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 paper will be a review of the changes made to the OpenSSL version 1.1.1 due to vulnerabilities during 2018-2022, from the first release to the most updated one. This paper is made from a secure programming point of view and thus will explain the changes made in the source code for OpenSSL, which can be found on their GitHub (https://github.com/openssl/openssl). The vulnerabilities are seen through the git commits, the changed code is presented to the reader and the reasoning behind the change is explained.

The vulnerabilities are classified by their severity and the classification is made by the OpenSSL team. The total number of changes made to this version is 18, which contains nine low vulnerabilities, three moderate vulnerabilities, six high vulnerabilities and zero critical vulnerabilities.

The vulnerabilities are also categorized into more classical examples of vulnerabilities, such as denial of service or buffer overflow attacks. The most common category was the denial of service attack with seven vulnerabilities found, then buffer overflow with three, forcing a side channel with three, predictable rng or nonce with two, certificate chain break with one, padding attack with one, writeable configuration with one and not analyzed in detail with one.

Keywords: Secure Programming, OpenSSL, OpenSSL 1.1.1

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 expired due to timeout, a peer wants to change the cypher suite or wants to request a peer certificate and hasn't done so. This can be exploited with a 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 occur 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, simple addition to clear the extensions in tmp.peer_sigalgslen removes 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 led 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 decrypting 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 decrypted 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 to be too 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 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 change 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 changelog doesn't explain, that is integral to the changed, specifically `e`. This is explained in the file as `now write |p| - 1 as` 2^e`*q`, which means in 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 Categorization of 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 showed 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

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 X509_issuer_and_serial_hash() - CVE-2021-23841

While handling X509 certificates, the public API function `X509_issuer_and_serial_hash()` attempts to create a unique hash based on the issuer and serial number. However, this function fails to handle any errors which may occur paring the issuer field and thus lead to a `NULL` pointer deference and crash leading to a denial of service attack. Since the function is never directly called by OpenSSL, the applications are only vulnerable if they use this function directly.

The code changed was from file `crypto/x509/x509_cmp.c`. Changed in commit `122a19ab48091c657f7cb1fb3af9fc07bd557bbf`. [10]

Change log 3.1. *CVE-2021-23841*

```
FILENAME: crypto/x509/x509_cmp.c
    if (ctx == NULL)
        goto err;
    f = X509_NAME_oneline(a->cert_info.issuer, NULL, 0);
+     if (f == NULL)
+         goto err;
    if (!EVP_DigestInit_ex(ctx, EVP_md5(), NULL))
        goto err;
    if (!EVP_DigestUpdate(ctx, (unsigned char *)f, strlen(f)))
```

A simple check if the pointer for `f` is `NULL` leads to this crash not happening and thus removes this vector of attack.

3.2 ASN1_STRING - CVE-2021-3712

OpenSSL handles ANS.1 strings with an ANS1_STRING structure, which contains a buffer holding the string data and buffer length. Compared to normal C strings, which contain a buffer with string data which is terminated with a NULL (0) byte. Applications can create a valid ANS1_STRING structure, which does not terminate the byte array with a NULL by directly setting the fields of "data" and "length". The vulnerability comes into play as multiple OpenSSL functions assume that the ANS1_STRING byte array will be terminated with NULL. With this, a read buffer overrun can occur. If a malicious actor can cause an application to directly construct an ANS1_STRING and then process it. This might cause a crash causing a Denial of Service attack and could even result in the disclosure of private memory content.

The code changed was from file crypto/ec/ec_asn1.c. Changed in commit 94d23fcff9b2a7a8368dfe52214d5c2569882c11. [11]

Change log 3.2. *CVE-2021-3712*

```
FILENAME: crypto/ec/ec_asn1.c
        ret->seed_len = params->curve->seed->length;
    }

-    if (!params->order || !params->base || !params->base->data) {
+    if (params->order == NULL
+            || params->base == NULL
+            || params->base->data == NULL
+            || params->base->length == 0) {
        ECerr(EC_F_EC_GROUP_NEW_FROM_ECPARAMETERS, EC_R_ASN1_ERROR);
        goto err;
    }
```

The vulnerability is fixed by checking if the ANS1_STRING contains the terminating NULL and not just trusting that it contains it.

3.3 MISP* carry propagation - CVE-2021-4160

Here we can see, what the moderate severity may lead to since the impact of this vulnerability wasn't analyzed in detail as the pre-requisites for the attack are considered unlikely and include reusing private keys. Also, the resources required for this attack would be significant. But in short, there was a carry propagation bug in the MIPS32 and MIPS64 squaring procedure, which meant that these procedures would compute a faulty result.

