Report on Findings for the Certication Path Validation Test Tool

Version 1.1, 16.05.2018

Armin Cordel[1], Heike Hagemeier[1], Evangelos Karatsiolis[2], Falko Strenzke[3]

[1]Federal Office for Information Security (BSI), Bonn, Germany
{heike.hagemeier | armin.cordel}@bsi.bund.de
[2]MTG AG, Darmstadt, Germany
ekaratsiolis@mtg.de
[3]cryptosource GmbH, Darmstadt, Germany
fstrenzke@cryptosource.de

Abstract. *In this report we present the findings of security vulnerabilities and compatibility issues that were discovered by applying the certification path validation test tool (CPT) to various test subjects. The CPT is a tool featuring the generation of valid and invalid X.509 certification paths and is equipped with a default test suite that addresses the most important aspects of the RFC 5280 standard for the certification path validation.*

1 Introduction

In a project contracted out by the German Federal Office for Information Security (BSI) to two companies, the Certification Path Validation Test Tool was created. This open source tool is available at the tool's main web page[1]. The tool receives as input a set of test specifications in an XML format. Each test case specifies a set of X.509 certificates and potentially also certificate revocation lists (CRLs) with specific properties and interdependencies. The CPT then creates the test data, i.e. certificates and CRLs, for each test case, and additionally the information whether the resulting certification path is valid or invalid. The produced test data can then be used to verify implementations of the X.509 certification path validation, as it is for instance used in the TLS protocol. The CPT is shipped with a test suite which tests numerous aspects of the certification path validation as specified in the relevant standard, namely RFC 5280. This document reports on the vulnerabilities that were identified in various test subjects that were tested using the default test suite of the CPT.

2 Test Subjects

In the course of the project, the following ten test subjects were tested against the default test suite of the CPT:

1. OpenSSL[2] version 1.1.0e,
2. Botan[3] in a early version of release 2.2.0 (git commit 5b2fe4a6d4dfdb28af364eec86a407327e64d1d7),
3. mbedTLS[4] version 2.4.2,
4. Bouncy Castle[5] version 1.57,
5. OpenJDK version[6] 1.8.0_121,

1 www.bsi.bund.de/CPT
2 https://www.openssl.org/
3 https://botan.randombit.net/
4 https://tls.mbed.org/
5 https://www.bouncycastle.org/
6 http://openjdk.java.net/

6. Apache HTTP Server[7] version 2.4.18 (Ubuntu),
7. Firefox[8] version 55.0.2,
8. strongSwan[9] version 5.5.3,
9. KMail[10] version 5.1.3 with Kleopatra[11] version 2.2.0 (using gpgsm(GnuPG) version 2.1.11),
10. OpenVPN[12] version 2.4.0 (using OpenSSL version 1.0.1t).

The first five are cryptography libraries. For the test execution on the libraries, for each test case the certification path validation was executed on a set of certificates, containing at least one trust anchor certificate, exactly one target certificate which is validated, a potentially empty set of untrusted intermediate certificates, and potentially also a set of certificate revocation lists (CRLs). The certificates and CRLs were provided to the respective library's routine for the certification path validation. The validation routines are implemented by two tools, one for the JAVA libraries[13], and one for the C/C++ libraries[14].

The remaining five test subjects are applications. These were tested as follows: KMail was tested using the CPT's built-in functionality for the creation of signed e-mails. The Apache web server and the Firefox browser were tested using the TLS and browser test tool extensions[15]. The certification path validation of the strongSwan IPsec implementation was tested using a modified strongSwan version together with a script for automated test execution[15]. OpenVPN was tested using a script for automated test execution.

Note that OpenJDK is not affected by any vulnerabilities or compatibility issues determined in our project and thus is completely absent from the issue lists.

3 Insecure Configurations

The tests CERT_PATH_ALGO_STRENGTH_01 and CERT_PATH_ALGO_STRENGTH_02 of the CPT's default test suite test the acceptance of MD5 as the hash signature algorithm for the target certificate and intermediate certificates, respectively. In the default configuration, Bouncy Castle, OpenSSL, Apache, KMail, OpenVPN, and strongSwan accepted such certificates. Configurations options to disable MD5 as the signature algorithm are available at least in most of the tested implementations.

