SSL/TLS Interception Challenge  
from the Shadow to the Light  

GIAC (GCIA) Gold Certification  

Author: Ngoc Huy Nguyen, nng.cybersecurity@protonmail.com  
Advisor: Rajat Ravinder Varuni  
Accepted: March 24th, 2019  

Abstract  

Secure Sockets Layer and Transport Layer Security (SSL/TLS) protocols are created to provide confidentiality for sensitive information exchange over the Internet. They can be used to protect privacy and confidentiality but can also be used to hide malicious activities. Organizations are currently facing traffic inspection challenges due to growing encrypted SSL/TLS traffic on the Internet. From criminal perspectives, attackers are moving more and more to encrypted traffic to hide their nefarious activities. Data exfiltration, malicious communication with Command and Control (C&C) and malicious downloads use SSL/TLS encrypted traffic. SSL/TLS interception is a double-edged sword that could be used to prevent and detect abnormal communications. This paper explains how organizations and security analysts can manage these challenges. It describes how to overcome them with advantages and drawbacks.
1. Introduction

Secure Sockets Layer and Transport Layer Security (SSL/TLS) protocols are created to provide confidentiality for sensitive information exchange over the Internet. They can be used to protect privacy and confidentiality but can also be used to hide malicious activities. Organizations are currently facing traffic inspection challenges due to growing encrypted SSL/TLS traffic on the Internet (Let’s Encrypt stats, 2018) and increasing the Internet network bandwidth (Akamai Research, 2017). From criminal perspectives, attackers are moving more and more to encrypted traffic to hide their nefarious activities. Cisco researchers found in their samples that the percentage of malware communication in TLS is increasing (Classifying Encrypted Traffic with TLS-Aware Telemetry, 2016). Data exfiltration, malicious communication with C&C and malicious downloads use SSL/TLS encrypted traffic. SSL/TLS interception is a double-edged sword that could be used to prevent and detect abnormal communications. This paper explains how organizations and security analysts can manage these challenges. It describes how to overcome them with advantages and drawbacks.

2. Background

SSL Protocol and TLS Protocol

Secure Sockets Layer (SSL) is a protocol created to secure communication over the Internet by providing privacy and reliability. The public SSL version 2.0 was released by Netscape in 1995 and was quickly updated in 1996 with SSL version 3.0 (IETF RFC 6101, 1995). In 1999, Transport Layer Security (TLS) protocol version 1.0 was released (IETF RFC 2246, 1999) and replaced SSL protocol version 3.0, although there were no major differences except an absence of interoperability between these two versions. In 2006, TLS protocol version 1.1 was released (IETF RFC 4346, 2006) and two years later TLS version 1.2 (IETF RFC 5246, 2008). In 2011, the Internet Engineering Task Force deprecated SSL version 2.0 (IETF RFC 6167, 2011) due to known vulnerability issues. In the same trend, in 2015, the IETF organization deprecated SSL version 3.0 (IETF RFC 7568, 2015) due to numerous attacks on the protocol and implementation of TLS which permitted negotiation with this vulnerable version. In October 2018, Apple, Google, Microsoft, and Mozilla made a public announcement to deprecate default activation of TLS 1.0 and 1.1 in their Internet browsers in early 2020 (Apple, Google, Microsoft & Mozilla, Deprecation of TLS 1.0 and 1.1, 2018). Recently, TLS protocol version 1.3 was released in August 2018 (IETF RFC 8446, 2018).

Ngoc Huy NGUYEN, nng.cybersecurity@protonmail.com
SSL 3.0 to TLS 1.2 handshake protocol

The aim of this paper is not to provide a complete study of SSL and TLS protocols. In order to help the security analyst understand the next chapters, we focus our background description on one layer of the protocol called SSL or TLS handshake. This protocol is used between the client and the server to authenticate each other, to negotiate a shared secret and exchange data in a reliable manner. SSL or TLS handshakes are negotiated on top of Transmission Control Protocol (TCP). It starts with an SSL/TLS Client Hello where the client provides to the server its supported protocol version, cryptographic algorithms (Cipher Suites), and other supported features. The server replies with an SSL/TLS Server Hello, selects the strongest compatible Cipher Suite and provides its certificate to the client. The client verifies the server certificate, sends a change cipher spec message to the server and finished message. The handshake is completed when the server sends its change cipher spec and finished messages; then the application communication between client and server could be started. In an optional case, the server could request that the client provide a certificate to be authenticated.