The code changed was in files crypto/bn/asm/mips.pl and changes to test cases in `test/bntest.c`. Changed in commit `e9e726506cd2a3fd9c0f12daf8cc1fe934c7dddb`. [12]

The changes can be found in Appendix E. The changes are due to the nature of the vulnerability calculation changes within the file and changes to the test cases.

3.4 Categorization of moderate vulnerabilities

As these were moderate vulnerabilities, the impact on servers is lower and even as some might have caused unavailability or instability on the servers running OpenSSL, the fact that these require specific configurations limits them to the moderate class. Below is a table of the different attacks.

Denial of Service	2
Not analyzed in detail	1

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 Categorization of Low Vulnerabilities

Even as the table which categorizes the low vulnerabilities seems to be up to critical in their categorized value, the OpenSSL team has classified these to be low. These are mainly due to affecting such a low number or unlikely configurations. For example, the World writeable configuration required the configuration to be changed so that the `OPENSSLDIR`, the place where configurations are stored is a world writeable. This may occur when using OpenSSL in Mingw. But this would require the user of the server to set the directory world writeable.

Force a side-channel	3
Predictable RNG or Nonce	2
Buffer overflow	2
Denial of service	1
Padding attack	1
World writeable configuration	1

5. SUMMARY

All in all the OpenSSL version 1.1.1 has had its own share of vulnerabilities in the timescale which it has been out. The vulnerabilities can be categorized as follows:

Denial of Service	7
Buffer overflow	3
Force a side-channel	3
Predictable RNG or Nonce	2
Certificate chain break	1
Not analyzed in detail	1
Padding attack	1
World writeable configuration	1

As we can see most of the vulnerabilities come from the possibility to crash OpenSSL thus leading to a denial of service attack. Surprisingly high is also the possibility of the overflow attacks, as in the timescale there were three total overflow attacks, though as two of these were classified as low vulnerability due to the specific configuration required for these attacks to be viable.

REFERENCES

[1] *ANNOUNCE: OpenSSL (Take 2)*. Jan. 6, 1999. URL: https://marc.info/?l=ssl-users&m=91566086807308&w=2 (visited on 05/03/2022).

[2] *OpenSSL Management Committee*. URL: https://www.openssl.org/community/omc.html (visited on 05/03/2022).

[3] *Security Policy*. Mar. 16, 2021. URL: https://www.openssl.org/policies/secpolicy.html2 (visited on 05/03/2022).

[4] *Fix NULL dereference in SSL_check_chain() for TLS 1.3*. URL: https://github.com/openssl/openssl/commit/eb563247aef3e83dda7679c43f9649270462e5b1 (visited on 05/03/2022).

[5] *Correctly compare EdiPartyName in GENERAL_NAME_cmp()*. URL: https://github.com/openssl/openssl/commit/f960d81215ebf3f65e03d4d5d857fb9b666d6920 (visited on 05/03/2022).

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

[7] *check_chain_extensions: Do not override error return value by check_curve*. URL: https://github.com/openssl/openssl/commit/2a40b7bc7b94dd7de897a74571e7024f (visited on 05/03/2022).

[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_mod_sqrt()*. URL: https://github.com/openssl/openssl/commit/3118eb64934499d93db3230748a452351d1d9a65 (visited on 05/03/2022).

[10] *Fix Null pointer deref in X509_issuer_and_serial_hash()*. URL: https://github.com/openssl/openssl/commit/122a19ab48091c657f7cb1fb3af9fc07bd557bbf (visited on 05/03/2022).

[11] *Fix EC_GROUP_new_from_ecparameters to check the base length*. URL: https://github.com/openssl/openssl/commit/94d23fcff9b2a7a8368dfe52214d5c2569882c (visited on 05/03/2022).

[12] *Fix a carry overflow bug in bn_sqr_comba4/8 for mips 32-bit targets*. URL: https://github.com/openssl/openssl/commit/e9e726506cd2a3fd9c0f12daf8cc1fe934c7dd (visited on 05/03/2022).

