hard.

***CVE-2014-8275 – Impact analysis on WR SSL and test approach***

**Prepared For RICOH**

**CONFIDENTIAL**

Version: 1.00

Date :May 8, 2015

Version: 1.00

Date: May 8, 2015

Wind River Systems, Inc.

9855 Scranton Road, Building 5

San Diego, CA92121

858-824-3100 phone

858-824-3198 fax

[www.windriver.com](http://www.windriver.com)

Copyright

Unless otherwise agreed in writing, all copyright and intellectual property rights embodied in this document are and shall remain the property of Wind River Systems, Inc. This document is provided solely for the purposes of evaluating the work proposed and no other rights whatsoever to use the information herein are granted. The contents of this document may not be disclosed to any third party without the prior written consent of Wind River Systems, Inc.

Trademarks

Wind River, the Wind River logo, Tornado, and VxWorks are registered trademarks of Wind River Systems, Inc. Any third party trademarks referenced are the property of their respective owners. For further information regarding Wind River trademarks, please see <http://www.windriver.com/company/terms/trademark.html>.

| Revision History | | | |
| --- | --- | --- | --- |
| Date | Version | By | Description of Change |
| May 8, 2015 | 1.00 | Wind River | Official Release. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Overview 5](#_Toc418859252)

[1.1 Purpose and Scope 5](#_Toc418859253)

[1.2 Applicable Documents 5](#_Toc418859254)

[1.3 Glossary 5](#_Toc418859255)

[2 Impact analysis and Test Approach 6](#_Toc418859256)

[2.1 Vulnerability Summary 6](#_Toc418859257)

[2.2 Details of Vulnerability & Method of Exploit 6](#_Toc418859258)

[2.2.1 Background 6](#_Toc418859259)

[2.2.2 Method of Exploit 7](#_Toc418859260)

[2.3 Implementation Behavior – Openssl & Wind River SSL 7](#_Toc418859261)

[2.4 Code changes & Impact of OpenSSL Fix 8](#_Toc418859262)

[2.5 Required Actions on WR-SSL 11](#_Toc418859263)

[2.5.1 Code Changes 11](#_Toc418859264)

[2.5.2 Test/verification Approach & Results 11](#_Toc418859265)

[2.5.3 Development and Test Environment – HOWTO 12](#_Toc418859266)

[2.6 Summary/Conclusion 12](#_Toc418859267)

[2.7 Attachments 12](#_Toc418859268)

# Overview

## Purpose and Scope

The Purpose of this Document is to explain details of impact analysis carried out on Wind River SSL stack used RICOH for security vulnerabilities as described in CVE-2014-8275.

Document also includes details of fix released by OpenSSL community, applicability and methods of test and verification of same under Wind River SSL.

## Applicable Documents

The following documents are referenced within:

| No. | Document | Version | Scope |
| --- | --- | --- | --- |
| 1. 1 | CVE-2014-8275\_Test\_Report1 | 0.01 | Test report of WRSSL rejects non-zero unused bit signature. |
| 1. 2 | CVE-2014-8275\_Test\_Report2 | 0.01 | Test report of WRSSL mismatch algorithm identifier in certificate. |
| 1. 3 | CVE-2014-8275\_Test\_Report3 | 0.01 | Test report of WRSSL rejects non-DER variation. |

## Glossary

|  |  |
| --- | --- |
| Term | Definition |
| WR | Wind River |
| SSL | Secure Sockets Layer |
| TLS | Transport Layer Security |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Impact analysis and Test Approach

## Vulnerability Summary

OpenSSL accepts several non-DER-variations of certificate signature algorithm and signature encodings. OpenSSL also does not enforce a match between the signature algorithm between the signed and unsigned portions of the certificate. By modifying the contents of the signature algorithm or the encoding of the signature, it is possibleto change the certificate's fingerprint.

This does not allow an attacker to forge certificates, and does not affect certificate verification or OpenSSL servers/clients in any other way. It also does not affect common revocation mechanisms. Only custom applications that rely on the uniqueness of the fingerprint (e.g. certificate blacklists) may be affected.

