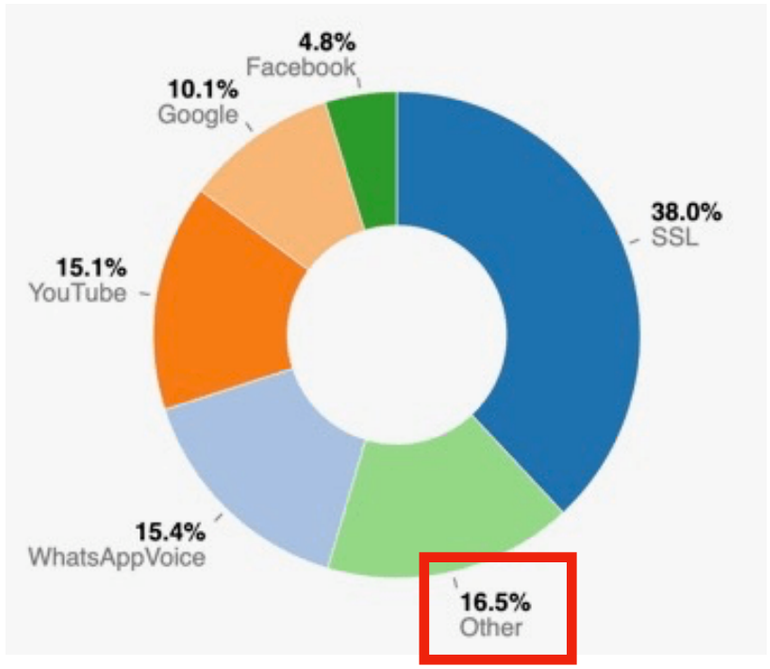
Encrypted Traffic Analysis: A Primer

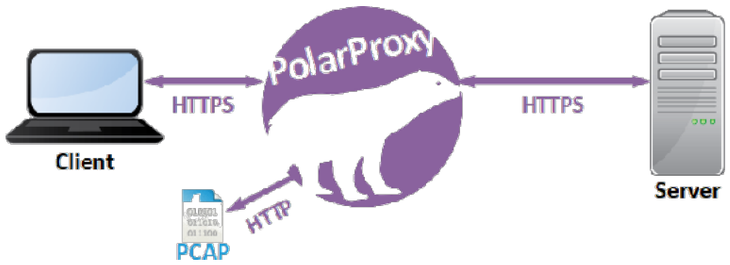
Luca Deri <deri@ntop.org>
@lucaderi

Trends in Network Traffic

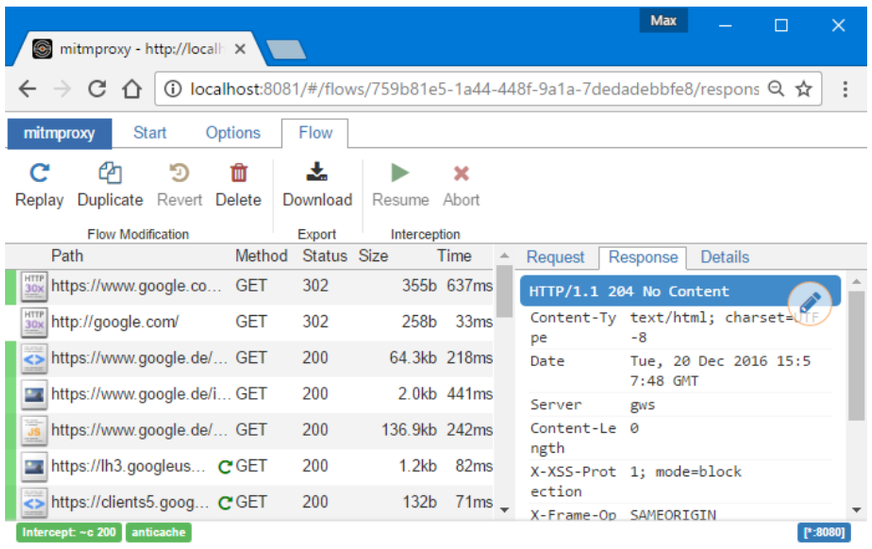
- Zero Trust Architectures require permanent traffic inspection to guarantee operations and prevent malicious activities to take place.
- Internet traffic is mostly encrypted thus limiting visibility and introspection (malware are migrating to TLS).
- Security based on clear-text payload inspection doesn't look a great idea…



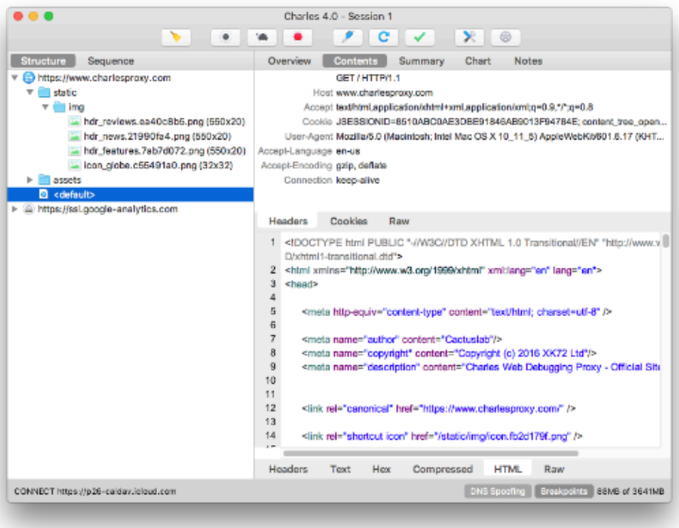
MITM (Man in The Middle)



https://www.netresec.com/?page=PolarProxy



https://mitmproxy.org



https://www.charlesproxy.com

MITM? No Thanks

- MITS is the wrong answer (decryption) to a valid question (visibility). Beside legal issue or GDPR:
  - It's not possible to decode all TLS traffic unless I inject to all my clients a fake certificate authority.
  - Not all encrypted traffic is TLS (e.g. QUIC, VPN, SSH, BitTorrent…) so it's a lost battle.
  - Ethical reasons: if traffic is encrypted there should be a reason.
  - Decryption+Re-Encryption will be detected.

How MITM Works [1/2]

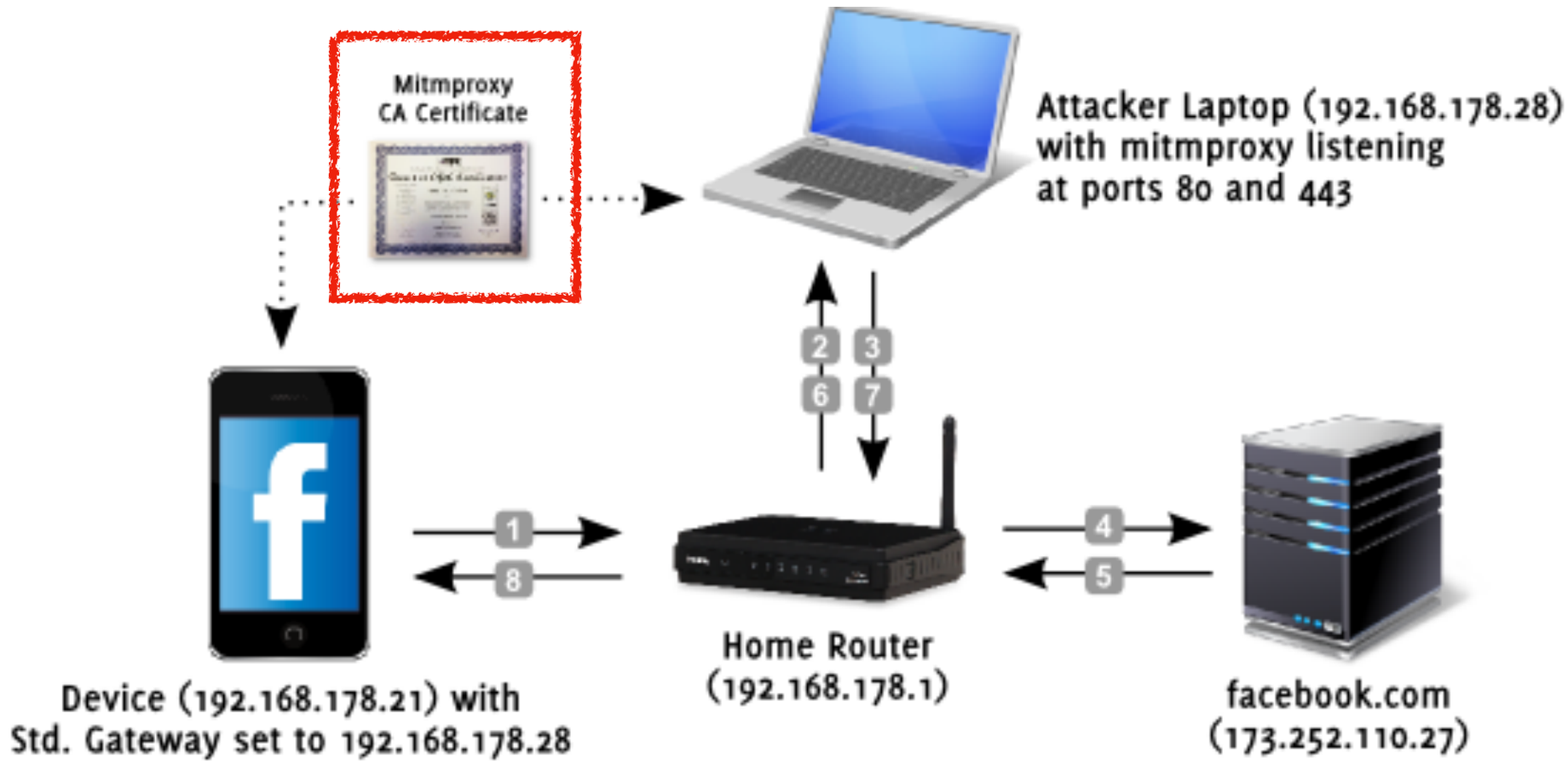
Charles does this by becoming a man-in-the-middle. Instead of your browser seeing the server's certificate, Charles dynamically generates a certificate for the server and signs it with its own root certificate (the Charles CA Certificate). Charles receives the server's certificate, while your browser receives Charles's certificate. Therefore you will see a security warning, indicating that the root authority is not trusted. If you add the Charles CA Certificate to your trusted certificates you will no longer see any warnings – see below for how to do this.

Charles still communicates via SSL to the web server. The communication is SSL (encrypted) from web browser to Charles and also SSL (encrypted) from Charles to the web server.

This functionality is essential for debugging secure (SSL) web applications.

https://www.charlesproxy.com/documentation/proxying/ssl-proxying/

How MITM Works [2/2]



https://medium.com/testvagrant/intercept-ios-android-network-calls-using-mitmproxy-4d3c94831f62

HTTP Public Key Pinning

HTTP Public Key Pinning (HPKP) [RFC 7469] is a security feature that tells a web client to associate a specific cryptographic public key with a certain web server to decrease the risk of MITM attacks with forged certificates.

```
Public-Key-Pins:
  pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM=";
  pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
  pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
  max-age=2592000; includeSubDomains; report-uri="http://example.com/pkp-report"
```

For 365 Days (2592000 / 86400)
do not accept new certificates: too long!

https://developer.mozilla.org/en-US/docs/Web/HTTP/Public_Key_Pinning

Is HPKP a Good Idea?

Yes. However: "If your expectations don't match reality your users suffer from not being able to access your app or website. Smashing Magazine learnt about this the hard way in late 2016 when they blocked users access for up to a year because of a mismatch between the pins and the certificates. On mobile fixing an invalid pin means pushing out a new version of an app which can still take a while to reach every user."

https://medium.com/babylon-engineering/android-security-certificate-transparency-601c18157c44

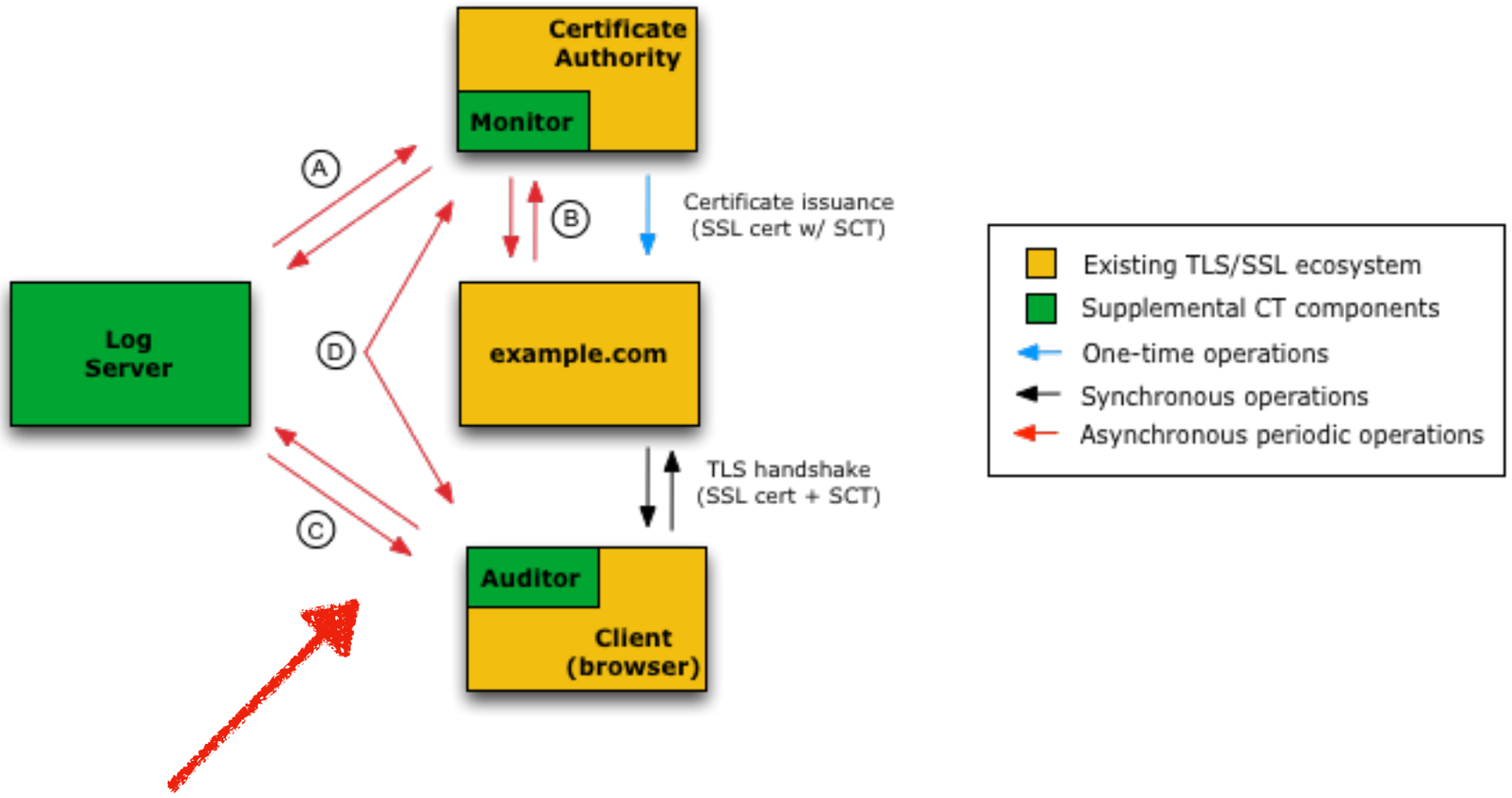
What is Certificate Transparency? [1/3]

Ⓐ Monitors watch logs for suspicious certs and verify that all logged certs are visible.

