The Age of Encryption

GIAC GCIA Gold Certification

Author: Wes Whitteker, wes_whitt@yahoo.com

Advisor: Richard Carbone

Accepted: November 2nd, 2016

Abstract

Over the last few years, there has been an increasing movement toward encrypting Internet communication. Though this movement increases the confidentiality of transmitted information, it also severely limits the ability of security tools to analyze Internet traffic for malicious content. This paper investigates the growth of encrypted Internet traffic (i.e. HTTPS) and its impact on Cybersecurity. This paper also proposes an open source solution for decrypting and inspecting Internet traffic accommodating IPv4 and v6 for both home and small-to-medium sized business (SMB) use.

1. Introduction

Since the beginning of the Internet, encryption has been in use to protect sensitive information such as financial data, personal data, etc. As the Internet has evolved, the use of encryption has been steadily increasing in parallel through the use of both the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols (Netcraft, n.d.). Over the last few years, this increase in SSL and TLS Internet communication has been very significant due in large part to disclosures and disputes that have created public concern for privacy. These privacy concerns have created additional momentum around the growth of SSL and TLS communication on the Internet (IETF, 2014 & Letsencrypt.org, n.d.).

This increased momentum for encrypting Internet communication is creating a significant dilemma for cybersecurity professionals. Since networking is the glue that pulls all computing technologies together, it serves as a key environment for cybersecurity professionals to identify malicious activity. However, the burst of growth in encrypted communication is creating "blind spots." Often times, cybersecurity professionals need deep visibility into the traffic traversing their networks in order to determine if communication is legitimate or illegitimate. Unfortunately, this analytical capability is severely hampered when communication is encrypted (Mayfield, 2015).

There have been several examples of areas where malicious actors and malware are leveraging the growth in encryption to their benefit. In a 2015 report, it was shown that malware command and control servers using SSL and TLS communication increased by 200 times, and malware samples using encrypted communication increased by 58 times. When considering these data points, it is clear that cybersecurity professionals are facing a significant challenge around encrypted communication analysis. Cybersecurity professionals can address this challenge by decrypting and inspecting SSL and TLS communication (A10 Networks, 2015, Davis, 2016, & Segura, 2015).

2. Overview of the Internet

To understand the current state of SSL and TLS communication, it is important to briefly look at the history of the Internet and its use of both SSL and TLS. The following subsections provide an overview of the creation of the World Wide Web (WWW) and the usage of SSL and TLS with the Hypertext Transfer Protocol (HTTP), commonly referred to as "HTTPS." For the remainder of the paper, SSL and TLS will be collectively referred to as "TLS" unless referred to specifically by version.

2.1. World Wide Web and HTTP Background

The inception of the Internet can be traced back to the Cold War, which created a sense of urgency within the United States to increase their focus on science and technology developments. This initiative resulted in the establishment of several key organizations and the publishing of several academic research papers that laid the foundation for packet switched networks. This foundation led to the standardization of the Transmission Control Protocol/Internet Protocol (TCP/IP) suite; the first Domain Name System (DNS); the Internet Assigned Numbers Authority (IANA); and the Internet Engineering Task Force (IETF) (History.com, n.d., Pew Research Center, n.d., World Wide Web Foundation, n.d., & World Wide Web Consortium, n.d.).

In 1991, CERN introduced the World Wide Web (WWW) to the masses (World Wide Web Foundation, n.d., & World Wide Web Consortium, n.d.). The component of the WWW that is used to transfer data is the Hyper Text Transfer Protocol (HTTP). The IETF began oversight of the HTTP in 1996 (IETF, 1996). The latest IETF recommended version of HTTP is 1.1, which became an IETF draft standard in June of 1999 (IETF, 1999 & W3C, 2004). The core of the HTTP 1.1 protocol is now covered as a proposed standard under two IETF Request for Comments (RFCs): RFC7230 & RFC7231 (IETF HTTP WG, n.d). The latest approach to HTTP is HTTP/2, which is an extension of the Google SPDY project. HTTP/2 is now an IETF proposed standard (IETF, 2015). The intent of HTTP/2 is to improve upon the overall performance of HTTP 1.1 (e.g. less TCP connections). However, HTTP/2 is being advertised by the IETF as an alternative to HTTP 1.1 instead of obsoleting it (IETF HTTP WG, n.d.a.).

HTTP/2 does not require encryption for use. However, there are no browsers currently being developed that support HTTP/2 without the use of encryption. HTTP/2 does seek to improve TLS based security by defining a profile that includes a specific version of TLS, a cipher suite blacklist, and extensions used (IETF HTTP WG, n.d.a.). Specifically, HTTP/2 requires that TLS 1.2 or greater is used, and recommends the use of RFC 7525, Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). In addition to the TLS version minimum being set to 1.2, HTTP/2 also uses a blacklist for cipher suites that can be found here: https://http2.github.io/http2-spec/#BadCipherSuites (IETF HTTP WG, n.d.b.).

Further, under HTTP/2, TLS must leverage Server Name Indication (SNI), which requires that a client specify the domain name with which it intends to communicate. SNI is explained under RFC 6066, Transport Layer Security (TLS) Extensions: Extension Definitions (IETF HTTP WG, n.d.b). If HTTP/2 uses TLS 1.3 or higher the use of only SNI is required. However, if TLS 1.2 is used, SNI is still required but there are additional restrictions. The restrictions under TLS 1.2 include disabling compression; disabling renegotiation (except prior to connection preface information); support for ephemeral key exchange sizes of at least 2048 bits for cipher suites that use ephemeral finite field Diffie-Hellman (DHE) and 224 bits for cipher suites that use ephemeral elliptic curve Diffie-Hellman (ECDHE); and clients need to accept DHE sizes of up to 4096 bits (IETF HTTP WG, n.d.b).

2.2. SSL and TLS Background

Though HTTP has done an outstanding job in formatting and transmitting information for the WWW, it only offers basic authentication capabilities and does not offer a mechanism for encryption (IETF, 1999b). The primary response to this was the development of SSL. In 1994, Netscape, supported by MasterCard, Bank of America, MDI & Silicon Graphics, introduced SSL (Hwang, 2012, Ristic, 2014, & Wireshark.org, n.d.). Today, the SSL and TLS (successor protocol to SSL) protocols provide the mechanisms necessary for confidentiality, authentication, integrity, and non-repudiation around much of the secure communications on the modern WWW (CIO.gov, n.d. & Evsslcertificate.com, n.d.).

The first public release of SSL was version 2 with version 3 being released in 1996 (IETF, 2011). It was at this time that the IETF began to officially acknowledge SSL by the establishment of the TLS working group (IETF TLS WG, n.d.). The TLS working group used SSL 3.0 as the foundation for developing the TLS 1.0 protocol. In 1999, the IETF published TLS 1.0 under RFC 2246 (IETF, 1999a). Since the initial release of TLS there have been 2 additional versions released, TLS 1.1 (RFC 4346) and TLS 1.2 (RFC 5246) (IETF, 2006 & IETF, 2008). Currently, there is a draft of TLS 1.3 (IETF, 2016). Due to security issues in previous versions of both SSL and TLS, the IETF encourages the use of TLS 1.2 through RFC 7568, which also officially deprecates SSL 3.0 (IETF, 2015a). Further, the IETF offers a specific RFC for using TLS in a secure manner (IETF, 2015b).

In addition to the IETF recommending the use of TLS version 1.2, the US National Institute of Standards and Technology (NIST) recommends the use of TLS version 1.2 as well (Chokhani, McKay, Polk, 2014). The recommendation for moving to TLS 1.2 is due, in large part, to security issues (wolfSSL.com, 2010). This can be seen when comparing the changes of TLS 1.1 and TLS 1.2. The following tables offer the major improvements in TLS versions 1.1 and 1.2 as well as the security updates to TLS 1.2:

Table 1: Major updates to TLS 1.1 and TLS 1.2 (IETF, 2006 & IETF, 2008)

TLS 1.1 Major Updates	***TLS 1.2 Major Updates***
The implicit Initialization Vector (IV) is replaced with an explicit IV to protect against CBC attacks.	The MD5/SHA-1 combination in the pseudorandom function (PRF) has been replaced with cipher-suite-specified PRFs. All cipher suites in this document use P_SHA256.
Handling of padding errors is changed to use the bad_record_mac alert rather than the decryption_failed alert to protect against CBC attacks.	The MD5/SHA-1 combination in the digitally-signed element has been replaced with a single hash. Signed elements now include a field that explicitly specifies the hash algorithm used.
IANA registries are defined for protocol parameters.	Substantial cleanup to the client's and server's ability to specify which hash and signature algorithms they will accept. Note that this also relaxes some of the constraints on signature and hash algorithms from previous versions of TLS.
Premature closes no longer cause a session to be nonresumable.	Addition of support for authenticated encryption with additional data modes.
Additional informational notes were added for various new attacks on TLS.	TLS Extensions definition and AES Cipher Suites were merged in from external [TLSEXT] and [TLSAES].
	Tighter checking of EncryptedPreMasterSecret version numbers.
	Tightened up a number of requirements.
	Verify_data length now depends on the cipher suite (default is still 12).

	Cleaned up description of Bleichenbacher/Klima attack defenses.
	Alerts MUST now be sent in many cases.
	After a certificate_request, if no certificates are available, clients now MUST send an empty certificate list.
	TLS_RSA_WITH_AES_128_CBC_SHA is now the mandatory to implement cipher suite.
	Added HMAC-SHA256 cipher suites.
	Removed IDEA and DES cipher suites. They are now deprecated and will be documented in a separate document.
	Support for the SSLv2 backward-compatible hello is now a MAY, not a SHOULD, with sending it a SHOULD NOT. Support will probably become a SHOULD NOT in the future.
	Added limited "fall-through" to the presentation language to allow multiple case arms to have the same encoding.
	Added an Implementation Pitfalls sections
	The usual clarifications and editorial work.

Table 2: Security updates to TLS 1.2 (IETF, 2008)

RFC Number	***RFC Title***
RFC 5746	TLS Renegotiation Indication Extension
RFC 5878	TLS Authorization Extensions
RFC 6176	Prohibiting SSL Version 2.0
RFC7465	Prohibiting RC4 Cipher Suites
RFC 7507	TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks
RFC 7568	Deprecating Secure Sockets Layer Version 3.0
RFC 7627	TLS Session Hash and Extended Master Secret Extension
RFC 7685	A TLS ClientHello Padding Extension
RFC 7905	ChaCha20-Poly1305 Cipher Suites for TLS
RFC 7919	Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for TLS

As can be clearly seen when looking at the improvements in the list of major updates to TLS 1.1 and 1.2, TLS version 1.2 has had a significant amount of work completed around its security functionality. In general, TLS 1.1 added support for TLS extensions and TLS 1.2 added support for authenticated encryption and removed hard-coded security features (Ristic, 2015).

2.3. HTTP and TLS

The parent IETF RFC that ties TLS to HTTP is 2818, though TLS can be used with several other popular Internet protocols (IETF, 2000 & Wireshark.org, n.d.). RFC 2818 has been updated by both RFCs 7230 and 5785. In addition, RFC 2817 provides guidance on upgrading an HTTP connection to an HTTPS connection, and RFC 2817 has been updated by both RFCs 7230 and 7231.

3. Technicalities of HTTPS Communication on the Internet

It is extremely import for cybersecurity professionals looking for a possible HTTPS decryption solution to understand the protocol at a detailed level. As such, the following section will cover the key technical aspects of the TLS protocol. The major areas of discussion will be the Public Key Infrastructure (PKI) and the TLS protocol.

3.1. Public Key Infrastructure (PKI)

Much of the HTTPS communication performed today relies on what is known as the Public Key Infrastructure (PKI). Today's PKI is based on an architecture standard known as X.509. At the core of the Internet's PKI are certificate authorities, which are trusted to issue certificates for use in several key TLS encryption functions. There are four key entities in PKI known as the Subscriber, Registration Authority (RA), Certification Authority (CA), and the Relying Party. The subscriber is an entity that desires to support secure communication (i.e. the website owner). An RA performs many of the management activities needed for certification issuance such as verifying an identity. A CA is the key to the Internet's PKI because it is trusted with issuing and revoking certificates. The relying party is normally associated with an end user's web browser. The web browser performs validation of certificates by leveraging a local root trust store. If a website's public certificate is received from a CA the browser has a root trust store for, the browser will trust the website certificate, otherwise the user will receive a browser message that the certificate is unknown or untrusted (Risitc, 2015).

3.1.1. Certificates

The overall reason the PKI has been built is to share certificates that are used to establish trust between a server and a web browser. Certificates are essentially a digital document with several specific fields and associated values depending on the version of the certificate. Certificates are formatted based on a standard known as the Abstract

Syntax Notation One (ASN.1) and often communicated in Privacy-Enhanced Mail (PEM) encoding (Ristic, 2015). Table 4 provides the key fields seen in a certificate.

Table 3: Key Certificate Fields (Ristic, 2015)

Field Name	**Field Description**
Version	Provides the specific certificate version of either 1, 2, or 3.
Serial Number	Uniquely identifies the CA of the certificate.
Signature Algorithm	Provides the signature algorithm used for the certificate signature.
Issuer	Contains a distinguished name (DN) of the certificate issuer.
Validity	Provides the length of time that the certificate is valid via both a start and end date.
Subject	The DN for the entity associated with the public key. This field has been deprecated for Subject Alternative Name Extension Field.
Public Key	The public key.
Extensions	Extensions were added in certificate version 3 to ensure future flexibility. There are now several extensions available for use including: Subject Alternative Name, Name Constraints, Basic Constraints, Key Usage, Extended Key Usage, Certificate Policies, CRL Distribution Points, Authority Information Access, Subject Key Identifier, and Authority Key Identifier.