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) */
```

APPENDIX E: CVE-2021-4160

Change log E.1. *CVE-2021-4160-1*

```
FILENAME: crypto/bn/asm/mips.pl
        sltu        $at,$c_2,$t_1
        $ADDU        $c_3,$t_2,$at
        $ST        $c_2,$BNSZ($a0)
+        sltu        $at,$c_3,$t_2
+        $ADDU        $c_1,$at
        mflo        ($t_1,$a_2,$a_0)
        mfhi        ($t_2,$a_2,$a_0)

---
@@ -2194,6 +2196,8 @@ ()
        sltu        $at,$c_2,$t_1
        $ADDU        $c_3,$t_2,$at
+        $ST        $c_2,$BNSZ($a0)
+        sltu        $at,$c_3,$t_2
        $ADDU        $c_1,$at
        mflo        ($t_1,$a_2,$a_0)
        mfhi        ($t_2,$a_2,$a_0)

---
```

Change log E.2. *CVE-2021-4160-2*

```
FILENAME: test/bntest.c
    if (!TEST_BN_eq(c, d))
        goto err;

+    /*
+     * Regression test for overflow bug in bn_sqr_comba4/8 for
+     * mips-linux-gnu and mipsel-linux-gnu 32bit targets.
+     */
+    {
+        static const char *ehex[] = {
+            "95564994a96c45954227b845a1e99cb939d5a1da99ee91acc962396ae999a9ee'
+           "38603790448f2f7694c242a875f0cad0aae658eba085f312d2febbbd128dd2b5"
+           "8f7d1149f03724215d704344d0d62c587ae3c5939cba4b9b5f3dc5e8e911ef9a"
+           "5ce1a5a749a4989d0d8368f6e1f8cdf3a362a6c97fb02047ff152b480a4ad985"
+           "2d45efdf0770542992afca6a0590d52930434bba96017afbc9f99e112950a8b1"
+           "a359473ec376f329bdae6a19f503be6d4be7393c4e43468831234e27e3838680"
+           "b949390d2e416a3f9759e5349ab4c253f6f29f819a6fe4cbfd27ada34903300e"
+           "da021f62839f5878a36f1bc3085375b00fd5fa3e68d316c0fdace87a97558465"
+           NULL};
+       static const char *phex[] = {
+           "f95dc0f980fbd22e90caa5a387cc4a369f3f830d50dd321c40db8c09a7e1a241"
+           "a536e096622d3280c0c1ba849c1f4a79bf490f60006d081e8cf69960189f0d31"
+           "2cd9e17073a3fba7881b21474a13b334116cb2f5dbf3189a6de3515d0840f053"
+           "c776d3982d391b6d04d642dda5cc6d1640174c09875addb70595658f89efb439"
+           "dc6fbd55f903aadd307982d3f659207f265e1ec6271b274521b7a5e28e8fd7a5"
+           "5df089292820477802a43cf5b6b94e999e8c9944ddebb0d0e95a60f88cb7e813"
+           "ba110d20e1024774107dd02949031864923b3cb8c3f7250d6d1287b0a40db6a4"
+           "7bd5a469518eb65aa207ddc47d8c6e5fc8e0c105be8fc1d4b57b2e27540471d5"
+           NULL};
+       static const char *mhex[] = {
+           "fef15d5ce4625f1bccfbba49fc8439c72bf8202af039a2259678941b60bb4a8f"
+           "2987e965d58fd8cf86a856674d519763d0e1211cc9f8596971050d56d9b35db3"
+           "785866cfbca17cfdbed6060be3629d894f924a89fdc1efc624f80d41a22f1900"
+           "9503fcc3824ef62ccb9208430c26f2d8ceb2c63488ec4c07437aa4c96c43dd8b"
+           "9289ed00a712ff66ee195dc71f5e4ead02172b63c543d69baf495f5fd63ba7bc"
+           "c633bd309c016e37736da92129d0b053d4ab28d21ad7d8b6fab2a8bbdc8ee647"
+           "d2fbcf2cf426cf892e6f5639e0252993965dfb73ccd277407014ea784aaa280c"
+           "b7b03972bc8b0baa72360bdb44b82415b86b2f260f877791cd33ba8f2d65229b"
+           NULL};
+
+       if (!TEST_true(parse_bigBN(&e, ehex))
+               || !TEST_true(parse_bigBN(&p, phex))
+               || !TEST_true(parse_bigBN(&m, mhex))
+               || !TEST_true(BN_mod_exp_mont_consttime(d, e, p, m, ctx, NULL)
+               || !TEST_true(BN_mod_exp_simple(a, e, p, m, ctx))
+               || !TEST_BN_eq(a, d))
+           goto err;
+    }
+
   /* Zero input */
   if (!TEST_true(BN_bntest_rand(p, 1024, 0, 0)))
```

```
goto err;
```