Even though the application data is encrypted, the handshake protocol on TLS version 1.2 and its previous versions are negotiated in clear text. With this property, the client parameters and the server certificate are transmitted in clear text and could be inspected by the security analyst. How this information can be exploited is shown in some of the following chapters.

Figure 1. SSL 3.0 to TLS 1.2 protocol handshake
**TLS Handshake 1.3**

This version update improves the performance by reducing exchange between client and server during TLS Handshake. Moreover, this version adds a zero round-trip time mode which improves performance at the sacrifice of security. As part of its major changes, this new version improves privacy by encrypting, in particular, the server certificate, and thereby reduces the possibility of inspection by the security analyst. Following the TCP handshake protocol, TLS version 1.3 handshake starts with an SSL/TLS Client Hello where the client provides to the server its supported protocol version, cryptographic algorithms (Cipher Suites), and other supported features. The server replies with an SSL/TLS Server Hello selects the strongest compatible Cipher Suite and encrypts its certificate and its extensions messages with secret keys derived from the client. The client verifies the server certificate and sends a finished message. The handshake is completed; then the application communication between client and server could be started. In an optional case, the server could request that the client provide a certificate to be authenticated.

---

**Figure 2. TLS 1.3 protocol handshake**
3. SSL/TLS protocol inspection

There are a number of defensive strategies to detect malware communication. Network traffic capture and protocol inspection are one of them. The advantage of this method resides in the fact that the defender could be stealthy from the attacker point of view. This passive method preserves also the privacy of user communication. The main drawback is that malicious activities through encrypted communication remain unknown from the security analyst. Malware could use TCP port 443 or other common web TCP ports to communicate with their C&C server because these ports are commonly opened from corporate infrastructure to untrusted networks. In addition, SSL/TLS could be used with an unusual TCP port, so protocol fingerprinting is one key to manage these different cases. However, it does not mean that SSL/TLS protocols are used in all cases. Of course, the malware developer could use, by convenience, clear text communication through these common ports. In this case, the security analyst could investigate the payload to try to understand the malware activities. However, even though it does not use SSL/TLS protocols, some malware could use their own encryption protocol to hide their exchanges, making it more difficult for the security analyst to uncover the real objectives of the attacker. In case of real usage of SSL/TLS protocols and even though these protocols are used to encrypt traffic, defenders can inspect clear exchange parts in these protocols. The further sections explain how defenders can use tools like Wireshark, Tshark and Zeek Bro in their investigation. This paper presents the following methods for inspection:

- SSL Client Hello extension server name
- SSL Client Hello Cipher Suites
- SSL Server Hello certificates

**Inspect SSL Client Hello extension server name**

One method of inspection is to check the extension server name on Client Hello Handshake. This inspection method could be useful to detect abnormal activities, however, the defender should be aware that the Client Hello server name extension could be crafted by the attacker to evade intrusion detection.

Server name extension is information that could be correlated with domain name request from the same client to detect incoherence.

In the Wireshark tool, the analyst can display the server name SSL extension by editing the column preferences and adding a custom field as shown in the following figure.
In the following figure, the capture displays all SSL Client Hello requests with the filter “SSL.handshake.type==1”. The server name information could be displayed for visual control of the security analyst.

When the captured file size cannot be handled by Wireshark, the analyst should use tshark command line:
tshark -n -r [capture file] -Y 'ssl.handshake.type==1' -T fields -e ip.src -e ip.dst -e \ssl.handshake.extensions_server_name

Figure 5. Tshark command line to find Handshake server name extension

Inspect server certificate

During the SSL handshake, the server presents its certificate. One method of inspection is to analyze the content of the certificate presented by the server. Wireshark provides a set of filters to perform extraction of certificate extensions (Wireshark Display Filter Reference: X.509 Certificate Extensions, 2019). The main objective for the security analyst is to extract interesting strings in certificate attributes that could be correlated with a threats database. Referring to the RFC 5280 (PKIX Certificate and CRL Profile, 2008), Internationalized Names “may be encountered in numerous certificate and CRL fields and extensions”. The following fields are useful to extract:

