by
Frederic
Dahm

Level: All
Works with: Domino 5.0
Updated: 02/01/2001

The convenience and ubiquitousness of the Web has caused many Notes/Domino customers to decide to use the Web for employee mail access, particularly while employees are traveling. Webmail, and the newly announced iNotes Web Access, provide the user interface for the user's access, while Domino serves up the data with fidelity and security.

Even with Domino's standard security features, however, there are times when additional security measures are prudent—or even mandatory. Such is the case for companies dealing with extremely sensitive information—companies with government contracts, for instance, or those that handle financial data.

This case study takes a look at the basis for securing Web access to mail and for looking at the possibility of securing traffic within the customer's network. It was originally written to help a customer consider some of the options available to them when determining the services they want to provide and the mechanisms for providing them. It should not be taken as a definite architecture document, but rather as a presentation of possibilities from which the customer could choose.

This article assumes knowledge of Notes/Domino architecture and its security features.

Customer requirements

The customer in question needed to provide a secure messaging architecture to its employees while they were traveling or were outside of their primary area of work. The requirements were that the system providing e-mail services be universally accessible and that the client (and/or mechanism) for accessing the e-mail be equally as universal and ubiquitous.

The simplest and most straightforward way of meeting these requirements was to use the Internet as the communication medium, a Web browser as the messaging client, and a Web application server for providing messaging services.

The Web browser would access the specific mail file of the user via HTML using the HTTP protocol. The Web application server would provide—in addition to mail store and forward services (typically using SMTP)—access to the user's mail box via HTML using the HTTP protocol, and everything (message contents and messaging interface elements) would be available.

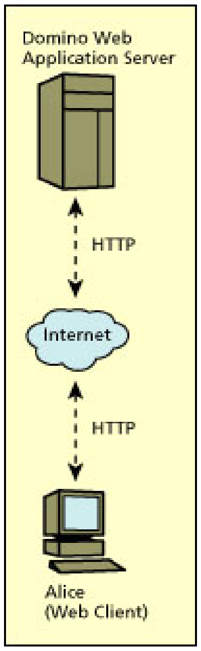
The Web browsers might be on terminals found at airports, workstations provided to airline customers in business class lounges, or at one of many Internet cafés. In addition, the company might provide employees with company laptops fitted with modems and the proper software. Providing such services to Web browsers is, of course, something that the Domino Web application server is apt.

The main issue was to determine how to do this in as secure a fashion as possible. Providing access is one thing; restricting access to only those people who are authorized to have it is another thing. The rest of this article discusses the fundamental security issues related to providing security around these services and ensuring adhesion to the company's policies regarding confidential information.

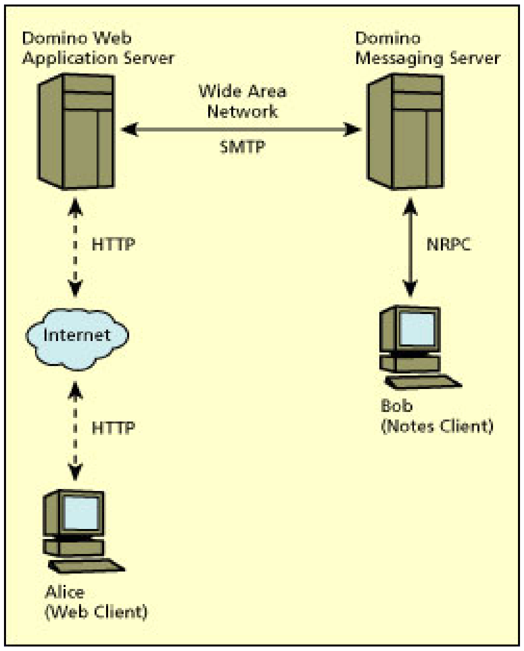
The basic architectures

Before delving into security ramifications of providing the required services, it is important to understand the details of the architecture of a ubiquitous mail access solution. Then it will be easy to add the necessary security services and discuss the caveats that exist, if any. For the purposes of discussion, let's name the company's employees Alice, Bob, Carole, Dennis, Eva and Fred.

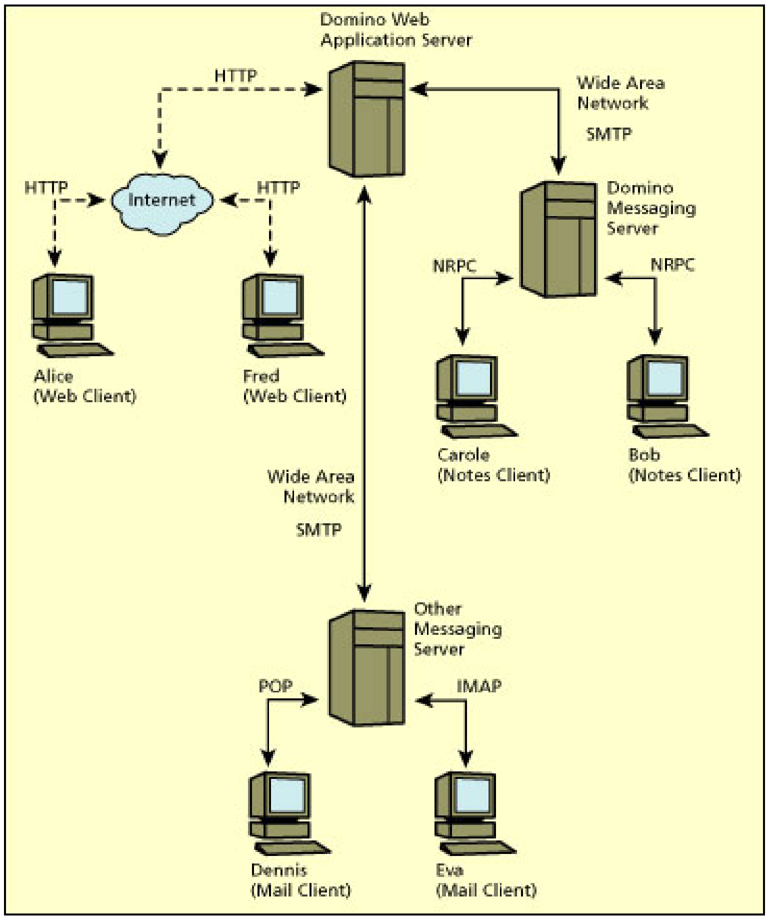
Architecture 1
The illustration below details the communication between one workstation and one server, which is typically what is involved when a user (in this case, Alice) accesses her mail over the Internet.



