

Transport Layer Protection Cheat Sheet

From OWASP

Contents

1 Introduction

This article provides a simple model to follow when implementing transport layer protection for an application. Although the concept of SSL is known to many, the actual details and security specific decisions of implementation are often poorly understood and frequently result in insecure deployments. This article establishes clear rules which provide guidance on securely designing and configuring transport layer security for an application. This article is focused on the use of SSL/TLS between a web application and a web browser, but that we also encourage the use of SSL/TLS or other network encryption technologies, such as VPNs, on back end and other non-browser based connections.

Architectural Decision

An architectural decision must be made to determine the appropriate method to protect data when it is being transmitted. The most common options available to corporations are Virtual Private Networks (VPN) or a SSL/TLS model commonly used by web applications. The selected model is determined by the business needs of the particular organization. For example, a VPN connection may be the best design for a partnership between two companies that includes mutual access to a shared server over a variety of protocols. Conversely, an Internet facing enterprise web application would likely be best served by a SSL/TLS model.

This cheat sheet will focus on security considerations when the SSL/TLS model is selected. This is a frequently used model for publicly accessible web applications.

Providing Transport Layer Protection with SSL/TLS

Benefits

The primary benefit of transport layer protection is the protection of sensitive data from unauthorized disclosure and network sniffing. This is transmitted between clients (web browsers) and the web application server and between the web application server and back end and other non-browser based servers and primary editors.

The server validation component of TLS provides authentication of the server to the client. If configured to require client side certificates, TLS can also play a role in client authentication to the server. However, in practice client side certificates are not often used in lieu of username and password based authentication models for clients.

TLS also provides two additional benefits that are commonly overlooked; integrity guarantees and replay prevention. A TLS stream of communication contains built-in controls to prevent tampering with any portion of the encrypted data. In addition, controls are also built-in to prevent a captured stream of TLS data from being replayed at a later time.

It should be noted that TLS provides the above guarantees to data during transmission. TLS does not offer any of these security benefits to data that is at rest. Therefore appropriate security controls must be added to protect data while at rest within the application or within data stores.

Basic Requirements

The basic requirements for using TLS are: access to a Public Key Infrastructure (PKI) in order to obtain certificates, access to a directory or an Online Certificate Status Protocol (OCSP) responder in order to check certificate revocation status, and agreement/ability to support a minimum configuration of protocol versions and protocol options for each version.

SSL vs. TLS

The terms, Secure Socket Layer (SSL) and Transport Layer Security (TLS) are often used interchangeably. In fact, SSL v3.1 is equivalent to TLS v1.0. However, different versions of SSL and TLS are supported by modern web browsers and by most modern web frameworks and platforms. For the purposes of this cheat sheet we will refer to the technology generically as TLS. Recommendations regarding the use of SSL and TLS protocols, as well as browser support for TLS, can be found in the rule below title "Only Support Strong Protocols".

When to Use a FIPS 140-2 Validated Cryptomodule

If the web application may be the target of determined attackers (a common threat model for Internet accessible applications handling sensitive data), it is strongly advised to use TLS services that are provided by FIPS 140-2 validated cryptomodules (<http://csrc.nist.gov/groups/STM/cmvp/validation.html>) .

A cryptomodule, whether it is a software library or a hardware device, basically consists of three parts:

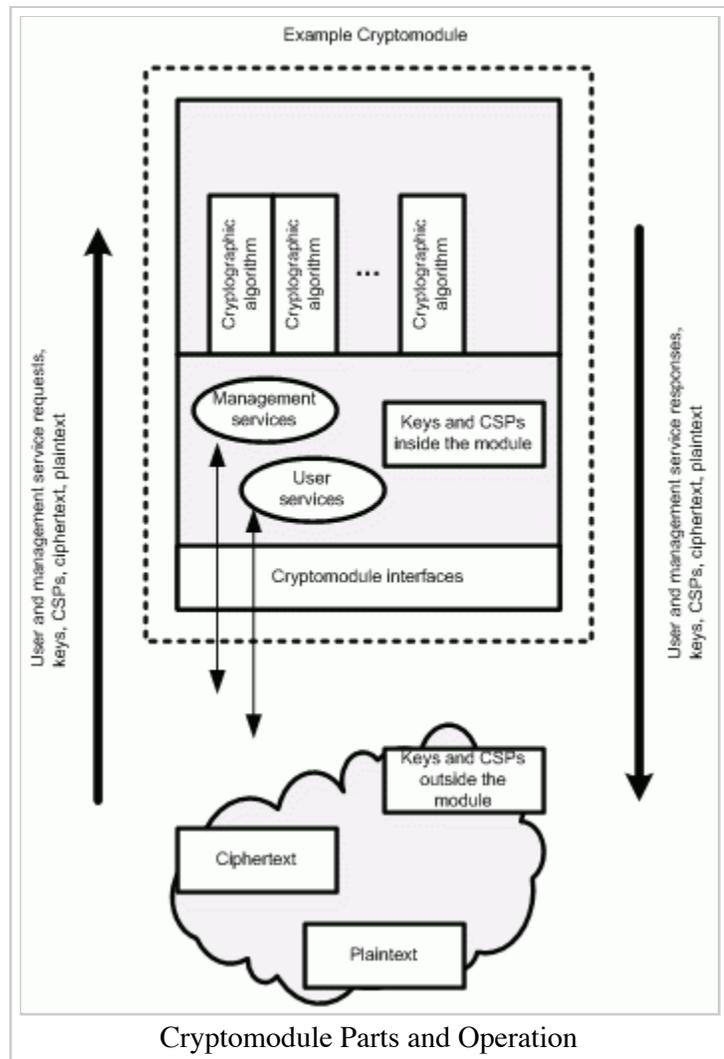
- Components that implement cryptographic algorithms (symmetric and asymmetric algorithms, hash algorithms, random number generator algorithms, and message authentication code algorithms)
- Components that call and manage cryptographic functions (inputs and outputs include cryptographic keys and so-called critical security parameters)
- A physical container around the components that implement cryptographic algorithms and the

components that call and manage cryptographic functions

The security of a cryptomodule and its services (and the web applications that call the cryptomodule) depend on the correct implementation and integration of each of these three parts. In addition, the cryptomodule must be used and accessed securely. This includes consideration for:

- Calling and managing cryptographic functions
- Securely Handling inputs and output
- Ensuring the secure construction of the physical container around the components

In order to leverage the benefits of TLS it is important to use a TLS service (e.g. library, web framework, web application server) which has been FIPS 140-2 validated. In addition, the cryptomodule must be installed, configured and operated in either an approved or an allowed mode to provide a high degree of certainty that the FIPS 140-2 validated cryptomodule is providing the expected security services in the expected manner.



If the system is legally required to use FIPS 140-2 encryption (e.g., owned or operated by or on behalf of the U.S. Government) then TLS must be used and SSL disabled. Details on why SSL is unacceptable are described in Section 7.1 of Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program (<http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf>).

Further reading on the use of TLS to protect highly sensitive data against determined attackers can be viewed in SP800-52 Guidelines for the Selection and Use of Transport Layer Security (TLS) Implementations (<http://csrc.nist.gov/publications/nistpubs/800-52/SP800-52.pdf>)

Secure Server Design

Rule - Use TLS for All Login Pages and All Authenticated Pages

The login page and all subsequent authenticated pages must be exclusively accessed over TLS. The initial login page, referred to as the "login landing page", must be served over TLS. Failure to utilize

TLS for the login landing page allows an attacker to modify the login form action, causing the user's credentials to be posted to an arbitrary location. Failure to utilize TLS for authenticated pages after the login enables an attacker to view the unencrypted session ID and compromise the user's authenticated session.

Rule - Use TLS on Any Networks (External and Internal) Transmitting Sensitive Data

All networks, both external and internal, which transmit sensitive data must utilize TLS or an equivalent transport layer security mechanism. It is not sufficient to claim that access to the internal network is "restricted to employees". Numerous recent data compromises have shown that the internal network can be breached by attackers. In these attacks, sniffers have been installed to access unencrypted sensitive data sent on the internal network.

Rule - Do Not Provide Non-TLS Pages for Secure Content

All pages which are available over TLS must not be available over a non-TLS connection. A user may inadvertently bookmark or manually type a URL to a HTTP page (e.g. <http://site.com/myaccount>) within the authenticated portion of the application. If this request is processed by the application then the response, and any sensitive data, would be returned to the user over the clear text HTTP.

Rule - REMOVED - Do Not Perform Redirects from Non-TLS Page to TLS Login Page

This recommendation has been removed. Ultimately, the below guidance will only provide user education and cannot provide any technical controls to protect the user against a man-in-the-middle attack.

--

A common practice is to redirect users that have requested a non-TLS version of the login page to the TLS version (e.g. <http://site.com/login> redirects to <https://site.com/login>). This practice creates an additional attack vector for a man in the middle attack. In addition, redirecting from non-TLS versions to the TLS version reinforces to the user that the practice of requesting the non-TLS page is acceptable and secure.

