Establishing and Verifying the Stunnel SSL Encryption of Pine IMAP Email Sessions

Christopher Ursich

SANS Security Essentials GSEC Practical Assignment
Assignment version 1.2f

ABSTRACT: This paper documents one method for establishing and verifying the operation of SSL encryption using Stunnel for Pine IMAP email sessions. Several technologies are introduced, and briefly explained, including IMAP and SSL. Detailed information is given regarding the establishment of SSL functionality using Stunnel. After completing the installation, verification is performed using TCPDUMP to show that encryption is operating.

I. Introduction

We first introduce the technologies that are relevant to this effort. Each is described only to the extent of its motivational usefulness.

A. IMAP

IMAP (Internet Mail Access Protocol) is a mail access protocol primarily characterized by comparison with the older and more familiar POP (Post Office Protocol) protocol. When a mail user agent (MUA) (commonly termed an "email client") uses POP to access mail, the MUA downloads the messages to the local system, and usually deletes them from the server. The principal copy of the messages is now stored locally. In contrast, in an IMAP system, the principal copy of messages remains on the server until it is eventually deleted. This system allows users to access their mail from multiple locations and/or using multiple devices. ([MANAGING-IMAP])

The important aspects of IMAP in terms of this security effort are the encryption of the authentication exchange that begins an IMAP session, and the encryption of the data (email messages) that is transferred between client and server during a session. The IMAP protocol specifies two commands for user authentication, LOGIN and AUTHENTICATE. Authentication using LOGIN transmits a simple username & password pair in clear text. This is obviously a bad practice. Authentication using AUTHENTICATE performs the authentication using one of several more secure mechanisms, and can be used if the client and server both support at least one such mechanism. ([IMAP4REV1])

Although the authentication process can be completed securely, the IMAP specification does not itself support encryption of the payload data that is transmitted between client and server. This is the second area where IMAP security could be improved. The way we have chosen to make this improvement in this paper is using SSL encryption.

B. SSL

SSL (Secure Sockets Layer) is the network-layer protocol originally developed by Netscape that is widely accepted on the Internet as a means to authenticate and encrypt client/server connections. SSL runs on top of the TCP/IP layer, and under the application layer. In the SSL system, parties are identified using _certificates_. An SSL-enabled web server, for example, is identified with a certificate that is _signed_ (vouched for) by a _certificate_authority_ (CA). When a user's web browser connects to the site and is presented with the certificate, the browser confirms that the CA who signed the certificate is one it recognizes. (Browsers come with certificates for the major CAs.) ([SSL-INFO])

The build process for Stunnel includes the generation of an SSL certificate. The administrator has the option to purchase a CA signature for this certificate, but doing so is not required for this effort. In the author's small environment, although we want secure communications, it isn't appropriate for us to obtain the signature of a CA on the SSL certificate of our server. Instead, we used the _self-signed_certificate_ that the Stunnel build process generates.

When the IMAP communications we've discussed take place over an SSL connection, the entire session takes place securely. This includes both the authentication process and the transmission of payload data. SSL addresses both IMAP security problems identified above.

C. Stunnel

Stunnel is free software that allows arbitrary network services that do not already support SSL to be given an SSL "wrapper" so that they do. Stunnel is not itself an SSL implementation. Rather, it makes it possible to connect network service daemons with an existing SSL implementation on the system. The fact that Stunnel does not implement SSL does not mean that a large, necessary piece remains missing from this scheme. Free SSL implementations such as OpenSSL are available, and may already be installed on many systems. The clearest example of this is OpenSSH, a widely used SSH implementation. The OpenSSH package requires OpenSSL, so systems running OpenSSH will already have OpenSSL installed. If OpenSSL is not already installed, it is available conveniently. Other SSL implementations such as SSLeay also work with Stunnel. ([STUNNEL], [STUNNEL-MIRT])

II. Building and Installing Stunnel on the Server

This section details the process of building and installing the Stunnel software.

The IMAP server system to which SSL functionality was added was a Sun UltraSparc 5 system running Solaris 8. OpenSSL had already been installed. Source code for Stunnel 3.20 was obtained from the Stunnel website ([STUNNEL]). After decompressing and extracting the code, the build process was very ordinary. The familiar "configure, make, make install" process was all that was required. The GCC compiler was used in this step. Interesting output of these steps is shown below: (Uninteresting lines have been stripped out and are not indicated.)

