Feature: SSL – it’s not just for commerce anymore

by Hugo Curbelo and Russ Lipton

[Editor's note: This article resides in "Notes Today", the technical Webzine located on the http://www.notes.net Web site produced by Iris Associates, the developers of Domino/Notes.]

Introduction

Outside the quiet, secure and friendly land of Domino lies an Internet world filled with excitement, thrills and danger. Notes-to-Notes networks can authenticate, encrypt and ensure trust. But what happens when you connect to a non-Domino server? Are you communicating with a friend or a stranger?

SSL (Secure Sockets Layer) was created originally by Netscape to add a certificate-authenticated encryption layer around normal HTTP transmissions. It will soon become the routine first-level protocol for protecting all Internet protocol and application traffic. This article will help you understand what SSL is, how it co-exists with existing Domino and Notes security protocols, and how Domino implements SSL support.

Why SSL Matters To Notes Users

Let's cut straight to the chase. Notes users already have a wealth of authentication mechanisms that SSL can't touch, providing that communications are restricted to an intra-Notes environment. But how many of your mail communications this very day are either sent to or received from non-Notes servers?

If SSL is enabled for your POP3 client, all data sent across the wire, including your password, is encrypted. It can only be read by the receiving party. Without SSL enabling, your mail is transmitted in cleartext form. So is your password. Effectively, the authentication and protection you take for granted as a Notes user is completely nullified.

SSL matters, big time.

TCP/IP, SSL, and Netscape

The simplicity and flexibility which has fueled the Internet's runaway success has a darker side. It is reasonably simple - too simple - for a knowledgeable hacker to impersonate a person or an organization during an Internet session. Strictly speaking, you do not know for sure with whom or to what you are connecting during a conventional HTTP session.

The TCP/IP protocol forms the underlying foundation for scaleable Internet communications. Unfortunately, TCP/IP is, by its very nature, unsecured. However, using Netscape's SSL protocol in conjunction with TCP/IP brings reasonable first-level security to this previously unsecured environment.

Perhaps the most important function that SSL brings to standard TCP/IP and HTTP sessions is the ability to mutually authenticate servers and clients. By enabling SSL, you can establish trusted relationships between clients and servers using certificates and digital signatures. While this ability has long been available to Notes users, it was a groundbreaking development in security across TCP/IP.

While early use of SSL was HTTP-centric, SSL is now coming into use to wrap many Internet protocols and applications. SSL can be used, for instance, to encrypt POP3 client-server communications and may soon be used to wrap Java applets as they are downloaded from servers. Internet Message Access Protocol (IMAP) and Lightweight Directory Access Protocol (LDAP) protocols will also receive SSL support.

In other words, SSL should be understood as a simple, general, useful specification that developers can implement to protect any bi-directional network communications. While we confine most of the following

discussion to servers and clients, these can be understood to represent any two communicating entities that need to guarantee a trust relationship before sharing critical data.

How SSL Works

SSL works by providing three security services, each of which use public-key encryption techniques.

The first and most basic function of SSL is message privacy. SSL offers feature support so applications can exchange and authenticate user names and passwords without exposing them to eavesdroppers. Hackers can use IP sniffers to download copies of all packets that pass between a client and server during a session. This information is then available in an unencrypted, cleartext format. With SSL however, all transmissions following the initial handshake are encrypted to prevent such transmissions from being read.

Two different types of encryption team together to ensure message privacy: public-key and symmetric key encryption. Every piece of traffic between the SSL server and SSL client is encrypted using a key and an encryption algorithm negotiated during the SSL handshake which occurs at session initialization.

Secondly, the SSL protocol ensures that messages between the sender and receiver have not been tampered with in any way between the two points. This maintains session integrity and keeps a secure channel open between the client and server.

To do this, the SSL server uses a combination of special mathematical functions called hash functions and a shared secret to encrypt the data with the strongest available cipher. If, during a session, a packet is damaged or indecipherable, we must assume that tampering has occurred. By constantly verifying and preserving the integrity of messages, the SSL server is able to ensure safe and continuous transmissions.

The final and most important function of SSL is mutual authentication. This is the process by which the client and server can convince its peer of its identity through the exchange of X.509 certificates. A server's identity is coded in the form of a public-key certificate that is exchanged during the SSL handshake.

The SSL Handshake

SSL was designed to make its security services as transparent as possible for the end user. Typically, a user follows a standard HTML page link that connects automatically to an SSL-capable server.

Most SSL-capable Web servers take the SSL connection on a different port (443 by default) than standard HTTP requests (port 80). Emerging protocols will be assigned additional ports by default (for instance, port 991 for IMAP). When the application establishes a port connection, it initiates a handshake that establishes the SSL session.

From this point forward, all communications between the client and server are encrypted and message integrity checks are performed until the SSL session has been disconnected. In an SSL session, the handshake only occurs once.

Three events occur within the confines of the SSL handshake. First, client and server (or server and server) exchange X.509 certificates to prove their identity. The certificate is checked against expiration dates as well as for evidence of the non-tampered signature of a trusted third-party authority.