Ⓑ Certificate owners query monitors to verify that nobody has logged illegitimate certs for their domain.

Ⓒ Auditors verify that logs are behaving properly; they can also verify that a particular cert has been logged.

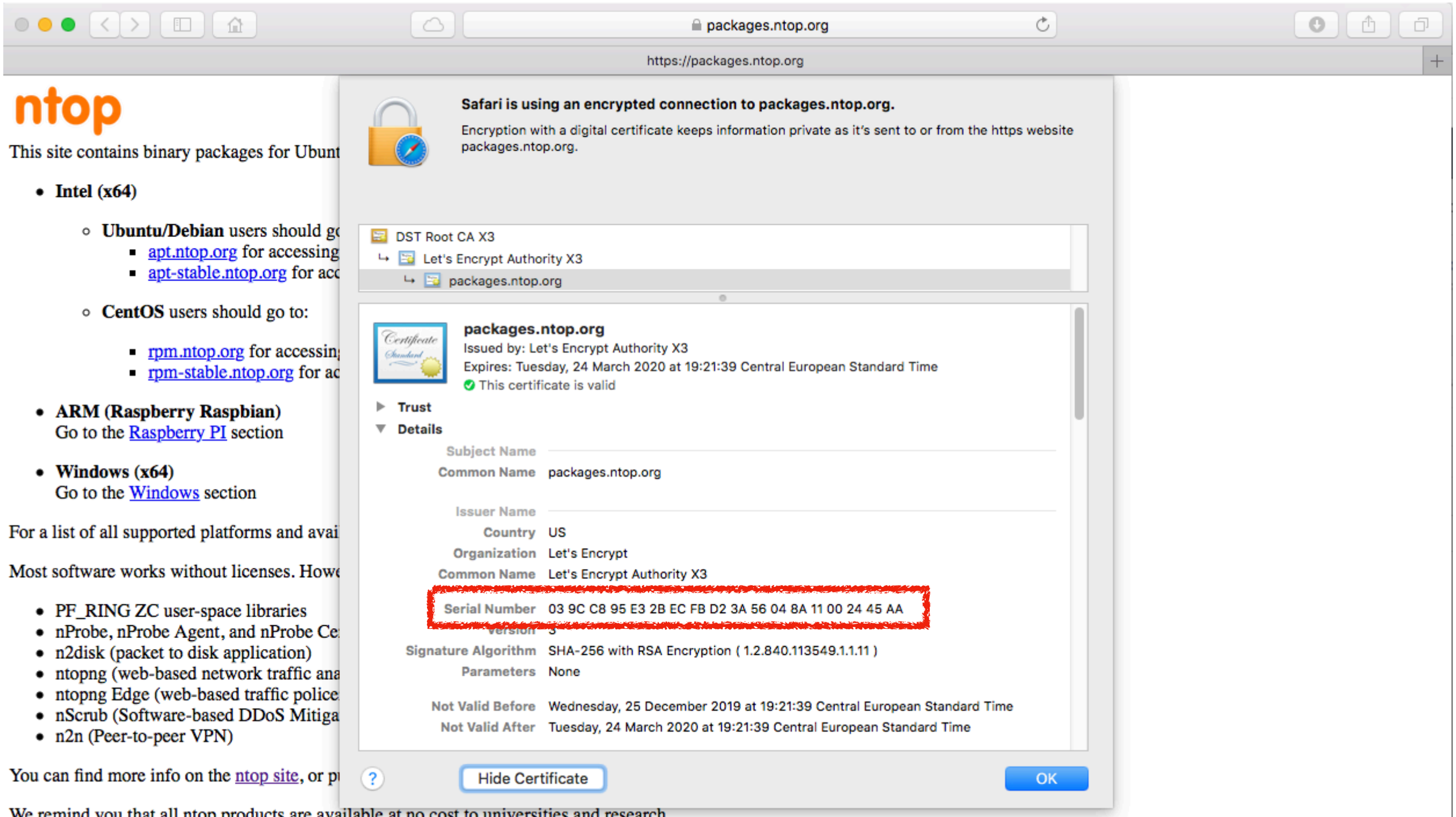
Ⓓ Monitors and auditors exchange information about logs to help detect forked or branched logs.



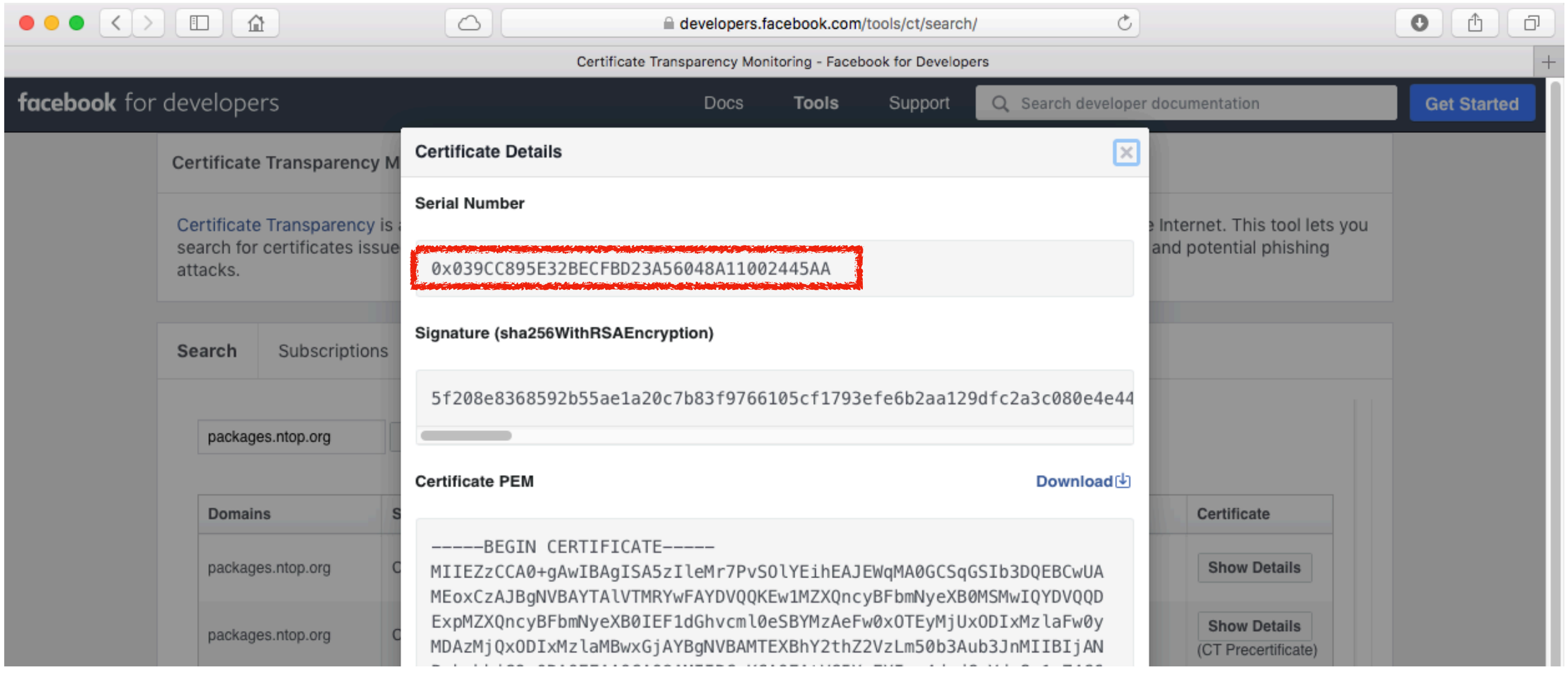
Certificate Transparency (**CT**) is an Internet security standard and open source framework for monitoring and auditing digital certificates.[1] The standard creates a system of public logs that seek to eventually record all certificates issued by publicly trusted certificate authorities, allowing efficient identification of mistakenly or maliciously issued certificates

https://www.certificate-transparency.org/how-ct-works

What is Certificate Transparency? [2/3]



What is Certificate Transparency? [3/3]



Motivation

These first few slides should be enough to motivate:

- Encrypted traffic analysis (without decryption), from the network standpoint, in order to detect issues invisible to users.
- Network behaviour analysis: detect if network traffic behaves as we expect or if something has changed.

```
0000  00 50 56 c0 00 08 00 0c  29 a5 45 e0 08 00 45 00   ·PV····· )·E···E·
0010  03 04 54 d4 40 00 40 06  ae 54 ac 10 ee a8 ac 10   ··T·@·@· ·T······
0020  ee 01 00 16 e4 1b 42 a2  79 f4 37 5e 83 f0 80 18   ······B· y·7^····
0030  00 93 a6 0f 00 00 01 01  08 0a 00 13 23 13 1c 95   ········ ····#···
0040  b0 03 00 00 02 bc 09 21  00 00 01 15 00 00 00 07   ·······! ········
0050  73 73 68 2d 72 73 61 00  00 00 01 23 00 00 01 01   ssh-rsa· ···#····
0060  00 bf a4 02 3b 0d e2 c5  11 89 44 45 45 31 fb 97   ····;··· ··DEE1··
0070  62 9f da ff 2b 37 03 7f  4c 9d fd 56 44 0c 7f 30   b···+7·· L··VD··0
0080  2b 6e ff ca d7 61 cf 01  d4 1a 8f 98 3a 38 91 7a   +n···a·· ····:8·z
0090  d9 ee 1f 4f 87 dd 9d 05  db c9 7a 53 2a c8 33 da   ···O···· ··zS*·3·
00a0  bb 00 88 f0 97 06 47 15  a5 d2 b5 b9 ef 82 55 bc   ······G· ······U·
00b0  49 ce 69 09 24 1b 45 84  89 72 d2 2a 08 2c a9 8e   I·i·$·E· ·r·*·,··
00c0  24 93 47 0a ff 50 08 5c  c2 d4 03 89 f3 36 0e f1   $·G··P·\ ·····6··
00d0  22 41 e1 e2 6e 9c 02 19  b8 15 c0 02 4c 8a b2 b1   "A··n··· ····L···
```

On a nutshell:
Is this encrypted traffic good or bad ?

Monitoring Requirements

- Network administrators need to enforce network policies hence:
  - Limit the bandwidth of specific protocols (e.g. BitTorrent).
  - Block malicious communications that might travel over encrypted traffic channels.
  - Prioritise specific traffic protocols (e.g. WhatsApp/ Skype) or cloud protocols.
  - As previously stated, traffic decryption is not an option for many reasons, in particular as it is useless in many reason while violating privacy.

What Do We Want to Accomplish?

- Fingerprint network traffic to detect if both the protocol (e.g. the certificate) has changed or its behaviour.
- Prevent specific traffic flows (e.g. unsafe TLS communications) to happen on our network.
- Provide metrics for measuring the nature of specific communications (e.g. HTTPS) while not being able to inspect the content.
- Identify malware in network communications.

Monitoring Strategies

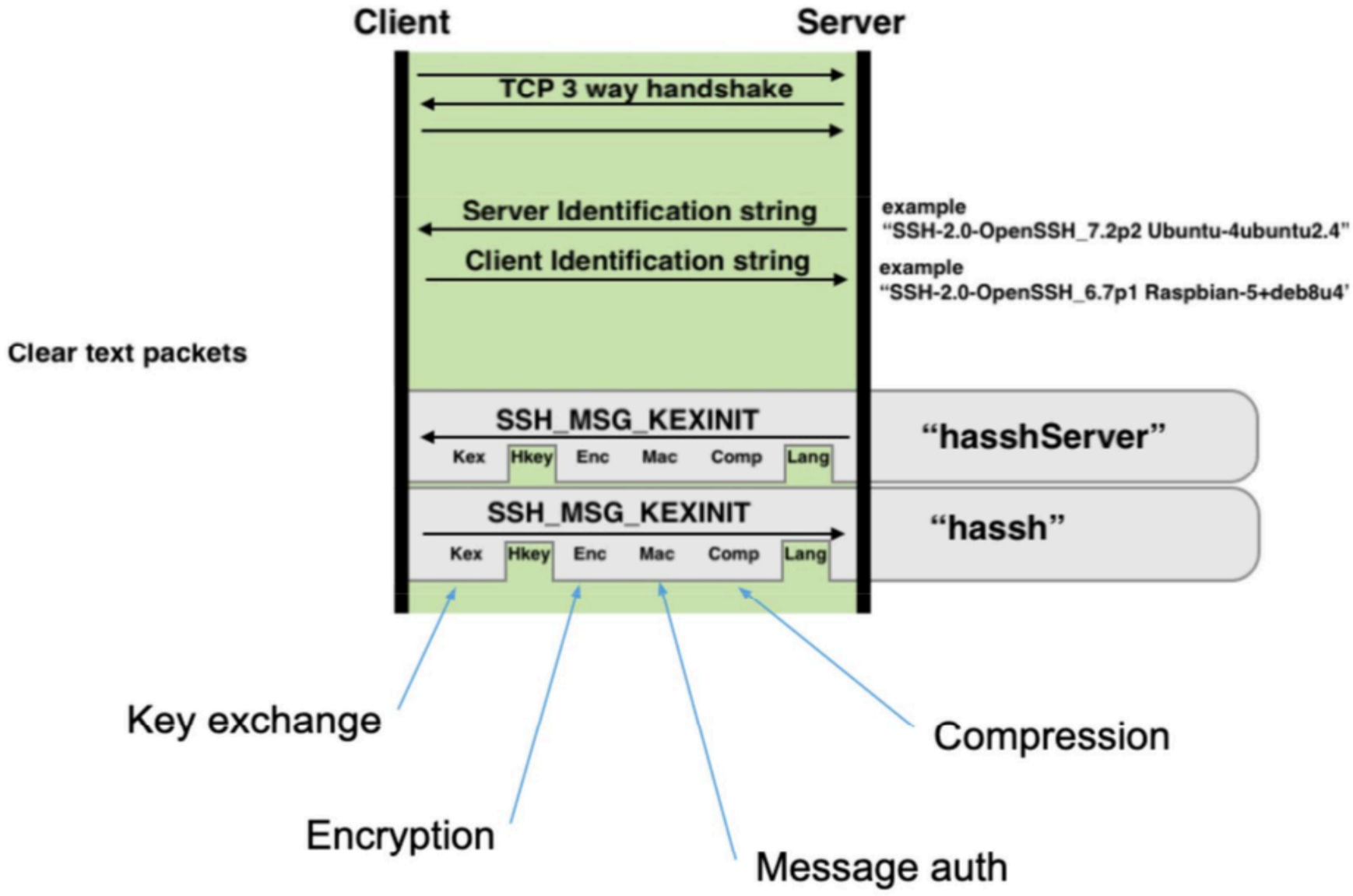
- Analyse protocol messages to detect if we can report changes in protocol communication "aesthetics" (i.e. protocol messages).
- Develop algorithms for monitoring traffic overtime and spot changes in behaviour that might indicate changes in the remote peers configuration or a malware infection.