Architecture 2
The next illustration details the same communication between one workstation and one server, but adds an additional server and an additional user. Here, one user (Alice) accesses and exchanges mail over the Internet with another user (Bob), who is in another department and who uses a Notes client.



Architecture 3
The following illustration details the same configuration as above but adds an additional server and additional users. In this case, Web users (Alice and Fred) are accessing and exchanging mail over the Internet to other users (Bob and Carole) in another department, who are using the Notes client. Additionally, they are also exchanging mail with other users (Dennis and Eva) who are using another brand of messaging server (it doesn't matter which, assuming it is able to route via a common messaging protocol, such as SMTP) and POP or IMAP messaging clients.



A step-by-step approach

Given the increasingly complex architecture, it is important to take it one step at a time and think through the threat model (who would attack, why they would attack, and how would they do it) and what security measures can be applied to ensure that the information is as safe as possible, while keeping access as universal as possible.

When designing a security architecture you should perform the following steps:

1. Determine the sensitivity of the information and what the consequences would be if the information were to be captured and placed in the wrong hands.

2. Determine the threats the information can be under. This will help you formulate the risks involved in the solution and what must be done to contain or minimize the risks. "Minimizing" is a good term, as there are no absolutes in computer security. For each security measure, more sophisticated countermeasures are built that erode the defense. It is thus necessary to build the defense, but to also realize the remaining risks and determine whether it is possible to live with them. If so, then the solution can be implemented; if not, then either the solution is not implemented or further security measures are applied until you arrive at a situation where the risk is minimized to a point you can live with.

3. Build the system and monitor it. To draw an analogy, it is not only necessary to build a fence to prevent intruders from breaking in, it is also necessary to guard it. Doing so minimizes the risk and ensures that if a better intrusion mechanism is built, it can be circumvented with vigilance.

As an additional step, a third-party might be contracted to test the implementation and to ensure that unforeseen security vulnerabilities are not present, that all configuration parameters have been properly set, and that all security mechanisms have been appropriately applied.

In this case, the company knows it is dealing with highly confidential information, which means that special care is needed. The next step is to determine the threat model.

Understanding the threat model

The portion of the architecture most susceptible to attack is the one that lies outside of the company's network, namely, the portion where information is exchanged over the Internet between the Web client and the server. However, there also must be some attention paid to communication within the company network, since threats can come from insiders as well.

Who are these attackers and why do they do it?
In dealing with information exchanged outside of the company network, attackers could be anyone on the Internet with the means to intercept packets exchanged between the Web client and the Web application server. In dealing with information exchanged inside of the company network, it could be insiders that are either employees or contractors.

Attackers were at first indiscriminately labeled as *hackers*. However, a distinction had to be quickly made: not all hackers are malevolent. A good portion are really seeking mastery of computer science and of lateral thinking (which is basically the ability to think outside of the proverbial box). So a new definition was made of malevolent hackers and they were labeled *crackers*, whose main goal was, in their most benign form, the defacement of Web sites and in their most malign form, the misappropriation, dissemination, and possible destruction of knowledge capital.

These days, even these definitions leave quite a lot to be desired. The categories are too broad and don’t account for specific motives. So, more current categories include Script Kiddies, White Hat Hackers, Black Hat Hackers, Hacktivists, Corporate Spies, and Insiders (including employees). See the **Types of hackers** sidebar for descriptions of these categories.

How would they attack?
Let's consider security threats it from two perspectives: outside the network and inside the network.

Outside the network, eavesdropping is the main type of attack. It involves simply capturing the packets exchanged between the client and the server. This is called a passive attack, since information is obtained in a passive manner, simply through monitoring the communication channel between the two devices.

Eavesdropping leads to two things being captured:
- The data of the communication itself
- The user ID and password of the user if only basic HTTP authentication is used. This could lead in the attacker having the means to log onto the messaging server and being able to read all the e-mails intended for the user.

There are also man-in-the-middle attacks where the local DNS server is hacked and the user is brought to a fake server and thus, could be fooled into divulging authentication information to that machine.

Another consideration, which in itself is not an attack, is the fact that Web browsers cache a certain amount of information and thus, some information exchanged during the session could be stored in the Web browser's cache, ready to be examined by the next person using the workstation.

The server store is also at risk, as someone could mount an attack and get access to the data on the server. Depending on the OS used, there are many ways to get access to the file system and the files it contains (buffer overflows and identified yet unpatched security vulnerabilities are chief among them).

Inside the network, the same risks exist as outside the network. A passive attack can be mounted by which communications can be eavesdropped, and active attacks can also be mounted with insiders gaining access to the mail files on the servers, making a copy of the mail file, and viewing it locally with the messaging client.