Certificates are issued using a specific process associated with the type of certificate requested. There are three processes used for certificate issuance: Domain Validation (DV), Organizational Validation (OV), and Extended Validation (EV). EV is the most rigorous and was developed out of security issues with the other two processes. Once the appropriate validation process is completed, the requesting party receives the certificate. In order for a relying entity to validate certificates, they maintain a list of "trusted" root CA certificates. In general, most applications (e.g. browsers) rely on the underlying operating system to manage certificate trust store. However, there are cases where applications will not rely on the OS trust store (e.g. Mozilla & Chrome) (Ristic, 2015).

3.1.2. Certificate Revocation, HTTP Strict Transport Security (HSTS), & Certificate Pinning

The two methods of certificate revocation are Certificate Revocation Lists (CRL) and the Online Certificate Status Protocol (OCSP). The CRL provides a list of revoked certificates that are still unexpired – these are large lists. The OCSP addresses the CRL issue of large lists by looking up a certificate's status in real-time. To address some of the security concerns around the TLS and the current PKI architecture, there have been several initiatives to improve these weaknesses. Two of the most significant developments have been HTTP Strict Transport Security (HSTS) and Certificate Pinning (Ristic, 2015).

HSTS is a standard covered under IETF RFC 6797 and is focused on ensuring the confidentiality and authenticity of communication between a browser and a server. This capability mitigated the SSL Stripping attack presented by Moxie Marlinspike at Blackhat in 2009 (Marlinspike, n.d.). The two key functions of HSTS are it forces the browser to use HTTPS communication instead of HTTP and it does not allow users to click through browser certificate warnings. In terms of decrypting TLS traffic, this could create some issues (e.g. browser security warnings) depending on the specific architecture. The simplest way to mitigate possible issues is to leverage a Man-In-The-Middle (MITM) decryption solution and install a trusted certificate used by all internal system browser-trusted stores (superconfigure.wordpress.com, 2013).

The key feature with pinning is that it allows domain owners to specify specific CAs that are authorized to issue certificates for their domains. This prevents an illegitimate entity from creating unauthorized certificates for a domain of their choosing. There are 2 basic types of certificate pinning: Hard Certificate Pinning and CA Pinning. Hard certificate pinning requires the exact match of the server certificate presented to the client or the communication is prevented. Hard certificate pinning is rarely used because many enterprises block it and there could be complications if the hard coded certificate needs changed. CA Pinning, on the other hand, trusts a limited set of CAs. As such, this type of pinning can be successfully decrypted using a root certificate that is manually added to system browser-trusted stores. This allows MITM capabilities for encrypted traffic inspection (Bluecoat, 2016 & Ristic, 2015).

3.2. The TLS Protocol

From a high level, TLS is broken into two major protocols known as the record protocol and the handshake protocol. The record protocol manages the transportation, encryption and integrity, compression, and extensibility feature of the TLS protocol. The handshake protocol negotiates the connection settings and authentication between the client and server. The handshake protocol consists of three "subprotocols" referred to as the change cipher spec protocol, application data protocol, and the alert protocol. The handshake protocol can take three forms: full handshake including server authentication, handshake with client/server authentication, and abbreviated handshake (resuming an earlier session) (Ristic, 2015).

The main difference between a full handshake with server only authentication and a full handshake with both server and client authentication (i.e. mutual authentication) is in the latter method the server requests a certificate from the client as part of its handshake messages. In response to the servers' additional request, the client provides its certificate and verifies it is the owner of the certificate. The handshake that happens during a session resumption leverages the Session IDs contained in the ClientHello and ServerHello messages. The benefit of this handshake method is that it saves time in establishing a new session by cutting the amount of handshake message in half. The most common handshake is the full handshake with server authentication. Figure 1 provides a depiction of the TLS handshake and Table 4 provides its description.

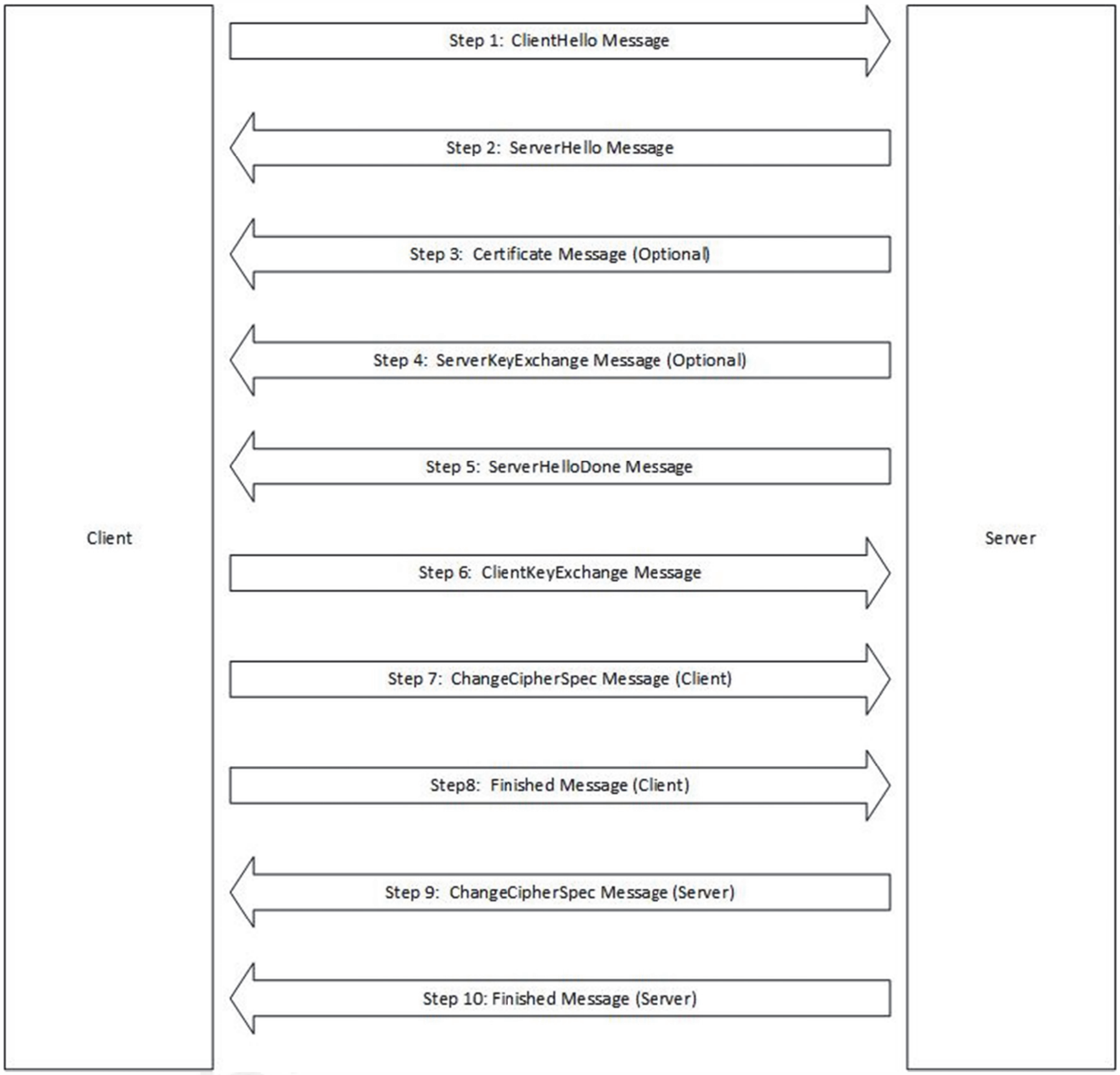

Client
Server
Step 1: ClientHello Message
Step 2: ServerHello Message
Step 3: Certificate Message (Optional)
Step 4: ServerKeyExchange Message (Optional)
Step 5: ServerHelloDone Message
Step 6: ClientKeyExchange Message
Step 7: ChangeCipherSpec Message (Client)
Step8: Finished Message (Client)
Step 9: ChangeCipherSpec Message (Server)
Step 10: Finished Message (Server)

Figure 1: TLS Handshake (Ristic, 2015)

Table 4: Handshake Description (Ristic, 2015)

Step Number	Step Description
Step 1	The client sends a ClientHello message, which includes the following fields: Protocol Version, Random, Session ID, Cipher Suites, Compression, and Extensions. The protocol version provides the latest TLS protocol version supported by the client. The random field provides randomness during the handshake, which assists in authentication and integrity functions. The session ID is used when session is resumed but blank on an initial connection. The Cipher Suites field provides the list of client supported suites. Compression provides a listing of client supported compression methods. Extensions provide a listing of additional functionality used within the current communication session. The latest list of extensions can be found here: http://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml.
Step 2	The server sends a ServerHello message, which is like the ClientHello message in Step 1 except it limits the options per field to one value.

Step 3	The certificate message is an optional message since not all cipher suites rely on authentication and some forms of authentication do not use certificates. When certificates are used this message normally contains the server's X.509 certificate chain for identification (main certificate and all intermediary certificates) but other forms of identification could be used such as a PGP key. The certificate must support the selected cipher suite. If a server supports multiple cipher suites, it may need multiple certificates.
Step 4	The ServerKeyExchange message is optional. If used, it depends on the selected cipher suite and carries server data used to assist in establishing the key exchange method.
Step 5	The ServerHelloDone message informs the client that the server is finished sending handshake information.
Step 6	The ClientKeyExchange message is dependent on the cipher suite in use and provides the client's information supporting the key exchange process.
Step 7	The ChangeCipherSpec message informs each respective side that the other side is switching to encrypted communication.
Step 8	Provides the notification to the other side that all requirements for a successful handshake have been met. A final verification is performed (via a MAC) ensuring that nothing has compromised the integrity of the entire handshake process.
Step 9	Same as Step 7.
Step 10	Same as Step 8.

4. Decryption Solution

4.1. Solution Selection

There are several open source projects underway that offer the ability to design a decryption solution for specific environments. Some of these projects include the following: Mitmproxy, Sslsniff, SSLSplit, SSLDump, and Fiddler. For the solution proposed by this paper, the following requirements were used as selection criteria: open source software (OSS); actively supported; scalable for home and SMB use; transparently decrypt and inspect egress HTTPS traffic (i.e. Forward Proxy); support for IPv4 & v6; support TLS 1.2; support for the most common TLS Ciphers, and easily configurable. Taking these requirements into consideration, the most sensible solution was to set up a pfSense firewall running Squid Proxy as a Man-In-The-Middle (MITM). With the MITM configuration the Internet Content Adaptation Protocol (ICAP), discussed in RFC 3507 can be used to analyze decrypted HTTPS content. The specific implementation of ICAP used in Squid Proxy is C-ICAP (C-ICAP project, n.d.). Essentially, pfSense intercepts the HTTPS traffic on port 443 and routes it through Squid Proxy, which performs the decryption and analysis of the HTTP content.

4.2. Conceptual Network Architecture

A diagram of the test network architecture is shown in Figure 2. As can be seen, all Internet network traffic is routed directly through the pfSense system. This solution requires the generation of a certificate used by all internal system browser-trusted stores, which is a common approach for a decryption solution. The major drawback to this solution is the inability to dynamically intercept HTTPS traffic on ports other than 443 (some commercial solutions do offer this ability).

