early
    - Reverse proxy
    - Firewall

- Used in
  - Suricata
  - Wireshark
  - Arkime
  - Splunk
  - AWS Firewall
  - Azure Firewall
  - Far more…
- https://github.com/salesforce/ja3

And then something strange happened (1/3)

- Stamus Security Platform has a feature named Host Insight
  - Store information about IP on the network
  - Using uniquely information coming from Suricata
  - Track characteristics seen on IP like
    - Username
    - Hostname
    - HTTP user agent
    - TLS agent
- TLS agent:
  - JA3 correspond to an implementation
  - JA3 can be mapped to a agent name
    - Using existing mapping database

And then something strange happened (2/3)



10.7.5.101 zaragoza-win-pc marlo.zaragoza@phantasmedia.com Unclassified home 0 6 hours ago 6 hours ago

Host profile

Usernames 1

	22:46	22:48
marlo.zaragoza@phantasmedia.com		

Hostnames 1

	22:46	22:48
zaragoza-win-pc		

Application Layers 5

	22:46	22:48
ıb		
p		
erpc		
ı5		

HTTP Agents 7

	22:46	22:48
WinHTTP loader/1.0		
test		
Mozilla/4.0 (compatible; MSIE 8.0; Windows N…		
Mozilla/4.0 (compatible; MSIE 7.0; Windows N…		
Mozilla/5.0 (Windows NT 6.1; Win64; x64) Appl…		
Mozilla/5.0 (Windows NT 6.1; WOW64; Trident…		

TLS Agents 4

		22:46	22:48
ja4	Mozilla/5.0 (Linux; Android 7.0; KOB-L09 Build/…		
ja4	Malware Test FP: trickbot-infection-from-usdata.…		
ja4	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT…		
ja4	Mozilla/4.0 (compatible; Win32; WinHttp.WinHtt…		

And then something strange happened (3/3)

- Massive overflowing of TLS agent table
- In production
  - one host was triggering 1000 JA3 per minutes
  - Far more than sum of the other 10000 hosts on this network
  - Strong impact on performance
- It was not making sense
  - Code was unchanged
  - This did suddenly appear
- What did change
  - Could just be on the client itself

And the responsible is Google

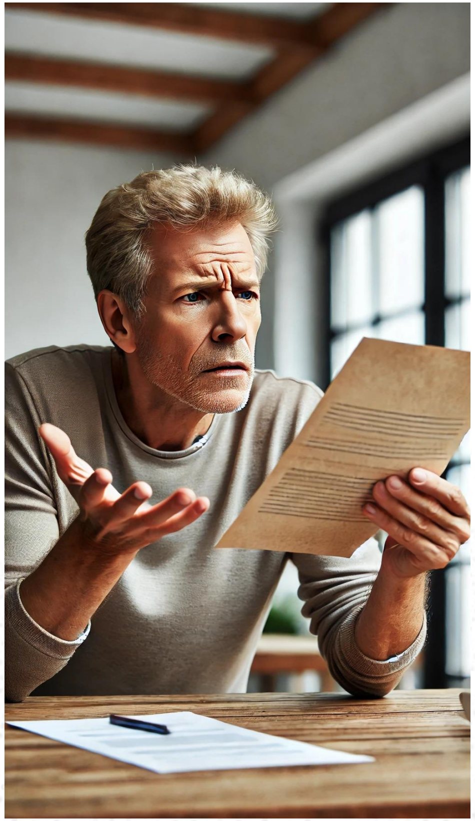
- Chrome feature: https://chromestatus.com/feature/5124606246518784

Don’t be evil

- Extensions are always send in same order
  - But server should not act based on this
  - RFC stipulate that only the last one should be in fixed place
  - Let’s randomized the extension list when sending them
- I’ve read it multiple times
  - Still make no sense
  - Any hint welcome
- Reminder, this is JA3:

  SSLVersion,Cipher,SSLExtension list,

  EllipticCurve,EllipticCurvePointFormat
- This completely break JA3 fingerprint



Impact of the change

- One implementation has millions of ja3 fingerprints
    - Can’t identify an implementation anymore
- Impact on some usages
    - Keeping a list of TLS agents on an IP address
        - Ending up with on agent per new TLS connection
    - Per ja3 policy is dead
        - Firewall, reverse proxy
- A way out for detection of malwares
    - Just add a function to randomize the list
    - Don’t get detected