(from the "configure" step)

```
   checking for SSL directory... /usr/local/ssl
   checking whether to use the libwrap (TCP wrappers) library...
   checking for hosts_access in -lwrap... yes
```

The existing OpenSSL installation was detected, as was the TCP Wrappers library. Access control using TCP Wrappers is supported, but is outside the scope of this document.

(from the "make" step)

```
   /usr/local/ssl/bin/openssl req -new -x509 -days 365 -nodes \
          -config stunnel.cnf -out stunnel.pem -keyout stunnel.pem
   Using configuration from stunnel.cnf
   Generating a 1024 bit RSA private key
   ...........++++++
   ......++++++
   writing new private key to 'stunnel.pem'
   -----
   You are about to be asked to enter information that will be
   incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name
   or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [PL]:US
   State or Province Name (full name) [Some-State]:state_sanitized
   Locality Name (eg, city) []:city_sanitized
   Organization Name (eg, company) [Stunnel Developers
   Ltd]:organization_sanitized
   Organizational Unit Name (eg, section) []:section_sanitized
   Common Name (FQDN of your server) []:fqdn_sanitized
   /usr/local/ssl/bin/openssl x509 -subject -dates -fingerprint -noout \
          -in stunnel.pem
   subject=
   /C=US/ST=state_sanitized/L=city_sanitized/O=organization_sanitized/
   OU=section_sanitized/CN=fqdn_sanitized
   notBefore=date1_sanitized 2001 GMT
   notAfter=date2_sanitized 2002 GMT
   MD5 Fingerprint=FI:NG:ER:PR:IN:T_:SA:NI:TI:ZE:D_:__:__:__:__:__
```

Unlike many builds of free software, the Stunnel build requires user interaction. The final step in the build process is the generation of the SSL certificate for the server. To this end, an RSA key is automatically generated, and saved in the "stunnel.pem" file. Then, as the sample output shows, the administrator is asked to provide detailed identification information to be incorporated into the server's certificate. (These details have been "sanitized" in the sample output, as has program output that includes them.)

The "make install" step installed /usr/local/sbin/stunnel and /usr/local/lib/stunnel.so, as well as a manpage and other supporting files. The stunnel.pem file can be found in the build directory. Its contents were two text blocks of the form shown below.

```
   -----BEGIN RSA PRIVATE KEY-----
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ********************  (sanitized)  *****************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ********************************************
   -----END RSA PRIVATE KEY-----
   -----BEGIN CERTIFICATE-----
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ********************  (sanitized)  *****************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ****************************************************************
   ************************************
   -----END CERTIFICATE-----
```

The permissions on stunnel.pem were changed to 600, and the file was moved to /usr/local/ssl/certs, a preexisting directory. As the Stunnel manpage instructs, blank lines were inserted after both "END" lines in this file.

With Stunnel successfully installed, the final step on the server side to support IMAP SSL encryption was to wrap the imap service. The process followed here is a combination of information from [DTCC], [INFN-SECURE-IMAP] and [STUNNEL].

The first step was the addition of an "imaps" line in the server's /etc/services file:

```
   imap    143/tcp     imap2  # Internet Mail Access Protocol v2
   imaps   993/tcp
```

Port 993 is the standard port for the SSL-encrypted IMAP service. Some documentation uses the name "simap" for this service.

Stunnel was then started with the command

```
   stunnel -D 7 -d imaps -p /usr/local/ssl/certs/stunnel.pem \
   -l /usr/local/sbin/imapd -- imapd
```

Explanation of options:

-D enables maximal debugging output, which was useful at this stage

-d specifies daemon mode, and to listen for connections on the specified port. Had the imaps line not been added to the /etc/services file, it would be necessary to specify port 993 numerically here.

-p specifies the location of the SSL certificate

-l specifies the service daemon to execute

The administrator can incorporate this command into an appropriate startup script.

III. Email client (MUA) configuration

Client configuration is the next task to be performed. This paper describes how to configure the MUA "Pine" to access a user's INBOX using the newly-established imaps service. ([PINE-SSL-FREEBSD], [PINE-EXCHANGE]) Configuring Pine to access other mailboxes works the same way.