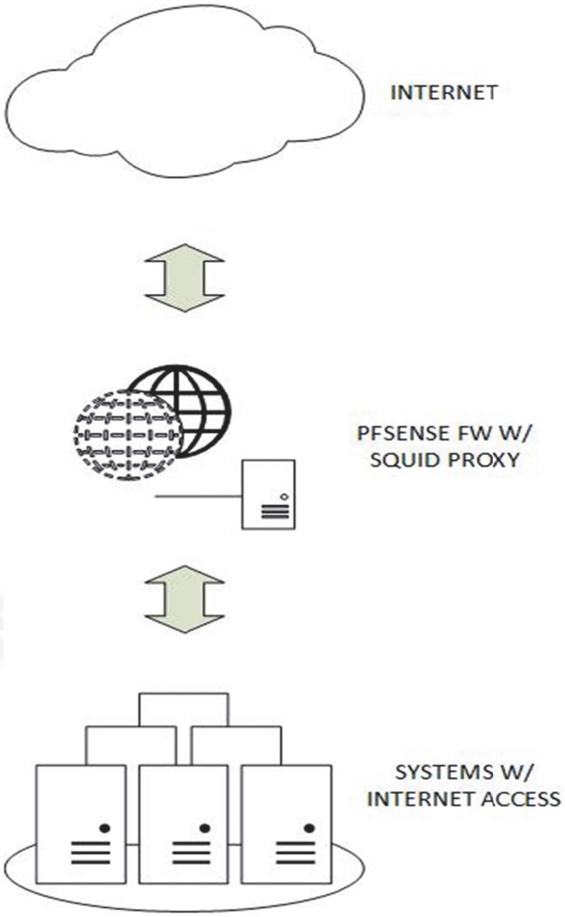


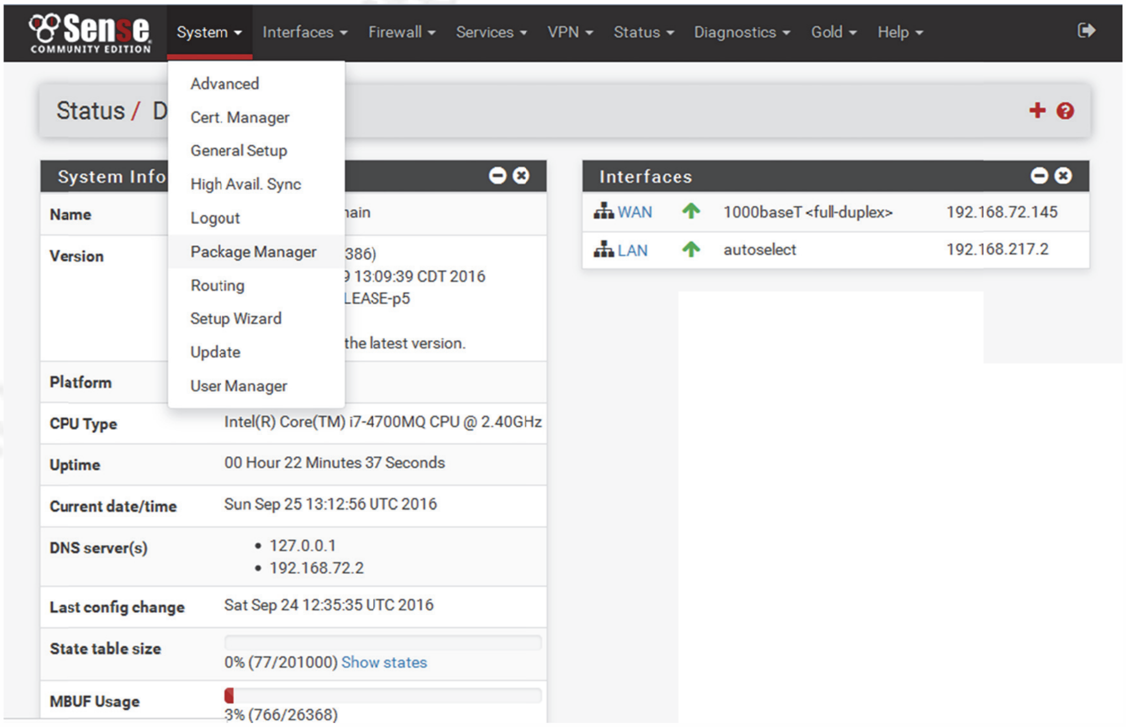
Figure 2: Test Network Architecture

4.3. Solution Setup and Configuration

PfSense is a very popular FreeBSD-based firewall and router that offers functional extensibility through additional packages (pfSense, n.d.). The latest version of pfSense can be downloaded from: https://www.pfsense.org/download/. One of the packages that comes integrated with pfSense is Squid Proxy. Squid Proxy is a very popular software proxy that has the ability to decrypt TLS sessions (Squid-cache.org, n.d.). One key reason Squid Proxy incorporates decryption capability is to analyze HTTPS traffic for malicious content.

When setting up this solution you have to consider the resources needed for both the pfSense firewall and the Squid Proxy. The hardware specifications for a pfSense firewall installation are located here: https://www.pfsense.org/hardware/. There is no "master" specification listing for Squid Proxy because every environment varies significantly. However, Squid Proxy is very CPU and Memory intensive. To this end, there are a few webpages that provide an overview for understanding what HW requirements might be: http://wiki.squid-cache.org/BestOsForSquid; http://wiki.squid-cache.org/SquidFaq/SquidMemory; and http://wiki.squid-cache.org/SquidFaq/SquidProfiling.

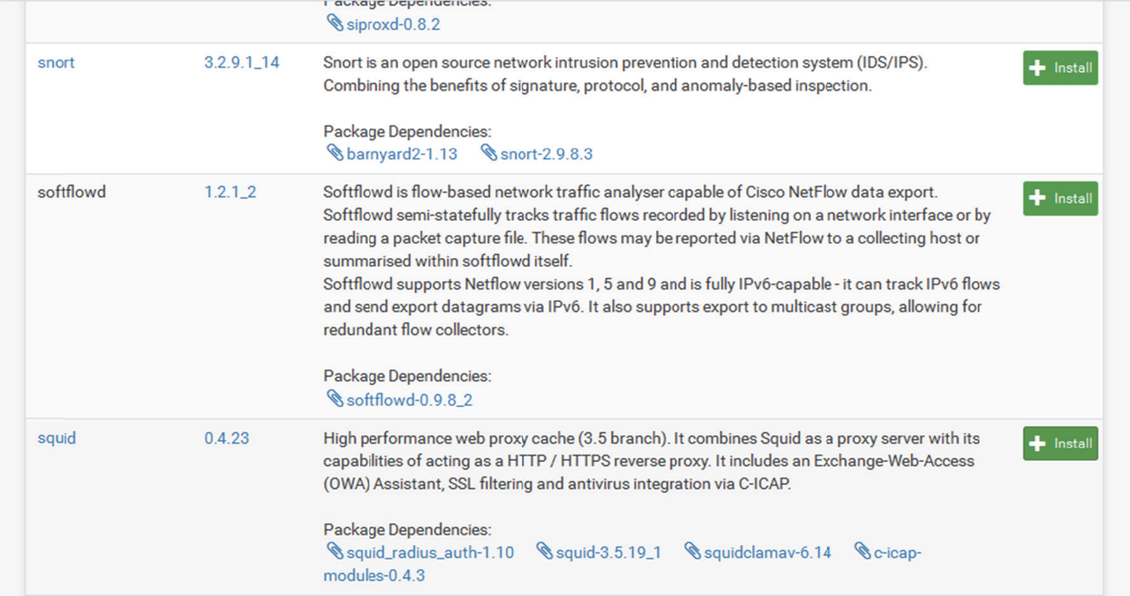
The following configuration and setup information assumes that the basic pfSense installation has already been completed. With the pfSense system completely configured for basic functionality, it is now time to setup Squid Proxy for decrypting TLS communication. The first step is to add the Squid Proxy package to pfSense, which is simple with the pfSense package manager. Start by navigating to System > Package Manager.



Name	ain
Version	386) 9 13:09:39 CDT 2016 LEASE-p5 the latest version.
Platform	
CPU Type	Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz
Uptime	00 Hour 22 Minutes 37 Seconds
Current date/time	Sun Sep 25 13:12:56 UTC 2016
DNS server(s)	• 127.0.0.1 • 192.168.72.2
Last config change	Sat Sep 24 12:35:35 UTC 2016
State table size	0% (77/201000) Show states
MBUF Usage	3% (766/26368)

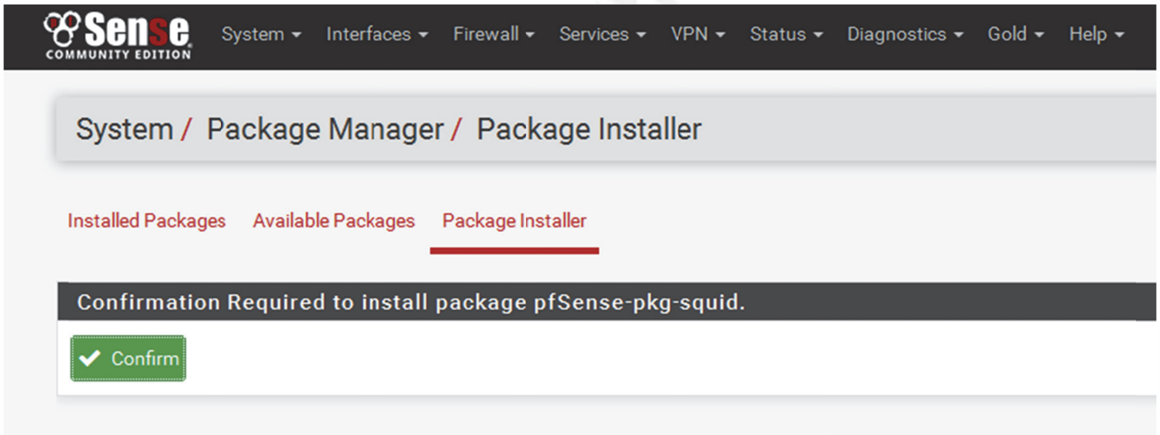
WAN	1000baseT <full-duplex>	192.168.72.145
LAN	autoselect	192.168.217.2

Select the “+ Install” button beside the squid package.



		Package Dependencies: siproxd-0.8.2	
snort	3.2.9.1_14	Snort is an open source network intrusion prevention and detection system (IDS/IPS). Combining the benefits of signature, protocol, and anomaly-based inspection. Package Dependencies: barnyard2-1.13 snort-2.9.8.3	+ Install
softflowd	1.2.1_2	Softflowd is flow-based network traffic analyser capable of Cisco NetFlow data export. Softflowd semi-statefully tracks traffic flows recorded by listening on a network interface or by reading a packet capture file. These flows may be reported via NetFlow to a collecting host or summarised within softflowd itself. Softflowd supports Netflow versions 1, 5 and 9 and is fully IPv6-capable - it can track IPv6 flows and send export datagrams via IPv6. It also supports export to multicast groups, allowing for redundant flow collectors. Package Dependencies: softflowd-0.9.8_2	+ Install
squid	0.4.23	High performance web proxy cache (3.5 branch). It combines Squid as a proxy server with its capabilities of acting as a HTTP / HTTPS reverse proxy. It includes an Exchange-Web-Access (OWA) Assistant, SSL filtering and antivirus integration via C-ICAP. Package Dependencies: squid_radius_auth-1.10 squid-3.5.19_1 squidclamav-6.14 c-icap-modules-0.4.3	+ Install

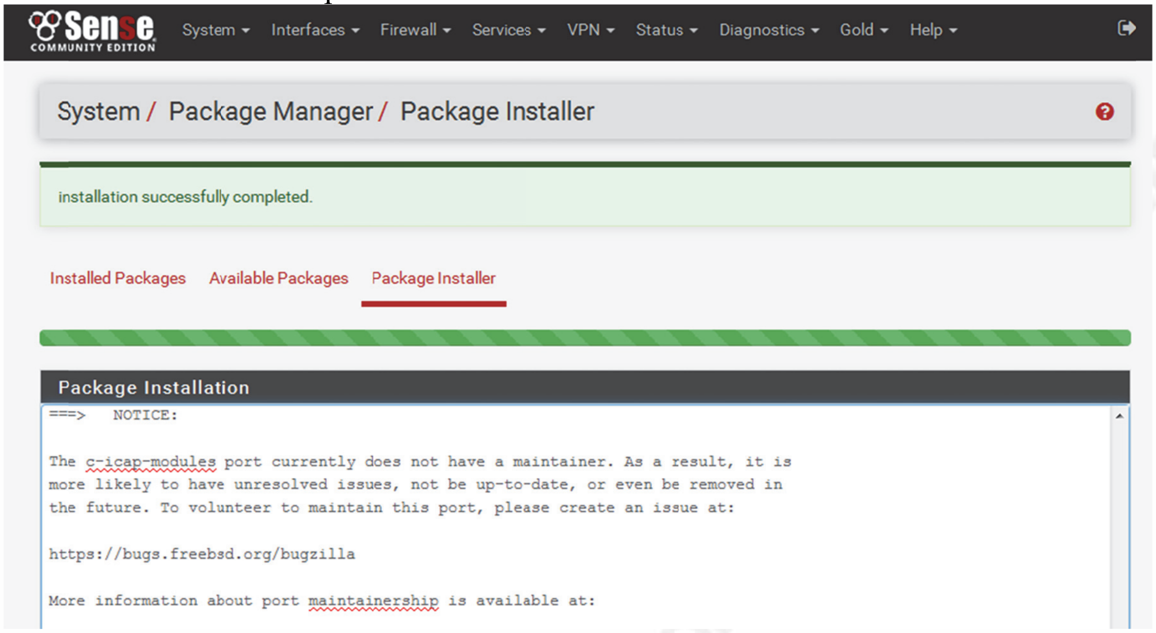
Select the “Confirm” button.



System / Package Manager / Package Installer

Confirmation Required to install package pfSense-pkg-squid.

