Guide to sslcaudit v1.0rc1

May 1st 2012

GREMWELL BVBA
Sint-Katherinastraat 24, 1742 Ternat
Tel: +32 2 215 53 58 Fax: +32 2 791 98 38

Document Properties

Subject	Guide to sslcaudit v1.0rc1
Author	Alexandre Bezroutchko E-mail: abb@gremwell.com Web site: http://www.gremwell.com/
Document history	1/05/2012 - Initial version

Table of Contents

1 INTRODUCTION

The goal of sslcaudit project is to develop a utility to automate testing SSL/TLS clients for resistance against MITM attacks. It might be useful for testing a thick client, a mobile application, an appliance, pretty much anything communicating over SSL/TLS over TCP.

This document is based on sslcaudit version 1.0rc1. It contains some background information, explanations of how the tool works, and several examples. An impatient reader can jump directly to 4 HOW TO USE THE TOOL.

2 BACKGROUND INFORMATION

SSL/TLS protocols are widely used to protect confidentiality and integrity of communication over untrusted networks. For protection to be effective, client and server both have to be implemented correctly. Security properties and common implementation flaws in servers are well understood and documented [WIKI-TLS, SCANIT-SSL, OWASP-TLS]. There is the OWASP Testing Guide [OWASP-TLS], a rating guide [SSL-RATING], and tools to automate the tests, such as sslaudit [SSLAUDIT].

When it comes to client security, things are less advanced. Till recently sslsniff [SSLSNIFF] attacking tool was probably the most interesting effort in this direction. A recent Blackhat presentation [BH-SSL-TTRUST] focuses on security issues introduced by SSL-aware proxies and mentions common implementation flaws in SSL clients. The authors of that presentation have published an online testing service [SSLTEST] suitable for testing web browsers.

3 WHAT WE TEST FOR

In general, the tool is designed to automate testingTo assess security of SSL/TLS clients we have to check:

- what server certificates it trusts enough to fully establish a connection,
- what flavors of SSL protocol the client supports

Behavior related to server certificate validation:

C1	Reject self-signed certificates or certificates not signed by a trusted CA	In practice a failure to implement C1, C2, or C3 is the most dangerous and allows for a straightforward MITM attack.
C2	Validate basic constraints of intermediate CAs	
C3	Only accept server certificate with CN matching the intended destination	
C4	Do not accept expired and revoked certificates	Testing for or exploitation of C4 has the prerequisite of an attacker being able to obtain a legitimate, but

		expired or revoked certificate for the server or an intermediate/root CA. Some SSL clients (especially embedded ones) don't have a reliable clock source nor CRL/OSCP support at all by design.
C5	Do not be fooled by NUL-character in CN.	To test or exploit C5 a valid certificate with NUL-byte in CN is needed. It is difficult to obtain a such a certificate in practice.

Protocol version and cipher support:

P1	Do not support SSLv2 (version/cipher downgrade)	The failure to implement P1 leads to theoretical possibility of cipher downgrade attacks. To our knowledge practical exploitation is very tricky, no free or commercial tool exist.
P2	Do not support SSL and TLS 1.0 (CBC attack)	P2 was demonstrated to allow cookie theft in web browsers, and has a prerequisite of an attacker being able to inject malicious JavaScript code into victim's browser [BS-BEAST]
P3	Do not support weak key exchange protocols, low key lengths, low ciphers strengths	If strong ciphers are supported by the peers, the presence of weak ones is only exploitable via cipher downgrade attack.

Testing for C1-C3 is already implemented in v1.0. Checking C4, C5, and protocol-level tests will come in v1.1.

3.1 Server certificate validation tests

Sslcaudit starts with a user-supplied certificate, if provided with --user-cert parameter. An unprotected private key must be provided as well with --user-key parameter.

Next, sslcaudit automatically generates certificates with the following properties:

1. With hardcoded CN (nonexistent.gremwell.com). Can be disabled with --no-default-cn
2. With user-specified CN, if any. Use --user-cn
3. Matching attributes of a certificate fetched from from user-specified SSL/TLS server. To enable use --server HOST:PORT.

Each certificate will be signed in the following ways:

1. Self-signed. To disable, specify --no-self-signed.
2. Signed by the user-supplied certificate (--user-cert / --user-key). To disable use --no-user-cert-signed.
3. Signed by the user-supplied CA (--user-ca-cert / --user-ca-key).

