

New issue

Jump to bottom

Fernet fails to encrypt/decrypt large data #5615

Closed gablank opened this issue on Dec 9, 2020 · 6 comments

Labels

bugs

Milestone

Thirty Fourth Rel...

gablank commented on Dec 9, 2020 • edited

Hello,

Thank you for this fine library.

I've been having some issues when encrypting large data (>2GiB) using the Fernet class. There seems to be several failure modes, I've seen everything from a segfault, SIGABRT to the decrypted plaintext differing from the original plaintext. Please note that I've had some issues with the RAM on my computer (so that could potentially be the source of *some* of the failures), but I've verified at least the SIGABRT failures on three different computers.

It seems to me that the issue seems to be an integer overflow in OpenSSL, but I'm not sure if Cryptography is at fault for passing an integer that is too large, or OpenSSL is at fault for not checking the integer, or a combination. If you think this should be fixed in OpenSSL, please let me know so I can report the issue to them.

Please see the attached script for more detailed information, as I think it speaks for itself.

- Software versions:
 - Python 3.8.2/3.9.0
 - OpenSSL 1.1.1h
 - cryptography: 3.3
 - cffi: 1.14.4
 - pip: 20.3.1
 - setuptools: 51.0.0
- Installed cryptography through pip

To reproduce:

```
import cryptography.fernet as cf

# OK
data_size = 2**31 - 1


# Fails with SIGABRT and stderr: "free(): invalid size" or "munmap_chunk(): invalid pointer"
#
# OK if _CipherContext._MAX_CHUNK_SIZE is set to 2 ** 31 - 1 - 8
data_size = 2**32 - 1

# Fails with SIGABRT or SIGSEGV and stderr: "free(): invalid size" or "munmap_chunk(): invalid pointer" or
# cryptography.exceptions.InternalError: Unknown OpenSSL error.
# This error is commonly encountered when another library is not cleaning up the OpenSSL error stack.
# If you are using cryptography with another library that uses OpenSSL try disabling it before reporting a bug.
# Otherwise please file an issue at https://github.com/pyca/cryptography/issues with information on how to reproduce this.
# ([_OpenSSL.ErrorWithText(code=101560482, lib=6, func=219, reason=162, reason_text=b'error:060800A2:digital envelope routines:evp_EncryptDecryptUpdate:partially overlapping buffers')])
#
# Fails due to plaintext != original_plaintext if _CipherContext._MAX_CHUNK_SIZE is set to 2 ** 31 - 1 - 8
#
# OK if _CipherContext._MAX_CHUNK_SIZE is set to 2 ** 31 - 1 - 16
data_size = 2**33 - 1

original_plaintext = bytes(data_size)

fernet = cf.Fernet(cf.Fernet.generate_key())
ciphertext = fernet.encrypt(original_plaintext)
plaintext = fernet.decrypt(ciphertext)

assert plaintext == original_plaintext
```

 alex added the bugs label on Dec 9, 2020

gablank commented on Dec 9, 2020 • edited

Author

Here is some more information to aid with debugging the issue. I have verified these results on two different computers, so I am quite certain this is not in any way related to the RAM issues I have mentioned.

```
# _MAX_CHUNK_SIZE = 2 ** 31 - 1 (unmodified):
data_size = 2**32 - 1024**2 # OK
data_size = 2**32 - 1024 # OK
data_size = 2**32 - 512 # OK
data_size = 2**32 - 128 # OK
data_size = 2**32 - 64 # OK
data_size = 2**32 - 48 # OK
data_size = 2**32 - 33 # OK
data_size = 2**32 - 32 # FAIL (ValueError: The length of the provided data is not a multiple of the block length)
data_size = 2**32 - 31 # FAIL (ValueError: The length of the provided data is not a multiple of the block length)
data_size = 2**32 - 17 # FAIL (ValueError: The length of the provided data is not a multiple of the block length)
```

```
data_size = 2**32 - 16      # FAIL (SIGABRT: munmap_chunk(): invalid pointer)
data_size = 2**32 - 1      # FAIL (SIGABRT: munmap_chunk(): invalid pointer)
data_size = 2**32          # FAIL (SIGABRT: munmap_chunk(): invalid pointer or SIGSEGV)
data_size = 3*2**32        # FAIL (SIGSEGV)
```

