Spoofing JARM signatures. I am the Cobalt Strike server now!

grimminck.medium.com/spoofing-jarm-signatures-i-am-the-cobalt-strike-server-now-a27bd549fc6b

Stefan Grimminck December 25, 2020

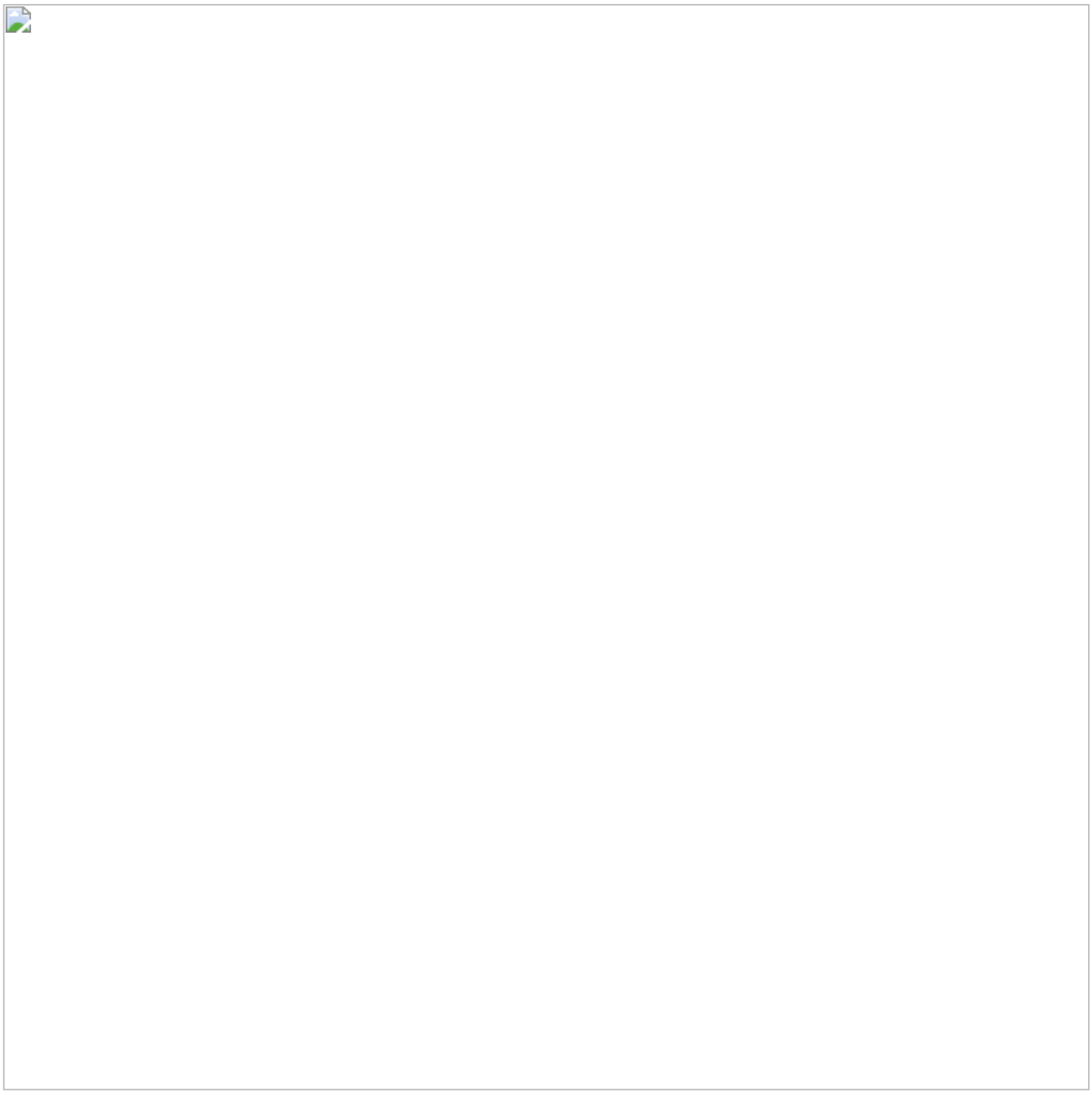
Stefan Grimminck

Dec 25, 2020

.

