Pentest Report: Botan Crypto Library, Version 1.11.18

Dr. Juraj Somorovsky, René Korthaus

Juraj.Somorovsky@3curity.de
r.korthaus@sirrix.com

3curity GmbH
Universitätsstraße 150
44801 Bochum
Germany

Sirrix AG security technologies
Im Stadtwald, Geb. D3.2
66123 Saarbrücken
Germany

Contents

1 Summary

3curity and Sirrix AG executed a security evaluation of the Botan library and its main cryptographic functionalities. We concentrated on basic cryptographic methods and side channels allowing one to extract private data. Furthermore, we tested TLS and certificate failures. This report provides a short summary of our findings, dedicated to Botan developers.

2 Methods

During our tests, we used manual source code analysis as well as support of specific tools:

- TLS-Attacker (TLS evaluation tool, currently private development)
- x509test[1]

[1] https://github.com/yymax/x509test

3 Vulnerabilities

In the following, we describe the found vulnerabilities. Every vulnerability contains a security impact score and a reference ID.

3.1 Padding Oracle Attacks – *High*, *#S1*

Padding oracle attacks on the CBC (Cipher Block Chaining) mode of operation were first described by Vaudenay [Vau02]. These attacks belong to the group of adaptive chosen-ciphertext attacks. Thereby, the attacker modifies the ciphertext, sends it to the server, and observes server responses. If a server responds with a different error message by an invalid CBC padding, the attacker can decrypt the ciphertext by sending several oracle queries.

The CBC decryption and unpadding functionality in the TLS record layer is implemented correctly and attempts to thwart timing side channels [AP13]. However, directly after the record data decryption, the implementation evaluates the length of the unpadded data and whether this data has enough length for MAC validation (typically, 20 bytes).[2]

```
if(record_len < mac_pad_iv_size)
    throw Decoding_Error("Record sent with invalid length");
```

If there is not enough data for MAC validation, the server responds with a DECODE_ERROR alert. This alert differentiates from a typical case when an invalid padding occurs.

In order to trigger the DECODE_ERROR alert in a typical case (20 bytes long MAC, 16 bytes long AES block cipher), the decrypted message has to consist of at least 32 bytes (two blocks). This means that the attacker has to know at least 12 bytes to trigger the alert and adopt padding oracle attacks. This is possible in scenarios where the attacker knows parts of the plaintext, for example if the victim uses a browser which typically sends to the website known HTTP headers [DR11].

3.2 Certificate Validation Problems – *Medium*, *#S2*

We evaluated Botan's TLS client functionality by running the Botan TLS command line tool against the x509test test suite. We started both tools using the following command lines[3]:

```
python3 x509test.py www.tls.test -p 4433
```

[2] https://github.com/randombit/botan/blob/master/src/lib/tls/tls_record.cpp#L400

[3] We added a corresponding entry to /etc/hosts matching the www.tls.test DNS name to 127.0.0.1

```
./botan tls_client www.tls.test 4433
```

x509test expects the TLS client under test to abort the TLS handshake for negative tests. We had to modify the Botan TLS client command line tool such that it does this by uncommenting the relevant `throw;` statement[4].

Our evaluation with x509test revealed the following problems.

3.2.1 Name Constraints – *Medium*, *#S2.1*

x509test contains various positive and negative tests for X.509 name constraints (RFC 5280, section 4.2.1.10). Botan already fails the positive test `ValidNameConstraint` and therefore x509test skips the related negative tests `InvalidNameConstraintExclude`, `InvalidNameConstraintPermit`, `InvalidNameConstraintPermitRight` and `InvalidNameConstraintPermitThenExclude`.

3.2.2 Invalid Extended Key Usage – *Low*, *#S2.2*

In the test case `InvalidExtendedKeyUsage`, x509test presents a server certificate that has an unsuitable value in the extended key usage extension (`serverAuth=false`). Botan does not reject this certificate, although it should according to RFC 5280, section 4.2.1.12.

3.2.3 Invalid Key Usage – *Low*, *#S2.3*

In the test case `InvalidKeyUsage`, x509test presents a server certificate that has an unsuitable value in the key usage extension (`cRLSign=true`). Botan does not reject this certificate, although it should according to RFC 5280, section 4.2.1.3.

3.2.4 Invalid Wildcard in CN – *Medium*, *#S2.4*

In the test case `InvalidWildcardLeft`, x509test presents a wildcard certificate that tries to extend its matching effect to its left in the server certificate's common name (CN). In our test setup, it tries to match `www.tls.test` with `*.test`. Botan does not reject this certificate, although it should according to RFC 6125, section 6.4.3.

[4] `https://github.com/randombit/botan/blob/master/src/cmd/credentials.h#L109`

3.2.5 Invalid Wildcard in SAN – *Medium*, *#S2.5*

In the test case `InvalidWildcardLeftAltName`, x509test presents a wildcard certificate that tries to extend its matching effect to its left in the server certificate's X.509v3 subject alternative name (SAN) extension. In our test setup, it tries to match `www.tls.test` with `*.test`. Botan does not reject this certificate, although it should according to RFC 6125, section 6.4.3.