Traceback from the cases that fail with a ValueError:

```
Traceback (most recent call last):
  File "/home/<redacted>/projects/cryptography_bug/venv/lib/python3.9/site-packages/cryptography/fernet.py", line 134, in _decrypt_data
    plaintext_padded += decryptor.finalize()
  File "/home/<redacted>/projects/cryptography_bug/venv/lib/python3.9/site-packages/cryptography/hazmat/primitives/ciphers/base.py", line 164, in finalize
    data = self._ctx.finalize()
  File "/home/<redacted>/projects/cryptography_bug/venv/lib/python3.9/site-packages/cryptography/hazmat/backends/openssl/ciphers.py", line 181, in finalize
    raise ValueError(
ValueError: The length of the provided data is not a multiple of the block length.
```

During handling of the above exception, another exception occurred:

```
Traceback (most recent call last):
  File "/home/<redacted>/projects/cryptography_bug/main.py", line 40, in <module>
    plaintext = fernet.decrypt(ciphertext)
  File "/home/<redacted>/projects/cryptography_bug/venv/lib/python3.9/site-packages/cryptography/fernet.py", line 76, in decrypt
    return self._decrypt_data(data, timestamp, ttl, int(time.time()))
  File "/home/<redacted>/projects/cryptography_bug/venv/lib/python3.9/site-packages/cryptography/fernet.py", line 136, in _decrypt_data
    raise InvalidToken
cryptography.fernet.InvalidToken
```

Process finished with exit code 1

reaperhulk commented on Dec 9, 2020 • edited

Member

This is *likely* to be our bug since we chunk items specifically to work around OpenSSL's integer size limits. We will investigate.

Update: This isn't our bug but we have worked around it by altering our chunking rules.

alex added this to the **Thirty Fourth Release** milestone on Dec 29, 2020

alex modified the milestones: **Thirty Fourth Release**, **Thirty Fifth Release** on Feb 6, 2021

alex added a commit to alex/cryptography that referenced this issue on Feb 7, 2021

correct buffer overflows cause by integer overflow in openssl ...

fae28c1

alex mentioned this issue on Feb 7, 2021

correct buffer overflows cause by integer overflow in openssl #5747

→ Merged

reaperhulk pushed a commit that referenced this issue on Feb 7, 2021

correct buffer overflows cause by integer overflow in openssl (#5747) ...

✓ 82b6ce2

alex closed this as completed on Feb 7, 2021

abergmann commented on Feb 8, 2021

CVE-2020-36242 got assigned to this issue.

s0undt3ch added a commit to s0undt3ch/salt that referenced this issue on Feb 10, 2021

Bump cryptography requirement to 3.3.2 due to CVE-2020-36242 ...

1a88085

s0undt3ch mentioned this issue on Feb 10, 2021

Bump cryptography requirement to 3.3.2 due to CVE-2020-36242 saltstack/salt#59468

→ Merged

bhearn7 mentioned this issue on Feb 11, 2021

address VULNDB-248976/CVE-2020-36242 for next release anchore/anchore-engine#909

🔒 Closed

github-actions (bot) mentioned this issue on Feb 12, 2021

Security Alert xiaolinpeter/rasa2.0_nlu#20

🔓 Open

piraz mentioned this issue on Feb 13, 2021

CVE-2020-36242: Symmetrically encrypting large values can lead to integer overflow candango/automatoes#84

🔒 Closed

- s0undt3ch added a commit to s0undt3ch/salt that referenced this issue on Feb 16, 2021

Bump cryptography requirement to 3.3.2 due to [CVE-2020-36242](#) ...

4ded60f
- s0undt3ch added a commit to s0undt3ch/salt that referenced this issue on Feb 18, 2021

Bump cryptography requirement to 3.3.2 due to [CVE-2020-36242](#) ...

5cb1246
- s0undt3ch added a commit to s0undt3ch/salt that referenced this issue on Feb 18, 2021

Bump cryptography requirement to 3.3.2 due to [CVE-2020-36242](#) ...

dfddb6
- s0undt3ch added a commit to s0undt3ch/salt that referenced this issue on Feb 19, 2021

Bump cryptography requirement to 3.3.2 due to [CVE-2020-36242](#) ...

4a517e0
- s0undt3ch added a commit to s0undt3ch/salt that referenced this issue on Feb 19, 2021

Bump cryptography requirement to 3.3.2 due to [CVE-2020-36242](#) ...