STAMUS NETWORKS

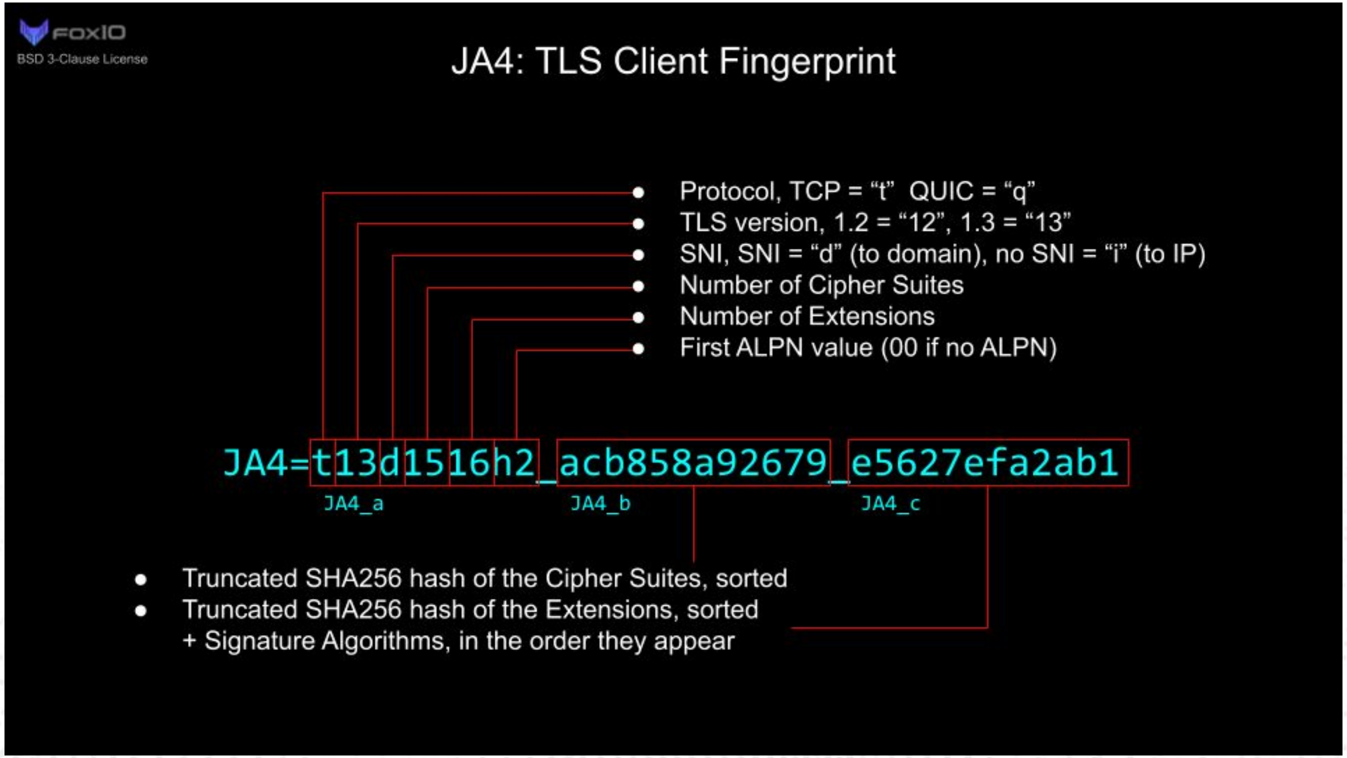
Why this failure ?

- JA3 use implementation behavior
- RFC should be the minimum degree of freedom
  - Because real life is even worst
  - Server must work with client violating RFC
- Design should at least be resistant to variation in RFC scope

JA4 to the rescue (or not)

- Evolution of JA3 & more
  - Developed by John Althouse
  - Under FoxIO LLC umbrella
- A set of fingerprinting techniques
  - TLS JA4, replacement of JA3
  - JA4HTTP, JA4Latency, …
- https://github.com/FoxIO-LLC/ja4

JA4: TLS fingerprint



JA4: TLS Client Fingerprint

- Protocol, TCP = “t” QUIC = “q”
- TLS version, 1.2 = “12”, 1.3 = “13”
- SNI, SNI = “d” (to domain), no SNI = “i” (to IP)
- Number of Cipher Suites
- Number of Extensions
- First ALPN value (00 if no ALPN)