Security mechanisms for architecture 1

Now that you understand the threat model, it's easier to see how to apply security mechanisms to the architectures shown above to ensure the confidentiality of the information, no matter from where or which client the data is being accessed. Let's begin with the first—and simplest—architecture: a user accessing her e-mail over the Internet.

Encrypting the communications channel

The first thing you can do is to encrypt the communications channel to ensure protection against eavesdroppers. The way to do it in a consistent, standard manner is to use SSL (Secure Socket Layers). The question is whether to use client certificate authentication in addition to server certificate authentication. The decision to implement one or both is based on the level of security required and the mechanisms at the user's disposal.

In either case, SSL Version 3 should be used. Here are the differences between the two versions of SSL:

- **SSL Version 2** supports server certificate authentication only. This means that only the identity of the server is verified. The basis of this verification is via an X.509 certificate that is signed by a Certificate Authority that is trusted by the browser used. Most commercial implementations of SSL on the Internet use SSL v2.
- **SSL Version 3** supports server certificate authentication, but also has optional support for client certificate authentication. When using both, it means that both the identity of the client and the server are verified. You still need to have an X.509 certificate for the server, which establishes the SSL security. However, you may also choose to implement client certificates in order to ensure the identity of individual clients. SSL v3 adds support for additional ciphers and can be used even if client certificate authentication is not enabled.

As far as the certificates are concerned, X.509 *server* certificates uniquely and securely identify the server. As such, they are a prerequisite for implementing the SSL protocol. X.509 *client* certificates are optional. They enable you to further extend the security model to uniquely and securely identify each client.

In this customer's case, however, the implementation of X.509 *client* certificates was problematic. It required the following steps:

1. The client needs to issue a Certificate Request. To do this, he/she goes to the server's Web site, selects the Request Client Certificate option, enters the Distinguished Name components, enters Additional Contact Information, selects the Encryption strength (1024, ..., 512), and then clicks the Submit Certificate Request button.
2. There has to be a mechanism to generate the certificate. To do this, the client clicks OK in the Generate Private Key dialog box, enters the Certificate's password in the next dialog box, and then clicks OK.

3. The Certificate Authority has to sign the certificate. The administrator goes to the Certificate Authority database, clicks Client Certificate Requests, opens the client certificate request that is to be signed, clicks the Approve or Deny button (depending on the decision to sign it or not), and enters the password for the CA key ring.
4. Finally, the client needs to pick the certificate up. He/she goes to the server's Web site, selects the Pick Up Client Certificate option, enters the sequence number to identify his/her client certificate, clicks the Pick Up Signed Certificate button, and then clicks the Accept Certificate button.

This is a lot of work and it must be done in a timely manner (even if it is only the client pick-up portion). Furthermore, the certificate is placed in the certificate storage area specific to the browser (it differs between Netscape and Internet Explorer), which means that it then is available for people using the workstation subsequently, so the attempt to determine trust in the user, really only determines trust in the browser. If X.509 *client* certificates are left here and there on workstations with browsers, then this suddenly confers more trust than should be to those workstations and does nothing really to confer trust to the user. Certificates can, and should be, password protected.

For these reasons, it was better to implement X.509 *server* certificates. The down side is that this does nothing to address the authentication question (that is, that HTTP authentication is inherently insecure), other than ensuring that an eavesdropper can no longer intercept the user ID and password.

Aside from encrypting the channel, and thus, shielding it from eavesdroppers, using X.509 server certificates ensure an additional level of trust. No man-in-the-middle attacks can occur, since the browser will indicate whether the server is indeed the one being connected to.

It is important to note that when SSL is used, the TCP/IP port changes, depending on the service being used. The following table shows the ports when the data is unencrypted and when the data is encrypted using SSL:

Service	Port	SSL Port
HTTP	80	443
POP	110	995
IMAP	143	993
SMTP	25	465
LDAP	389	636
NNTP	119	563
IIOP	53148	53149

This is important in the firewall configuration. If the firewall is set to listen on port 80 for HTTP requests and SSL is used, port 443 will have to be opened as well to ensure that requests forwarded to the server are answered.

Clearing the browser cache
Clearing the browser cache is not a security mechanism *per se*, but it should be mentioned. When educating users, you should stress that the browser cache should always be cleared. There is no telling who will follow the employee on a public workstation.

Hardening the server
The main issue here is to ensure that the server is hardened against attacks. While it is not possible to plan for each contingency, a minimum of effort should be made to ensure that all vendor-supplied patches and fixes

are applied to the server. This includes not only the messaging (or Web application) server patches, but also the operating system and driver patches.

Hacked systems are generally hacked because the system administrators didn't apply the necessary patches that close known vulnerabilities. So applying vendor-provided patches will prevent at least some hackers (Script Kiddies) from successfully attacking the server from outside the network.

Encrypting the message store
To provide consistent security outside the network, it is important not to focus all of your attention on the channel, but also to look at the server itself and the data it contains. The reason for this is simple, and we can learn a lot from past mistakes of e-commerce sites who focused solely on securing the communications channel. Attackers noticed this and saw that there was no point in wasting time with that approach, deciding instead to determine where else they could attack. They saw that the database containing all the transaction information (which included names, addresses, items purchased, and even better, credit card numbers and expiration dates) was not adequately protected and attacked that instead. The result was that instead of successfully hacking a single user's session and managing to get one credit card number, they were able to get hundreds with considerably less effort.

Encrypting the message store on a Domino server is fairly straightforward. You can use the local encryption option in the Database properties box to encrypt databases on the server with the server ID. Then Domino administrators at the server can access databases only if they have access to the server ID that was used to encrypt the databases. This means that if for some reason, the server is hacked at the OS level (which should not happen if the server has been hardened) and the intruders get access to the file system and are able to copy the database, they will not be able to read its data because it is encrypted.