This way, sslcaudit will use between 1 and 10 certificates.

3.2 Testing for protocol version and cipher support

Will come in v1.1. The functionality will be similar to sslaudit [SSLAUDIT], but backwards.

3.3 Other security issues

Just for completeness, there are two other attacks having an impact on SSL/TLS communication:

- "SSL 3.0/TLS 1.0 renegotiation attack" [TLS-RENEG], but has no client-side effects.
- Another related (but not SSL/TLS-specific) attack is [OSCP-ATTACK], not testing for it.

4 HOW TO USE THE TOOL

4.1 Installation and dependencies

There is no procedure for installation yet. Just grab the code:

- Download ZIP archive at https://github.com/grwl/sslcaudit/zipball/release_1_0_rc1
- Or clone leading edge master GIT repository: git clone git://github.com/grwl/sslcaudit.git
- Find sslcaudit in the top level directory and run it with -h option.

Sslcaudit uses M2Crypto Python library. If you dependencies problem, you might see following:

```
$ ./sslcaudit
Traceback (most recent call last):
…
ImportError: No module named M2Crypto
```

On Debian system M2Crypto library can be installed with the following command:

```
$ sudo apt-get install python-m2crypto
```

4.2 Lab setup

To use sslcaudit, a penetration tester has to convince the client under test to establish a series of connections to the listener of sslcaudit. This can be done in number of ways, for example with Marvin [MARVIN], but this topic is outside of the scope of this document. Sslcaudit plays a role of a rogue SSL/TLS server, presenting the client with various certificates and logging the outcome of the tests.

Sslcaudit does not (yet) do any risk assessment. Instead it displays the information about what certificate configurations have been tried and what has been observed. It is up to the user to make conclusions, which are obvious in most cases.

Below we will consider four examples showing how sslcaudit helps testing the behavior of SSL

clients.

4.3 Example 1

Open two terminal windows. In one window run 'sslcaudit'.

```
$ ./sslcaudit
```

The command starts and runs silently. If you dependencies problem, you might see

```
$ ./sslcaudit
Traceback (most recent call last):
…
ImportError: No module named M2Crypto
```

On debian system required M2Crypto library can be installed with

```
$ sudo apt-get install python-m2crypto
```

By default it listens on all interfaces on port 8443.

In another terminal run openssl to connect to sslcaudit.

```
$ openssl s_client -connect localhost:8443
CONNECTED(00000003)
depth=0 /CN=nonexistent.gremwell.com/C=BE/O=Gremwell bvba
verify error:num=18:self signed certificate
verify return:1
depth=0 /CN=nonexistent.gremwell.com/C=BE/O=Gremwell bvba
verify return:1
```

In the first terminal you will see a the result of the test.

```
$ ./sslcaudit
127.0.0.1:58375  sslcert(('nonexistent.gremwell.com', 'SELF')) connected, 2.9s timeout
```

The output says:

- a connection was received from 127.0.0.1:58375
- the connection was handled by sslcert module, using a self-signed certificate with CN=nonexistent.gremwell.com
- SSL connection was established successfully, client did not close for 3s

4.4 Example 2: A client who verifies the chain of trust

Now do the same as above, but use socat instead of openssl. Socat validates server certificates by default and will not connect to an arbitrary peer.

```
$ socat - OPENSSL:localhost:8443
2012/05/01 10:50:50 socat[18692] E SSL_connect(): error:14090086:SSL
routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
```

On the side of sslcaudit we can see:

```
$ ./sslcaudit
```

```
127.0.0.1:52749        sslcert(('nonexistent.gremwell.com', 'SELF'))
tlsv1 alert unknown ca
```

Again:

- a connection was received from 127.0.0.1:527
- the connection was handled by sslcert module, using a self-signed certificate with CN=nonexistent.gremwell.com
- SSL connection setup has failed

4.5 Example 3: A client who validates the chain of trust, but ignores CN mismatch

Let's take a more complicated example of testing a thick client. We will simulate a situation when

- the client runs on a host under our control,
- we have already created a test CA and imported it into the list of trusted CAs on the client host,
- we know that the client originally communicates with 62.213.200.252:443 server