3 min read

TL;DR: JARM is very useful fingerprinting tool, but can be deceived by replaying server hello's from other services.

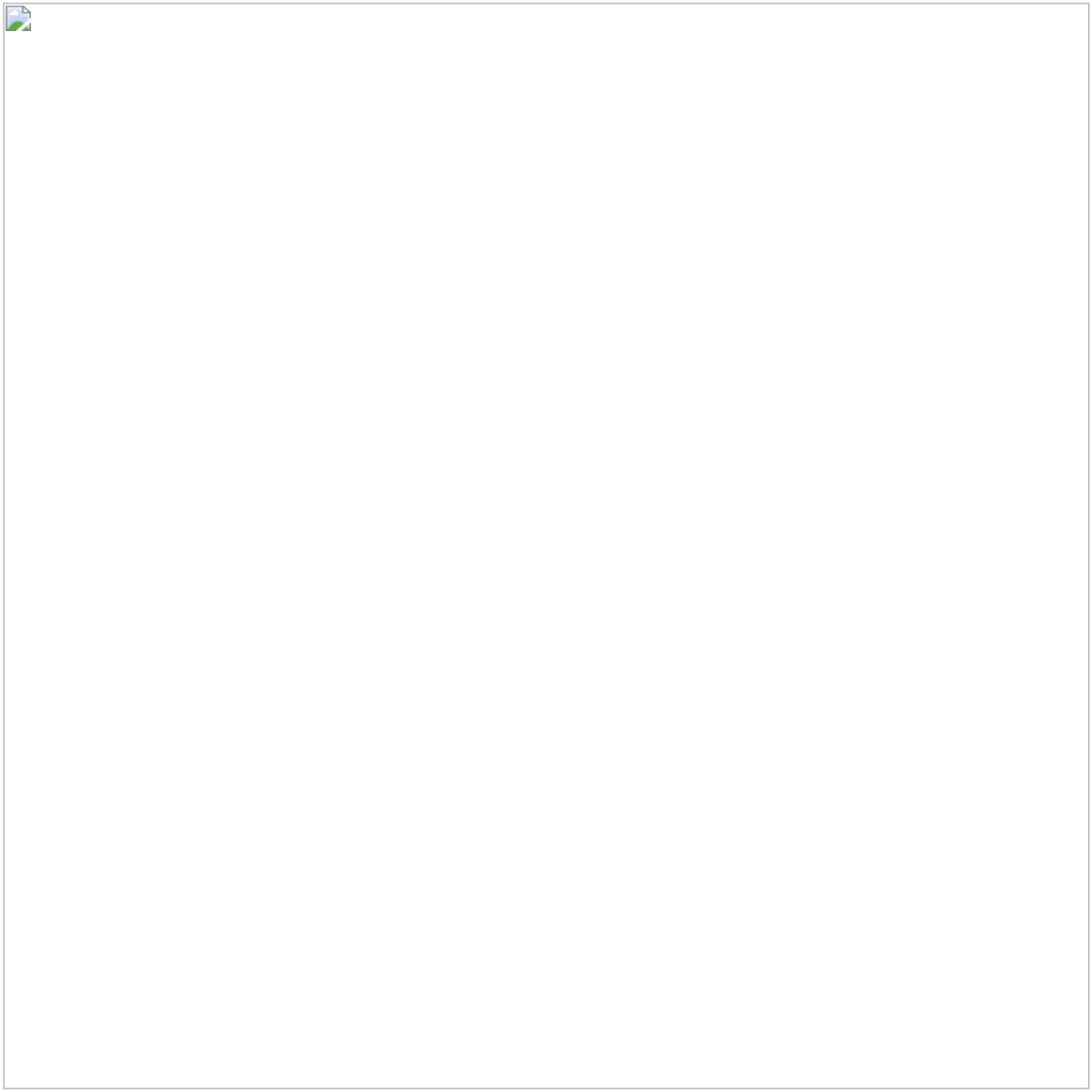


The JARM scanner created by @SalesforceEng is quite an effective tool for system fingerprinting. It uses the Server Hello responses from a TLS handshake to generate a signature. These can then be used to find similar software or services. Ideal for finding C2 or other malicious servers that implement TLS. So, It doesn't come as a surprise that Shodan.io uses this fingerprinting mechanism