Limited Proxying for Content Filtering Based on X.509 Proxy Certificate Profile*

Islam Faisal and Sherif El-Kassas

Computer Science and Engineering Department, The American University in Cairo, Egypt
{islamfm,sherif}@aucegypt.edu

Abstract. Use of proxy servers to filter content is very critical in achieving both personal and enterprise security. A common practice to perform this task is by allowing a man-in-the-middle to intercept the traffic unconditionally and act as a proxy between the client and the server. While this method is good enough for unencrypted HTTP connections, it is not a good practice in encrypted HTTPS (SSL/TLS) connections. In this paper, we introduce an access-controlled limited proxying framework to allow HTTPS content filtering based on the Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. Limited proxying allows the client and the server to decide which content can be accessed by a proxy to avoid compromise of sensitive content. The proposed framework grants the user full control to grant or revoke specific proxy privileges which enhances the user's privacy online. We also define and argue about the security properties of the framework as well as some practical considerations for its implementation.

Keywords: Proxy Servers · Content Filtering · Proxy Certificate Profile · Privacy

1 Introduction

Content filtering is used extensively to achieve different tasks. For example, content filtering systems are used by parents to filter inappropriate content [45,27], by anti-malware systems such as [7] to filter content in order to ensure enterprise or personal security, and by governments for surveillance and censorship [17,8].

There exist different types of content control systems [40] such as email filtering [6,1], DNS-based filtering [4], network-based filtering, and web proxy filtering [40].

In HTTP proxy filtering, the client does not connect directly to the server, yet, the network is configured to allow (or force) users to connect to HTTP servers via a *proxy server*. This man-in-the-middle approach is sometimes used for the same purposes to filter connections to HTTPS servers. However, this

* This is an accepted manuscript version of a conference paper presented in the International Conference on Security for Information Technology and Communications. The final authenticated version will be available on Springer LNCS.

approach doesn't differentiate between legitimate proxying approved by users and forged SSL certificate attacks initiated against users. Moreover, this interception exposes the security of the SSL/TLS protocol [1] and can result in severe consequences [19,33,31]. This motivates the need for a framework that legalizes the use of HTTPS proxy servers so that a clear distinction is made between accountable proxying and unauthorized attacks.

Contributions The following contributions are included in this paper:

- We introduce a use case and a policy language for the X.509 Proxy Certificate Profile described in RFC 3820 [49] to enable limited proxying capabilities for the task of web content filtering.
- We propose a framework and practical advise for the mechanism of verifying the identity and privileges of content filtering systems and handling sensitive data on connections established with a principal identifying by a proxy certificate. This privacy-enhancing framework gives the user full control to grant or revoke specific proxy privileges.
- We discuss the security properties of the proposed framework and demonstrate an application as well as practical considerations when implementing in the real world.

Outline The rest of the paper is organized as follows. Section 2 explains the necessary background and related work. Then, section 3 defines the threat model, system requirements, and the proposed framework. Then, section 4 discusses the security properties, applications, and practical advise for implementation. Finally, section 5 concludes the paper and introduces the future directions.

2 Background and Related Work

2.1 Content Filtering Systems

Content filtering systems are systems that inspect web content and/or web URLs for removing or blocking unwanted content [40]. These systems vary in both the method and the application. Nowadays, most content filtering systems use machine learning or artificial intelligence [44,28] to rule out unwanted content. Some systems also use this in combination with a list of unwanted websites [40] such as Google's safe browsing API [2]. Content filtering is used at scale for a wide variety of applications such as:

- Parental access control to block adult or unsafe content [45,27]
- Malware detection by antivirus software such as Symantec WebFilter [3]
- Government mass surveillance and content censorship [17,8]

[1] In this paper's scope, we are not interested in differentiating between SSL and TLS connections. Unless clearly stated or suffixed by a version number, we consider both terms as a method to communicate encrypted web traffic payload.

[2] https://developers.google.com/safe-browsing/

[3] https://www.symantec.com/products/webfilter-intelligent-services

2.2 HTTPS Interception

The two standard protocols to encrypt HTTP traffic are *Secure Sockets Layer (SSL)* [25] and its successor *Transport Layer Security (TLS)* [15,13,14,46]. When the TLS protocol was introduced [15], it was assumed that all the functionalities reside on the connection endpoints. Middlebox network security solutions are not "legal" under this assumption, so middleboxes resort to going around the protocol. *HTTPS interception* refers to the common man-in-the-middle (MITM) attack done by inserting a middlebox between the HTTPS client and server which is demonstrated in figure 1. Intercepting HTTPS traffic using this naive MITM approach is not recommended [19] because it can severely reduce connection security. In the literature, multiple alternatives to naive interception were proposed [19] such as:

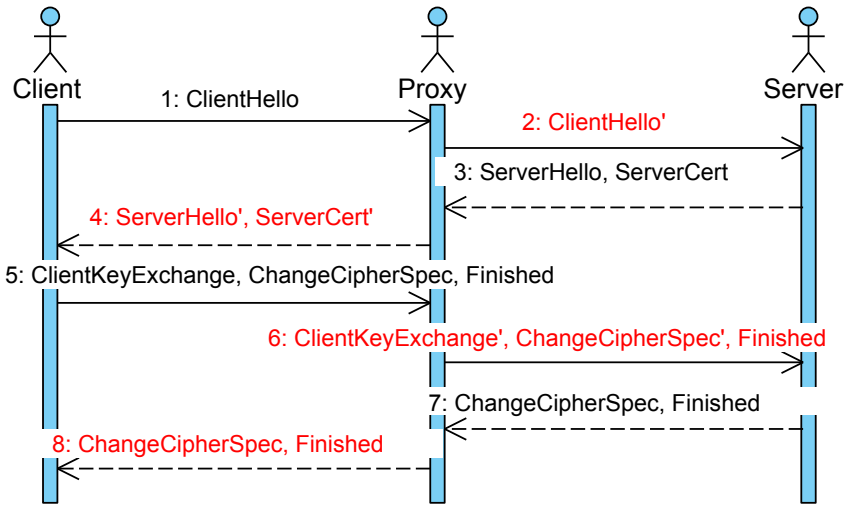

Fig. 1. Man-in-The-Middle HTTPS handshake interception by a proxy: The proxy intercepts the connection by creating two client-server connections with the original parties. Usually, a new *certificate authority (CA)* is added on the client side to trust that "forged" ServerCert' certificate. Messages transmitted by the proxy are colored in red and a prime is added to distinguish possibly altered messages. In addition to the fact that the proxy will use a "forged" certificate, this approach is not safe and can degrade security because the proxy can use out-dated TLS versions or weak cipher suites [19].

1. **HTTP 2.0 Explicit Trusted Proxy RFC [36]:** This very simple solution requires middleboxes to explicitly notify the client of the interception.
2. **TLS Proxy Server Extension [37]:** This extends the idea of explicit proxy in HTTP to HTTPS, requiring the proxy to indicate the interception, and to additionally relay proxy-server session information back to the client.

3. **Multi-context TLS (mcTLS) [41]:** This approach introduced an extended version of TLS that requires endpoints to explicitly specify permitted middleboxes in order to securely authenticate each hop and cryptographically control exactly what data middleboxes can access. This protocol, however, was proven to be insecure because it lacked formal analysis [3].
4. **BlindBox [47]:** In this approach, instead of thinking how to "legitimately" insert the proxy, the proxy is just let to observe the traffic in its encrypted form and advise what to do after performing *deep packet inspection (DPI)* over encrypted traffic.
5. **Hardware-Assisted Middleboxes [26,29,43,48,18,11,35]:** These systems leverage the use of trusted hardware such as Intel's *Software Guard Extensions (SGX)* [2,38,30,10] to support the middlebox's functionality and security properties.
6. **Formally verified accountable proxying [3]:** This paper introduced a new notion of security for middleboxes intercepting the TLS connections called *Authenticated and Confidential Channel Establishment with Accountable Proxies (ACCE-AP)*. They use this formal model to show the insecurity of existing middlebox techniques such as [41] and introduce a formally verified *ACCE-AP*-secure framework for accountable proxying.

Our method is different from prior work in the sense that it employs an access-control strategy by properly delegating an honest proxy via proxy certificates [49] without reliance on trusted hardware. Except for relaying back the server certificate along with the proxy certificate to the client and adding extra restrictive checks, we don't modify the handshakes of the TLS. Additionally, our method is independent of the TLS version used and doesn't require much software updates to the connection endpoints.

2.3 Analyzing SSL Certificates

It is not a good practice to blindly trust an SSL certificate. Many problems including private keys compromise, and forged SSL certificates can arise in the wild. For example, [31] analyzed SSL certificates from a sample collected via a research network as well as active websites and found that over 40% of their sample exhibit broken certificate chains. In a study done by Facebook [33], it was found that 0.2% of the SSL connections sampled from a small portion of Facebook traffic were tampered with forged SSL certificates, most of them related to antivirus software and corporate-scale content filters. Although some of those "forged" certificates are issued based on the user's consent, it is not a good idea to pool an attacker and a legitimate antivirus in the same pool.

2.4 Proxy Certificate Profile

The concept of *computational grids* [22] has emerged in the late 90's to support high performance computing needs. The Globus project [23] was introduced to enable the construction of computational grids providing pervasive, dependable,

and consistent access to high-performance computational resources, despite geographical distribution of both resources and users. As these computational grids grew and became popular, their security concerns grew rapidly as well. [24] was one of the earliest efforts to introduce security requirements, develop a security policy and corresponding security architecture for computing grids. This came with an implementation within the Globus metacomputing toolkit [23]. Later, the implementation was separated as an online repository credentials management system called *MyProxy*[4] [42] which was later proposed as IETF RFC 3820 [49]. RFC 3820[5] defines a certificate profile based on the Internet X.509 Public Key Infrastructure defined in RFC 3280 [32] and updated in RFC 5280 [9].

