Testing your TLS Implementation with Maxwell Pro TLS Test Suite

What is the Transport Layer Security (TLS) protocol?

The Transport Layer Security (TLS) protocol provides a mechanism for encrypted network communications at ISO Layer 4. It is primarily used to provide a secure environment for data exchange between client-server applications.

TLS was developed in an IETF (Internet Engineering Task Force) Working Group to address the shortcomings of its predecessor, SSL (Secure Sockets Layer). Unlike most other protocols, TLS has two external requirements for the server side:(1) a digital certificate and (2) a public key.

What's Inside...

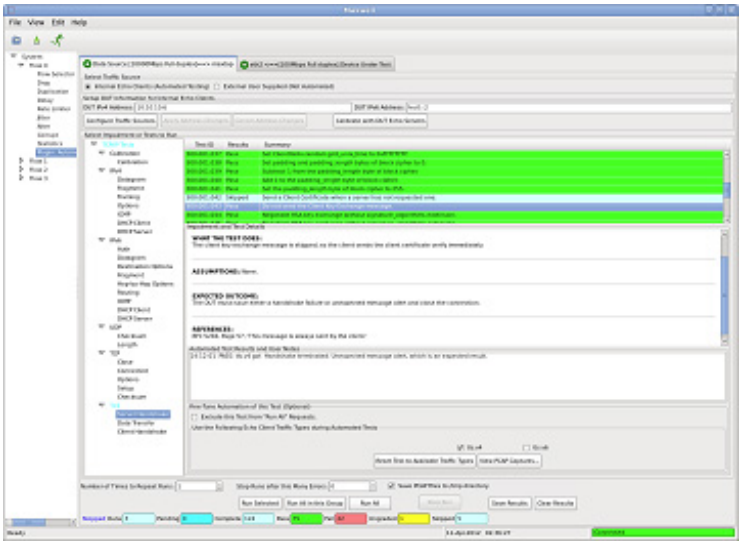
Kings Village Center #66190
Scotts Valley, CA 95067
iwl.com
+1.831.460.7010
info@iwl.com

Migrating from SSL to TLS 1.2

Recent revelations like the POODLE vulnerability, have demonstrated that SSL 3.0 should be disabled and removed from any products that still contain it. Companies taking this step include Twitter, Apple, eBay, Mozilla Firefox, Google, and PayPal. Immediate migration to TLS 1.2 is necessary. TLS must be properly implemented in the web browser and the server. IWL offers a TLS 1.2 Test Suite to identify bugs, problems, and flaws in TLS implementations (including POODLE).

Testing TLS Implementations with Maxwell Pro

The Maxwell Pro TLS Test Suite is used by design engineers, quality assurance engineers and testers to find and fix bugs in their TLS stack or engine. The tests help ensure that the TLS implementation is sufficiently robust so that it is not vulnerable to the wide range of attacks in today's Internet.



Tests cover both the TLS client side and the TLS server side.

The tests make use of the Maxwell Pro network emulation environment, so that each test sequence can intelligently impair all aspects of TLS 1.2, as defined in:

- IETF RFC 5246 "Transport Layer Security, Version 1.2"
- IETF RFC 6176 "Prohibiting Secure Sockets Layer (SSL) Version 2.0"

The TLS Test Suite contains more than 100 unique test cases that take on parameters for greater coverage. The tests ensure TLS 1.2 compliance through vulnerability and robustness testing. The Test Suite supports TLS over TCP/IPv4 or TCP/IPv6.

- Are you under time pressure to test and verify that your TLS implementation is correct?
- Are you disappointed by test suite products that do not let you incorporate your own data sources?
- Are you frustrated that you have to do a porting exercise to instrument the TLS stack before you can use a test suite product?
- Are you underwhelmed by the support engineer who knows very little about the TLS protocol?
- Would you like to see pass/fail results so you do not have to analyze complicated outputs?
- Do you need to replicate a customer reported bug, but don't have a way to customize the environment to properly test and replicate the problem?

The Maxwell Pro TLS Test Suite can provide you with the customization and flexibility you need to accurately test your implementation and meet your schedules. The tests are grouped into categories: Server Handshake, Data Handshake, and Client Handshake, with two traffic sources: "TLS Client (tls)" and "TLS Server (tlss)."

