# Indicators of Malicious SSL Connections

Riccardo Bortolameotti[1], Andreas Peter[1], Maarten H. Everts[1,2], and Damiano Bolzoni[1,3]

[1] University of Twente, Enschede, NL
{r.bortolameotti,a.peter}@utwente.nl
[2] Netherlands Organisation for Applied Scientific Research (TNO), Groningen, NL
maarten.everts@tno.nl
[3] SecurityMatters, Eindhoven, NL
damiano.bolzoni@secmatters.com

**Abstract.** Internet applications use SSL to provide data confidentiality to communicating entities. The use of encryption in SSL makes it impossible to distinguish between benign and malicious connections as the content cannot be inspected. Therefore, we propose and evaluate a set of indicators for malicious SSL connections, which is based on the unencrypted part of SSL (i.e., the SSL handshake protocol). We provide strong evidence for the strength of our indicators to identify malicious connections by cross-checking on blacklists from professional services. Besides the confirmation of prior research results through our indicators, we also found indications for a potential (not yet blacklisted) botnet on SSL. We consider the analysis of such SSL threats as highly relevant and hope that our findings stimulate the research community to further study this direction.

**Keywords:** SSL, Malicious Connection Indicators, Handshake Analysis

## 1 Introduction

The Transport Layer Security (TLS) and its predecessor the Secure Socket Layer (SSL) are the *de-facto* standard protocols for secure communication over the Internet.[1] They provide end-to-end security that guarantees data confidentiality, integrity and authenticity to the communicating entities. In particular, they provide protection against possible active Man-in-the-Middle (MitM) attacks, where the attacker has the control over the entire network. Most of the services that handle sensitive data use SSL to protect the confidentiality of their users' data. In the past years, many papers have been published on SSL that assess its reliability and identify potential vulnerabilities. Some general analyses on characteristics of SSL traffic have shown several practical problems related to its infrastructure. Holz et al. [1] report issues regarding the usage of X.509 certificates: errors within certificate chains, absence of certificate subjects, common

[1] For the remaining of the paper, we refer to both as SSL

usage of expired certificates, etc. Amman et al. [2] have highlighted the complexity of the entire SSL infrastructure by stating that many specifications are left to interpretation while features aiming to improve weaknesses are still poorly implemented. Other works instead focused more on the security vulnerabilities of SSL. For instance, Amman et al. [3] analyze SSL traffic to understand the trust relationships among Certificate Authorities (CA) and to detect transparent MitM attacks. In such attacks, the attacker tries to compromise CAs with the goal of being able to generate a valid certificate for any domain and to start a MitM attack. The authors conclude their work stating that the certificate structure does not give enough information to be able to distinguish between malicious and benign certificates. Georgiev et al. [4] present a security analysis on SSL library implementations for non-browser software, identifying several vulnerabilities that make many applications vulnerable to MitM attacks. In [5], Fahl et al. introduce MalloDroid, a tool for Android apps used to detect SSL implementation vulnerabilities to MitM attacks, identifying more than 1000 potentially vulnerable apps. Although the SSL protocol (its most recent version, TLS 1.2) is considered to be secure from a theoretical perspective, it still shows several practical issues. In response to such problems, researchers started to propose security enhancements, which are not yet widely implemented in current applications [10]. Conti et al. developed MITHYS [6], a proxy for Android applications that addresses the SSL vulnerabilities examined in [5] and [4], and that guarantees MitM protection against rogue access points. Bates et al. [8] propose CERTSHIM, a lightweight retrofit that patches SSL implementations against several SSL vulnerabilities, including those highlighted in [4]. Holz et al. [9] suggest Crossbear, a system that detects MitM attacks on SSL/TLS over the Internet, collecting data in a centralized system from several online probes, which works on browsers.

In contrast to all existing works, we investigate SSL with a focus on malicious connections. Previous works have analyzed SSL connections assuming the client is benign. We define a connection as malicious when both end points of the communication are controlled by the attacker. A possible scenario is data exfiltration, where a compromised machine communicates with an external server, owned by the attacker, over an SSL channel in order to bypass security measures and to camouflage within normal traffic. Botnets are an example of such a scenario.

