

ITPro™
SERIES

WindowsITPro eBooks

By Jan De Clercq

Understanding and Leveraging

SSL-TLS for Secure Communications



Contents

| | |
|--|-----------|
| Chapter 3: Advanced SSL/TLS for Secure Web Communications | 42 |
| Certificate Validation Process | 42 |
| Digital Signature and Trust Check | 43 |
| Chain Construction | 45 |
| Chain Validation | 47 |
| Time Check | 49 |
| Revocation Check | 49 |
| CRL Distribution Points | 51 |
| Complete CRLs | 52 |
| Delta CRLs | 53 |
| Online Certificate Status Protocol | 54 |
| Formatting Check | 55 |
| SSL/TLS in Firewall and Proxy Environments | 55 |
| Optimizing SSL Server-Side Performance | 59 |
| SSL and Load Balancing | 60 |
| Conclusion | 61 |

Chapter 3:

Advanced SSL/TLS for Secure Web Communications

This chapter focuses on advanced Secure Sockets Layer/Transport Layer Security (SSL/TLS) topics that architects and technologists might encounter when they are dealing with and configuring the SSL/TLS protocols for secure Web communications.

These topics include the SSL/TLS certificate validation process, SSL/TLS in proxy and firewall environments, SSL/TLS server-side performance optimization, and SSL/TLS and load balancing.

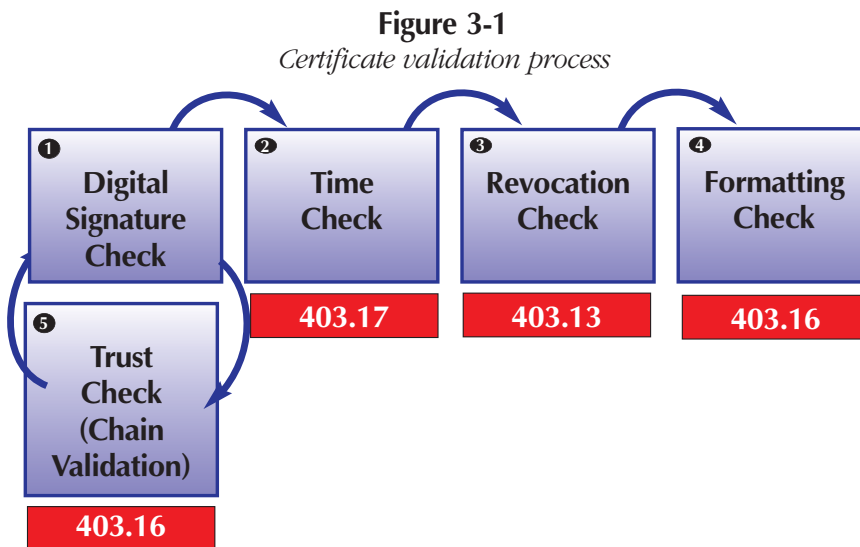
Certificate Validation Process

In a full-blown SSL/TLS protocol exchange (which includes both client and server authentication), both the browser and the Web server must validate each other's certificates. On the server side, this requirement is true only if client-side certificate authentication also has been enabled.

SSL/TLS certificate validation includes the following checks (illustrated in Figure 3-1):

- An X.509 digital signature check that includes a trust check
- A time check
- A revocation check
- A formatting check

Figure 3-1 also shows the HTTP error numbers that the Microsoft IIS Web server returns to the Web browser when one of these checks fails.



The following sections explain SSL/TLS server certificate validation as it works in a Microsoft Internet Explorer (IE) browser and Internet Information Services (IIS) Web server environment. You can apply most validation logic steps to the browser and Web server environments of other software vendors.

Digital Signature and Trust Check

Every X.509 certificate includes a digital signature that is validated when the certificate is used. During the digital signature check, the certificate validation logic validates the digital signature that the issuer of the certificate has applied to the certificate content. The digital signature check includes

- An integrity check to determine whether the certificate content has been tampered with.
- An authentication check to determine whether the Web site owns the private key that is associated with the public key stored in the certificate.

A trustworthy public key is needed to validate a certificate's digital signature. This public key can be the public key of the issuing CA or the public key of another CA that is part of the certificate's certificate chain. But the availability of a public key is not enough to validate a signature: The public key also must be trusted.

In Windows PKI, a trusted CA certificate and public key are known as a *trust anchor*—these components are available from specific containers in a Windows entity's certificate store—specifically, the Trusted Root Certification Authorities, the Enterprise Trust, and the Intermediate Certification Authorities containers.

The process of discovering a trusted CA certificate is part of the trust check. The trust check validates whether the issuer of the Web site's certificate is trusted. The trust check is also referred to as *certificate chain validation*. Certificate chain validation might trigger a different certificate validation loop for each certificate in the certificate chain. We explain certificate chain validation in more detail later in this section.

If the certificate's issuer is not trustworthy, IIS will generate a 403.16 HTTP error. On the browser side, a security alert will be generated stating that “The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority,” as you see in Figure 3-2.

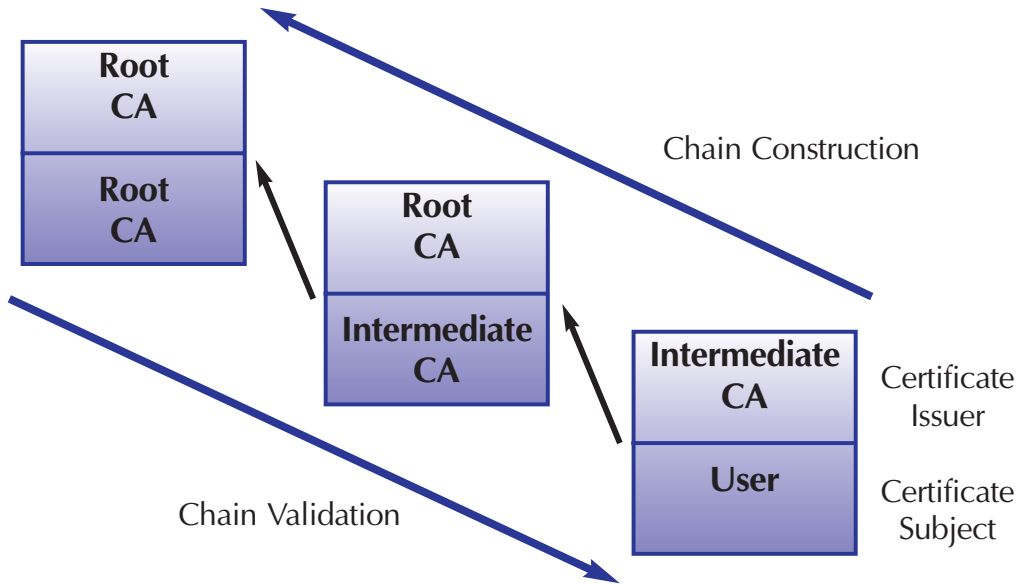
Figure 3-2
Browser-side certificate trust error



So what is a certificate chain, and why do we need to process it during certificate validation? We'll explain this concept using the example of a hierarchical trust model.

