Tampere University

Santeri Röning

OPENSSL 1.1.1 UPDATE HISTORY

COMP.SEC.300

:

May 2020

ABSTRACT

Santeri Röning: OpenSSL 1.1.1 update history
Tampere University
May 2020

This work will be a review of the critical changes made to the OpenSSL project during XXX-XXX

https://www.openssl.org/news/vulnerabilities-1.1.1.html

Keywords: The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

CONTENTS

1. INTRODUCTION

OpenSSL is a open-source software library designed to be a robust, commercial-grade, full-featured toolkit for general-purpose cryptography and secure communications. The OpenSSL project has been ongoing since 1998 in order to provide a free set of encryption tools for the internet and the code that it uses. [1] The project is controlled by OpenSSL management committee, who represent the official voice of this project. [2]

As all highly used software OpenSSL too has had their own share of vulnerabilities. In this paper I will focus on the OpenSSL version 1.1.1 first released on the 11th of September 2018. The vulnerabilities and their fixes can be found at the OpenSSL's page `https://www.openssl.org/news/vulnerabilities-1.1.1.html` where each vulnerability is classified by their severity, time these were found and fixed. The severity scale is from lowest to highest - Low, Moderate, High and Critical severity. In the timescale from the beginning of this release to the time of writing this paper OpenSSL version 1.1.1 has had 18 vulnerabilities found and fixed their severities being:

Low	9
Moderate	3
High	6
Critical	0

In this paper I'll go through the vulnerabilities by their severity, starting from the high and going to the low severity vulnerability from oldest to the newest. Each vulnerability will have their code reviewed on what has been changed, why this was changed from a secure programming point of view and if there are any Common Vulnerabilities and Exposures (CVE) attached to the vulnerability. All the code in this paper is taken from the OpenSSL GitHub page, where the changelog, possible CVEs and reasoning behing the change can be seen. OpenSSL GitHub page can be found at `https://github.com/openssl/openssl`.

As there are no critical vulnerabilities found in 1.1.1 I will not be covering them in detail, but a quick intro to them will suffice. Critical vulnerabilities affect common configurations that are likely to be exploited. These might include significant disclosure of the contents of sever memory or a vulnerability where remote code execution is considered likely. [3]

2. HIGH VULNERABILITIES

High vulnerabilities are characterized by issues that are lower risk than critical, due to differing factors, such as affecting less common configurations or which are less likely to be exploitable. But due to the security concerns these issues are kept private and will trigger a new release for the supported versions. [3]

2.1 SSL_check_chain vulnerability - CVE-2020-1967

The vulnerability here was whenever a server or client application calls the `SSL_check_chain()` function during or after a TLS 1.3 handshake with an invalid or unrecognised signature algorithm, which may crash due to a `NULL` pointer dereference. This can be exploited in a Denial of Service attack.

The code changed was from file `ssl/t1_lib.c`. Changed in commit `eb563247aef3e83dda7679c43f9649270462e5b1`. [4]

Change log 2.1. *CVE-2020-1967*

```
FILENAME: ssl/t1_lib.c
        sigalg = use_pc_sigalgs
                 ? tls1_lookup_sigalg(s->s3->tmp.peer_cert_sigalgs[i])
                 : s->shared_sigalgs[i];
-        if (sig_nid == sigalg->sigandhash)
+        if (sigalg != NULL && sig_nid == sigalg->sigandhash)
            return 1;
    }
    return 0;
```

As we can see from the code snippet, a simple check if the `sigalg` is not null was enough to remove this vector of attack.

2.2 GENERAL_NAME_cmp - CVE-2020-1971

The X.509 `GeneralName` type is a generic type to represent different names. One included in this is known as `EDIPartyName`. For handling these names OpenSSL provides `GENERAL_NAME_cmp` function to compare different instances of `GENERAL_NAME` if these

are equal or not. The problem arrives whenever both parties GENERAL_NAME contain an EDIPartyName. Here a NULL pointer deference may occur thus crashing OpenSSL creating a Denial of Service attack.

