OpenSSL

Ryan Matteson
matty@daemons.net
http://daemons.net/~matty

Atlanta Linux Enthusiasts
August 14, 2003

A Tiny Bit Of Theory

What is SSL and TLS?

The SSL protocols define a set of rules about when to send and not send messages

SSL provides confidentiality, authentication and message integrity

Utilizes cryptographically sound message digests and symmetric encryption algorithms

Provides the ability to resume sessions (SSL session IDs allow this)

SSLv1 and SSLv2 were developed by Netscape

SSLv3 and TLSv1 devloped by a community

Defines two roles: client and server

How Does SSL Work?

SSL works on a message passing paradigm

Client initates connection, proposing ciphers and digest algorithms (specified in ClientHello)

Server selects the cipher/MD/SSL version to be used (specified in ServerHello)

ChangeCipherSpec is used to enable the negotiated security services

What is an x.509 Certificate?

Digital equivalent of a drivers license

Attests to the truth of an individual or Organization

Usually issued by a certificate authority

Come in three flavors: Personal, Code signing and Web Site certificates

CA certificates are special, the issuer and subject are the same

Certificate Authorities certify their own identity

Digital certificates contain a variety of information, including:

Version	Serial Number	Issuer
Subject	Algorithm Identifier	Extensions
Signature	Period Of Validity	Subject's Public Key
Subject Unique ID	Issuer Unique ID	

What is a Certificate Authority?

Establishes trust (You trust the big CAS, don't you?)

When a CA “signs” your certificate, they are vouching that you are who you say you are
(Verisign accidentally issued two certs to a hacker claiming to be MSFT, oops)

Two forms of Certificate Authorities: Private and Public

Public Certificate Authorities (Verisign, Thawte, etc.) issue certficates to the world

Private Certificate Authorities (OpenCA, Entrust CA) issue certificates to an organization

Certificate delegation/chaining allows a CA to delegate signing capabilities

What Tools Are Available To Work With SSL And TLS?

OpenSSL

Set of opensource cryptographic libraries

Implements SSLv2, SSLv3 and TLSv1

Support for OCSP Version 1 and 2

Support Certificate Revocation Lists

APIs and CLI can be used to encrypt and sign data

If you have a recent version of Linux or FreeBSD/OpenBSD, openssl is included

Configurable via a single configuration file (Default openssl.cnf)

Can be configured to use hardware cryptographic accelerators

Previously known as SSLeay (Eric A. Young was original maintainer)

Current version as of this writing is 0.9.7b

You can find the software at: http://www.openssl.org

Stunnel (Universal Secure Tunnel)

Works as a wrapper for insecure services

Adds wire-level security for POP3, IMAP, HTTP and NNTP and SMTP

Supports client and server certificates

Stunnel version 4.X uses a runtime configuration file

Supports chroot() jails

Current version as of this writing is 4.04

You can find the software at: http://www.stunnel.org/

SSL Dump

SSLv3/TLS network protocol analyzer

Packet Capture (PCAP) is used to intercept packets on the wire

Works on most UNIX platforms (FreeBSD, Linux, Solaris, HP-UX, etc)

Provides facilities to view SSL messages, application data and state changes

Syntax is similar to tcpdump

The current version as of this writing is 0.9b3

You can find the software at: http://www.rtfm.com/ssldump/

Fun Things To Do With OpenSSL And Friends

Creating RSA & DSA Keys

"genrsa" and "gendsa" allow you to create RSA and DSA private keys

"rsa" and "dsa" allow you to extract public keys from the private key

"rsautl" allows you to encrypt, decrypt and sign data (it is not intended for this purpose)

Example:

```
# Generate a set of DSA parameters
> openssl dsaparam -out dsaparam 2048

# Generate a DSA key
> openssl gendsa -out dsaprivkey.pem -des3 dsaparam

# Print the DSA public key associated with the private key above
>  openssl dsa -in dsaprivkey.pem -pubout -out dsapubkey.pem

# Generate an RSA private key and encrypt it with 3DES
> openssl genrsa -des3 -out rsa.key 2048

# Print the public key associated with the RSA private key
> openssl rsa -in rsa.key -pubout
```

Creating X.509 Certificates

openssl allows you to create CSRs and x.509 certificates

Support DSA and RSA keys

Supports PEM and DER formatted certificates

-nodes can be used to strip passwords from private keys (not recommended!)

Default certificate fields can be specified in openssl.cnf

Example:

```
# Generate an X.509 Certificate
> openssl req -x509 -outform PEM -keyform PEM -keyout cert.key \
                -days 500 -out cert.crt -newkey rsa:2048

Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Georgia
Locality Name (eg, city) [Newbury]:Atlanta
Organization Name (eg, company) [My Company Ltd]: H8x0R Computing
Organizational Unit Name (eg, section) []: IT
Common Name (eg, your name or your server's hostname) []: www.h8x0r.com
Email Address []: postmaster@h8x0r.com
```

Printing The Contents Of A Certificate

"x509" and "req" options allow you to print certificates

Individual options to print issuer, serial number, hash, subject, fingerprint etc. etc.