<table>
<thead>
<tr>
<th>X.509 Certificate fields</th>
<th>Wireshark / Tshark filter</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>dNSName</td>
<td>x509ce.dNSName</td>
<td>Extract Domain Name</td>
</tr>
<tr>
<td>uniformResourceIdentifier</td>
<td>x509ce.uniformResourceIdentifier</td>
<td>Extract URI</td>
</tr>
</tbody>
</table>

In the Wireshark tool, the security analyst can display the dNSName field entries to quickly extract domain names associated with server certificates. An example is provided in the following figure.
Figure 6. Wireshark - filter SSL extension x509ce.dNSName

With the following tshark command line the security analyst could export the top 30 list of attributes in dNSName format:

```
tshark -n -r [capture file] -Y ssl.handshake.type==11 -T fields -e x509ce.dNSName | tr -s , ' ' | sort | uniq -c | sort -rn | head -30
```

Figure 7. Tshark - export top 30 list of dNSName

The certificate could contain fields which include Uniform Resource Identifier (URI). In the Wireshark tool, the security analyst can display the uniformResourceIdentifier field entries to find out a pattern of an identified threat. The following figure shows the URI of certificate revocation lists. Note that the Certificate Revocation List (IETF RFC 5280, 2008) or Online Certificate Status Protocol (IETF RFC 6960, 2013) requests could also be captured and analyzed to give further information in the investigation.
With the following tshark command line the security analyst could export the list of attributes in uniformResourceIdentifier format:

```
tshark -n -r [capture file] -Y 'x509ce.uniformResourceIdentifier' -T fields -e x509ce.uniformResourceIdentifier | tr -s ',' '
' | sort -u
```

The certificate information could also be extracted with their string types (Wireshark Display Filter Reference: X.509 Selected Attribute Types, 2019)
NOTE: TeletexString, BMPString, and UniversalString are included for backward compatibility.

tshark -n -r [capture file] -Y ssl.handshake.type==11 -T fields -e x509sat.IA5String -e x509sat.uTF8String -e x509sat.PrintableString -e x509sat.TeletexString -e x509sat.BMPString -e x509sat.UniversalString | tr -s ',' '
' | sort -u

Figure 10. Tshark - Certificate strings extraction command line

Inspect Client Hello Cipher Suites

Another method of detection is to inspect client SSL Cipher Suites. Each SSL client has its own Cipher Suites list that could be used as a pattern to recognize software. Researchers from Masaryk University of Czech Republic discovered the possibility to correlate the Cipher Suites list and the fingerprint of software in the SSL/TLS network traffic (Husák, M., Cermák, M., Jirsík, T., & Čeleda, P., 2016). The main idea of this method of detection is to build a Cipher Suites list of trusted software and detect Cipher Suites list usage as a point of investigation. By comparing a trusted list with the list collected in the live detection, the security analyst could detect potentially unauthorized software communication in his or her network. Note that a software update could introduce a new Cipher Suites list.
Some Cipher Suites could be deprecated and removed from software after update. The security analyst should not rely exclusively on this method of detection because a Cipher Suites list could be forged on the client side. Moreover, malware could use the same Cipher Suites list as legitimate software to remain undetectable, or malicious activities could be done through legitimate software. In the following figure, the Wireshark capture displays Cipher Suites supported by client software.

![Wireshark - supported SSL Cipher Suites](image)

**Figure 11.** Wireshark - supported SSL Cipher Suites

The following tshark command provides a different pattern of Cipher Suites list used by the same host. To illustrate the figure below, the network capture was done on a host where TLS 1.2 connection was performed with Mozilla Firefox, Konqueror and openssl s_client software.

```
tshark -r [capture file] -Y ssl.handshake.type==1 -T fields -e ip.src -e ip.dst -e ssl.handshake.ciphersuite
```

**Figure 12.** tshark: extraction of Cipher Suites pattern by source IP address

With the following command line, we are able to build a signature sha1 database for each Cipher Suite.
SSL/TLS INTERCEPTION FROM THE SHADOW TO THE LIGHT | 12

In the following figure we can see, in red, the sha1 hash of the Cipher Suite, and, in blue, the associated Cipher Suite string list extracted with tshark command from SSL/TLS Client Hello. This kind of database could be combined with Zeek Bro intelligence framework to detect malware communication.