In a hierarchical PKI trust model, every end entity's certificate chain consists of all CA certificates that are lying on the path between the user and the root CA of the PKI hierarchy. In a PKI hierarchy, every certificate always contains a pointer to its parent or issuing CA, which is stored in the certificate issuer field. All of this is illustrated in Figure 3-3, which shows the certificate chain of a user certificate that has been issued by a CA that is part of a two-level PKI hierarchy. For simplicity, the certificates in the figure are represented using the certificate subject and the certificate issuer. In this example, the user's certificate subject is the user; its issuer is the intermediate CA. The intermediate CA's certificate subject is the intermediate CA; it has been issued by the root CA. In a hierarchy, the root CA always has a self-signed certificate. In this case, the certificate subject and issuer are identical.

Figure 3-3
Certificate chain processing



During certificate validation, the certificate validation software processes a certificate's certificate chain. This process can be split into two subprocesses: chain construction and chain validation.

Chain Construction

During chain construction, the certificate validation software runs through the certificate's chain until it finds a trusted CA certificate, also known as a *trust anchor*. In the example of Figure 3-4, the validation software finds a trust anchor at the root CA level, and in Figure 3-5, at the intermediate CA level. When a trust anchor is found, the chain construction subprocess stops, and the validation logic switches to chain validation.

Figure 3-4
Certificate chain processing, example 1

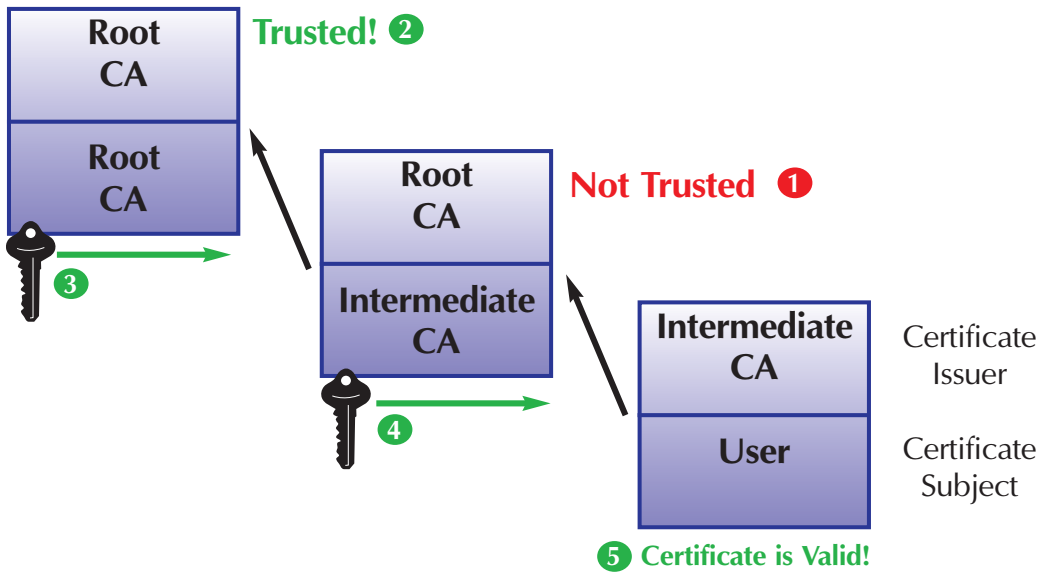
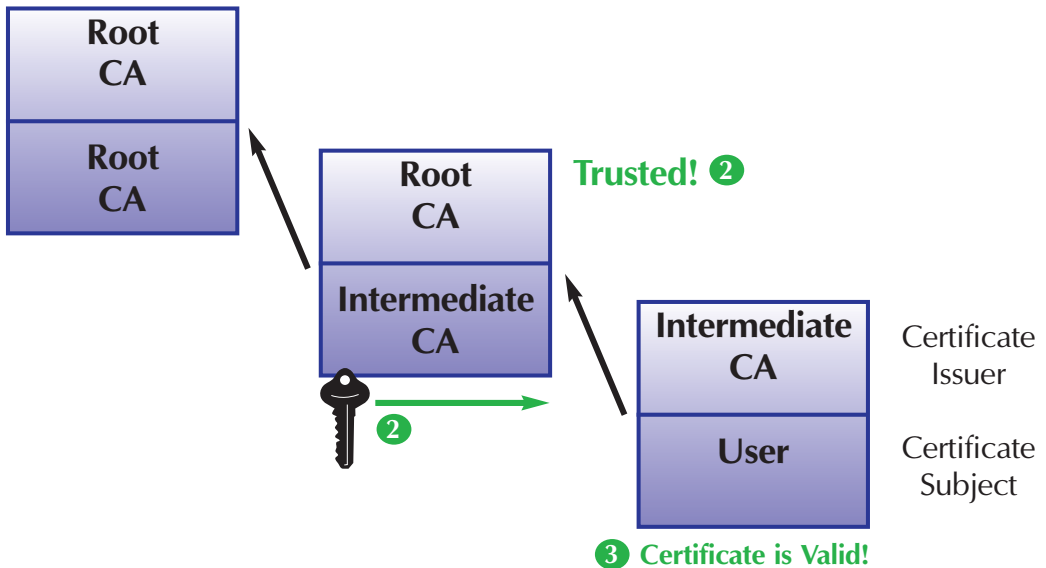


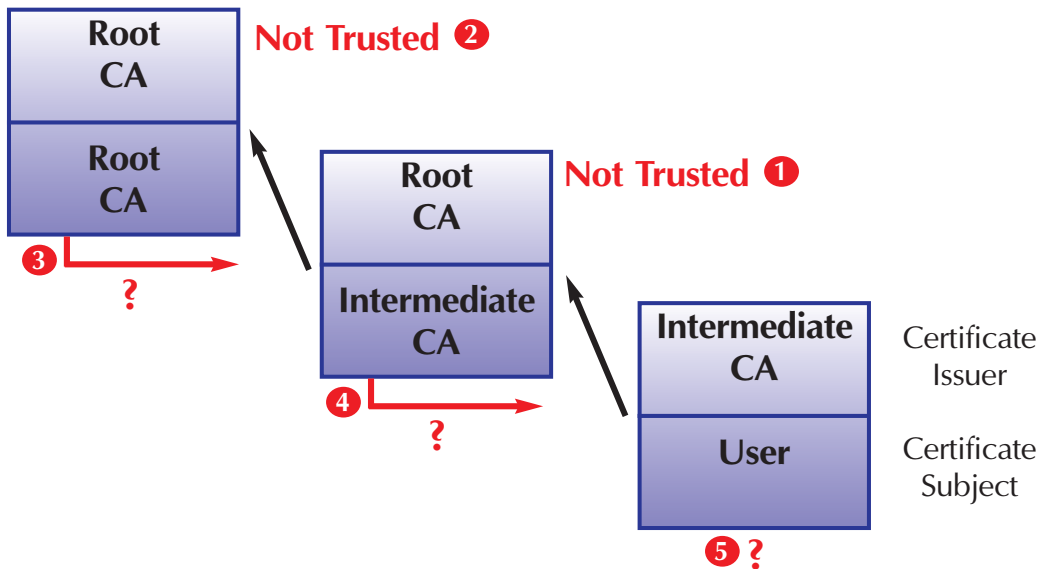
Figure 3-5
Certificate chain processing, example 2



In Figure 3-6, the validation logic cannot find a trust anchor. In this case, the certificate chain process stops, and no decisions can be made regarding the trustworthiness of the certificate.