Our main contributions are the following: (1) we present an initial study on malicious connections within SSL network traffic, by looking at the SSL handshake protocol, (2) we found good indicators for malicious connections using unencrypted information exchanged during the SSL handshake, (3) we verify prior research findings [1, 3, 4, 10] with our newly found indicators, and (4) we discovered the presence of malicious connections examining our indicators within the network traffic of an international university and of an international financial corporation. Within the IT infrastructure of the university we found 34 connections that show the same communication patterns (e.g., expired certificates), where 10 of the associated IP addresses are blacklisted by a professional service, called ThreatStop [16]. Furthermore, analyzing the network traffic of the financial corporation we found two other malicious connections, also blacklisted. Moreover, we found one of these analyzed malicious connections (i.e., the IP address of the server) way before ThreatStop itself and four of them are marked as potential botnets. We consider this a significant result as it shows the strength of our indicators. We hope that our work stimulates the research community to further study these new findings.

## 2 Our Approach and Assumptions

Our goal is to identify a set of features based on the SSL handshake protocol that could indicate the presence of malicious connections within SSL traffic. In our approach, we first select a set of features to analyze within SSL traffic that we think can help us to indicate connection misbehaviors. After an evaluation of this set of features on real data, we identify those that are more promising as maliciousness indicators. Our approach is based on two assumptions:

1. The encrypted part of the SSL protocol is assumed to be secure, meaning that we cannot inspect it.
2. Malware authors have complete control over the client and server applications, therefore they can easily avoid following the SSL standards, and make their "own rules" creating broken SSL connections (e.g., do not properly authenticate application connections).

The first assumption has two positive side-effects: whatever analysis we do, it will (1) respect data confidentiality, and (2) be lightweight as we only focus on the initialization of the SSL connection (i.e., in the SSL handshake) and because we do not have to use complex algorithm to analyze our features. The negative side-effect of assumption (1) is that it is not possible to examine the content of the payload message in order to verify the maliciousness of a connection. The second assumption implies an enforcement of authentication checks on SSL connections at network level. This is done because browsers do not check the validity of SSL connections generated by applications running on the background. The drawback of the second assumption is that malicious connections would not be identified whenever they follow correctly the specifications of the protocol.

Selecting features from the handshake protocol is not a novel approach. In 2014, Pukkawanna et al. [7] proposed different classifiers to automatically assess the security of SSL servers, analyzing handshake protocol features. However, their work focuses only on the security parameters of the server and not on the parameters of the client. The authors use information from the *Server Hello* and *Certificate* messages of the protocol. To achieve our goal, we consider also characteristics of the *Client Hello* message (e.g., *server name*) and we relate them with those from other messages (e.g., *fourth feature* in Section 2) in order to evaluate the behaviour of both communicating parties.

**Selected Set of Features.** The *first feature* that we have selected is the validity of the X.509 certificates. With this feature, we want to check if the certificate is valid, self-signed, revocated, etc. The validity of the certificate can help us to detect misbehaviors, because for normal benign traffic we do not expect to see expired certificates during the authentication phase, neither we expect to see `facebook.com` to use a self-signed certificate. This feature is commonly used by researchers when analyzing the security of SSL. For instance, self-signed certificates are used in [6] to identify vulnerable applications to MitM attacks. However, in our case we do not restrict our attention to self-signed certificates, because malicious connections can also be authenticated with expired, valid or revocated certificates.

Our *second feature* is the release date of the certificate, especially for self-signed certificates. This feature is appropriate in the context of malicious software, where the lifetime of web domains is short. Therefore, we assume that criminals could generate new self-signed certificates either for each connection or for a short period of time (e.g., one day). We focus on self-signed certificates because they are easy and cheap (i.e., free) to generate and seem more suitable considering the lifetime of domains, unlike expensive commercial certificates.

Our *third feature* is the existence of mutual-authentication. SSL provides the option for a server to require client authentication (e.g., *CertificateRequest* and *CertificateVerify* messages). This feature can be leveraged by criminals in the context of a peer-to-peer botnet, in order to avoid external peers to infiltrate within their system.

Our *fourth feature* is the relation between the SSL extension *server name*, which is included in the *Client Hello* message, and the subjects (i.e., *Subject* and *subjectAltName* X.509 certificate fields) of the X.509 certificate. This is the typical browser authentication check that verifies whether the certificate is valid for the domain requested by the client or not. Whenever there is a mismatch, the connection should be considered untrusted, and potentially vulnerable to MitM attacks (e.g., in the case of DNS poisoning [4], where the attacker can redirect a user from a website to another).