Maxwell Pro TLS Test Suite List of Tests:

- Set DigestInfo.AlgorithmIdentifier.parameters field for cryptographic hash functions to invalid values.
- Set ProtocolVersion major and minor fields to invalid values in ClientHello.
- Set CBC pad value to invalid lengths.
- Set Handshake length field to invalid lengths.
- Client replaces ClientHello with ServerHello handshake type.
- Client replaces ClientKeyExchange with ServerKeyExchange handshake type.
- Client sends CertificateRequest after ClientHello.
- Set record fragment length field to invalid values.
- Send invalid values during handshake.
- Send Application data record before first ClientHello.
- Set Client.SessionID fields to invalid values.
- Set Client.CipherSuite fields to invalid values.
- Set Client.CompressionMethod fields to invalid values.

- Change the ChangeCipherSpec message's type to various values.
- Set HandshakeType of finished type to invalid value.
- Set ClientHello.extensions length so that it extends past end of message.
- Manipulate SignatureAndHashAlgorithm.hash value to various values.
- Change the first SignatureAndHashAlgorithm.signature value to various values.
- Do not send a Certificate when the server sends a CertificateRequest.
- Set ClientHello.random.gmt_unix_time to various values.
- Manipulate padding and padding_length bytes of block cipher.
- Send a Client Certificate when a server has not requested one.
- Do not send the Client Key Exchange message.
- Negotiate RSA key exchange without signature_algorithms extension.
- Negotiate DSS key exchange without signature_algorithms extension.
- Test that the server sends ServerKeyExchange for DHE_DSS key exchange.
- Test that the server sends ServerKeyExchange for DHE_RSA key exchange.
- Test that the server sends ServerKeyExchange for DH_anon key exchange.
- Test that the server doesn't send ServerKeyExchange for RSA key exchange.
- Send an RSA-Encrypted Premaster Secret Message that cannot be processed.
- Send Finished message immediately after ClientHello sent.
- Manipulate fields and values of ClientHello.extensions.
- Compute PRF in Client Finished message using an incorrect finished_label.
- Verify implementation of TLS_RSA_WITH_AES_128_CBC_SHA.
- Create compressed record that decompresses to various values.
- Test that the server implements TLS_RSA_WITH_NULL_SHA256.
- Test that the server implements TLS_RSA_WITH_AES_128_CBC_SHA256.
- Test that the server implements TLS_RSA_WITH_AES_256_CBC_SHA256.
- Test that the server implements TLS_DHE_DSS_WITH_AES_128_CBC_SHA256.
- Test that the server implements TLS_DHE_RSA_WITH_AES_128_CBC_SHA256.
- Test that the server implements TLS_DH_anon_WITH_AES_128_CBC_SHA256.
- Test that the server implements TLS_DHE_DSS_WITH_AES_256_CBC_SHA256.
- Test that the server implements TLS_DHE_RSA_WITH_AES_256_CBC_SHA256.
- Test that the server implements TLS_DH_anon_WITH_AES_256_CBC_SHA256.
- Reuse SessionID but exclude previous cipher suite from ClientHello.
- Reuse SessionID but exclude previous compression method from ClientHello.
- Renegotiate handshake only up to but not including ChangeCipherSpec.
- Manipulate ContentType with invalid values.
- Introduce max and min TLSPlaintext fragments.
- Send various types of Alerts, both valid and invalid, at different points in the handshake.

- Set record fragment length field to zero for ChangeCipherSpec types.
- Set TLSCiphertest.length one longer than actual block length.
- Server replaces ServerHello with ClientHello handshake type.
- Client replaces ServerKeyExchange with ClientKeyExchange handshake type.
- Introduce invalid operations and values with ServerHello.
- Introduce invalid operations and values with Server.SessionID.
- Introduce invalid operations and values with Certificate.certificate_list.
- Compute PRF in Server Finished message using an incorrect finished_label.
- Verify implementation of TLS_RSA_WITH_AES_128_CBC_SHA.
- Send signature_algorithms extension to TLS client
- Manipulate fields and values of ServerKeyExchange message.
- As anonymous server, issue various invalid requests
- Verify that a TLS server does not accept downgrade to SSLv3.*
- Verify that a TLS server does not accept zero padding in block ciphers.*
- Verify that a TLS client does not accept downgrade to SSLv3.*
- Verify that a TLS client does not accept zero padding in block ciphers.*
- Verify that a TLS client does not accept non-standard padding in block ciphers.*

TLS Client and Server Side Test Coverage

Maxwell Pro supports both server side and client side TLS testing.

Server side testing is possible straight “out of the box” with no programming needed. Client side testing is almost as simple, only requiring you to find some way to automate repeated initiations of client connections from the target test machine to Maxwell Pro.

