SMTP via TLS with Evolution, Kmail, and Mutt

Safer Mail

The secure transmission of electronic mail via the Transport Layer Security (TLS) cannot replace individual encryption of mail messages. Unfortunately, although modern mail programs are capable of using the former method, they tend to make life unnecessarily difficult for any users who want to leverage the benefits of confidentiality and security.

BY PATRICIA JUNG

E-Mail messages are like postcards – any "mail person" can read them. Although most admins will studiously avoid reading other people's private mail, at least in theory, the privileged or non-authorized owners of the *root* password (like hackers) of a machine used to forward or store mail will have access. Secret services all over the world are full of praise for the founding fathers (and mothers) of the Internet, who were so trustful as to envisage transmitting all kinds of traffic in the clear across the wire.

Encryption, for example with GnuPG and PGP, is the electronic equivalent of an envelope in the real world. There is an issue here: If a country denies its citizens access to secure encryption methods – typically claiming that this will help in the fight against crime – the mail transport protocol, SMTP, ("*S*imple *M*ail *T*ransfer *P*rotocol"), turns traitor. The content of a PGP or GnuPG encrypted message sniffed off the wire cannot be read, but you do get see which messages have been encrypted, and which have not. Thus the use of strong encryption methods is condemning evidence against all those "criminals" who demand confidentiality for their mail messages.

To avoid a world full of glass houses, more and more mail providers are beginning to exchange mail via encrypted TLS ("*T*ransport *L*ayer *S*ecurity") tunnels rather than in the clear, thus preventing sniffing attacks. There are less idealistic reasons for following the trend away from cleartext messaging, using SSH instead of the venerable telnet protocol, or ***https*** URLs on websites that prompt you for personal data.

Introducing a policy that enforces the use of encrypted mail within a distributed enterprise can cause a few issues: What happens if a third party not involved in the original communication needs access to the messages later? What happens to keys used by a member of staff that leaves the company, or the encrypted correspondence? As encryption is often "only" used to prevent sensitive company data from crossing the wire in the clear, a server side encryption solution that is completely transparent to the users, like SMTP/TLS, is often the answer.

Secret Deals

This solution also offers the advantage of being maintenance free, following the initial server configuration. If the mail client and mail server both "speak" TLS, they will negotiate a secure exchange without external intervention. To enable this, both computers exchange certificates during the initial handshake to provide mutual authentication. This is like each computer showing its passport

THE AUTHOR

Patricia Jung has spent her working life as an editor, technical writer and system administrator on Linux and FreeBSD systems. Having been an editor for Linux Magazine for quite a while now she feels privileged to continue building a carrier based on free Unix systems exclusively.

GLOSSARY

https: *Although you might not expect this, https is not an independent protocol. Communication between browsers and servers on websites that use https URLs is still based on the cleartext "Hypertext Transfer Protocol", HTTP, but it is encapsulated in a TLS tunnel.*

to the other. For practical reasons only the mail recipient will authenticate with the client in a mail environment, and it makes no difference whether the client is another forwarding mail server, or the user's mail program in this case. As the client can now trust the server, the two machines agree on an encryption algorithm and a secret encryption key, which is use to encode any ensuing traffic, making it unintelligible to a third party.

People tend to trust passports because they trust the authority that issues them. Similarly, the mail client will only accept the server's certificate if it is convinced that the issuing authority has done so in all conscience. The authority authenticates the certificate by signing it with its own certificate, and this certificate may in turn have been signed by a higher ranking trust provider.

However, to be able to trust any authority, the client will need to store the certificates of the highest trust authorities, the root certificates.

Know-How

Whereas mail servers ("Mail Transfer Agents") like Postfix or Sendmail will automatically use TLS if both partners are capable of doing so, today's mail client [1] ("Mail User Agent") developers tend to place little emphasis on ease of use. Instead of pre-configuring their programs to use SMTP/TLS whenever possible, they expect users to be so au fait with the subject matter that they will choose and know how to implement these safe settings.

