The Security Impact of HTTPS Interception

Network Security - Research Project

Authors:
DEMOLINIS Rémy
GASSINE Alan
QUETEL Grégor
THAY Jacky

Supervisor:
GOESSENS Mathieu

M1 Cybersecurity
University of Rennes 1

May 15, 2022

Contents

1 Introduction

1.1 The original paper

Our research takes its roots from a paper written in 2017 [4] by a team composed of Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J. Alex Halderman and Vern Paxson. One of the many focuses from their paper was the HTTPS Interception from antiviruses, the use of downgraded ciphers from their interception and improper behavior when checking certificate (when there was one) resulting in weakening the security of intercepted connection.

A summary of their findings can be found in section 4.1 figure 4.

At the time of their writings, some antiviruses studied would intercept incoming HTTPS transmission from a server to the client, decrypt it, check their content for potential attacks, and re-encrypt it back to the client using different ciphers from the one used by the server. Most of the time ciphers were weaker than the one the client originally advertise.

The insecure behavior of antivirus was not limited to improper use of weak ciphers, but could also sometimes be from the lack of even basic certificate validation or flawed validation of such [6].

This raises a few security and privacy concerns, using different ciphers from the server may create weaknesses that can be exploited if using outdated ciphers and may also make the user more traceable if the antivirus uses a set of ciphers that are specific to them.

1.2 Goals

The main goal of this project is to re-evaluate the security of antivirus four years after the disclosure of their bad implementation of the TLS protocol.

From this comparison, we can assume whether or not the original paper has had any impact on any antivirus software's security policy since its publishing.

All our resources (code sources, fingerprints, slides, report, etc.) can be found on our GitLab.

2 TLS Interception

2.1 TLS Handshake

When initiating a TLS connection, your browser sends a ClientHello with a set of parameters. Each browser or TLS client can choose the parameters it wants to use. Those parameters are ordered lists of ciphers, compression methods, signature algorithms and other extension types [5]. After the server receives this ClientHello it does an intersection between its own supported parameters and the one the client asked for to select the cipher suite, signature algorithm etc. that will be used for the exchange.

2.2 TLS Interception detection

Antiviruses intercepting TLS connection usually have their own TLS libraries, different from the ones used by your browser. A way to detect an interception would be to check for a mismatch between the fields in the ClientHello sent by the antivirus and the ones that are usually sent by legitimate client based on the user-agent header sent.

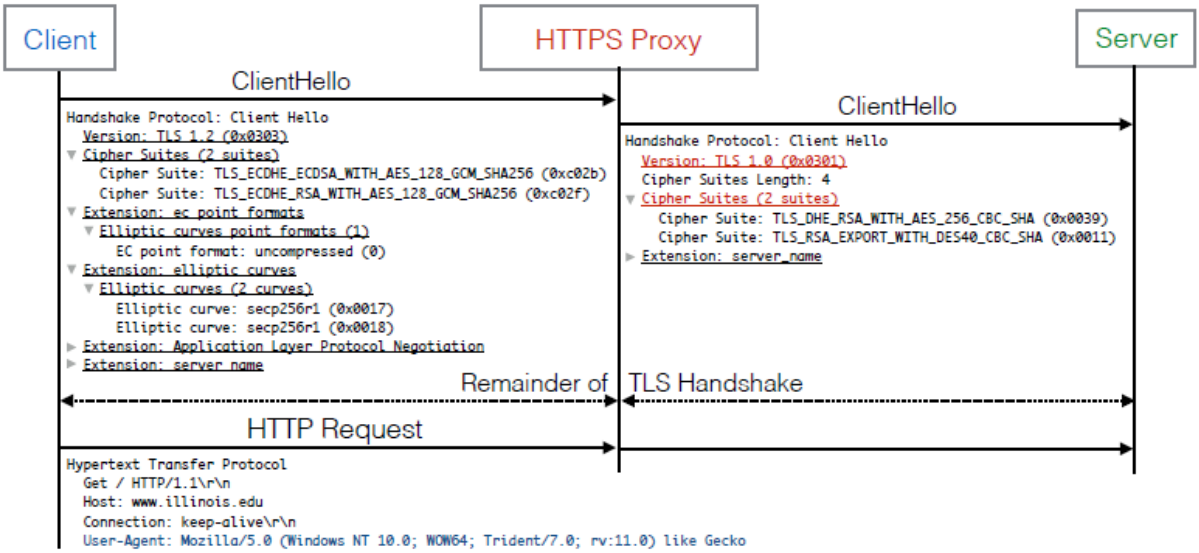

Figure 1: Mismatch between expected TLS parameters and received ones based on the user-agent

It can be interesting to notice that client-side, manually looking at the root certificate will tell you if an interception occurred. Indeed if your antivirus wants to transparently intercept your connection. It usually needs to add its self-signed CA certificate as trusted one in your browser store.