SSH (Secure Shell) Protocol Monitoring

SSH Fingerprinting: HASSH

- HASSH is a network fingerprinting standard created by Salesforce which can be used to identify specific client and server SSH implementations.
- Fingerprints can be easily stored, searched and shared in the form of an MD5 fingerprint.
- They can be computed for both client and server and are useful to detect changes in SSH client software/configuration.

HASSH Client [1/2]



HASSH Client [2/2]

Function	Algorithms seen in SSH_MSG_KEXINIT packets
Key Exchange methods	`curve25519-sha256@libssh.org,diffie-hellman-group-exchange-sha256,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group-exchange-sha1,diffie-hellman-group1-sha1,diffie-hellman-group14-sha1,diffie-hellman-group14-sha256,diffie-hellman-group15-sha512,diffie-hellman-group16-sha512,diffie-hellman-group17-sha512,diffie-hellman-group18-sha512,diffie-hellman-group14-sha256@ssh.com,diffie-hellman-group15-sha256,diffie-hellman-group15-sha256@ssh.com,diffie-hellman-group15-sha384@ssh.com,diffie-hellman-group16-sha256,diffie-hellman-group16-sha384@ssh.com,diffie-hellman-group16-sha512@ssh.com,diffie-hellman-group18-sha512@ssh.com`
Encryption	`aes128-cbc,aes128-ctr,aes192-cbc,aes192-ctr,aes256-cbc,aes256-ctr,blowfish-cbc,blowfish-ctr,cast128-cbc,cast128-ctr,idea-cbc,idea-ctr,serpent128-cbc,serpent128-ctr,serpent192-cbc,serpent192-ctr,serpent256-cbc,serpent256-ctr,3des-cbc,3des-ctr,twofish128-cbc,twofish128-ctr,twofish192-cbc,twofish192-ctr,twofish256-cbc,twofish256-ctr,twofish-cbc,arcfour,arcfour128,arcfour256`
Message Authentication	`hmac-sha1,hmac-sha1-96,hmac-md5,hmac-md5-96,hmac-sha2-256,hmac-sha2-512`
Compression	`zlib@openssh.com,zlib,none`

HASSH Client Examples

`de30354b88bae4c2810426614e1b6976`	Powershell Renci.SshNet.SshClient.0.0.1 (used by Empire exploit modules)
`fafc45381bfde997b6305c4e1600f1bf`	Ruby/Net::SSH_5.0.2 x86_64-linux (used by Metasploit exploit modules)
`b5752e36ba6c5979a575e43178908adf`	Python Paramiko_2.4.1 (used by Metasploit exploit modules)
`16f898dd8ed8279e1055350b4e20666c`	Dropbear_2012.55 (used in IOT embedded systems)
`8a8ae540028bf433cd68356c1b9e8d5b`	CyberDuck Version 6.7.1 (28683)
`06046964c022c6407d15a27b12a6a4fb`	OpenSSH_7.7p1 Ubuntu-4

HASSH Server

Function	Algorithms seen in SSH_MSG_KEXINIT packets
Key Exchange methods	`diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1`
Encryption	`aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se`
Message Authentication	`hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96`
Compression	`none,zlib@openssh.com`

HASSH Server Examples

`c1c596caaeb93c566b8ecf3cae9b5a9e`	SSH-2.0-dropbear_2016.74
`d93f46d063c4382b6232a4d77db532b2`	SSH-2.0-dropbear_2016.72
`2dd9a9b3dbebfaeec8b8aabd689e75d2`	SSH-2.0-AWSCodeCommit
`696e7f84ac571fdf8fa5073e64ee2dc8`	SSH-2.0-FTP

What Problems does HASSH Addresses ? [1/2]

- HASSH adds contextual information to packet header information.
- The HASSH client is used to fingerprint the client, and thus:
  - Allow blocking clients outside of the "allowed set”.
  - Detect exfiltration if data when using SSH clients with multiple distinct hashes.
  - NAT won’t shield different SSH clients as they can now be detected with this technique.
  - Identify specific client versions.

What Problems does HASSH Addresses ? [2/2]

- The HASSH server can be used to detect if the server configuration is insecure or different from the past.
- In IoT or datacenter where configurations are static (or at least under strict control) administrators, fingerprint should be predictable.
- Same as HASSH client it can be used to block insecure servers, or detect unexpected changes in server configuration.

HASSH Server Database

Search or jump to... / Pull requests Issues Marketplace Explore

0x4D31 / hassh-utils

Watch 3 | Star 27 | Fork 7

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki | Security | Insights

Branch: master ▾ hassh-utils / hasshdb

Find file | Copy path

droberson add hassh lookup feature 98cc33e on Oct 1, 2019

1 contributor

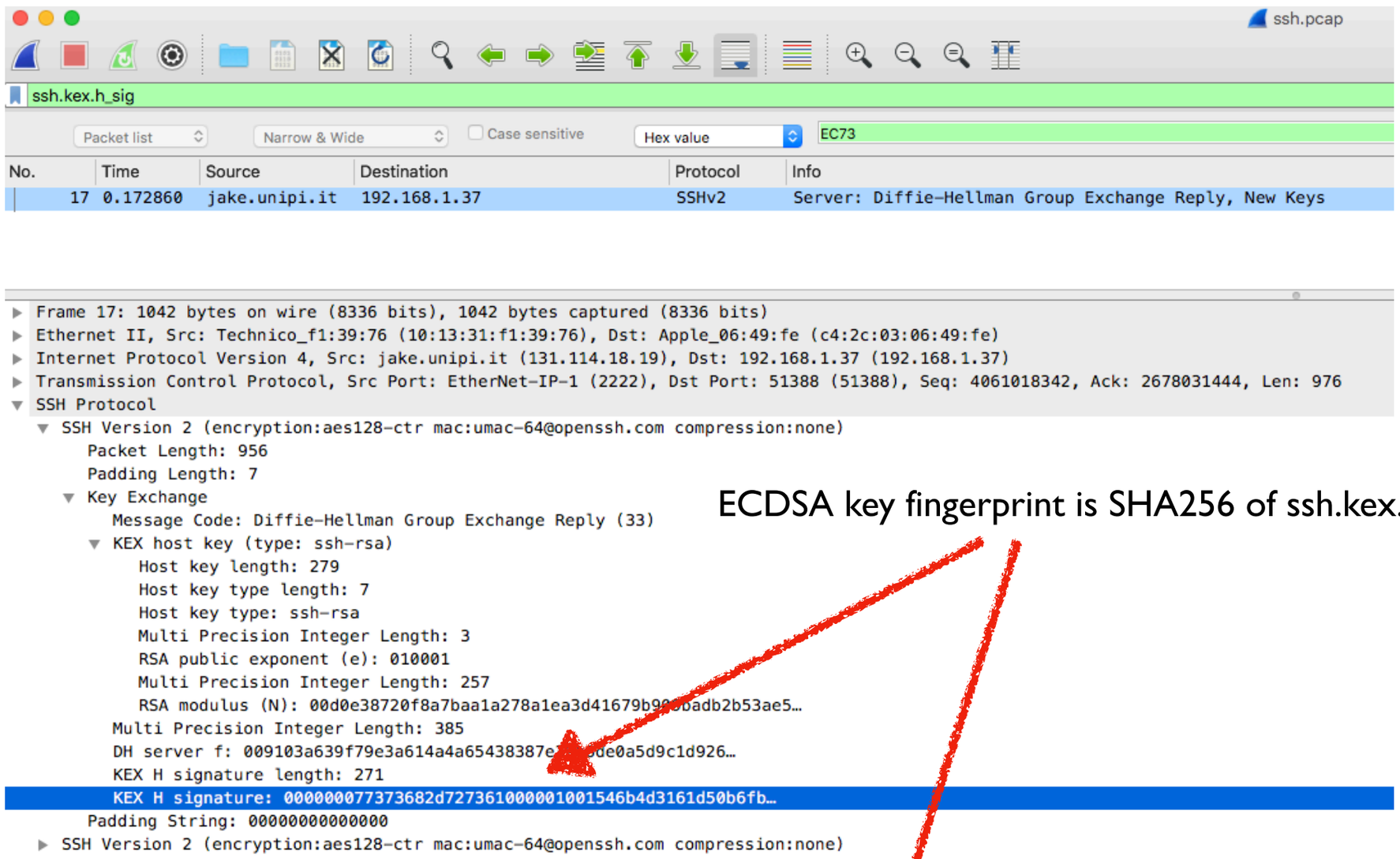
1474 lines (1474 sloc) | 156 KB

Raw | Blame | History

```
1   00885509c9f043998c3d66fb9add68b3 SSH-2.0-OpenSSH_7.4 (100%)
2   009f6d712179bd0c25fc41248dec0f39 SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8 (76%) || SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.6 (23%)
3   00d352967f27037847ef46466c07c06b SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u7 (85%) || SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u6 (14%)
4   00e3c6a18a2236eebb2f0b00df6c9397 SSH-2.0-OpenSSH_7.9 (100%)
5   010d61593035422f8a1a5db0f14943ea SSH-2.0-OpenSSH_7.2p2-hpn14v10 (57%) || SSH-2.0-OpenSSH_7.2 (35%) || SSH-2.0-OpenSSH_7.2p2 Ubu
6   0126b201f99decf00268f18f50720029 SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4 (100%)
7   0134c54c020d31261e5c4d3c8678f79e SSH-2.0-CerberusFTPServer_10.0 (40%) || SSH-2.0-CerberusFTPServer_9.0 (31%) || SSH-2.0-Cerberu
8   0155ef1eb685120d32592340369ec87e SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8 (88%) || SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.6 (7%)
9   01a18ff46c29417870cc2a675226f582 SSH-2.0-JSCAPE (100%)
10  01afc3399d31a69e3577fa696e9a8583 SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3 (100%)
11  01b495a1a6f604e2873a75762bc85714 SSH-2.0-OpenSSH_7.4 (100%)
```

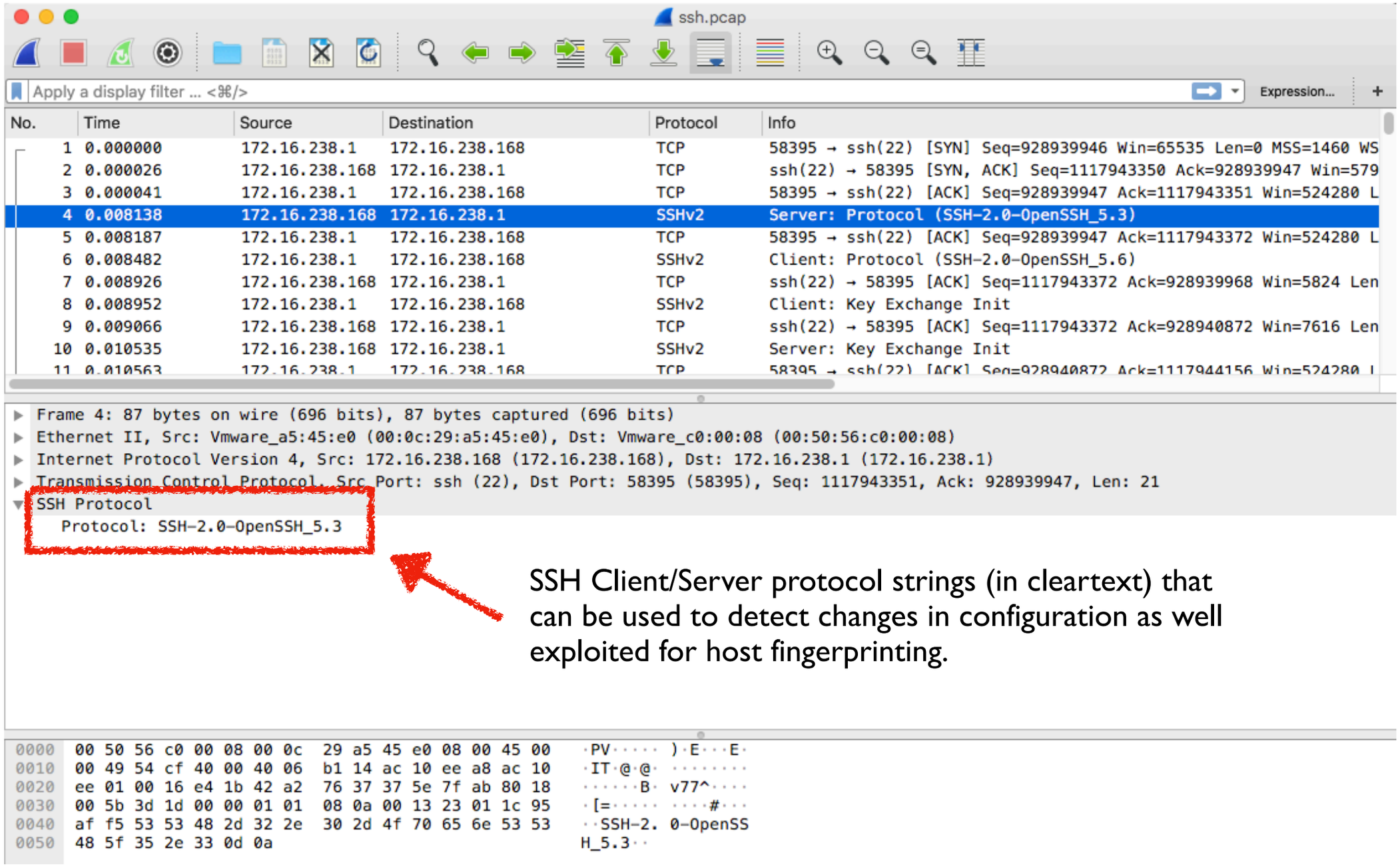
https://github.com/0x4D31/hassh-utils/blob/master/hasshdb

Other SSH Contextual Info [1/2]




```
$ ssh 210.172.195.202
The authenticity of host '210.172.195.202 (210.172.195.202)' can't be established.
RSA key fingerprint is SHA256:oM1NOBCQLu1paUX3MY8lqgicbMsHEofO4F6XsHQVNMU.
Are you sure you want to continue connecting (yes/no)?
```

Other SSH Contextual Info [2/2]



SSH Client/Server protocol strings (in cleartext) that can be used to detect changes in configuration as well exploited for host fingerprinting.

