openssl / **openssl**

Code | Issues 1k | **Pull requests** 685 | Discussions | Actions | Projects

Wiki | Security and quality | Insights

[NSF] [CVE-2026-121481] Buffer overflow in EVP_EncodeBlock() #33652

Open **nmouha** wants to merge 4 commits into `openssl:master` from `nmouha:cve-2026-121481`

Conversation 28 | Commits 4 | Checks 103 | Files changed 4

nmouha commented last week Contributor

This fixes "Buffer overflow in EVP_EncodeBlock()" (CVE-2026-121481) by returning `-1` for invalid inputs to `EVP_EncodeBlock()`: `NULL` input or output buffers, negative lengths, and lengths for which the encoded size does not fit in an `int`. The OpenSSL SRT has asked for it to be fixed publicly; the advisory is still pending.

Returning `-1` for invalid inputs is the most straightforward fix, and matches what happens in wolfSSL's compatibility layer for OpenSSL's EVP_EncodeBlock().

I'm suggesting a similar guard for `EVP_DecodeBlock()`. No length limit is needed here because the decoded length always fits in an `int`, but it is a good idea to reject `NULL` buffers and negative lengths.

Moreover, I propose to fix the incorrect length check in the `base64encode()` function of `http_client.c`, which rejected only `len > INT_MAX`.

I am adding tests in `test/base64_dec_avx2_test.c` to cover the new behavior, and documenting the `-1` returns in `doc/man3/EVP_EncodeInit.pod`.

Exploiting this vulnerability from the command line is possible by passing a large `-proxy_pass` to `openssl s_client`. I confirmed that GOST accepts these large passwords out of the box. However, the password has to be supplied through an environment variable (`env:VAR`), otherwise the command line will be too long.

Note: the pattern of CVE-2026-121481 is similar to that of CVE-2021-23840.

Checklist

- ☑ documentation is added or updated
- ☑ tests are added or updated

👍 4

openssl-ci-bot Bot added the **approval: review pending** label last week

nhorman added the **team: NSF** label last week

github-project-automation Bot added this to **NSF Issues** last week

github-project-automation Bot moved this to **To triage** in **NSF Issues** last week

jogme added **branch: master** **branch: 4.0** **branch: 4.1** no_news_changes_needed labels last week

t8m added **triaged: bug** **severity: important** labels last week

`[NSF] [CVE-2026-121481] Buffer overflow in EVP_EncodeBlock()` … ✗ `efc46cb`

nmouha force-pushed the `cve-2026-121481` branch from **`b632810`** to **`efc46cb`** last week Compare

nhorman moved this from **To triage** to **In Review** in **NSF Issues** last week

nhorman added this to **Development Board** last week

nhorman moved this to **Waiting Review** in **Development Board** last week

nhorman previously approved these changes last week View reviewed changes

jogme requested changes last week View reviewed changes

jogme left a comment Contributor

This patch changes functions so they can fail. Can you please also ensure the call sites check the return value of this function?

The problematic places I found:

- `PEM_SignFinal()`
- `NETSCAPE_SPKI_b64_encode()`

github-project-automation Bot moved this from **In Review** to **In progress** in **NSF Issues** last week

t8m commented last week Member

> This patch changes functions so they can fail. Can you please also ensure the call sites check the return value of this function?
>
> The problematic places I found:
>
> ```
> * `PEM_SignFinal()`
>
> * `NETSCAPE_SPKI_b64_encode()`
> ```

Unless these callers can plausibly trigger the error conditions, I am not sure it needs to be done as part of this PR.

It should definitely be a separate commit.

t8m requested changes last week View reviewed changes

Reviewers

- **andrewkdinh** ✓
- **nhorman** ●
- **jogme** ●
- **t8m** ●
- **bob-beck** ●

Requested changes must be addressed to merge this pull request.