2.3 Fingerprinting

As said earlier, to be able to detect on the server-side an intercepted connection, we had to build fingerprints of the sent ClientHello and to compare them to fingerprints of known clients (Firefox, Chrome, Edge, antivirus).

We simply cannot make a fingerprint of the ClientHello since some of their fields are ephemeral. We thus need to choose which extensions to take into account. Based on the chosen fields of the paper 5, we took the following set of extensions:

- **Cipher suite**: Ordered list of ciphers that will be used to encrypt the exchange.
- **Client TLS**: Maximum TLS version supported by the client.
- **Compression methods**: Compression algorithm to use before encrypting the data, note that it is nowadays advised to disable compression since it exposes exchange to the CRIME vulnerability [7].
- **Elliptic curve point format**: Elliptic Curve algorithm that the client supports.
- **Signature algorithm**: List of signature/hash algorithm pairs to be used in digital signatures.
- **Supported groups:** Elliptic curve groups that the client prefers to use when proceed to key exchange.

3 The experiment

3.1 Operating System

All tests were made using Windows 10, none of our team members use a Macintosh computer, therefore any test with Mac OS from the original paper was skipped.

3.2 List of Antiviruses

We selected several antiviruses from the paper for our experiments by their popularity and the availability of a free version. We used their latest versions with the newest updates of Windows 10 as of May 4th, 2022.

- **Kaspersky Total Security**: Before the start of the Russian invasion of Ukraine on February 24th, 2022 and the blacklisting of Kaspersky's product in the United States of America by the Federal Communications Commission (FCC) for security concerns, Kaspersky was renowned for having its own team of researchers do their investigations on cybersecurity threats and other malware operations. Their antiviruses received several awards from PC Magazine, PC World or AV-TEST.
- **ESET Smart Security**: Since 1987, ESET has won several security solutions awards and has always placed into the most popular ones, especially for professionals. The company has a global sales network covering 180 countries, and regional offices in Bratislava, San Diego, Singapore and Buenos Aires.
- **BitDefender Total Security**: BitDefender ranks ninth globally among Microsoft Windows anti-malware application sellers as of May 2020 and BitDefender products are distributed through partners in over 150 countries.
- **Avast Antivirus**: Avast has 435 million monthly active users and has the second largest market share among anti-malware application vendors worldwide as of April 2020. Avast is a Czech antivirus software whose company's headquarter is in Prague, Czech Republic.
- **Avira Free Antivirus**: Avira now owned by Symantec Corporation since December 2020, one of the biggest players in cybersecurity, is an antivirus that received several awards and good review between 2008 to 2016 from the like of PC World or AV-Test.
- **AVG AntiVirus**: AVG is an antivirus made by AVG Technologies who merged with Avast Software s.r.o. in 2017. According to themselves, AVG boast an userbase of 200 million active users worldwide. It was selected as PC Magazine Editors' Choice in the free antivirus category back in 2012.

A brand-new computer with **McAfee Antivirus** installed for a free 30-day trial was procured by one of our team member. This antivirus was also tested and no interception was detected, hence its results will be skipped in section 4.

3.3 List of Tests

After selecting the antiviruses, we needed a set of tests to check for each of them, in order to evaluate their security. Like in the paper, the first thing that we thought about was the fingerprint and especially the cipher suites and the TLS version:

- We wanted to check for weak ciphers such as RC4, DES or 3DES. If an antivirus advertises one of those, there is a problem because they are full of vulnerabilities.
- Also, with the information contained in the fingerprint, we are able to detect if a client is likely vulnerable to some common attacks. For example, BEAST attack requires a TLS 1.1 with a CBC cipher.