TLS (Transport Layer Security) Protocol Monitoring

Welcome to TLS [1/2]

TLS is a cryptographic protocol designed to provide both privacy and security between computers.

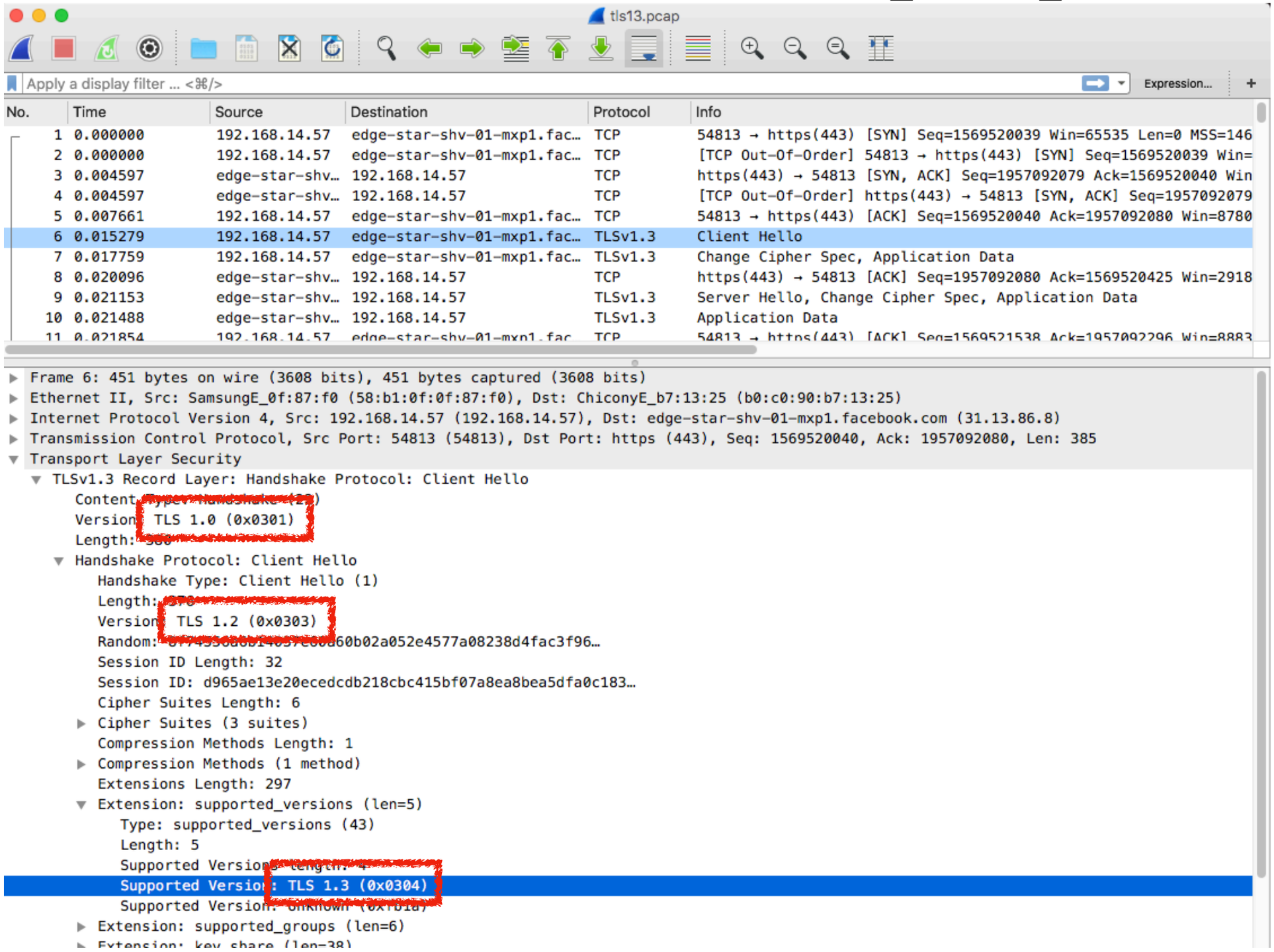
Most Popular →
Latest Version →

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecation planned in 2020[11]
TLS 1.1	2006	Deprecation planned in 2020[11]
TLS 1.2	2008	
TLS 1.3	2018	

SSL For Dummies:

- https://www.wst.space/ssl-part1-ciphersuite-hashing-encryption/
- https://www.wst.space/ssl-part-2-diffie-hellman-key-exchange/
- https://www.wst.space/ssl-part-3-certificate-authority/
- https://www.wst.space/ssl-part-4-tls-handshake-protocol/

Welcome to TLS [2/2]

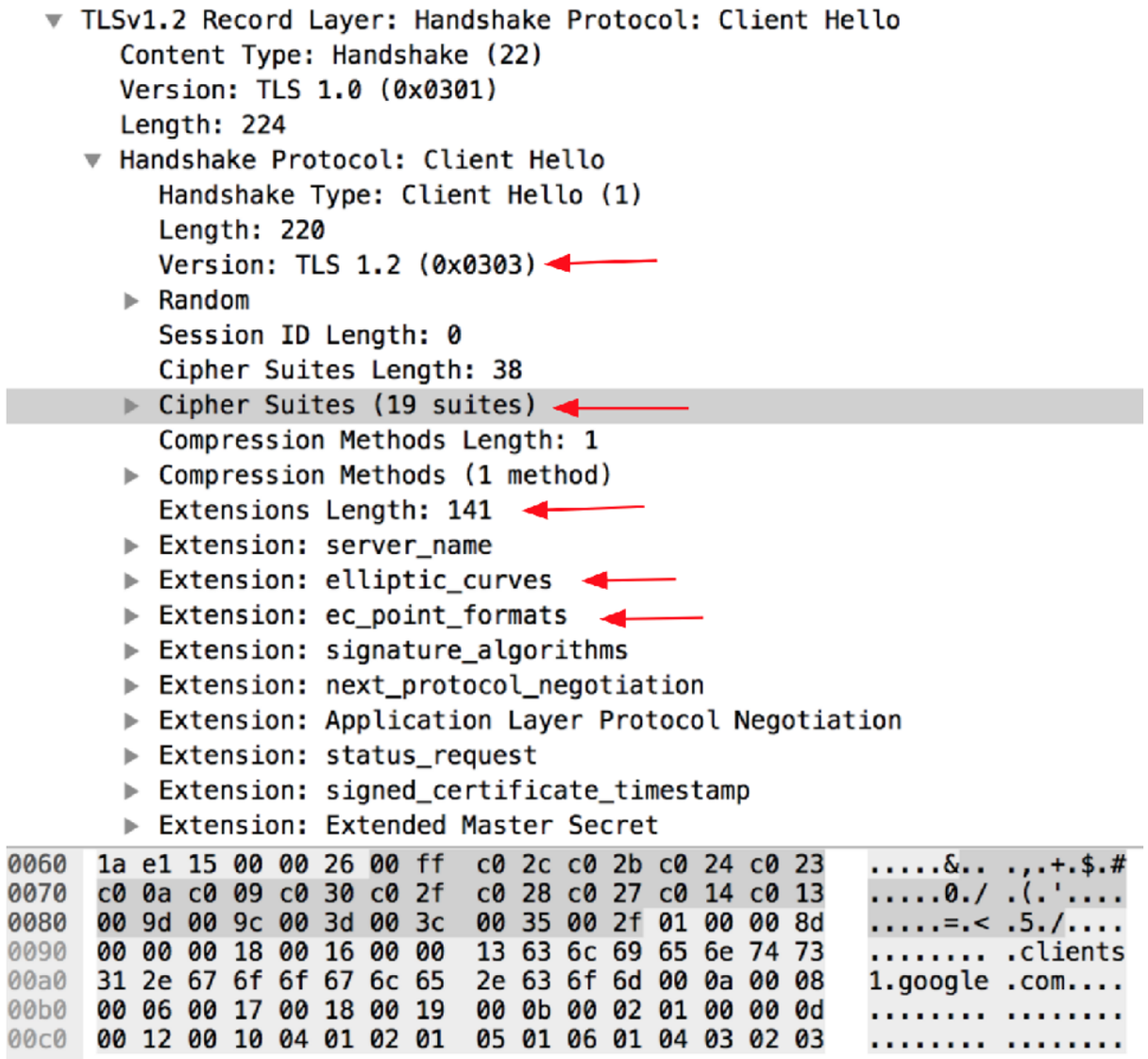


JA3: TLS Fingerprinting

- Similar to HASSH but for TLS/SSL, it has been designed for malware detection.
- JA3 fingerprints the way that a client application communicates over TLS.
- JA3S fingerprints the server response.
- They essentially create a fingerprint of the cryptographic negotiation between client and server.

https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967

JA3: TLS Fingerprinting [1/2]



JA3C `TLSVersion,Ciphers,Extensions,EllipticCurves,EllipticCurvePointFormats`

JA3S `TLSVersion,Cipher,Extensions`

JA3:TLS Fingerprinting [2/2]

TLS ClientHello

```
Secure Sockets Layer
  TLSv1 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 214
    Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 210
        Version: TLS 1.0 (0x0301)
      > Random
        Session ID Length: 0
        Cipher Suites Length: 120
      > Cipher Suites (60 suites)
        Compression Methods Length: 1
      > Compression Methods (1 method)
        Extensions Length: 49
      > Extension: ec_point_formats
      > Extension: elliptic_curves
      > Extension: SessionTicket TLS
      > Extension: Heartbeat
```

Possible Clients

True Client

OpenSSL
(v: 1.0.1r)

https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967

https://blogs.cisco.com/security/detecting-encrypted-malware-traffic-without-decryption

Using JA3 and TLS Fingerprint

ntop | Flows | Hosts | Interface | System | { eth0 } | Search

Flow: [redacted]:47722 ⇄ 162.125.19.9:443 Overview

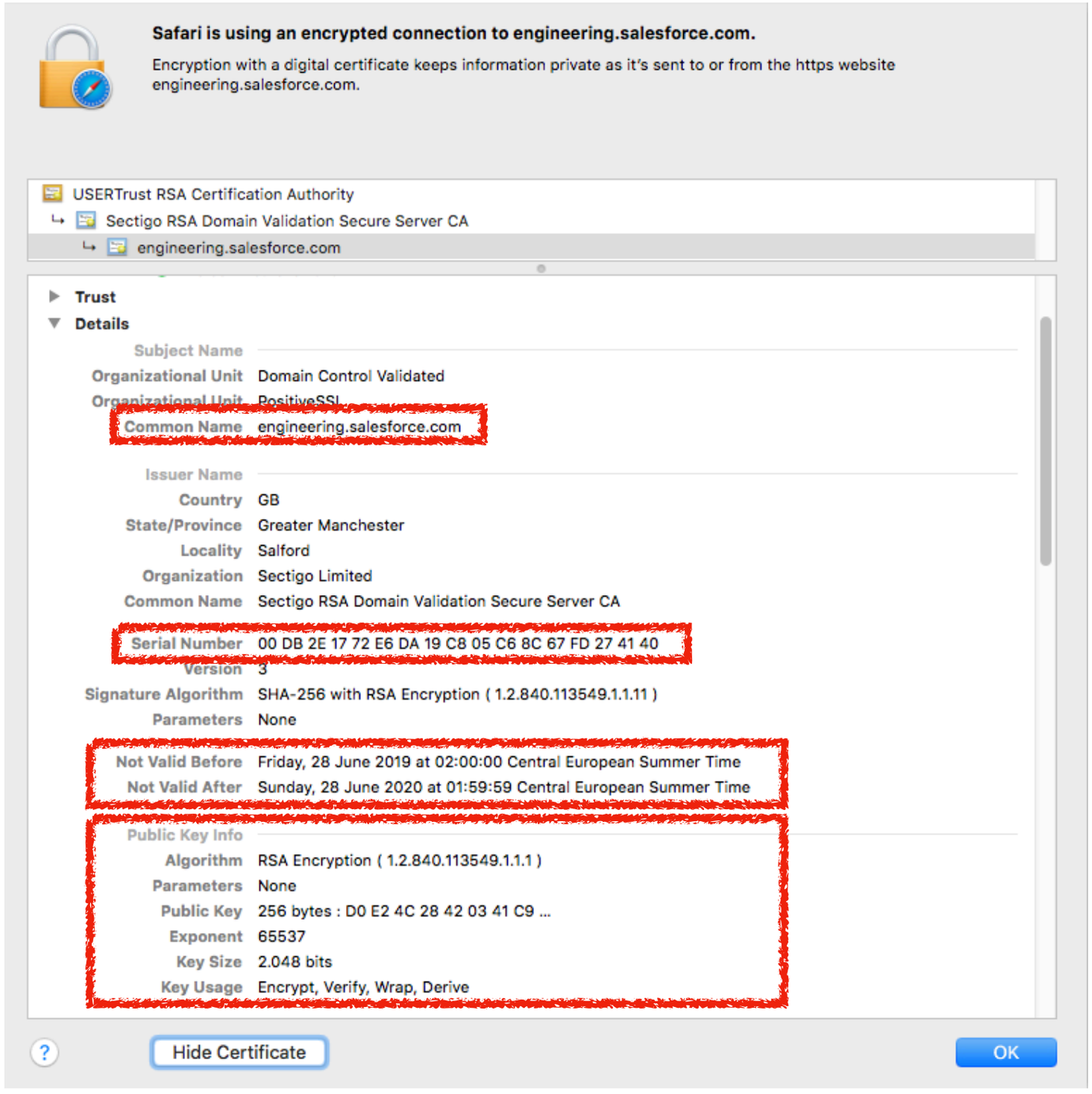
Flow Peers [Client / Server]	[redacted] :47722 [64:00:6A:63:35:CC] ⇄ 162.125.19.9 :443 [F4:B5:2F:FC:AF:F0]	
Protocol / Application	TCP / TLS.Dropbox (Cloud) [TLS v1.2]	
First / Last Seen	24/01/2020 00:47:33 [00:47 ago]	24/01/2020 00:47:34 [00:46 ago]
Total Traffic	Total: 3.6 KB	Goodput: 2.5 KB (70.9 %)
	Client → Server: 9 Pkts / 2.7 KB	Client ← Server: 7 Pkts / 933 Bytes
	pc-deri.nic.it:47722 / 162.125.19.9:443	
Round-Trip Time Breakdown	50.298 ms (server)	
Client/Server Estimated Distance	10,108 Km	6,271 Miles
Application Latency	100.82 ms	
Packet Inter-Arrival Time [Min / Avg / Max]	Client → Server: < 1 ms / 50.12 ms / 100 ms	Client ← Server: < 1 ms / 73.67 ms / 141 ms
TLS Certificate	Client Requested: telemetry.dropbox.com +	
JA3	60efcb8db48448eabf4aa50e4afb28af	2de81c22ea32a57162df5cb08d4a2795
Max (Estimated) TCP Throughput	Client → Server: 5.09 kbit/s	Client ← Server: 79.64 kbit/s
TCP Flags	Client → Server: SYN PUSH ACK	Client ← Server: SYN PUSH ACK
	Flow is active.	
Flow Score	0	