In this scenario, the man-in-the-middle attack is used by the attacker to intercept the non-TLS to TLS redirect message. The attacker then injects the HTML of the actual login page and changes the form to post over unencrypted HTTP. This allows the attacker to view the user's credentials as they are transmitted in the clear.

It is recommended to display a security warning message to the user whenever the non-TLS login page is requested. This security warning should urge the user to always type "HTTPS" into the browser or bookmark the secure login page. This approach will help educate users on the correct and most secure method of accessing the application.

Currently there are no controls that an application can enforce to entirely mitigate this risk. Ultimately, this issue is the responsibility of the user since the application cannot prevent the user from initially typing <http://site.com/login> (<http://owasp.org>) (versus HTTPS).

Note: Strict Transport Security (http://www.w3.org/Security/wiki/Strict_Transport_Security) will address this issue and will provide a server side control to instruct supporting browsers that the site should only be accessed over HTTPS

Rule - Do Not Mix TLS and Non-TLS Content

A page that is available over TLS must be comprised completely of content which is transmitted over TLS. The page must not contain any content that is transmitted over unencrypted HTTP. This includes content from unrelated third party sites.

An attacker could intercept any of the data transmitted over the unencrypted HTTP and inject malicious content into the user's page. This malicious content would be included in the page even if the overall page is served over TLS. In addition, an attacker could steal the user's session cookie that is transmitted with any non-TLS requests. This is possible if the cookie's 'secure' flag is not set. See the rule 'Use "Secure" Cookie Flag'

Rule - Use "Secure" Cookie Flag

The "Secure" flag must be set for all user cookies. Failure to use the "secure" flag enables an attacker to access the session cookie by tricking the user's browser into submitting a request to an unencrypted page on the site. This attack is possible even if the server is not configured to offer HTTP content since the attacker is monitoring the requests and does not care if the server responds with a 404 or doesn't respond at all.

Rule - Keep Sensitive Data Out of the URL

Sensitive data must not be transmitted via URL arguments. A more appropriate place is to store sensitive data in a server side repository or within the user's session. When using TLS the URL arguments and values are encrypted during transit. However, there are two methods that the URL arguments and values could be exposed.

1. The entire URL is cached within the local user's browser history. This may expose sensitive data to any other user of the workstation.
2. The entire URL is exposed if the user clicks on a link to another HTTPS site. This may expose sensitive data within the referral field to the third party site. This exposure occurs in most browsers and will only occur on transitions between two TLS sites.

For example, a user following a link on <https://site.com> (<http://owasp.org>) which leads to <https://someOtherSite.com> (<http://owasp.org>) would expose the full URL of <https://site.com> (<http://owasp.org>) (including URL arguments) in the referral header (within most browsers). This would not be the case if the user followed a link on <https://site.com> (<http://owasp.org>) to

<http://someHTTPsite.com> (<http://owasp.org>)

Rule - Prevent Caching of Sensitive Data

The TLS protocol provides confidentiality only for data in transit but it does not help with potential data leakage issues at the client or intermediary proxies. As a result, it is frequently prudent to instruct these nodes not to cache or persist sensitive data. One option is to add a suitable Cache-Control header to relevant HTTP responses, for example "Cache-Control: no-cache, no store". For compatibility with HTTP/1.0 the response should include header "Pragma: no-cache". More information is available in HTTP 1.1 RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>) , section 14.9.

Server Certificate & Protocol Configuration

Note: If using a FIPS 140-2 cryptomodule disregard the following rules and defer to the recommended configuration for the particular cryptomodule.

Rule - Use an Appropriate Certificate Authority for the Application's User Base

An application user must never be presented with a warning that the certificate was signed by an unknown or untrusted authority. The application's user population must have access to the public certificate of the certificate authority which issued the server's certificate. For Internet accessible websites, the most effective method of achieving this goal is to purchase the TLS certificate from a recognize certificate authority. Popular Internet browsers already contain the public certificates of these recognized certificate authorities.

Internal applications with a limited user population can use an internal certificate authority provided its public certificate is securely distributed to all users. However, remember that all certificates issued by this certificate authority will be trusted by the users. Therefore, utilize controls to protect the private key and ensure that only authorized individuals have the ability to sign certificates.

The use of self signed certificates is never acceptable. Self signed certificates negate the benefit of end-point authentication and also significantly decrease the ability for an individual to detect a man-in-the-middle attack.