Figure 3-6

Certificate chain processing, example 3



Chain Validation

During chain validation, the certificate validation software walks the chain in the opposite direction (top down) and validates every CA certificate that is part of the chain. To validate a certificate, the certificate must be available locally in a user's certificate store container. When a certificate is not available locally, the Windows PKI software will use the Authority Information Access (AIA) method that is explained next to obtain a copy of the certificate.

The identification of a CA certificate during chain validation is based on the authority key identifier (AKI) field in the certificate under verification. A certificate's AKI field can contain different types of information:

The AKI field might contain the issuer name and the serial number of the issuer's certificate. In that case, the chain validation logic will try to find a matching certificate using a certificate's Serial number and Subject fields. This way of identifying a certificate is called an *exact match*.

The AKI field might contain the public key identifier (KeyID) of the issuer's certificate. In that case, the chain validation logic will try to find a matching certificate using a certificate's Subject Key Identifier (SKI) extension. This way of identifying a certificate is called a *key match*.

If the certificate under verification does not contain an AKI field, the chain-validation logic will try to identify the issuing CA's certificate by matching the name in the Issuer field of the certificate under verification with the name in the Subject field. This way of identifying a certificate is called a *name match*.

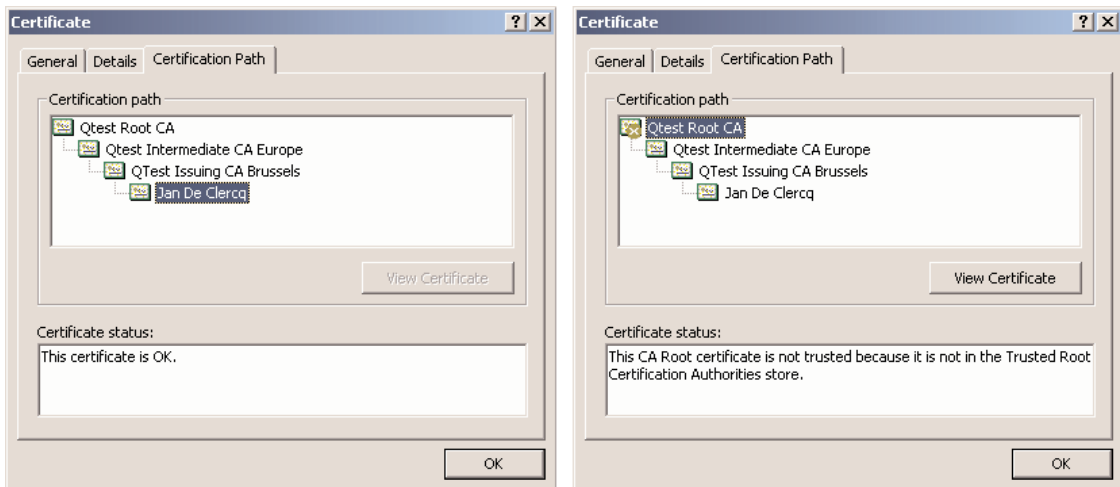
The *AIA method* that the validation logic uses to obtain a local copy of the certificate simply means that the software will try to download the certificate from an online location. To do this, it will use a certificate's Authority Info Access (AIA) field, which contains a Lightweight Directory Access Protocol (LDAP), an HTTP, or a File System pointer to a location where the CA's certificate is stored. If the AIA field has multiple entries, all entries will be tried out in the order that they are listed in the AIA field. All certificates that are downloaded from an AIA location will be cached in the certificate store and on the file system for future use. On the file system, they are cached in the \Documents and Settings\\Local Settings\Temporary Internet Files folder. Note the use of the <username> variable in the previous file system path: The cache location is dependent upon the user security context under which the calling application is running.

If the certificate is not available, certificate verification will fail. If the certificate is available, the certificate validation logic will run (for every certificate in the chain) through all of the steps explained earlier: digital signature, time, revocation, and formatting checks.

You can view a certificate's certificate chain from the Certificate properties dialog box in the "Certification path" tab. Both illustrations in Figure 3-7 show a certificate chain as it is displayed from this dialog box. Figure 3-7 (a) shows the certificate chains of a certificate that ends in a trusted CA certificate; Figure 3-7 (b) shows the certificate chain of a certificate that ends in an untrusted CA certificate.

Figure 3-7

*Certificate chain viewed from the certificate Properties dialog box:
(a) trusted CA certificate and (b) untrusted CA certificate*



As this section illustrates, certificates can be invalid for different reasons: expiration or other time problems, invalid signatures, unavailability of a trusted CA certificate, improper use, improper formatting, revocation, and so forth. That is why finding out the exact reason a certificate is not valid is sometimes a tough job.

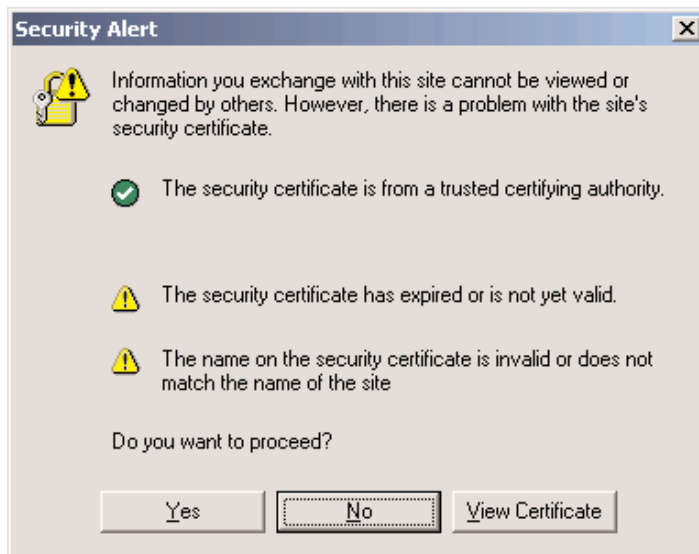
Time Check

The time check determines whether or not the certificate has expired. Every X.509 certificate has a limited lifetime. One reason a certificate's lifetime is limited is to cope with the advances in computer technology. For example, breaking a 512-bit key-rooted asymmetric cipher becomes easier every year. Another reason worth mentioning from a PKI standpoint is that limiting the period of validity forces re-authentication and is therefore a safety measure that, in addition to the revocation check, limits risk exposure due to changes in trusted status/certificate details over time.

If the certificate has expired, IIS will generate a 403.17 HTTP error. On the browser side, a security alert is generated that states "The security certificate has expired or is not yet valid" (see Figure 3-8).