https://github.com/ntop/ntopng

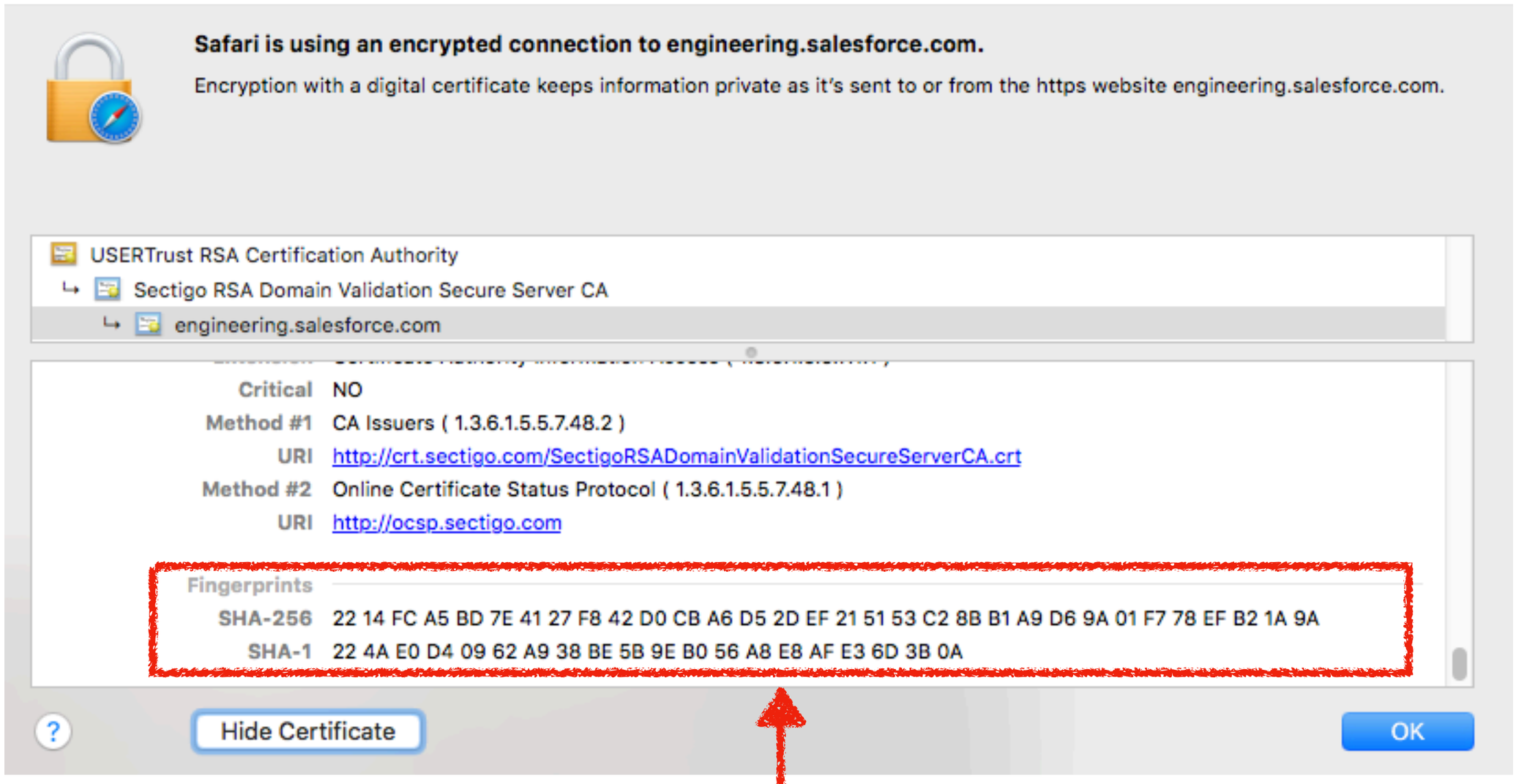
SSL Certificate Fingerprint [1/3]

- Similar to HASSH, JA3 does not take into account the certificate that instead fingerprints the server identity.
- The certificate contains the server name, the trusted certificate authority (CA) that asserts for the authenticity of the certificate, and the server's public encryption key.
- In essence, the certificate is a seal used to guarantee the authenticity of the source of the information.

SSL Certificate Fingerprint [2/3]



SSL Certificate Fingerprint [3/3]



When this changes, the HTTP server configuration has been modified

Fingerprinting Everything

fatt.

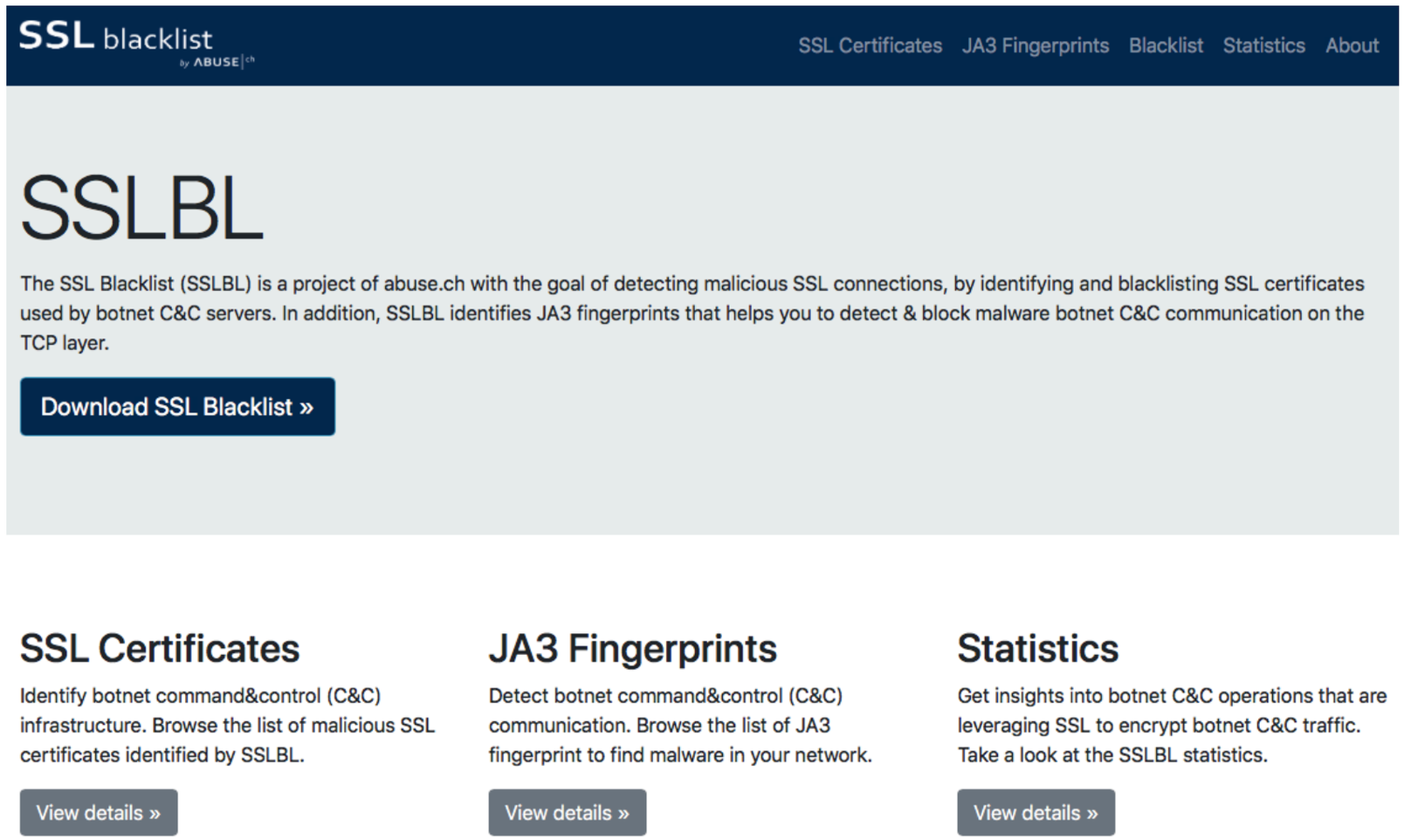
Features

- Protocol support: SSL/TLS, SSH, RDP, HTTP, gQUIC.
  - To be added soon: IETF QUIC, MySQL, MSSQL, etc.
- Fingerprinting
  - JA3: TLS client/server fingerprint
  - HASSH: SSH client/server fingerprint
  - RDFP: my experimental RDP fingerprint for standard RDP security protocol (note that other RDP security modes use TLS and can be fingerprinted with JA3)
  - HTTP header fingerprint
  - gQUIC/iQUIC fingerprint will be added soon
- JSON output

https://github.com/0x4D31/fatt

Fingerprints Databases

- Once fingerprints are computed, they can be matched against known signatures.



https://sslbl.abuse.ch

Are Fingerprint Databases Reliable? [1/2]

- Answer: it depends.
  - SSL Certificate Blacklist can be reliably used to identify for Command&Control or other types of malware.

SSL Certificates

Here you can browse all malicious SSL certificates identified by SSLBL. An SSL certificate is identified by a unique *SHA1 hash* (aka *SSL certificate fingerprint*). You can find more information about how to leverage SSLBL to spot botnet C&C traffic here.

Show 50 entries Search:

Listing Date (UTC)	SSL Certificate	Listing Reason	Malware Samples
2018-11-30 12:38:23	7669103ea0a2e900179e5220a13bf3415438b665	Dridex C&C	3'092
2014-07-09 07:14:48	eb84133c8978541c09ace6044728e621add30726	Shylock C&C	1'420
2015-04-01 13:40:30	d62e065311dffcecad9f8e92c316aafb6019394b	Adwind C&C	1'264
2018-11-18 16:50:32	c17c2bb738627e819e9339f57e8b98967e09f3cb	Gozi C&C	1'219
2018-03-21 15:07:55	e9fbf9ce3a3eea7ba2bdadbab163cff2148dc9e7	TrickBot C&C	1'046

Are Fingerprint Databases Reliable? [2/2]

- JA3 Fingerprint is not reliable as it does NOT identify a specific malware but rather the TLS library used by the malware (e.g. OpenSSL) that can also be used by other apps.

JA3 Fingerprint Blacklist (CSV)

JA3 is an open source tool used to fingerprint SSL/TLS client applications. In the best case, you can use JA3 to identify malware and botnet C2 traffic that is leveraging SSL/TLS. The CSV format is useful if you want to process the JA3 fingerprints further, e.g. loading them into your SIEM. The CSV contains the following values:

- JA3 Fingerprint
- First seen (UTC)
- Last seen (UTC)
- Listing reason

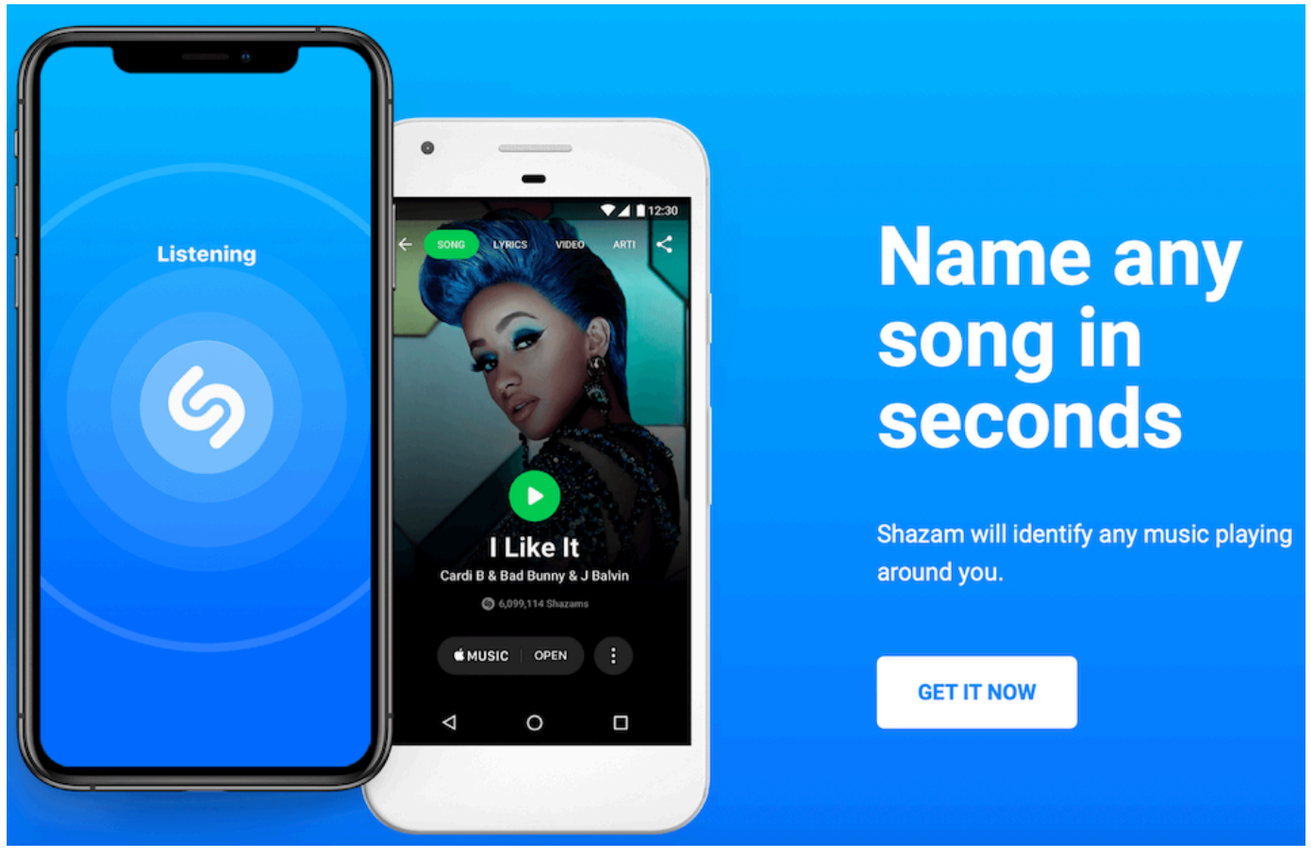
The JA3 Fingerprint Blacklist (CSV) gets generated every 5 minutes. Please do not fetch it more often than every 5 minutes.