Details of the vulnerability are described in following section.

## Details of Vulnerability & Method of Exploit

### Background

During the X.509 certificate signing process, entire content of the certificate is not signed. Details are below.

X.509 v3 certificates contents are 3 parts.

       tbsCertificate

       signatureAlgorithm

       signatureValue

Out of above, DER encoded tbsCertificate is what is signed. This part includes,

       Subject’s distinguished Name

       Subject’s public key

       and Other attributes.

(RFC5280 chapter 4.1.2 describes details of tbsCertificate).

However, a certificate fingerprint is SHA-1 hash of the ***entire*** certificate in DER form. The OpenSSL command “*openssl sha1 <certificate-filename>*” can be used to find out fingerprint of the certificate.

Hence, even if some content of the certificate that is not included in the signed part is modified, signature is not affected but fingerprint is.

This can be a security loophole. OpenSSL made code changes to fix such cases. There are additional scenarios as well such as various cases including garbage after signatureand use of BER or invalid ASN.1 INTEGERs(negative or with leading zeroes).various cases including garbage after signature

(thanks to AnttiKarjalainen and TuomoUntinen from the Codenomicon CROSS

program for discovering this case) and use of BER or invalid ASN.1 INTEGERs

(negative or with leading zeroes).

### Method of Exploit

An attacker can carefully hand-edit contents of the certificate or use a customized SSL program (client or server) to make a variant of an originally blacklisted certificate. The SSL peer processing such a certificate may receive a favorable response from a certificate blacklist database as the received certificate variant generates a different fingerprint.

Certificates can be modified in following ways.

Non-zero unused bits in the BITSTRING encoding of signature

Different encoding of Signature

Non-matching values of SigatureAlgorithm

## Implementation Behavior – Openssl& Wind River SSL

This section explains implementation behavior for following 3 cases.

1. Signatures with non-zero unused bits

OpenSSL(1.0.1 branch 1.0.1j and before, 1.0.0 branch 1.0.0o and before, 0.9.8 branch 0.9.8zc and before) verifies a certificate using function X509\_verify(). This can be called from various contexts where OpenSSL is required to verify a given certificate. This function invokes ASN1\_item\_verify() further.

ASN1\_item\_verify currently does not check for signatures with unused bits. This may resultOpenSSL to allow an invalid certificate to be accepted.

Note: The behavior is same in case of SSLeay also. (when SSLEAY\_MACROS are defined. In such case X509\_verify() will invoke ASN1\_verify() function).

WR-SSL behavior is same as above.

1. Not matching Signature algorithm names in certificate

As mentioned in previous section, function X509\_verify() is invoked to verify a certificate under various situations where it is required. The function also does not check if the Algorithmname in the signed and unsigned parts of the certificates are matching or not. X.509 certificates include AlgorithmName in the tbsCertificate part (included in signing) and signatureAlgorithm that is not included in signing.

Following diagram explains this scenario.

|  |  |
| --- | --- |
| X509 |  |
| X509\_CINF \*cert\_info | |
|  | ver |
|  | serial |
|  | X509\_ALGOR \*signature |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X509\_ALGOR \*sig\_alg | |
| .. |  |
| .. |  |
|  |  |
|  |  |
|  |  |

WR-SSL behavior is same as above.

1. Invalid or non-DER signature in certificate

Functions DSA\_verify or ECDSA\_verify functions are invoked to verify a DSA or ECDSA signature. This is called when SSL server is processing a CertificateVerify from client or client receiving ServerKeyExchange message.

In both above cases, DSA\_verify/ECDSA\_verify functions currently do NOT ensure that signature is indeed in DER format and not any other. In order to make sure that fingerprint of the certificate will not have problems, it is required that received signature matches the DER encoded format every byte.

WR-SSL behavior is same as above except that WR-SSL does not support ECDSA.

## Code changes & Impact of OpenSSL Fix

OpenSSL implements following code changes to fix the issue.