Figure 3-8:

Browser-side SSL/TLS time- and formatting-check errors



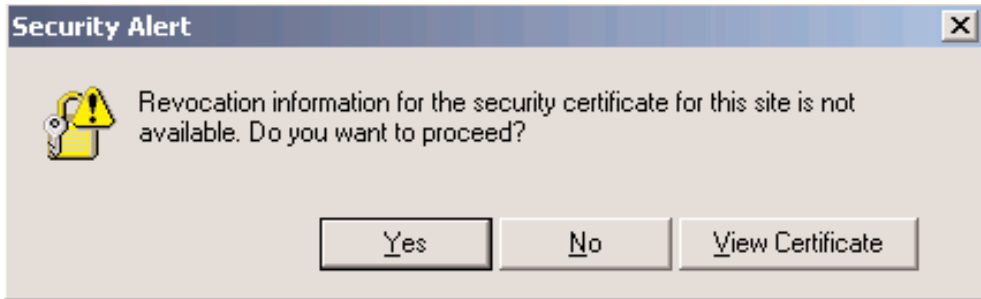
Revocation Check

The revocation check validates whether the certificate has been revoked. This certificate validation step is very important because it assures browser users that they are talking to a Web server whose SSL private key hasn't been compromised. It also ensures that the company or individual in question has not violated its trusted status, which would also result in certificate revocation.

Windows XP, Windows 2000, and Windows Server 2003 PKI support the following revocation-checking mechanisms: complete certificate revocation lists (CRLs) and CRL distribution points (CDPs). Windows Server 2003 PKI and Windows XP also support delta CRLs. CRLs, delta CRLs, and CDPs can provide automated certificate-revocation checking.

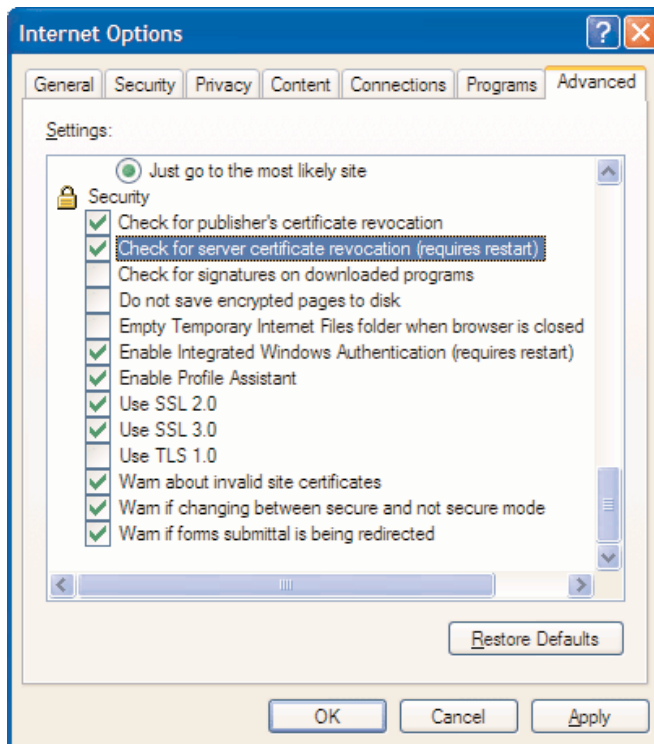
By default, IE does not do a revocation check. On the IE browser side, a security alert such as the one you see in Figure 3-9 will be generated if the CRL is not available.

Figure 3-9
Browser-side SSL/TLS revocation-check error



In IE, you can control revocation checking with the “Check for server certificate revocation” entry in the browser’s Internet Options dialog box (illustrated in Figure 3-10).

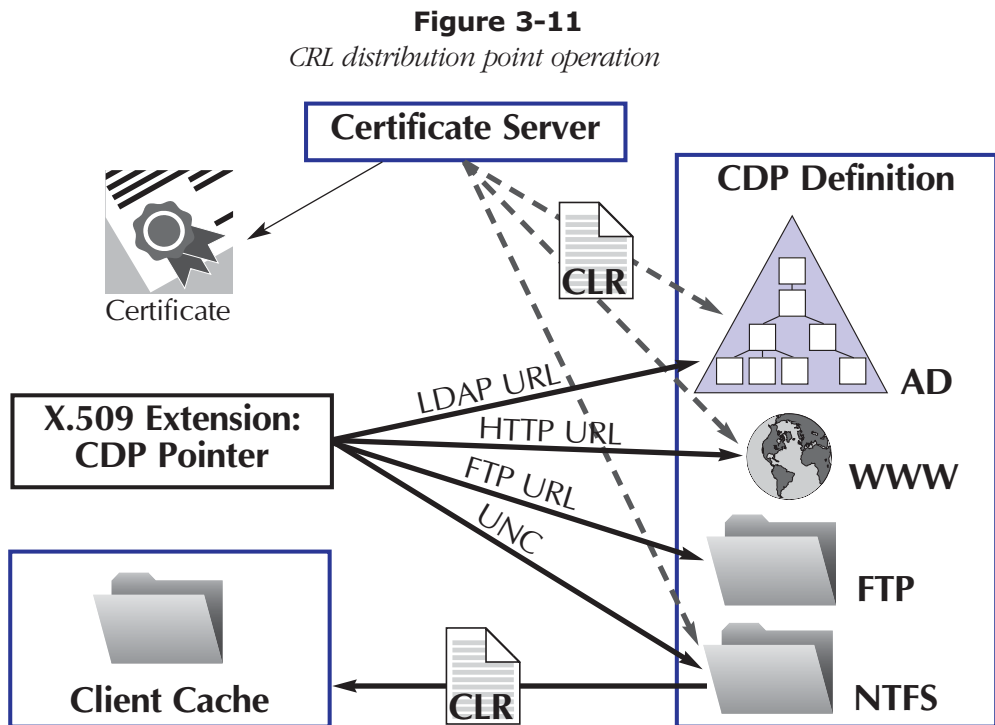
Figure 3-10
Browser-side SSL/TLS certificate revocation-checking option



In IIS, you can control revocation checking by using the `certcheckmode` metabase parameter (value 1 means that revocation checking is disabled; value 0 means that revocation checking is enabled). If the certificate has been revoked, or if the CRL is not available, IIS generates a 403.13 HTTP error.

CRL Distribution Points

CDPs offer a convenient way to automate revocation checking. Each certificate generated by a Windows 2000 or Windows Server 2003 CA can include one or more CDP pointers. These pointers are stored in the CRL Distribution Points X.509 certificate extension. A CDP can be a URL (HTTP or LDAP) or a file share. Once a certificate has been issued, its CDP pointers cannot be modified. How CDPs work is illustrated in Figure 3-11.



If a Windows PKI-enabled application that is CDP-enabled does not find a local copy of the CRL or delta CRL, it will check the certificate's CDPs for an up-to-date CRL or delta CRL. If a CRL or delta CRL is available from the CDPs, the application will download the CRL or delta CRL and cache it locally for the lifetime of the CRL or delta CRL. If a certificate does not contain any CDPs, the PKI-enabled application will query the certificate's issuing CA for a CRL or delta CRL.

In addition to automated revocation checking, CDPs also can increase CRL or delta CRL availability. Each certificate can contain different CDPs. If one CDP is unavailable, the PKI logic will try another CDP.