in their scanners. Read the Salesforce post for more information about the JARM library, scanner and its uses.

The question, then, arises: Is it possible to spoof these JARM signatures? Let's find out! Salesforce stated in their post that scanning a Cobalt Strike server would result in the following signature `07d14d16d21d21d07c42d41d00041d24a458a375eef0c576d23a7bab9a9fb1`

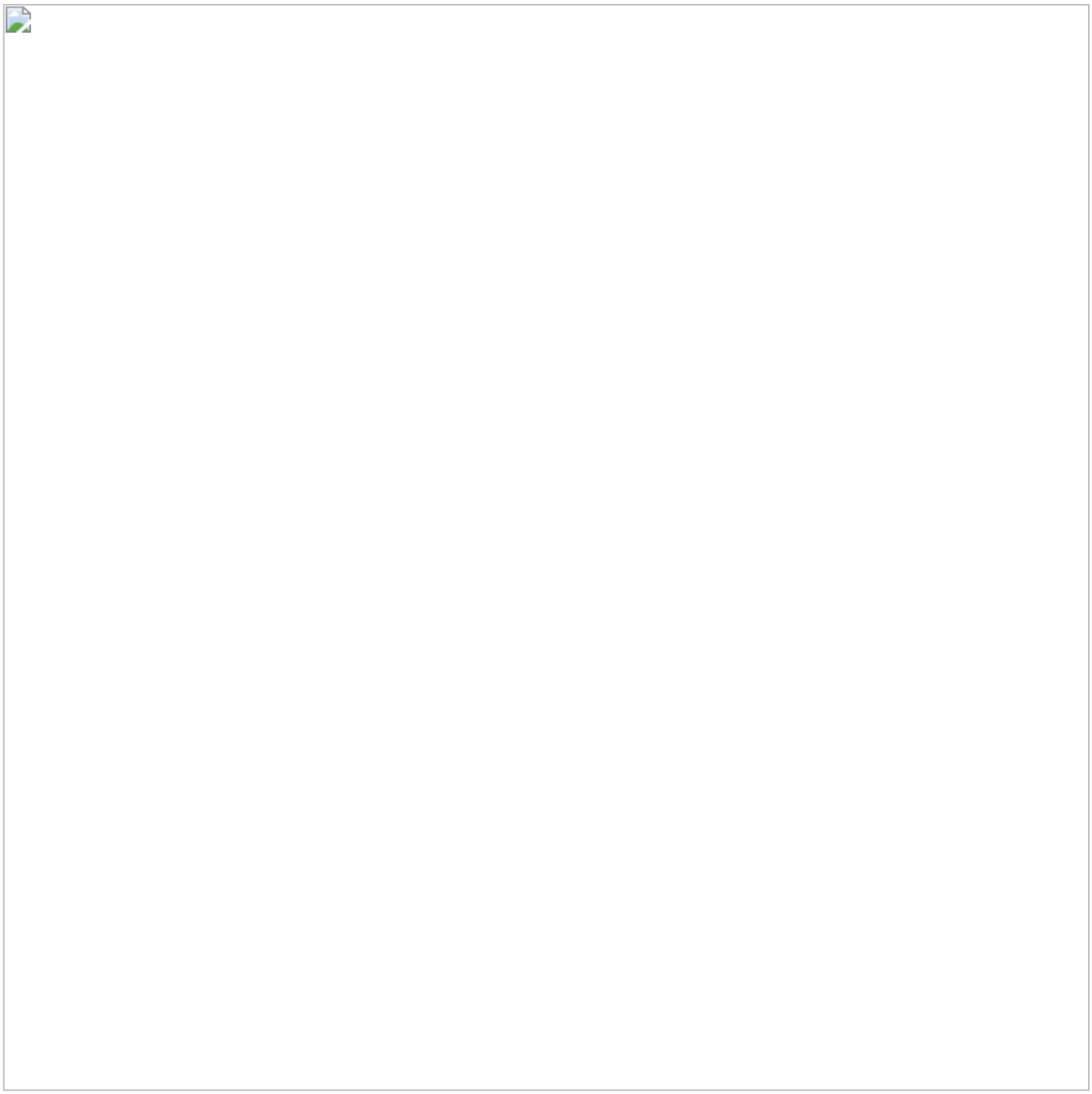
That this signature isn't Cobalt Strike specific, was revealed in the Cobalt Strike blog. Let's still use it as a starting point anyway.