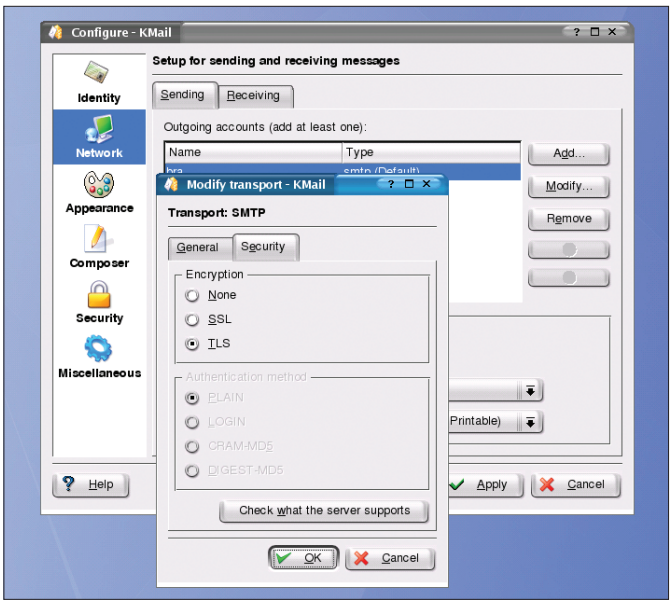

Figure 1: KMail leaves it up to the user to discover whether the mail server is TLS-aware

In the case of *KMail* on KDE 3.x, if you select *Settings / Configure…* and then add a new "Outgoing account" with *SMTP* transport in the *Sending* tab of the *Network* option, or modify an existing account, you additionally need to select the *Security* tab in the dialog box that appears (Figure 1). This will at least provide you with the option of clicking on *Check what the server supports* to ascertain how encryption friendly the smarthost selected in the *General* tab is. If you receive a positive response (the alternative, SSL, ("Secure Socket Layer"), is a predecessor to TLS), you can then click on *OK* to ensure that KMail will use encrypted communications.

Version 1.0.x of *Evolution* does provide a secure SSL option for the smarthost. Unfortunately, simply checking *Use secure connection (SSL)* turns out to be a pitfall in many cases. Most servers do not support the obsolete SMTP via SSL method; on the other hand Evolution 1.0.x does not speak TLS. If you try to transmit mail with this configuration, the mail program will refuse to cooperate and issue a highly informative message such as *Connection to name.of.mailservers (Port 465) could not be established. The connection was reset by the communication partner*. The Client will not even attempt non-encrypted communication until your remove the stubbon checkmark in your *Mail Preferences*.

Version 1.2 resolves both issues, although the *Use secure connection (SSL)* in the wizard or configuration dialog box (see Figure 2) again only refers to SSL, without mentioning TLS, which is also supported. Heaven only knows why the value *Whenever possible* is not the default here.

When you first attempt to send encrypted mail, Evolution will display information on the Server certificate and ask you to accept or to do without the SSL/TLS tunnel (see Figure 3). As the program does not help you to reach this decision (i.e. what it means if a signature is *BAD*), it might be better for the program to make this decision itself, rather than worry the user by prompting her.

The command line program *mutt* is almost exemplary in contrast: As it always uses the local mail Server to transfer mail via the */usr/sbin/sendmail* interface, it does not need to support SMTP/TLS itself. The message will automatically take a safe path across the network, if the local Mail Transfer Agent and the target server can provide one; this does not require any interaction from the user. But you cannot avoid providing your local mail server with these capabilities on your own computer… ■

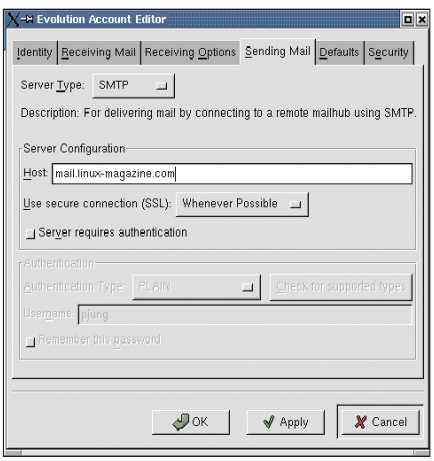

Figure 2: Why is this not the default?

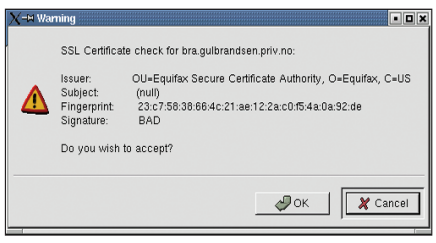

Figure 3: Do you trust this certificate?

INFO

[1] Patricia Jung and Andrea Müller: "Mail and more", Linux Magazine Issue 29, April 2003, page 44, *http://www.linux-magazine.com/issue/29/MailUserAgents.pdf*