What if they could also get the server ID? If the administrator has placed a passphrase (not a pass*word*, but a pass*phrase*, with a mixture of symbols and numbers, such as t34m_m33ting@120'cl0ck%r00m-222D, which would prevent brute-force dictionary attacks) on the server ID, it won't be possible to use it, since the user must provide the passphrase to decrypt the contents, which contains the key to decrypt the encrypted database.

Note that the only issue with this is that applying a passphrase to a Domino server ID means that you cannot have the Domino server restart by itself. The console will require the administrator to enter the ID's password, thus affecting possible 7x24 availability. You should think this through. One possibility is to use SNMP (Simple Network Management Protocol) to trap a server restart event and to dispatch a message to the administrator to enter the password in the console screen.

How it works
Assume that Alice is on a business trip and that she can get to the business center of the airport lounge. With proper security settings in place, she can access the system in the following manner:

1. She logs on to the workstation and connects to the Domino Web application server.
2. The server acknowledges the request and negotiates the SSL handshake, namely negotiating the SSL session and session key.
3. Because a server certificate is used, the server then prompts Alice for her credentials (using session-based authentication) by producing a special log-on form.
4. Data is exchanged with the server using the encrypted link.
5. Once the session is complete, Alice logs off, the session is closed, and the session key is destroyed.

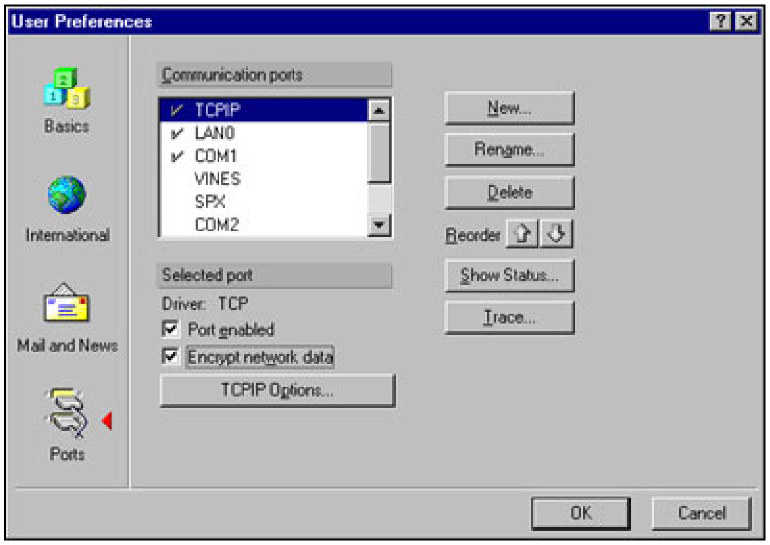
6. Finally, Alice clears the cache of the browser.

Security mechanisms for architecture 2

The next step is to look at security when the e-mail goes from outside the network to inside, so that Bob may receive a message and if need be, reply to Alice.

Encrypting the communications channel

In this case, it would be good to also encrypt the link between Bob's machine and his Domino Messaging Server. Since Bob is using the Notes Remote Procedure Call (NRPC, typically over TCP/IP), you just need to select that the port be encrypted. This is done in the User Preference dialog box under the Ports section.



Communication ports: ✓ TCPIP, ✓ LAN0, ✓ COM1, VINES, SPX, COM2

Selected port
Driver: TCP
[✓] Port enabled
[✓] Encrypt network data

You should keep three things in mind. First, network encryption—such as the port encryption—occurs at the network transfer layer of a selected protocol and is independent of other forms of encryption. Network data is encrypted only while it is in transit. Once the data has been received and stored, network encryption is no longer in effect.

Second, unlike any service over SSL, the port stays the same (which is 1352 and is a registered port); there is no separate port for encrypted NRPC traffic.

Third, while this appears to be a user-defined setting (which means users might disable port encryption and potentially open a security hole on the client), it is important to note that network data encryption occurs if you enable it on either side of a network connection. So, if you enable encryption on a TCP/IP port on a server, you don't need to enable encryption on the TCP/IP ports on the workstations that connect to the server. This eliminates the need for users to set the port encryption setting themselves.

Hardening the server

Again, the main issue is to ensure that the server is hardened against attacks. As mentioned, there is potential for attacks from inside the network. A minimum of effort should be made to ensure that all vendor supplied patches and fixes are applied to the server. This includes not only the messaging (or Web application) server patches, but also the operating system and driver patches.

Encrypting the message store

As with the external server in architecture 1, you should look at the internal

Domino messaging server itself and the data that it contains. Depending on its sensitivity, the data on the messaging server should be encrypted. As with the external server, this is a straightforward process, using the local encryption option in the Database properties box to encrypt databases on the server with the server ID.

How it works
Assuming that Alice has sent an e-mail to Bob, this e-mail is routed to the Domino messaging server, where Bob's mail database is located. Bob can then access this e-mail in the following manner:

1. He loads his Notes client and attempts to access his mail database.
2. This requires that Bob authenticate in order to access the server. The Notes client displays the password entry dialog box.
3. Bob enters his password, which decrypts the public/private key pair.
4. Bob is authenticated and validated using his public/private key pair.
5. Data is exchanged with the server using the encrypted link.
6. Once the session is complete, Bob logs off and the session is closed.

Security mechanisms for architecture 3

Because the customer has more than one flavor of messaging server running on their internal network, the final step is to look at security when the e-mail goes from a user using a Domino messaging server to a user using a different kind of messaging server.