1. Reject signatures with non-zero unused bits.

If the BIT STRING containing the signature has non zero unused bits rejectthe signature. All current signature algorithms require zero unused bits.

~/target/src/security/certificate\_support/crypto/asn1/a\_verify.c, ASN1\_verify()

|  |  |
| --- | --- |
| Code Change | Explanation |
| ASN1err(ASN1\_F\_ASN1\_VERIFY,ASN1\_R\_UNKNOWN\_MESSAGE\_DIGEST\_ALGORITHM); |  |
| goto err; |  |
| } |  |
| + |  |
| + if (signature->type == V\_ASN1\_BIT\_STRING && signature->flags & 0x7) | Checking if signature BITSTRING containing unused bits |
| + { |  |
| + ASN1err(ASN1\_F\_ASN1\_VERIFY, ASN1\_R\_INVALID\_BIT\_STRING\_BITS\_LEFT); |  |
| + goto err; | Give error for the same above |
| + } |  |
|  |  |
| inl=i2d(data,NULL); |  |
| buf\_in=OPENSSL\_malloc((unsignedint)inl); |  |
| @@ -150,6 +156,12 @@ int ASN1\_item\_verify(const ASN1\_ITEM \*it, X509\_ALGOR \*a, |  |
| return -1; |  |
| } |  |
|  |  |
| + if (signature->type == V\_ASN1\_BIT\_STRING && signature->flags & 0x7) | Checking if signature BITSTRING containing unused bits. |
| + { |  |
| + ASN1err(ASN1\_F\_ASN1\_VERIFY, ASN1\_R\_INVALID\_BIT\_STRING\_BITS\_LEFT); |  |
| + return -1; | Return -1 error for the same |
| + } |  |
| + |  |
| EVP\_MD\_CTX\_init(&ctx); |  |
|  |  |
| /\* Convert signature OID into digest and public key OIDs \*/ |  |

2. Check certificate algorithm consistency.

Check the AlgorithmIdentifier inside TBS matches the one in thecertificate signature. NB: this will result in signature failureerrors for some broken certificates.

~\target\usr\src\security\certificate\_support\x509\x\_all.c, X509\_verify()

|  |  |
| --- | --- |
| Code Change | Explanation |
| intX509\_verify(X509 \*a, EVP\_PKEY \*r) |  |
| { |  |
| + if (X509\_ALGOR\_cmp(a->sig\_alg, a->cert\_info->signature)) | Compare algorithm identifier in the certificate(one is in signature and other one is in non-signed part.) |
| + return0; | If mismatch , return 0. |
| return(ASN1\_item\_verify(ASN1\_ITEM\_rptr(X509\_CINF),a->sig\_alg, |  |
| a->signature,a->cert\_info,r)); |  |
| } |  |
|  |  |

3. Check DSA/ECDSA signatures use DER.

Re-encode DSA/ECDSA signatures and compare with the original receivedsignature. Return an error if there is a mismatch.

~/target/src/security/compatibility/dsa/dsa\_vrf.c, DSA\_verify()

|  |  |
| --- | --- |
| Code Change | Explanation |
| constunsignedchar \*sigbuf, intsiglen, DSA \*dsa) |  |
| { |  |
| DSA\_SIG \*s; |  |
| + constunsignedchar \*p = sigbuf; |  |
| + unsignedchar \*der = NULL; |  |
| + intderlen = -1; |  |
| int ret=-1; |  |
|  |  |
| s = DSA\_SIG\_new(); |  |
| if (s == NULL) return(ret); |  |
| - if (d2i\_DSA\_SIG(&s,&sigbuf,siglen) == NULL) goto err; |  |
| + if (d2i\_DSA\_SIG(&s,&p,siglen) == NULL) goto err; | Re-encode the signature in the sigbuf and stored in s. |
| + /\* Ensure signature uses DER and doesn't have trailing garbage \*/ |  |
| + derlen = i2d\_DSA\_SIG(s, &der); | Decode the signature into der. |
| + if (derlen != siglen || memcmp(sigbuf, der, derlen)) | Compare both signatures, they are using DER encoding and does not have trailing garbage |
| + goto err; | goto error if mismatch |
| ret=DSA\_do\_verify(dgst,dgst\_len,s,dsa); |  |
| err: |  |
| + if (derlen>0) |  |
| + { |  |
| + OPENSSL\_cleanse(der, derlen); |  |
| + OPENSSL\_free(der); |  |
| + } |  |
| DSA\_SIG\_free(s); |  |
| return(ret); |  |
| } |  |

## Required Actions on WR-SSL

### Code Changes

Code changes are applicable to Wind River SSL version.