The configuration changes necessary for Pine can be made from within the program, but can easily be made by editing the user's .pinerc file directly. The original .pinerc file read:

```
   # Path of (local or remote) INBOX, e.g. ={mail.somewhere.edu}inbox
   # Normal Unix default: local INBOX (usually /usr/spool/mail/$USER).
   inbox-path={imap_server_hostname_sanitized:143}INBOX
```

The syntax is peculiar to Pine, but should be understandable. Configuring Pine to use SSL was simply a matter of setting the "inbox-path" definition as shown below:

```
   inbox-path={imap_server_hostname_sanitized/SSL/NoValidate-Cert}INBOX
```

The syntax here deserves more explanation. Following the hostname specification is a sequence of two options, each preceded by the '/' character. The "SSL" option tells Pine to use the imaps service rather than the imap service. The "NoValidate-Cert" option furthermore tells Pine that the self-signed certificate we are using need not be validated. (This arrangement enables encryption, but does not address authentication. In our small environment, this is acceptable. Administrators of important production IMAP servers will probably want to do more than this.)

Clearly, for the Pine configuration to be so easy, the program itself must be designed to support SSL. Pine 4.33 is designed that way, and will incorporate SSL functionality if it is compiled with an SSL library. Just as with Stunnel, this was no problem since OpenSSL was already installed.

At this point, Pine can be used to securely check mail. Not only will the authentication process be encrypted, but so will the entire session.

IV. Encryption Verification

Although we have configured both client and server to communicate using SSL, it was our first ever attempt to do so. It seems very worthwhile to verify that the encryption is in fact working, rather than relying on an assumption that everything is operating as we think it ought. Having done so, we can be satisfied that the IMAP sessions are in fact now secure. We performed that verification using TCPDUMP. ([TCPDUMP])

TCPDUMP is the network sniffing software described earlier in this SANS course. It's important to note that the extent of our TCPDUMP analysis will be to _visually_ determine whether the IMAP communications are secure. We assume that if the communications _appear_ encrypted, they truly are. (We do not attempt any mathematical proof of SSL functionality itself.)

To perform the verification, we created two ".pinerc"-like files named ".pinerc-clear" and ".pinerc-ssl". These files simply represent the .pinerc file before and after the configuration change to SSL. We then used a symbolic link, ".pinerc", to activate one of these two files as desired. We began with .pinerc linked to .pinerc-clear.

The test we performed was to simply open Pine, go to the message index, view the first new message, and then quit. This action was taken on the client system, another Sun UltraSparc machine running Solaris 8. It was also on this system that we ran TCPDUMP.

The sequence of events was as follows:

1. Run TCPDUMP.

```
      tcpdump -s 0 -w pine-clear-tcpdump host client_sanitized and \
      host server_sanitized
```

```
   Explanation of options:
      -s   specifies that entire packets should be captured
```

```
    -w   specifies that the collected packets should be saved to a
         file rather than displayed
```

The packet selection expression specifies that all communications between the two systems should be captured. Since no other communication between the two would happen during this short test, a more stringent selection wasn't needed.

2. In another window, run Pine. Execute the series of steps described above.

3. Interrupt TCPDUMP.

4. Toggle the .pinerc link to point at .pinerc-ssl.

5. Run TCPDUMP again, this time to a file, "pine-ssl-tcpdump".

6. In the other window, run Pine as before.

7. Interrupt TCPDUMP.

8. Display the captured packets using TCPDUMP in a way that shows their full contents.

```
    tcpdump -r pine-clear-tcpdump -x -X
    tcpdump -r pine-ssl-tcpdump -x -X
```

Explanation of options:

```
    -r   tells TCPDUMP to take input from the file rather than from
         the network interface

    -x,-X  tells TCPDUMP to print the output in ASCII
```