OpenVPN seems only to be able to verify the revocation status of the target certificate, but not that of intermediate certificates. A configuration option to enforce the checking of the revocation status also of intermediate certificates is not apparent in this application. The tests were executed using the option crl-verify. OpenVPN offers additionally the option capath to provide the location of valid CA certificates, which, however, was not examined in the tests.

4 Identified Vulnerabilities

In this section we present those issues identified by the application of the CPT's default test suite on the test subjects that imply potential security risks. The mapping of the vulnerabilities to the test subjects can be read from Table 2.

The naming of the identified compatibility issues follows the same pattern as that of the compatibility issues presented in Section 5. Specifically, each name of a security or vulnerability issue is derived from the test case name from the CPT's default test suite that revealed the issue. The name is built according

7 https://httpd.apache.org/
8 https://www.mozilla.org/de/firefox/
9 https://www.strongswan.org/
10 https://www.kde.org/applications/internet/kmail/
11 https://www.kde.org/applications/utilities/kleopatra/
12 https://openvpn.net/
13 https://github.com/MTG-AG/cpt/
14 https://github.com/cryptosource-GmbH/cpt-native-lib-test
15 https://github.com/cryptosource-GmbH/cpt-add-test-tools

to the pattern <type-letter><test module code>_<test case number>. Here, <type-letter> is either “V” for “vulnerability” or “C” for compatibility issue, the <test case number> is the number contained as the last part of the test case name. The only exception is CCR_00, which describes a general issue that cannot be associated with any specific CRL test. Lastly, the <test module code> is derived according to the mapping shown in Table 1.

Test module	Test module code	Description
COMMON	CM	Covers general aspects of certificates
CRL	CR	Covers aspects regarding revocation lists.
EXT	EX	Covers aspects of certificate extensions.
IPSEC	IP	Covers aspects of IPSec.

Table 1: Mapping of test module names to issue names.

Vulnerability	Description	Botan	Bouncy Castle	mbedTLS	OpenSSL	Apache	Firefox	KMail	OpenVPN	strongSwan
VCM_08	acceptance of expired intermediate or target certificates							E		
VCM_11	acceptance of an invalid certificate version		E		E				E	
VEX_06	acceptance of intermediate certificate without basic constraints extension					E			E	
VEX_11	acceptance of intermediate certificate without `keyCertSign` key usage									E
VIP_04	acceptance of target certificate with Key Usage extension only featuring `keyAgreement` key usage									E
VCR_01	ignoring lack of matching CRL			E						
VCR_06	acceptance of unknown critical CRL extensions	E		E						
VCR_08	acceptance of not yet valid CRLs									E
VCR_13	acceptance of mismatching certificate's CRL-DP and CRL's IDP	E		E						E
VCR_15	ignoring lack of matching CRL for an intermediate CA									E

Table 2: Overview of the vulnerabilities identified in the test subjects using the CPT's default test suite. The letter “E” indicates that the respective test subject is affected by the issue.

4.1 VCM_08 – acceptance of expired intermediate or target certificates

The affected implementation fails to display whether an intermediate or target certificate has expired. This is important since expired certificates are allowed to be removed from the CRL. Not displaying

whether a certificate has expired may result in a missed revocation. In validity models other than the one used in the internet, like the chain or the hybrid model, accepting expired certificates is allowed. However, revocation in these models is also handled in a different manner.

4.2 VCM_11 – acceptance of an invalid certificate version

A number of tested implementations accept X.509 certificates which indicate the value "5" in their X.509 version field, even though currently the highest version is 3. Implementations with this behavior may exhibit security vulnerabilities in the future if newer versions of the X.509 standard with non-compatible processing rules are issued.

4.3 VEX_06 – acceptance of intermediate certificate without basic constraints extension

According to RFC 5280, any intermediate CA certificate must contain the basic constraints extension with the field cA having the content true. The affected implementations do not check this requirement.

For both affected applications, this is due to using OpenSSL versions which implement a more complex check for the classification of certificates as CA certificates.[16] One such rule obviously seems to allow CA certificates without the basic constraints extension if they feature a key usage extension with the `keyCertSign` usage set.