The changes in the code can be seen in Appendix A. Changed in commit f960d81215ebf3f65e03d4d5d857fb9b666d6920. [5]

The largest changes in the file were the creation of the function edipartyname_cmp, which compares two EDIPartyNames and raises an error with return -1 if there is a NULL in any EDIPartyName or partyname. This successfully mitigates the possibility if any of the given partyname is a NULL pointer and thus removes the attack vector.

2.3 ClientHello - CVE-2021-3449

OpenSSL supports renegotiations, which is when either side has expired the session and continues to send data. This means that either the session was expired due to timeout, a peer wants to changes the cipher suite or wants to request a peer certificate and hasn't done so. This can be exploited with crafted renegotiation ClientHello message from a client. If the client omits the signature_algorithms extension, where it was present in the initial ClienHello, but includes a signature_algorithms_cert extension then a NULL pointer dereference will occure and result in a crash. This can be exploited to create a Denial of Service attack.

The changed file was from file ssl/statem/extensions.c. Changed in commit fb9fa6b51defd48157eeb207f52181f735d96148. [6]

Change log 2.2. *CVE-2021-3449*

```
FILENAME: ssl/statem/extensions.c
    /* Clear any signature algorithms extension received */
    OPENSSL_free(s->s3->tmp.peer_sigalgs);
    s->s3->tmp.peer_sigalgs = NULL;
+    s->s3->tmp.peer_sigalgslen = 0;

    return 1;
}
```

As we can see, a simple addition to clear the extensions in tmp.peer_sigalgslen removes the any of the old pointers and mitigates this attack.

2.4 X509_V_FLAG_X509_STRICT - CVE-2021-3450

A coding error in the implementation of the X509_V_FLAG_X509_STRICT flag caused a bypass in the check that non-CA (Certificate Authority) certificates should not be able

to issue other certificates thus breaking verification. The flag itself disables workarounds for some broken certificates and makes the verification strictly apply to the X509 rules by checking the certificates present in a certificate chain.

This caused changes in two files, `crypto/x509/x509_vfy.c` and `test/verify_extra_test.c`. Changed in commit `2a40b7bc7b94dd7de897a74571e7024f0cf0d63b`. [7]

The changed files can be found in Appendix B. The changes in file `crypto/x509/x509_vfy.c` can be attributed to the clearing of old variables, as the `ret` variable in checking wasn't cleared after the check was done and so it would have lead to an unwanted execution path. And the changes in the file `test/verify_extra_test.c` can be attributed to the proper use of tests.

2.5 EVP_PKEY_decrypt() - CVE-2021-3711

When decrpyting SM2 encrypted data, the application is expected to call the function `EVP_PKEY_decrypt()` twice. On the first time, on entry, the `out` parameter can be `NULL` and on exit, the `outlen` parameter is populated with the correct buffer size the hold the decrpyted plaintext. With this `outlen` parameter the application can then allocate sufficient buffer and do the second call to `EVP_PKEY_decrypt()`, but this time passing a non-`NULL` value for the `out` parameter. A coding error in the implementation of the SM2 decryption code can cause the buffer size required to hold the plaintext be to small for the second call thus leading us to a buffer overflow of a maximum of 64 bytes, possibly changing the application behaviour or causing a crash.

This caused changes in four files, `crypto/sm2/sm2_crypt.c`, `crypto/sm2/sm2_pmeth.c`, `include/crypto/sm2.h` and `test/sm2_internal_test.c`. Changes made in commit `59f5e75f3bced8fc0e130d72a3f582cf7b480b46`. [8]

The largest change in the code can be found at the first file `crypto/sm2/sm2_crypt.c`. Here we can see that removing code can be useful to remove some vulnerabilities, as before the overhead size was incorrectly thought to be have a fixed element of 10 bytes. Thus when the logic of removing the overhead was changed, the easier and more readable way of requesting the size of the `sm2_text`, the size of the decrypted plaintext was used. This changed caused ripple effects on the other three changed files as some of the function parameters were either incorrect or not used at all. These are the reasoning behind changes to the other files `crypto/sm2/sm2_pmeth.c`, `include/crypto/sm2.h` and the test cases in `test/sm2_internal_test.c`.