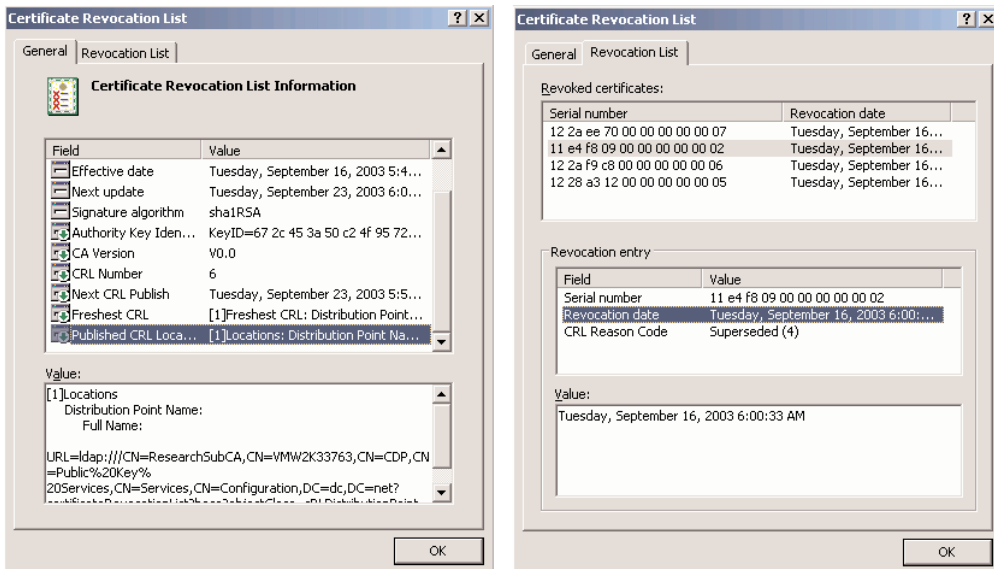
Complete CRLs

A CRL contains a time-stamped list of revoked certificates, which is signed by a CA and made available to PKI users in a public repository. In a CRL, each revoked certificate is identified by its certificate serial number. CRLs are defined in the ITU-T X.509 standard and RFC 2459, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile” (available from the IETF Web site at <http://www.ietf.org/rfc/rfc2459.txt>).

CRLs in their most basic format are known as *complete CRLs*. They are also referred to as *base CRLs* or *full CRLs*. Complete CRLs tend to be huge because revocation information accumulates in each of them. CRLs support versioning, but a new version automatically inherits all revocation information from the preceding version, so a CRL becomes no smaller until a certificate expires. Also, each new CRL version causes the client to download the complete CRL, which is not an efficient use of network bandwidth. As a result, many administrators configure longer CRL lifetimes. But long CRL lifetimes reduce the revocation information’s timeliness because new revocation information is not immediately available.

The first screen of Figure 3-12 shows the layout of a complete CRL issued by a Windows Server 2003 CA as it shows up in the built-in CRL viewer. Notice the presence of some typical CRL extensions: Effective date, Next update, CA Version, CRL Number, Next CRL Publish, Freshest CRL, and Published CRL Locations. A list of the revoked certificates on a CRL is available from the Revocation List tab shown in the second screen of Figure 3-12.

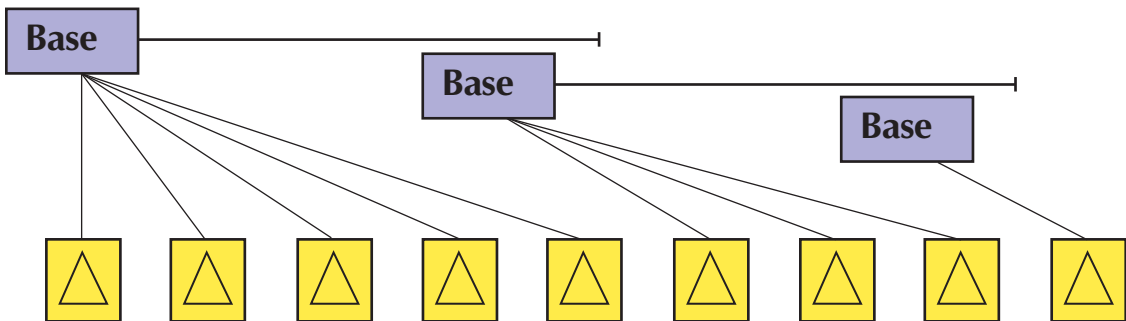
Figure 3-12
Windows CRL viewer



Delta CRLs

Delta CRLs resolve the complete CRL problems (bandwidth impact and revocation information up-to-dateness). As Figure 3-13 illustrates, delta CRLs are relatively small CRLs that contain only the revocation changes that have occurred since the most recent base CRL. Because delta CRLs are small, PKI clients can download them more regularly, and the CA can provide more accurate revocation information to its clients. Only Windows XP Professional and later Windows clients can check a certificate's validity against a delta CRL. Delta CRLs are not a Microsoft invention; they are defined in RFCs 2459, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile" (available from the IETF Web site at <http://www.ietf.org/rfc/rfc2459.txt>), and 3280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" (available from the IETF Web site at <http://www.ietf.org/rfc/rfc3280.txt?number=3280>).

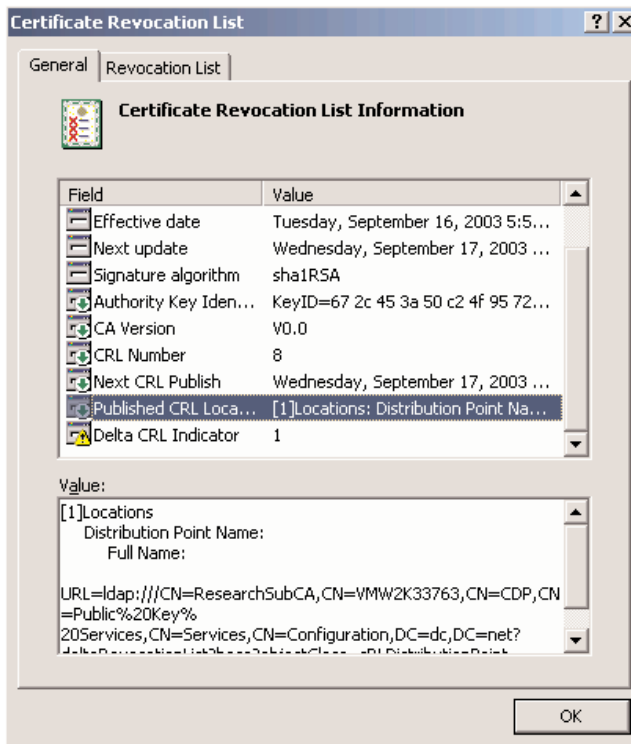
Figure 3-13
Delta CRL operation



As with complete CRLs, Windows clients also cache delta CRLs. If a base CRL expires, the client retrieves from the CDP a new base CRL that is specified in the certificate. If the base CRL is valid but the cached delta CRL is expired, a Windows client retrieves from the CDP only the delta CRL mentioned in the certificate.

Figure 3-14 shows the layout of a delta CRL issued by a Windows Server 2003 CA as it shows up in the built-in CRL viewer. Notice the presence of the Delta CRL Indicator extension, which means that this is a delta CRL, not a complete CRL. The value in the Delta CRL Indicator extension is the CRL number of the base CRL this delta must be associated with. As for a complete CRL, a list of the revoked certificates on a delta CRL is available from the Revocation List tab.

Figure 3-14
Delta CRL information in the Windows CRL viewer



Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) can offer real-time certificate-revocation information to PKI users. OCSP overcomes the main limitation of CRLs: the fact that updates must be downloaded frequently to keep the CRL list current at the client side. The OCSP protocol is defined in RFC 2560, “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP” (available from the IETF Web site at <http://www.ietf.org/rfc/rfc2560.txt?number=2560>).