In this discussion, in an effort not to be repetitious, we only look at sending encrypted mail between Carole and Dennis. Again, for reasons of brevity, we assume that the security mechanisms outlined in the previous sections have been implemented and that the department—which is using a different kind of messaging server—has applied all pertinent security mechanisms to secure their portion of the data.

Sending encrypted mail
Earlier we focused the concepts of X.509 certificates on the HTTP protocol for Web browsers, so as to encrypt the channel between the Web browser and the Web application server. Using the same technique, you can use X.509 certificates to perform secure electronic mail communications.

Internet protocols commonly used for mail
First, consider the various Internet protocols that are typically used for sending and receiving Internet mail: SMTP, MIME, POP3, and IMAP4.

SMTP (Simple Mail Transport Protocol) is a protocol for sending e-mail messages between hosts, although with the use of Domain Names Service (DNS) and Mail eXchange (MX) records, it can be thought of as sending e-mail messages to users between domains. Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another. In addition, SMTP is generally used to send messages from a mail client to a mail server. Any host that supports SMTP can also act as an SMTP relay, which can forward messages to another SMTP host.

MIME (Multi-purpose Internet Mail Extensions) is a specification for formatting non-ASCII messages so that they can be sent over the Internet. One of the problems with the original SMTP specification was that it assumed e-mail messages consisted primarily of text; the format specifies the use of plain ASCII text. MIME extends the specification by allowing binary data to be repackaged in text form and transmitted over the Internet in mail messages that are compliant with the original specification. Typically, an e-mail message that supports MIME has extra header messages after the Subject field. Most e-mail clients now support MIME, which enables them to send and receive any kind of binary data such as graphics, audio, and video files via the Internet mail system. In addition, MIME supports messages in character sets other than ASCII.

POP (Post Office Protocol, Version 3, or POP3) and IMAP (Internet Message Access Protocol, Version 4, or IMAP4) are protocols that specify protocols for accessing mail from an Internet mail box or post office.

POP3 is used to pick up e-mail across a network. Not all computer systems that use e-mail are connected to the Internet 24 hours a day, 7 days a week. Some users dial into a service provider on an as-needed basis, and others may be connected to a LAN with a permanent connection but may not always be powered on. In cases such as these, the e-mail addressed to the users on these systems is sent to a central e-mail post office system where it is held for the user until pickup. POP3 allows a user to log onto an e-mail post office system across the network, validates the user by ID and password, allows mail to be downloaded, and optionally allows the user to delete the mail from the server.

IMAP (Internet Message Access Protocol) is a newer protocol that allows for e-mail clients to retrieve e-mail messages from, and work with, the mailboxes on a mail server. The latest version, IMAP4, is similar to POP3 but offers additional and more complex features. With IMAP, for example, you can work with your e-mail on the server, sorting and managing your e-mail on server-side folders.

Security issues of these protocols

The simplicity of these protocols means that they pose security issues for anyone sending and receiving mail as detailed in architecture 3, that is, between different types of messaging servers.

For SMTP, the protocol does not use any authentication process when establishing communications with another SMTP host for relaying and delivering mail. The sending host basically sends a command to the receiving SMTP host saying who it is and that it wants to communicate. The receiving host believes who it says it is, and readily awaits further commands. The sending host then sends another command saying who the mail is from, which the receiving SMTP host accepts. The sending host then sends another command saying who the intended recipient of this mail is, and once again the receiving SMTP host accepts. The sending host then sends a command, stating that what follows is the text message, with, finally, an end of message string at the completion of the message.

It's easy to see that in this scenario, anybody with a network sniffer could pick up this traffic over the network, since it is all sent in clear text. Even worse, it would be quite simple for anybody to spoof a message on any SMTP server. It would be easy to initiate the communication with an SMTP host and pretend that the mail was sent by someone else.

For POP3 and IMAP4, the original POP3 specification does not contain any authentication methods; similar to SMTP, the communication between a POP3 client and a POP3 server is sent in clear text. In fact, the commands USER and PASS are used for passing the user name and password for authorization to connect to a POP3 server for receiving mail.

SSL can be used (as with HTTP) to encrypt the session when communicating using POP3 or IMAP4. This resolves the problem of the weak authentication schemes that are used by POP3 and IMAP4.

For message encryption, SMTP with support for the SMTP extensions can ensure that the initial client-to-server communication has been correctly authenticated. But that does not guarantee that the full SMTP hops will use that same authentication. Also, the message itself is not encrypted. However, you can solve this by using another SMTP extension that ensures the SMTP communications (client/server or server/server) are encrypted using public/private key pairs. Again, this does not guarantee that during its complete series of SMTP hops the message will be encrypted all the way to the recipient. Even if you could guarantee that your e-mail message was correctly authenticated with trusted SMTP servers and fully encrypted during its hops to the client, there is always the possibility that the message was spoofed from someone else.

Thus, the only sure way to provide confidentiality, authentication, and integrity of your e-mail messages is to make sure that the MIME content of your messages is encrypted. Until recently, there were two competing standards for achieving this: PGP and S/MIME. We'll quickly look at PGP before we delve into S/MIME.

PGP (Pretty Good Privacy) is a highly secure public key cryptographic system designed for sending secure mail anywhere around the world. It was developed by Mike Zimmerman, and is available for free on the Internet. PGP does not have key management capabilities; in fact its certificate structure is a very loose one. Instead of having authorities issue certificates to individuals, it works on a “web of trust” model, where certificates gain authority by being signed by people you know. A newer standard called OpenPGP, will permit a hierarchical approach to accommodate Certificate Authorities (CAs), X.509 certificates, and other accepted standards. It is unclear how much broad support OpenPGP will gain since it uses the Diffie-Hellman algorithm. This will immediately make it incompatible with the 20 millions users currently using PGP, which employs RSA encryption algorithms.