Our *fifth feature* is the Levenshtein distance between the *server name* and a list of the 100 most visited websites, whenever a self-signed certificate is encountered during the handshake. When a user connects to a server (e.g., `www.google.com`) he should expect to receive a valid certificate, and not a self-signed certificate valid for a similar domain (e.g., `www.gogle.com`), otherwise it could be a symptom of a MitM attack.

The *sixth feature* is the structure of the *server name* string. A current trend in botnets is to use Domain Generation Algorithms (DGAs)[11] to generate many random domain names that can be exploited as rendezvous point with botnet servers. Therefore, we want to check whether these strings can be identified as random-looking domains or not. This feature is not new in literature and it has been previously used in the context of HTTP [14].

Finally, our *seventh feature* is the format of the *server name* string, which should have a DNS hostname format as described in the specifications of the

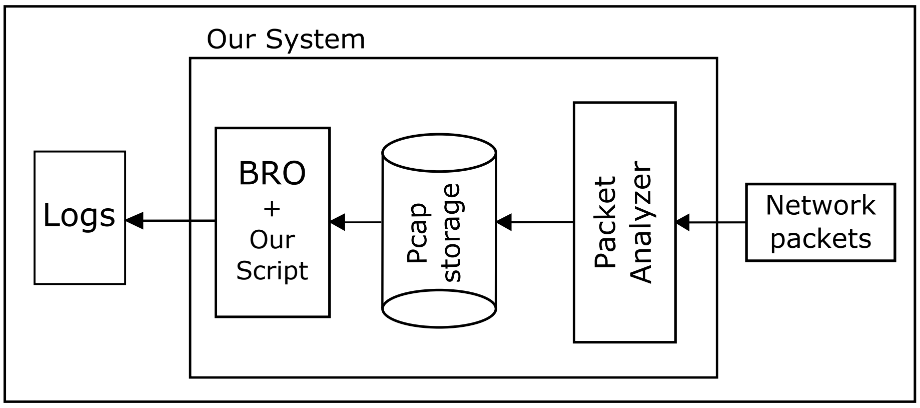

**Fig. 1.** System Architecture

SSL extensions [15]. This protocol field represents the domain the client wants to connect to, therefore we do not expect to see weird values or strings that could represent an exchange of messages, perhaps used by criminals as commands.

A summarized description of the features is shown in Table 1.

## 3 Architecture and Implementation

The architecture of our system is shown in Figure 1. The system takes as input the network traffic and first filters it through a packet analyzer module that recognizes the SSL traffic and stores it on the disk as pcap (packet capture) files. Those files are then given as input to Bro [12], an open source network analysis framework that we use to analyze the traffic. This analysis on SSL connections is based on our own Bro scripts that implement the aforementioned set of features. Once Bro has analyzed the whole traffic, it outputs a set of log containing the connections that, according to our features, might indicate the presence of malicious behavior.

In our implementation we use *tcpdump* [17] as packet analyzer. We filter the traffic on port 443 (i.e., we analyze on HTTPS in our implementation). Once the data is stored on the disk, we run an offline analysis over captured data using Bro. The first feature (see Table 1) uses an already existing script for the Bro's framework, called *validate-certs.bro* [2], which uses the Mozilla root store as trusted base. All the other features are implemented by us through Bro's scripting language, except for the sixth feature related to DGA domains, which uses an *n-gram* technique [13] to determine the level of randomness of a string.

## 4 Analysis of Selected Features

The selection of features is based on our assumptions (see Section 2), therefore we analyze them to see whether they can be useful as indicators of malicious connections or not. In a second step, we discuss the findings of our analyzed features on the SSL network traffic. We ran two different analysis. The first is done on 300 GB of SSL traffic. The goal of this analysis is to define which of the proposed features are helpful to determine the presence of malicious connections. The second evaluation is tailored to the outcomes of the previous analysis and is applied to a different dataset of 1 TB of SSL traffic. The set of features in the second evaluation is smaller as it only includes those features that we have identified as good indicators. In both analyses, our implementation examines only SSL connections that successfully completed the handshake protocol (i.e., the *Finished* message is sent [18]). A connection represents a unique instance of a successful handshake. Therefore, connections are not unique for each pair of hosts.

**Datasets.** We ran our first analysis on the network traffic of an international university[2]. This analysis is done by mirroring the whole traffic of the gateway of the university network to our own server. The traffic is then filtered and analyzed by our system. Our analyzed dataset is collected at the end of May 2014 (from 26th to 29th of May) and consists of 300 GB of SSL traffic. The second dataset, of 1 TB of traffic has been collected between the 8th and the 28th of July 2014. The goal of this second analysis is to further investigate the malicious connections previously identified.