Figure 13. tshark & shell command line: build sha1 signature with client Cipher Suites

Use IDS Zeek (Bro)

Zeek, through its former name Bro, is a framework of Intrusion Detection System. The security analyst could use Zeek to extract valuable information from a network capture file. From the SSL/TLS traffic perspective, Zeek could simplify the job of information extraction with its SSL/TLS protocol analyzer (Using the Bro SSL analyzer, 2019). Zeek supports dynamic protocol detection for SSL/TLS protocols which means that it can detect SSL/TLS independently of port usage. To perform a manual processing for a network capture with Bro, the security analyst could use this command line:

```
bro -C -r [network capture file] local
```

After processing a network capture file with Zeek, the result presents a log file series in the folder where it was executed. In our case, the security analyst should focus on log file conn.log, ssl.log, x509.log, files.log and intel.log.
Log file | Description
---|---
conn.log | This log file provides connection information. The security analyst will be interested in tuple IP addresses, connection ports, and protocols.

ssl.log | This log file provides information about the SSL connection. The security analyst will be interested in the SSL/TLS version used, connection ports, Cipher Suites negotiated.

x509.log | This log file provides information certificate x509 analyzed. The security analyst will be interested in certificate attributes.

files.log | This log file provides information about files extracted from traffic. The security analyst will be interested in the file attributes.

intel.log | This log file provides information about pattern detected with the intellectual framework. The security analyst will be interested in the indicator and source of the intelligence information.

**Figure 14. Bro logs**

Note that the *uid* field could be useful to correlate events from these different files.

**Figure 15. Bro logs - list of connections**

**Check certificate with notary**

Researchers from International Computer Science Institute (ICSI) and University of California set up a certificate notary service to help the client to identify a malicious certificate through a third-party certificate database (Amann, Vallentin, Hall, & Sommer,
This database is built with Zeek Bro which was used to analyze SSL/TLS real traffic from the wild. Zeek Bro includes a script to check certificate notary. Also, the security analyst could activate a full certificate chain validation script to provide additional insight. To do so, the configuration files local.bro needs to be updated by uncommenting the following lines:

```plaintext
# This script enables SSL/TLS certificate validation.
@load protocols/ssl/validate-certs
# Uncomment the following line to check each SSL certificate hash against the ICSI
# certificate notary service; see http://notary.icsi.berkeley.edu.
@load protocols/ssl/notary
```

Zeek Bro certificate notary results are provided in the ssl.log file. The following fields help the security analyst in the evaluation of the malicious certificate:

- **notary.first_seen**: The date of first seen certificate on the notary service. The value is counted per days from 01/01/1970.
- **notary.last_seen**: The date of last seen certificate on the notary service. The value is counted per days from 01/01/1970.
- **notary.times_seen**: The days between the notary.last_seen and the notary.first_seen.
- **notary.valid**: The current validation status of the certificate from the Mozilla certificate root store. The value is a boolean, so it can be T (True) or F (False).

The following figure shows the result notary check and the certificate chain validation. The certificates with ok flag are validated.
**Identify malicious certificate with SSL blacklist**

SSLBL project (Sslbl.abuse.ch, 2019) provides to the security community a blacklist to identify malicious SSL certificates. These certificates are identified by their SHA-1 fingerprint. The security analyst must note that this method is signature based and is not efficient with a new fingerprint that is not in the database. The format which is provided by SSLBL needs to be converted to fit with Zeek Bro intelligence framework standard. A perl script is provided to perform this operation (Amann, 2019). To use the framework, the configuration file local.bro must be edited by adding the following lines:

```perl
redef Intel::read_files += { "/path_your_blacklist/sslblacklist.intel" };  
@load policy/frameworks/intel/seen  
@load policy/integration/collective-intel  
@load policy/frameworks/intel/do_notice
```

In the example below, an analysis was performed on network capture from a machine infected by Emotet with Trickbot malware (malware-traffic-analysis.net, 2019). The following command line provides useful information for the security analyst.

```bash
cat ssl.log | bro-cut uid id.orig_h id.orig_p id.resp_h id.resp_p cert_chain_fuids issuer
```

**Figure 17. Wireshark - supported SSL Cipher Suites**

One server X509 certificate sha1 signature is listed on SSL abuse database then it was detected by the intelligence framework as seen in the intel.log file.
Figure 18. Detection from Zeek Bro intelligence framework. Event log in intel.log

The security analyst could verify manually in the files.log which reference files were captured in the traffic. The figure below shows that sha1 hash value is effectively found in the SSL abuse database export file as TrickBot C&C.

