White Paper

Best Practices and Applications of TLS/SSL

The most well-known example of the use of public key infrastructure has proven flexible enough to assist in authentication, encryption and data integrity in numerous applications throughout the enterprise.

By Larry Seltzer
Security Analyst and Writer

Best Practices and Applications of TLS/SSL

The most well-known example of the use of public key infrastructure has proven flexible enough to assist in authentication, encryption and data integrity in numerous applications throughout the enterprise.

CONTENTS

Executive Summary

Transport Layer Security or TLS, widely known also as Secure Sockets Layer or SSL, is the most popular application of public key cryptography in the world. It is most famous for securing Web browser sessions, but it has widespread application to other tasks.

TLS/SSL can be used to provide strong authentication of both parties in a communication session, strong encryption of data in transit between them, and verification of the integrity of that data in transit.

TLS/SSL can be used to secure a broad range of critical business functions such as Web browsing, server-to-server communications, email client-to-server communications, software updating, database access, virtual private networking and others.

However, when used improperly, TLS can give the illusion of security where the communications have been compromised. It is important to keep certificates up to date and check rigorously for error conditions.

In many, but not all applications of TLS, the integrity of the process is enhanced by using a certificate issued by an outside trusted Certificate Authority (CA).

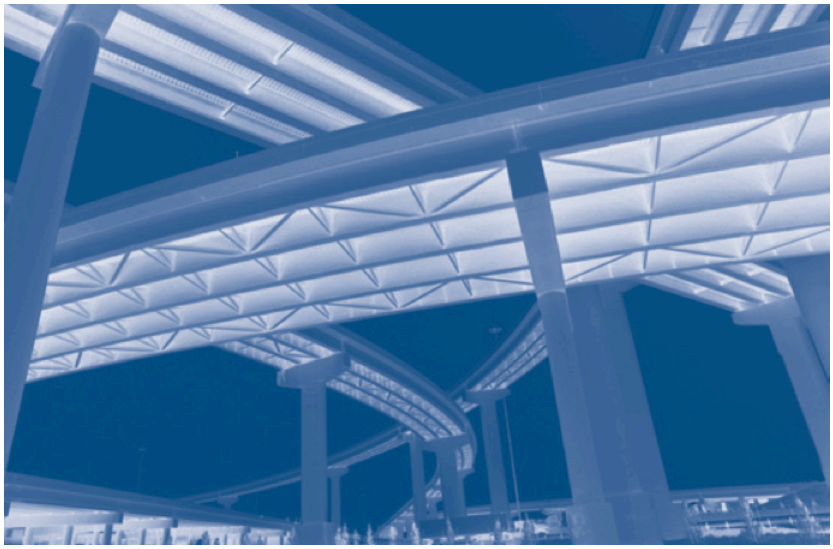
This paper will explore how TLS works, best practices for its use, and the various applications in which it can secure business computing.

Introduction

As the science of business computing, and of computing security in particular, has advanced, the trend has been to find security weaknesses everywhere. Where complexity and functionality grow, so do the opportunities for abuse of systems by malicious actors.

The solutions to these problems are varied and must be explored individually, but one technology shows up often: TLS or Transport Layer Security, often known by the name of the predecessor technology, SSL or Secure Sockets Layer.

TLS is best known as the technology which secures Web browser sessions for banking and other sensitive tasks, but it can be used for much more. Client-server communication with a variety of server types, in addition to Web servers, benefits from use of TLS. Server-to-server communications also need to be secured and can be through TLS. Clients updating applications and other software on their PCs should only do so through a secure connection, which is why such update applications usually use TLS or SSL. This paper will explore these and other applications of TLS that can secure the enterprise in the myriad places in which it can be attacked.



TLS provides 3 basic benefits:

- It provides authentication of the communicating parties, either one-way or in both directions
- It encrypts the communication session “on the wire”
- It ensures the integrity of the data transferred

This paper will also discuss trusted CAs and their role in the public key infrastructure or PKI. Though trusted CAs aren’t always necessary, in many cases they are beneficial and sometimes necessary for the protection to have any effect.

What is TLS/SSL?

TLS/SSL is a tunneling protocol that works at the transport layer. It provides encryption, authentication and integrity verification of data, and does so by means of digital certificates.

Digital Certificates

A digital certificate is an electronic document which confirms the identity of an entity – which could be a user, a server, a company, a program on a client, just about anything – and associates that entity with a public key. The digital certificate is the entity’s identification to the public key infrastructure. Each party to a TLS-secured communication can evaluate the contents of the certificate. The most examined field is the Common Name. Each then compares it to what they expect. It is also wise to check the issuer of the certificate. Is the issuer a trusted party? For more on these issuers see Trusted Certificate Authorities, page 8.