## 4.1 Insignificant Features

Our first analysis on 300 GB of SSL traffic has shown that the release date of the certificate (F2), the existence of mutual authentication (F3) and the Levensthein distance for self-signed certificates (F5) do not seem to indicate the presence of malicious connections. Feature F2 has not shown any evidence of malicious connections. We have identified several certificates with a release date close to the establishment of the connections (e.g., less than 10 minutes), but they were all related to TOR connections, because the certificate subjects matches a typical pattern for TOR certificates (see Section 4.3), therefore they cannot be considered malicious. We found 262 certificates generated 10 minutes before the connection was established, 276 released within 1 day before the connection and 6589 generated more than 1 day before the connection. We found 198 connections, unrelated to TOR, that provided certificates with a release date of less than 1 day, however none of these connections had further indications that they could be considered malicious. Therefore, we consider this feature not relevant for our purposes. Mutual authentication (F3) is not commonly used within SSL communications. Analyzing our dataset of 891110 SSL connections, just 0.38% (i.e., 3386 connections) use *CertificateRequest* and *CertificateVerify* messages during the SSL handshake. 78.8% of such connections are authenticated with a valid certificate and the large majority of them are generated by the Apple Push

[2] The university has approximately 12.000 students and employees (combined).

Notification Service. None of these connections that are using mutual authentication are malicious, therefore we mark also this feature as insignificant. Lastly, we did not find any connection authenticated with a self-signed certificate where the Levensthein distance between the subject and the 100 most visited websites indicate a similar domains. Thus, we consider the Levensthein distance as an insignificant feature as well.

### 4.2 Indicating Features

In our analysis, we have found that the certificate chain validation (F1), the relation between server name and certificate subject (F4), the structure of the server name string (F6) and its format (F7) seem to be indicators for malicious connections. During our first analysis we have found 5 different malicious connections, and all these features can be potential indicators. In addition, we found one of these IP addresses before the professional service ThreatStop [16] marked it as malicious. This fact shows the strength of our identified indicators of malicious connections. The certificate chain validation (F1) has shown that 71% of the certificates in our dataset is properly verified as a valid certificate. 21.5% of the certificates instead do not provide their issuer. The amount of self-signed certificates (including those having self-signed certificates in chain) is equivalent to 0.8%. The amount of expired certificates we have encountered is 0.01%, while the rest 6.7% of the certificates was not validated by the Bro script (i.e., *validate-certs.bro* [2]). We consider this feature to be a possible indicator because in the malicious connections we have found, none of them use a properly validated certificate, as shown in Figure 2. In particular, 3 of these connections use an expired certificate from Amazon. Considering the small amount of expired certificates we have found in all our dataset, and the patterns of these malicious communications, we think this feature could be a helpful indicator.

The second indicating feature is the relation between server name and certificate subject (F4). If it is not properly enforced it can lead to a connection vulnerable to MitM attacks. In our entire dataset, 83.2% of the connections use the TLS Server name extension. 98% of these connections provide a proper certificate for the requested domain by the client. This is never true for TOR connections, where the server name never matches (100% of the cases) the certificate subject. Nonetheless, we consider this feature as a potential maliciousness indicator because, as shown in Figure 2, all the malicious connections we found present a mismatch between server name and certificate subject.

The third indicator is the structure of the server name string. TOR connections have shown a clear pattern of random second level domain (SLD), also confirmed by [2]. Two of the malicious connections we analyzed have a random server name (see Figure 2). These are outgoing connections from a TOR node within the university network. We think this feature can be helpful as an indicator.

Lastly, the fourth indicator is the format of the server name string. 0.02% of SSL connections in our dataset have a server name with a format different from the DNS hostname, which is the standard defined by the RFC6066 [15]. 156

connections have the IP address of the server, as a server name value. While 111 connections have random values (e.g., `01cf645e.32fa6d90`). Considering that 3 malicious connections out of 5 have a server name string that does not follow the standard (e.g., they have IP addresses as server name), we include this feature in our set of indicators (see Figure 2).