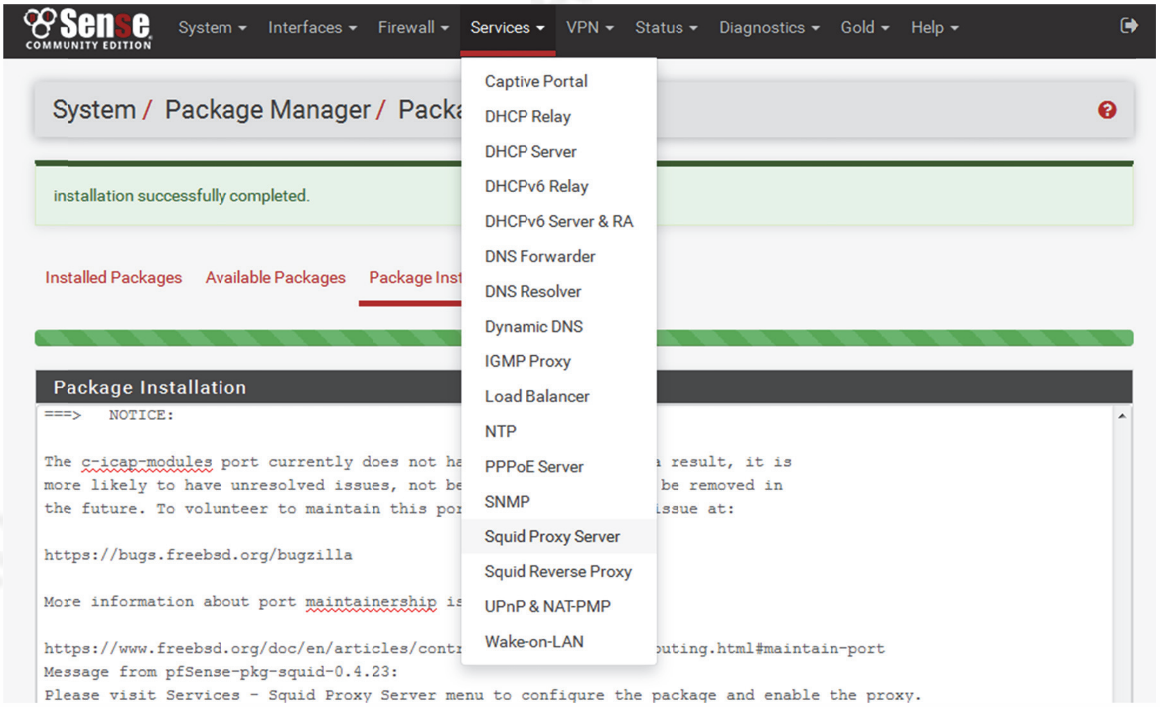
The installation will complete.



installation successfully completed.

```
===>   NOTICE:

The c-icap-modules port currently does not have a maintainer. As a result, it is
more likely to have unresolved issues, not be up-to-date, or even be removed in
the future. To volunteer to maintain this port, please create an issue at:

https://bugs.freebsd.org/bugzilla

More information about port maintainership is available at:
```

Now that Squid Proxy has been installed, it needs to be configured. Navigate to Services > Squid Proxy Server.

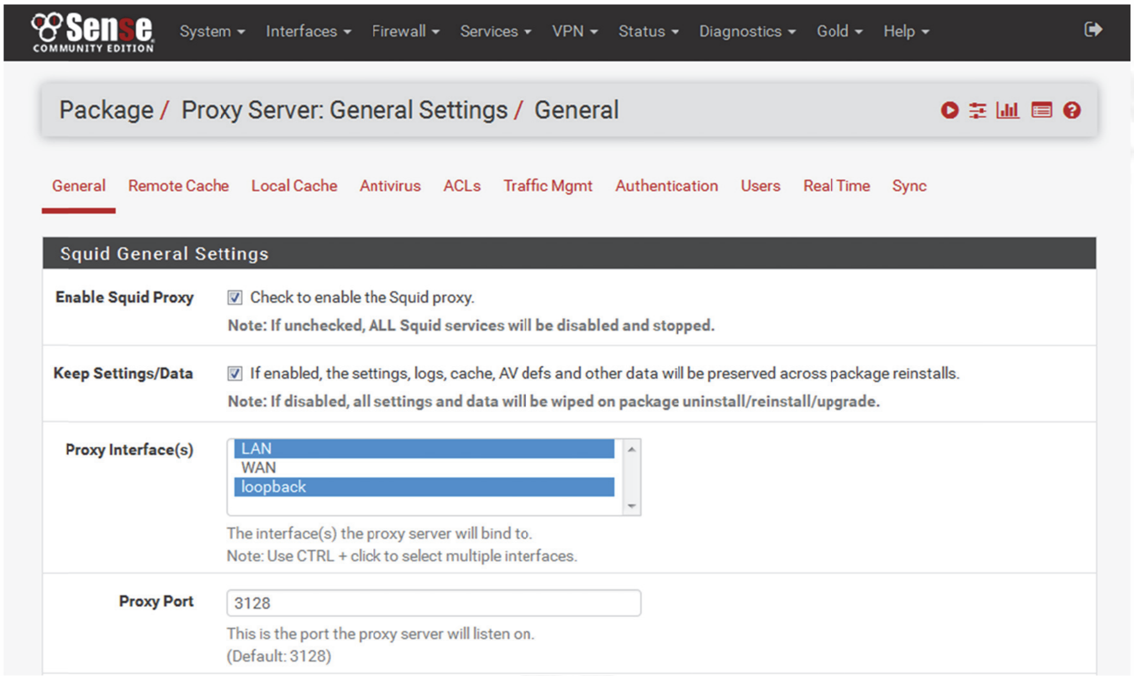


installation successfully completed.

```
===>   NOTICE:

The c-icap-modules port currently does not ha          a result, it is
more likely to have unresolved issues, not be          be removed in
the future. To volunteer to maintain this por          issue at:

https://bugs.freebsd.org/bugzilla

More information about port maintainership is

https://www.freebsd.org/doc/en/articles/contr          outing.html#maintain-port
Message from pfSense-pkg-squid-0.4.23:
Please visit Services - Squid Proxy Server menu to configure the package and enable the proxy.
```

Once within the Services > Squid Proxy Server section, the following Squid General Settings should be selected: Enable Squid Proxy; Keep Settings/Data; and LAN &

Loopback Proxy Interfaces. Nothing else needs to be completed within the General Settings section.



Package / Proxy Server: General Settings / General

Squid General Settings

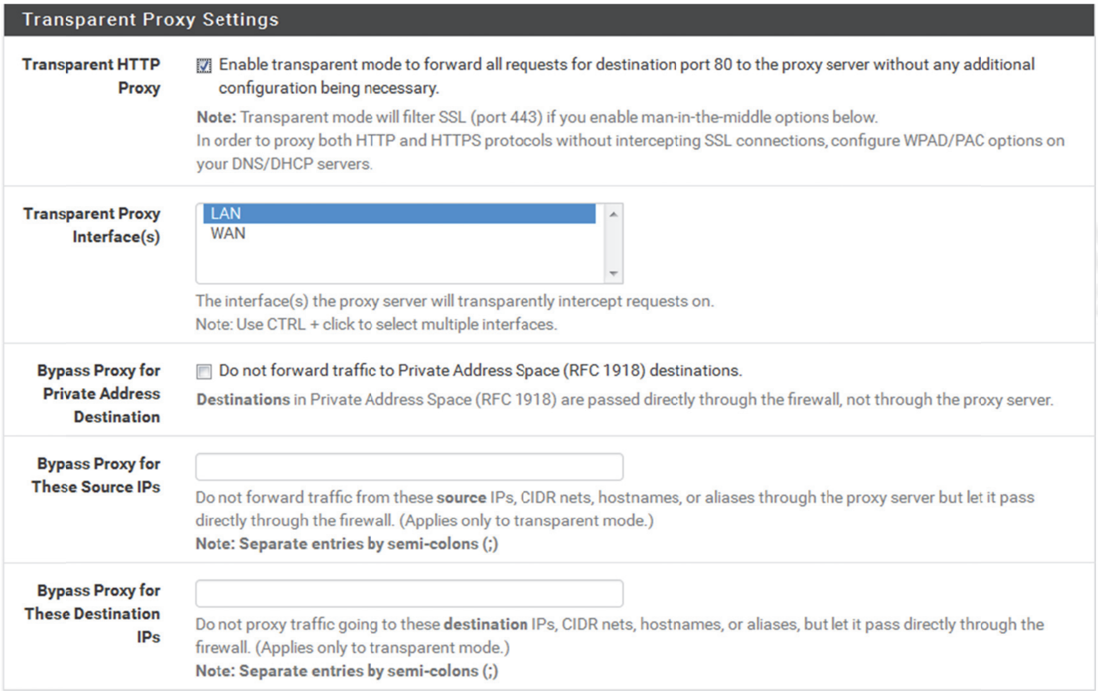
Enable Squid Proxy ☑ Check to enable the Squid proxy.
Note: If unchecked, ALL Squid services will be disabled and stopped.

Keep Settings/Data ☑ If enabled, the settings, logs, cache, AV defs and other data will be preserved across package reinstalls.
Note: If disabled, all settings and data will be wiped on package uninstall/reinstall/upgrade.

Proxy Interface(s): LAN (selected), WAN, loopback (selected)
The interface(s) the proxy server will bind to.
Note: Use CTRL + click to select multiple interfaces.

Proxy Port: 3128
This is the port the proxy server will listen on.
(Default: 3128)

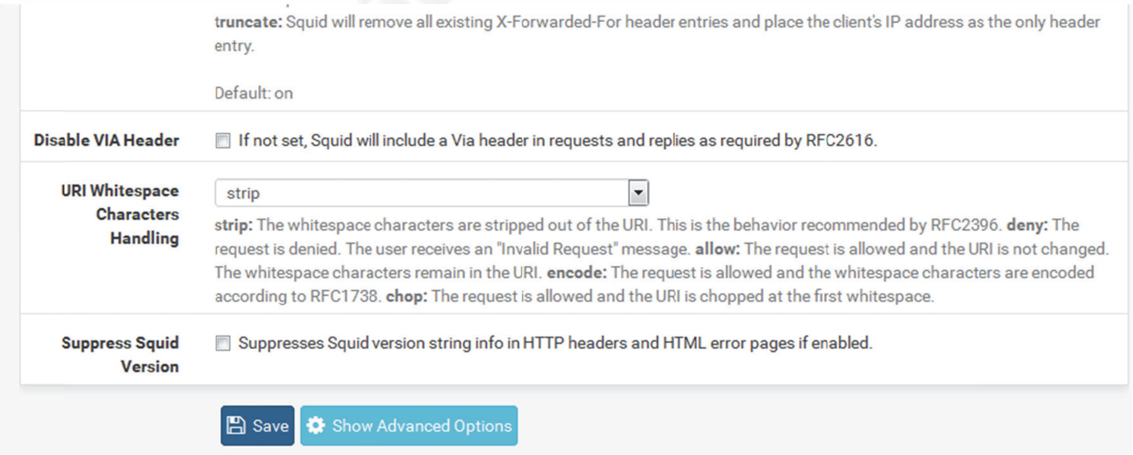
In the next section, Transparent Proxy Settings, enable Transparent HTTP Proxy and ensure LAN is the selected interface. This will allow for the seamless proxying of traffic on port 443 (i.e. TLS). If Transparent HTTP Proxy is not enabled, each browser would need to be configured to use the proxy and its assigned port.



Transparent Proxy Settings

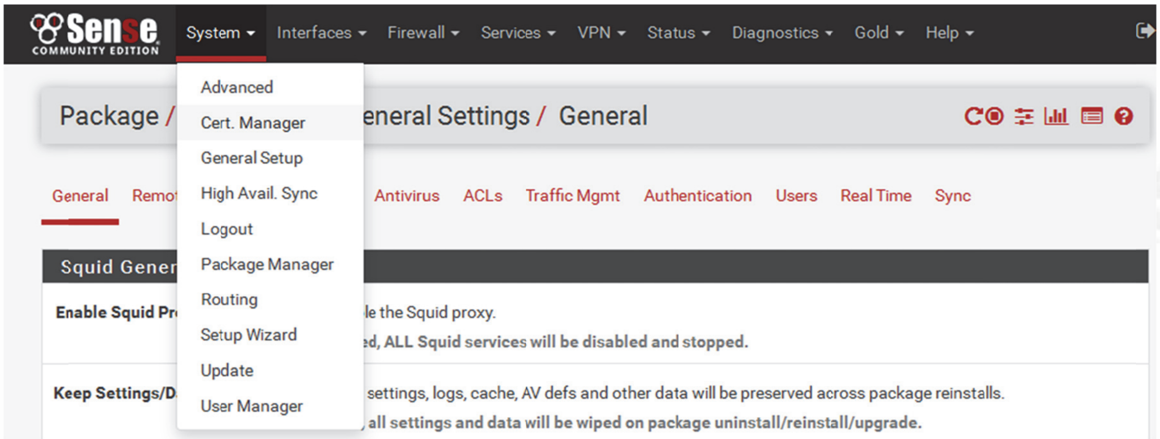
Transparent HTTP Proxy	☑ Enable transparent mode to forward all requests for destination port 80 to the proxy server without any additional configuration being necessary. **Note:** Transparent mode will filter SSL (port 443) if you enable man-in-the-middle options below. In order to proxy both HTTP and HTTPS protocols without intercepting SSL connections, configure WPAD/PAC options on your DNS/DHCP servers.
Transparent Proxy Interface(s)	LAN WAN The interface(s) the proxy server will transparently intercept requests on. Note: Use CTRL + click to select multiple interfaces.
Bypass Proxy for Private Address Destination	□ Do not forward traffic to Private Address Space (RFC 1918) destinations. **Destinations** in Private Address Space (RFC 1918) are passed directly through the firewall, not through the proxy server.
Bypass Proxy for These Source IPs	Do not forward traffic from these **source** IPs, CIDR nets, hostnames, or aliases through the proxy server but let it pass directly through the firewall. (Applies only to transparent mode.) **Note: Separate entries by semi-colons (;)**
Bypass Proxy for These Destination IPs	Do not proxy traffic going to these **destination** IPs, CIDR nets, hostnames, or aliases, but let it pass directly through the firewall. (Applies only to transparent mode.) **Note: Separate entries by semi-colons (;)**

Once the general and transparent proxy settings are set, they need to be saved. Note: You may need to save the Local Cache settings before you can save the settings for the General and Transparent Proxy settings. At this point, you should test that your internal system/s web browsers can still access the Internet.