To simulate the client side we will use socat in a loop (doc/example3-client-openssl.sh).

```
$ for _ in `seq 1 4` ; do socat -
OPENSSL:localhost:8443,cafile=/home/abb/certs/sslcaudit-test-cacert.pem ; done
2012/05/01 11:24:37 socat[19110] E SSL_connect(): error:14090086:SSL
routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
2012/05/01 11:24:37 socat[19112] E SSL_connect(): error:14090086:SSL
routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
```

We tell sslcaudit to use the test CA and the certificate of the remote server (doc/example3-server.sh):

```
$ ./sslcaudit --user-ca-cert \
      ~/certs/sslcaudit-test-cacert.pem \
      --user-ca-key ~/certs/sslcaudit-test-cakey.pem \
      --server 62.213.200.252:443
127.0.0.1:58352  sslcert(('nonexistent.gremwell.com', 'SELF'))                tlsv1
alert unknown ca
127.0.0.1:58353  sslcert(('brufeprd1.hackingmachines.com', 'SELF'))           tlsv1
alert unknown ca
127.0.0.1:58354  sslcert(('nonexistent.gremwell.com', 'sslcaudit-test'))      connected,
got nothing in 2.999s
127.0.0.1:58355  sslcert(('brufeprd1.hackingmachines.com', 'sslcaudit-test')) connected,
got nothing in 2.996s
```

As expected, socat rejects the self-signed certificate, but trusts the test CA. We can see that it knows nothing about CN validation, and happily accepts any CN.

4.6 Example 4: A client who verifies both the chain of trust and the CN

We can use curl to simulate the behavior of a proper SSL client, validating both the chain of trust

and the CN. To simplify the simulation we have edited our /etc/hosts to resolve brufeprd1.hackingmachines.com into 127.0.0.1 (not shown here).

Run curl in a loop (doc/example4-client-curl.sh):

```
$ for _ in `seq 1 4` ; do curl --cacert /home/abb/certs/sslcaudit-test-cacert.pem
https://localhost:8443/ ; done
curl: (60) SSL certificate problem, verify that the CA cert is OK. Details:
curl: (60) SSL certificate problem, verify that the CA cert is OK. Details:
curl: (51) SSL: certificate subject name 'nonexistent.gremwell.com' does not match
target host name 'brufeprd1.hackingmachines.com'
curl: (52) Empty reply from server
```

Sslcaudit is invoked exactly like in the previous example (doc/example3-server.sh).

```
$ ./sslcaudit --user-ca-cert /home/abb/certs/sslcaudit-test-cacert.pem --user-ca-key
/home/abb/certs/sslcaudit-test-cakey.pem --server 62.213.200.252:443
127.0.0.1:58470  sslcert(('nonexistent.gremwell.com', 'SELF'))                tlsv1
alert unknown ca
127.0.0.1:58471  sslcert(('brufeprd1.hackingmachines.com', 'SELF'))           tlsv1
alert unknown ca
127.0.0.1:58472  sslcert(('nonexistent.gremwell.com', 'sslcaudit-test'))      connected,
got EOF after 0.000686s
127.0.0.1:58473  sslcert(('brufeprd1.hackingmachines.com', 'sslcaudit-test')) connected,
got 175 octets after 0.000894s
```

As expected, curl has established a connection to sslcaudit only if the server certificate is fully valid.

5 SUPPORTED SYSTEMS

Written in Python, should work on any python2.x. Requires M2Crypto python library (binding to OpenSSL, http://chandlerproject.org/bin/view/Projects/MeTooCrypto).

Developed and tested on Ubuntu Natty 11.04, with stock python-m2crypto-0.20.1-1ubuntu5 package installed. Partially tested on BackTrack 5 R2.

OpenSSL library shipped with recent Linux distributions does not support SSLv2. This does not affect this version of sslcaudit. The next version of sslcaudit will feature protocol level tests and will require OpenSSL library supporting SSLv2.