We give to our features a different level of strength, which depends on how many times they are encountered within the set of malicious connections. As shown in Table 1, F1 and F4 have a value 5/5, which means that in all the 5 malicious connections these features were present. F6 and F7 have a lower level of strength since they are present just in 2 and 3 cases ,respectively, within the malicious connections. A more detailed representation is depicted in Figure 2.

| **F#** | **Feature** | **Description** | **Malic. Indic.** | **Indicator Strength** |
|---|---|---|---|---|
| F1 | Certificate chain validation | Typical validation chain of X.509 certificates | X | 5/5 |
| F2 | Certificate time generation | Check the time from certificate generation and connection | - | 0/5 |
| F3 | Certificate request & Certificate verify | Check if mutual authentication has been requested | - | 0/5 |
| F4 | Server name belong to certificate subject | Check if the certificate is correct for the requested server name | X | 5/5 |
| F5 | Levensthein distance for self-signed certificate | Check if famous domains provide self signed certificates | - | 0/5 |
| F6 | Random generated server name domain | Check whether the server domain is random or not | X | 2/5 |
| F7 | Format of server name domain | Check whether the server domain follow the DNS hostname standard | X | 3/5 |

**Table 1.** Descriptive summary of the selected features. The features used as malicious indicators and their level of strength are identified.

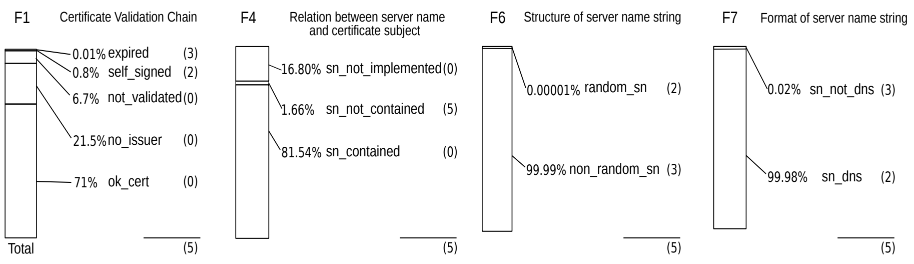

**Fig. 2.** Detailed representation of the analyzed indicating features with format: [% of dataset], [feature value], ([found in x-many malicious connections]).

### 4.3 Application

**Malicious Connections.** We identified 5 malicious connections within the SSL traffic. We verified their maliciousness using the public blacklist service offered by ThreatStop [16], a professional service that provides a blacklist of known criminal addresses. If the IP addresses are not blacklisted, we do not consider the connection as malicious. The connections have been analyzed using ThreatStop few days after the traffic has been captured, and this verification process has been done only once. Therefore, it is possible that our indicators would have identified more malicious connections, whose IP addresses were not yet blacklisted. We have chosen ThreatStop for two main reasons: (1) it is a professional service, meaning that the blacklist is always updated and properly maintained and (2) it focuses on threats that match our scenario such as criminal malware or botnets, which can be used for data exfiltration.

Two of these connections have IP addresses linked to SPAM activities (TOR connection), but do not share any pattern. The other three connections instead show exactly the same patterns: the servers use an expired certificate of Amazon to authenticate themselves (valid for the following domains `www.amazon.com`, `uedata.amazon.com`, `amazon.com`, `amzn.com`, `www.amzn.com`), and they have their destination IP address as server name field. Additionally, in the same dataset 3 other connections have been found using these same communication patterns. However, the IP addresses are not blacklisted, thus we cannot consider them to be malicious, although they seem very likely to be malicious due to the exact same characteristics.

We have ran a second analysis with a new dataset of 1TB of SSL traffic to investigate the malicious connections that we have found. In this analysis, we found another 28 connections that have the same patterns, but different IPs. In total, we have observed 34 connections over 14 different countries, where 10 of them have IPs labeled as malicious by ThreatStop [16] and all connections with the same source (i.e., a host inside the university network). Additionally, all these IP addresses, few weeks after being identified, were not reachable anymore on their port 443, as if the service was shutdown. Considering the following facts: (1) an expired certificate of Amazon.com has been used by several blacklisted IPs, (2) the location of these servers was spread all over Europe, and (3) the short lifetime of their services, we believe that what we have found can be considered a (not yet blacklisted) botnet. Another fact is that two of these IP addresses are also marked by ThreatStop as potential botnet. It is interesting to note that only 10 out of 34 IPs are blacklisted, although they share the same traffic characteristics. These connections have been identified due to the presence of features F1, F4 and F7. F1 identified that the certificate used by the server was expired. F7 showed that the format of the server name field was not following the DNS standard but was using an IP address. Finally, F4 showed that the server name and subject were not matching.