S/MIME (Secure Multi-purpose Internet Mail Extension) is an e-mail security technology developed by RSA for encrypting and digitally signing e-mail messages. Notes and Domino R5 support S/MIME. It is an IETF (Internet Engineering Task Force)-proposed standard that builds security on top of the industry-standard MIME protocol and a set of Public-Key Cryptographic Standards (PKCS). A message is encrypted by taking the entire content of a message or just certain MIME parts and running it through an encryption algorithm that uses the public key of the recipient.

S/MIME v2 is something of a *de facto* standard across the Internet for sending secure mail, although S/MIME v3 was ratified as a standard in June, 1999, by a working group for publication and final ownership of the IETF. Although S/MIME v2 is widely implemented by vendors, it is not an IETF standard, and quite likely will not become one. It requires the use of RSA key exchange, which is a US-based patent, and uses weak cryptography (RC2/40). S/MIME v3 resolves these problems by using stronger cryptography and a choice of encryption algorithms. (Notes R5 is implemented using S/MIME v2.) S/MIME uses a public-key algorithm for key exchange and for digital signatures. S/MIME recommends three symmetric encryption algorithms: DES, Triple-DES, and RC2. The adjustable key size of the RC2 algorithm makes it especially useful for applications intended for export outside the US where RSA is the required public-key algorithm.

Details of S/MIME in practice

Let's look in at how S/MIME works in detail to understand how Carole can send a message securely to Dennis.

S/MIME offers users the following basic features:

- Encryption for message privacy
- Tamper detection
- Signing, or the authentication of sender with digital signatures
- Interoperability with other S/MIME-compliant software
- Cross-platform messaging

With these features, you can be sure that:

- From the moment the message is sent by Carole to the moment that it arrives for Dennis, no one can see the contents of her message.
- The message came from Carole, which is who Dennis thinks it came

from.
- The message has not been tampered with or changed on route to delivery.

For message privacy, or confidentiality, S/MIME uses asymmetric keys (public/private keys) to encrypt messages, the same technique that has been employed in Notes for years. To send an encrypted message, Carole needs to obtain Dennis' public key, and encrypt the message using this key. Since the only person who has its associated private key is Dennis, the message can be sent with safe knowledge that only Dennis will be able to decrypt this message.

Note: It is due to the difficulty in ensuring that Carole has the recipient's public key in her Directory that we have consigned the conversation to sending secure e-mail to recipients *inside* the company's network, since the company has a Directory Service in place (whether based on X.500 or simply LDAP) that makes available X.509 certificates from all employees. Including people outside the company network would require quite a lot more work and bilateral agreements in regards to exchanging Directory entries and which attributes in the entries are exchanged.

S/MIME uses a technique often referred to as a digital envelope, whereby the message is actually encrypted using the shorter symmetric cipher. The symmetric cipher is then encrypted using the larger asymmetric key and sent along with the encrypted message. This approach is taken because it is far quicker to encrypt the whole message using the shorter symmetric key, than to encrypt the message using the longer asymmetric key. The message is still quite safe; this approach gives you the speed of symmetric encryption with the security of asymmetric encryption.

For tamper detection, or data integrity, S/MIME also can ensure that the message has not been tampered with. Again, it uses a technique that's already employed in Notes.

S/MIME provides message signing by using digital signatures. A Message Digest—which is a unique value computed for the message, whereby if even one character changes, the digest obtained will change significantly—is made. The Message Digest is encrypted with Carole's private key and sent with a certificate, which vouches for the authenticity of Carole's public key. Dennis, upon receipt, can decrypt the Message Digest with Carole's public key, which is freely available. (Remember not to confuse this with message encryption, where the message is encrypted with Dennis' public key.) In many cases, Carole may only want to sign the message, but it is also possible that she might want the message encrypted and signed, in which case the message goes through encryption with Dennis' public key and then the Message Digest is encrypted with Carole's private key. The S/MIME specification does not specify the order in which the encryption must occur when both encrypting and signing a message.

Let's look in closer detail at how Carole actually verifies that the message received is from whom it claims to be from.

As previously stated, the message is encrypted with Carole's private key. Carole also sends its X.509 certificate with the signed message, so now the certificate is nothing more than the signed public key of Carole. It is signed by another trusted party, a Certificate Authority (CA). What happens if you don't trust the CA that signed the sender's public key? S/MIME allows for that by employing what is known as a chain of trust. This means that when Carole sends the encrypted message and Carole's own certificate (which contains its public key signed by a third-party CA), it also sends the third-party CA's certificate. This may itself be signed by another CA or it may indeed be the root certificate. As long Dennis can trust any of the CA certificates in that hierarchy, Dennis can trust the CA that signed the sender’s public key.

So how do you trust that CA in the first place? Well, the S/MIME client being used will hold a list of CAs and their public keys that it trusts; this is pre-built in the client to ease distribution of CA certificates. The situation at this point is that we have the signed message and Carole’s certificate, and Dennis trusts the CA that has signed Carole’s certificate because Dennis has its public key in his S/MIME client.