Users can generate their own digital certificates, called self-signed certificates, with free tools[1]. But such certificates are inherently untrustworthy and the real value of certificates comes when they are issued by a trusted CA. Users can create and run their own CA on their network and sometimes this makes sense, but in many cases it is necessary to use an outside trusted CA which outside parties can also trust. Symantec™ is the largest CA.

[1]The OpenSSL Project - http://openssl.org/. Microsoft’s crypto tools (http://msdn.microsoft.com/en-us/library/aa380259(VS.85).aspx) are included in the Windows SDK (http://msdn.microsoft.com/en-us/windowsserver/bb980924.aspx)

Authentication and Verification

Public key cryptography allows two parties to authenticate each other. Each party has two keys, which are large numeric values. A message exchanged between the parties is run through a hashing algorithm. A hash function takes a block of data and creates a value from it, known as a hash or digest. Make even a small change in the data and the hash changes significantly. At the same time there is no way to recreate the data from the hash.

The sending party to the communications uses their private key to encrypt the hash value. This encrypted value is called a digital signature. The message and signature are sent to the recipient party. The recipient party uses the sender's public key to decrypt the signature. They generate a hash of the message using the same algorithm as the sender and compare the values.

If the values are the same then two things are certain: the data has not been tampered with and the sender is who they purport to be. This is because the private key corresponding to the public key in the certificate was used to sign the data, and the private key should only be accessible by the sender named in the certificate.

Neither authentication nor integrity verification are mandatory in TLS. You can use it simply so that the bits on the wire are encrypted. But authentication is a core feature, important to most customers.

Key Security

There are some absolute rules that need to be followed in order for the public key infrastructure to work properly.

Private keys must be private: The signer of a message needs to keep their private key absolutely confidential. Anyone who has it can effectively impersonate the sender.

Public keys must be public: Well, not necessarily public, but they have to be accessible to anyone who might have a valid reason to read the message or encrypt a message to the entity named in the certificate.

Hash algorithms must not collide: A collision is when the hash algorithm generates the same digest from two different data blocks. At some point this is inevitable, but the ability to generate collisions intentionally compromises all functions of public key cryptography. This is why new and better hash algorithms have been developed over time and put into public use.

Encryption

Even when authentication is not used, TLS can use encryption keys and a cipher algorithm to encrypt/decrypt data for communication. The cipher is also used for encrypting the message digest.

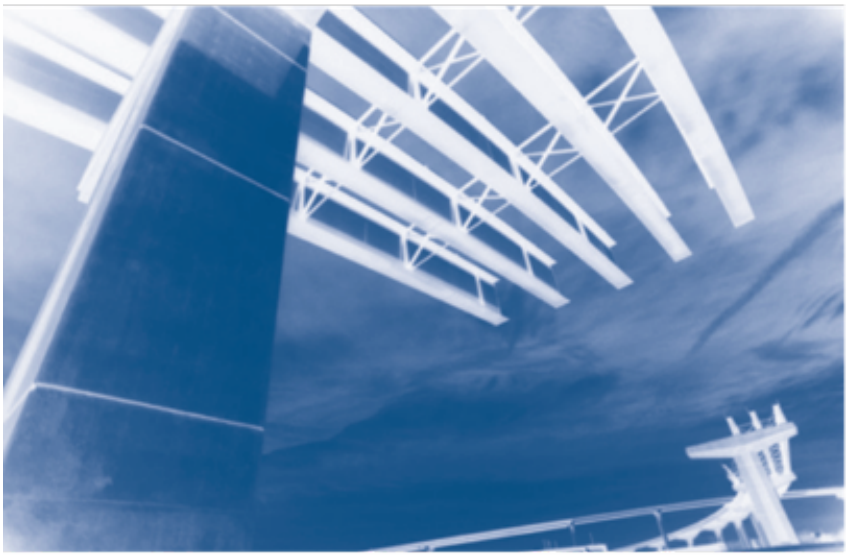
There are many different cipher algorithms for different circumstances. See the TLS vs. SSL section below for more on ciphers.

Where TLS Works In the Stack

TLS/SSL sits between the application and transport layers. In the context of the Internet and most local networking, this means it sits above TCP or Transmission Control Protocol, the protocol which provides reliable, ordered.data communications for applications. Applications that would normally work with TCP connections instead work with TLS connections for the purpose of establishing connections.