cat files.log | bro-cut source sha1 | grep SSL | grep -v ^ | sort -u | awk '{printf $2"\n"}' | xargs -I {} grep {} sslblacklist.csv --color

Figure 19. Extraction of SSL sha1 hash and detection with SSL abuse database

In a machine infected with another variant of Emotet with Gootkit (malware-traffic-analysis.net, 2019), the Zeek Bro ssl.log file shows the malware trying to connect to the same destination IP address, however, the server certificate changed at each new SSL/TLS connection. These connections were not detected with the SSL abuse database at the moment when this sample was tested (sslabuse file updated from 2019-02-06 17:32:42 UTC).

cat ssl.log | bro-cut ts id.orig_h id.orig_p id.resp_h id.resp_p server_name issuer

Figure 20. Emotet with Gootkit uses different X509 certificate
4. SSL/TLS interception

SSL/TLS interception is a controversial method to decrypt SSL communication. When it is implemented in a corporate environment, if there is no enlightened policy, in some cases it may conduct the company to privacy and security concerns. The main advantage of this method is the visibility provided to the security analyst. SSL/TLS decryption permits analysis of the communication payload.

The main drawback of this method is the modification of the end to end communication which could be detected by the attacker. Sophisticated malware can also detect that a server certificate is not issued by its own trusted Certificate Authority. Besides, the interception method cannot work in case of bidirectional SSL certificate authentication between the client malware and its command & control server. Moreover, the malware developer could add obfuscation or encryption in data payload to prevent reverse-engineering. From the offensive perspective, SSL/TLS interception solutions in corporate companies are interesting targets to be reached by attackers. Sensitive information could be retrieved on these points of interception. Defenders should be aware of the risk of this solution which could turn against them.

**Interception architecture model**

There are several concepts of architecture to intercept SSL/TLS traffic; one of the methods is to position a network interception appliance in the network path to the Internet. This equipment could be a dedicated interception box, but also could be a firewall with SSL/TLS interception capability. The following figure shows the architecture concept.

---

**Figure 21.** Undetected certificate sha1 signature in the SSL abuse database
The main advantage with this design is that interception is done with a single physical zone that could not be bypassed by clients coming from the internal zone because it is a transparent deployment from the client point of view. But, if there are multiple exit points to the Internet, the SSL/TLS interception solution needs to be replicated on each Internet exit point, otherwise, the security team would have only partial visibility on encrypted traffic. On the other hand, multiple interception points could drive to complex infrastructure management which is finally counterproductive. With this architecture, the more Internet bandwidth and SSL/TLS traffic grows in organizations, the more SSL/TLS interception requires to compute resources in a single point. This means that organizations will reach, at some point, a computer limitation due to hardware and software capacity limit in the physical interception zone.

Interception SSL/TLS could also be done with an explicit method by integrating this feature in filtering infrastructure functions like proxies, firewalls or intrusion prevention systems. Facing the growing number of SSL/TLS interception components in the network path to the Internet, the performance latency is an issue that organizations should take into account (Farrell, 2017).

Finally, another distributive method to intercept SSL/TLS traffic is to perform this function directly on the endpoints with a security software agent (Antivirus, Proxy agent). The main advantage of this technique is offloading directly on endpoints the interception SSL/TLS, but there is a risk of this function being hijacked by malware or simply being deactivated for malicious purposes.

**SSL/TLS Interception with proxy**

SSL/TLS interception has a significant impact on the security level of the communication by modifying properties of the initial SSL/TLS handshake as well as protocol version, Cipher Suites, and extensions. To verify this assumption an experimental lab was built and a network capture was done on internal and external interfaces of a proxy squid.
with ssl bump feature activated (wiki.squid-cache.org, Intercept HTTPS CONNECT messages with SSL-Bump, 2019). This lab was built on a Linux Ubuntu server version 18.0.15 x86_64 with openssl version 1.1.1. The squid proxy is installed in version 4.1 and required compilation with openssl options.

According to our results, this assumption was confirmed in the figure below; the client negotiates a handshake in TLS version 1.2 with the squid proxy, and the squid proxy negotiates with the web server www.sans.org in TLS version 1.3. Fortunately, in this experimentation, the proxy increased the security level by using the last version of the protocol available which is not the case for the client. Unfortunately, in most cases, SSL/TLS interception decreases the security level of the communication by using weak algorithms or introducing critical vulnerabilities (Durumeric, Ma, Springall, Barnes, Sullivan, Bursztein, Bailey, Halderman, & Paxson, 2017). The security analyst team should monitor available openssl libraries and vulnerabilities of their interception solutions if they do not want to introduce security risks for their organization.