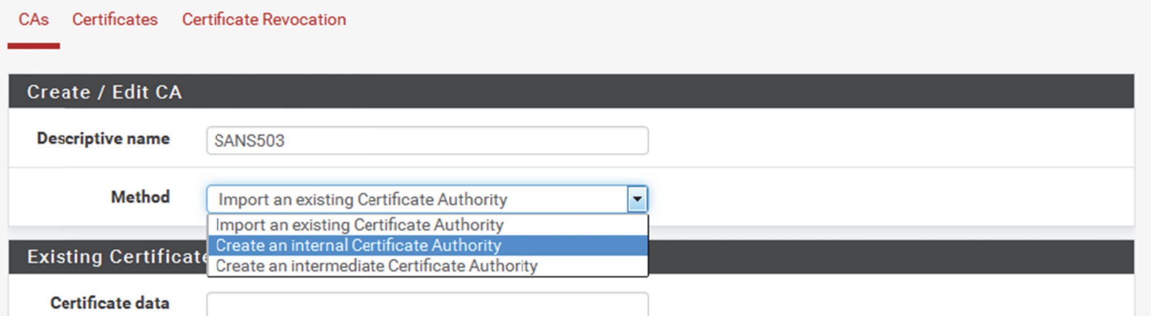


	truncate: Squid will remove all existing X-Forwarded-For header entries and place the client's IP address as the only header entry. Default: on
Disable VIA Header	□ If not set, Squid will include a Via header in requests and replies as required by RFC2616.
URI Whitespace Characters Handling	strip **strip:** The whitespace characters are stripped out of the URI. This is the behavior recommended by RFC2396. **deny:** The request is denied. The user receives an "Invalid Request" message. **allow:** The request is allowed and the URI is not changed. The whitespace characters remain in the URI. **encode:** The request is allowed and the whitespace characters are encoded according to RFC1738. **chop:** The request is allowed and the URI is chopped at the first whitespace.
Suppress Squid Version	□ Suppresses Squid version string info in HTTP headers and HTML error pages if enabled.

With the basic proxy functionality operational, it is now time to configure the Squid Proxy Man-In-The-Middle functionality. The first step is to setup pfSense as a Certificate Authority (CA). Navigate to System > Cert. Manager.



Ensure CA is selected and click the "+ Add" button. Fill in the desired descriptive name and select the desired method. For the test environment, Create an internal Certificate Authority is selected.



Create / Edit CA

Descriptive name: SANS503

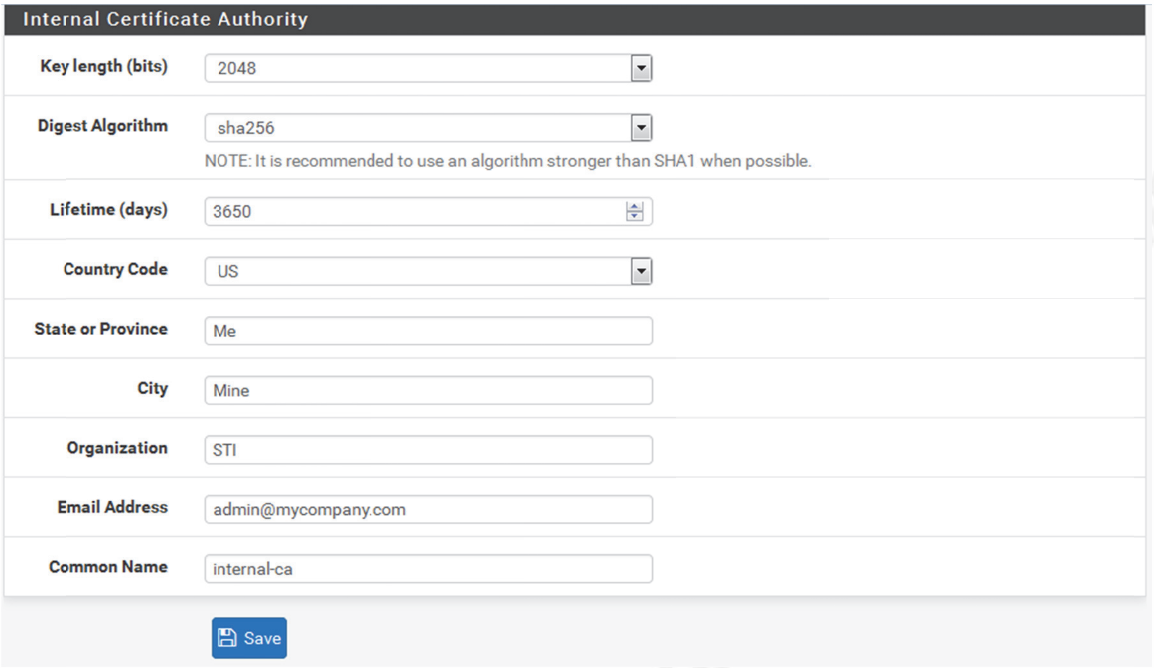
Method: Import an existing Certificate Authority

- Import an existing Certificate Authority
- Create an internal Certificate Authority
- Create an intermediate Certificate Authority

Existing Certificate

Certificate data

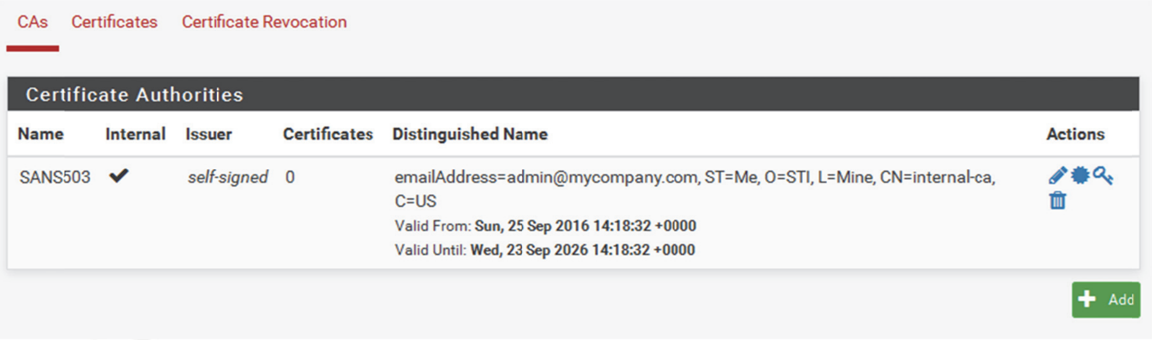
For the actual CA settings, the defaults are used and the location information is filled in. Once finished click Save.



Internal Certificate Authority

Key length (bits)	2048
Digest Algorithm	sha256 NOTE: It is recommended to use an algorithm stronger than SHA1 when possible.
Lifetime (days)	3650
Country Code	US
State or Province	Me
City	Mine
Organization	STI
Email Address	admin@mycompany.com
Common Name	internal-ca

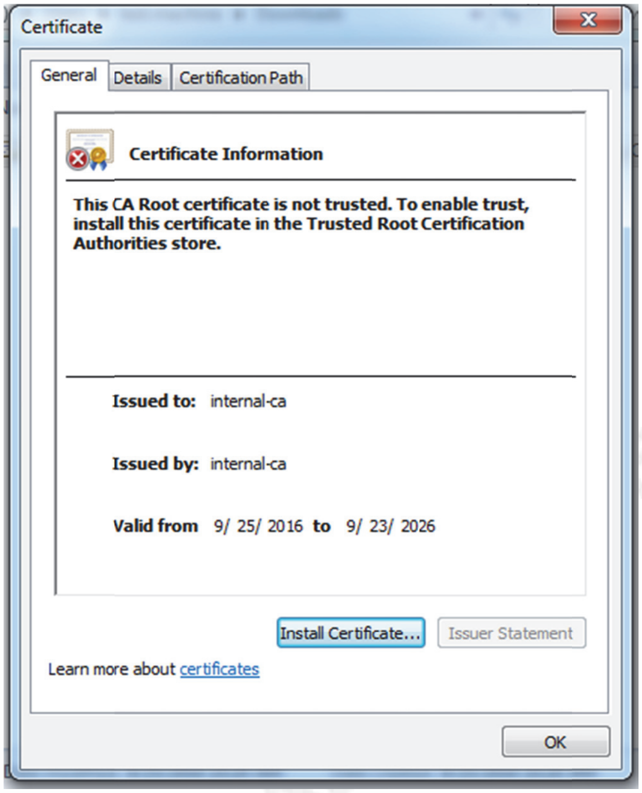
Now the CA is setup with its root certificate, which will need to be exported and installed in all browser Trusted Root Certificate stores. Select the export CA button to export the *.crt* certificate.



Certificate Authorities

Name	Internal	Issuer	Certificates	Distinguished Name	Actions
SANS503	✓	*self-signed*	0	emailAddress=admin@mycompany.com, ST=Me, O=STI, L=Mine, CN=internal-ca, C=US Valid From: **Sun, 25 Sep 2016 14:18:32 +0000** Valid Until: **Wed, 23 Sep 2026 14:18:32 +0000**	

The exported *.crt* certificate needs to be installed in the browser Trusted Root Certification store to prevent browser warnings since MITM proxying is being used. This can be completed manually or automated for larger environments. The manual steps for Windows 7 are described. On the system where you want the certificate installed, download the certificate to the system. Double click on the certificate and select Install Certificate.


Certificate

Certificate Information

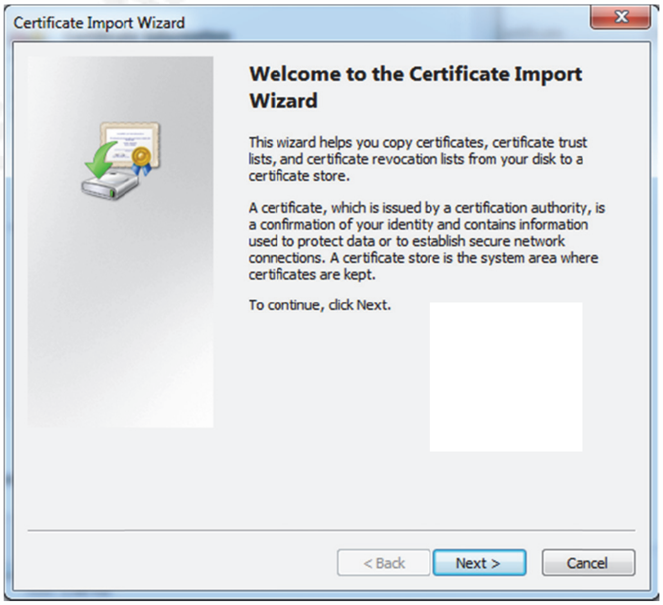
This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: internal-ca

Issued by: internal-ca

Valid from 9/ 25/ 2016 to 9/ 23/ 2026

Click Next.


Certificate Import Wizard

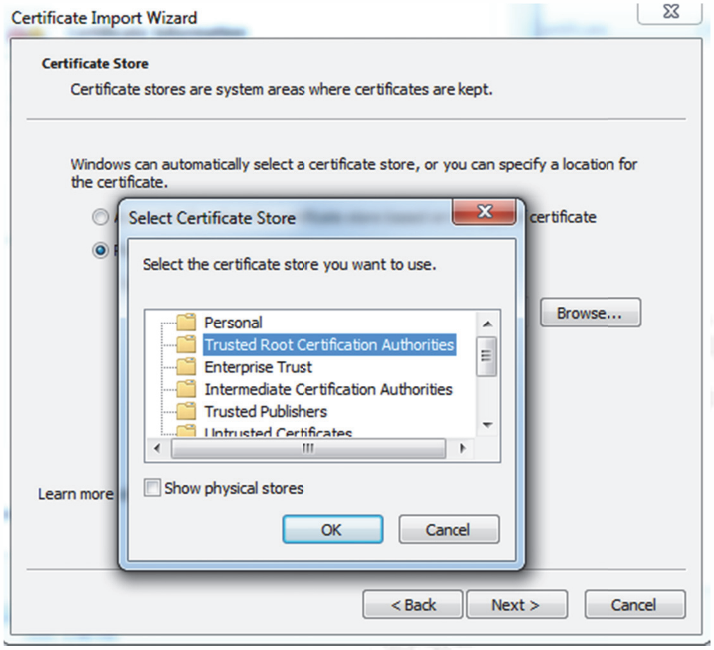
Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Select “Trusted Root Certification Authorities”.



Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

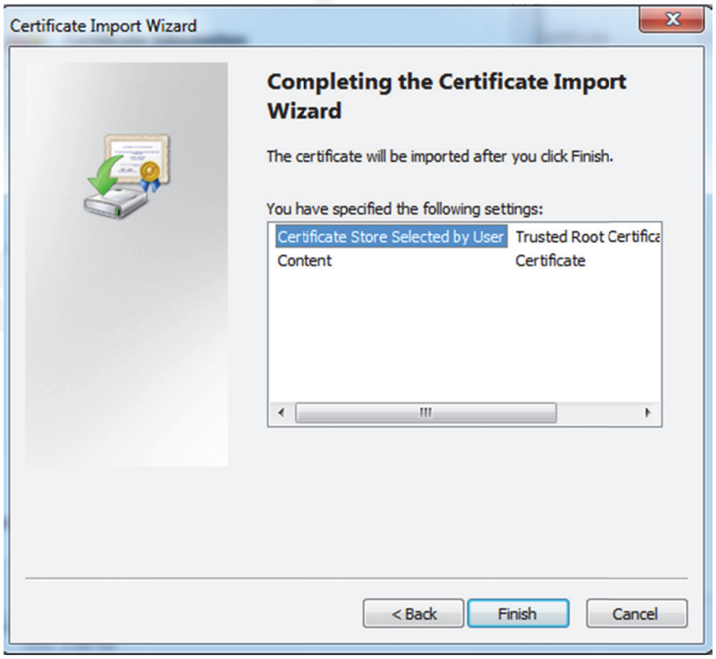
Select Certificate Store

Select the certificate store you want to use.