3.2.6 Invalid SAN with valid CN – *Low*, *#S2.6*

In the test case `InvalidNameAltNameWithSubj`, x509test presents a server certificate with an invalid SAN but correct CN. Botan does not reject this certificate, although it should according to RFC 6125, section 6.4.4.

3.2.7 Null-prefixed SAN with valid CN – *Low*, *#S2.7*

In the test case `InvalidNameNullAltNameWithSubj`, x509test presents a server certificate with a null-prefix attack in its SAN but a correct CN. Botan does not reject this certificate, although it should according to RFC 6125, section 6.4.4.

In addition to fixing the aforementioned vulnerabilities, we recommend adding relevant test cases to Botan's test suite and/or integrating x509test into Botan's continuous integration infrastructure to prevent regressions.

3.3 Bleichenbacher Attack on RSA-PKCS#1 v1.5 (Timing) – *Low*, *#S3*

Bleichenbacher's million message attack is an adaptive chosen-ciphertext attack, which allows an attacker to decrypt confidential messages without knowing the RSA private key [Ble98]. A prerequisite for this attack is an existence of an oracle that accepts an encrypted RSA-PKCS#1 v1.5 message, decrypts it, and responds with *true* or *false* based on the message structure validity. This oracle can be constructed from a server responding with different error messages, or by observing timing behavior of the server responses.

Our analysis of the TLS implementation revealed that there is *no* possibility to obtain a direct oracle based on different error messages. However, our source code analysis showed that the RSA-PKCS#1 v1.5 validation is not timing constant. It contains different branches, or even exceptions are thrown.[5] This could potentially lead to timing attacks, as described by Meyer et al. [MSW+14]. We thus encourage the developers to fix this problem

[5] `https://github.com/randombit/botan/blob/master/src/lib/pk_pad/eme_pkcs1/eme_pkcs.cpp#L40`

and make the validation timing constant. An example of timing constant implementation could be seen in the OpenSSL code.[6]

[6] https://github.com/openssl/openssl/blob/master/crypto/rsa/rsa_pk1.c#L182

4 Recommendations

In the following, we give recommendations for fixes to various problems discovered by our evaluation. These recommendations do not patch vulnerabilities leading to direct exploits, but they are recommended to improve the security of the analyzed library.

4.1 RSA Blinding (Missing Re-initialization) – #AE1

In 2003 Brumley and Boneh described practical timing attacks against RSA-based crypto systems [BB03]. The goal of these attacks was to extract private RSA keys, based on the timing measurements of RSA exponentiations (during decryption and signature creation). As a countermeasure against these attacks a method called RSA blinding is used.

The countermeasure works as follows. Before the first RSA operation is executed, values A and A_{inv} are generated, so that $A \cdot A_{inv} \equiv 1 \text{mod} N$. The decryptor then multiplies the ciphertext with A. After decryption, it obtains a correct plaintext by applying a multiplication with A_{inv}.

Values A and A_{inv} must be updated after each private key operation to provide sufficient resistance against side channel attacks. This is done by computing $A' \equiv A \cdot A$ and $A'_{inv} \equiv A_{inv} \cdot A_{inv}$. In addition, these values should be newly generated after several usages (for example, in OpenSSL after 32 usages).[7]

In Botan, the blinding reinitialization is missing.[8] We encourage the developers to add this functionality.

4.2 Diffie Hellman Secure Parameter Generation – #AE2

In order to generate secure cryptographic parameters for cyclic groups over Z_p, typically "safe primes" are used. In that case, p and $(p-1)/2 = q$ are both prime. Generator g has an order of $(p-1)$.

Botan always uses $g = 2$ as a generator. It does not verify whether this generator is of order $(p-1)$. A simple check should be provided, whether $g^q \equiv 1 mod q$. If this is the case, the order of the generator is $(p-1)/2$. Even though this is not security critical, we encourage the developers to implement a check, whether g has a correct order.

Additionally, it is possible to generate strong primes of a 512 bit length. The number of bits should be higher for security critical applications.

[7] https://github.com/openssl/openssl/blob/master/crypto/bn/bn_blind.c#L196

[8] http://botan.randombit.net/doxygen/blinding_8cpp_source.html

4.3 Certificate Validation Problems – #AE3

4.3.1 Intermediate CA Loop Hang – #AE3.1

In the test case `InvalidIntCALoop`, x509test presents a server certificate along with three intermediate CA certificates, where the intermediate CA certificates form a loop `lev3 -> lev1 -> lev2 -> lev3` (`->` denotes 'certifies'). When parsing the certificate chain, Botan hangs, although it should reject the certificate chain according to RFC 5280, section 6.1.4.

References

[AP13] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. *2013 IEEE Symposium on Security and Privacy*, 0:526–540, 2013. `http://www.isg.rhul.ac.uk/tls/TLStiming.pdf`.

[BB03] David Brumley and Dan Boneh. Remote timing attacks are practical. In *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*, SSYM'03. USENIX Association, June 2003.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1998.

[DR11] Thai Duong and Juliano Rizzo. Here come the ⊕ Ninjas. Unpublished manuscript, 2011.

[MSW+14] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks. In *23rd USENIX Security Symposium, San Diego, USA*, August 2014.

[Vau02] Serge Vaudenay. Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS... In *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, April 2002.