The output for a portion of the non-SSL session is shown below. Only the ASCII portion is shown. One can see that the communications can be read with ease. Several sections are annotated.

```
15:34:12.365066 server_sanitized.imap > client_sanitized.37934: P
1:139(138) ack 1 win 24820 (DF)
E.....@.@.FQ..s.
..r.............
P.`.LL..*.OK.[CA      <  Server capability string clearly visible
PABILITY.IMAP4.I      <
MAP4REV1.STARTTL      <
S.LOGIN-REFERRAL
S.AUTH=LOGIN].**
****.IMAP4rev1.2
000.287.at.Fri,.
14.Sep.2001.15:3
4:12.-0400.(EDT)
..
15:34:12.365163 client_sanitized.37934 > server_sanitized.imap: .
ack 139 win 24820 (DF)
15:34:12.366033 client_sanitized.37934 > server_sanitized.imap: P
1:30(29) ack 139 win 24820 (DF)
E..Em"@.@..2..r.
..s...........w
P.`..r..00000000
.AUTHENTICATE.LO     <  Client authenticating
GIN..                <

... etc ...

15:34:20.742613 server_sanitized.imap > client_sanitized.37934: P
175:345(170) ack 50 win 24820 (DF)
E.....@.@.F-..s.
..r...........I
P.`.....*.CAPABI
LITY.IMAP4.IMAP4
REV1.STARTTLS.NA
MESPACE.IDLE.MAI
LBOX-REFERRALS.S
CAN.SORT.THREAD=
REFERENCES.THREA
D=ORDEREDSUBJECT
.MULTIAPPEND..00
000000.OK.AUTHEN
TICATE.completed
```

```
    ..
    15:34:20.743009 client_sanitized.37934 > server_sanitized.imap: P
    50:73(23) ack 345 win 24820 (DF)
    E..?m&@.@..4..r.
    ..s........I...E
    P.`..W..00000001
    .SELECT.INBOX..
    15:34:20.836908 server_sanitized.imap > client_sanitized.37934: .
    ack 73 win 24820 (DF)
    15:34:21.533045 server_sanitized.imap > client_sanitized.37934: P
    345:699(354) ack 73 win 24820 (DF)
    E.....@.@.Es..s.
    ..r........E...`
    P.`.....*.466.EX     <  Details about the mailbox clearly visible
    ISTS..*.0.RECENT     <
    ..*.OK.[UIDVALID     <
    ITY.998577197].U
    ID.validity.stat
    us..*.OK.[UIDNEX
    T.11412].Predict
    ed.next.UID..*.F
    LAGS.(\Answered.
    \Flagged.\Delete
    d.\Draft.\Seen).
    .*.OK.[PERMANENT
    FLAGS.(\*.\Answe
    red.\Flagged.\De
    leted.\Draft.\Se
    en)].Permanent.f
    lags..*.OK.[UNSE
    EN.439].first.un
    seen.message.in.
    /var/mail/clu..0
    0000001.OK.[READ
    -WRITE].SELECT.c
    ompleted..
    15:34:21.537677 client_sanitized.37934 > server_sanitized.imap: P
    73:88(15) ack 699 win 24820 (DF)

    ... etc ...

    15:34:26.183299 client_sanitized.37934 > server_sanitized.imap: .
    ack 19041 win 24820 (DF)
    E..(m4@.@..=..r.
    ..s........)..GM
    P.`.....
    15:34:28.832248 client_sanitized.37934 > server_sanitized.imap: P
    530:573(43) ack 19041 win 24820 (DF)
    E..Sm5@.@.....r.
    ..s........)..GM
    P.`.%a..00000009
    .SEARCH.ALL.1:42
    5,451:466.DELETE
    D..
    15:34:28.833552 server_sanitized.imap > client_sanitized.37934: P
    19041:19081(40) ack 573 win 24820 (DF)
    E..P..@.@.F...s.
    ..r.......GM...T
    P.`.%`..*.SEARCH
    ..00000009.OK.SE
    ARCH.completed..
    15:34:28.834025 client_sanitized.37934 > server_sanitized.imap: P
    573:590(17) ack 19081 win 24820 (DF)
    E..9m6@.@..*..r.
    ..s........T..Gu
    P.`.2...0000000a
    .LOGOUT..
    15:34:28.916564 server_sanitized.imap > client_sanitized.37934: P
    19081:19165(84) ack 590 win 24820 (DF)
    E..|..@.@.Fm..s.
    ..r.......Gu...e
    P.`.n...*.BYE.**
    ****.IMAP4rev1.s
    erver.terminatin
    g.connection..00
    00000a.OK.LOGOUT
    .completed..
```