- Personal
- Trusted Root Certification Authorities
- Enterprise Trust
- Intermediate Certification Authorities
- Trusted Publishers
- Untrusted Certificates

□ Show physical stores

Click Finish.



Certificate Import Wizard

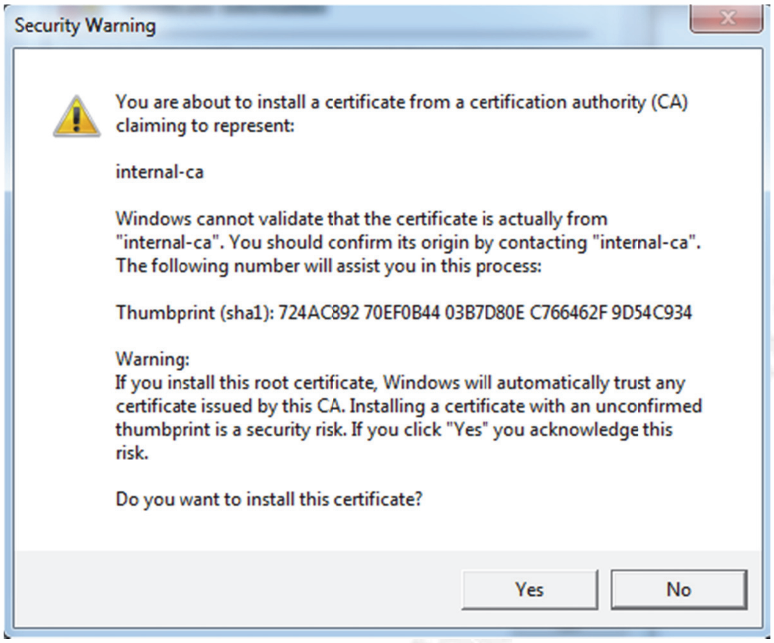
Completing the Certificate Import Wizard

The certificate will be imported after you click Finish.

You have specified the following settings:

Certificate Store Selected by User	Trusted Root Certifica
Content	Certificate

Select Yes.



You are about to install a certificate from a certification authority (CA) claiming to represent:

internal-ca

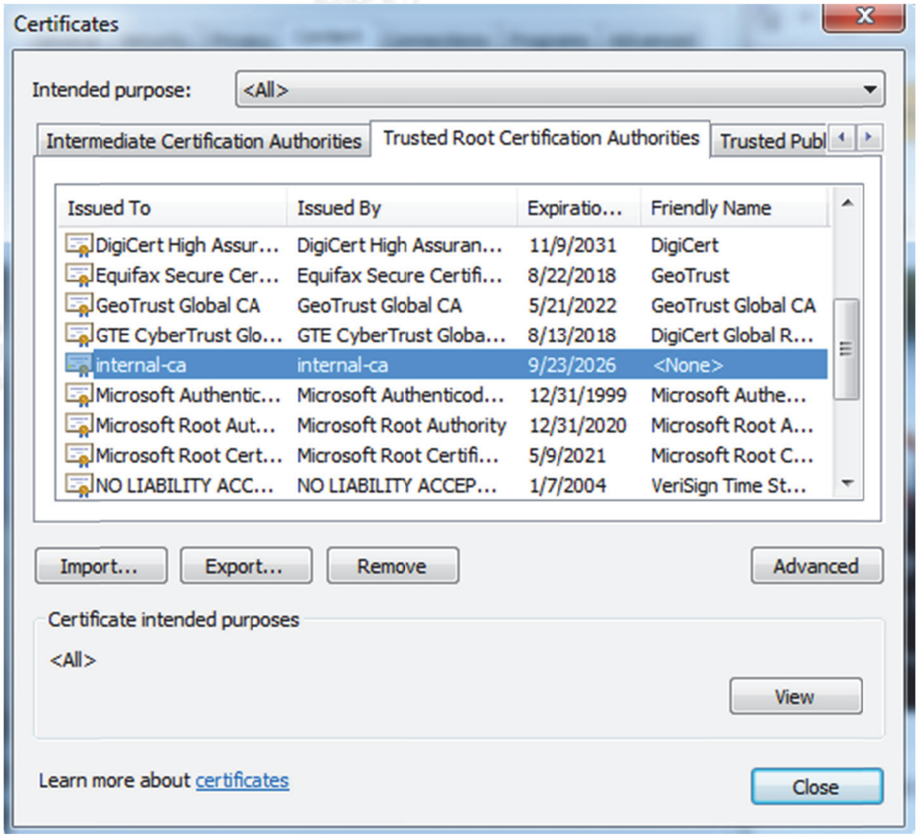
Windows cannot validate that the certificate is actually from "internal-ca". You should confirm its origin by contacting "internal-ca". The following number will assist you in this process:

Thumbprint (sha1): 724AC892 70EF0B44 03B7D80E C766462F 9D54C934

Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

Do you want to install this certificate?

Go into Internet Explorer and verify the certificate in the Trusted Root Certification Authorities store.

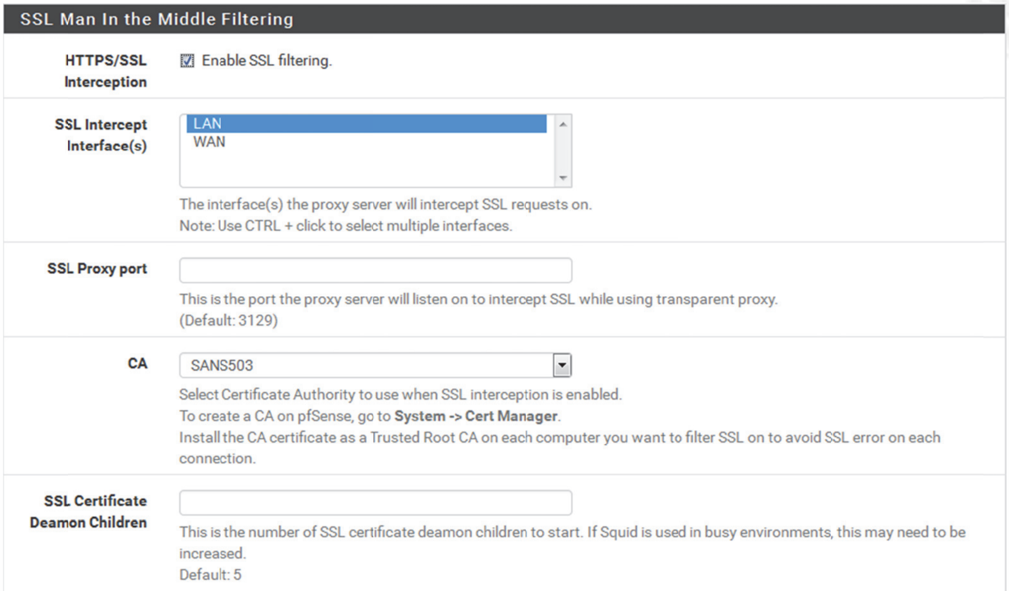


Certificates
Intended purpose:
<All>
Intermediate Certification Authorities
Trusted Root Certification Authorities
Trusted Publ

Issued To	Issued By	Expiratio...	Friendly Name
DigiCert High Assur...	DigiCert High Assuran...	11/9/2031	DigiCert
Equifax Secure Cer...	Equifax Secure Certifi...	8/22/2018	GeoTrust
GeoTrust Global CA	GeoTrust Global CA	5/21/2022	GeoTrust Global CA
GTE CyberTrust Glo...	GTE CyberTrust Globa...	8/13/2018	DigiCert Global R...
internal-ca	internal-ca	9/23/2026	<None>
Microsoft Authentic...	Microsoft Authenticod...	12/31/1999	Microsoft Authe...
Microsoft Root Aut...	Microsoft Root Authority	12/31/2020	Microsoft Root A...
Microsoft Root Cert...	Microsoft Root Certifi...	5/9/2021	Microsoft Root C...
NO LIABILITY ACC...	NO LIABILITY ACCEP...	1/7/2004	VeriSign Time St...

Wes Whitteker, wes_whitt@yahoo.com

With the root certificate installed on the proper internal systems, Squid Proxy MITM functionality can be enabled. Navigate to Services > Squid Proxy Server and scroll down to "SSL Man In the Middle Filtering". Once there, enable HTTPS/SSL Interception; select the LAN SSL Intercept Interface; choose the proper CA (SEC503 for the test environment). Click the Save button at the bottom of the page.



SSL Man In the Middle Filtering

HTTPS/SSL Interception: ☑ Enable SSL filtering.

SSL Intercept Interface(s): LAN (selected), WAN
The interface(s) the proxy server will intercept SSL requests on.
Note: Use CTRL + click to select multiple interfaces.

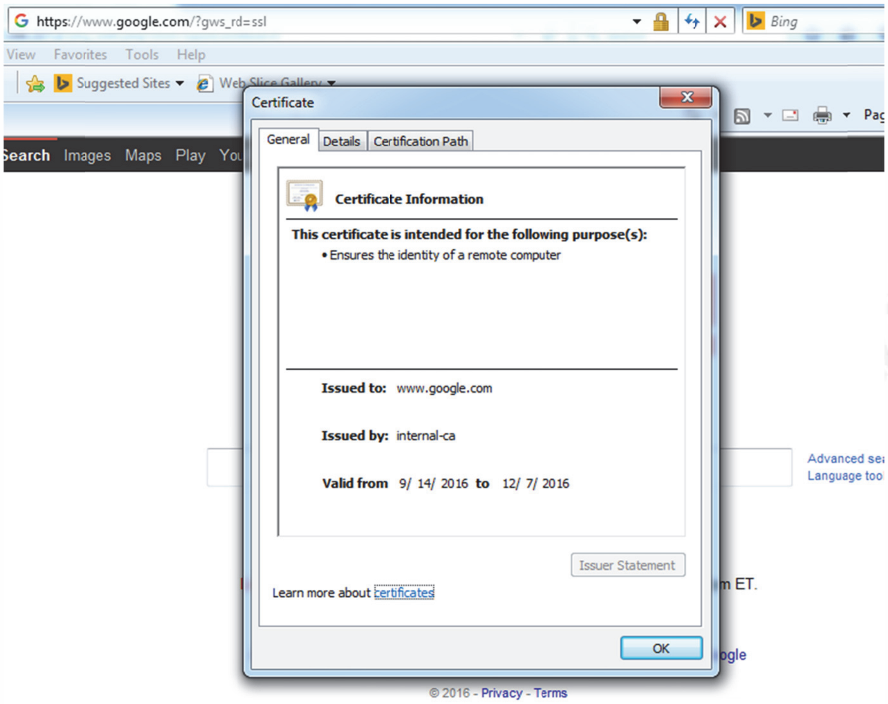
SSL Proxy port:
This is the port the proxy server will listen on to intercept SSL while using transparent proxy.
(Default: 3129)

CA: SANS503
Select Certificate Authority to use when SSL interception is enabled.
To create a CA on pfSense, go to **System -> Cert Manager**.
Install the CA certificate as a Trusted Root CA on each computer you want to filter SSL on to avoid SSL error on each connection.

SSL Certificate Deamon Children:
This is the number of SSL certificate deamon children to start. If Squid is used in busy environments, this may need to be increased.
Default: 5

Using Internet Explorer, navigate to an encrypted webpage and look at the certificate being used. As you can see below, the certificate is actually for our internal pfSense CA.