Terminology We list some useful terms from RFC 3820 [49] and RFC 5280 [9] that will be helpful in describing the proposed framework later:

- ***Certificate Authority (CA):*** An authority that is authorized to certify entities by certificates upon which the relying parties can depend.
- ***End Entity Certificate (EEC):*** Sometimes called *Public Key Certificate (PKC)*; An X.509 Public Key Certificate issued to an end entity, such as a user or a service, by a CA.
- ***Proxy Certificate (PC):*** A PKC with special fields issued by an end entity delegating some of its priveligies to another entity.
- ***Proxy Issuer (PI):*** An entity with an End Entity Certificate or Proxy Certificate that issues a Proxy Certificate.
- ***Attribute Certificate (AC):*** Sometimes called *Authorization Certificate*; (defined in RFC 3281 [21] which was obsoleted by RFC 5755 [20]) A certificate that contains the attributes associated with an end entity. While a PKC is used as a proof of identity, the AC is used as a proof of authorization.
- ***Attribute Authority (AA):*** (defined in RFC 3281 [21] which was obsoleted by RFC 5755 [20]) An authority that can issue attribute certificates.
- ***Certificate Revocation List (CRL):*** A list of certificates that were revoked before their expiry dates. Certificates in this list should not be trusted by relying parties.

Proxy certificates are associated with a public and private key pair that is separate from its holder's EEC. A proxy certificate can only sign other proxy certificates. The *relying party* is the party who is interested in verifying the validity of the certificate. For example, in a TLS connection where a server provides a server certificate, a relying party can be the client's web browser.

This profile adds the extension listed in listing 1.1 which introduces the following proxy certificate fields:

1. *pCPathLenConstraint*: An integer that defines the delegation depth for a certain proxy.

[4] http://grid.ncsa.illinois.edu/myproxy/

[5] Although dating back to 2004, this is the most updated version of the RFC to our knowledge.

2. *policyLanguage*: An identifier for the language used in interpreting the policy field. The RFC encourages acquiring an *object identifier* (OID) for these languages.
3. *policy*: The delegation policy from the issuer to the proxy that specified the scope of the privileges being delegated to the proxy.

Listing 1.1. ProxyCertInfo Extension Definition From [49] defined in the *Abstract Syntax Notation One (ASN.1)* [34]

```
1  -- The ProxyCertInfo Extension
2     ProxyCertInfoExtension   ::= SEQUENCE {
3           pCPathLenConstraint
                ProxyCertPathLengthConstraint OPTIONAL,
4           proxyPolicy              ProxyPolicy
5            }
6
7     ProxyCertPathLengthConstraint   ::= INTEGER
8     ProxyPolicy   ::= SEQUENCE {
9           policyLanguage           OBJECT IDENTIFIER,
10          policy                   OCTET STRING OPTIONAL
11           }
```

Issuing a proxy certificate To issue a proxy certificate, first a new public and private key pair is generated. A request for a PC is created. The proxy issuer verifies that the PC request is valid by checking that the PC fields are appropriately set. If the fields are appropriately set, the PI signs the PC using the private key of either an associated EEC or a PC. A PI may use the *pCPathLenConstraint* field of the *proxyCertInfo* to limit subsequent delegation of this PC. If it is set to zero, the proxy's delegated privileges can not be delegated further.

Verifying a proxy certificate Verifying a request made using a proxy certificate is the responsibility of the relying party. Definition 1 gives the conditions to validate the path of a proxy certificate. It has to be used in conjunction with path validation of EEC as in RFC 5280. In definition 2, the conditions for a valid proxy certificate are stated. These definitions are mentioned here because they are slightly different from RFC 3820 in the sense that we allow a proxy issuer to revoke a proxy certificate by publishing it to a CRL.

Definition 1. *A path containing a succession of proxy certificates* $\{C_1, ..., C_n\}$ *is valid if:*

1. C_1 *is a valid proxy certificate issued by a valid end entity certificate*[6]

[6] We don't describe how to verify an end entity certificate in this definition. Verifying an EEC is done in accordance with RFC 5280.

2. $\forall\, x \in \{1, ..., n-1\}$, *the subject of certificate* C_x *is the issuer of the proxy certificate* C_{x+1}
3. $\forall x \in \{1, ..., n\}$, C_x *is valid at the time in question (not expired nor revoked).*
4. $\forall x \in \{1, ..., n-1\}$, *if* C_x *contains a pCPathLenConstraint field, then* $n - x \leq$ *pCPathLenConstraint*

Definition 2. *Let* $\{C_1, ..., C_n\}$ *be a path of proxy certificates with policies written in the policy language policyLanguage and* C_0 *be the parent EEC of* C_1*. A request* Q *made via proxy certificate* C_n *is considered valid by a relying party* R *if it satisfies the following conditions:*

1. $\{C_1, ..., C_n\}$ *is a valid path of proxy certificates*
2. R *can interpret policyLanguage*
3. Q *is allowed under* C_n*'s policy*
4. *If the proxy issuer of* C_n *is an EEC, then* R *must authorize this PI to perform* Q
5. *If the proxy issuer of* C_n *is a PC, then either (i)* R *authorizes the owner of* C_{n-1} *to perform* Q*, or (ii)* C_{n-1} *inherits the right to perform* Q *by a parent proxy certificate* C_{n-2}*. This right is verified recursively using this definition by checking if* Q *made via* C_{n-2} *is considered valid by the relying party* R*.*

If the request is not valid, it is up to the relying party to either deny it or act as if no proxy certificate was provided.