First I used the list of addresses published by Salesforce to find a server with a matching hash. I scanned it using jarmscan and created a packet capture of the response. The ssl handshake (filter: `ssl.handshake.type == 1`) filter in Wireshark will display all TLS client hello's sent by the scanner.



Wireshark capture of 10 TLS Client Hello's

And in turn the "Cobalt Strike" server will return its Server Hello's. These are used by jarmscan to generate a unique signature (filter: `ssl.handshake.type == 2`).

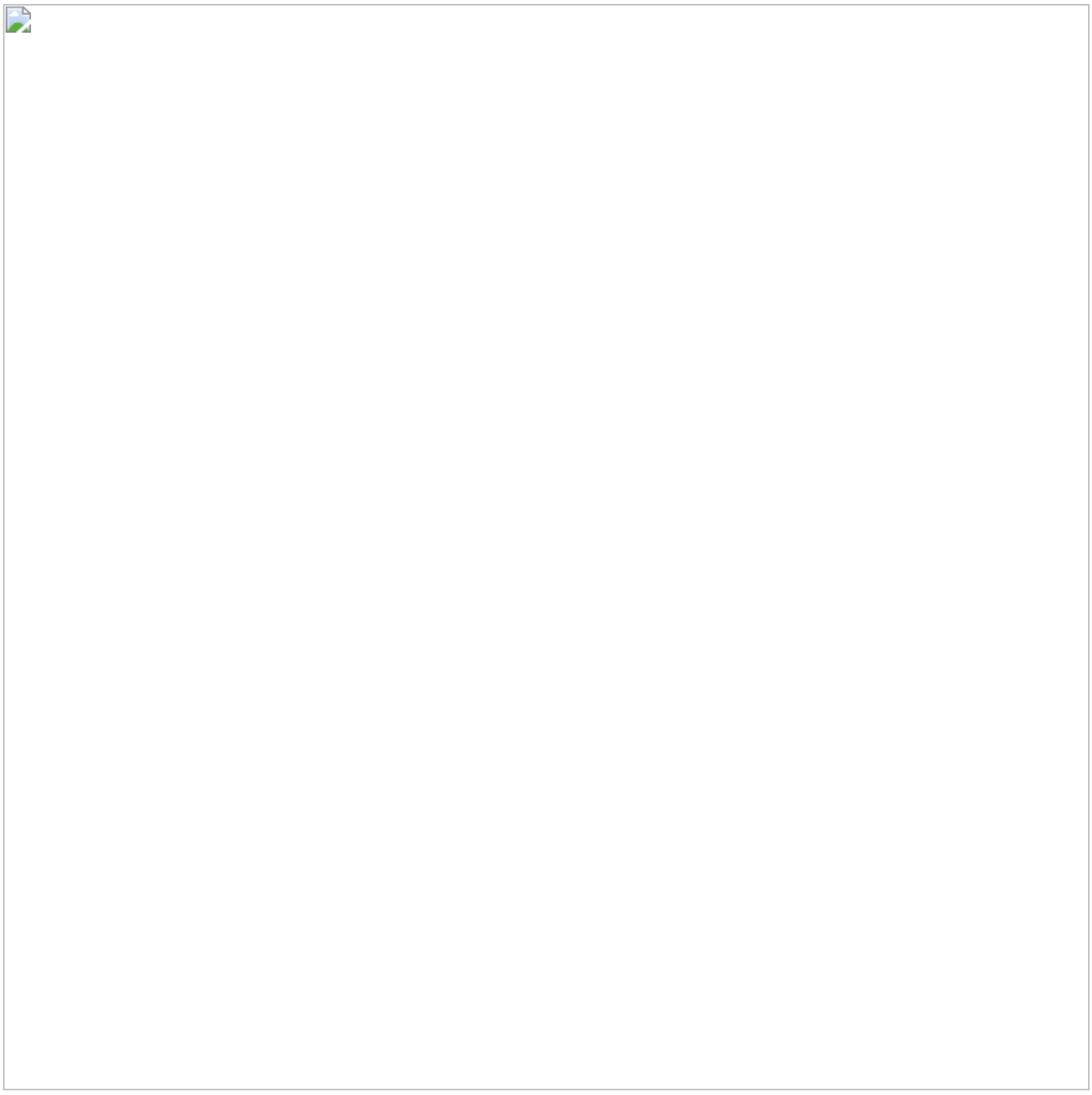


Wireshark capture of 10 TLS Server Hello’s

These Server Hello’s are the packets we want to replay. This can easily be done by setting up a TCP server listening for the specific Client Hello’s, then replaying their corresponding Server Hello’s captured from the alleged Cobalt Strike server. A rather lazy, but effective approach.

I scanned the server on three separate occasions and found the duplicate bytes for every request. I used these bytes to identify each specific Client Hello.