4c30c49
- Ch3LL pushed a commit to saltstack/salt that referenced this issue on Feb 23, 2021

Bump cryptography requirement to 3.3.2 due to [CVE-2020-36242](#) ...

db49815
- aequitas added a commit to internetstandards/Internet.nl-dashboard that referenced this issue on Mar 2, 2021

Fix security issue with cryptography: [pyca/cryptography#5615](#)

a52f818
- D10S0VskY-OSS mentioned this issue on Mar 7, 2021

Upgrade cryptography to version 3.3.2 D10S0VskY-OSS/Stack-Lifecycle-Deployment#2

Merged
- renovate (bot) mentioned this issue on Mar 8, 2021

Update dependency cryptography to v3.3.2 [SECURITY] andrewguest/cookiecutter-fastapi-jwt#35

Closed

1 task
- bhagwatyyas mentioned this issue on Mar 8, 2021

Bump cryptography from 3.2.1 to 3.3.2 oracle/oci-python-sdk#322

Closed
- vpolimenov mentioned this issue on Mar 11, 2021

Cryptography dependency lock odwyersoftware/azure-ad-verify-token#2

Closed
- renovate (bot) mentioned this issue on Apr 18, 2021

Update dependency cryptography to v3.3.2 [SECURITY] allenporter/mac-dev#3

Merged

1 task

This comment has been minimized.

Sign in to view

reaperhulk commented on Apr 21, 2021

Member

A `MemoryError` is not this bug. That exception occurs when Python runs out of memory.

- renovate (bot) mentioned this issue on May 6, 2021

Update dependency cryptography to v3.3.2 [SECURITY] - autoclosed wasimakh2/SpeedTestLogger#13

Closed

1 task

dboxers commented on May 9, 2021

Hello,

Thank you for this fine library.

I've been having some issues when encrypting large data (>2GiB) using the Fernet class. There seems to be several failure modes, I've seen everything from a segfault, SIGABRT to the decrypted plaintext differing from the original plaintext. Please note that I've had some issues with the RAM on my computer (so that could potentially be the source of *some* of the failures), but I've verified at least the SIGABRT failures on three different computers.

It seems to me that the issue seems to be an integer overflow in OpenSSL, but I'm not sure if Cryptography is at fault for passing an integer that is too large, or OpenSSL is at fault for not checking the integer, or a combination. If you think this should be fixed in OpenSSL, please let me know so I can report the issue to them.

Please see the attached script for more detailed information, as I think it speaks for itself.

1. Software versions:

- Python 3.8.2/3.9.0
- OpenSSL 1.1.1h
- cryptography: 3.3
- cfi: 1.14.4
- pip: 20.3.1
- setuptools: 51.0.0

2. Installed cryptography through pip

To reproduce:

```
import cryptography.fernet as cf

# OK
data_size = 2**31 - 1

# Fails with SIGABRT and stderr: "free(): invalid size" or "munmap_chunk(): invalid pointer"
#
# OK if _CipherContext._MAX_CHUNK_SIZE is set to 2 ** 31 - 1 - 8
data_size = 2**32 - 1

# Fails with SIGABRT or SIGSEGV and stderr: "free(): invalid size" or "munmap_chunk(): invalid pointer" or
# cryptography.exceptions.InternalError: Unknown OpenSSL error.
# This error is commonly encountered when another library is not cleaning up the OpenSSL error stack.
# If you are using cryptography with another library that uses OpenSSL try disabling it before reporting a bug.
# Otherwise please file an issue at https://github.com/pyca/cryptography/issues with information on how to reproduce this.
# ([_OpenSSL.ErrorWithText(code=101560482, lib=6, func=219, reason=162, reason_text=b'error:0600B0A2:digital envelope routines:evp_EncryptDecryptUpdate:partially overlapping buffers')])
#
# Fails due to plaintext != original_plaintext if _CipherContext._MAX_CHUNK_SIZE is set to 2 ** 31 - 1 - 8
#
# OK if _CipherContext._MAX_CHUNK_SIZE is set to 2 ** 31 - 1 - 16
data_size = 2**33 - 1

original_plaintext = bytes(data_size)

fernet = cf.Fernet(cf.Fernet.generate_key())
ciphertext = fernet.encrypt(original_plaintext)
plaintext = fernet.decrypt(ciphertext)

assert plaintext == original_plaintext
```

 **pyca** locked as resolved and limited conversation to collaborators on May 9, 2021

Assignees

No one assigned

Labels

bugs

Milestone

Thirty Fourth Release

Development

No branches or pull requests

6 participants