It is now possible to testify the validity of Carole as the sender, since the certificate that has been sent can be decrypted with the public key of the CA that is held in her S/MIME client. If this is successful, then it is possible to vouch for the authenticity of the certificate and its contents, Carole’s name, her public key, organization, country, and e-mail address. Now that it is possible to trust Carole’s public key, it is now possible to attempt to decrypt the message, to see if the message was signed by that same person. Also, on Carole’s certificate there is another piece of information—her e-mail address. This information is crucial in ensuring that e-mails are not spoofed, even if the message can be correctly decrypted with Carole’s public key. If its associated certificate does not have a matching e-mail address then this would suggest that the message was sent from a different user. If this is the case, how trustworthy can the message or the sender really be?

As it happens, this may cause problems in the future as people acquire multiple e-mail addresses. They may have a work address, a personal e-mail address, and perhaps a second work address if they are working temporarily at an alternate location. Does this mean that there is a need to maintain three sets of public/private key pairs and certificates?

This problem is further compounded by the fact that it is not easy to export S/MIME certificates from one client to another. S/MIME v3 hopefully will solve some of the problems regarding having multiple e-mail addresses associated with a common name. With regards to interoperability between clients from different vendors, this can be resolved only if customers always demand the highest level of interoperability.

If Carole attempts to send a signed message to a recipient that does not have an S/MIME client, there are two possible outcomes depending on the capabilities of the sending S/MIME client.

- If the message is sent as opaque, it means that the signature is sent as an application/pkcs7-signature MIME type. Thus, a non-S/MIME-compliant client will not be able to read the pkcs7-signature type. The S/MIME client will first split the incoming message, and then check the validity of the signature.
- If the message is sent as clear, it means that the signature is inserted as part of a multipart/signed MIME object type. The signature is generated from the message by hashing it, and the application/pkcs7-signature is inserted into the second part of the MIME type. This means that any receiving client, such as Dennis, will be able to receive both parts of the MIME type—the unsigned message and an attachment of the application/pkcs7-signature MIME type.

Interoperability with other S/MIME-Compliant Software is achieved using PKCS#12. The PKCS #12 standard specifies the format for certificate export and import. This is particularly important for ensuring that a backup copy of the private key can be made. Also if you need to send S/MIME e-mail from a different machine or from a different S/MIME client, you can simply take along the public/private key pair and have it installed in the new client. The purpose of the PKCS #12 standard is to provide interoperability of the private/public key pair and certificates with other S/MIME clients. This is important because if a certificate is requested with Internet Explorer from a Web CA like VeriSign, it can only be retrieved into Outlook Express. Similarly, if a certificate is requested with Netscape Navigator it can only be retrieved in Netscape Messenger.

To be able to send signed mail using S/MIME, an X.509 certificate will be required for the client software used. Current S/MIME-compliant clients, like Netscape Messenger, Lotus Notes, and Microsoft Outlook Express, provide the ability to generate a certificate request with a Web-based CA. Once the client certificate has been requested, it is installed in the S/MIME client so that it can be possible to sign any e-mail messages. It also is necessary to make that certificate available to anybody who wants to send encrypted mail to the associated e-mail address. As an illustration, encrypted mail messages addressed to Carole are encrypted with Carole's private key.

Here are the steps needed to request a client certificate to be installed in an S/MIME client:

1. From within the S/MIME client (Lotus Notes, Netscape Messenger, or Microsoft Outlook Express), a client certificate is requested. The Notes browser, Netscape Navigator, or Internet Explorer 4 prompts the user to fill in a client certificate request form at the Web site of a trusted CA.
2. When the request is submitted, it triggers the browser to generate and store a private key locally. (This process differs if the client is Internet Explorer.)
3. A corresponding public key is included in the HTTP header as part of the certificate request (in PKCS #10 format) to the Web CA.
4. The CA processes the request and returns instructions on how to pick up the certificate via e-mail. The instructions state a URL and a pickup ID (PIN) where the signed client certificate can be picked up.
5. The user connects to the stated URL, enters the PIN, and picks up the signed certificate.
6. It is then installed in the S/MIME client.
7. The certificate should then be published by sending it to one of the public directory providers. Often the CA’s themselves will have this facility available.

Finally, in Netscape Messenger (4.x) and Microsoft Outlook Express there are a couple of mechanisms for obtaining a recipient’s certificate.

The first is by having the desired recipient send a signed message to the sender. When the sender receives this message, these e-mail clients will automatically add the sender’s certificate to the list of stored certificates. Similarly, if signed mail is sent to another Netscape Messenger or Outlook Express user, they will obtain a copy of the sender's certificate.

The second method is by providing access via LDAP to search online directories such as Four11, Bigfoot, Switchboard, and so on. If the required certificate is stored in one of these directories, you can add it to the personal address list.

Sending and receiving encrypted S/MIME messages

We've discussed the technology basics for sending and receiving secure e-mails; let's look concretely at the *process* itself from Carole's perspective, using the Lotus Notes client.

When Carole attempts to send an encrypted message, Dennis' X.509 certificate is used. That is, Carole must have access to that certificate to send an encrypted message. Dennis' certificate must be registered in Carole's Personal Address Book or Domino Directory. To send an encrypted message, Carole clicks Delivery Options and selects Encrypt. Or, if Carole wants to encrypt all the mail messages she sends, she can choose File - Preferences - User Preferences, click Mail and News, and select Encrypt Sent Mail.

Carole can sign individual mail messages or sign all mail messages that she sends, including encrypted messages. Before signing a message, she must make sure she has obtained her own X.509 certificate in her Notes ID file. To sign an individual mail message, when she finishes composing the mail message, she clicks Delivery Options and selects Sign. Or, to sign all mail messages she sends, she chooses File - Preferences - User Preferences, clicks Mail and News, and selects Sign Sent Mail. She then sends her message. If the message is addressed to a non-Notes mail address or the message is in MIME format, the message is automatically signed for S/MIME.