Application	TLS Handshake
TLS Record	
TCP (reliable transport)	
IP	
Network Interface	
Wires (infrastructure)	

TLS requires a reliable transport, which means TCP. This means that UDP-only applications like DNS, SNMP, and VOIP, present a problem. See SSL VPNs for more on this problem.



TLS vs. SSL

TLS is the successor technology to SSL, which was developed by Netscape in 1994.[2] The first public release was SSL version 2, and was quickly followed by version 3. The TLS specification was released in 1999 in RFC 2246[3], and is only a minor modification of SSL 3.

[2]Mozilla.org SSL 0.2 PROTOCOL SPECIFICATION - http://www.mozilla.org/projects/security/pki/nss/ssl/draft02.html. This version was widely known as "version 2."
[3]RFC2246 - The TLS Protocol Version 1.0 - http://tools.ietf.org/html/rfc2246

Changes have come at a much slower pace since then, with TLS 1.1[4] and 1.2[5] largely concerned with security improvements. TLS is still widely called SSL, especially in product names, even if the term is strictly inaccurate. Don't be surprised to see the terms used interchangeably. TLS versions are designed to interact with and roll back to earlier protocols such as SSL 3. In fact, in the protocol handshake, TLS 1.0, 1.1 and 1.2 use the version numbers 3.1, 3.2 and 3.3.

One of the main differences you'll see between SSL and TLS versions are the cryptographic features, including the ciphers, hash algorithms and key exchange mechanisms they support. As time and versions advance, support for weaker features is dropped from the protocol and stronger ones added. Administrators on either end of the communication can set policies requiring or prohibiting particular protocols. It's reasonable to claim that the flexibility of TLS with respect to new developments in ciphers and other cryptographic features is one of the main reasons for its success.

Networks Are Insecure By Default

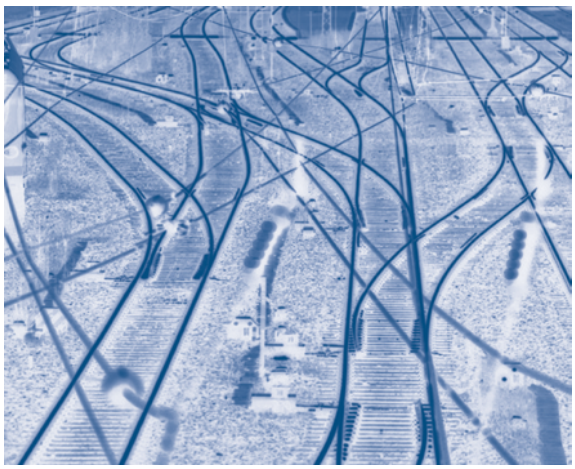
All of our important networking protocols were designed before security issues were properly appreciated. A great deal of research has gone into making communications secure, but in most cases it has to be added on. By default, our networks are insecure.

HTTP, SMTP and FTP were all designed to communicate in clear text. All of them can be enhanced to support encryption and often the solution uses TLS. Only with HTTPS, however, which is HTTP over TLS/SSL, is such security ubiquitous.

Authentication

Authentication in most networks is weak, usually relying on passwords controlled by rules which don't follow best practices. It's no surprise that things are this way; following best practices for passwords is difficult and unpleasant: you have to use passwords that are long and difficult to remember and you need to change them frequently.

Even server-to-server connections are often authenticated with passwords, which are hard-coded into programs or configuration files.



[4]RFC4346 - The Transport Layer Security (TLS) Protocol Version 1.1 - http://tools.ietf.org/html/rfc4346
[5]RFC5246 - The Transport Layer Security (TLS) Protocol Version 1.2 - http://tools.ietf.org/html/rfc5246

Privacy and Integrity

When communications are not secured properly you can't be certain that the data has not been monitored or tampered with in-transit. Attacks known as "man in the middle" (MITM) attacks, where the attacker sits on the network monitoring communications between one party and another, are not uncommon. Such attacks can, for example, steal usernames and other unsecured data on the connection.

When combined with spoofing techniques, the MITM may even modify the data, such as change destination URLs so that you click on a link, which directs you to an attack server.

Solutions

Many products are now designed better than they were in the past to provide secure configuration out of the box. Microsoft in particular has made significant progress in this regard in recent years. But at the level of network communications and authentication, security is still the responsibility of the network administrator.

In many of these cases TLS can help. It can provide authentication where none exists by default. TLS can provide strong encryption where data would normally flow in clear text. And it can ensure that data was not modified in-transit.