The second thing that is interesting to check is the certificate of the web page. Actually, antiviruses can have different behaviors regarding different types of certificates. Below is the list that we chose and that we will detail more after:

- Valid certificate
- Valid for another domain only (invalid on this domain)
- Self-signed certificate
- Wildcard certificate

Finally, we chose a few browsers to run those tests on. In the paper, they used Internet Explorer, Firefox and Chrome so we did too. Also, we added Edge because it is the new version of Internet Explorer, and Opera because one of us had it and we thought: why not?

3.4 TLSlite-ng

To proceed with our tests, we decided to build a python server because of its ease of use; we can indeed build an HTTPS server very easily. We only needed to find a way to be able to interact with the TLS handshake. To do so, we used the tlslite-ng package [8], which implements SSL and TLS protocols. We had to modify the python classes to fit our needs. We indeed had to access the ClientHello after proceeding with the TLS handshake, choose the correct certificate and content to give to the Client based on the asked server name.

The library thus allowed us to parse and extract data from the ClientHello fields, and establish the handshake with client based on the asked parameters. It was very convenient for us to find such a library since TLS or SSL handshake can act in different ways depending on their version used and parameters.

3.5 Website: ns-rp.tk

In order to implement the tests that we previously talked about, we needed a website with a few subdomains for each test. Therefore, we took the free domain ns-rp.tk on Freenom, and added several DNS entries for our subdomains:

- The first one, sub1, whose role is to load a certificate which, at generation, was filled with incorrect information. In our case, we generated this certificate for google.com, which implies that if we load it on sub1.ns-rp.tk, it is supposed to be invalid since it does not contain the information of our website.
- The second one, sub2, loads a self-signed certificate. As it is not verified by a third party authority like Let's Encrypt for example, it is supposed to be considered insecure by the browser.
- Subdomain sub3 is a little bit different because it loads a wildcard certificate. Basically, a wildcard certificate is valid only for the subdomains of the domain it was generated for. In our case, we generated one for *.ns-rp.tk, which means that if we load the certificate for sub3, it is supposed to be valid. But what is the behavior regarding the subdomains of this valid subdomain? And how does an antivirus react to this specific situation? That is the reason why we created sub.sub3 and sub.sub.sub3.
- Finally, we had several valid subdomains from sub4 to sub10 that we created just in case, but weren't used in our tests.

Subdomains	RFC	Tests
Sub1	RFC 5246	Invalid Cert.
Sub 2	RFC 3280	Self-Signed Cert.
Sub 3	RFC 4592	Wildcard
Sub Sub3	RFC 4592	Wildcard
Sub Sub Sub3	RFC 4592	Wildcard
Sub 4	RFC 5246	Valid Cert
Sub 5	RFC 5246	Valid Cert.

Figure 2: Summary of our subdomains and their corresponding RFCs

Regarding the structure of the website, we implemented a few routes to easily obtain exploitable information about the clients' fingerprints:

- /, the main domain, which prints the client fingerprint, the ciphers contained in it that are not recommended by the IANA [5], as well as a table with the tests of the certificates on our subdomains
- /api/ returns a JSON with the raw client fingerprint

- /fp/ gives the 10 last fingerprints' SHAs
- /fp/details/ outputs the 10 last fingerprints in detail (raw JSONs)
- /fp/[1-10] prints the Nth last fingerprint
- /code/ redirects to our GitLab repository
- /slides/ downloads the slides of our final presentation

When someone visits our main page, if his fingerprint has not already been encountered before, it is stored in our local database we created for this purpose with MongoDB. All the /fp/* pages issue a query to this database in order to print the data detailed above.