![Figure 23. Wireshark flow - Proxy SSL interception capture](image)

Moreover, our experimentation with some malware has shown that malware could sometimes bypass the proxy configuration of the system and use a direct connection to the Internet.

### 5. Malware lab traffic analysis

**Testing protocols**

Malware traffic analysis is a dynamic analysis process which requires execution and communication of the malware with attacker infrastructure. The testing protocol should be executed in a repeatable manner to provide comparable results. The malware inoculation is also an important step that should be documented to facilitate the event correlation between
the execution step and the network traffic. The prerequisite before performing the test protocol is to build a set of virtual machines which represents the machines of the protected environment.

The figure below explains the global methodology to collect the malware traffic network capture file. Because malware could alter the virtual machine properties, we recommend starting a snapshot version of the virtual machine at each test. The security analyst should keep in mind that malware can detect the virtual environment and change their execution path to stay undetected.

**Figure 24. Malware network traffic lab testing protocol**

**Processing information**

According to techniques of SSL/TLS inspection and interception explained in the previous chapters, the following figure modelizes the processing information flow for our malware traffic analysis framework.

**Figure 25. Malware lab processing information concept**
Lab infrastructure high level design

In order to provide an on-demand malware analysis infrastructure to the security community, we have developed a platform model in an Amazon Elastic Cloud Compute (EC2) infrastructure service to perform our malware traffic analysis. To compare with an on-premise virtual lab infrastructure concept with virtualbox (Equiarga, Garcia, & Garino, 2017), one advantage of our solution is the scalability of the compute resources and the network bandwidth for the Internet connection. Another advantage is staying anonymous as long as possible from attackers by avoiding exposure of the public IP address of the security analyst’s organization. Besides, the security analysts could transpose this concept with another cloud provider. The main concept of this architecture is presented in the figure below. This lab is composed of five security zones; the first is the infection zone where samples of malware are played in Amazon instances. The second is the NAT capture zone where network traffic is captured for further analysis; the third zone is the analysis instance where network capture files and logs are processed in order to provide information to the security analyst; the fourth zone is the proxy zone where there is a proxy SSL/TLS intercepting proxied traffic; the fifth zone is an admin zone where administration is performed remotely. Before executing malware in a test environment, the analyst should consult the acceptable use policy of the cloud provider.
Cloud-based traffic malware analysis proof of concept

In our proof of concept, AWS EC2 service is used to provide in a fast and simple manner the virtual machines and public IP addresses needed for the experimentation. If the allocated public IP address of NAT instance is blocked by the C&C server, the security analyst could release and allocate a new public IP address within a few seconds (cf. Figure 27).

Figure 26. High-level design - AWS Malware analysis platform

Figure 27. Amazon EC2 Elastic IPs allocation

In order to control and to isolate the communications of the infected machines in the lab infrastructure, the routing table of the infected subnet is forced to the NAT instance (docs.aws.amazon.com, VPC NAT Instance, 2019). Moreover, all security zones are filtered from each other with a dedicated security group policy. For the “infected zone” an additional network access control list is added to allow or deny traffic to the Internet during the testing phase. The following figure shows a successful packet capture from the NAT instance of an infected Windows host. The network capture could be collected and be analyzed in the analysis instance server.

Figure 28. NAT Instance network capture of an infected host

As part of our contribution for the security community, we have developed a set of technical procedures to configure key components for malware test and traffic analysis in a cloud-based Amazon EC2 environment (Nguyen, 2019).