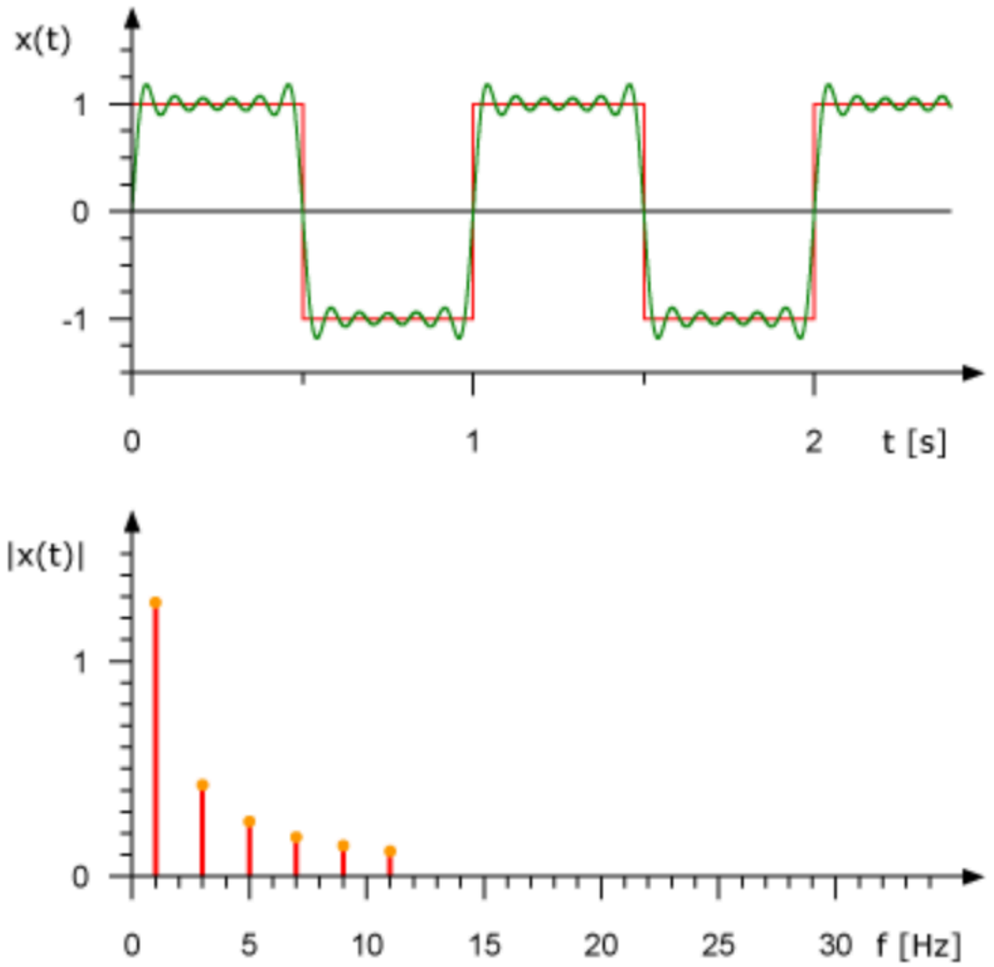
Catching Unexpected Behaviour

Options for Behaviour Analysis

- Option A: Classify good (normal operations) and bad behaviour (malware) and match the current behaviour against the model.
  - Limitations:
    - You will be limited to what you already know (i.e. you won't detect a new malware).
    - You need to annotate traffic for training them model, and this is not what people like to do.
- Option B: Cluster the traffic you have, and check if current traffic matches an existing cluster (good, it's a déjà vu) or if not (bad, as it looks we've something new to handle).
  - Drawback: you might misclassify malware traffic.

Have You Ever Heard of Shazam? [1/2]





https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition

http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

Have You Ever Heard of Shazam? [2/2]

```java
public final int[] RANGE = new int[] { 40, 80, 120, 180, 300 };

// find out in which range is frequency
public int getIndex(int freq) {
    int i = 0;
    while (RANGE[i] < freq)
        i++;
    return i;
}

// result is complex matrix obtained in previous step
for (int t = 0; t < result.length; t++) {
    for (int freq = 40; freq < 300 ; freq++) {
        // Get the magnitude:
        double mag = Math.log(results[t][freq].abs() + 1);

        // Find out which range we are in:
        int index = getIndex(freq);

        // Save the highest magnitude and corresponding frequency:
        if (mag > highscores[t][index]) {
            points[t][index] = freq;
        }
    }

    // form hash tag
    long h = hash(points[t][0], points[t][1], points[t][2], points[t][3]);
}

private static final int FUZ_FACTOR = 2;

private long hash(long p1, long p2, long p3, long p4) {
    return (p4 - (p4 % FUZ_FACTOR)) * 100000000 + (p3 - (p3 % FUZ_FACTOR))
            * 100000 + (p2 - (p2 % FUZ_FACTOR)) * 100
            + (p1 - (p1 % FUZ_FACTOR));
}
```

(1) Create bins

(2) Hash bins

Introduce some Fuzzy Logic

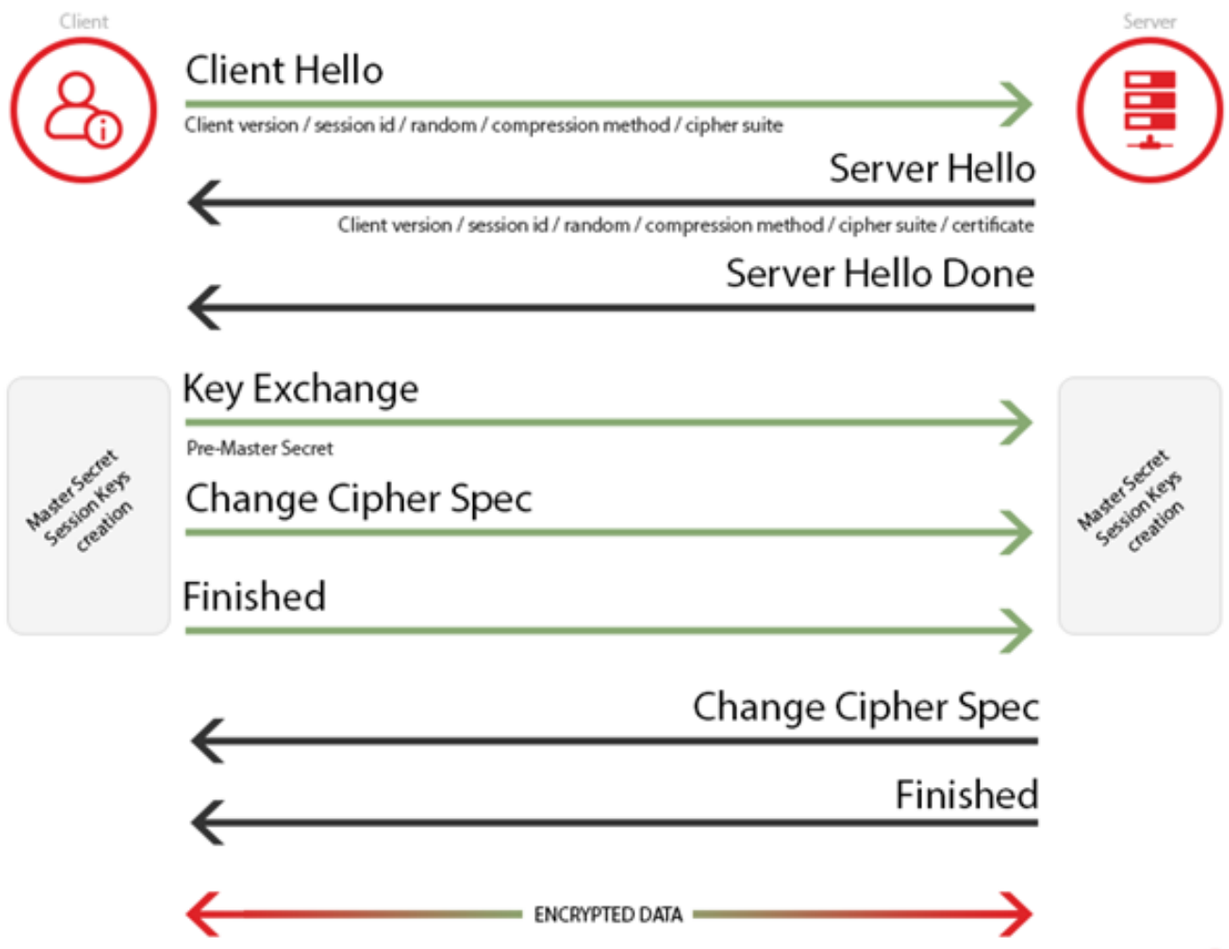
Have You Ever Heard of Shazam? [3/3]

Hash Tag	Time in Seconds	Song
`30 51 99 121 195`	53.52	Song A by artist A
`33 56 92 151 185`	12.32	Song B by artist B
`39 26 89 141 251`	15.34	Song C by artist C
`32 67 100 128 270`	78.43	Song D by artist D
`30 51 99 121 195`	10.89	Song E by artist E
`34 57 95 111 200`	54.52	Song A by artist A
`34 41 93 161 202`	11.89	Song E by artist E

(3) Compare hashes to guess the played song

Catching Malware with Fingerprints [1/3]

- Some malware randomise the clientHello (and thus JA3C) trying to deceive security tools.
- Question: is this a good idea?



Catching Malware with Fingerprints [2/3]

- Answer: no it is not a good idea because a monitoring tool will easily detect cases where one IP address features many JA3C fingerprints.
- Question: how JA3C can be used to fingerprint application behaviour?

Host: [redacted] Traffic Packets Ports Peers ICMP Applications DNS TLS SSH HTTP Flows Sites Processes

JA3 Fingerprint	Application Name	Num Uses
839868ad711dc55bde0d37a87f14740d	/snap/core/8268/usr/lib/snapd/snapd	12
054c9f9d304b7a2add3d6fa75bc20ae4	/home/deri/.dropbox-dist/dropbox-lnx.x86_64-88.4.172/dropbox	6
60efcb8db48448eabf4aa50e4afb28af	/home/deri/.dropbox-dist/dropbox-lnx.x86_64-88.4.172/dropbox	5
456523fc94726331a4d5a2e1d40b2cd7	/usr/bin/curl	4
c50d3f7a28fdbb5ed254ce01876abd6e	/usr/lib/git-core/git-remote-http	1

https://www.ntop.org/ntop/introducing-nprobe-agent-packetless-system-introspected-network-visibility/

Catching Malware with Fingerprints [3/3]

- Fingerprints leverage only on the initial flow bytes and thus are lightweight and predictable in cost.
- They can be used for
  - Hunting malware.
  - Profiling attackers and their tools.
  - Spotting new connections between the attackers/IPs.
  - Detecting evasion techniques that can be instead more evident with fingerprints.
- They can be used together with traffic fingerprint (Similar to Shazam that samples the whole song in 20 sec batches) for the best of both worlds.

Modelling Behaviour

- Fingerprints are used to understand if something "aesthetic" has changed.
- We need more than that:
  - Understand if the behaviour of each connection as well the overall host behaviour is acceptable or at least steady with respect to the known behaviour.
  - Detect unexpected behaviour in encrypted communications.

nDPI Traffic Analysis

- $ ./example/ndpiReader -J -i ./tests/pcap/instagram.pcap -v 2 -f "port 49355”

Behaviour

```
TCP 192.168.2.17:49355 <-> 31.13.86.52:443 [byte dist mean:
125.398474][byte_dist_std: 67.665465][entropy: 0.997011]
[total_entropy: 5609.185931][score: 1.0000][proto: 91.211/
TLS.Instagram][cat: SocialNetwork/6][456 pkts/33086 bytes <->
910 pkts/1277296 bytes][Goodput ratio: 9.0/95.3][14.29 sec]
[bytes ratio: -0.950 (Download)][IAT c2s/s2c min/avg/max/
stddev: 0/0 37.7/0.7 10107/274 546.6/11.8][Pkt Len c2s/s2c
min/avg/max/stddev: 66/66 72.6/1403.6 657/1454 57.2/231.0]
[TLSv1.3 (Fizz)][Client: scontent-mxp1-1.cdninstagram.com]
[JA3C: 7a29c223fb122ec64d10f0a159e07996][JA3S:
f4febc55ea12b31ae17cfb7e614afda8][Cipher:
TLS_AES_128_GCM_SHA256]
```

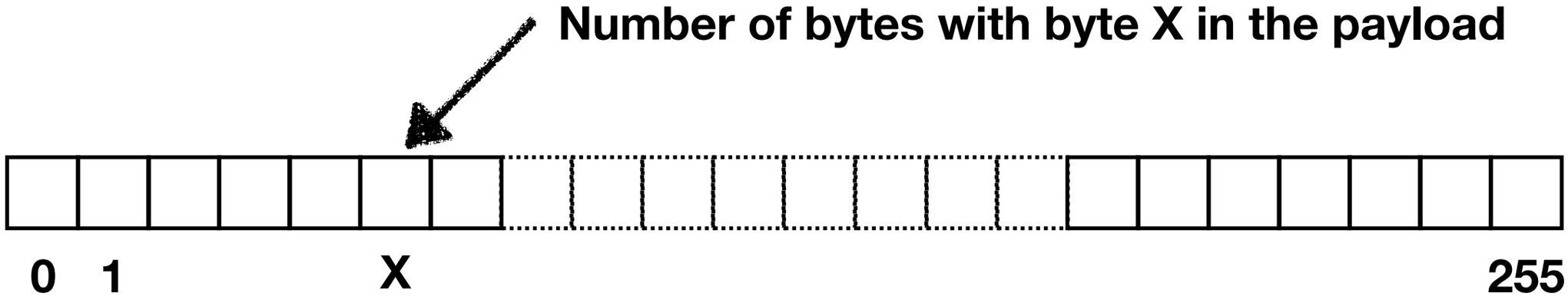
Fingerprint

Bytes Entropy [1/4]

- Metric used to measure how bytes are bytes are distributed: the larger the entropy, the greater the uncertainty in predicting the value of an observation.
- Formula: the entropy of X is determined by computing the sum of $-p(x)\log_2(p(x))$, where x varies over all possible values for an observation of X and p(x) is the (a priori) probability that an observation will have value x.