Rule - Only Support Strong Cryptographic Ciphers

The strength of the encryption used within a TLS session is determined by the encryption cipher negotiated between the server and the browser. In order to ensure that only strong cryptographic ciphers are selected the server must be modified to disable the use of weak ciphers. It is recommended to configure the server to only support strong ciphers and to use sufficiently large key sizes. In general, the following should be observed when selecting CipherSuites:

- Use AES, 3DES for encryption
- Use CBC mode
- Use SHA1 for digest

- MD5 may be used within the TLS protocol
- Do not provide support for NULL ciphersuites
- Do not provide support for anonymous Diffie-Hellman
- Support ephemeral Diffie-Hellman key exchange

Note: The TLS usage of MD5 does not expose the TLS protocol to any of the weaknesses of the MD5 algorithm (see FIPS 140-2 IG). However, MD5 must never be used outside of TLS protocol (e.g. for general hashing).

Note: Use of Ephemeral Diffie-Hellman key exchange will protect confidentiality of the transmitted plaintext data even if the corresponding RSA or DSS server private key got compromised. An attacker would have to perform active man-in-the-middle attack at the time of the key exchange to be able to extract the transmitted plaintext. All modern browsers support this key exchange with the notable exception of Internet Explorer prior to Windows Vista.

Additional information can be obtained within the TLS 1.1 RFC 4346 (<http://www.ietf.org/rfc/rfc4346.txt>) and FIPS 140-2 IG (<http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf>)

Rule - Only Support Strong Protocols

Weaknesses have been identified with older SSL protocols. The best practice for transport layer protection is to only provide support for the TLS protocols - TLS1.0, TLS 1.1 and TLS 1.2. This configuration will provide maximum protection against skilled and determined attackers and is appropriate for applications handling sensitive data or performing critical operations.

The following browsers support at least TLS 1.0. The earliest version supporting TLS is listed below.

- Internet Explorer in IE7 (also supported in IE6, but disabled by default)
- Firefox 2.0
- Chrome 1.0
- Apple Safari (version unknown)
- Opera 5

In situations where lesser security requirements are necessary, it may be acceptable to also provide support for SSL 3.0. It should be noted that this configuration is a derivation from the best practice and should only be used if the ensuing security risks are evaluated and can be accepted for the particular business operation.

In no situation should SSL 2.0 be enabled on the server. This protocol has multiple known weaknesses and does not provide effective transport layer protection.

Rule - Only Support Secure Renegotiations

A design weakness in TLS, identified as CVE-2009-3555 (<http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-3555>), allows an attacker to inject a plaintext of his choice into a TLS

session of a victim. In the HTTPS context the attacker might be able to inject his own HTTP requests on behalf of the victim. The issue can be mitigated either by disabling support for TLS renegotiations or by supporting only renegotiations compliant with RFC 5746 (<http://www.ietf.org/rfc/rfc5746.txt>) . All modern browsers have been updated to comply with this RFC.

Rule - Use Strong Keys & Protect Them

The private key used to generate the cipher key must be sufficiently strong for the anticipated lifetime of the private key and corresponding certificate. The current best practice is to select a key size of at least 2048. Keys of length 1024 will be obsolete beginning in 2010. Additional information on key lifetimes and comparable key strengths can be found in NIST SP 800-57 (http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf) . In addition, the private key must be stored in a location that is protected from unauthorized access.

Rule - Use a Certificate That Supports All Available Domain Names

A user should never be presented with a certificate error, including domain mismatch certificate errors. If the application is available at both <https://www.site.com> (<https://owasp.org>) and <https://site.com> (<https://owasp.org>) then an appropriate certificate, or certificates, must be selected to accommodate both situations. The presence of domain mismatch errors desensitizes users to TLS error messages and increases the possibility an attacker could launch a convincing man-in-the-middle attack.

For example, consider a web application accessible at <https://abc.site.com> (<https://owasp.org>) and <https://xyz.site.com> (<https://owasp.org>) . A wild card certificate with the domain name of *.site.com would be an appropriate choice.

Alternatively, subject alternate names can be used to provide a specific listing of multiple names where the certificate is valid. These certificates are sometimes referred to as "multiple domain certificates" or enhanced wildcard certificates.

Client (Browser) Configuration

The validation procedures to ensure that a certificate is valid are complex and difficult to correctly perform. In a typical web application model, these checks will be performed by the client's web browser in accordance with local browser settings and are out of the control of the application. However, these items do need to be addressed in the following scenarios:

- The application server establishes connections to other applications over TLS for purposes such as web services or any exchange of data
- A thick client application is connecting to a server via TLS

In these situations extensive certificate validation checks must occur in order to establish the validity of the certificate. Consult the following resources to assist in the design and testing of this functionality. The NIST PKI testing site includes a full test suite of certificates and expected outcomes of the test cases.