4.4 VEX_11 – acceptance of intermediate certificate without KeyCertSign Key Usage

RFC 5280 specifies that an intermediate CA certificate which contains the key usage certificate extension, must specify the key usage purpose `keyCertSign` therein. The affected implementation fails to check this.

4.5 VIP_04 – acceptance of target certificate with Key Usage extension only featuring keyAgreement key usage

According to RFC 4945, a certificate used in the IKEv1 or IKEv2 key exchange in IPsec must at least feature the key usages `digitalSignature` or `nonRepudiation`. The affected implementation fails to check this requirement when the test certificate contains a Key Usage extension marked as critical and featuring only the `keyAgreement` key usage purpose.

4.6 VCR_01 – ignoring lack of matching CRL

mbedTLS takes the following conscious decision regarding the CRL processing implemented by the function `mbedtls_x509_crt_verify()`, as stated in the corresponding API documentation: "It is your responsibility to provide up-to-date CRLs for all trusted CAs. If no CRL is provided for the CA that was used to sign the certificate, CRL verification is skipped silently, that is *without* setting any flag." This leads to a direct vulnerability regarding revocation checking: CRLs, being authenticated by their signature, are always downloaded over otherwise insecure connections[17]. If an attacker can manipulate a downloaded CRL by modifying for instance the issuer name therein, then mbedTLS will silently skip the revocation check. Thus, there is indeed no way of enforcing revocation checks with mbedTLS.

On our vulnerability report the mbedTLS team confirmed that this behavior is a conscious decision. However, they consider implementing a switch for the enforcement of revocation checks in a future version of the library.

16 https://www.openssl.org/docs/manmaster/man1/x509.html, accessed on 2018-05-14.

17 Using any certificate-based approach for this purpose such as TLS would only again lead to the requirement of further revocation checks – resulting in an infinite regression.

4.7 VCR_06 – acceptance of unknown critical CRL extensions

Both Botan and mbedTLS fail to reject CRLs that contain unknown critical CRL extensions. This can lead to a security problem since such extensions may arbitrarily modify the way such CRLs have to be processed. Both libraries are fixing this vulnerability in future versions.

4.8 VCR_08 – acceptance of not yet valid CRLs

strongSwan accepts CRLs which are, according to their `thisUpdate` field, not yet valid. This can be a potential security problem if clocks on two systems differ significantly. If the CRL's validity is not checked, this can lead to an unnoticed certificate revocation since expired certificates can be removed from the CRL.

Specifically, a revoked certificate can be removed from a CRL once the certificate's `notAfter` date has been reached. However, note that the first CRL issued beyond that date still has to list the certificate as revoked.

Assuming a second system in which CRLs are potentially issued very frequently, this can lead to the situation that the CRL issuer sees a revoked certificate as expired and thus removes it from a new CRL. Another system with a lagging clock verifies the same certificate, but still sees it as valid. However, due to the missing check of the `thisUpdate` date of the newly issued CRL, which lies in the future for the second system's clock, it sees it as unrevoked and accepts it.

4.9 VCR_13 – acceptance of mismatching certificate's CRL-DP and CRL's IDP

The test CERT_PATH_CRL_13 of the CPT's default test suite tests a subtle point of the CRL validation. If a certificate contains the CRL Distribution Point (CRL-DP) certificate extension and a CRL, that shall be used to check that certificate's revocation status, contains an Issuing Distribution Point (IDP) CRL, then the distribution point names specified in both extensions have to match. Otherwise, it must be inferred that the certificate is specifying a different CRL distribution point than that which the CRL was retrieved from. Failure to perform this check may lead to potential security problems in certain PKI configurations. With four affected test subjects, this is the most widespread security problem of the X.509 path validation identified in our project.

4.10 VCR_15 – ignoring lack of matching CRL for an intermediate CA

When „strictcrlpolicy=yes“ is configured for strongSwan, it verifies also the revocation status of intermediate CA certificates. However, if for an intermediate CA certificate a CRL is missing completely, then the revocation check for this certificate is silently skipped. This is issue is analogous to VCR_01, except that only intermediate certificates are affected.