2.6 BN_mod_sqrt() - CVE-2022-0778

Parsing and using elliptic curve cryptography requires the use of modular square roots and as such the OpenSSL needs to have its systems to calculate and verify the modular square roots. One of these functions contained a bug, which caused it to loop forever for non-prime moduli. This could be used by a malicious actor by crafting a certificate that has invalid explicit curve parameters, thus leading the function to loop forever and giving us the attack, Denial of service as the program is stuck in this function.

This caused changes in only the function it used, `crypto/bn/bn_sqrt.c`, the changes were made in commit `3118eb64934499d93db3230748a452351d1d9a65`. [9]

The changed file can be found in Appendix D. There is a variable that the change log doens't explain, that are integral to the changed, specifically `e`. This is explained in the file as `now write |p| - 1 as` 2^e`*q`, which means to our case, that if `i` were to surpass `e` we can be sure, that there isn't a modular square root and thus the function stops instead of running forever.

2.7 Classification for high vulnerabilities

As these were only high vulnerabilities and not critical, there weren't any vulnerabilities that could have caused damage to the servers running OpenSSL. Though the vulnerabilities shown could've caused unavailability and instability on the servers. Below is a table of the different attacks that were found in the high vulnerability classification.

Denial of Service	4
Buffer overflow	1
Certificate chain break	1

3. MODERATE VULNERABILITIES

Due to the scope of this work, I will not be going through the moderate vulnerabilities as specifically as I did with the high vulnerabilities. And so, this will be more of a general characterization of the vulnerabilities found.

Moderate severity vulnerabilities are characterized by issues that include crashes in client applications, flaws in protocols that are less commonly used and local flaws. These as well as critical and high vulnerabilities are kept generally private until the next release, and are scheduled so, that multiple flaws can be fixed once. [3]

3.1 Moderate Vulnerabilities

4. LOW VULNERABILITIES

Due to the scope of this work, I will not be going through the low vulnerabilities as specifically as I did with the high vulnerabilities. And so, this will be more of a general characterization of the vulnerabilities found.

Low vulnerabilities include issues such as those that only affect the command line utility or unlikely configurations. These are not kept secret, but will be generally fixed immediately in the latest development versions and may be backported to older versions that are still being updated. These might get their own CVE, but these do not trigger new releases. [3]

4.1 Low Vulnerabilities

5. YHTEENVETO

REFERENCES

[1] *ANNOUNCE: OpenSSL (Take 2)*. Jan. 6, 1999. URL: https://marc.info/?l=ssl-users&m=91566086807308&w=2 (visited on 04/02/2022).

[2] *OpenSSL Management Committee*. URL: https://www.openssl.org/community/omc.html (visited on 04/02/2022).

[3] *Security Policy*. Mar. 16, 2021. URL: https://www.openssl.org/policies/secpolicy.html2 (visited on 04/02/2022).

[4] *Fix NULL dereference in SSL$_c heck_c hain() for TLS1.3$*. URL: https://github.com/openssl/openssl/commit/eb563247aef3e83dda7679c43f9649270462e5b1 (visited on 07/02/2022).

[5] *Correctly compare EdiPartyName in GENERAL$_N AME_c mp()$*. URL: https://github.com/openssl/openssl/commit/f960d81215ebf3f65e03d4d5d857fb9b666d6920 (visited on 07/02/2022).

[6] *ssl sigalg extension: fix NULL pointer dereference*. URL: https://github.com/openssl/openssl/commit/fb9fa6b51defd48157eeb207f52181f735d96148 (visited on 07/02/2022).

[7] *check$_c hain_e xtensions : Donotoverrideerrorreturnvaluebycheck_c urve$*. URL: https://github.com/openssl/openssl/commit/2a40b7bc7b94dd7de897a74571e7024f0cf0d63 (visited on 07/02/2022).