Assignees

nmouha

Labels

approval: review pending
branch: master **branch: 4.0**
branch: 4.1 **hold: discussion**
no_news_changes_needed
severity: important **team: NSF**
triaged: bug

Projects

Development Board
Status: Waiting Review +7 more

NSF Issues
Status: In progress +3 more

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.
None yet

7 participants

crypto/evp/encode.c Outdated

```
+
+     /* Ensure the base64 output lengt
+     if (dlen > (INT_MAX / 4) * 3)
+         return -1;
```

t8m last week Member

I am really worried about the negative return value being added. Existing third party callers might not be ready to handle this correctly (i.e. if they cast the return to size_t without checking). Could we instead return 0? (In all the added error returns in this PR.)

nmouha last week Contributor Author

Hi Tomas,

I'm okay making any changes that you suggest, but are you really sure about this?

Both EVP_EncodeBlock() and EVP_DecodeBlock() can legitimately return 0, so 0 is not an error for either function. EVP_EncodeBlock() returns 0 when dlen == 0, and EVP_DecodeBlock() returns 0 when the input is empty after trimming.

Also, for EVP_DecodeBlock(), the current failure paths (e.g., trimmed length not a multiple of invalid base64 characters) all return -1. So, I wanted to make sure that both EVP_EncodeBlock() and EVP_DecodeBlock() consistently return -1 on any error. As I mentioned before, returning -1 is also what WolfSSL does in its OpenSSL compatibility layer.

I understand the concern that third-party callers may cast the return to size_t without checking. But are we not making things more difficult for third-party calls if both 0 and -1 can indicate errors, or if the existing EVP_DecodeBlock() errors will be changed from -1 to 0?

PS: I'm happy to make a separate commit for the problematic call site checks that Norbert pointed out.

Best,
Nicky

nhorman last week Contributor

maybe return 0 instead? Since EVP_EncodeBlock is defined in the docs as returning the number of bytes encoded, a 0 return would be reasonable, as it accurately indicates no bytes were encoded. We could additionally add an ERR_raise_data to indicate the reason why it happened.

👍 1

bob-beck last week Contributor

I've already been tagged for trying to raise an error on a success case - and probably legitimately - 0 is a *success* case.

bob-beck last week Contributor

So I think this is the incorrect fix, We already for STYLE.md say we should *NOT* add special case errors like this where we get a failure case that looks like a success. We should document that this function does not accept negative lengths, and simply use OPENSSL_ASSERT here.

Making something return a magic value that appears to be a success for this just adds dead code and impossible to reach and untested error cases everywhere.

bob-beck last week Contributor

this PR also adds cases *not* in the report, and the report's reproducer is completely synthetic, and probably should not be a CVE without evidence it really happens. I suggest this should be fixed in documentation

nhorman last week Contributor

then lets just make a dlen overflow an asserted failure

nmouha last week Contributor Author

Thanks for all the feedback! Following this discussion, I'm suggesting OPENSSL_assert() here.

nmouha mentioned this pull request last week

[NSF] Check if EVP_EncodeBlock() returns an error #33710 Draft

○ 1 task done

bob-beck requested changes last week View reviewed changes

crypto/evp/encode.c Outdated

```
    {
        int wrap_cnt = 0;

+       if (t == NULL || f == NULL || dle
```

bob-beck last week Contributor

This also rejects an empty input passed as f == NULL with dlen == 0. That call used to work: it wrote the NUL terminator and returned 0. An application encoding an empty buffer it never allocated would now get -1. Could you reject f == NULL only when dlen > 0? EVP_DecodeBlock() has the same issue, and there t is not written either when n == 0.

nmouha last week Contributor Author

Thanks for pointing this out! I've fixed the PR to preserve all existing well-defined behavior.

Small note here: for this input, the AVX2 path has src + 0, which is undefined behavior if src is NULL. So, I'm suggesting a code change that handles this case before invoking the AVX2 path.

`crypto/evp/encode.c` Outdated

```
      int wrap_cnt = 0;

+     if (t == NULL || f == NULL || dle
+         return -1;
```

bob-beck last week Contributor ···

nit: these are public functions, so STYLE.md ("Checking function arguments") wants an error on the stack here: ERR_R_PASSED_NULL_PARAMETER for the NULL pointers and ERR_R_PASSED_INVALID_ARGUMENT for a negative or oversized length. The same goes for the new checks in EVP_DecodeBlock().

nmouha last week Contributor Author ···

From the discussion so far, it seems that the best way forward is OPENSSL_assert() to for inputs that previously resulted in undefined behavior. This avoids breaking existing third-party callers, which do not expect a new error value. If you'd like ERR_R_PASSED_INVALID_ARGUMENT instead of OPENSSL_assert(), let me know!