Example:

```
# Print the contents on an x509 certificate
>  openssl x509 -in cert.crt -text

# Print the Subject Field on a certificate
>  openssl x509 -subject -in cert.crt -noout

# Print the Issuer of the certificate
> openssl x509 -issuer -in cert.crt -noout

# Print the date the certificate will expire
>  openssl x509 -enddate -in cert.crt -noout

# Print out the contents of a server certificate
> openssl s_client -connect www.giddie-up.net:443 -showcerts
```

Becoming A Certificate Authority

Create supporting files and directories

Create a key pair for signing certificates

Protect your private key with your life

Example:

```
# Setup directories and files
> mkdir /etc/ca
> mkdir /etc/ca/certs
> mkdir /etc/ca/keys
> chmod 700 /etc/ca /etc/ca/keys
> echo '1' > /etc/ca/serial.txt
> touch /etc/ca/index.txt

# Create your certifcate and self-sign it
> openssl req -x509 -newkey rsa:2048 -out cacert.pem -outform PEM \
 -keyout cakey.pem -keyform PEM

# Generate a certificate signing request
> openssl req -newkey rsa:1024 -keyout testkey.pem -keyform PEM \
                    -out testreq.pem -outform PEM

# Sign the CSR
> openssl ca -in testreq.pem
```

Revoking Certifcates

Allows your CA to revoke certs who have lost keys or certs

Example:

```
# Revoke a bad certificate
>  openssl ca -revoke testcert.pem

# Generate a CRL to post to our website/OCSP
> openssl ca -gencrl -out certs.crl
```

Encrypting And Decrypting Data

Supports various symmetric ciphers (AES, 3DES, Blowfish, RC4, Cast5)

Encrypted data can be Base64 encoded for easy trasport

Supports CBC, CFB, ECB and OFB modes of encryption

Data can be left as 0s and 1s for easy storage

Example:

```
# Encrypt /etc/services
> openssl enc -in /etc/services -out /tmp/services.enc.blfsh -base64 -blowfish

# Use a horrible encryption algorithm so others can view my data
> openssl enc -in Finances.xls -out Finances.xls.enc.rc4 -rc4-40

# Let's base64 encode /etc/services so I can send it to someone
> openssl enc -in file.bin -out file.bin.b64 -base64

# Decrypt a blowfish encrypted file
> openssl enc -d -in services.enc.blfsh -blowfish -base64
```

Generating Message Digests

Generates MD5, SHA1 and RIPEMD160 checksums

Can produce colon delimited checksums for easy parsing

Can verify a checksum using an existing public key

Can verify a checksum using a private key

Can sign a message digest with your private key

Can output checksum as binary or hex

Example:

```
# Generate a SHA1 checksum of /etc/services
> openssl dgst sha1 /etc/services

# Generate an MD5 checksum of /etc/services
> openssl md5 /etc/services

# Generate a RIPEMD160 Checksum of /etc/services
> openssl rmd160 /etc/services
```

Testing An SSL-Enabled Server

s_client option can mimic an SSL client

s_server option can mimic an SSL-enabled server

Supports SSLv2/SSLv3 and TLSv1

Allows for client and server side certificates

s_client can be used to make sure your web servers are up and running

Example:

```
#!/bin/sh

HOST="localhost"
OPENSSL_BINARY="/usr/local/ssl/bin/openssl"
PORT="443"
LOG_ERROR_MESSAGE="apache_connect_failure"
TMP=/tmp/connect.$$
OPT="s_client -tls1 -quiet"

${OPENSSL_BINARY} ${OPT} -connect ${HOST}:${PORT} > ${TMP} 2>&1 << EOF
GET / HTTP/1.0


EOF

if egrep "Server:" ${TMP} > /dev/null
then
   :
else
   logger -p daemon.notice "WEB_SERVER_ERROR (${LOG_ERROR_MESSAGE}):  \
                            Could not connect to ${HOST} on TCP Port ${PORT}"
fi

rm -f ${TMP}
```

Securing Insecure Services With Stunnel

"[]" defines the service to be used with this connection

"client" specifies if this is a server or client

"accept" specifies the host and port to listen on

"connect" is the remote end of the connection to connect to upon an accept()

Logfiles can be specified with the "output" directive

" Cafile" can be used with "verify" to validate certificates

Server certificates can be specified with the " cert" directive

"setuid" and "setgid" can be used to change the UID/GID the server runs as

```
# Run the Stunnel Daemon
>/usr/sbin/stunnel /etc/stunnel/stunnel-client.conf

# Stunnel client/server configuration
/etc/stunnel/stunnel-client.conf

client=yes
Cafile=/etc/stunnel/cacert.pem
pid=/var/run/stunnel.pid
output = /var/log/stunnel.log

[imaps]
accept=127.0.0.1:imap
connect=imap.matty.com:imaps
```

Example:

Debugging SSL and TLS Problems

ssldump can dump messages, state changes and application data

Traffic can be isolated by port, src/dst IP, and network

Example:

```
# Dump the SSL messages and state changes
> ssldump -t -a -A -i eth0 host zoom

# Dump out all SSL traffic destined for zoom, including application data
> ssldump -d -t -a -A -k privatekey.pem -p password  -i eth0 host zoom
```

Learning More?

SSL and TLS Essentials, Stephen Thomas

*Network Security With openSSL, V*iega, Messier and Chandra

SSL and TLS RFCs

OpenSSL, SSLDump and Stunnel websites

Questions?