After the initial connection handshake, the TLS protocol dialog for data transfer is effectively identical for the client and server side. Thus, only a small fraction of any tests from any supplier are relevant to the data transfer phase. The Maxwell Pro tests provide 100% dialog coverage, and the most complex portion, the handshake, can always be tested with no extra porting or programming effort. You can just use an existing TLS client or server application on your target test machine. Only a couple of the data transfer tests may benefit from writing a TLS echo service program for the target test machine.you change ‘ANYIF=No;’ to ‘ANYIF=Yes;’ then the server will listen on any interface on this server computer.

Establishing a source of authority

The Maxwell Pro TLS Test Suite references the RFCs that correlate to each test area. These official IETF documents detail the Internet standards and best current practices that can point the user toward abetter understanding of the problem.

Sample Test Documentation...

PURPOSE OF THE TEST

Make the length field of the first extension in ClientHello.extensions invalid.

WHAT THE TEST DOES

The length field of the first extension in the ClientHello.extensions list is set to 2^15-1, which will exceed the Handshake.length field.

ASSUMPTIONS

None.

EXPECTED OUTCOME

DUT should issue an illegal parameter alert and must close the connection.

RELEVANT TRAFFIC SOURCES

tls.v4, tls.v6

REFERENCES

RFC 5246, Page 37, 'struct ... Handshake', page 41, 'extensions'.'

Real World Application

Application 1: Maxwell Pro discovered SSL Security Flaw in iOS

Executive Summary: The Maxwell Pro TLS Test Suite includes nine tests related to ServerKeyExchange; three of these tests would have yielded failing grades if flawed TLS or SSL code had been tested with the Test Suite.

The problem found in the open source SSL code used in iOS may be reviewed here:

http://opensource.apple.com/...Exchange.c?txt

In the function SSLVerifySignedServerKeyExchange() there are two goto statements where there should only be one:

```
...
        if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
                 goto fail;
                 goto fail; /* <<----- ****** Problem! ******/
        if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
                goto fail;

        err = sslRawVerify(ctx,
                           ctx->peerPubKey,
                           dataToSign, /* plaintext */
                           dataToSignLen, /* plaintext length */
                           signature,
                           signatureLen);
        if(err) {
                sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                           "returned %d\n", (int)err);
                goto fail;
                }
fail:
        SSLFreeBuffer(&signedHashes);
        SSLFreeBuffer(&hashCtx);
        return err;
...
```

The second indented “goto” causes a section of code to get skipped (it is unreachable, and the compiler allegedly fails to warn about that.) The code that gets skipped is verifying the ServerKeyExchange message for SSL. That code validates that the host is who it claims to be.

The Maxwell Pro TLS Test Suite includes nine tests related to ServerKeyExchange; three of these tests would have yielded failing grades if flawed TLS or SSL code had been tested with the test suite.

Application 2: Gnu TLS cryptographic bug

The Red Hat Bugzilla report explained:

> “A malicious server could use this flaw to send an excessively long session ID value, which would trigger a buffer overflow in a connecting TLS/SSL client application using GnuTLS, causing the client application to crash or possibly execute arbitrary code.”

IWL used our Maxwell Pro TLS Test Suite to test one version of the GnuTLS library. It failed 39 out of 116 of our tests! Fortunately most of those failures are benign, typically caused by responding with the wrong response to a malformed or out of sequence message. Others, such as improper handling of buffer overruns, can be open to exploits. Because this bug attacks clients rather than servers, it requires extra steps at subterfuge to be taken by attackers to exploit it.

The key point: anyone using our Maxwell Pro TLS Test Suite would have identified this bug, thanks to our test for session ID overflow.

Application 3: POODLE Bug

The POODLE Bug allows an attacker to perform a man-in-the-middle attack in order to decrypt HTTP cookies. This attack can force a connection to “fallback” to a less secure version of the protocol. The Maxwell Pro TLS Test Suite has five tests to detect this vulnerability.

Summary

Anything less than vigorous testing of your security protocols leaves you open to cyber attacks -- attacks that can ruin your reputation and end your career. Why take these unnecessary risks when the Maxwell Pro TLS Test Suite can systematically find and report bugs in your TLS client and TLS server implementations? To learn more, contact IWL.

Want to learn more?

IWL
Kings Village Center #66190
Scotts Valley, CA 95067
iwl.com
+1.831.460.7010
info@iwl.com