Future work

For our future work, we will develop and improve our project with the main goal of saving time for the security analyst community. Our first direction to reach this objective is...
the automation and the versioning management of the cloud infrastructure setup. The second direction is the automation of the security analysis features included in our lab infrastructure. The third is the capability to easily add a new security analysis feature with a modular approach.
6. Inspection and interception comparison

Following the different techniques of SSL/TLS analysis described in the previous chapters, the table below summarizes the advantages and drawbacks for each method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Advantages</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspect client hello server name extension</td>
<td>Passive method and the result could be correlated with the DNS queries to verify coherence.</td>
<td>Client hello server name extension could be forged by the malware.</td>
</tr>
<tr>
<td>Inspect SSL/TLS Cipher Suites</td>
<td>Passive method and the result could be correlated with known Cipher Suites of trusted software to detect abnormal communication.</td>
<td>False negative detection could occur if the malware use trusted Cipher Suites.</td>
</tr>
<tr>
<td>Inspect certificate strings attributes</td>
<td>Passive method and the certificate strings could be correlated with the known indicator of compromise.</td>
<td>Certificate string attributes could be forged by the attacker to evade detection.</td>
</tr>
<tr>
<td>Inspect the certificate validity with a notary certificate service</td>
<td>Passive method to check certificate validity with third-party referential.</td>
<td>The notary service builds its database based on observed traffic which is not an exhaustive method. This method requires DNS queries and could be visible by an attacker who can capture the local network traffic.</td>
</tr>
<tr>
<td>Inspect the certificate fingerprint with an SSLBL database</td>
<td>Passive method and the result could be correlated with threat intelligence database.</td>
<td>The attacker could forge new certificates on C&amp;C servers which are not in the SSLBL database.</td>
</tr>
<tr>
<td>Intercept the SSL/TLS traffic on the endpoint client</td>
<td>An active method which provides visibility on the data payload which is encapsulated in SSL/TLS protocols. Consumes resources only on the endpoint client.</td>
<td>The malware could detect the interception and change its execution. It could disable the SSL/TLS interception module by exploiting a vulnerability on the endpoint client.</td>
</tr>
<tr>
<td>Intercept the SSL/TLS traffic on the network</td>
<td>An active method which provides visibility on the data payload which is encapsulated in SSL/TLS protocols. Could not be disabled on the endpoint.</td>
<td>The attacker could detect the interception. This method could decrease the security level of the SSL/TLS communication. Consumes resources on a single point of the network.</td>
</tr>
</tbody>
</table>
7. Conclusion

As discussed in the introduction, malware hides their activities by moving to encrypted communication with protocols like SSL/TLS. Due to the increasing adoption in industries of SSL/TLS inspection and interception solutions, attackers have begun to adapt their strategies to slow down network forensics analysis. Security researchers of UNCuoyo and CTU University discovered that SSL blocking and interception measures could modify malware behavior and future malware communication features (Equiarga, Garcia, & Garino, 2017). The question of SSL/TLS interception obsolescence and its efficiency as a defensive strategy arises. Abandoning SSL/TLS interception is also not a solution. Hopefully, our paper has shown that the defenders still have passive methods to collect SSL/TLS artifacts and discover malicious activity, but for how much longer with the adoption of TLS 1.3 on the Internet? This new version introduces more privacy for network communication and reduces capabilities for the defenders to inspect TLS Handshake. We can make an assumption that covert channel or hiding information could be another strategy that attackers would adapt to operate under the radar of the blue teams and circumvent detection with SSL/TLS interception. These strategies could be the next challenges of our security analysts. To illustrate this theory, according to security researchers (Cabaj, Caviglione, Mazurczyk, Wendzel, Woodward, & Zander, 2018) modern malware uses hiding or obfuscating techniques to communicate with their Command & Control server. In addition, a security researcher has demonstrated with proof of concept, that covert channel could be operated by abusing certificate extension X.509 (Reaves, 2018). What could happen if the attackers use the covert channel with TLS 1.3?

Facing these evolutions from the defenders’ as well as the attackers’ sides, we know that malware and botnets still need to communicate with their Command & Control server in one way or another. Even though communications are encrypted or obfuscated with known or unknown protocols, network intrusion analysts should keep in mind their foundation to find malicious network activities. Knowing the normal state of the network traffic, observing duration, the frequency of communication, data volume exchanged, and knowing the standard working principle of protocols are the best techniques to highlight malicious activities in a network. Finally, the question of how to perform the analysis in a controlled environment led us to build a cloud-based infrastructure concept as a contribution for the security community.
References


[https://www.icir.org/johanna/papers/icsi12extractingcertificates.pdf](https://www.icir.org/johanna/papers/icsi12extractingcertificates.pdf)


[https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=449962](https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=449962)

Apple, Google, Microsoft, & Mozilla. (2018). *Public announcement deprecation of TLS version*