During the second handshake event, the client randomly generates a set of keys to be used for encrypting the messages. This, in conjunction with the server's public keys, creates a total of four keys, two of which are used to encrypt the data transmission. Basically, the client encrypts a cipher key with the public key that it received from the server. The client then sends the encrypted message to the server. The server now decrypts that message with its private key. If all is well, the keys that were exchanged can now be used throughout the session to protect the data from lurkers.

As a part of the mutual negotiation phase, the client and server determine the hash function (for integrity) and the message encryption algorithm (for encryption). In this process the server asks the browser to provide a list of all possible ciphers. The server then encrypts the information using the strongest allowable cipher that it holds in common with the client.

Leveraging the Domino Architecture

The key point about the Domino SSL implementation is our ability to leverage our architecture to ensure a single point of control. Because the Domino network abstraction layer is completely portable across platforms, the same simple routines apply throughout, even though different platforms communicate differently with their varying networks.

By contrast, many vendors must directly recode SSL routines in each of their products. Consequently, even though functionality is consistent for their users, code maintenance and performance varies. SSL may run well on one product and crash on another.

Basically, we have hooked into the send-receive routines for network communications to make these routines SSL-aware, once a SSL handshake has been completed.

We have purposely not hooked connect routines, since we can envision a computing world where applications negotiate the specific type of encryption suitable for a given transaction on the fly during the connect phase. The encryption level required for a CIA transmission is of a different order than that required for a casual e-mail.

A nice side-benefit of our approach is that it paves the way to enabling single-port solutions. Currently, non-SSL HTTP transmissions require a port assignment and SSL transmissions require assignment of another port; IMAP demands still another. The potential that a port will be unavailable rises accordingly. If we enable applications to negotiate their port as well as their encryption, we can increase the reliability and the performance of secure network connections.

Implementing SSL For Domino

Please note in our discussion about Domino implementation of SSL that we are describing network API internals at a deep level. We expose most of our API to Notes developers by design, but we hide these routines from view, mainly because they are tightly bound to the Notes kernel.

In fact, these routines are extremely simple, given their expressive power. Fundamentally, we added a new routine to the common network code for managing Web communication (let us just call it ICT, though it has a longer system name). ICT makes the appropriate API calls to establish the SSL dialog with the remote Web server.

The routine is invoked after the Web browser retrieve process is fully initialized and the TCP/IP connection has been established with the remote server. Once the SSL handshake has been properly established, no other action is required on the part of the browser.

When the browser has completed communications with the remote server, no special work needs to be done to disconnect the SSL session. We describe this from the perspective of the SSL process below.

Implementing The SSL Handshake

Looking a bit more closely at the SSL handshake, we track in our implementation the steps already discussed in this article when two parties make contact across the network.

1. A handshake process is begun. The initiating server sends a message with a (presumably) valid certificate. The recipient client checks the validity of the certificate and of the digital signature that

accompanies it. If all is well, the handshake concludes with an agreement to proceed. If something is fishy, the communication is terminated by the receiving server.
2. The internal Notes ICT routine is hooked to each send operation to perform appropriate encryption of a message preparatory to its routing to the target.
3. In the case of an incoming, encrypted message, another internal ICT routine is hooked to the receive operation to execute the appropriate decryption algorithm.
4. Upon disconnect, memory or other computing resources are automatically released by Domino.

The Role Of Certificate Authorities

The SSL handshake manages the exchange of certificates and digital signatures. Domino fully supports industry-standard RSA encryption and CAs (Certificate Authorities).

The role of certificate authorities is to enhance mutual trust. In street language, they stand as a third party between two other parties to verify that one of the parties is not pulling the wool over the over. The process works like this:

1. The certificate authority signs a certificate (yours, we will say) with their private key to validate it.
2. You send me your certificate.
3. I use the public key given to me by the certificate authority to check that your certificate was indeed issued by them and that it has not been altered since it was issued. Of course, to do this, I have to be able to find the public key for your certificate authority in a reservoir of keys that are managed by my server.

As in so many areas, Notes has long anticipated what is today's "new news" on the Internet. Notes certifiers are, in fact, nothing less than a true certificate authority, functioning within a Domino environment, of course. Consequently, understanding the role of a Notes certifier will help you both to understand and interact with Internet-centric certifiers.

What About Self-Signed Certificates?

Self-signed certificates have become widely known in recent months, but what are they and are they reliable? Self-signed certificates provide a container which certificate authorities (for instance, VeriSign) use to distribute their public key or used as an untrustworthy certificate created for the remote system.

The level of trust to be placed in a self-signed certificate can never be greater than the trust in the way in which they were delivered. A certificate handed to you personally by someone who themselves are known to you as trustworthy, will be trustworthy. A certificate you receive over the Internet from a party unknown to you and unverifiable is, of course, untrustworthy. Caveat emptor. The trade-off here is between distribution convenience and reliability.