4 Results

4.1 Interception

AV	Browser MITM					TLS Version	Remarks
	I. Explorer	Edge	Firefox	Chrome	Opera		
Kaspersky *(Total Security: 21.3.10.391)*	●	●	●	●	●	1.2	Mirrors Client Ciphers
ESET *(Smart Security: 15.1.12.0)*	●	●	●	●	●	1.2	Mirrors Client Ciphers
BitDefender *(Total Security: 26.0.14.65)*	●	●	●	●	●	1.2	Mirrors Client Ciphers
Avast *(Free Version: 22.4.6011)*	●	○	○	○	○	1.2	Mirrors Client Ciphers for Internet Explorer
Avira *(Free Version: 1.1.65.28718)*	○	○	○	○	○	1.2	N/A
AVG *(Free Version: 22.4.3231)*	●	○	○	○	○	1.2	Mirrors Client Ciphers for Internet Explorer

●	Interception
○	No Interception

Figure 3: Our array of interception, inspired of the one in the paper

Product	OS	Browser MITM				Grade	Validates Certificates	Modern Ciphers	TLS Version	Grading Notes
		IE	Chrome	Firefox	Safari					
Avast ...										
AV 11	Win	●	○	○		A*	✓	✓	1.2	Mirrors client ciphers
AV 11.7	Mac		●	●	●	F	✓	✓	1.2	Advertises DES
AVG ...										
Internet Security 2015–6	Win	●	●	○		C	✓	✓	1.2	Advertises RC4
Bitdefender ...										
Internet Security 2016	Win	●	●	●		C	✓	○	1.2	RC4, 768-bit D-H
Total Security Plus 2016	Win	●	●	●		C	✓	○	1.2	RC4, 768-bit D-H
AV Plus 2015–16	Win	●	●	●		C	✓	○	1.2	RC4, 768-bit D-H
Bullguard ...										
Internet Security 16	Win	●	●	●		A*	✓	✓	1.2	Mirrors client ciphers
Internet Security 15	Win	●	●	●		F	✓	✗	1.0	Advertises DES
CYBERsitter ...										
CYBERsitter 11	Win	●	●	●		F	✗	✗	1.2	No cert. validation, DES
Dr. Web ...										
Security Space 11	Win	●	●	●		C	✓	○	1.2	RC4, FREAK
Dr. Web 11 for OS X	Mac		●	●	●	F	✓	✗	1.0	Export ciphers, DES, RC2
ESET ...										
NOD32 AV 9	Win	●	●	●		F	○	○	1.2	Broken cert. validation
Kaspersky ...										
Internet Security 16	Win	●	●	●		C	✓	✓	1.2	CRIME vulnerability
Total Security 16	Win	●	●	●		C	✓	✓	1.2	CRIME vulnerability
Internet Security 16	Mac		●	●	●	C	✓	✓	1.2	768-bit D-H
KinderGate ...										
Parental Control 3	Win	●	●	●		F	○	✗	1.0	Broken cert. validation
Net Nanny ...										
Net Nanny 7	Win	●	●	●		F	✓	✓	1.2	Anonymous ciphers
Net Nanny 7	Mac		●	●	●	F	✓	✓	1.2	Anonymous ciphers
PC Pandora ...										
PC Pandora 7	Win	●	◐	◐		F	✗	✗	1.0	No certificate validation
Qustodio ...										
Parental Control 2015	Mac		●	●	●	F	✓	✓	1.2	Advertises DES

Interception:
○ No Interception (conn. allowed)
◐ Connections Blocked
● Connections Intercepted

Certificate Validation:
✗ No Validation
○ Broken Validation
✓ Correct Validation

Modern Ciphers:
✗ No Support
○ Non-preferred Support
✓ Preferred Support

Figure 4: The array of the original paper [4]

4.1.1 Our results

As you can see above (Figure 3), we made an array similar to the one in the original paper, to see the differences between both to check if the antiviruses improved or not.
To start, Kaspersky, ESET and BitDefender are doing interception on all tested browsers, but they do "*Mirrors Client Ciphers*"[1] this means that they copy the ciphers of the original certificate and they use them in their own certificates (see an example in appendix with BitDefender cipher suites on Firefox and the same one without antivirus, page 15), but they sometimes add a `TLS_EMPTY_RENEGOTIATION_INFO_SCSV` cipher (which is not really a cipher) which permits to avoid fails and prevents attacks, and it's possible that depending on the browser, they add some random ciphers at the beginning of the cipher list, but we ignored them.
To continue, Avast and AVG are doing interception on Internet Explorer only (also *Mirrors Client Ciphers*), but this is not very important because today it is not supported (by default) by recent Windows versions.
And finally Avira does not do any interception at all.
We can also see that now they all use TLS 1.2.