`crypto/evp/encode.c` Outdated

```
+
+     /* Ensure the base64 output lengt
+     if (dlen > (INT_MAX / 4) * 3)
+         return -1;
```

bob-beck last week Contributor ···

I've already been tagged for trying to raise an error on a success case - and probably legitimately - 0 is a *success* case.

`crypto/evp/encode.c` Outdated

```
+
+     /* Ensure the base64 output lengt
+     if (dlen > (INT_MAX / 4) * 3)
+         return -1;
```

bob-beck last week Contributor ···

So I think this is the incorrect fix, We already for STYLE.md say we should *NOT* add special case errors like this where we get a failure case that looks like a success. We should document that this function does not accept negative lengths, and simply use OPENSSL_ASSERT here.

Making something return a magic value that appears to be a success for this just adds dead code and impossible to reach and untested error cases everywhere.

bob-beck added the **hold: discussion** label last week

bob-beck reviewed last week View reviewed changes

bob-beck left a comment Contributor ···

Note for our fixer: Just because Wolf does it doesn't mean it's right. I think Wolf's approach here is wrong and dangerous because it adds other problems to solve one that should be non-existent in the first place.

esyr commented last week Member ···

So, this API is broken, I'd suggest to write just the terminating `NUL` byte (and return 0) if the expected output size is more than `INT_MAX`, deprecate the function, and add some `EVP_EncodeBlock_ex()` with a saner API contract.

👍 1

nmouha dismissed **nhorman**'s stale review via `10723d8` last week

nmouha commented last week Contributor Author ···

Thanks for all the feedback! I really appreciate all the insights.

In the context this PR, it seems best to focus only on fixing a CVE, and maintaining all existing well-defined behavior of the API. So, I've added OPENSSL_assert() for inputs where there was previously undefined behavior. This avoids that third-party callers need to handle new errors.

Note that encode.c already uses OPENSSL_assert(n < sizeof(ctx->enc_data)) in EVP_DecodeUpdate(), so there is some precedent for this.

This design makes follow-up PR #33710 unnecessary. Deprecation of the API would seem outside of scope for this PR, its goal is to fix the CVE only. (And of course I never meant to say whatever wolfSSL did was right!)

nmouha force-pushed the `cve-2026-121481` branch 2 times, most recently from **a9a7715** to **9839c4e** last week Compare

`fixup! [NSF] [CVE-2026-121481] Buffer overflow in EVP_EncodeBlock()` ··· ✗ `ca24cdc`

44 hidden items
Load more...

nmouha force-pushed the `cve-2026-121481` branch from **9839c4e** to **ca24cdc** last week Compare

nhorman assigned **nmouha** 4 days ago

andrewkdinh requested review from **bob-beck**, **jogme**, **nhorman** and **t8m** 4 days ago

bob-beck previously approved these changes 3 days ago View reviewed changes

bob-beck left a comment Contributor ···

LGTM if you want to fix my two nits, and land this for now, (I'll approve it again for you)

crypto/evp/encode.c Outdated

```
    {
        int wrap_cnt = 0;

+       OPENSSL_assert(t != NULL);
```

bob-beck 4 days ago Contributor ···

You do not need to add an OPENSSL_assert for this. it will eat the NULL and crash. let it.

nmouha 2 days ago Contributor Author ···

Okay! I've removed this OPENSSL_assert.

crypto/evp/encode.c

```
+       OPENSSL_assert(t != NULL);
+       OPENSSL_assert(dlen >= 0);
+       /* Ensure the base64 output lengt
+       OPENSSL_assert(dlen <= (INT_MAX /
```

bob-beck 3 days ago Contributor ···

So this sticks out like a sore thumb here. why is this magic computation correct ?

I think your answer is that "oh when I call the internal function afterwards this can happen"

that's fine then, but now, hey, unlike this public API that you are getting beat up by people like me for trying to change (I'm sorry) the *INTERNAL* function you are free to change!

So you *could* change the internal function to return success or failure as per current STYLE.md (putting the length in an output variable on success). and then simply note and assert if that function fails. meanwhile the other callers of that internal function that can return an error can do so.