Wes Whitteker, wes_whitt@yahoo.com



Certificate

General | Details | Certification Path

Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer

Issued to: www.google.com

Issued by: internal-ca

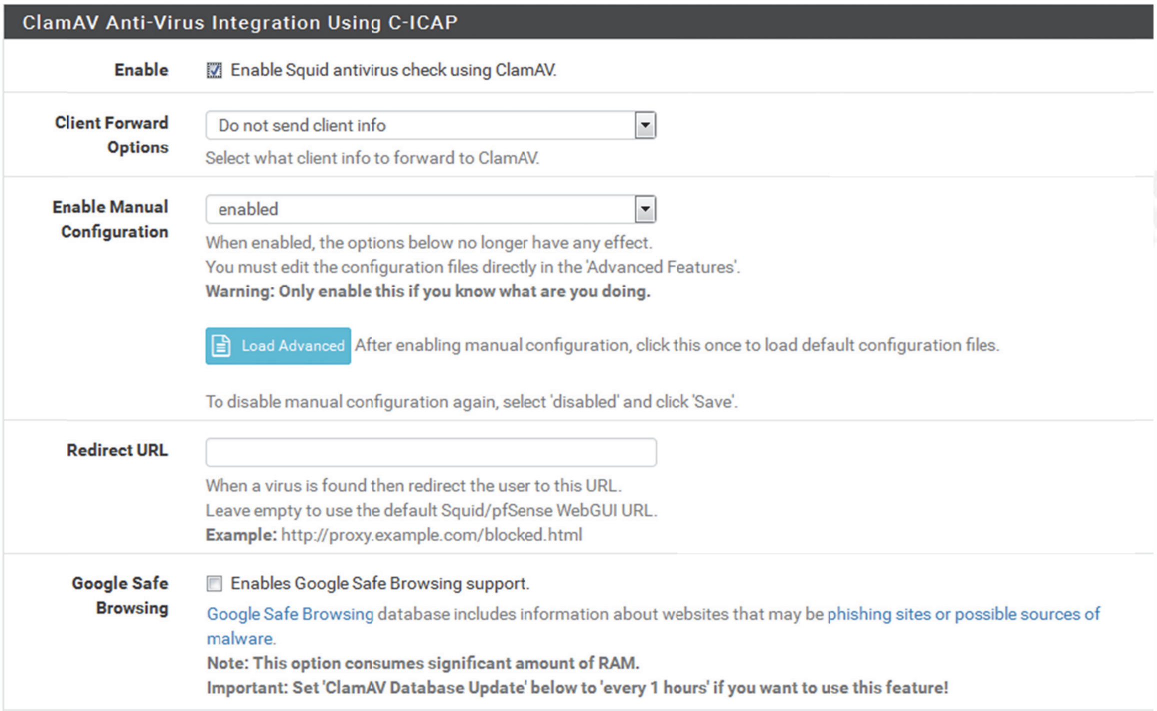
Valid from 9/ 14/ 2016 **to** 12/ 7/ 2016

Issuer Statement

Learn more about certificates

OK

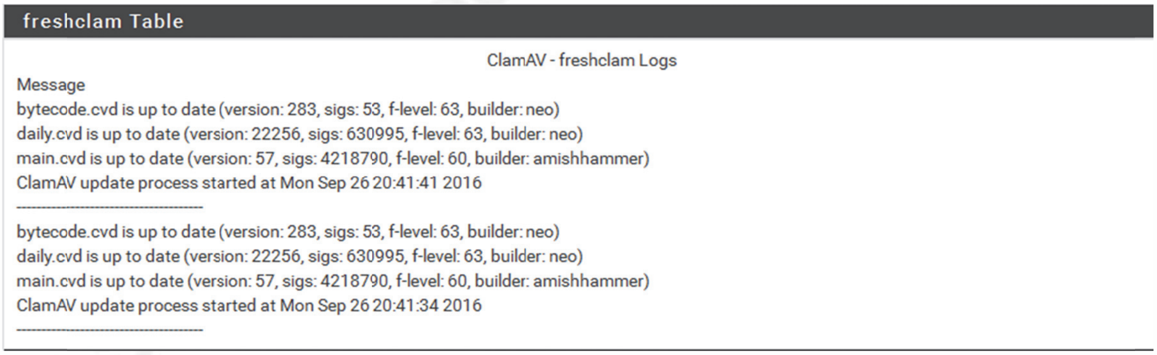
With the TLS decryption enabled and working, the AV scanning can be enabled. Navigate to Services > Squid Proxy Server > Antivirus. Select the Enable checkbox. For the test environment client forward options, the option was selected to not send any data to Clam AV. Make sure to change the ClamAV Database update period to the desired setting – every hour was selected for the test environment. At this point, it is worth noting that the ClamAV default signatures do not have the best reputation for detections. As such, it is recommended to augment the default signatures with the signatures from sansecurity.com (http://sanesecurity.com/usage/signatures). The *freshclam.conf* file will need updated with the locations for the desired additional signatures. This can be completed by adding the URL locations to the following *freshclam.conf* directive: *DatabaseCustomURL*. In order to update these signatures, you will have to load the advanced configuration.



ClamAV Anti-Virus Integration Using C-ICAP

Enable	☑ Enable Squid antivirus check using ClamAV.
Client Forward Options	Do not send client info Select what client info to forward to ClamAV.
Enable Manual Configuration	enabled When enabled, the options below no longer have any effect. You must edit the configuration files directly in the 'Advanced Features'. **Warning: Only enable this if you know what are you doing.** Load Advanced After enabling manual configuration, click this once to load default configuration files. To disable manual configuration again, select 'disabled' and click 'Save'.
Redirect URL	 When a virus is found then redirect the user to this URL. Leave empty to use the default Squid/pfSense WebGUI URL. **Example:** http://proxy.example.com/blocked.html
Google Safe Browsing	□ Enables Google Safe Browsing support. Google Safe Browsing database includes information about websites that may be phishing sites or possible sources of malware. **Note: This option consumes significant amount of RAM.** **Important: Set 'ClamAV Database Update' below to 'every 1 hours' if you want to use this feature!**

The AV definition update status can be reviewed by navigating to Services > Squid Proxy Server > Real Time, and scroll down to the "freshclam Table" section.

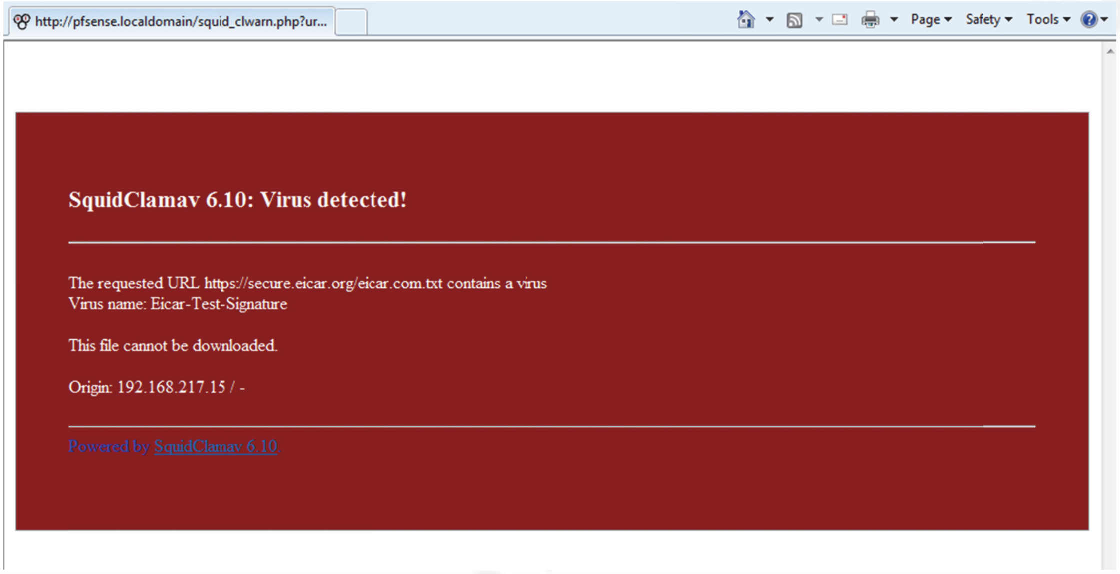


freshclam Table

```
ClamAV - freshclam Logs
Message
bytecode.cvd is up to date (version: 283, sigs: 53, f-level: 63, builder: neo)
daily.cvd is up to date (version: 22256, sigs: 630995, f-level: 63, builder: neo)
main.cvd is up to date (version: 57, sigs: 4218790, f-level: 60, builder: amishhammer)
ClamAV update process started at Mon Sep 26 20:41:41 2016
--------------------------------------
bytecode.cvd is up to date (version: 283, sigs: 53, f-level: 63, builder: neo)
daily.cvd is up to date (version: 22256, sigs: 630995, f-level: 63, builder: neo)
main.cvd is up to date (version: 57, sigs: 4218790, f-level: 60, builder: amishhammer)
ClamAV update process started at Mon Sep 26 20:41:34 2016
--------------------------------------
```

With the AV functionality setup, it is now time to test that it is working as planned. We will use the standard EICAR test signature. These are offered via both HTTP and HTTPS connections (www.eicar.org/85-0-Download.html). Below you can see the results when the HTTPS MITM functionality is enabled and the results when the HTTPS MITM functionality is disabled, respectively. You can see that the EICAR file is not blocked when MITM functionality is disabled. It is worth noting that the EICAR HTTPS site certificate is self-signed so you may get prompted with browser warnings.

Additionally, there is a setting in Squid Proxy under Services > Squid Proxy Server > SSL Man In the Middle Filtering > Remote Cert Checks where you will need to select "Accept remote server certificate with errors". If this option is not checked within Squid Proxy, you will receive a self-signed certificate error in Squid Proxy.



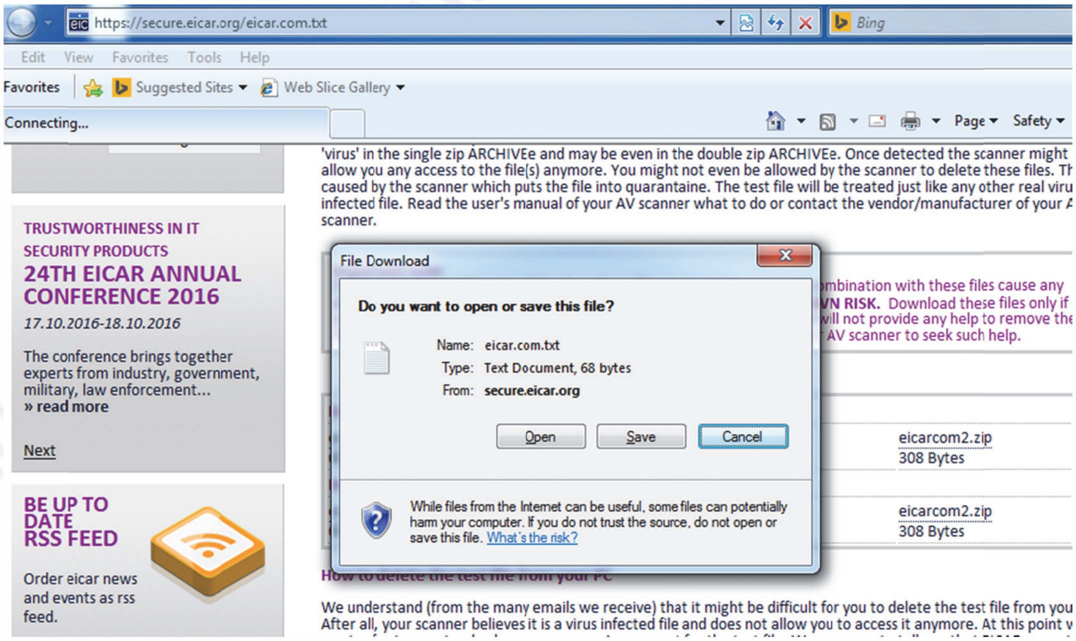
SquidClamav 6.10: Virus detected!

The requested URL https://secure.eicar.org/eicar.com.txt contains a virus
Virus name: Eicar-Test-Signature

This file cannot be downloaded.

Origin: 192.168.217.15 / -

Powered by SquidClamav 6.10



'virus' in the single zip ARCHIVEe and may be even in the double zip ARCHIVEe. Once detected the scanner might allow you any access to the file(s) anymore. You might not even be allowed by the scanner to delete these files. Tł caused by the scanner which puts the file into quarantaine. The test file will be treated just like any other real viru infected file. Read the user's manual of your AV scanner what to do or contact the vendor/manufacturer of your A scanner.

TRUSTWORTHINESS IN IT SECURITY PRODUCTS

24TH EICAR ANNUAL CONFERENCE 2016

17.10.2016-18.10.2016

The conference brings together experts from industry, government, military, law enforcement...
» read more

BE UP TO DATE RSS FEED

Order eicar news and events as rss feed.

File Download

Do you want to open or save this file?

Name: eicar.com.txt
Type: Text Document, 68 bytes
From: secure.eicar.org

While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

mbination with these files cause any N RISK. Download these files only if vill not provide any help to remove the AV scanner to seek such help.

eicarcom2.zip
308 Bytes

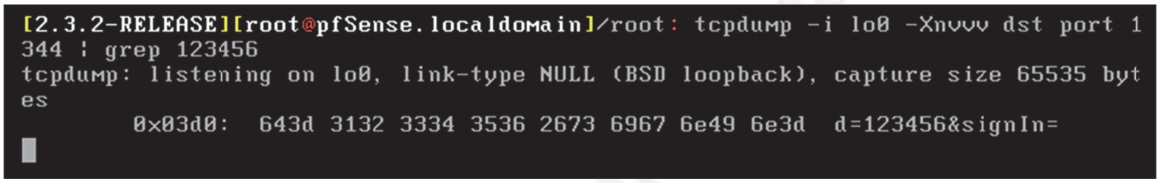
eicarcom2.zip
308 Bytes

We understand (from the many emails we receive) that it might be difficult for you to delete the test file from you After all, your scanner believes it is a virus infected file and does not allow you to access it anymore. At this point v

Another way to assess the decryption functionality between the Squid Proxy Service and the C-ICAP extension is to leverage *tcpdump* on the pfSense system. Below *tcpdump* is being used to find the string "123456," which is found in the password field of a TLS encrypted login page:

Command used: tcpdump -i lo0 -Xnvvv dst port 1344 | grep <insert text>

This command dumps the traffic on port 1344 of the local interface (used by the C-ICAP extension). It prints the output in HEX & ASCII, does not do name resolution and produces verbose output while grepping for a specified string: "http://www.tcpdump.org/tcpdump_man.html."



```
[2.3.2-RELEASE][root@pfSense.localdomain]/root: tcpdump -i lo0 -Xnvvv dst port 1
344 | grep 123456
tcpdump: listening on lo0, link-type NULL (BSD loopback), capture size 65535 byt
es
        0x03d0:  643d 3132 3334 3536 2673 6967 6e49 6e3d  d=123456&signIn=
```

At this point, the solution is both decrypting TLS traffic and inspecting it for potential viruses. There are obviously several other features and configuration options available in both pfSense and Squid Proxy. However, this is one possible solution that could be leveraged to address encrypted communication concerns.