Trusted Certificate Authorities

If you are using TLS purely for communication over your own networks, it may be adequate to use an internal CA and set your systems to trust it. However, the hassle and cost of setting up an internal CA often drives businesses to do otherwise. Also, if data is sent over the Internet, where you don't control all points of transit, the only way all parties can trust the certificate is if it was issued by a trusted third-party CA.

Trusted Roots

Because it's unwise to rely completely on Internet communications to prove the issuer of a certificate, many software products come with a list of "trusted root certificates." These certificates are inherently trusted and other certificates signed by those trusted roots are also trusted.

Microsoft Windows includes a list of trusted root certificates which they update periodically through Windows Update and their other updating mechanisms. Many other operating systems have similar lists Application software may use the operating system trusted root list or include their own. Some Web browsers on Windows use the Windows list, but Firefox uses its own list.

One more characteristic of certificate trust is that root CA often have affiliate programs. This allows other companies to sell certificates on behalf of the trusted root CAs. In fact the affiliates can have their own affiliates. The "parent" CA signs the affiliate CA's certificate so software can prove that they are a valid affiliate. The TLS client software "walks" up this hierarchy of CAs, checking the validity of the signatures at each step, until they reach one which is a trusted root. This establishes the trust of the whole hierarchy.

Self-Signed Certificates

Digital certificates need not be signed by a trusted CA. Such certificates, when generated by tools separately, are called self-signed certificates. Such certificates can be used to provide encryption of data, but no authentication. See the section on self-signed certificates for more on this subject.

Authentication Does Not Prove Trust

There are many unfortunate cases of users and organizations being attacked successfully for trusting a communication simply because it was signed. But a signature, even one issued by a CA, is not proof that the party is trustworthy.

Authentication gives you definitive information as to the identity of the other party, which you can use to make an informed trust decision. You can look at the identifying fields in the certificate, such as the Common Name, to see if it matches the entity you expected. You can also look at the Issuer field, which identifies the certificate authority that issued the certificate, and make decisions as to its trustworthiness. As we will see below, different certificates have different amounts of information on them about the identified entity and about the issuer.

Extended Validation (EV) SSL

The workings of digital certificates themselves have been well-established for many years, but the rules for their issuance by trusted CAs has not. The standards of many CAs for issuing certificates reached such a point several years ago that many CAs and vendors of TLS software formed the CA/Browser Forumm (www.cabforum.org) to establish standards. The resulting standard is EV SSL, where EV stands for Extended Validation.

Users generally know EV SSL as the thing that turns their Web browser bar green. More fundamentally it is a new class of TLS certificate issued when the applicant meets certain strict standards for proving its identity established by the CA/Browser Forum. There are also differences in the way browsers behave when interacting with EV SSL certificates, particularly the green browser bar indicator and a clear display of the organization name.

The rules for the entities are rather strict: they must be a legally-recognized entity, either incorporated or a governmental body or some registered non-profit. You cannot get a personal EV SSL certificate. The CA has to verify the legal existence as well as the physical existence and operational existence of the applying entity. The CA must confirm that the entity has the right to use the domain and that the person applying is authorized to do so.[6]

The end result of these standards is that it would be extremely difficult, and likely expensive, to obtain a false EV SSL certificate. As a result of all the work needed to meet the standards, EV SSL certificates are much more expensive than conventional TLS certificates.

[6]CA/Browser Forum EV SSL Certificate Guidelines - http://www.cabforum.org/documents.html

Not Just for Web Browsers

TLS and SSL are best known for securing Web browser communications, but they are by no means limited to HTTP. Many other applications and protocols use TLS for security, generally as an option. Some examples follow:

- **FTPS** – A secure version of the ubiquitous file transfer protocol secured with TLS. Conventional FTP transmissions are in clear text. FTPS has been an official RFC since 2005[7] and an unofficial one for 10 years before that. It has widespread support in FTP software. (Not to be confused with SFTP, which is FTP through an SSH session, or many other less-standardized approaches.)
- **Outlook to Exchange** – Outlook to Exchange over RPC over TLS is becoming popular as a secure external access method, but there are reasons to use it internally as well. In this case you replace the default self-signed certificate in the Exchange server with a real TLS certificate from a trusted CA. This is standard operating procedure now with Exchange hosting services.
- **Windows Update** – Before Windows Vista, Windows Update was simply a website in Internet Explorer which used HTTPS. In more recent versions of Windows, it is a custom app secured by TLS. Many other online software update programs, such as the getPlus program used by Adobe, use TLS connections for security.