OCSP requires a client-side component and a server-side component (the OCSP responder). When a user accesses an SSL-secured server, the OCSP client-side software sends a request for certificate status information to the OCSP server.

Windows XP, Windows 2000, and Windows Server 2003 do not support OCSP out of the box. Microsoft will add support for OCSP in the upcoming IE browser version (IE 7.0). OCSP software is available from the following vendors:

- Tumbleweed (formerly Valicert): www.tumbleweed.com
- Corestreet: www.corestreet.com
- Alacris: www.alacris.com

Formatting Check

The formatting check validates whether the content of an X.509 certificate conforms to the X.509 certificate format specification. This check also validates whether the common name in the certificate matches the name of the Web site as it is entered in the URL. If the certificate is ill-formed, or if the common names do not match, IIS will generate a 403.16 HTTP error. On the browser side, a security alert will be generated stating that “The name of the security certificate is invalid or does not match the name of the site” (illustrated previously in Figure 3-8).

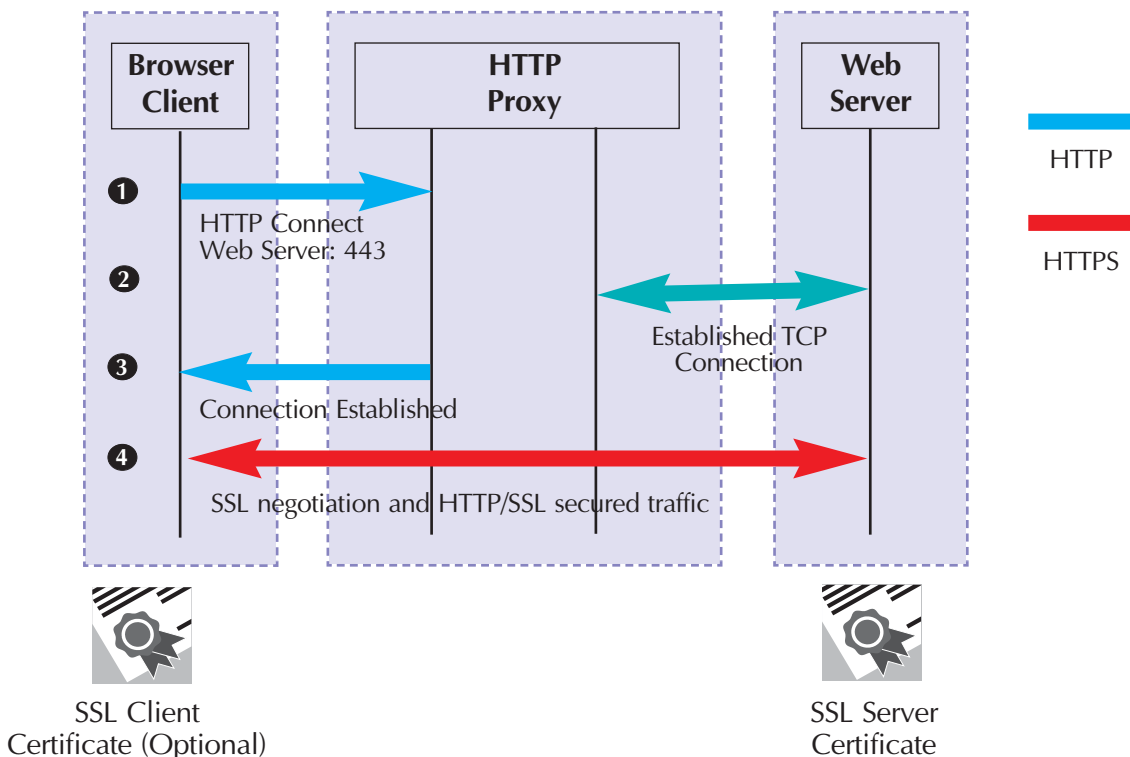
SSL/TLS in Firewall and Proxy Environments

SSL can be configured in different ways when you’re dealing with HTTP proxies. These different approaches can be categorized as either *SSL tunneling* or *SSL bridging*. HTTP proxies are used in almost every perimeter security solution that includes either application proxy-based firewalls or HTTP proxies, or both.

An SSL tunneling setup provides true end-to-end SSL: The SSL tunnel starts on the browser and ends on the Web server. This configuration is illustrated in Figure 3-15. SSL tunneling requires an SSL authentication certificate on the Web server and optionally an SSL client certificate on the client side.

Figure 3-15

SSL tunneling



The problem with SSL tunneling is that it breaks the role of an HTTP proxy. An HTTP proxy typically inspects the content of an HTTP request before it lets the request through. When you are using SSL, however, the HTTP request cannot be inspected because the HTTP content is encrypted by the SSL tunnel. The reason for this encryption is that SSL operates on the session level of the TCP/IP networking stack, while HTTP operates on the application level, and an SSL tunnel is always set up before the HTTP application-level traffic reaches the destination host.

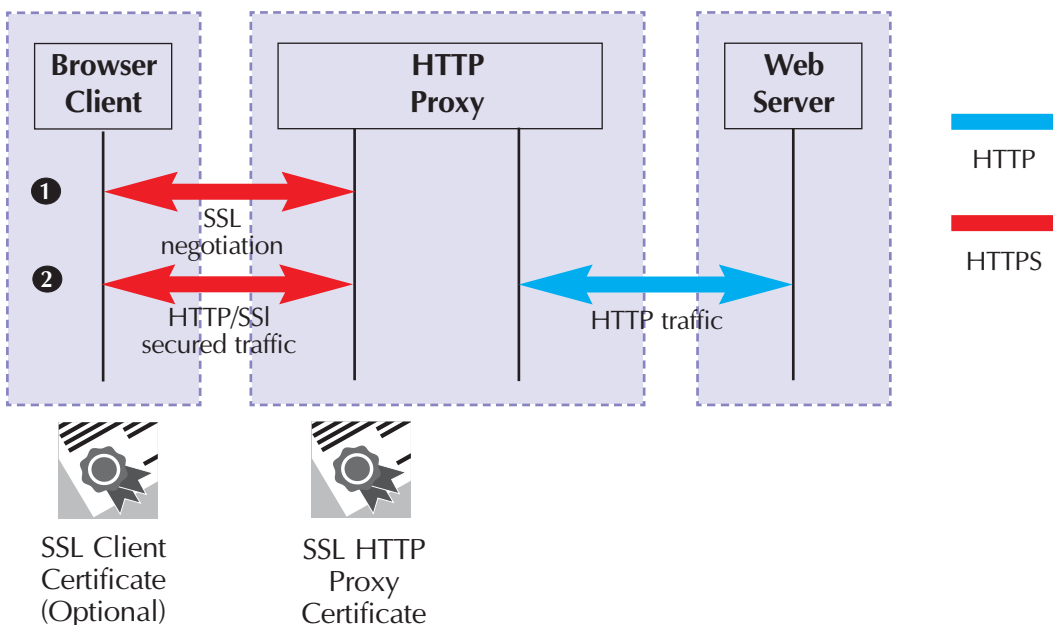
Does all this mean that letting an SSL tunnel flow through a firewall or HTTP proxy is impossible? No. To enable SSL tunnels to flow through HTTP proxies, SSL-enabled applications can use the HTTP CONNECT method. This method tells the proxy to ignore the content of an SSL session, and to simply forward the SSL packets to the destination host (in this case, a Web server). The HTTP CONNECT method is defined in RFC 2817, “Upgrading to TLS Within HTTP/1.1” (available from the IETF Web site at <http://www.ietf.org/rfc/rfc2817.txt?number=2817>). From purely a security point of view, it is important to stress that SSL tunneling is not the best approach because it basically punches holes in your firewalls.

