Securing Email Through Proxies: Smap and Stunnel

Jim Cabral

Securing Email Through Proxies: Smap and Stunnel

Prepared by Jim Cabral as part of SANS GSEC certification, on 9/7/2001.

Introduction

Electronic mail was the first "killer application" on the Internet and continues to be one of the most commonly used Internet applications. However, Internet email and sendmail, the venerable application behind most Internet email, have a long and storied history with regard to security. As email has grown in usage and complexity, so have sendmail and other mail applications including POP and IMAP servers, Microsoft Exchange and mail clients such as Eudora, Microsoft Outlook and Netscape Messenger. As with other complex applications, new versions of these mail applications have introduced new security issues and vulnerabilities.

This paper describes an approach to securing complex mail application using a common firewall technology, application proxies. The advantage of this approach is that it does not require replacing the mail applications. Application proxies simply provide a simple, secure front-end to the insecure back-end applications.

Application Proxies

Firewalls are hardware and/or software designed to provide a layer of protection between trusted and untrusted systems. In order of increasing security and decreasing throughput, the three basic approaches to firewall design include packet filters, stateful inspection filters and application proxies. Packet filters forward or drop network traffic simply based on the information in the data-link, network and transport layer headers. Stateful inspection filters extend packet filters by adding some application protocol-specific checking before deciding whether to forward or drop network traffic. Since packet filters and stateful inspection filters only forward or drop traffic, these filters are typically implemented in the operating system kernel.

Application proxies, on the other hand, run as applications outside the operating system kernel and do not simply forward or drop network traffic. Instead, proxies act as intermediaries between the client and server and accept the traffic on one port and analyze it at the application layer before reconstructing it and sending it out another port.

There has been quite a bit of confusion among vendors of commercial firewalls regarding the meaning of "application proxy". In order to provide some clarity regarding the functions and security that a particular proxy provides, five ratings for application proxies have been suggested. The proxy ratings, in order of increasing security, included circuit-level, traffic-aware, command-aware, content type-aware and policy-aware proxies.

Circuit-level proxies are similar to packet filters in that they accept or deny traffic based on information in the data-link, network and transport layer headers. On IP networks, this includes source and destination addresses, TCP/UDP source and destination ports and TCP SYN (synchronization) and ACK (acknowledgement) flags. TCP wrappers are a common implementation of circuit-level proxies.

Traffic-aware proxies are very similar to circuit-level proxies but include some basic processing of the data stream. This could include some simple buffer management or transport-layer prioritization of traffic based on TCP URG (urgent) flags.

Command-aware proxies segment data streams into separate commands and responses and check each command and response for correct syntax before passing it in or out of the protected network. This level of visibility into the application protocol provides significantly greater logging and authorization capabilities than circuit-level or traffic-aware proxies. For instance, a command-aware proxy for FTP could distinguish between FTP GETs and FTP PUTs and could even log the filenames and sizes of the files being transferred through the proxy. In addition, command-aware proxies usually enable administrators to limit the commands permissible through the proxy. For instance, the command-aware FTP proxy could be configured to specifically allow FTP GETs (reads) but block FTP PUTs (writes).

Content type-aware proxies extend the capabilities of command-aware proxies slightly by including checks for the content format. If the protocol indicates that a command should include an object of a certain type, a content type-aware proxy could check to verify that the object conforms to that format. These proxies can also block certain types of content completely. Content type-aware proxies are useful for protecting applications with known vulnerabilities in handling certain objects. For instance, content-type aware proxies could block all HTTP transfers of Word and Excel documents that include macros.

Policy-aware proxies enforce local security policies regarding allowable content. For example, a proxy that could detect and block a sexually explicit image based on the local sexual harassment policy could be considered a policy-aware proxy. However the computational complexity of the detection algorithms currently makes this type of proxy mostly theoretical.

Securing inbound mail

Internet email is transferred using software servers known as mail transfer agents (MTAs). Sendmail, the most common MTA, is the evolution of delivermail, a pre-TCP/IP MTA developed by Eric Allman at UC Berkeley in 1979. Sendmail extended delivermail with support for mail delivery on the ARPANET using SMTP and DNS and was included in BSD 4.1c, the first UNIX distribution to support TCP/IP. Over the last 20 years, sendmail has continued to evolve as new mail standards emerged. This long history and the requirements for backward compatibility among mail servers have resulted in sendmail becoming an extremely complex program with old security flaws fixed and new security flaws introduced with each new release. As a result, a great number of UNIX mail servers have been compromised through flaws in sendmail.

Sendmail is no longer the only TCP/IP MTA. Sendmail alternatives such as Qmail, Zmailer, Postfix, Smail and Exim attempt to provide better performance and security then sendmail by starting from a fresh code base and abandoning some of the support for legacy protocols and mail formats. However, compatibility issues are common with these mailers and some of these alternatives have had their own security issues. Non-UNIX MTAs such as Microsoft Exchange and Lotus Notes/Domino have also had their share of security and compatibility issues.