Client Security with TLS/SSL

Client side certificates can be used with TLS to prove the identity of the client to the server, and vice-versa. This is called “two-way TLS” and requires the client and server both provide certificates to each other.

There are standards-based systems for administering TLS in this role. Active Directory in Windows manages them[8] and uses certificates stored either in the client or using smart cards or other strong authentication devices.[9] OpenLDAP is an open source directory service that accomplishes much the same.[10]

In spite of the greatly improved security from using strong authentication mechanisms like TLS certificates, it has not been a typical configuration. In exchange for the security, you get a fair amount of complexity and administrative work, plus clients can only log in when they have their client certificate. But recent versions of Windows Server have made it much easier to have certificate and other credential data independent of the user profile, allowing users to move to different computers and still authenticate. Still, this is a higher-cost setup than the use of passwords and most organizations would consider it only for high-value or vulnerable connections.

The default for the LDAP protocol itself is unencrypted by default. Under LDAP version 2 it was common to use a non-standard setup called “LDAPS” which tunneled LDAP through a TLS tunnel on port 636.[11] LDAP version 3 added a standard extension for TLS.[12]

[7]RFC 4217 - http://tools.ietf.org/html/rfc4217
[8]Active Directory Certificate Services - http://technet.microsoft.com/en-us/windowsserver/dd448615.aspx
[9]Smart card and other certificate authentication - http://technet.microsoft.com/en-us/library/cc758410(WS.10).aspx
[10]OpenLDAP, Using TLS - http://www.openldap.org/doc/admin24/tls.html
[11]Microsoft Knowledge Base, How to enable LDAP over SSL with a third-party certification authority - http://support.microsoft.com/kb/321051
[12]RFC 2830, Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security - http://tools.ietf.org/html/rfc2830

Wireless

A secure wireless network requires stronger authentication than the shared secret password used on your home router. Through the WPA and WPA2 security standards, all wireless networks support Extensible Authentication Protocol (EAP), a framework for authentication methods for access to the network. Dozens of such methods have been implemented.[13]

The main point of EAP in a wireless access point or router is to allow it to hook into an enterprise authentication system such as your LAN or WAN. In this scenario, you should demand the highest level of authentication on the access point itself and very high levels from clients connecting to it. TLS is a good choice.

EAP-TLS[14] is one of the methods widely, if not universally, supported in the wireless industry and one of the strongest, if carefully implemented. It uses normal PKI methods to secure connections to a RADIUS server. EAP-TLS is supported out of the box in all major operating systems, including the open source ones.

The main difficulty with EAP-TLS is that clients need to have individual certificates, and they need careful management. In this sense it is the same as client certificates on the LAN: a straight trade-off of security for ease of administration.

Several other EAP methods use TLS in different ways, generally in order to avoid the need for client-side certificates and the administrative burden. They necessarily trade off security for this ease of administration, as credentials can be lost relatively easily and their loss may be difficult to detect.

SSL VPN

Virtual private networking is a must for secure communications over the Internet. There are two popular methods: IPSec and SSL VPN.

IPSec[15] is an end-to-end protocol for authenticating each packet in a TCP/IP network. It operates at the Internet layer of the network, i.e. below the transport layer. This is significant for its application.

SSL VPNs create a tunnel above the transport layer. All application network communications goes through the tunnel. IPSec is a widely-implemented and deployed Internet standard. SSL VPN is not a standard as such, but widely implemented and deployed.

Because IPSec operates at one of the lower levels of the Internet protocol stack, it can transport almost anything you need and with minimal overhead. SSL VPNs operate above the transport layer and require a reliable transport; this means applications that use UDP for transport cannot be supported in a straightforward way through an SSL VPN. Either they have to run outside the tunnel or the UDP packets have to be re-encapsulated in TCP before transport.

[13]Wikipedia, Extensible Application Protocol - http://en.wikipedia.org/wiki/Extensible_Authentication_Protocol
[14]The EAP-TLS Authentication Protocol - http://tools.ietf.org/html/rfc5216
[15]An Illustrated Guide to IPsec - http://www.unixwiz.net/techtips/iguide-ipsec.html

Whether encapsulation of UDP is a problem for the application depends on a number of factors: The quality of the SSL implementation, the speed of the connection and – perhaps most importantly – the tolerance of the application for "lossy" transport.