3 Proposed Framework

3.1 Threat Model

We assume the attacker has the capabilities of the Dolev-Yao active attacker model [16]. This attacker can eavesdrop, delete, replace, re-transmit, and delay messages that honest parties communicate over the network. They can also send any message over the network. We assume the security of private keys and private communication channels as well as perfect cryptography. We assume that the proxy server is a trusted party and won't collude or tamper with the traffic. We don't consider the process of certifying entities and its socioeconomics. However, in section 4.3, we give advice on bringing this method to the real world with some consideration to these factors.

3.2 Security Requirements

The client-proxy and proxy-server connections must conform to the security requirements of an HTTPS connection. Besides these requirements, the proposed framework has to satisfy some requirements that we list below and argue about their validity in section 4:

1. **Authorized Proxying:** Proxy connections are only accepted from proxies with a valid EEC and PC.

2. **Limited Proxying:** The client and the server have control over what pages or parts of traffic can be shared with the proxy. This is based on:
   (a) Proxy Trust Level
   (b) Content Sensitivity
3. **Proxy Detection:** Both the client and the server must be able to detect and distinguish between direct client-server connections and connections established via an intermediate proxy.
4. **Limited-Depth Proxying:** The depth of the chain of proxy certificates is controlled by the entity delegating the proxy
5. **Certificate Path Validation:** The relying party can trace the path of the delegation and verify that the delegation is legitimate.

3.3 Limited Proxying Framework

We propose a framework for providing access-controlled content filtering services based on the proxy certificate profile defined in section 2.4. The framework definition includes defining a filtering-specific access-control policy language, processes for issuing and verifying proxy certificates, and the handshakes used to create the client-proxy and proxy-server connections.

Content Filtering Policy Language We define a policy language in the proxy certificate profile that is suitable to the task of content filtering. This policy can be used to determine if the proxy is allowed to visit the content they are trying to visit. The policy in this policy language is defined as the concatenation of:

1. A integer representing the *trust level* associated with that proxy certificate.
2. A list of allowed and disallowed domains that the proxy can visit on behalf of the user.
3. A set of allowed cipher suites to be used by the proxy.

We elaborate more on this definition in appendix A. Given a certain content on one of the proxy's allowed domains, the server, can determine if the proxy can serve this content based on the trust level of the proxy server and the sensitivity of the content. For example, a high-quality anti-virus can be trusted more than a cheap parental-control software.

Managing End Entity Certificates Similar to the X.509 PKI, the identities of all entities including clients, servers, and proxies is verified using EEC. The issuance of an identity certificate is done typically via one of the trusted certificate authorities. An example is the free, automated, and open-source certificate authority *Let's Encrypt*[7]. Proxies are required to identify by an identity end entity certificate that is different from the proxy certificate issued by the corresponding client proxy issuer. An end entity certificate serves as the proxy's long-term identity, while a proxy certificate is a proof of delegation issued by every client. In the framework, a proxy can't function if their EEC is expired or revoked even if their PC is valid.

[7] https://letsencrypt.org/

Managing Proxy Certificates Proxy certificates are issued by a proxy issuer, which is in our case either the client or another proxy server. The proxy certificate is issued using the procedure described in section 2.4 while making sure the certificate request conforms to the policy language defined previously by including these fields:

- Proxy depth
- The policy fields (trust level, allowed and disallowed domains, allowed cipher suites) or customized privileges identifier in the case of use of a customized policy language.

To validate a proxy certificate by the relying party, the conditions in definitions 1 and 2 mentioned in section 2.4 have to be satisfied besides the following language-specific conditions if the proxy issuer is already a proxy:

1. The trust level is at most as privileged as the parent proxy.
2. The set of allowed domains is a subset of the parent proxy's allowed domains.
3. The set of disallowed domains is a superset of the parent proxy's disallowed domains.
4. The cipher suites set is a subset of the parent proxy's cipher suites.

Revoking Proxy Certificates At any point in time, the client may decide to revoke a proxy certificate they have issued at a point in time before its expiry date. To do so, they publish the certificate to a certificate revocation list (CRL) as explained in section 2.4. A note is given on the practicality of this process in section 4.3.