Smap

In October 1993, prior to the introduction of most commercial firewalls, Trusted Information Systems (TIS) released their Firewall Toolkit (FWTK) source code and made it freely available. Since then, thousands of firewalls have been developed using the firewall toolkit and TIS (now part of Network Associates) produced and sold the highly successful Gauntlet firewall based originally on the firewall toolkit. The current version of the firewall toolkit includes support for most UNIX variants including Solaris, HP/UX, Ultrix, BSDI, BSD, AIX, SCO, OSF and Linux.

The firewall toolkit includes a command-aware application proxy called smap. Smap is a simple (~ 1000 lines in C) SMTP server that receives an incoming email, performs some basic integrity checks and saves the email to a mail queue. Another daemon, smapd periodically checks the mail queue and passes any new emails to an MTA such as sendmail for delivery. The result is an extremely simple front-end that completely isolates the complex back-end MTA from direct exploits over the Internet.

Smap and smapd were written specifically for firewalls and therefore prioritize security over functionality. Unlike sendmail, which usually requires root privileges, smap and smapd can run as unprivileged users. Smap also runs in a chrooted environment so that, even if an attacker were somehow able to compromise the proxy, they would only have access to the mail queue.

Smap and smapd form a very good foundation for protecting a mail server. However, the simplicity of smap results in an open relay mail server that spammers can exploit. As spam developed into a major Internet nuisance, patches for smap were developed to address these issues. The latest patches for smap add support for the following anti-spam features:

1. A list of IP addresses or domains for which mail relaying is allowed,
2. Simple SMTP authentication (SMTP AUTH) before accepting mail for non-local domains, and
3. Spam filtering using the Mail Abuse Prevention System (MAPS) lists.

There are also patches available for smap that encrypt SMTP traffic using SSL and which virtually prevent the sniffing of SMTP traffic.

Smap example

Here is an example smap configuration on a Red Hat Linux 7.1 system using sendmail as the MTA.

After downloading the FWTK and patches, applying the patches, compiling and installing smap, the smap and smapd executables are installed in /usr/local/etc. To start smap automatically, the following file should be created in /etc/xinetd.d/smtp and xinetd should be restarted.

```
service smtp
{
        flags           = REUSE
        socket_type     = stream
        protocol        = tcp
```

```
        wait            = no
        user            = root
        server          = /usr/local/etc/smap
}
```

Smap and smapd are configured through the FWTK file netperm-table which is usually installed in /usr/local/etc/. The following netperm-table options are a good template for most smap configurations but will require some customization:

```
# Run as unprivileged user
smap,smapd: userid  99
smap,smapd: groupid 99

# Location of smap mail queue
smap,smapd: directory /var/spool/smap

# Location of smapd executable
smapd:      executable /usr/local/etc/smapd

# Location of MTA
smapd:      sendmail /usr/sbin/sendmail

# Location of SSL certificate
smap:       ssl /var/ssl/certs/mail.pem

# Do not require SSL authentication
smap:       ssl-auth optional

# Require SMTP AUTH for mail relaying
smap:       auth login
smap:       md5file /etc/smtppass

# Domains for which we relay mail
smap:       relay-domain localdomain.com

# Spam Filtering
smap:       nospam RFC821
smap:       nospam ns-required
smap:       nospam maps-rbl
smap:       spam-database /etc/spamdb

# Directory for mail rejected by the MTA
smapd:      baddir /var/spool/smap/bad

# Timeout while accepting an email
smap:       timeout 3600
```

```
# Timeout before checking mail queue
smapd:      wakeup 15

# Maximum email recipients and size
smap:       maxrecip 100
smap:       maxbytes 4000000
```

Securing remote access to mailboxes

Just as SMTP is the standard for the exchange of mail between mail servers, Post Office Protocol version 3 (POP3) and Internet Message Access Protocol version 4 (IMAP4) are the most common standards for proving users with remote access to their mailboxes. Users use POP/IMAP client applications to login to POP/IMAP daemons on their mail server and retrieve their mail. From a security point of view, the POP and IMAP protocols are somewhat safer than SMTP in that they require users to authenticate. However, POP and IMAP do not by themselves support strong encryption and typically transmit usernames, passwords and emails in plaintext. This makes POP and IMAP connections extremely vulnerable to sniffing attacks.

A common solution to securing remote access to mailboxes is by tunneling POP/IMAP connections over Secure Sockets Layer (SSL) or Transport Layer Security (TLS). However, although most mail clients (including Outlook, Outlook Express, Netscape Messenger and Eudora) support SSL/TLS tunneling, many POP and IMAP servers do not support SSL or TLS by default.