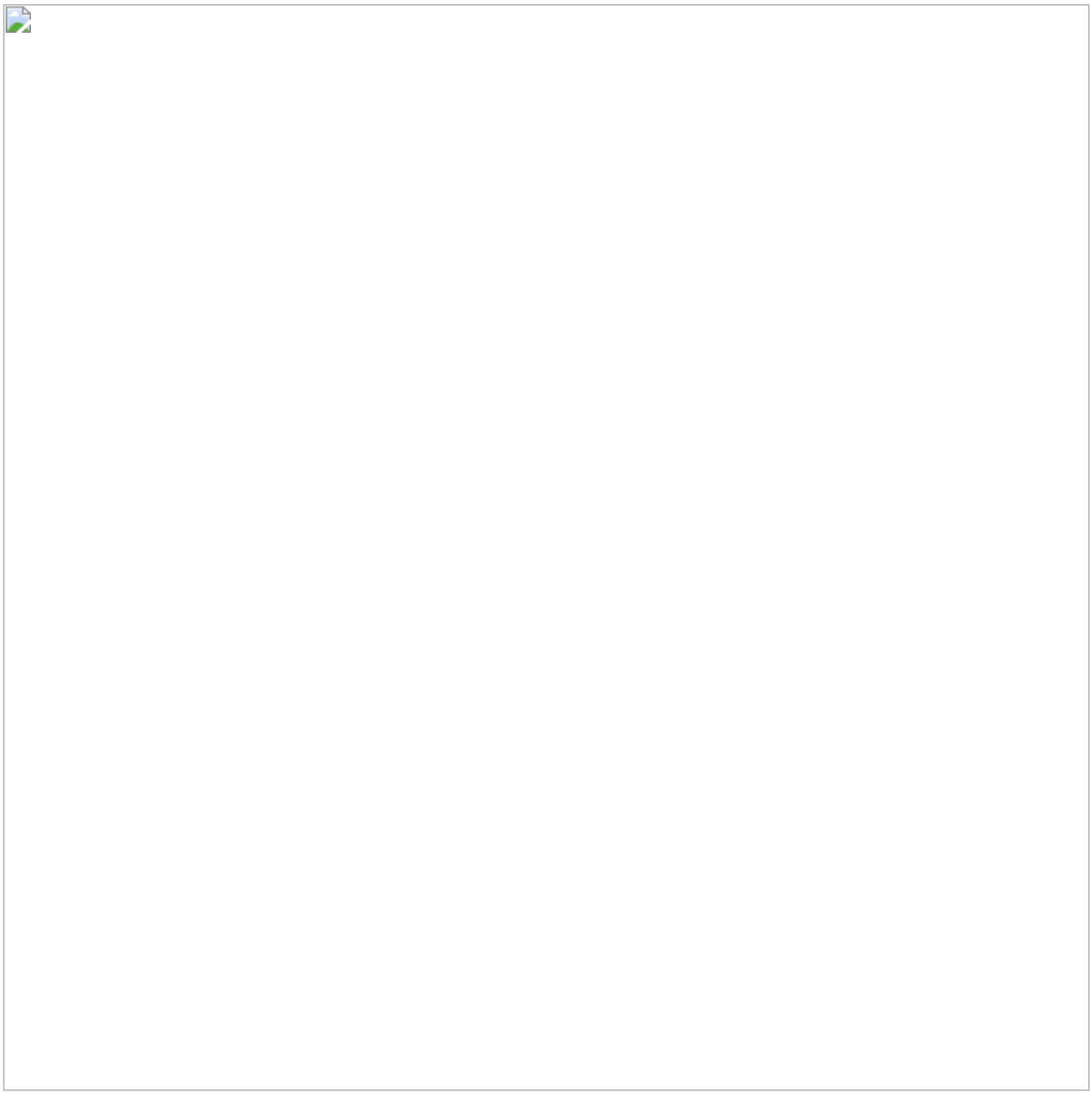
Luckily Wireshark has an option to display packets as C Arrays. This made it pretty easy to get the Server Hello’s working in my Golang spoofing application.



By replaying these responses, slowly but steadily the fingerprint can be rebuilt.

```
if bytes.Contains(request, [] byte {     0x00, 0x8c, 0x1a, 0x1a, 0x00, 0x16, 0x00, 0x33,
0x00,     0x67, 0xc0, 0x9e, 0xc0, 0xa2, 0x00, 0x9e, 0x00, 0x39,     0x00, 0x6b, 0xc0,
0x9f, 0xc0, 0xa3, 0x00, 0x9f, 0x00,     0x45, 0x00, 0xbe, 0x00, 0x88, 0x00, 0xc4, 0x00,
0x9a,     ... ...  }) {     fmt.Println("replaying: tls12Forward")     conn.Write([]
byte {         0x16, 0x03, 0x03, 0x00, 0x5a, 0x02, 0x00, 0x00,         0x56, 0x03, 0x03,
0x17, 0xa6, 0xa3, 0x84, 0x80,         0x0b, 0xda, 0xbb, 0x3d, 0xe9, 0x3e, 0x92, 0x65,
0x9a, 0x68, 0x7d, 0x70, 0xda, 0x00, 0xe9, 0x7c,         ... ...     }) }
```

A full signature can be faked after implementing a reply for all ten different requests.



(Mis)usage of spoofed signatures

You're probably thinking: So what? What is the use of spoofed TLS fingerprints? They could be used by malicious actors to hide their applications when tools like JARM scanners are deployed to identify services in a network or on the internet. It can also be used for good. A honeypot replaying the fingerprint of a specific service can be used to setup a digital smokescreen for attackers.

Notes

jarmscan (jarm-go) is not a product of Salesforce. They've published a Python based JARM scanner implementation. Jarmscan (the scanner used here) is a Golang based implementation by