https://csrc.nist.gov/csrc/media/publications/sp/800-90b/draft/documents/draft-sp800-90b.pdf

Bytes Entropy [2/4]

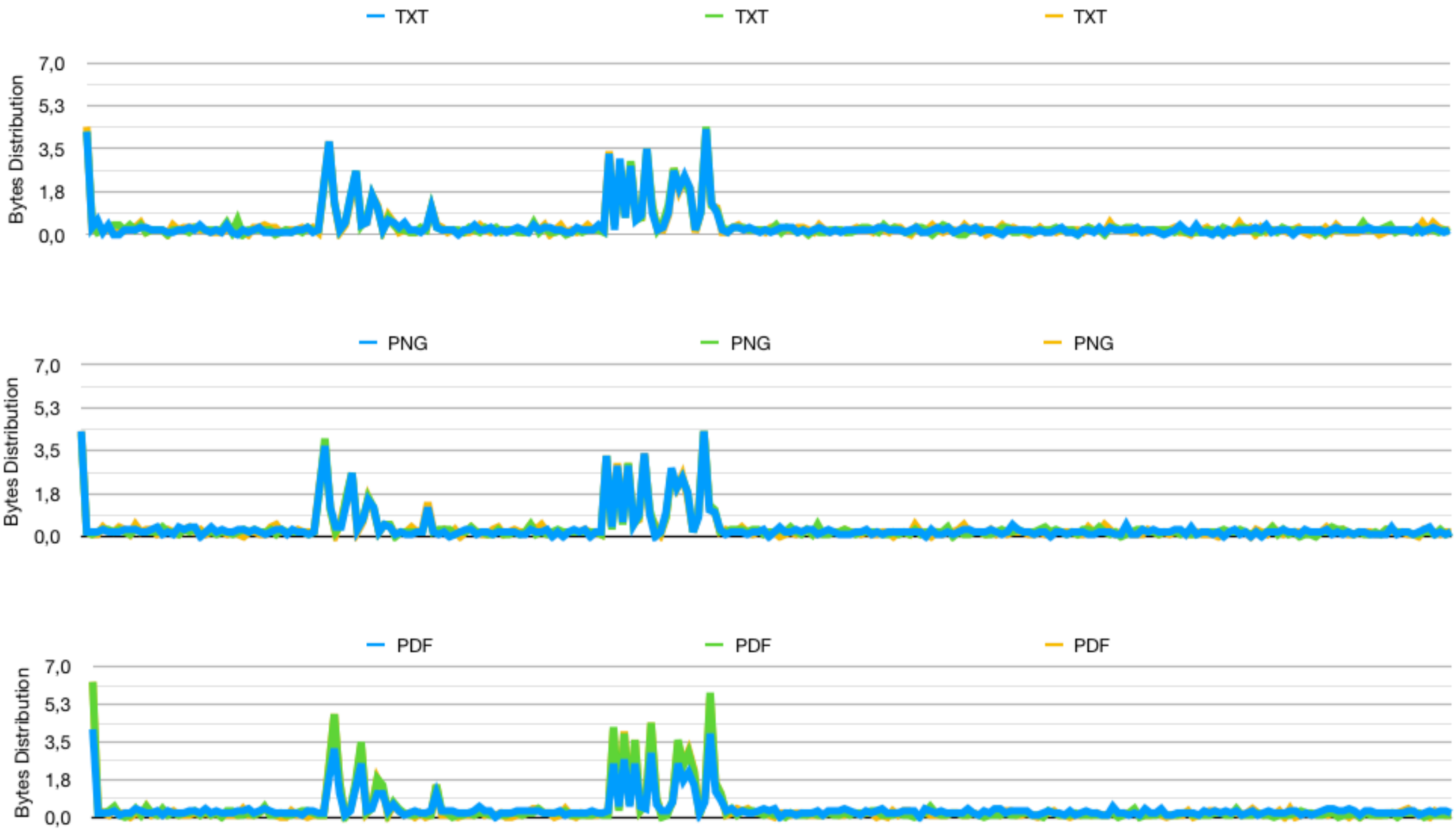


- Entropy of raw data before and after encryption (TLS) changes but is it within limited boundaries for homogeneous data.

Server Entropy (SCP)

PDF	PNG	TEXT	
6,418	7,014	7,008	
6,431	7,019	7,007	
6,428	6,994	7,011	
6,426	7,009	7,009	Average
0,007	0,013	0,002	StdDev

Bytes Entropy [3/4]



Bytes Entropy [4/4]

Client Entropy (TLS)

Instagram	Skype	Belkin	DNS
6,702	6,497	6,497	3,493
6,701	5,429	5,429	3,131
6,338	7,698	7,698	3,74
7,868	7,135	7,135	3,363
6,747	4,817	4,817	3,363
4,798	7,157	7,157	3,106
6,730	7,308	7,308	3,160
6,555	6,577	6,577	3,337
0,910	1,069	1,069	0,229

Average	
StdDev	

Server Entropy (TLS)

Instagram	Skype	Belkin	DNS
7,975	7,512	7,973	4,375
7,964	7,649	7,987	4,153
7,964	7,378	7,991	4,159
7,949	7,682	7,995	4,625
7,983	7,574	7,894	4,496
7,937	7,503	7,995	4,494
7,983	7,691	7,913	4,262
7,965	7,570	7,964	4,366
0,017	0,114	0,042	0,182

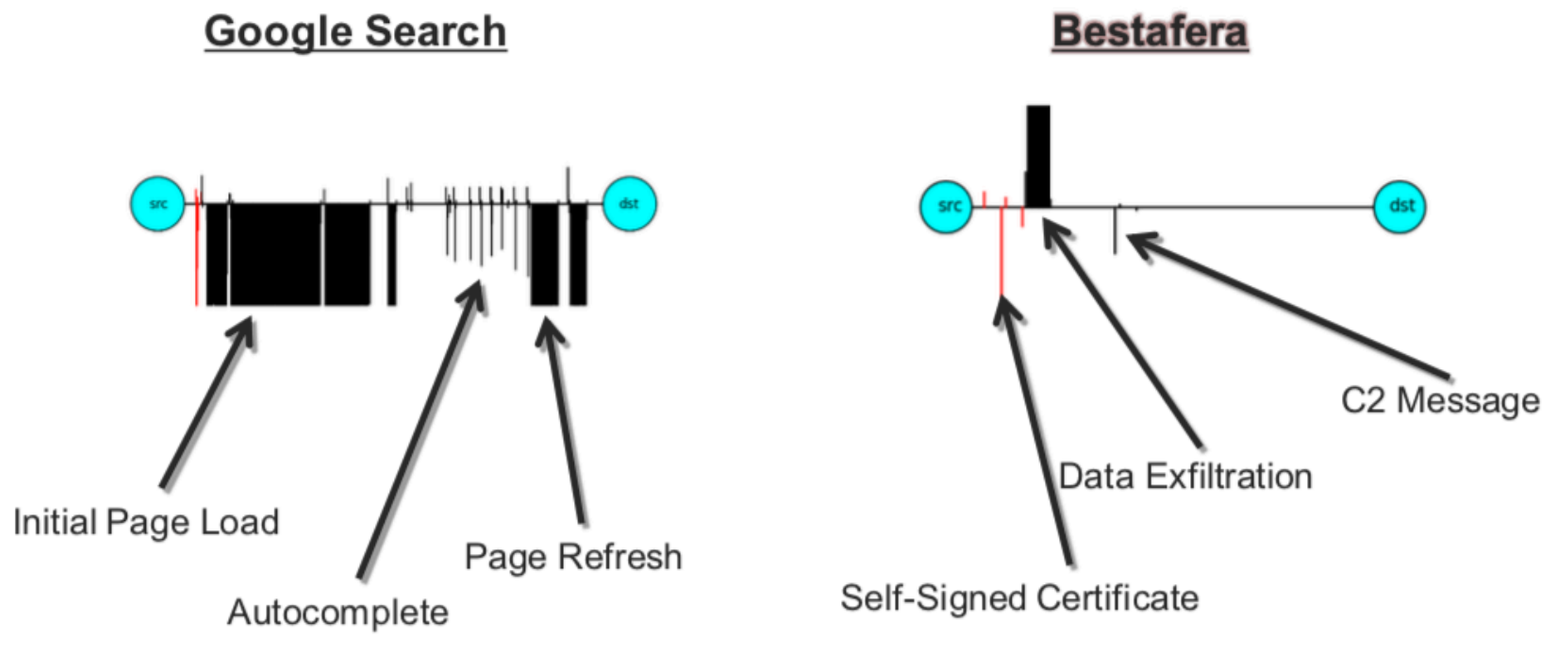
- Entropy is "nice to have" but it cannot be used alone to check protocol compliancy.

SPLT [1/3]

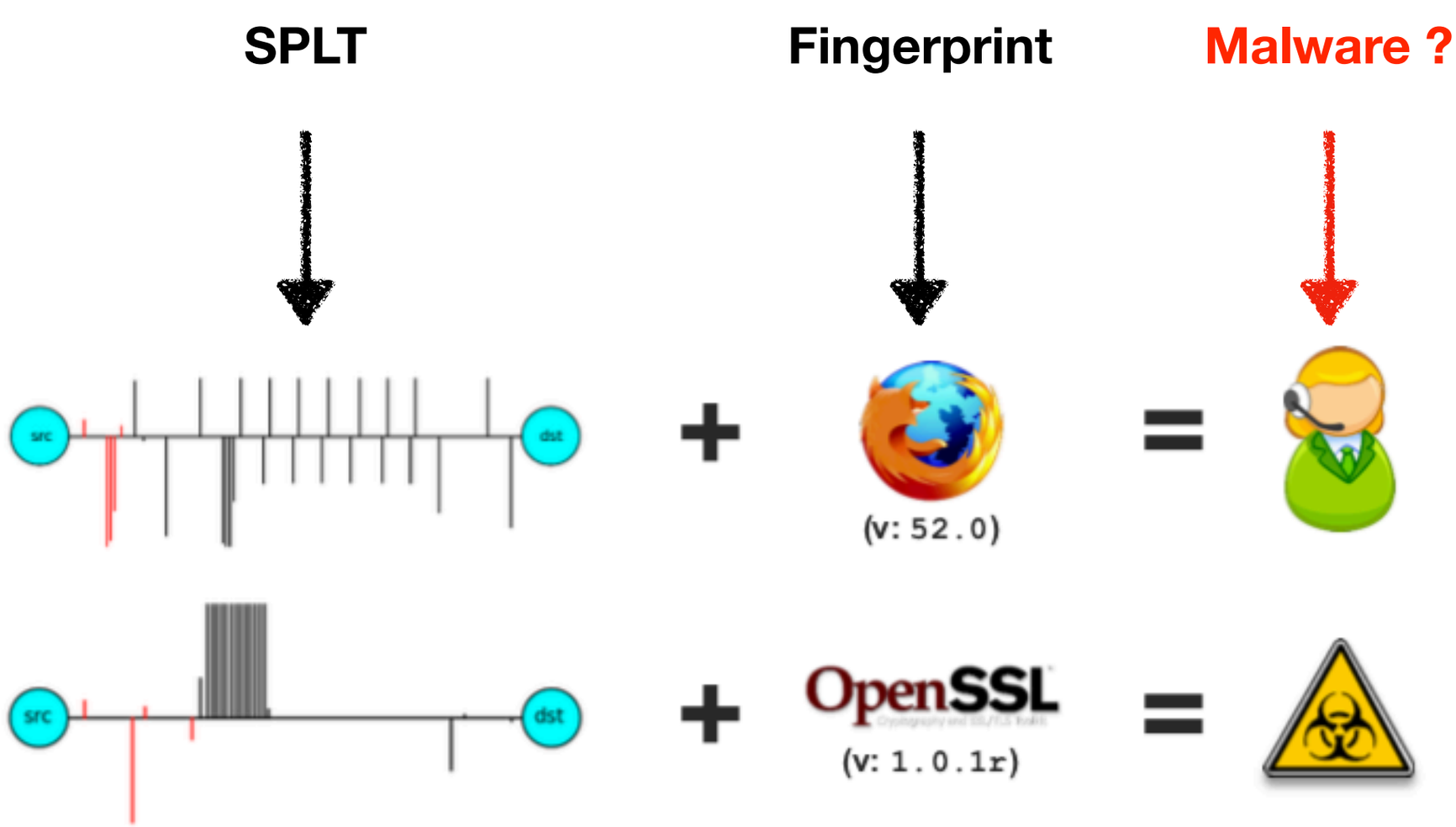
- SPLT (Sequence of Packet Length and Time) is a metric also used by Cisco Joy (https://github.com/cisco/joy) to fingerprint malware.
- For the first 50 packets, payload len and time are divided in 10 bins (i.e. 0-149 bytes, 150-299…).
- Then a matrix 10 x 10 is created: in each cell (i,j) there's a number that represents the number of times that a packet x in bin i, has as packed x+1 packet in bin j. In essence this is a Markov chain with cells representing the transition probability.

SPLT [2/3]

- SPLT is similar to Google PageRank (probability to go from site X to site Y).
- SPLT is used as a set of 100 (10 x 10) features for a ML (Machine Learning) algorithm for identifying malware.