[8] *Correctly calculate the length of SM2 plaintext given the ciphertext*. URL: https://github.com/openssl/openssl/commit/59f5e75f3bced8fc0e130d72a3f582cf7b480b46 (visited on 07/02/2022).

[9] *Fix possible infinite loop in BN$_m od_s qrt()$*. URL: https://github.com/openssl/openssl/commit/3118eb64934499d93db3230748a452351d1d9a65 (visited on 07/02/2022).

APPENDIX A: CVE-2020-1971

Change log A.1. *CVE-2020-1971*

```
FILENAME: crypto/x509v3/v3_genn.c
+static int edipartyname_cmp(const EDIPARTYNAME *a, const EDIPARTYNAME *b)
+{
+    int res;
+
+    if (a == NULL || b == NULL) {
+        /*
+         * Shouldn't be possible in a valid GENERAL_NAME, but we handle it
+         * anyway. OTHERNAME_cmp treats NULL != NULL so we do the same here
+         */
+        return -1;
+    }
+    if (a->nameAssigner == NULL && b->nameAssigner != NULL)
+        return -1;
+    if (a->nameAssigner != NULL && b->nameAssigner == NULL)
+        return 1;
+    /* If we get here then both have nameAssigner set, or both unset */
+    if (a->nameAssigner != NULL) {
+        res = ASN1_STRING_cmp(a->nameAssigner, b->nameAssigner);
+        if (res != 0)
+            return res;
+    }
+    /*
+     * partyName is required, so these should never be NULL. We treat it in
+     * the same way as the a == NULL || b == NULL case above
+     */
+    if (a->partyName == NULL || b->partyName == NULL)
+        return -1;
+
+    return ASN1_STRING_cmp(a->partyName, b->partyName);
```

```
+}
/* Returns 0 if they are equal, != 0 otherwise. */
int GENERAL_NAME_cmp(GENERAL_NAME *a, GENERAL_NAME *b)
{
    int result = -1;
    if (!a || !b || a->type != b->type)
        return -1;
    switch (a->type) {
    case GEN_X400:
+        result = ASN1_TYPE_cmp(a->d.x400Address, b->d.x400Address);
+        break;
+
    case GEN_EDIPARTY:
-        result = ASN1_TYPE_cmp(a->d.other, b->d.other);
+        result = edipartyname_cmp(a->d.ediPartyName, b->d.ediPartyName);
        break;

    case GEN_OTHERNAME:

        result = OTHERNAME_cmp(a->d.otherName, b->d.otherName);
        break;
    case GEN_EMAIL:
    case GEN_DNS:
    case GEN_URI:
        result = ASN1_STRING_cmp(a->d.ia5, b->d.ia5);
        break;
    case GEN_DIRNAME:
        result = X509_NAME_cmp(a->d.dirn, b->d.dirn);
        break;
    case GEN_IPADD:
        result = ASN1_OCTET_STRING_cmp(a->d.ip, b->d.ip);
        break;
    case GEN_RID:
        result = OBJ_cmp(a->d.rid, b->d.rid);
        break;
    }
    return result;
}
/* Returns 0 if they are equal, != 0 otherwise. */
int OTHERNAME_cmp(OTHERNAME *a, OTHERNAME *b)
```

```
{
    int result = -1;
    if (!a || !b)
        return -1;
    /* Check their type first. */
    if ((result = OBJ_cmp(a->type_id, b->type_id)) != 0)
        return result;
    /* Check the value. */
    result = ASN1_TYPE_cmp(a->value, b->value);
    return result;
}
void GENERAL_NAME_set0_value(GENERAL_NAME *a, int type, void *value)
{
    switch (type) {
    case GEN_X400:
+         a->d.x400Address = value;
+         break;
+
    case GEN_EDIPARTY:
-         a->d.other = value;
+         a->d.ediPartyName = value;
         break;

    case GEN_OTHERNAME:
        a->d.otherName = value;
        break;
    case GEN_EMAIL:
    case GEN_DNS:
    case GEN_URI:
        a->d.ia5 = value;
        break;
    case GEN_DIRNAME:
        a->d.dirn = value;
        break;
    case GEN_IPADD:
        a->d.ip = value;
        break;
    case GEN_RID:
        a->d.rid = value;
        break;
```

```
    }
    a->type = type;
}
void *GENERAL_NAME_get0_value(const GENERAL_NAME *a, int *ptype)
{
    if (ptype)
        *ptype = a->type;
    switch (a->type) {
    case GEN_X400:
+        return a->d.x400Address;
+
    case GEN_EDIPARTY:
-        return a->d.other;
+        return a->d.ediPartyName;

    case GEN_OTHERNAME:
        return a->d.otherName;
```