Looking at the features, it seems that the server name field plays an important role in identifying malicious connections. Not only the format of the domain, but also its relation with the subject of the certificate, which we believe is the main part of authentication of the server because it shows whether the answer of the server (i.e. certificate) matches the request of the client (i.e., server name). The randomness of the domain name seems also to be useful. Perhaps, using a more sophisticated technique to detect random domains could improve its impact as an indicator. Not surprisingly, also the validity of the certificate seems to be relevant. Malicious connections use certificates that are either expired or self-signed. Therefore, identifying connections that are not identified by proper validated certificates, can be an indicator.

Therefore, we believe our indicators analyze malicious traffic from a new and different perspective. During this second analysis, we have also found three other malicious connections (i.e., blacklisted by ThreatStop [16]) coming from the TOR exit node. Therefore, considering the two TOR malicious connections identified in the 300 GB of SSL traffic, we found a total of 5 malicious connections within TOR network traffic. The features present in these connections were F1, F4 and F6. In these cases the certificates were self-signed, therefore F1 was triggered. F6 identified them as malicious connections due to their "random looking" SLD. Lastly, F4 showed that the provided certificate was not valid to authenticate the requested server name.

**Connections Vulnerable to MitM Attacks.** Although our analysis focuses on malicious connections, the selected features allowed us to identify other misbehaviours, which weaken the security of SSL connections. We identified 5326 connections (0.6% of the entire dataset) that are potentially vulnerable to MitM attacks, due to a bad implementation of the authentication mechanism (i.e., the requested domain is not contained in the list of subjects of the certificate). 1251 of these connections are authenticated with valid certificates. Several of these misconfigurations are related to Akamai, a well known Content Distribution Network that provides its customers a single SSL certificate valid for different domains (e.g., `*.akamaihd.net` [3]). Also the Google Project SDPY, an open networking protocol for transporting web content, does not follow the specifications of SSL correctly, using hash values (e.g., `01cf645e.32fa6d90`) as the server name in the *ClientHello* request. With this outcome we confirm, as discussed in [4], that there are still several web applications using SSL in a wrong way. However, we apply a further analysis on these connections, and determined we can group the misconfiguration into two sets: *light* and *heavy*. A *light* misconfiguration is when the SLD of the server name matches with the SLD of the certificate subject, but there is a mismatch between subdomains (e.g., `example.website.com` and `fake.website.com`). We called them *light* because for an attacker it is hard to create a MitM attack, since he should generate or "steal" a certificate with the same SLD and a different subdomain, or he should compromise a CA and release a certificate with same SLD and different subdomain. A *heavy* misconfiguration is when the SLD of the server name is different from the SLD domain of the certificate subject (e.g., `www.example.com` and `www.malicious.com`). In this case the connection can be easily attacked by a MitM, because the certificate is not verified to match with the requested domain.

**TOR.** Analyzing the randomness of domains we encountered several TOR connections within the HTTPS traffic. We identified a simple pattern to distinguish it from normal HTTPS traffic: ServerName= `www.[randomstring].com` AND Subject=`www.[randomSLD].net` AND certificate_validation="Unable to get Certificate Issuer". This is a constant pattern in all TOR communications over port 443. All the TOR connections (i.e., 7127) have this pattern. Moreover, we were able to identify an exit node which was generating a lot of TOR traffic within the University network. Amman et al. [2] in their work, arrived at the same conclusions, with a very similar pattern: where Issuer and Subject match the pattern CN=`www.[randomstring].[tld]`. Both patterns successfully identify connections among TOR nodes. The TOR exit node presents within the university network is responsible for two of the malicious connections that we have found in our analysis. The server name is a random string, typical characteristic of TOR traffic. In these two cases, the connections were exiting the onion network, therefore the certificate provided by the real destination was not respecting the pattern of TOR nodes.

**Financial Corporation.** We analyzed the indicating features (see Section 4.2) in a further scenario. The dataset in this case is approximately 2 TB of network traffic, which has been captured over a period of two weeks, from the 23rd of March to the 9th of April 2015, within the infrastructure of an international financial corporation. Figure 3 shows a simplified architecture of our approach for data collection. The network traffic of client networks is mirrored to our solution, deployed on one of their machines. The traffic seen by our system is previously filtered by state-of-the-art security solutions deployed by the company that prevent clients to communicate with malicious domains. Moreover, the gateway firewall is a state-of-the-art solution capable of inspecting the SSL traffic. It does this decrypting the traffic, analyzing it and re-encrypting it and finally forwarding it towards its destination. However, the inspection is based on certain criteria, therefore not all the traffic is inspected. This firewall highly limits the number of connections analyzed by our system, because it uses self-signed certificates to re-encrypt the traffic, therefore the handshake that our system analyzes is not the original and features like F1 and F4 can be altered. For this reason, we filter these connections: we analyze the handshake of those connections that were not inspected by the firewall.