Applications which use UDP do so for performance reasons, but need to expect some level of packet loss, if only because UDP doesn't guarantee delivery. The most important UDP application is probably VOIP, and some implementations tolerate encapsulation fairly well. Other real-time UDP applications, such as streaming video and some multi-player games don't fare as well. Some VPN/firewall products support both TLS and IPSec for users who need both.[16]

So why not just use IPSec for all VPNs? Because they are comparatively difficult to administer and use. When you're setting up individual users or small groups remotely, an SSL VPN will be much easier to support, especially if the application needs are simple. Site-to-site VPNs, where whole networks are connected over the Internet, need the power and flexibility of IPSec.

Many application-focused remote access methods are TLS-based, and some are SSL VPNs repackaged as remote access for the product.

Server-to-Server Security with TLS

Many server-to-server connections offer TLS as an option and it's an excellent one, especially when the connection is run over the public Internet. A full VPN in such cases is inadvisable, as the application needs are narrow and certificates can be used to provide strong authentication of the servers to each other.

[16]Thanks to Gary Tomlinson and John Gmuender of Sonicwall (www.sonicwall.com) for their help in understanding the problem of UDP on SSL VPNs.

Connections such as these are almost always high-value. Consider Web servers talking to database servers or mail servers synchronizing with each other. The data in the connection is a treasure trove for an attacker. Products such as Microsoft's Exchange Server and Oracle Application Server support TLS with certificates out of the box.

When interacting across the Internet, certificates from a trusted CA are highly recommended. Each party should make as few assumptions about the security of the other as possible.

Beyond that, there are good reasons in such cases for using EV SSL certificates. Research has shown that weaknesses in the CA validation procedures formany conventional SSL certificates leave them vulnerable to man-in-the-middle attacks.[18]

Briefly, the problem comes with the very inexpensive domain-validated certificates. These certificates validate a domain name, not an organization, and the domain name is the only identifying data in the certificate.

Part of the reason they are so inexpensive is that the application and validation procedures are completely automated in many cases. A confirmation request is sent in e-mail to the administrative contact for the domain. Whoever controls that email account can authorize the certificate. Combined with software vulnerabilities in some TLS software, this process can be abused to allow spoofing of a domain certificate and perhaps a man-in-the-middle attack.

The advantage of EV SSL, in this case, is that the EV SSL specification guarantees that an organization name and other data will be in the certificate and will be verified by the CA. For a high-value connection it's worth having this assurance. EV SSL support in such cases may require custom programming.

Web and Intranet Servers

Perhaps it's too obvious to point out, but TLS can be used to secure a browser-Web server connection. It is worth thinking about the extent of your TLS usage for both internal and external sites.

HTTP is a profoundly insecure protocol without TLS. It is plain-text, easily sniffable, and easily manipulated. Even on an internal network you should use TLS/HTTPS for all sufficiently valuable business connections.

Common TLS Mistakes

Don't make the mistake of assuming that a TLS implementation secures your website and that's it. There are many common site design mistakes which defeat the security of TLS.

[18]Spoofing Server-Server Communication: How You Can Prevent It - go.symantec.com/ssl-certificates
*Configuration and alert options vary, depending on the management console you use.

Keep sensitive data out of URLs – When you use HTTP GET methods with parameters that have critical data in them, you expose that data to the whole world. Envision a URL like this one:

There are many reasons why this is a bad idea. But first, the TLS connection will encrypt the URL, so at least users sniffing the network won't see the URL and the parameters. On the other hand, the URL will be stored in the Web server log, in the client browser history, and in referrer headers. This last point means that if there is a link, particularly a non-TLS link, out of the page linked to just above, the full URL will be included in the headers sent to that server.

Use "Secure" Cookie Flag – The "secure" flag will tell the server to send cookies only over a TLS connection. There are numerous attacks to hijack cookies, and cookies often contain the most sensitive of data. Cookies can be stolen through simple packet sniffing, session hijacking, cross-site scripting, or cross-site request forgery. It's still not uncommon to find significant websites that use insecure cookies in HTTPS sessions.

Do Not Mix TLS/SSL and Non-TLS/SSL Content – Just because your page has an HTTPS link doesn't mean all the content will. If there's a frame or an image or some other element that is called with an HTTP link, it will not be TLS-protected. Most browsers won't warn users about the inconsistency.

Do Not Perform Redirects from Non-TLS/SSL Page to TLS/ SSL Login Page – Even today there are major banks which use http on their login pages and redirect to https from there. This is an invitation for phishing pages and other sorts of abuse. Unfortunately, users usually get to https pages from http pages, but the better way would be from a browser bookmark known to be secure or to enter the URL manually.