### Test/verification Approach& Results

1. Fix for unused bits in certificate signature BITSTRING -- Reject signatures with non zero unused bits.

Modify the OpenSSL server program to alter the DER encoding of signature to include non-zero unused bits. Let the original certificate be C and modified one be C’. C and C’ both should be accepted by OpenSSL client without fixes, but C and C’ will have different fingerprints.

After the fix is applied, the fingerprints should still be different but C’ should not be accepted by OpenSSL client.

Result is error message is displayed after C’ is processed. Since the error return from the ASN1\_item\_verify() is not propagated by the calling function internal\_verify() SSL handshake continues and gets completed successfully. Report1 includes error messages on the vxWorks console.

example screenshot:-

creating ssl context

calling SSL\_new

calling init\_client

CONNECTED(00000004)

func: ASN1\_item\_verify line: 147 ASN1\_err:ASN1\_R\_INVALID\_BIT\_STRING\_BITS\_LEFT

func: internal\_verify line: 737 X509\_verify\_err: Collected return value is less than 0, Action: Ignore

1. Check certificate algorithm consistency.

As mentioned above, signatureAlgorithm part of the certificate is not included in “to be signed part” (i.e. tbsCertificate). The field signatureAlgorithm contains one ASN.1 structure ‘AlgorithmIdentifier’ that describes algorithm and associated parameters. There is also “signature” field inside tbsCertificate that contains an alogorithm identifier.  Let the first one be AI\_CERT and second one AI\_TBS.

Let original certificate be C. Modify C to create C’ in such a way that AI\_CERT is different from AI\_TBS. Certificate C and C’ should be accepted by OPENSSL client without fix, C’ should be rejected by OPENSSL client with fix. Again fingerprint shall be different for C and C’.

Result is error message is displayed after C’ is processed. Since the error return from the X509\_verify() is not propagated by the calling function internal\_verify() SSL handshake continues and gets completed successfully. Report2 includes error messages on the vxWorks console.

example screenshot:-

depth=1 /C=IN/ST=karnataka/L=bangalore/O=NM/OU=NM/CN=chetan/emailAddress=nm@nm.com

verify return:1

func: X509\_verify line:72 X509\_err: AlgorithmIdentifier Mismatch

func: internal\_verify line: 765 X509\_verify\_err: Collected return value is less than 0, Action: Ignore

1. Check DSA/ECDSA signatures use DER

Fix is to re-encode the signature data to DER format, compare original and newly generated signatures and ensure that there is no mismatch.

To test, create two certificates – one with DER encoded signature and other with signature encoding other than DER. Let these certificates be CERT\_DSA\_DER and CERT\_DSA\_NOTDER. OpenSSL without fix will accept both but version with fix will reject CERT\_DSA\_NOTDER. Report3 includes error messages on the vxWorks console.

example screenshot:-

verify return:1

func: DSA\_verify line: 114 DSA\_err: non-DER variation

ERROR

verify error:certificate signature failure

shutting down SS

## Summary/Conclusion

It has been found that current version of WR-SSL used by customer is vulnerable to the threat described in CVE-2014-8275, and code changes are applicable. Required code changes are made and tests are performed to confirm that vulnerability is not applicable after applying code changes.

## Attachments

1. Report 1 – WRSSL output error for non-zero un used bit.
2. Report 2 – WRSSL output error for Algorithm mismatch.
3. Report 3 – WRSSL output error for non-DER variation.