Ngoc Huy NGUYEN, nng.cybersecurity@protonmail.com

docs.aws.amazon.com (2019). *VPC NAT Instance*

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_NAT_Instance.html


Equiarga, M. J., Garcia, S., & Garino, C. G. (2017). *Observer effect: How intercepting HTTPS traffic forces malwares to change their behavior*

http://sedici.unlp.edu.ar/handle/10915/63935


https://www.sans.org/reading-room/whitepapers/dlp/rock-deployment-strategies-tls-decryption-38240

Husák, M., Cermák, M., Jirsík, T., & Čeleda, P. (2016). *HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting*


IETF RFC 822 (1982). *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES*


IETF RFC 5280 (2008). *PKIX Certificate and CRL Profile*


IETF RFC 6176 (2011). *Prohibiting Secure Sockets Layer (SSL) Version 2.0*


Let’s Encrypt stats (2018). *Percentage of Web Pages Loaded by Firefox Using HTTPS.*

Retrieved 16 December, 2018, from Let’s Encrypt:

https://letsencrypt.org/stats/

malware-traffic-analysis.net (2019). *EMOTET INFECTION WITH GOOTKIT*

https://malware-traffic-analysis.net/2019/01/14/index.html

malware-traffic-analysis.net (2019). *EMOTET INFECTION WITH TRICKBOT*

https://malware-traffic-analysis.net/2019/01/22/index2.html

Nguyen, N. (2019). *Cloud-based malware traffic analysis lab*

Retrieved from:

https://github.com/nng-cybersecurity/cloud-based-malware-traffic-analysis

Ngoc Huy NGUYEN, nng.cybersecurity@protonmail.com
Radichel, T. (2017). *Packet capture on AWS*

https://www.sans.org/reading-room/whitepapers/detection/packet-capture-aws-37905

Reaves, J. (2018). *Covert Channel by Abusing X509 Extensions*

http://vixra.org/abs/1801.0016

Sssl.bl.abuse.ch (2019). *Malicious SSL certificates identified by SSLBL*

https://ssl.bl.abuse.ch/ssl-certificates/

wiki.squid-cache.org (2019). *Intercept HTTPS CONNECT messages with SSL-Bump*

https://wiki.squid-cache.org/ConfigExamples/Intercept/SslBumpExplicit


https://www.wireshark.org/docs/dfref/x/x509ce.html


https://www.wireshark.org/docs/dfref/x/x509sat.html

Zeek.org (2019). *Intelligence Framework*


Zeek.org (2019). *Using the Bro SSL analyzer*


Zeek.org (2019). *Protocol Analyzer : Bro-SSL*

## Upcoming Training

<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Security West 2020</td>
<td>San Diego, CA</td>
<td>May 11, 2020 - May 16, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Live Online - SEC503: Intrusion Detection In-Depth</td>
<td>United Arab Emirates</td>
<td>May 28, 2020 - Jul 06, 2020</td>
<td>vLive</td>
</tr>
<tr>
<td>SANSFIRE 2020</td>
<td>DC</td>
<td>Jun 13, 2020 - Jun 20, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Blue Team vs Purple Team 2020</td>
<td>United Arab Emirates</td>
<td>Jul 13, 2020 - Jul 18, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Rocky Mountain Summer 2020</td>
<td>CO</td>
<td>Jul 20, 2020 - Jul 25, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Live Online - SEC503: Intrusion Detection In-Depth</td>
<td>United Arab Emirates</td>
<td>Jul 29, 2020 - Sep 04, 2020</td>
<td>vLive</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>Aug 3 ET</td>
<td>MA</td>
<td>Aug 03, 2020 - Aug 08, 2020</td>
</tr>
<tr>
<td>SANS vLive - SEC503: Intrusion Detection In-Depth</td>
<td>SEC503 - 202008,</td>
<td>Aug 10, 2020 - Sep 16, 2020</td>
<td>vLive</td>
</tr>
<tr>
<td>SANS Essentials Live Online 2020</td>
<td>Australia</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Baltimore Fall 2020</td>
<td>Baltimore, MD</td>
<td>Sep 08, 2020 - Sep 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Network Security 2020</td>
<td>Las Vegas, NV</td>
<td>Sep 20, 2020 - Sep 27, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Online</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
<tr>
<td>SANS SelfStudy</td>
<td>Books &amp; MP3s Only</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>