Hosted Service Security with TLS

Outsourced services through the Web are increasingly popular for good reason. Web-based services can work at least as well and save on staff, infrastructure and capital costs. Any respectable service offered in this way will use a protected connection, probably TLS.

A common reason why companies move to outsourced servers, mostly Microsoft Exchange, is to get easier support for mobile devices like Blackberries. It's important, if this is one of your needs, to buy TLS certificates that are natively supported by ActiveSync or whatever service you will be using for your devices. Hosting services report that getting TLS connections working properly is typically the biggest ramp-up problem they have.

Certificate Expiration

Digital Certificates come with an expiration date, and for good reason. By issuing a certificate, the CA is vouching for the applicant, and it's possible for such information to get "stale". Even with internal certificates there's reason to have and check expiration dates. In fact it's more reasonable for internal certificates to have short expiration dates, since it's easier to distribute them, and the issuing authority in IT is probably the same group who administer the certificates in the user directory.

Some argue that expired certificates are just as trustworthy, or almost as trustworthy, as a "live" one. But an expired certificate is, at the very least, a sign of administrators who don't pay close attention to their servers.

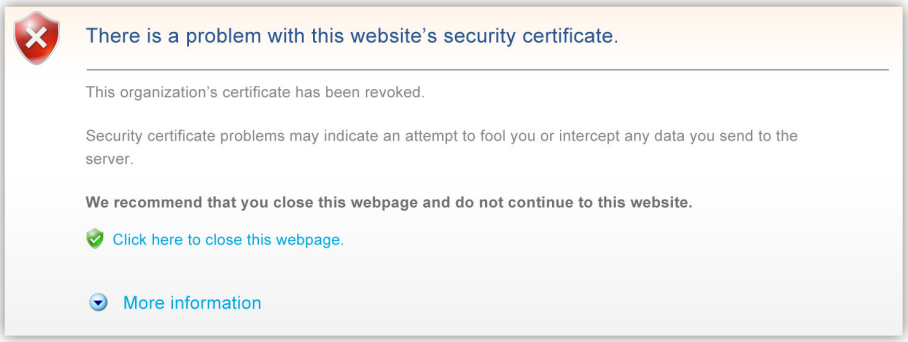
Certificate Revocation

Revocation is an essential feature of the public key infrastructure, one which can't be taken too seriously.

For a number of reasons a CA may revoke the certificate of a customer. It could be for violating the terms of service, for instance by serving malware from the site, or it could be because the customer informed the CA that the private key had been compromised.

When a certificate is revoked, you have to assume that the holder is not trustworthy. It's a lot worse than a site not having a certificate at all.

How do you check? You don't usually check it yourself. Your TLS software (usually a browser) does it by one of two methods: CRL, which stands for certificate revocation lists, is the old method. The certificate itself may contain the address of a revocation list for the CA, and this list is basically a list of certificate serial numbers. If the certificate you are checking is in the list then it's revoked. Don't trust it. Users browsing the site (in Internet Explorer) will see a message like this:



CRLs are simple, but there's a problem with them: over time they can get big, especially for a large CA. They can be cached, but then the CRL check may be out of date. So a protocol named OCSP (online certificate status protocol) was developed to let programs check certificates one at a time. OCSP is the preferred method now.

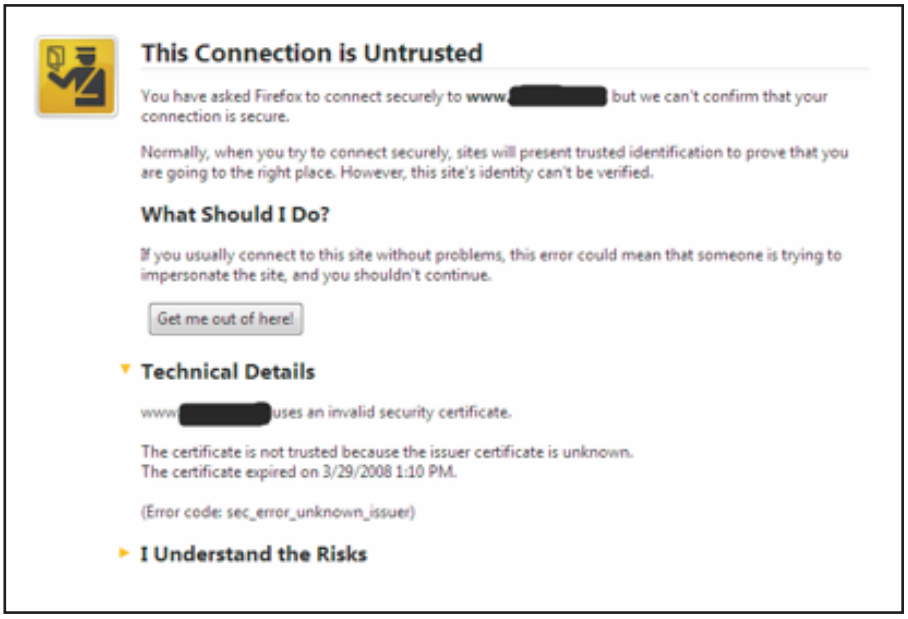
Not all CAs support OCSP. As with CRL, the certificate will contain a field with the address of the OCSP server if the CA supports OCSP.