5 Compatibility Issues

Table 3 gives an overview of the compatibility issues that were found by applying the CPT's default test suite to the test subjects. In the following, the identified issues are explained in detail.

Comp. Issue	Description	Botan	Bouncy Castle	mbedTLS	OpenSSL	Apache	Firefox	KMail	OpenVPN	strongSwan
CCR_00	problems with CRL cache							E		
CCM_01	too restrictive handling of path length	E								

CCM_13	too restrictive handling of path length with self-issued certificates		E	E			E	E		E
CCM_14	performing non-exhaustive path search	E		E	E	–	–	–	–	–

Table 3: Overview of the compatibility issues identified in the test subjects using the CPT's default test suite. The letter "E" indicates that the respective test subject is affected by the issue.

5.1 CCR_00 -problems with CRL cache

KMail caches CRL, with the help of dirmngr, based on the issuer DN. Since the default setup of the CRL test cases uses the same issuer but different distribution points it was not possible to obtain reliable results for basically all CRL related tests for this test subject.

5.2 CCM_01 – too restrictive handling of path length in Botan

The initially tested Botan version was 2.1.0. This version exhibited the problem, that the path length constraint specified in the basic constraints certificate extension was interpreted too restrictively. Thus, Botan rejected correct certificated chains. As this disrupted many tests in the CPT's default test suite, a consequence the above specified pre-release of version 2.2.0 was used in further testing.

5.3 CCM_13 – too restrictive handling of path length with self-issued certificates

A number of tested implementations do not account correctly for self-issued certificates within the certificate path. A self-issued certificate is one in which issuer and subject DN are equal. Such certificates can be used by CAs to cross-certify different keys of their own. RFC 5280 mandates that such certificates contained within the certification path are not accounted for the path length constraint specified in the basic constraints certificate extension. The affected implementations do not realize this special treatment of self-issued certificates and thus perform too restrictive checks for certain certification paths.

5.4 CCM_14 – performing non-exhaustive path search

The test of the cryptography libraries includes the step of the path construction, i.e. the building of the correct chain from the target certificate up to a trust anchor using any number of untrusted intermediate certificates from a supplied certificate pool. One test verifies whether during this path construction, the tested implementation performs an exhaustive search for all possible certification paths until a valid path is found. This is realized by including untrusted intermediate certificates within the certificate pool which allow both to build a valid path from the target certificate to the trust anchor as well as an invalid path. This is realized by including a valid intermediate certificate which correctly links the target certificate to the trust anchor, as well as an invalid variant of that certificate, which features the same subject DN and public key but features itself an issuer DN which cannot be resolved. The intermediate certificates are added to the certificate pool in a random order and the test is executed multiple times. Thus in a certain number of executions the invalid intermediate certificate will be checked first, and in the remaining cases the valid intermediate certificate.

An implementation which does not features exhaustive path search fails to find a path once it tries to complete the chain with the invalid intermediate certificate first. Such implementations are detected by observing that the validation result varies between the different test executions with randomized order of the intermediate certificates.

Note that this behavior is sometimes deliberately realized. For instance the OpenSSL team confirmed this to us for their library.

This behavior of an implementation is not a security vulnerability directly, but it can lead to denial-of-service problems, if an attacker is able to inject invalid intermediate certificates into an application's certificate cache.

This issue does not apply to the applications tested within our project, since all them use ordered certificates and thus do not implement the certification path construction.

6 Responsible Disclosure

The maintainers of the cryptography libraries and applications that were used as test subjects in our project have all been informed about the determined vulnerabilities and compatibility issues more than 6 months prior to their publication.

This report was sent to the maintainers prior to its publication and they were given two weeks time for comments.

7 Conclusion

The CPT's default test suite was systematically derived by iterating through the requirements from RFC 5280 and addresses various aspects of certificate and CRL validation. The application of the default test suite to ten selected test subjects revealed numerous security and compatibility issues of varying severity. The test results for the certificate validation did not exhibit any severe vulnerabilities, which can be directly exploited in general settings. However, a number of minor security problems and relevant compatibility issues were identified in a number of test subjects.

The tests covering the CRL validation revealed much more severe security issues, showing that this functional aspect deserves greater attention by implementers.