APPENDIX B: CVE-2021-3450

Change log B.1. *CVE-2021-3450-1*

```
FILENAME: crypto/x509/x509_vfy.c
-        if ((ctx->param->flags & X509_V_FLAG_X509_STRICT) && num > 1) {
+        if (ret > 0
+            && (ctx->param->flags & X509_V_FLAG_X509_STRICT) && num > 1) {
            /* Check for presence of explicit elliptic curve parameters */
            ret = check_curve(x);
-            if (ret < 0)
+            if (ret < 0) {
                ctx->error = X509_V_ERR_UNSPECIFIED;
-            else if (ret == 0)
+                ret = 0;
+            } else if (ret == 0) {
                ctx->error = X509_V_ERR_EC_KEY_EXPLICIT_PARAMS;
+            }
        }
-        if ((x->ex_flags & EXFLAG_CA) == 0
+        if (ret > 0
+            && (x->ex_flags & EXFLAG_CA) == 0
            && x->ex_pathlen != -1
            && (ctx->param->flags & X509_V_FLAG_X509_STRICT)) {
            ctx->error = X509_V_ERR_INVALID_EXTENSION;
```

Change log B.2. *CVE-2021-3450-2*

```
FILENAME: test/verify_extra_test.c
    i = X509_verify_cert(sctx);

-    if (i == 0 && X509_STORE_CTX_get_error(sctx) == X509_V_ERR_INVALID_CA) {
+    if (i != 0 || X509_STORE_CTX_get_error(sctx) != X509_V_ERR_INVALID_CA)
+        goto err;
+
+    /* repeat with X509_V_FLAG_X509_STRICT */
```

```
+    X509_STORE_CTX_cleanup(sctx);
+    X509_STORE_set_flags(store, X509_V_FLAG_X509_STRICT);
+
+    if (!X509_STORE_CTX_init(sctx, store, x, untrusted))
+        goto err;
+
+    i = X509_verify_cert(sctx);
+
+    if (i == 0 && X509_STORE_CTX_get_error(sctx) == X509_V_ERR_INVALID_CA)
        /* This is the result we were expecting: Test passed */
        ret = 1;
-    }
+
 err:
    X509_STORE_CTX_free(sctx);
    X509_free(x);
```

APPENDIX C: CVE-2021-3711

Change log C.1. *CVE-2021-3711-1*

```
FILENAME: crypto/sm2/sm2_crypt.c

-int sm2_plaintext_size(const EC_KEY *key, const EVP_MD *digest, size_t msg_len
-                        size_t *pt_size)
+int sm2_plaintext_size(const unsigned char *ct, size_t ct_size, size_t *pt_siz
{
-    const size_t field_size = ec_field_size(EC_KEY_get0_group(key));
-    const int md_size = EVP_MD_size(digest);
-    size_t overhead;
+    struct SM2_Ciphertext_st *sm2_ctext = NULL;

-    if (md_size < 0) {
-        SM2err(SM2_F_SM2_PLAINTEXT_SIZE, SM2_R_INVALID_DIGEST);
-        return 0;
-    }
-    if (field_size == 0) {
-        SM2err(SM2_F_SM2_PLAINTEXT_SIZE, SM2_R_INVALID_FIELD);
-        return 0;
-    }
+    sm2_ctext = d2i_SM2_Ciphertext(NULL, &ct, ct_size);

-    overhead = 10 + 2 * field_size + (size_t)md_size;
-    if (msg_len <= overhead) {
+    if (sm2_ctext == NULL) {
        SM2err(SM2_F_SM2_PLAINTEXT_SIZE, SM2_R_INVALID_ENCODING);
        return 0;
    }

-    *pt_size = msg_len - overhead;
+    *pt_size = sm2_ctext->C2->length;
```

```
+    SM2_Ciphertext_free(sm2_ctext);
+
    return 1;
}
```