This is a bit more of an extensive change, so if you wanted to do it in a follow on, this might warrant having a comment here with an issue number saying why this magic calculation is here and what should be done about it. I'm ok if you want to do it that way, or add that to this pr.

nmouha 2 days ago Contributor Author ···

Okay! I've added a short comment explaining the bound, and I've added a comment with an issue number (#33812), so that the follow-up issue does not block the CVE fix.

nmouha mentioned this pull request 2 days ago

Move base64 overflow check to the internal functions #33812 Open

nmouha dismissed **bob-beck**'s stale review via 6d5bd94 2 days ago

`fixup! fixup! [NSF] [CVE-2026-121481] Buffer overflow in EVP_EncodeBl…` ··· ✕ ce28e0f

nmouha force-pushed the `cve-2026-121481` branch from **`6d5bd94`** to **`ce28e0f`** 2 days ago Compare

andrewkdinh reviewed 2 days ago View reviewed changes

`crypto/evp/encode.c` Outdated

Comment on lines +502 to +503

```
502  +      OPENSSL_assert(n >= 0);
503  +      OPENSSL_assert(n == 0 || (t != NUI
```

andrewkdinh 2 days ago Contributor ···

Suggested change

```
-     OPENSSL_assert(n >= 0);
-     OPENSSL_assert(n == 0 || (t != NULL && f != NULL));
+     if (n <= 0)
+         return n == 0 ? 0 : -1;
```

same nit as **@bob-beck** suggested

nmouha yesterday Contributor Author ···

Makes sense, -1 is already the documented error return for EVP_DecodeBlock(). I've applied your proposed change and I've added a new test vector as well.

bob-beck previously approved these changes yesterday View reviewed changes

bob-beck left a comment Contributor ···

Fix your last null check nits, and you're good.

`crypto/evp/encode.c` Outdated

```
237  +       * into the internal functions an
238  +       */
239  +      OPENSSL_assert(dlen <= (INT_MAX /
240  +      OPENSSL_assert(f != NULL || dlen
```

bob-beck 2 days ago Contributor ···

nit: the reasoning that dropped the `t != NULL` assert applies here too. With the `dlen == 0` early return, `f` is only read when `dlen > 0`, so a NULL `f` just crashes on first use. The same goes for the NULL part of the assert in EVP_DecodeBlock(). The documented precondition is enough for both.

nmouha yesterday Contributor Author ···

Thanks! I just made those two changes. If anything else is needed, just let me know!

× **nmouha** dismissed **bob-beck**'s stale review via `6e9cbe3` yesterday

nmouha force-pushed the `cve-2026-121481` branch from **`6e9cbe3`** to **`2c675d8`** 18 hours ago Compare

`fixup! fixup! fixup! [NSF] [CVE-2026-121481] Buffer overflow in EVP_E…` ··· × `ad8169a`

nmouha force-pushed the `cve-2026-121481` branch from **`2c675d8`** to **`ad8169a`** 18 hours ago Compare

andrewkdinh approved these changes 16 hours ago View reviewed changes

andrewkdinh left a comment Contributor ···

Backports will require separate PR's

daedalus pushed a commit to daedalus/fuzzer that referenced this pull request 6 hours ago

`Add openssl_b64_diff: AVX2 vs scalar base64 differential` ··· Verified `beed01c`

daedalus mentioned this pull request 6 hours ago

Add openssl_b64_diff; differential targets crash on mismatch Merged
daedalus/fuzzer#443

Repzei mentioned this pull request 6 hours ago

7am AI 2026-11-20: Google's Gemini can now use your Mac for you Merged
Repzei/7amai#38

github-actions Bot mentioned this pull request 5 hours ago

🚀 HN 精选 | 2026-11-20 Open
Everett406/daily-report#1209

jogme requested a review from **bob-beck** 2 hours ago

Foxhunt mentioned this pull request 2 hours ago

Daily Hacker News 20-11-2026 Open
Foxhunt/daily-hackernews#864

github-actions Bot mentioned this pull request 40 minutes ago

Episode 2026-11-20 — rate stories Open
vihucode/Aamukatsaus#95

Sign up for free **to join this conversation on GitHub**. Already have an account? Sign in to comment

© 2026 GitHub, Inc.