- NIST PKI Testing (http://csrc.nist.gov/groups/ST/crypto_apps_infra/pki/pkitesting.html)
- IETF RFC 3280 (<http://www.ietf.org/rfc/rfc3280.txt>)

As specified in the above guidance, if the certificate can not be validated for any reason then the connection between the client and server must be dropped. Any data exchanged over a connection where the certificate has not properly been validated could be exposed to unauthorized access or modification.

Additional Controls

Extended Validation Certificates

Extended validation certificates (EV Certificates) require more intensive investigation into the requesting party. The purpose of EV certificates is to provide the user with greater assurance that the owner of the certificate is a verified legal entity for the site. Browsers with support for EV certificates distinguish an EV certificate in a variety of ways. Internet Explorer will color a portion of the URL in green, while Mozilla will add a green portion to the left of the URL indicating the company name.

High value websites should consider the use of EV certificates to enhance customer confidence in the certificate. It should also be noted that EV certificates do not provide any greater technical security for the TLS. The purpose of the EV certificate is to increase user confidence that the target site is indeed who it claims to be.

Client-Side Certificates

Client side certificates can be used with TLS to prove the identity of the client to the server. Referred to as "two-way TLS", this configuration requires the client to provide their certificate to the server, in addition to the server providing their's to the client. If client certificates are used, ensure that the same validation of the client certificate is performed by the server, as indicated for the validation of server certificates above. In addition, the server should be configured to drop the TLS connection if the client certificate cannot be verified or is not provided.

The use of client side certificates is relatively rare currently due to the complexities of certificate generation, safe distribution, client side configuration, certificate revocation and reissuance, and the fact that clients can only authenticate on machines where their client side certificate is installed. Such certificates are typically used for very high value connections that have small user populations.

Providing Transport Layer Protection for Back End and Other Connections

Although not the focus of this cheat sheet, it should be stressed that transport layer protection is necessary for back-end connections and any other connection where sensitive data is exchanged or where user identity is established. Failure to implement an effective and robust transport layer security

will expose sensitive data and undermine the effectiveness of any authentication or access control mechanism.

Secure Internal Network Fallacy

The internal network of a corporation is not immune to attacks. Many recent high profile intrusions, where thousands of sensitive customer records were compromised, have been perpetrated by attackers that have gained internal network access and then used sniffers to capture unencrypted data as it traversed the internal network.

Related Articles

OWASP - Testing for SSL-TLS, and OWASP Guide to Cryptography

OWASP – Application Security Verification Standard (ASVS) – Communication Security Verification Requirements (V10) (<http://www.owasp.org/index.php/ASVS>)

OWASP – ASVS Article on Why you need to use a FIPS 140-2 validated cryptomodule

SSL Labs - SSL Server Rating Guide (<http://www.ssllabs.com/projects/rating-guide/index.html>)

NIST - SP 800-52 Guidelines for the selection and use of transport layer security (TLS) Implementations (<http://csrc.nist.gov/publications/nistpubs/800-52/SP800-52.pdf>)

NIST - FIPS 140-2 Security Requirements for Cryptographic Modules (<http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf>)

NIST - Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program (<http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf>)

NIST - SP 800-57 Recommendation for Key Management, Revision 2 (http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf)

NIST - SP 800-95 Guide to Secure Web Services (<http://csrc.nist.gov/publications/drafts.html#sp800-95>)

IETF - RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (<http://www.ietf.org/rfc/rfc3280.txt>)

IETF - RFC 4346 The Transport Layer Security (TLS) Protocol Version 1.1 (<http://www.ietf.org/rfc/rfc4346.txt>)

OWASP Cheat Sheet Series

- Authentication Cheat Sheet
- Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet

- **Transport Layer Protection Cheat Sheet**
- Cryptographic Storage Cheat Sheet
- Input Validation Cheat Sheet
- XSS (Cross Site Scripting) Prevention Cheat Sheet
- DOM based XSS Prevention Cheat Sheet
- Forgot Password Cheat Sheet
- SQL Injection Prevention Cheat Sheet
- Session Management Cheat Sheet
- HTML5 Security Cheat Sheet
- Web Service Security Cheat Sheet
- Application Security Architecture Cheat Sheet

Draft OWASP Cheat Sheets

- PHP Security Cheat Sheet
- Password Storage Cheat Sheet

Cheat Sheets Project Homepage

- Cheat Sheets

Authors and Primary Editors

Michael Coates - michael.coates[at]owasp.org

Dave Wichers - dave.wichers[at]aspectsecurity.com

Michael Boberski - boberski_michael[at]bah.com

Tyler Reguly -treguly[at]sslfail.com

Retrieved from "https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet"
Category: Cheatsheets

- Powered by MediaWiki OWASP Foundation © 2011