Establishing Proxy Connections As illustrated in figure 2, to establish a full proxy connection, the proxy creates two connections; one to the client initiated by the client and one to the server initiated by the proxy on behalf of the client as follows:

1. **Client-Proxy Connection**: The client initiates a connection to the proxy server using their EEC and the proxy uses their PC and EEC. The user uses Definition 2 to determine if this connection request is valid based on the validity of the proxy's introduced proxy certificate and based on that the client decides whether to proceed.
2. **Server-Proxy Connection**: The proxy initiates a connection to the server identifying by its EEC and PC while the server identifies by a typical X.509 certificate as in RFC 5280. Here, we mandate that the proxy notifies the server that the connection is a proxy-type connection and that the proxy provides their proxy certificate which will be considered the client certificate in this proxy-server TLS connection. The proxy has also to use one of the cipher suites required by the client in the policy language. In section 4.3, we give a note on the practicality of verifying the client credentials. The server here must verify that the cipher suite being used is one of the suites given in the proxy certificate.

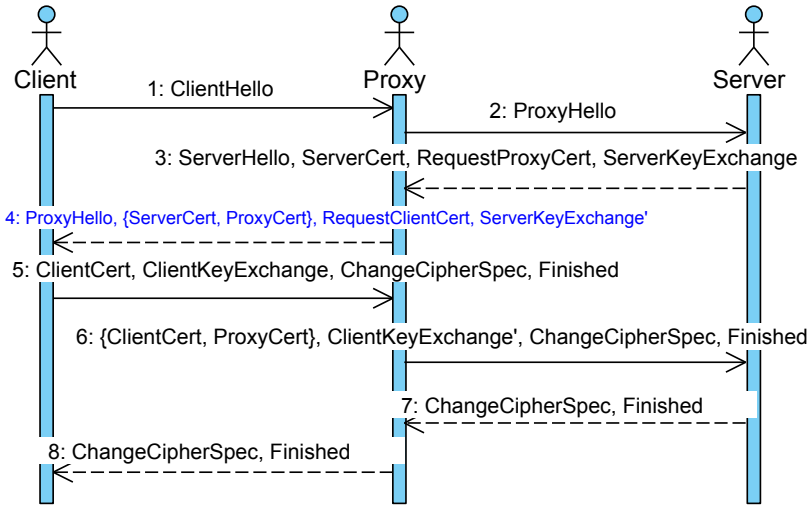

Fig. 2. HTTPS interception Handshake in the proposed framework: Note that the proxy relays back the server certificate information in step 4.

Relaying Server Certificate Information When establishing the proxy-server connection, we require the proxy to relay back the server certificate information to the user (step 4 in figure 2). The client uses the certificate information to decide if they want to proceed with the connection. As explained in section 3.1, it is assumed that the proxy is an honest principal who will relay the certificate information as is. The client can decide then based on their trusted certificate authorities whether to accept the relayed server certificate.

Sharing Content via Proxy After establishing both connections successfully, the client tunnels their data to the server via the proxy. For each payload the client wishes to send, they decide based on the trust level of the connected proxy if they want to send this specific content over the proxy connection. The server does the same by selectively deciding which content to send over via the proxy.

4 Discussion

4.1 Security Properties Analysis

In this subsection, we argue how the security requirements described in section 3.2 are satisfied in this framework. Considering the client-proxy and proxy-server connections atomically, the only modification to the TLS handshakes is that we mandate that the proxy relays back to the client the server certificate (step 4 in figure 2) and some extra restrictive checks to be done by the client and the server. Therefore, the security properties of each connection by itself depends

on the version of the TLS protocol used, and assuming a secure version such as TLS 1.3 [46], both proxy connections don't expose the security properties of the individual connections. We now give a short argument for each of the other properties about the framework as a whole:

1. **Authorized Proxying:** Based on the request validity criteria described in Definition 2, a connection can be only established via a proxy with a valid EEC and PC with valid delegation privileges.
2. **Proxy Detection:** The proxy connection is done via a proxy certificate which is distinguishable from an EEC. A relying entity can clearly determine if the connection is direct or via a proxy.
3. **Limited Proxying:** In the proxy certificate fields, the level of trust and allowed domains to be accessed is specified. The relying party (client or server) can choose what content to share over a proxy server based on these fields.
4. **Limited-Depth Proxying:** The depth of delegation is specified in the proxy certificate field pCPathLenConstraint as described in RFC 3820 and enforced in the certificate issuance and validation processes.
5. **Certificate Path Validation:** The framework uses the path validity criteria described in Definition 1 which relies on the chain of trust principle. This enables the relying user to trace the trail of certificates up to a root certificate they trust.

4.2 Applications

This framework has the typical applications mentioned in section 2.1 that a content filtering system can perform. We demonstrate an application of our framework by the following example from the enterprise security domain.

Enterprise Content Filtering For a company X to use the proposed framework to do company-wide content filtering using the proxy server Y to achieve enterprise security, one strategy can be as follows. The company employees can issue a proxy certificate to the company with a depth greater than 1. The company can then use this proxy certificate to subsequently delegate the content filtering proxy Y to act as a proxy between the employee and the requested web service. When the employees start surfing the web, they will find that their connections are being tunneled through Y as instructed by X. Since this path can be validated up to the client's EEC, the client can trust this connection as long as the other request validity conditions are satisfied.