4.1.2 Comparison with the original paper

Compared to the original paper (Figure 4):

- ⇛ Kaspersky: Fixed its CRIME Vulnerability [7] (No compression methods in its fingerprints);
- ⇛ BitDefender: Stopped advertising RC4 ciphers;
- ⇛ ESET: Now uses TLS version 1.2 instead of 1.0 and has corrected the "Broken Certificate Validation" (we did not use NOD32 but apparently it seems to be included in the Smart Security pack[2]);
- ⇛ Avast: Is the same as before on Windows, only intercepts on Internet Explorer;
- ⇛ Avira: Also the same as before, it still does not do any interception (it was said in the paper but not in their array);
- ⇛ AVG: Stopped advertising RC4 ciphers too, and no longer does interception on Google Chrome.

The conclusion of those results is that globally they all improved and fixed their problems (except Avira which did not have those problems, and Avast).

[1] In order to see interceptions with "Mirrors Client Ciphers", we just looked directly the certificate itself, and we saw the names of the antiviruses.

[2] When we tried to install it the free license gave us Smart Security.

4.2 Tests on the website

4.2.1 Subdomains tests

Like we mentioned in section 3.3, we did some tests directly on our website when somebody connects to it. Here is the table of the results of those tests with all the antiviruses we tested (on Firefox):

Subdomains	RFC	Tests	Expected outcome	Reached
Sub1	RFC 5246	Invalid Cert.	NO	NO
Sub 2	RFC 3280	Self-Signed Cert.	NO	NO
Sub 3	RFC 4592	Wildcard	YES	YES
Sub Sub3	RFC 4592	Wildcard	NO	NO
Sub Sub Sub3	RFC 4592	Wildcard	NO	NO
Sub 4	RFC 5246	Valid Cert	YES	YES
Sub 5	RFC 5246	Valid Cert.	YES	YES

Figure 5: The table that you can find on the website

You can find in this array the tested subdomain, the RFC belonging to the test, the test itself (for more information cf. section 3.5), the expected behavior of a good functioning browser, and if the browser used actually reached the subdomain [3].

4.2.2 "Not recommended" ciphers

The last thing we implemented on our website is a part which shows the "not recommended" ciphers. But this does not mean that they are flawed, rather it indicates that the item either has not been through the IETF consensus process, has limited applicability, or is intended only for specific use cases.
The list of not recommended ciphers we used is from IANA [5].
Here is an example of what you can find on the website:


```
{
  "weak_ciphers": [
    "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
    "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
    "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA",
    "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA",
    "TLS_RSA_WITH_AES_128_GCM_SHA256",
    "TLS_RSA_WITH_AES_256_GCM_SHA384",
    "TLS_RSA_WITH_AES_128_CBC_SHA",
    "TLS_RSA_WITH_AES_256_CBC_SHA"
  ]
}
```


Figure 6: List of not recommended ciphers (on Firefox without any antivirus)

[3] Out of all the web browsers used, Firefox showed the best results.

5 Impact on security

Our results are unequivocal, the rising interest of everyone for security forced the antiviruses constructors to increase the level of security of their software. In 2017 [4], only 2 of the 21 tested antiviruses were mirroring client ciphers. Today, the 5 antiviruses that did interception that we tested were also mirroring ciphers, being the best practice they could have. Since all the browsers that we used had good practice in terms of TLS we were not able to see the behavior of antivirus in the case of a client that advertises weak ciphers or other TLS parameters, and thus seeing if the antivirus blindly copies the advertised ciphers of the client, or if there is a real wish from the software companies to increases the security of HTTPS traffic.

No improper certificate validation was noticed, even when talking about delicate cases such as the wildcard certificate. All the vulnerabilities detected in 2017 on these antiviruses got patched, but we do not know about more recent or complex vulnerabilities.

We also noticed that to avoid attacks where someone could dump the TLS traffic and wait for a key leak, all the browsers and consequently all the antiviruses we analysed preferred ciphers that would ensure perfect forward secrecy which would correspond with the findings of another paper [3]. For those kinds of ciphers, an unique session key is generated for every new session exchange.

We can safely declare that between those 4 years, security of antivirus interception has been increased. We're not announcing that it is now perfect, neither are we telling that HTTPS traffic interception is relevant for antiviruses, but the reveal of such bad practices from security vendors five years ago had an impact on how these companies deal with their users' security today.