SPLT [3/3]



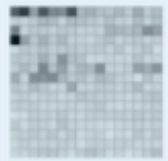
Malware Traffic Analysis [1/5]

		Client	Session	Server
Contextual Flows	TCP/IP	Source Address Source Port	# Bytes # Packets	Destination Address Destination Port
	Intraflow		Packet Lengths Packet Arrival Times	
	TLS	Ciphersuite Offer Vector Extensions Offer Supported Elliptic Curves SNI	Record Length Record Times Record Types	Certificate Chain Selected Ciphersuite
	DNS	Name		Response Code TTL
	HTTP	Headers	Headers File Magic	Headers

Malware Traffic Analysis [2/5]

- SPLT – Sequence of Packet Lengths and Arrival Times

- Byte Distribution



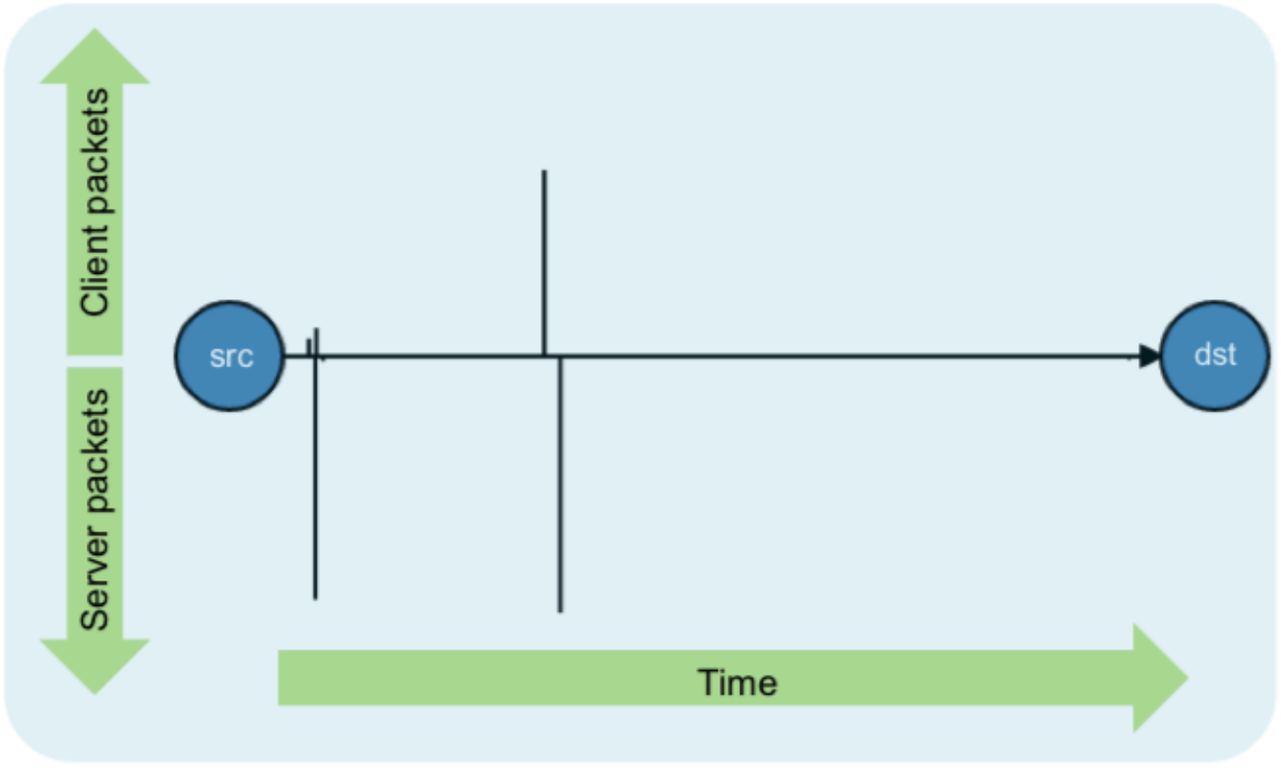
- Byte Entropy
- TLS unencrypted header data
  - Certificates, SNI, Ciphersuites, Extensions

- DNS linked flows

- HTTP linked flows

Malware Traffic Analysis [3/5]

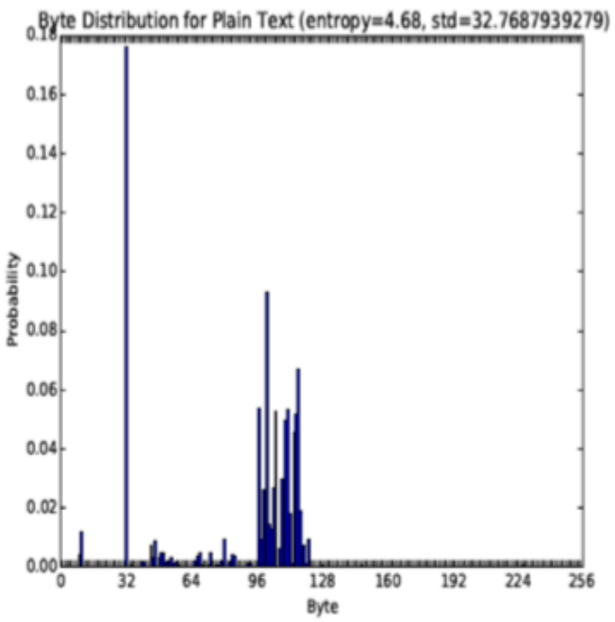
Sequence of Packet Lengths and Times

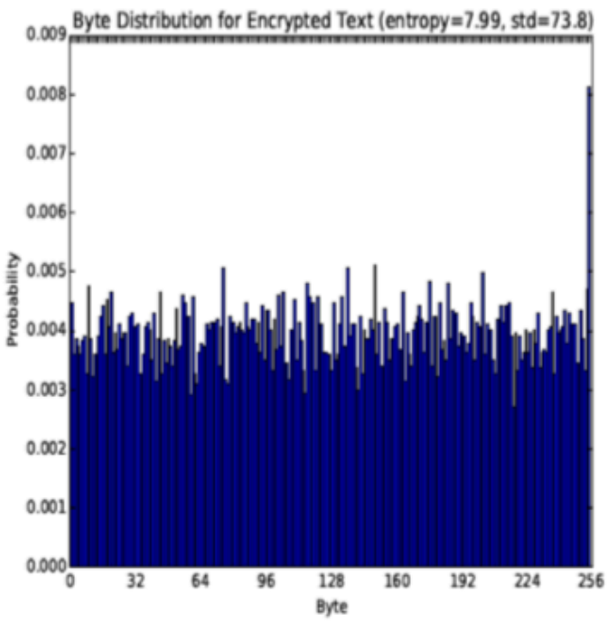



```
"packets": [
  {  "b": 22,  "ipt": 33, "dir": ">" } ,
  {  "b": 1432,  "ipt": 4, "dir": "<" } ,
  {  "b": 30, "ipt": 1,  "dir": ">" } ,
  {  "b“: 4, "ipt": 145, "dir": "<" },
  ...
]
```

Malware Traffic Analysis [4/5]

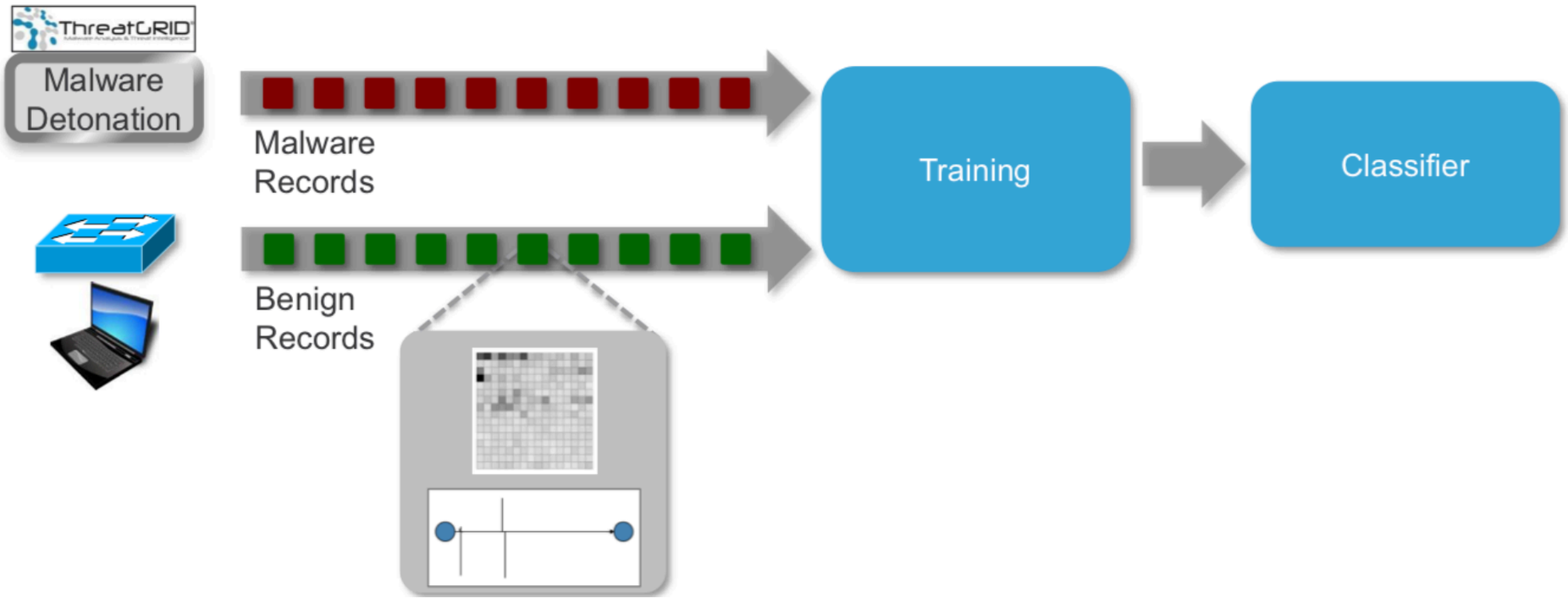
Byte Distribution and entropy





“entropy": 7.165,
"bd": [
23, 7, 4, 8, 4, 12, 7, 4,
12, 5, 98, 6, 5, 101, 14, 8,
9, 9, 6, 8, 10, 6, 10, 6,
16, 8, 3, 16, 7, 7, 3, 11,
189, 6, 24, 9, 10, 10, 5, 7,
19, 8, 16, 8, 34, 79, 61, 90,
102, 91, 56, 47, 35, 47, 30, 25,
...
]

Malware Traffic Analysis [5/5]



In Summary…

- New trends in Internet traffic make traditional payload-inspection-based monitoring tools useless as traffic is now encrypted.
- Encryption does not mean that we have to be blind, but rather that we need to change the approach to the problem.
- We have shown what are the techniques used to fingerprint and characterise network traffic.
- nDPI is an open-source toolkit able to implement all the algorithms covered in this talk.

What's Next? [1/3]

- Flow-based analysis (DPI, fingerprint…) is the baseline for traffic analysis, but it is not sufficient to detect all the flaws as it is unable to model the overall host behaviour.
- Host misbehaving with respect to its "known" behaviour can be an indication of issues such as a scan, a malware, multiple login failures, or changes in behaviour due to a software update or reconfiguration.
- These facts are invisible at flow level, unfortunately.

What's Next? [2/3]

Active Alerted Flows

10 ▾ Hosts ▾ Status ▾ Direction ▾ Applications ▾ Categories ▾ IP Version ▾ Protocol ▾ VLAN ▾

	Application	Protocol	VLAN	Client	Server	Duration	Score	Breakdown	Actual Thpt	Total Bytes	Info
Info	? Unknown	⚠ TCP	125	:32319	1 :5500	< 1 sec	180	Client	0 bit/s —	64 Bytes	
Info	? Unknown	⚠ TCP	125	:44184	1 :52869	< 1 sec	180	Client	0 bit/s —	64 Bytes	
Info	? Unknown	⚠ TCP	125	:44184	1 :52869	< 1 sec	180	Client	0 bit/s —	64 Bytes	
Info	SMBv23 👍	⚠ TCP	125	:53501	1 :microsoft-ds	< 1 sec	180	Client	0 bit/s —	64 Bytes	
Info	? Unknown	⚠ TCP	125	:44743	1 :81	< 1 sec	180	Client	0 bit/s —	64 Bytes	
Info	MsSQL-TDS 👍	⚠ TCP	125	7869	9 :ms-sql-s	< 1 sec	180	Client	0 bit/s —	128 Bytes	
Info	? Unknown	⚠ TCP	125	:32953	1 :81	< 1 sec	180	Client	0 bit/s —	64 Bytes	
Info	? Unknown	⚠ TCP	125	:40940	9 :81	< 1 sec	180	Client	0 bit/s —	64 Bytes	
Info	? Unknown	⚠ TCP	125	:48095	9 :81	< 1 sec	180	Client	0 bit/s —	64 Bytes	
Info	? Unknown	⚠ TCP	125	:50096	1 :81	< 1 sec	180	Client	0 bit/s —	64 Bytes	

Showing 1 to 10 of 37747 rows. Idle flows not listed.

« < 1 2 3 4 5 > »

NOTES:

- Check out the online documentation for a description of the misbehaving flows.

ntopng Enterprise Edition v.3.9.200126
User admin Interface

70%
702.80 Mbit/s [91.5 Kpps]

0 bps
702.80 Mbit/s

19:04:10 +0100 | Uptime: 05:41
223 ⚠ 38,245 Flows ⚠ 16,581 💻 6,302 💻 282 Devices
194,220 Flows

What's Next? [3/3]

All Hosts

10 ▾ | IP Version ▾ | VLAN ▾ | Direction ▾ | Filter Hosts ▾

	IP Address	VLAN	Location	Flows	Score	Name	Seen Since	Breakdown	Throughput	Total Bytes
Flows	2	125	Remote	1619	32,871		08:24	Sent Rcvd	41.6 kbit/s ↓	2.21 MB
Flows		125	Remote	280	29,400		08:23	Sent	1.43 kbit/s ↓	74.12 KB
Flows		125	Remote	2180	17,908		08:38	Sent	179.15 kbit/s ↑	11.19 MB
Flows		125	Remote	35522	15,350		08:38	Sent	197.27 kbit/s ↑	10.49 MB
Flows		125	Remote	22085	13,073		08:38	Sent	661.49 kbit/s ↑	32.62 MB
Flows	4	125	Remote	117	6,005		08:22	Sent	511.85 bit/s ↓	27.19 KB
Flows		125	Remote	27	5,500		08:38	Sent	13.79 kbit/s ↓	1.41 MB
Flows		125	Remote	36	5,450		08:23	Sent	1.78 kbit/s ↓	247.51 KB
Flows		125	Local	107	5,356		08:38	Sent Rcvd	31.2 kbit/s ↑	1.62 MB
Flows		125	Local	1856	4,765		08:38	Rcvd	60.71 Mbit/s ↑	3.13 GB

Showing 1 to 10 of 22785 rows. Idle hosts not listed.

« < 1 2 3 4 5 > »

ntopng Enterprise Edition v.3.9.200126
User admin Interface

75%
749.50 Mbit/s [97.1 Kpps]

↑ 0 bps
↓ 749.50 Mbit/s

19:07:13 +0100 | Uptime: 08:44
1,683 | 51,256 Flows | 16,578 | 6,375 | 285 Devices
185,878 Flows


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
рахмат
Баярлалаа
спасибо
gracias
merci
obrigado
dziękuje
bedankt
go raibh maith agat
mochchakkeram
sukriya
kop khun krap
terima kasih
grazie
arigatô
dakujem
takk
dank je
tapadh leat
hvala
mauruuru
sagolun
ευχαριστώ
감사합니다
ありがとう
мерси
хвала
mersi
barka
welalin
tack
spas
kia ora
faafetai lava
vinaka
спасибі
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
asante
manana
obrigada
tenki
murakoze
chokrane
mamnun
дякую
djiere dieuf
tau
kiitos
dankie
nanni
nandri
dhanyavad
bayarlalaa
gracie
köszönöm
dankon
aciu
akun
enkosi
sobodi
dėkuji
chnorakaloutioun
gratias ago
gracies
sulpay
taiku
mèsi
didi madloba
najis tuke
kam sah hamnida
rahmat
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
merce
trugarez
xièxie