6 COMMAND LINE PARAMETERS

```
$ ./sslcaudit -h
Usage: sslcaudit [OPTIONS]

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -l LISTEN_ON          Specify IP address and TCP PORT to listen on, in
                        format of [HOST:]PORT
  -m MODULE             Launch specific audit module. For now the only
                        functional module is 'sslcert'. There is also 'dummy'
                        module used for internal testing or as a template code
                        for new modules. By default 'sslcert' is started.
  -d DEBUG_LEVEL        Set debug level. Default is 0, which disables debugging
                        output. Try 1 to enable it.
  -c NCLIENTS           Number of clients to handle before quitting. By
                        default sslcaudit will quit as soon as it gets one
```

```
                        client fully processed.
-N TEST_NAME            Set the name of the test. If specified will appear in
                        the leftmost column in the output.
--user-cn=USER_CN       Set user-specified CN.
--server=SERVER         Where to fetch the server certificate from, in
                        HOST:PORT format.
--user-cert=USER_CERT_FILE
                        Set path to file containing the user-supplied
                        certificate.
--user-key=USER_KEY_FILE
                        Set path to file containing the user-supplied key.
--user-ca-cert=USER_CA_CERT_FILE
                        Set path to file containing certificate for user-
                        supplied CA.
--user-ca-key=USER_CA_KEY_FILE
                        Set path to file containing key for user-supplied CA.
--no-default-cn         Do not use default CN (nonexistent.gremwell.com)
--no-self-signed        Don't try self-signed certificates
--no-user-cert-signed
                        Do not sign server certificates with user-supplied one
```

7 TO BE IMPLEMENTED

- Better report formatting (v1.1)
- Support protocol-level tests (v1.1)
- Save a copy of all keys/certificates used during the test (right now they remain under /tmp) (v1.1)
- Capture logs and relevant packet traces (v1.1)
- Synchronize test execution with external world (>v1.1)
- Support SSL server-side tests to allow end-to-end analysis of client-server communication (>v1.1)
- Optionally run same test several times, to detect random glitches. (>1.1)
- Is there a need for “SSL Client Rating” scheme?
- Embedded HTTP server to automate testing of web browsers [low]
- Allow certificate and private key to be specified in a single file [low]
- Support password-protected private keys [low]

8 REFERENCES

SSL/TLS security - the server side
[WIKI-TLS] http://en.wikipedia.org/wiki/Transport_Layer_Security
[SCANIT-SSL] http://www.scanit.be/uploads/ssl%20security%20in%20be%20-%2003-2008.pdf
[OWASP-TLS] https://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29
[SSL-RATING] https://www.ssllabs.com/projects/rating-guide/index.html
[SSLAUDIT] http://code.google.com/p/sslaudit/
[TLS-RENEG] http://www.g-sec.lu/practicaltls.pdf

SSL/TLS security - the client side

[SSLSNIFF] http://www.thoughtcrime.org/software/sslsniff/
[SSLSTRIP] http://www.thoughtcrime.org/software/sslstrip/
[BH-SSL-STRIP] http://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf
[BH-SSL-TTRUST] https://media.blackhat.com/bh-eu-12/Jarmoc/bh-eu-12-Jarmoc-SSL_TLS_Interception-Slides.pdf
[SSL-TTRUST] http://www.secureworks.com/research/threats/transitive-trust/
[SSLTEST] https://ssltest.offenseindepth.com/
[BS-BEAST] http://www.schneier.com/blog/archives/2011/09/man-in-the-midd_4.html
[OPERA-BEAST] http://my.opera.com/securitygroup/blog/2011/09/28/the-beast-ssl-tls-issue
[OSCP-ATTACK] http://www.thoughtcrime.org/papers/ocsp-attack.pdf

IE5 SSL Spoofing vulnerability
[IE-SSL-CHAIN] http://www.thoughtcrime.org/ie-ssl-chain.txt
[BID-2737] http://www.securityfocus.com/bid/2737
[MS01-027] http://technet.microsoft.com/en-us/security/bulletin/ms01-027

Multiple Vendor Invalid X.509 Certificate Chain Vulnerability
[BID-5410] http://www.securityfocus.com/bid/5410

Apple iOS Data Security Certificate Chain Validation Security Vulnerability
[TWSL2011-007] https://www.trustwave.com/spiderlabs/advisories/TWSL2011-007.txt
[CVE-2011-0228] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-0228

[MARVIN] http://www.gremwell.com/marvin-mitm-tapping-dot1x-links

9 CONTACTS

Sslcaudit is written by Alexandre Bezroutchko, abb@gremwell.com, http://www.gremwell.com/.

10 ABOUT GREMWELL

Gremwell offers security consulting services in the area of penetration testing, ethical hacking, vulnerability assessments and security code and configuration reviews. We are located in the neighbourhood of Brussels, and service clients in Belgium and abroad. Gremwell's consultants have more than 10 years experience in IT security.

Gremwell develops MagicTree - a data management tool for penetration testers.