4.3 Practical Considerations

Server-Side Proxy Certificates In our framework, we chose to let the proxy issuer be the client. In some cases, it may be important for the server to selectively determine which proxies can be used to access their services and therefore requiring issuing proxy certificates from both the client and the server.

Trusting Proxy Certificates In today's world, most HTTPS connections are done without providing a client certificate or caring about its validity. Although it is crucial for the client to verify that the proxy certificate is valid, it may be an overhead for the server to verify client-issued proxy certificates, specially that not all normal users own trusted end entity certificates. In this case, a server may choose to deal with the proxy connection as if it were a direct connection and leaving the choice of what content is shareable over the proxy to the client. One more issue is the normal users' ability to publish revoked proxy certificate serial numbers to a global CRL that web servers rely on. While this may be hard in practice, we still include it in the framework design for completeness.

Relaying Server Certificate Information In the proposed framework, relaying the server certificate information back to the user provides the user with finer control on the certificates they want to trust. Not only that the proxy has to trust the server certificate, but also the client has to give their final approval via their own chain of trust.

Limitations Although we have been struggling to distinguish between an attacker and a legitimate proxy, it doesn't mean that this method eliminates the risks associated with practical use of SSL certificates in the wild mentioned in section 2.3 such as forged, expired, or misconfigured SSL certificates.

5 Conclusions

We have proposed a novel method to establish proxy connections for content-filtering purposes between clients and web services. The proposed framework establishes an access-controlled limited proxying method that enhances the privacy of the users online. We have also analyzed the security properties and some practical considerations of this framework and demonstrated how it can be applied in the domain of enterprise security as an application.

In the future, we want to consider the applicability of this method to the recent TLS 1.3 protocol draft [46]. It is also interesting to verify our security requirements using formal methods such as ProVerif[8] [5] or Tamarin Prover[9] [39] by extending existing formal models such as [12]. We are also interested in implementing this framework by adding software support to the participating parties; clients, proxies, and servers.

References

1. Almomani, A., Gupta, B., Atawneh, S., Meulenberg, A., Almomani, E.: A survey of phishing email filtering techniques. IEEE communications surveys & tutorials **15**(4), 2070–2090 (2013)

[8] http://prosecco.gforge.inria.fr/personal/bblanche/proverif/
[9] https://tamarin-prover.github.io/

2. Anati, I., Gueron, S., Johnson, S., Scarlata, V.: Innovative technology for CPU based attestation and sealing. In: Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy. vol. 13. ACM New York, NY, USA (2013)
3. Bhargavan, K., Boureanu, I., Delignat-Lavaud, A., Fouque, P., Onete, C.: A formal treatment of accountable proxying over tls. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 799–816 (May 2018). https://doi.org/10.1109/SP.2018.00021
4. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive DNS analysis. In: Ndss (2011)
5. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Proceedings of the 14th IEEE Workshop on Computer Security Foundations. pp. 82–. CSFW '01, IEEE Computer Society, Washington, DC, USA (2001), http://dl.acm.org/citation.cfm?id=872752.873511
6. Blanzieri, E., Bryl, A.: A survey of learning-based techniques of email spam filtering. Artificial Intelligence Review **29**(1), 63–92 (2008)
7. Canali, D., Cova, M., Vigna, G., Kruegel, C.: Prophiler: A fast filter for the large-scale detection of malicious web pages. In: Proceedings of the 20th International Conference on World Wide Web. pp. 197–206. WWW '11, ACM, New York, NY, USA (2011). https://doi.org/10.1145/1963405.1963436, http://doi.acm.org/10.1145/1963405.1963436
8. Chen, T.M., Wang, V.: Web filtering and censoring. Computer **43**(3), 94–97 (March 2010). https://doi.org/10.1109/MC.2010.84
9. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor (May 2008), http://www.rfc-editor.org/rfc/rfc5280.txt, http://www.rfc-editor.org/rfc/rfc5280.txt
10. Costan, V., Devadas, S.: Intel SGX explained. IACR Cryptology ePrint Archive **2016**(086), 1–118 (2016)
11. Coughlin, M., Keller, E., Wustrow, E.: Trusted click: Overcoming security issues of nfv in the cloud. In: Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization. pp. 31–36. SDN-NFVSec '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3040992.3040994, http://doi.acm.org/10.1145/3040992.3040994
12. Cremers, C., Horvat, M., Hoyland, J., Scott, S., van der Merwe, T.: A comprehensive symbolic analysis of tls 1.3. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1773–1788. CCS '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134063, http://doi.acm.org/10.1145/3133956.3134063
13. Dierks, T., Rescorla, E.: The transport layer security (tls) protocol version 1.1. RFC 4346, RFC Editor (April 2006), http://www.rfc-editor.org/rfc/rfc4346.txt, http://www.rfc-editor.org/rfc/rfc4346.txt
14. Dierks, T., Rescorla, E.: The transport layer security (tls) protocol version 1.2. RFC 5246, RFC Editor (August 2008), http://www.rfc-editor.org/rfc/rfc5246.txt, http://www.rfc-editor.org/rfc/rfc5246.txt
15. Dierks, T., Allen, C.: The TLS protocol version 1.0. RFC 2246, RFC Editor (January 1999), http://www.rfc-editor.org/rfc/rfc2246.txt, http://www.rfc-editor.org/rfc/rfc2246.txt
16. Dolev, D., Yao, A.C.: On the security of public key protocols. In: Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science. pp.