The term "self-signed" means that the authority itself ("me") is sending a certificate that is not itself attested by a third-party party. It says to the recipient, "here is my certificate. I issued it and I verify it myself." Asking about trust in this context is like entering an endless loop or an Abbott and Costello routine: "Who are you?" "Me." "Who is me?" "Whoever I say me is." "OK, who do you say 'me' is?" "Me." So it goes.

Self-signed certificates are used in practice as a convenient distribution mechanism for handing out keys to developers, with verification deferred until a later time. A server administrator may issue them to give an untrustworthy certificate for a server to use. The administrator is, in effect, telling its consumers that they have reached the server they requested.

Domino can issue self-signed certificates today. In the near future, Domino, will make it possible to distribute and manage these certificates in the way that VeriSign does today. We expect this to increase the number of certificate authorities, especially within corporations.

After all, most corporations can ensure at least the same level of trust internally by issuing their own certificates, while bypassing the need to pay an outside vendor for the privilege, a privilege which has to be renewed annually as certificates expire. Self-signed certificates will probably continue to exist in the gray area between full Internet security and security judged "good enough" for most internal operations.

Domino, SSL and Notes Security

You can conveniently enable SSL today, both on the Domino server and client-browser side (e.g. Web Navigator). Consult your product documentation for details. In the near future, SSL will be furnished on-board with Domino. You will be able to turn SSL on or off for a range of Internet and Intranet-supported applications and services.

Our SSL implementation does not in any way abrogate the availability of or our commitment to Notes-specific security. Existing Notes network protocols retain full support. In fact, you can and should continue to use the Notes security, authentication and encryption model wherever possible. There are two reasons for this:

First, in an intra-Domino environment, the Notes certification model is richer than SSL. We employ a deep, hierarchical naming structure which simplifies both the issuance of certificates and system verification of those certificates. Simplifying, we follow chains of trust both up and down within a certificate tree and across trees to other chains that have been verified for trust. In effect, Notes maintains a semantic certification network, where the Internet model is merely syntactical.

Second, SSL in no way affects your management of application access, using ACL and other mechanisms. SSL is indifferent to who actually accesses an application, once the veracity of the connection itself is established. This is critical. After all, authentication and encryption are merely a minimum, not a maximum, condition for mission-critical computing. Within a Notes environment, we can open or close access to the field level.

Therefore, Internet security solutions so far can be viewed as a subset, not a superset of onboard Notes security. They are a vital, usable subset (after all, that is the entire thrust of this article). But, they are a subset. That said, our implementation of SSL fully supports open, Internet standards, protocols and applications up to their current stage of maturity. Over time, we expect the Internet to match or surpass today's Notes security arsenal. We are working hard to make that happen.

SSL Version Two Or Three?

SSL Version 3 is coming onboard to replace Version 2. Version 3 is not a dramatic upgrade, but it yields needed code improvements and enhances SSL portability. Applications should experience a noticeable reduction in message traffic between client and server, enabling faster handshakes. Security reliability is tweaked by strengthening handshake termination and end-of-message authentication. Netscape is also positioning V3 to incorporate smart-card encryption technologies.

SSL V3 does add support for several new key-exchange and encryption algorithms and has also enhanced its approach to the certificate request protocols. Servers and clients can now exchange certificate chains, enabling hierarchies that are more than two certifications deep. While V2 defined client certificates, they were not used much due to practical marketplace issues. Mutual authentication is only now coming into prominence.

While the groundwork for SSL-style technology had been laid years earlier by pioneers in the mathematics of encryption, Netscape must be given full credit for recognizing and responding to a critical market requirement. Its simplicity raises some questions about future direction as Internet security requirements become more complex.

The Bottom Line: SSL Now

Nothing comes entirely for free. Networks experience a modest performance hit whenever SSL is enabled. Still, compared to the internal gridlock of a 28.8 modem connection or even an ISDN line, degradation is imperceptible. If you think that life slows down over the Web for humans, think how your server feels. On second thought …

Since the performance hit is negligible, and the security benefits for connection to non-Domino servers is real, you should make SSL-enabling a complementary aspect of your Notes network management today. Since our implementation abstracts the network and provides a single design and management point, you can gain Domino SSL protection across platforms and the entire enterprise essentially for free.

Momma always told you to watch out for strangers. Implementing SSL will help Momma feel better and your user will sleep easier.

ABOUT THE AUTHORS

Hugo Curbelo is the Principal Software Engineer for the Network Development Group at Iris. He has been with Iris for 3 ½ years and currently specializes in, of course, SSL, TCP/IP and NetBios designs and implementations. He reports that he remains a Boston Red Sox fan, despite everything.

Russ Lipton is an independent consultant and frequent contributor to Notes.Net. Though still a relative Domino newbie, Russ has fifteen years of Fortune 500 consulting experience in software systems. He has been heard to ask, "Why didn't anyone tell me about Notes sooner?" Russ thanks Joshua Lipton for his research assistance on this article.