Self-Signed Certificates

It's not uncommon to find self-signed certificates both on the Internet and internally in corporations. These are certificates which have not been issued by a CA, either internal or external, but generated statically by any of a number of available free tools.

Anyone can make them and their expiration date may be set decades in the future, so they may seem like a good deal. But, as with most things, you get what you pay for with digital certificates. Self-signed certificates don't prove a lot about the site they're protecting.

A self-signed certificate will allow software to encrypt data and ensure its integrity in transit, but it provides no authentication. You can't make any assumptions about the other party.



Note that self-signed certificates are not the same as creating an internal CA. A company can create their own CA, put the name/address of that CA in the browser or system's list of trusted root CAs and then it becomes as trusted as any from a real CA company. Self-signed certificates are not vouched for by anyone.

The vendors of all the major Web browsers know that self-signed certificates are trouble and they issue dire-looking warnings when one is encountered.

Certificate Management

Managing large numbers of certificates in different roles can be a difficult task, and if it's not done in an organized fashion it's an invitation to trouble.

Without a system for certificate management you are likely to find individuals tracking certificates on their own using a spread-sheet or text file. This is how companies get surprised by expiring certificates, and if the employee managing the certificates leaves, the records could end up lost.

Good CAs have online management tools for their certificates, such as Symantec Managed PKI for SSL. There are also third party management packages which track both externally-granted certificates and those from internal CAs.

Conclusions

TLS, widely known as SSL, is the standard for secure application network communications both within the enterprise and across the Internet. TLS can help secure your applications by strengthening authentication, encrypting data communications, and ensuring integrity of data in transit.

TLS is a flexible technology that adapts well to numerous situations from Web server access to server-server communications to virtual private networking. Industry has picked up on this and implemented it widely. It's not always plug and play, but it's ubiquitous enough that expertise is fairly common.

Finally, in many cases the capabilities of TLS are enhanced by certificates from a trusted CA and in some, a trusted CA is necessary for TLS/SSL to make sense.

Additional Reading

Oppliger, Rolf. SSL and TLS: Theory and Practice (Information Security and Privacy (2009). Artech House Publishers, ISBN 1596934476

Rescorla, Eric SSL and TLS: Designing and Building Secure Systems (2000). Addison-Wesley Professional. ISBN 0201615983

Symantec SSL Resources: go.symantec.com/ssl-certificates

Mozilla Developer Center: Introduction to Public-Key Cryptography: https://developer.mozilla.org/en/Introduc- tion_to_Public-Key _Cryptography

The IETF TLS Working Group: http://datatracker.ietf.org/wg/tls/charter/

Microsoft – How to Set Up SSL on IIS7: http://learn.iis.net/page.aspx/144/how-to-set-up-ssl-on-iis-7/

Microsoft – SSL Capacity Planning: http://technet.micro- soft.com/en-us/library/cc302551.aspx

TLS and SSL – What's the Difference?: http://luxsci.com/blog/ssl-versus-tls-whats-the-difference.html

More Information

Visit our website

http://go.symantec.com/ssl-certificates

To speak with a Product Specialist in the U.S.

Call 1 (866) 893-6565 or 1 (650) 426-5112

To speak with a Product Specialist outside the U.S.

For specific country offices and contact numbers, please visit our website.

About Symantec

Symantec is a global leader in providing security, storage, and systems management solutions to help consumers and organizations secure and manage their information-driven world. Our software and services protect against more risks at more points, more completely and efficiently, enabling confidence wherever information is used or stored.

Symantec Corporation World Headquarters

350 Ellis Street
Mountain View, CA 94043 USA
1 (866) 893 6565
www.symantec.com