Stunnel

Stunnel is a circuit-level proxy server that tunnels almost any TCP-based protocols over SSL, including POP and IMAP. It requires the OpenSSL open source SSL/TLS toolkit and supports most UNIX and Windows platforms.

Stunnel can encrypt secure POP and IMAP connections on the standard TCP ports for POPS (995) and IMAPS (993) and pass traffic to mail servers running on the standard TCP ports for POP (110) and IMAP (143). Stunnel itself is very simple and secure, especially when it is running as an unprivileged user in a chroot environment.

Stunnel includes support for TCP wrappers that can limit which IP addresses are allowed to connect to the mail servers. This capability is useful when users connect from static IP addresses but is not practical for mobile users. Another option for controlling access that is more appropriate for mobile users is authentication through client certificates. Client certificates are SSL certificates that are installed in the mail clients. When the client attempts to connect to the POP or IMAP server, the Stunnel proxy requests authentication, the client provides the client certificate, and the Stunnel proxy attempts to validate the certificate against a database of allowed certificates. If the certificate is validated, the connection is allowed through to the POP or IMAP server.

Stunnel example

Here is an example stunnel configuration on a Red Hat Linux 7.1 system using standard POP and IMAP servers.

Stunnel is included with Red Hat Linux 7.0 and later distributions. To create an SSL server certificate, execute the following as root:

```
# cd /usr/share/ssl/certs
# make stunnel.pem
```

The simplest stunnel configuration uses xinetd and TCP wrappers. Configure TCP wrappers to block POP and IMAP traffic by default by adding the following line to /etc/hosts.deny:

```
pop3,imap: ALL
```

To allow certain clients to access the POP and IMAP servers, add the range of allowed IP addresses to the /etc/hosts.allow file as follows:

```
pop3,imap: 10.0.0.0/255.0.0.0
```

To enable secure POP, create the following file in /etc/xinetd.d/pop3s and restart xinetd:

```
service pop3s
{
     socket_type    = stream
     wait           = no
     user           = root
     server         = /usr/sbin/stunnel
     server_args    = -r imap -s nobody -g nobody \
          -p /usr/share/ssl/certs/stunnel.pem
     log_on_success += USERID
     log_on_failure += USERID
     disable        = no
}
```

To enable secure IMAP, create the following file in /etc/xinetd.d/imaps and restart xinetd:

```
service imaps
{
     socket_type    = stream
     wait           = no
     user           = root
     server         = /usr/sbin/stunnel
     server_args    = -r imap -s nobody -g nobody \
          -p /usr/share/ssl/certs/stunnel.pem
     log_on_success += DURATION USERID
     log_on_failure += USERID
     disable        = no
}
```

To run in a chroot environment, stunnel should run as a daemon, rather than through xinetd. A reference to Dave Lugo's instructions for a chroot configuration has been provided below.

Conclusion

Although new security issues with sendmail and other mail servers are constantly being discovered, application proxies provide simple barriers that protect mail servers from most of these vulnerabilities. Proxies can also add much-needed features such as encryption and strong authentication without requiring any changes to the mail servers themselves.

References

Zwicky, Elizabeth D., Cooper, Simon, Chapman, D. Brent, "Building Internet Firewalls," O'Reilly & Associates, Inc., 2000.

"Firetower Inc Forum – Application Proxy," URL: http://www.firetower.com/forum/applicationproxy.html, May 2001.

Richardson, Michael, "Rating of Application Level Proxies," URL: http://www.sandelman.ottawa.on.ca/SSW/proxyrating/proxyrating.html, November 1996.

Gates, Dan, "TCP Wrapper: A Tool to Help Protect Your Data," URL: http://www.sans.org/infosecFAQ/unix/TCP.htm, December 2000.

"Sendmail FAQ", URL: http://www.sendmail.org/faq/, August 2001.

Ranum, Marcus, "Installing the Trusted Information Systems Internet Firewall Toolkit," URL: http://www.fwtk.org/fwtk/docs/mjr-slides/sld068.htm, 1996.

"TIS FWTK," URL: http://www.fwtk.org/fwtk, July 2000.

"FWTK Patches," URL: http://www.fwtk.org/fwtk/patches/patches.html, January 2001.

"MAPS Realtime Blackhole List," URL: http://www.mail-abuse.org/rbl/, July 2001.

"Stunnel: Universal SSL Wrapper," URL: http://www.stunnel.org/, August 2001.

"OpenSSL: The Open Source toolkit for SSL/TLS," URL: http://www.openssl.org/, July 2001.

"The IMAP Connection," URL: http://www.imap.org/, August 2001.

Lugo, Dave, "Running Stunnel in a Chroot Environment", URL: http://www.etherboy.com/stunnel/stunnelchroot.html, December 2000.