Change log C.2. *CVE-2021-3711-2*

```
FILENAME: crypto/sm2/sm2_pmeth.c
    if (out == NULL) {
-        if (!sm2_plaintext_size(ec, md, inlen, outlen))
+        if (!sm2_plaintext_size(in, inlen, outlen))
            return -1;
        else
            return 1;
```

Change log C.3. *CVE-2021-3711-3*

```
FILENAME: include/crypto/sm2.h
int sm2_ciphertext_size(const EC_KEY *key, const EVP_MD *digest, size_t msg_len
                        size_t *ct_size);

-int sm2_plaintext_size(const EC_KEY *key, const EVP_MD *digest, size_t msg_len
-                        size_t *pt_size);
+int sm2_plaintext_size(const unsigned char *ct, size_t ct_size, size_t *pt_siz

int sm2_encrypt(const EC_KEY *key,
                const EVP_MD *digest,
```

Change log C.4. *CVE-2021-3711-4*

```
FILENAME: test/sm2_internal_test.c
    if (!TEST_mem_eq(ctext, ctext_len, expected, ctext_len))
        goto done;

-    if (!TEST_true(sm2_plaintext_size(key, digest, ctext_len, &ptext_len))
+    if (!TEST_true(sm2_plaintext_size(ctext, ctext_len, &ptext_len))
            || !TEST_int_eq(ptext_len, msg_len))
        goto done;
```

APPENDIX D: CVE-2022-0778

Change log D.1. *CVE-2022-0778*

```
FILENAME: crypto/bn/bn_sqrt.c
/*
 * Returns 'ret' such that ret^2 == a (mod p), using the Tonelli/Shanks
 * algorithm (cf. Henri Cohen, "A Course in Algebraic Computational Number
- * Theory", algorithm 1.5.1). 'p' must be prime!
+ * Theory", algorithm 1.5.1). 'p' must be prime, otherwise an error or
+ * an incorrect "result" will be returned.
 */
{
    BIGNUM *ret = in;
@@ -301,18 +302,23 @@ BIGNUM *BN_mod_sqrt(BIGNUM *in, const BIGNUM *a, const BI
            goto vrfy;
        }

-        /* find smallest  i  such that  b^(2^i) = 1 */
-        i = 1;
-        if (!BN_mod_sqr(t, b, p, ctx))
-            goto end;
-        while (!BN_is_one(t)) {
-            i++;
-            if (i == e) {
-                BNerr(BN_F_BN_MOD_SQRT, BN_R_NOT_A_SQUARE);
-                goto end;
+        /* Find the smallest i, 0 < i < e, such that b^(2^i) = 1. */
+        for (i = 1; i < e; i++) {
+            if (i == 1) {
+                if (!BN_mod_sqr(t, b, p, ctx))
+                    goto end;
+
+            } else {
```

```
+                if (!BN_mod_mul(t, t, t, p, ctx))
+                    goto end;
             }
-            if (!BN_mod_mul(t, t, t, p, ctx))
-                goto end;
+            if (BN_is_one(t))
+                break;
+        }
+        /* If not found, a is not a square or p is not prime. */
+        if (i >= e) {
+            BNerr(BN_F_BN_MOD_SQRT, BN_R_NOT_A_SQUARE);
+            goto end;
         }

         /* t := y^2^(e - i - 1) */
```