The analysis of the financial corporation network has also revealed vulnerable connections to MitM (i.e., heavy misconfiguration). We have identified 129 connections, where 118 provide an expired certificate (115 are connections to the same website). The remaining 11 connections have a self signed certificate in chain. We were also able to identify 14 TOR connections, despite the fact that the network traffic was filtered through a blacklisting mechanism to block TOR traffic. This result remarks how blacklisting solutions are not perfect. Lastly, we have identified two malicious connections. They share the same IP address, which is blacklisted by ThreatStop [16] as a potential botnet. Considering the well-protected infrastructure where the malicious connections have been identified, we believe this result underlines the strength of our indicators for the identification of malicious connections over SSL.

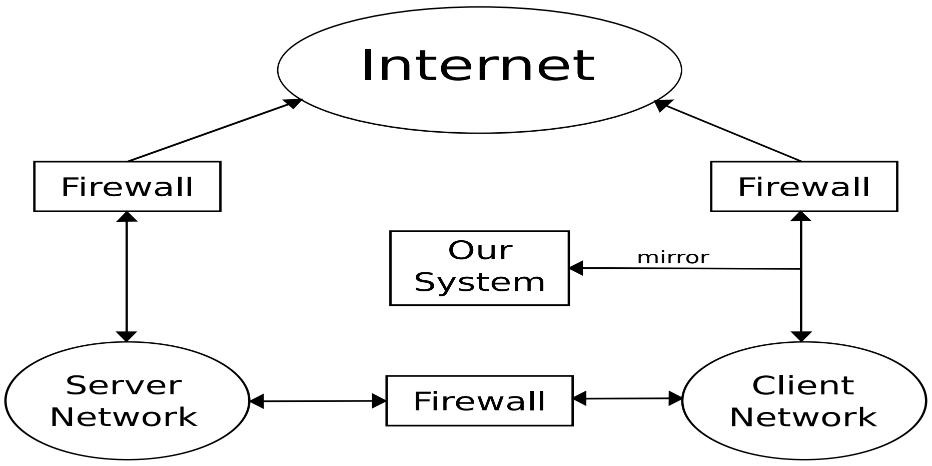

**Fig. 3.** High-Level representation of the data capturing within the Financial Corporation network

## 5 Limitations and Future Works

As we mentioned in the introduction, our work is an initial analysis of SSL malicious connections and it still has its limitations. First of all, the weight of our features has been computed from a small number of observation during the analysis of the first dataset. Although it can give an intuition about the effectiveness, because during the second analysis we found more malicious connections having such indicators, a more extensive validation should be done. This can be realized through an analysis of additional traffic. Another limitation of this work is that the set of features might be under fitting. There could be more handshake features that could be helpful in identifying malicious connection. Moreover, it is possible that more malicious connections have not been identified due to a limited set of indicators. Indeed, in the first analysis we ran ThreatStop against the subset of connections containing at least one of our indicators. This was a design decision, because it does not seem sensible to evaluate our indicators over connections that were not including them. As future work, extending the set of features is a necessary step. The features proposed by Pukkawanna et al. [7] could be used as helpful reference. The usage of machine learning techniques could also be useful in the analysis. Our judgement on the maliciousness of a connection heavily relies on the ThreatStop service. Therefore, it is possible that more connections were malicious and we did not consider them because they were not blacklisted by ThreatStop. This is a typical drawback of using blacklisting.

This might be improved using additional sources to verify the maliciousness of IP addresses. Lastly, the design and implementation of an intrusion detection system based on an extension of our work (e.g. with additional indicators) can be considered as interesting future work.