Receiving signed S/MIME messages is done in a similar manner. Upon receipt of a signed e-mail, Notes will try to verify the validity of the signature. If Carole "trusts" the signing certificate—that is, if she has the signer's certificate or an Internet Cross-Certificate to the sender—she will receive a message in the status bar indicating the validity of the signature. For example:

Signed By: Dennis, at 10:52 AM, According To: CoCertAuthority

If she doesn't "trust" the signing certificate, she will receive a prompt to create an Internet Cross-Certificate on demand. She can select the subject name of the certificate in the message that she wishes to trust. Note that signed S/MIME messages contain the certificate chain of senders and signers. The resulting Internet Cross-Certificate is stored in Carole's Personal Address Book. By creating the cross certificate, she is asserting that she trusts a certificate contained in the S/MIME signed message. Signature verification can then proceed. Also, she can manually store the sender's address and X.509 certificates in her Personal Address Book. When viewing S/MIME signed mail, she would choose Actions - Tools - Add Sender to the Address Book. Note that this certificate is not an Internet Cross-Certificate, that is, it is not used when sending or receiving signed S/MIME mail. It is used to encrypt messages from her to the sender.

How it works
Assuming that Carole has sent an e-mail to Dennis, this e-mail is routed to the Domino messaging server where Carole's mail database is located and then routed using S/MIME over SMTP to Dennis' server, where he can consult it with the appropriate client for that server. The process for Carole to send an e-mail to Dennis is:

1. Carole loads her Notes client and attempts to access her mail database.
2. This requires that Carole authenticate in order to access the server. The Notes client displays the password entry dialog box.
3. Carole enters her password, which decrypts the public/private key pair.
4. Carole is authenticated and validated using her public/private key pair.
5. Data is exchanged with the server using the encrypted link.
6. To send the e-mail, the Notes client will get Dennis' X.509 certificate from an available Directory.
7. The message is encrypted using Dennis' X.509 public key and sent along in S/MIME format.
8. The e-mail is routed to the Domino server, which routes it to the non-Notes messaging server, still in S/MIME format.
9. The e-mail is received by Dennis and is decrypted using the private key of his X.509 certificate.
10. Once the session is complete, Carole and Dennis log off and the sessions are closed.

Conclusion
By looking at three different architectures, we've explored in detail a variety of security options available for securing Web-based mail that were considered by the company in question. Their ultimate choices reflected their particular situation and needs, just as any security decisions you make will ultimately reflect the nature of your network and users.

ABOUT THE AUTHOR
Frederic Dahm has been working with Lotus Notes for nearly six years and working as a Lotus employee for three-and-a-half years. This means that he has experience working with Notes and Domino as both a customer and as a Lotus employee, giving him an interesting perspective on the use of this technology.

In his tenure as a Lotus employee, he worked as a Systems Engineer in Ottawa, Canada for almost three years and is now a Systems Architect for Lotus Professional Services in Zürich, Switzerland. He is a coauthor of the IBM Redbook "Lotus Notes and Domino R5 Security Infrastructure Revealed, SG24-5341-00" (available at **http://www.lotus.com/redbooks**) and is interested in all facets of security, both the security in Notes and the broader topic of e-security.

When not working on a project or keeping up-to-date on technical matters, he likes to spend time with his family as well as contemplate the many truths of life present in his favorite form of literature: his collection of Superman comic books.

Script Kiddies are novices that have little skill, few resources, little expertise and little knowledge of the system they are attacking. They typically use tools written by White Hat Hackers and think that they know a whole lot more than they really do. Both Black and White Hat Hackers sneer in contempt at Script Kiddies. However, they do exist and they can wreak havoc on unprotected systems.

White Hat Hackers are very technical people that try to access systems for the sheer thrill of it. They are often referred to as Old School or Noble Hackers (and hence the original definition above). Their goal is typically twofold: the first is to see if they can increase their technical knowledge by hacking into systems and seeing what makes them tick; the other is to verify that if a vendor claims that their product is secure, that it is indeed so. They will identify potential vulnerabilities and alert the vendors and after time is given for the vendor to react, publish the vulnerability to ensure that *everyone* knows about it (otherwise, only a few people would know, which is not a good thing). So why worry about White Hat Hackers? The problem is that in publishing the vulnerability, they sometimes publish an attack tool that greatly simplifies the use of the vulnerability. This tool is usually what is used by Script Kiddies in their attacks.

Black Hat Hackers are very technical people that try to access systems for malevolent purposes. These people seek to enter systems usually with the intention of embarrassing companies by destroying, defacing, or corrupting their systems. In addition to their high skills, they tend to have sophisticated resources and tools at their disposal and have intricate knowledge of the architecture and setup of the systems they are attacking.

Hacktivists are people generally with fewer skills and resources than Black Hat Hackers, but whose motive is to attack systems for political or religious reasons. Their sole goal is to make a statement, and if it requires hacking into a system to access information, deface it, or destroy it, they will do it.

Corporate Spies are people with fair to excellent computer science skills whose single motive is financial. If they are contracted to get access to information pertaining to the attribution of a major government contract, for example, they will hack whatever systems will need to be hacked in order to acquire that information.

Insiders are people that are hired by a company and work there. They can either be employees or contractors and usually have one of two motives for hacking your systems: the first is for financial gain, the other is for revenge. In the first case, their motives are the same as Corporate Spies. In the second case, it could be a way to compensate for some kind of perceived offense that was done to them, such as a demotion, a cut in pay, or an administrative note in their personnel record. Many security problems occur considerably more often with insiders than outsiders.