The alternative to SSL tunneling is SSL bridging. Using SSL bridging takes away the security concern expressed earlier for SSL tunneling. SSL bridging basically means that the SSL tunnel is started or terminated on the HTTP proxy. As a consequence, there is no more end-to-end SSL tunnel setup. You can set up SSL bridging in different ways:

- Using a single SSL tunnel that starts on the client side and terminates on the HTTP proxy. This approach, illustrated in Figure 3-16, requires an SSL certificate for the HTTP proxy and optionally an SSL certificate for the client side (if strong client-side authentication is required).

Figure 3-16

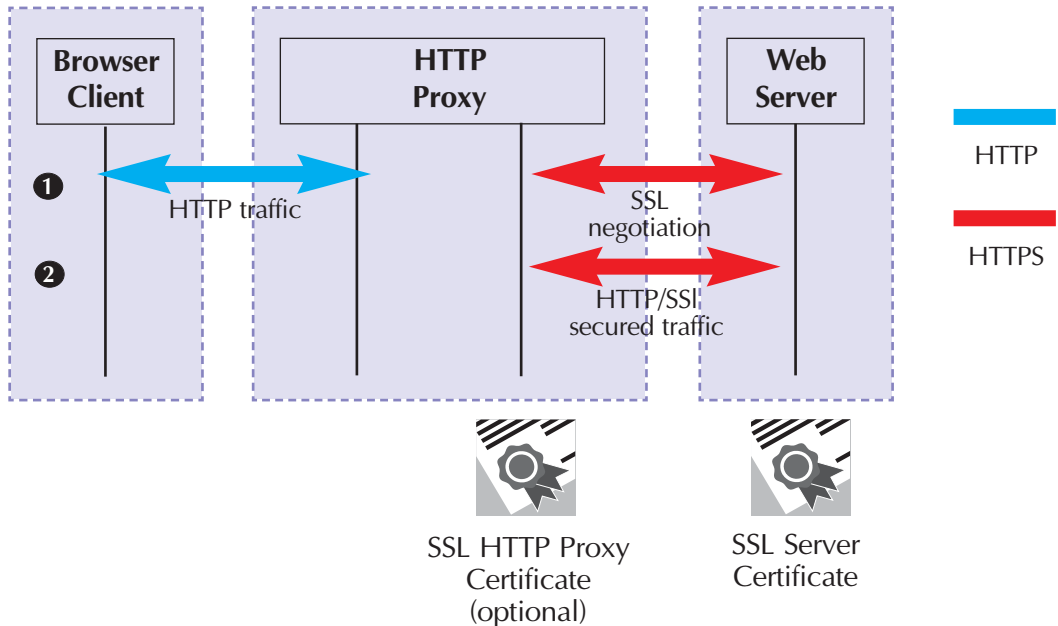
SSL bridging: Single SSL tunnel terminated at proxy



- Using a single SSL tunnel that starts on the HTTP proxy and terminates on the Web server. This approach, illustrated in Figure 3-17, requires an SSL certificate for the Web server and optionally an SSL certificate for the HTTP proxy (if strong client-side authentication is required).

Figure 3-17

SSL bridging: Single SSL tunnel terminated at Web server



- Using two SSL tunnels: one tunnel that starts on the client and terminates on the HTTP proxy, and another tunnel that starts on the HTTP proxy and terminates on the Web server. This approach is illustrated in Figure 3-18. Using two SSL tunnels requires an SSL certificate on the HTTP proxy and Web server and optionally an SSL certificate on the client side (if strong client-side authentication is required).

Figure 3-18
SSL bridging: Two SSL tunnels

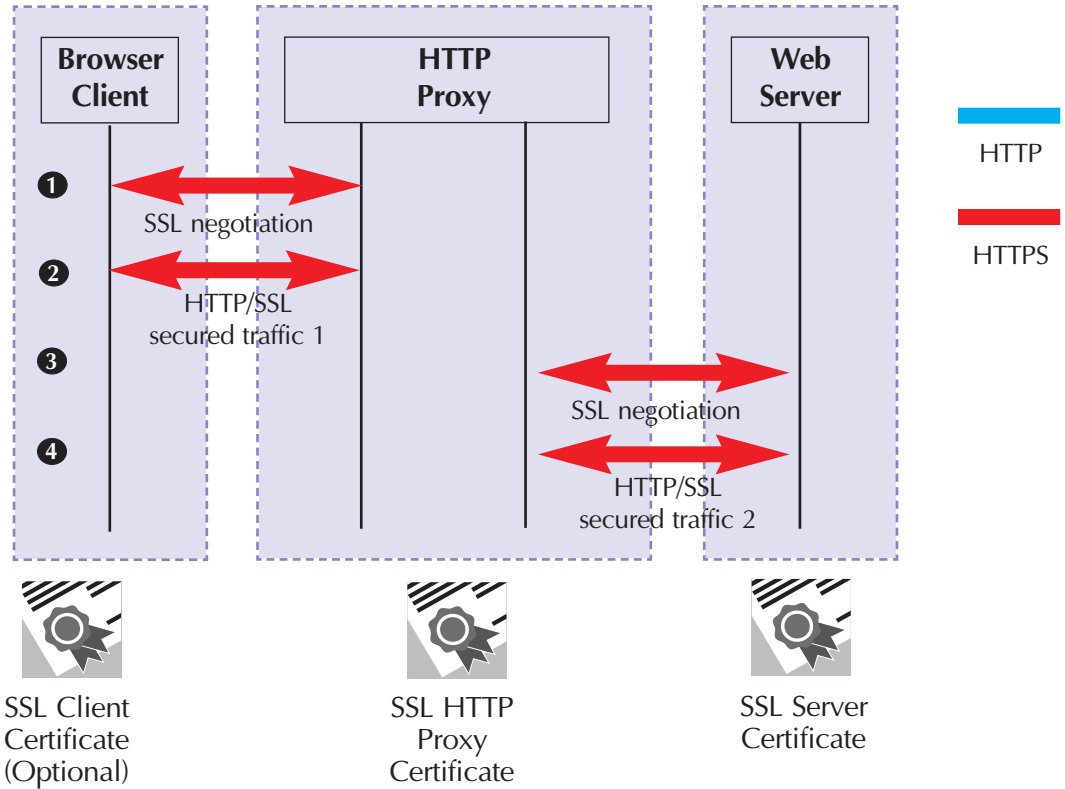


Table 3-1 shows the different SSL approaches and their advantages and disadvantages.