The output of the SSL-encrypted session, however, is quite different:

```
15:35:02.046844 client_sanitized.37935 > server_sanitized.993: P
1:131(130) ack 1 win 24820 (DF)
E...m;@.@.....r.
..s../...P.~.t..
P.`.%.........W.
................
.f..............
.........e..d..c
..b..a..`.......
....@...........
..........L.WF.a
T$...l....8...HN
i7"...D...
15:35:02.059043 server_sanitized.993 > client_sanitized.37935: . ack
131 win 24820 (DF)
E..(..@.@.F...s.
..r..../.t...P..
P.`.?...UUUUUU
15:35:02.089882 server_sanitized.993 > client_sanitized.37935: P
1:801(800) ack 131 win 24820 (DF)
E..H..@.@.C...s.
..r..../.t...P..
P.`.r.......J...
F..;.[...3cD]+W.
......V........V
E...7JE........R
J...8.ZW}..9{...
................
......0...0.....
......0...*.H...
.....0..1.0...U.
...US1.0...U....
****1.0...U....*
********1(0&..U.     <  certificate exchange (sanitized, but at
...*************     <  least some was readable)
****************     <
**1.0...U....***
**************1.
0...U....*******
************0...
*************..0
20905202130Z0..1
.0...U....US1.0.
..U....****1.0..
.U....*********1
(0&..U....******
****************
*********1.0...U
....************
*****1.0...U....
****************
***0..0...*.H...
.........0......
...v(#..........
.f.0_......W8..i
.R~...cB.\.fD...
....].......s...
.....iI.......S.
.b[E7.q<.W.g....
.Gb.........`...
D.,7?.h....@....
.......0.0...`.H
...B.......@0...
*.H............c
...|W.B.t.r..g..
..i..%R..n.w.[..
.xI.B....N..\...
....i.0e1......n
h......ur..:'.-.
4.0..s{.@~.$#...
m.0...w.4.b.#...
....X...SK......
........

... etc ...

15:35:12.891013 server_sanitized.993 > client_sanitized.37935: P
1847:2428(581) ack 676 win 24820 (DF)
E..m..@.@.Dm..s.
..r..../.t.K.P.!
```

```
    P.`.........@..1
    ..Q2..].$...q.,.
    .....I.h.5...y.F
    EY..@.^X........
    ...L.yz....p....
    ...1.8\...4H=...
    .Kzq~.;X...yA..w
    <....5s..).....2
    .}.-.;....P`L...
    .......a.P.....U
    vd.P.C\..o..p...     <  All communication is unreadable.
    .ap.....E82S`..&
    ..r......>U..F.F
    K....i5*~2..a...
    .a.D.......[..c.
    .).-...5..y..|..
    `..P..n.^.8OI.<.
    .;<...6.i...t,.i
    !.[.....<....Y..
    L.....t........^
    ....>:XxL<...}..
    N.....L...[.....
    ....'K...^..>"6.
    ........+..%....
    ...fW.....Js...o
    B.O".*.>..@!<~A1
    ....0H......Ekp.
    ~...q...c.rU...E
    ....a..D....-S,.
    8.Z..4.5...|.T3k
    .,.F$.}h.rW}N...
    ..`....Qu.c.I..V
    .X7.zn.>.f$2s..,
    .BM..F../....g..
    ....<\L...s..e.G
    .....K?..+<d.#..
    .M.......i`J(
```

Having seen that the communications are unreadable, we can be satisfied that SSL encryption is working properly.

V. Conclusion

In this paper, we have described the relevant concepts, and detailed a simple method for securing an IMAP session with SSL using Stunnel. This same method can be used to encrypt and verify other network services as well.

APPENDIX: References

[DTCC] http://www.dtcc.edu/cs/admin/notes/ssl/

[IMAP4REV1] RFC 2060 ("Internet Message Access Protocol - Version 4rev1."). Available from various locations, including http://src.doc.ic.ac.uk/computing/internet/rfc/rfc2060.txt

[INFN-SECURE-IMAP] http://security.fi.infn.it/tools/stunnel/index-en.html

[MANAGING-IMAP] Managing IMAP. Dianna Mullet & Kevin Mullet. O'Reilly. Sep 2000.

[PINE-EXCHANGE] http://www.cs.washington.edu/lab/sw/email/pine-exchange.html

[PINE-SSL-FREEBSD] http://www.viperstrike.com/~lopaka/sysadmin/pine-4.30-ssl-freebsd

[SSL-INFO] http://developer.netscape.com/docs/manuals/security/sslin/contents.htm

[STUNNEL] http://www.stunnel.org

[STUNNEL-MIRT] http://stunnel.mirt.net/

[TCPDUMP] http://www.tcpdump.org/

FINAL NOTE

The typography and other artifacts of this paper are the author's combination of various styles he has seen. They should not be considered an example of good style. Apologies are given where any accepted practices for documentation with which the author is unfamiliar have been violated.