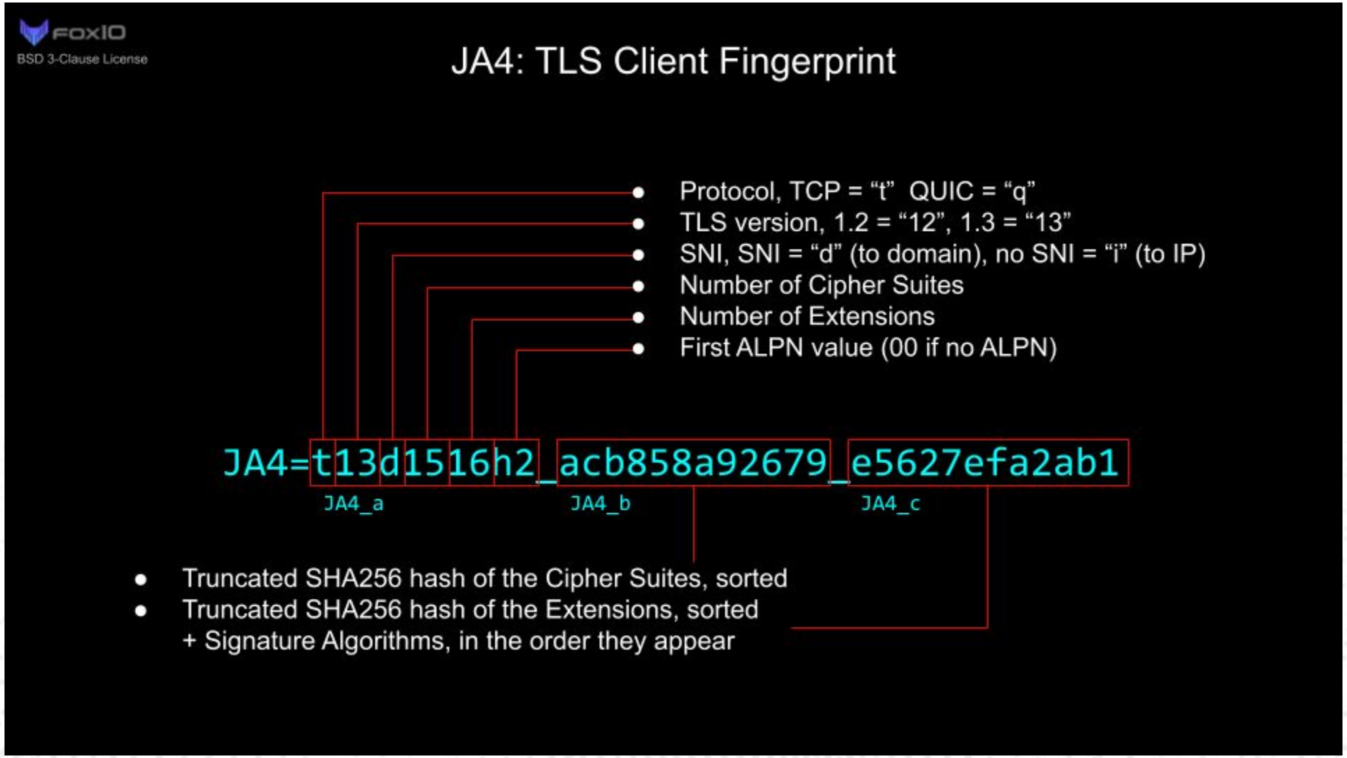
```
JA4=t13d1516h2_acb858a92679_e5627efa2ab1
    JA4_a       JA4_b        JA4_c
```

- Truncated SHA256 hash of the Cipher Suites, sorted
- Truncated SHA256 hash of the Extensions, sorted
  + Signature Algorithms, in the order they appear

License

- JA4 (TLS) is BSD 3-Clause
- JA4S & the others: FoxIO License 1.1.
  - not permissive for monetization
- License on an algorithm ?

A look at ja4



```
JA4=t13d1516h2_acb858a92679_e5627efa2ab1
```

- Protocol, TCP = “t” QUIC = “q”
- TLS version, 1.2 = “12”, 1.3 = “13”
- SNI, SNI = “d” (to domain), no SNI = “i” (to IP)
- Number of Cipher Suites
- Number of Extensions
- First ALPN value (00 if no ALPN)

- Truncated SHA256 hash of the Cipher Suites, sorted
- Truncated SHA256 hash of the Extensions, sorted
  + Signature Algorithms, in the order they appear

Take aways

- Lists are sorted so this fix the “problem” of randomization
  - This lower the separation capability of the fingerprint
- ALPN: Application-Layer Protocol Negotiation
  - Interesting information about the protocol
    - Client proposes protocol in TLS handshake
    - Usually: h2, http/1.1
    - Server answer negotiated protocol
- Information on connection
  - SNI
  - quic/tls

Fingerprinting implementation ?

- One implementation can do
  - TLS and QUIC
  - Potentially SNI or not
  - Propose different alpn
- One implementation has multiple ja4
  - From 2 to 8 on just connection dependant information

Conclusion

- JA3 is now mostly useless
  - Detection can now be easily evaded by updating implementation
  - Mapping to agent can not be done
- JA4 is a nice replacement
  - Adoption seems to take

Thank You!

Contact:

Web: https://www.stamus-networks.com/

Mail: el@stamus-networks.com

STAMVS NETWORKS