Iteration VI report

IU Project 2020 Spring

Daria Vaskovskaya

1 Intro

- **Student name**: Daria Vaskovskaya.
- **Topic**: **TODO**.
- **Supervisor**: Phillip Braun.
- **Iteration number**: 6.

2 Actual work done

So, turns out version of `mitmproxy` from the master branch contained only a part of ciphers from the OpenSSL spec. After multiple manipulations with the source codes, the problem is still not resolved.

2.1 Initial cipher fix

When we look at the request fingerprints from the previous report, we can see that there are multiple ciphers that are present in the original firefox request and are missing in the `mitmproxy` one – `1301,1303,1302`, to be precise.

When we look at the `mitmproxy` source code, in [1], we can see that these ciphers are not implemented in the map!

What I did to fix that (and possibly future problems with other browsers) was run an openssl command to extract the ciphers with their ids:

```
$ openssl ciphers -V ALL
0xC0,0x30 - ECDHE-RSA-AES256-GCM-SHA384
0xC0,0x2C - ECDHE-ECDSA-AES256-GCM-SHA384
0xC0,0x28 - ECDHE-RSA-AES256-SHA384
0xC0,0x24 - ECDHE-ECDSA-AES256-SHA384
0xC0,0x14 - ECDHE-RSA-AES256-SHA
```

Then, I have injected these into the source code, and the response fingerprint was starting to look a lot better, except for one small problem at the very end of the string – firefox had `0xA`, and mitmproxy had `0xFF`.

2.2 POODLE and renegotiation

When we look at the cipher specs, we can see that `0xA` is `DES-CBC3-SHA`, a legacy cipher that is vulnerable to the POODLE attack (padding oracle for SSL3 ciphers with CBC mode of operation), so that explains why mitmproxy discarded the cipher (the underlying library didn't want that); but the `0xFF` was not in the list of ciphers.

After further exploration, turns out that it represents something called `TLS_EMPTY_RENEGOTIATION_INFO_SCSV`, which, as far as I can get, is a solution to some attack described in [2], which is not really relevant to the topic. What is relevant though is that we need to somehow get rid of that, and it seems that downgrading the TLS library version used by `mitmproxy` would suffice, but I did not really have time for that.

Also, it seems that downgrading the library to some even lower version would probably also resolve the problem with `0xA`, before POODLE was discovered. A fun thing would be to dynamically load multiple TLS libraries based on some client-based heuristic (or deterministic test).

3 References to results

The cipher patch can be found at the github repo [3].

4 Current issues

Not enough time to finish the "library downgrade" described above.

References

[1] "Mitmproxy source code, cipher list," [Online]. Available: https://github.com/mitmproxy/mitmproxy/blob/7fdcbb09e6034ab1f76724965cfdf45f3d775129/mitmproxy/proxy/protocol/tls.py#L9%7D.

[2] "RFC 5764," [Online]. Available: https://tools.ietf.org/html/rfc5746.

[3] [Online]. Available: https://github.com/BananaAndBread/Project/blob/master/results/mitmproxy-ciphers.patch.