5. Conclusion

As the paper has shown, the existing trend of TLS usage is increasing. This trend is being reinforced because of significant public concern for privacy. The inevitable result will be a very large growth in TLS communication. This "Age of Encryption" will have an obvious impact on cybersecurity. To this end, the ability for security analysts to do their jobs is challenged because they lack the visibility to comprehensively detect malicious activity on their networks. The bottom line is the path to more encryption has opened a major avenue for malicious activity.

To combat this, organizations can turn to decrypting both inbound and outbound Internet communications. There are certainly risks and rewards when organizations are deciding on a decryption solution. The overarching risk is "seeing" sensitive data. The sensitive data could be corporate data, personal info, or any other data that is of a sensitive nature. The overarching benefit is that decryption provides insight into the embedded encrypted data of the communication. This content could be a file, URL, commands, or any other information that could be associated with malicious activity.

Due to the sensitive nature of an organization's decision to decrypt Internet communication, it is import that businesses involve legal departments in the decryption strategy decision making process. Further, organizations need to comprehensively understand the data on their networks. In many cases the decryption decision making process will result in a balancing act between policy and security. This balancing act will require that organizations develop an appropriate strategy on traffic inspection within the confines of what is legally authorized (Boss, 2016, Butler, 2013, Casey, 2013, & Dormann, 2015).

Once organizations have a comprehensive decryption policy in place and a firm understanding of their network traffic, they can move forward with deciding on a solution. There are several open source decryption alternatives available, which allows some flexibility when determining what is needed for a particular situation. If an open source solution is not a workable option, there are also several commercial options available. Throughout the decision-making process, it is important that the technical details of the TLS communication process are considered to ensure the final solution is successful.

References

A10 Networks. (2015). TOP 6 DANGERS OF NOT INSPECTING YOUR SSL TRAFFIC. Retrieved from https://www.a10networks.com/sites/default/files/A10-GR-70198-EN.pdf

Bloch, K. (2015, January 22). Top 10 ICT Trends in 2015: are you ready for the digital transformation?. Retrieved from http://gblogs.cisco.com/asiapacific/top-10-ict-trends-in-2015-are-you-ready-for-digital-transformation/

Bluecoat. (2016). Certificate Pinning. Retrieved from https://www.bluecoat.com/ko/documents/download/7ff09c94-7b88-4319-a766-191c9dedde22&usg=afqjcnhmc8cwvi7rxfyu5gho8hfw1sic9a&bvm=bv.137132246,d.ewe&cad=rja

Boss, S. (2016, February 19). When your Employer Can See Your Banking Information: Decrypting SSL | AlienVault. Retrieved from https://www.alienvault.com/blogs/security-essentials/when-your-employer-can-see-your-banking-information-decrypting-ssl

Butler, M. (2013, November). Finding Hidden Threats by Decrypting SSL. Retrieved from https://www.sans.org/reading-room/whitepapers/analyst/finding-hidden-threats-decrypting-ssl-34840

Casey, B. (2013, April). The pros and cons of SSL decryption for enterprise network monitoring. Retrieved from http://searchsecurity.techtarget.com/answer/The-pros-and-cons-of-SSL-decryption-for-enterprise-network-monitoring?src=itke%20stub

Castro, D., & McQuinn, A. (2016, March). Unlocking Encryption: Information Security and the Rule of Law. Retrieved from http://www2.itif.org/2016-unlocking-encryption.pdf

Chokhani, S., McKay, K., Polk, T. (2014). Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations. Retrieved from http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf

C-ICAP Project. (n.d.). About. The C-ICAP Project. Retrieved from http://c-icap.sourceforge.net/about.html

CIO.gov. (n.d.). Introduction to HTTPS. Retrieved from https://https.cio.gov/faq/

CIO.gov. (n.d.a). The HTTPS-Only Standard. Retrieved from https://https.cio.gov/

Clinton, W. J. (1996, November 15). Executive Order 13026. Retrieved from https://www.gpo.gov/fdsys/pkg/WCPD-1996-11-18/pdf/WCPD-1996-11-18-Pg2399.pdf

Davis, J. S. (2016, August 2). C&C using SSL to hide malware increased by 200 times – report. Retrieved from http://www.scmagazine.com/cc-using-ssl-to-hide-malware-increased-by-200-times--report/article/513622/

Digitaltrends.com. (2016, April 3). Apple vs. the FBI: A Complete Timeline of Events | Digital Trends. Retrieved from http://www.digitaltrends.com/mobile/apple-encryption-court-order-news/

Djekic, M. (2013, November 25). A Scytale – Cryptography of the Ancient Sparta. Retrieved from http://www.australianscience.com.au/technology/a-scytale-cryptography-of-the-ancient-sparta/

Dormann, W. (2015, March 13). The Risks of SSL Inspection. Retrieved from https://insights.sei.cmu.edu/cert/2015/03/the-risks-of-ssl-inspection.html

Ekersley, P. (2014, November 18). Launching in 2015: A Certificate Authority to

Encrypt the Entire Web | Electronic Frontier Foundation. Retrieved from https://www.eff.org/deeplinks/2014/11/certificate-authority-encrypt-entire-web

Evsslcertificate.com. (n.d.). History of SSL Certificate | When was SSL Certificate Introduced. Retrieved from https://www.evsslcertificate.com/ssl/ssl-history.html

Exec. Order No. 13026, 3 C.F.R. 2399 (1996), *reprinted as amended in* 6 U.S.C. §121 at 17 (2002).

Fraunhofer FKIE. (2016, January). White Paper | Encrypted Traffic Management. Retrieved from http://dc.bluecoat.com/encrypted-traffic-management-fraunhofer-institute

Google.com. (n.d.). Encryption is under attack. – Take Action – Google. Retrieved from https://www.google.com/takeaction/issue/encryption/

Hagemann, R., & Hampson, J. (2015, November 9). Encryption, Trust, and the Online Economy | An Assessment of the Economic Benefits Associated With Encryption. Retrieved from https://niskanencenter.org/wp-content/uploads/2015/11/RESEARCH-PAPER_EncryptionEconomicBenefits.pdf

HISTORY.com. (n.d.). The Invention of the Internet – Inventions. Retrieved from http://www.history.com/topics/inventions/invention-of-the-internet

Hwang, J. (2012, June 6). The Secure Sockets Layer and Transport Layer Security. Retrieved from http://www.ibm.com/developerworks/library/ws-ssl-security/

IETF HTTP Working Group (WG). (n.d.). HTTP Documentation. Retrieved from http://httpwg.org/specs/

IETF HTTP Working Group (WG). (n.d.a). HTTP/2 Frequently Asked Questions. Retrieved from https://http2.github.io/faq/

IETF HTTP Working Group (WG). (n.d.b). Hypertext Transfer Protocol Version 2 (HTTP/2). Retrieved from http://http2.github.io/http2-spec/#TLSUsage

IETF TLS Working Group (WG). (n.d.) Transport Layer Security (TLS). Retrieved https://datatracker.ietf.org/wg/tls/charter/

IETF. (1996). Hypertext Transfer Protocol -- HTTP/1.0. Retrieved from https://www.ietf.org/rfc/rfc1945.txt

IETF. (1999). Hypertext Transfer Protocol -- HTTP/1.1. Retrieved from https://www.ietf.org/rfc/rfc2616.txt

IETF. (1999a). The TLS Protocol Version 1.0. Retrieved from https://www.ietf.org/rfc/rfc2246.txt

IETF. (1999b). HTTP Authentication: Basic and Digest Access Authenticaton. Retrieved from https://www.ietf.org/rfc/rfc2617.txt

IETF. (2000). HTTP Over TLS. Retrieved from https://tools.ietf.org/html/rfc2818

IETF. (2006). The Transport Layer Security (TLS) Protocol Version 1.1. Retrieved from https://www.ietf.org/rfc/rfc4346.txt

IETF. (2008). The Transport Layer Security (TLS) Protocol Version 1.2. Retrieved from https://tools.ietf.org/html/rfc5246

IETF. (2011). The Secure Sockets Layer (SSL) Protocol Version 3.0. Retrieved from https://www.ietf.org/rfc/rfc6101.txt

IETF. (2014). Pervasive Monitoring is an Attack. Retrieved from https://tools.ietf.org/html/rfc7258

IETF. (2015). Hypertext Transfer Protocol Version 2 (HTTP/2). Retrieved from https://tools.ietf.org/html/rfc7540

IETF. (2015a). Deprecating Secure Sockets Layer Version 3.0. Retrieved from https://www.ietf.org/rfc/rfc7568.txt

IETF. (2015b). Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Retrieved from https://tools. ietf.org/html/rfc7525

IETF. (2016). The Transport Layer Security (TLS) Protocol Version 1.3 draft-ietf-tls-tls13-15. Retrieved from https://tools.ietf.org/html/draft-ietf-tls-tls13-15

Internet Hall of Fame. (n.d.). Timeline. Retrieved from http://www.internethalloffame .org/internet-history/timeline

Karsten, J., & West, D. M. (2016, April 16). A brief history of U.S. encryption policy | Brookings Institution. Retrieved from https://www.brookings.edu/blog/techtank /2016/04/19/a-brief-history-of-u-s-encryption-policy/

Kear, S. (2016, May 4). How to Set Up an HTTP Anti-Virus Proxy Using pfSense and HVAP. Retrieved from https://turbofuture.com/internet/How-to-Set-Up-an-HTTP-Anti-Virus-Proxy-Using-pfSense-and-HAVP

Kear, S. (2016, May 23). Intercepting HTTPS Traffic Using the Squid Proxy Service in pfSense. Retrieved from https://turbofuture.com/internet/Intercepting-HTTPS-Traffic-Using-the-Squid-Proxy-in-pfSense

Krebs, B. (2016). KrebsonSecurity | In-depth security news and investigation. Retrieved from http://krebsonsecurity.com/

Letsencrypt.org. (n.d.). About - Let's Encrypt - Free SSL/TLS Certificates. Retrieved from https://letsencrypt.org/about/

Lewis, J. A. (2014, January 10). 2014 as the Year of Encryption: A (Very) Brief History of Encryption Policy | Center for Strategic and International Studies. Retrieved from https://www.csis.org/analysis/2014-year-encryption-very-brief-history-encryption-policy

Mayfield, G. (2015, April 6). The Risks and Rewards of SSL Encryption | Blue Coat. Retrieved from https://www.bluecoat.com/company-blog/2015-04-06/risks-and-rewards-ssl-encryption

Marlinspike, M. (n.d.). Software >> sslstrip. Retrieved from https://moxie.org/software /sslstrip/

Netcraft.com. (n.d.). Netcraft | SSL Survey. Retrieved from https://www.netcraft.com/ internet-data-mining/ssl-survey/

OWASP.org. (2016, August 10). HTTP Strict Transport Security Cheat Sheet - OWASP. Retrieved from https://www.owasp.org/index.php/HTTP_Strict_Transport _Security_Cheat_Sheet

Pew Research Center. (n.d.). Web History Timeline. Retrieved from http://www.pew internet.org/2014/03/11/world-wide-web-timeline/

pfSense. (n.d.). pfSense Overview. Retrieved from https://www.pfsense.org/about-pfsense/

Ristic, I. (2015). *Bulletproof SSL and TLS*. London: Feisty Duck.

Rivest, R. L. (2011, February 8). The Growth of Cryptography. Retrieved from https://people.csail.mit.edu/rivest/pubs/Riv11a.slides.pdf

Sandvine.com. (2015, May 8). Global Internet Phenomena Spotlight | Encrypted Internet Traffic. Retrieved from https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/encrypted-internet-traffic.pdf

Segura, J. (2015, August 13). SSL Malvertising Campaign Continues (UPDATED) | Malwarebytes Labs. Retrieved from https://blog.malwarebytes.com/threat-analysis/2015/08/ssl-malvertising-campaign-continues/

Squid-cache.org (n.d.) Introduction. Retrieved from http://www.squid-cache.org/Intro/

Superconfigure.wordpress.com. (2013, January 29). Pen-testing HSTS (Http Strict Transport Security) Sites with Burp. Retrieved from https://superconfigure .wordpress.com/2013/01/29/pen-testing-hsts-http-strict-transport-security-sites-with-burp/

Tan, J. (2010, May 19). Decrypting SSL 'protected' web traffic for further analysis [PoC]. Retrieved https://jez4christ.com/view/decrypting-ssl-protected-web-traffic-for-further-analysis-poc/

Vincent, B. (2016, July 29). Google is turning on HSTS encryption on its domain. Retrieved from https://www.engadget.com/2016/07/29/google-hsts-encryption/

Wireshark.org. (n.d.). SSL - The Wireshark Wiki. Retrieved September 4, 2016, from https://wiki.wireshark.org/SSL

WolfSSL.com. (2010, December 14). A Comparison of TLS 1.1 and TLS 1.2. Retrieved from https://www.wolfssl.com/wolfSSL/Blog/Entries/2010/12/14_A_ Comparison_of_TLS_1.1_and_TLS_1.2.html

World Wide Web Consortium (W3C) (n.d.). Index of History. Retrieved from https://www.w3.org/History/

World Wide Web Consortium (W3C) (2004). Change History for HTTP. Retrieved from https://www.w3.org/Protocols/History.html

World Wide Web Foundation. (n.d.). History of the Web. Retrieved from http://web foundation.org/about/vision/history-of-the-web/

Zakon, R. H. (n.d.). Hobbes' Internet Timeline - the definitive ARPAnet & Internet history. Retrieved from http://www.zakon.org/robert/internet/timeline/