350–357. SFCS '81, IEEE Computer Society, Washington, DC, USA (1981). https://doi.org/10.1109/SFCS.1981.32, https://doi.org/10.1109/SFCS.1981.32
17. Dornseif, M.: Government mandated blocking of foreign web content. arXiv preprint cs/0404005 (2004)
18. Duan, H., Yuan, X., Wang, C.: Lightbox: Sgx-assisted secure network functions at near-native speed. CoRR **abs/1706.06261** (2017), http://arxiv.org/abs/1706.06261
19. Durumeric, Z., Ma, Z., Springall, D., Barnes, R., Sullivan, N., Bursztein, E., Bailey, M., Halderman, J.A., Paxson, V.: The security impact of https interception. In: Proc. Network and Distributed System Security Symposium (NDSS) (2017)
20. Farrell, S., Housley, R., Turner, S.: An internet attribute certificate profile for authorization. RFC 5755, RFC Editor (January 2010)
21. Farrell, S., Housley, R.: An Internet Attribute Certificate Profile for Authorization. RFC 3281, RFC Editor (April 2002), http://www.rfc-editor.org/rfc/rfc3281.txt
22. Foster, I., Kesselman, C.: Computational grids: The future of high performance distributed computing (1998)
23. Foster, I., Kesselman, C.: The globus project: a status report. In: Heterogeneous Computing Workshop, 1998. (HCW 98) Proceedings. 1998 Seventh. pp. 4–18 (Mar 1998). https://doi.org/10.1109/HCW.1998.666541
24. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A security architecture for computational grids. In: Proceedings of the 5th ACM Conference on Computer and Communications Security. pp. 83–92. CCS '98, ACM, New York, NY, USA (1998). https://doi.org/10.1145/288090.288111, http://doi.acm.org/10.1145/288090.288111
25. Freier, A., Karlton, P., Kocher, P.: The secure sockets layer (ssl) protocol version 3.0. RFC 6101, RFC Editor (August 2011), http://www.rfc-editor.org/rfc/rfc6101.txt, http://www.rfc-editor.org/rfc/rfc6101.txt
26. Goltzsche, D., Rüsch, S., Nieke, M., Vaucher, S., Weichbrodt, N., Schiavoni, V., Aublin, P.L., Costa, P., Fetzer, C., Felber, P., et al.: Endbox: Scalable middlebox functions using client-side trusted execution. In: Proceedings of the 48th International Conference on Dependable Systems and Networks. DSN. vol. 18 (2018)
27. Hammami, M., Chahir, Y., Chen, L.: Webguard: Web based adult content detection and filtering system. In: Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003). pp. 574–578 (Oct 2003). https://doi.org/10.1109/WI.2003.1241271
28. Hammami, M., Chahir, Y., Chen, L.: Webguard: a web filtering engine combining textual, structural, and visual content-based analysis. IEEE Transactions on Knowledge and Data Engineering **18**(2), 272–284 (Feb 2006). https://doi.org/10.1109/TKDE.2006.34
29. Han, J., Kim, S., Ha, J., Han, D.: Sgx-box: Enabling visibility on encrypted traffic using a secure middlebox module. In: Proceedings of the First Asia-Pacific Workshop on Networking. pp. 99–105. APNet'17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3106989.3106994, http://doi.acm.org/10.1145/3106989.3106994
30. Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., Del Cuvillo, J.: Using innovative instructions to create trustworthy software solutions. HASP@ ISCA **11** (2013)
31. Holz, R., Braun, L., Kammenhuber, N., Carle, G.: The ssl landscape: A thorough analysis of the x.509 pki using active and passive measurements. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference. pp. 427–444. IMC '11, ACM, New York, NY,