6 Conclusion

From the results found in section 4, it is safe to assume mirroring ciphers has become the norm for antiviruses software since the publishing of the original paper from 2017 [4], it may have played a hand in changing their security policy.

However, HTTPS interception is still being done for some of the antiviruses studied, this may result in trust issues in regard to privacy concerns, even if the antivirus only checks the data intercepted for potential attacks, the possibility still remains for it to be read for other reasons, it all boils down to how much we trust the antiviruses' developers.

With more time and budget, more antiviruses could be studied and some other features could be added to our website. One of them would be the increase of the parameters harvested by our website to make a fingerprint. A more complex analysis of all the parameters of a browser such as Firefox could allow us to detect differences between an intercepted connection using Firefox user-agent and the actual cryptographic parameters the antivirus uses. Having more information could give us a hint on the antivirus from the client-side. We could also extend our tests to androids antiviruses, who may suffer from outdated TLS protocol [1].

The original paper covered a few middleboxes which we did not because of time restraint, it is important to check the evolution of their security policy and see if they have made any changes since the original paper [4].

References

[1] P. Gill A. Razaghpanah ; A. Akhavan Niaki ; N. Vallina-Rodriguez ; S. Sundaresan ; J. Amann. Studying TLS Usage in Android Apps, November 2017.

[2] ANSSI. Recommandations relatives à l'interconnexion d'un système d'information à internet (Section 4.3), June 19, 2020.

[3] P. Kotzias; A. Razaghpanah ; J. Amann ; K. G. Paterson ; N. Vallina-Rodriguez ; J. Caballero. Coming of Age: A Longitudinal Study of TLS Deployment, October 2018.

[4] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. A. Halderman, and V. Paxsonk. The Security Impact of HTTPS Interception, February - March 2017.

[5] IANA. IANA extension values, April 27, 2022.

[6] Tavis Ormandy. Kaspersky: SSL interception differentiates certificates with a 32bit hash, Nov 1 2016.

[7] Thai Rizzo, Juliano Duong. The CRIME attack, August 2008.

[8] tlsfuzzer. tlslite-ng, May 11 2022.

[9] Nikolai Tschacher. Incolumitas, June 19, 2020.

Appendix


```
{
  "tls_fingerprints": {
    "cipher_suites": [
      "TLS_AES_128_GCM_SHA256",
      "TLS_CHACHA20_POLY1305_SHA256",
      "TLS_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256",
      "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA",
      "TLS_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_RSA_WITH_AES_128_CBC_SHA",
      "TLS_RSA_WITH_AES_256_CBC_SHA"
    ],
    "client_version": "TLS 1.2",
    "signature_algorithm": [
      "ecdsa_secp256r1_sha256",
      "ecdsa_secp384r1_sha384",
      "ecdsa_secp521r1_sha512",
      "rsa_pss_rsae_sha256",
      "rsa_pss_rsae_sha384",
      "rsa_pss_rsae_sha512",
      "rsa_pkcs1_sha256",
      "rsa_pkcs1_sha384",
      "rsa_pkcs1_sha512",
      "ecdsa_sha1",
      "rsa_pkcs1_sha1"
    ]
  }
}
```


Figure 7: Cipher suites on Firefox: Without antivirus

```
{
  "tls_fingerprints": {
    "cipher_suites": [
      "TLS_AES_128_GCM_SHA256",
      "TLS_CHACHA20_POLY1305_SHA256",
      "TLS_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256",
      "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA",
      "TLS_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_RSA_WITH_AES_128_CBC_SHA",
      "TLS_RSA_WITH_AES_256_CBC_SHA"
    ],
    "client_version": "TLS 1.2",
    "signature_algorithm": [
      "ecdsa_secp256r1_sha256",
      "ecdsa_secp384r1_sha384",
      "rsa_pss_rsae_sha256",
      "rsa_pss_rsae_sha384",
      "rsa_pss_rsae_sha512",
      "rsa_pkcs1_sha256",
      "rsa_pkcs1_sha384",
      "rsa_pkcs1_sha512",
      "ecdsa_sha1",
      "rsa_pkcs1_sha1"
    ]
  }
}
```


Figure 8: Cipher suites on Firefox: With BitDefender