## 6 Conclusion

We presented a set of indicators of malicious connections for SSL. Analyzing the network traffic of an international university and a secured network of an international financial corporation with our indicators, we have identified several malicious connections. In the university setting we have found in total (see Section 4.3): 5 connections related to TOR traffic, and 10 connections that share the same communication patterns and use an expired certificate of Amazon for authentication. In the financial corporation we have identified 2 malicious connections. All these connections have IP addresses blacklisted by ThreatStop [16]. One of these IP addresses was blacklisted after we have identified it, and 4 addresses are associated with botnet activities. Furthermore, we have identified 24 other connections that are also using the expired certificate of Amazon and share the same traffic characteristics but are not blacklisted (yet). Having 34 connections sharing these characteristics, we strongly believe that we have found a potential botnet on SSL. Nonetheless, we have also verified several results of prior research on the identification of many vulnerable SSL connections to MitM attacks, but by using a different method. This work is not intended as an intrusion detection system, although our indicators, through further validation, could potentially be part of such a system. A deeper understanding and further research is needed to turn our work into an intrusion detection system. We consider this as future work. The goal of this work is to identify features that could indicate malicious behaviors. We believe this set of indicators is a good starting point. Further extensions are still needed, as we suggested in Section 5.

**Acknowledgements** This work was partially supported by the AVATAR project (funded by the Ministry of Security and Justice of the Kingdom of the Netherlands) and the INAETICS project (funded by the European Regional Development Fund). Furthermore, Riccardo Bortolameotti carried out much of his work as part of his final master thesis within the EIT Digital Master program on Security and Privacy. Andreas Peter's and Maarten H. Everts' contributions to this paper were supported by the THeCS project as part of the Dutch national program COMMIT/.

## References

1. Holz, R., Braun, L., Kammenhuber, N., Carle, G. The SSL Landscape: A thorough Analysis of the X.509 PKI Using Active and Passive Measurements. SIGCOMM IMC 2011, pp. 427-444, ACM.
2. Amann, B., Vallentin, M., Hall, S., Sommer, R. Revisiting SSL: A Large-Scale Study of the Internets Most Trusted Protocol. Technical Report 2012, ICSI.

3. Amann, B., Sommer, R., Vallentin, M., Hall, S. No Attack Necessary: The Surprising Dynamics of SSL Trust Relationships. ACSAC 2013, pp. 179-188, ACM.
4. Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., Shmatikov, V. The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software. CCS 2012, pp. 38-49, ACM.
5. Fahl, S., Harbach, M., Muders, T., Baumgrtner, L., Freisleben, B., Smith, M. Why Eve and Mallory Love Android: An Analysis of Android SSL (in) Security. CCS 2012, pp. 50-61, ACM.
6. Conti, M., Dragoni, N., Gottardo, S. MITHYS: Mind The Hand You Shake - Protecting Mobile Devices from SSL Usage Vulnerabilities. In Security and Trust Management 2013, pp. 65-81, Springer.
7. Pukkawanna, S., Kadobayashi, Y., Blanc, G., Garcia-Alfaro, J., Debar, H. Classification of SSL Servers based on their SSL Handshake for Automated Security Assessment. BADGERS 2014, to appear.
8. Bates, A., Pletcher, J., Nichols, T., Hollembaek, B., Tian, D., Butler, K. R., Alkhelaifi, A. Securing SSL Certificate Verification through Dynamic Linking. CCS 2014, pp. 394-405, ACM.
9. Holz, R., Riedmaier, T., Kammenhuber, N., Carle, G. X. 509 Forensics: Detecting and Localising the SSL/TLS Men-in-the-Middle. ESORICS 2012, pp. 217-234, Springer.
10. Clark, J., van Oorschot, P. C. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements. Symposium on Security and Privacy (SP) 2013, pp. 511-525, IEEE.
11. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou II, N., Abu-Nimeh, S., Lee, W., Dagon, D. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. USENIX Security Symposium, pp. 491-506, USENIX.
12. Paxson, V. Bro: a System for Detecting Network Intruders in Real-Time. USENIX Security 1998, USENIX.
13. Wang, K., Parekh, J. J., Stolfo, S. J. Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. RAID 2006, pp. 226-248, Springer.
14. Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S. Phoenix: DGA-based Botnet Tracking and Intelligence. In Detection of Intrusions and Malware, and Vulnerability Assessment 2014, pp. 192-211, Springer.
15. RFC6066. Internet Engineering Task Force (IETF). Transport Layer Security (TLS) Extensions: Extension Definitions. `https://tools.ietf.org/html/rfc6066`
16. ThreatStop Check IP service. `http://www.threatstop.com/checkip`
17. Tcpdump & Libpcap. `http://www.tcpdump.org/`
18. RFC5246. Internet Engineering Task Force (IETF). The Transport Layer Security (TLS) Protocol Version 1.2 - The TLS Handshaking Protocols `https://tools.ietf.org/html/rfc5246#section-7`