USA (2011). https://doi.org/10.1145/2068816.2068856, http://doi.acm.org/10.1145/2068816.2068856
32. Housley, R., Ford, W., Polk, T., Solo, D.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, RFC Editor (April 2002), http://www.rfc-editor.org/rfc/rfc3280.txt
33. Huang, L.S., Rice, A., Ellingsen, E., Jackson, C.: Analyzing forged ssl certificates in the wild. In: 2014 IEEE Symposium on Security and Privacy. pp. 83–97 (May 2014). https://doi.org/10.1109/SP.2014.13
34. Abstract Syntax Notation One (ASN.1): Specification of basic notation. Standard, International Telecommunication Union (Aug 2015)
35. Kuvaiskii, D., Chakrabarti, S., Vij, M.: Snort intrusion detection system with intel software guard extension (intel SGX). CoRR **abs/1802.00508** (2018), http://arxiv.org/abs/1802.00508
36. Loreto, S., Mattsson, J., Skog, R., Spaak, H., Druta, D., Hafeez, M.: Explicit trusted proxy in http/2.0. Internet-Draft draft-loreto-httpbis-trusted-proxy20-01, IETF Secretariat (February 2014), http://www.ietf.org/internet-drafts/draft-loreto-httpbis-trusted-proxy20-01.txt
37. McGrew, D., Wing, D., Gladstone, P.: TLS proxy server extension. Internet-Draft draft-mcgrew-tls-proxy-server-01, IETF Secretariat (July 2012), http://www.ietf.org/internet-drafts/draft-mcgrew-tls-proxy-server-01.txt
38. McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C.V., Shafi, H., Shanbhogue, V., Savagaonkar, U.R.: Innovative instructions and software model for isolated execution. HASP@ ISCA **10** (2013)
39. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The tamarin prover for the symbolic analysis of security protocols. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification. pp. 696–701. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
40. Murdoch, S.J., Anderson, R.: Tools and technology of internet filtering. Access denied: The practice and policy of global internet filtering **1**(1), 58 (2008)
41. Naylor, D., Schomp, K., Varvello, M., Leontiadis, I., Blackburn, J., López, D.R., Papagiannaki, K., Rodriguez Rodriguez, P., Steenkiste, P.: Multi-context TLS (mcTLS): Enabling secure in-network functionality in tls. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. pp. 199–212. SIGCOMM '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2785956.2787482, http://doi.acm.org/10.1145/2785956.2787482
42. Novotny, J., Tuecke, S., Welch, V.: An online credential repository for the grid: Myproxy. In: Proceedings 10th IEEE International Symposium on High Performance Distributed Computing. pp. 104–111 (2001). https://doi.org/10.1109/HPDC.2001.945181
43. Poddar, R., Lan, C., Popa, R.A., Ratnasamy, S.: Safebricks: Shielding network functions in the cloud. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI'18), Renton, WA (2018)
44. Polpinij, J., Chotthanom, A., Sibunruang, C., Chamchong, R., Puangpronpitag, S.: Content-based text classifiers for pornographic web filtering. In: 2006 IEEE International Conference on Systems, Man and Cybernetics. vol. 2, pp. 1481–1485 (Oct 2006). https://doi.org/10.1109/ICSMC.2006.384926
45. Polpinij, J., Sibunruang, C., Paungpronpitag, S., Chamchong, R., Chotthanom, A.: A web pornography patrol system by content-based analysis: In particular text and image. In: 2008 IEEE International Conference on Systems, Man and Cybernetics. pp. 500–505 (Oct 2008). https://doi.org/10.1109/ICSMC.2008.4811326

46. Rescorla, E.: The transport layer security (tls) protocol version 1.3. RFC 8446, RFC Editor (August 2018)
47. Sherry, J., Lan, C., Popa, R.A., Ratnasamy, S.: Blindbox: Deep packet inspection over encrypted traffic. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. pp. 213–226. SIGCOMM '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2785956.2787502, http://doi.acm.org/10.1145/2785956.2787502
48. Trach, B., Krohmer, A., Gregor, F., Arnautov, S., Bhatotia, P., Fetzer, C.: Shieldbox: Secure middleboxes using shielded execution. In: Proceedings of the Symposium on SDN Research. pp. 2:1–2:14. SOSR '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3185467.3185469, http://doi.acm.org/10.1145/3185467.3185469
49. Tuecke, S., Welch, V., Pearlman, D.E.L., , Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820, RFC Editor (June 2004), http://www.rfc-editor.org/rfc/rfc3820.txt

Appendix A Content Filtering Policy Language

In the proposed framework, we have defined a policy language to be used with the proxy certificate profile. In this section, we list the structure of that language.

The policy field of the proxy certificate extension is encoded as a string in the field *policy* in listing 1.1. This string is an encoding of the structure listed in listing 1.2. This structure is defined in the *Abstract Syntax Notation One (ASN.1)* [34] which is a standard interface description language. The structure consists of the fields mentioned in 3.3.

Listing 1.2. The definition of the policy structure in the ASN.1 notation before encoding.

```
1 FilteringPolicy ::= SEQUENCE {
2     allowedDomains        SEQUENCE(SIZE(0..255)) OF OCTET
          STRING,
3     disallowedDomains     SEQUENCE(SIZE(1..255)) OF OCTET
          STRING,
4     cipherSuites          SEQUENCE(SIZE(1..100)) OF
          CipherSuite,
5     -- Assuming there is an enum of type CipherSuite
          already defined.
6     trustLevel            INTEGER(0..1000)
7 }
```