Table 3-1 Comparison of SSL and HTTP proxy approaches

| SSL Approach | Pros | Cons |
|---|---|--|
| SSL Tunneling | Provides end-to-end SSL. | Security hole exists on HTTP proxy and firewall level; the proxy cannot screen for malicious HTTP content any more. |
| SSL Bridging—Option 1: Single SSL tunnel, started on client and terminated on proxy | <ul style="list-style-type: none"> Includes HTTP content inspection on HTTP proxy and firewall level (no security hole). Offloads SSL processing from Web server. | <ul style="list-style-type: none"> Provides no end-to-end SSL. HTTP traffic goes in the clear between proxy and Web server. |
| SSL Bridging—Option 2: Single SSL tunnel, started on proxy and terminated on Web server | <ul style="list-style-type: none"> Includes HTTP content inspection on HTTP proxy and firewall level (no security hole). HTTP traffic is encrypted between proxy and Web server. . | <ul style="list-style-type: none"> Provides no end-to-end SSL HTTP traffic goes in the clear between browser and HTTP proxy. |
| SSL Bridging—Option 3: Two SSL tunnels | <ul style="list-style-type: none"> Includes HTTP content inspection on HTTP proxy and firewall level (no security hole). HTTP traffic is encrypted both between browser and proxy and between proxy and Web server. | |

Optimizing SSL Server-Side Performance

The asymmetric cryptographic operations behind the certificate-based authentication in SSL/TLS can have an impact upon Web server performance. To deal with this performance issue, you can do three things:

- Use hardware crypto-accelerator devices. These devices offload the main system processor by using a dedicated processor for the cryptographic operations. Table 3-2 gives some example devices and vendor-contact information (this is not an exhaustive list).

Table 3-2 SSL acceleration-device products and vendors

| Vendor | Device | More Information at: |
|----------|---|--|
| Broadcom | CryptoNetX BCM800 CryptoNetX BCM1600 CryptoNetX BCM4000 | http://www.broadcom.com |
| F5 | Big IP SSL Accelerator | http://www.f5.com/f5products/products/bigip/security/ |
| HP | Atalla AXL 300 Atalla AXL 600L | http://h18000.www1.hp.com/products/servers/security/axl300 http://h18000.www1.hp.com/products/servers/security/axl600l/index.html |
| Intel | Netstructure | http://www.intel.com/support/netstructure/index.htm |
| nCipher | nFast Ultra nFast 800 nFast 300 | http://www.ncipher.com/nfast/index.html |
| SafeNet | Rainbow Cryptoswift | http://www.rainbow.com/products/cryptoswift/PCI.asp |

- Limit the Web page size. Limiting the Web page size reduces the amount of data that needs to be cryptographically processed.
- Reuse cached SSL sessions to limit the number of SSL negotiations. On the Microsoft IIS Web server, for example, you can fine-tune the SSL caching behavior by using the registry parameters in Table 3-3. These parameters are located in the HKEY_LOCAL_MACHINE \system\currentcontrolset\control\securityproviders\schannel registry key.

Table 3-3 Microsoft IIS SSL session cache tuning parameters

| Registry Parameter | Data Type | Meaning |
|--------------------|-----------|--|
| MaximumCacheSize | REG_DWORD | Maximum number of SSL sessions to maintain the cache |
| ClientCacheTime | REG_DWORD | Time in milliseconds to expire a client-side cache element |
| ServerCacheTime | REG_DWORD | Time in milliseconds to expire a server-side cache element |

SSL and Load Balancing

When you're using SSL in a Web load-balancing environment (e.g., in a Web farm setup), you must make sure that your load-balancing solution supports sticky sessions. Sticky sessions assure that the HTTP connection always returns to the same Web server in a server farm during a load-balanced, SSL-secured HTTP connection.

The electronic shopping-cart example clearly illustrates this problem associated with SSL and load balancing, which is also referred to as *SSL persistence*. A shopping-cart is a logical repository for the items a customer selects while shopping online. The selected items are typically maintained on the Web server to which the customer first connected. If at any point during the session the client is switched to another server, the customer's shopping-cart data is lost.

HTTP load-balancer devices can use either of two mechanisms to assure unsecured HTTP connection persistence: the source IP address or HTTP cookies.

1. When they use *source IP persistence*, load balancers identify a user by his or her source IP address and then use this identifier to link users to the appropriate server. This method cannot be used, however, when the HTTP connection travels across proxy servers and Network Address Translation (NAT) devices. Proxies represent different users by a single IP address; NAT devices change IP addresses throughout the life of a session.
2. *Cookie persistence* uses a browser cookie to uniquely identify a user. In this case, either the Web application or the load balancer itself hands out a cookie to a user at the start of a session. The user's browser then automatically returns the cookie during each Web server hit. By tracking the cookie information, load balancers can determine which Web server should receive the subsequent traffic. Cookie persistence does not work with SSL. SSL encrypts all data, including cookies; as a consequence, load balancers are unable to inspect the browser cookie.

SSLv3 introduced a solution specifically for SSL persistence called *SSL session ID-based persistence*. SSLv3 moves the SSL Session ID, a unique 32-byte session identifier, out of the encrypted portion of the data into a clear portion. Load-balancing and content-switch vendors are able to read this unique identifier and so can use it to balance the traffic to the appropriate server.

SSL session-ID based persistence works with all Microsoft IE browsers up to version 5.0 (which is not included). In IE 5.0, Microsoft changed the behavior of its SSL libraries to force a renegotiation of a new SSL session every two minutes. This action breaks the SSL session ID-based persistence mechanism.

Given the above issue with IE 5.0 and later browsers, the only viable solution to provide SSL persistence is to use the combination of a load-balancer and an SSL accelerator device. In this scenario, when SSL-secured HTTP traffic arrives at the load balancer, the traffic is redirected to an SSL accelerator. The SSL accelerator decrypts the content, including cookies, and then sends the decrypted content back to the load balancer. The content switch can then inspect cookies and other URL data and use them to provide SSL persistence.

This solution also lets load balancers set persistence cookies for outgoing HTTP traffic. Note that this method terminates the secured SSL tunnel at the load balancer-SSL accelerator level. Today's load-balancing networking devices from vendors such as Cisco, Sonicwall, F5, and others commonly support this combination load-balancer/SSL accelerator device solution.

Conclusion

This chapter illustrates the complexity and importance of certificate validation for SSL-secured Web traffic. The chapter also delineates the following set of important deployment issues you must consider

- SSL/TLS and HTTP proxy and firewall combined operation. Both SSL tunneling and SSL bridging have advantages and disadvantages. Given the two following issues however, SSL bridging with tunnel termination on the load balancer-content switch level might be the only workable approach.
- SSL/TLS has a performance impact on the Web server side. When you are securing large parts of your Web site using SSL/TLS, you must consider SSL accelerator devices, either on the Web-server level or on the content-switch, load-balancing level.
- The combination of SSL/TLS and HTTP load balancing. To assure SSL persistence, you must terminate the SSL tunnel on the level of the load-balancing device.

In the next and final chapter (chapter 4) of this eBook, we will focus on how to enable and configure SSL/TLS for secure LDAP, Network News Transfer Protocol (NNTP), and SMTP communications. The process of enabling and configuring SSL/TLS is illustrated in Active Directory (AD), IIS, and Exchange environments. Chapter 4 will also provide a set of SSL/TLS best practices for each of the above applications.