New issue

Jump to bottom

ECDSA verification fails for extreme value in k and s^-1 (P-256, SHA-256) #52

Closed **adelapie** opened this issue on Apr 13, 2020 · 4 comments

adelapie commented on Apr 13, 2020

Hello,

When verifying a ECDSA signature (P-256, SHA-256) with a extreme value in k and s^-1, the verification fails even if the signature is correct. It is possible to check this using the Google Wycheproof test 345 (https://github.com/google/wycheproof/blob/master/testvectors/ecdsa_secp256r1_sha256_test.json):

```
{
      "key" : {
        "curve" : "secp256r1",
        "keySize" : 256,
        "type" : "EcPublicKey",
        "uncompressed" : "04c6a771527024227792170a6f8eee735bf32b7f98af669ead299802e32d7c3107bc3b4b5e65ab887bbd343572b3e5619261fe3a073e2ffd78412f726867db589e",
        "wx" : "00c6a771527024227792170a6f8eee735bf32b7f98af669ead299802e32d7c3107",
        "wy" : "00bc3b4b5e65ab887bbd343572b3e5619261fe3a073e2ffd78412f726867db589e"
      },
      "keyDer" :
"3059301306072a8648ce3d020106082a8648ce3d03010703420004c6a771527024227792170a6f8eee735bf32b7f98af669ead299802e32d7c3107bc3b4b5e65ab887bbd343572b3e5619261fe3a073e2ffd78412f726867db589e
      "keyPem" : "-----BEGIN PUBLIC KEY-----\nMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAExqdxUnAkIneSFwpvju5zW/Mrf5iv\nZp6tKZgC4y18MQe8O0teZauIe700NXKz5WGSYf46Bz4v/XhBL3JoZ9tYng==\n-----
END PUBLIC KEY-----",
      "sha" : "SHA-256",
      "type" : "EcdsaVerify",
      "tests" : [
        {
          "tcId" : 345,
          "comment" : "extreme value for k and s^-1",
          "msg" : "313233343030",
          "sig" : "304502207cf27b188d034f7e8a52380304b51ac3c08969e277f21b35a60b48fc47669978022100b6db6db6249249254924924924924924625bd7a09bec4ca81bcdd9f8fd6b63cc",
          "result" : "valid",
          "flags" : []
        }
      ]
    },
```

I've added a PoC using fast-ecdsa and python-cryptography (below).

```
#!/usr/bin/env python3

"""
$ python3 -m pip freeze | grep -i fastecdsa
fastecdsa==2.1.1
$ python3
Python 3.7.3 (default, Oct  7 2019, 12:56:13)
[GCC 8.3.0] on linux
"""

from fastecdsa import keys, curve, ecdsa
from fastecdsa.curve import P256
from fastecdsa.encoding.sec1 import SEC1Encoder
from fastecdsa.encoding.der import DEREncoder

from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.asymmetric import utils

from hashlib import sha256

import json
import sys
import binascii

def import_key_string(key_str: str, curve= P256, public: bool = False, decoder=SEC1Encoder):
    data = bytearray.fromhex(key_str)

    if public:
        return decoder.decode_public_key(data, curve)
    else:
        return decoder.decode_private_key(data)

"""
{
      "key" : {
        "curve" : "secp256r1",
        "keySize" : 256,
        "type" : "EcPublicKey",
        "uncompressed" : "04c6a771527024227792170a6f8eee735bf32b7f98af669ead299802e32d7c3107bc3b4b5e65ab887bbd343572b3e5619261fe3a073e2ffd78412f726867db589e",
        "wx" : "00c6a771527024227792170a6f8eee735bf32b7f98af669ead299802e32d7c3107",
        "wy" : "00bc3b4b5e65ab887bbd343572b3e5619261fe3a073e2ffd78412f726867db589e"
      },
      "keyDer" :
"3059301306072a8648ce3d020106082a8648ce3d03010703420004c6a771527024227792170a6f8eee735bf32b7f98af669ead299802e32d7c3107bc3b4b5e65ab887bbd343572b3e5619261fe3a073e2ffd78412f726867db589e
      "keyPem" : "-----BEGIN PUBLIC KEY-----\nMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAExqdxUnAkIneSFwpvju5zW/Mrf5iv\nZp6tKZgC4y18MQe8O0teZauIe700NXKz5WGSYf46Bz4v/XhBL3JoZ9tYng==\n-----
END PUBLIC KEY-----",
      "sha" : "SHA-256",
      "type" : "EcdsaVerify",
      "tests" : [
        {
          "tcId" : 345,
          "comment" : "extreme value for k and s^-1",
          "msg" : "313233343030",
          "sig" : "304502207cf27b188d034f7e8a52380304b51ac3c08969e277f21b35a60b48fc47669978022100b6db6db6249249254924924924924924625bd7a09bec4ca81bcdd9f8fd6b63cc",
          "result" : "valid",
          "flags" : []
        }
      ]
    },

"""

### using fast-ecdsa

public_key = import_key_string("04c6a771527024227792170a6f8eee735bf32b7f98af669ead299802e32d7c3107bc3b4b5e65ab887bbd343572b3e5619261fe3a073e2ffd78412f726867db589e", curve=P256,
public=True, decoder=SEC1Encoder)
decoded_r, decoded_s =
DEREncoder.decode_signature(bytearray.fromhex("304502207cf27b188d034f7e8a52380304b51ac3c08969e277f21b35a60b48fc47669978022100b6db6db6249249254924924924924924625bd7a09bec4ca81bcdd9f8fd
msg = bytearray.fromhex("313233343030")
```

```
valid = ecdsa.verify((decoded_r, decoded_s), msg, public_key)

print("Result fast-ecdsa:", valid)

### using python cryptography

curve = ec.SECP256R1()
algo = ec.ECDSA(hashes.SHA256())

pubnum = ec.EllipticCurvePublicNumbers(
    int("00c6a771527024227792170a6f8eee735bf32b7f98af669ead299802e32d7c3107", 16), int("00bc3b4b5e65ab887bbd343572b3e5619261fe3a073e2ffd78412f726867db589e", 16), curve)

data = bytes(bytearray.fromhex("313233343030"))

public_key = pubnum.public_key(default_backend())
signature =
bytes(bytearray.fromhex("304502207cf27b188d034f7e8a52380304b51ac3c08969e277f21b35a60b48fc47669978022100b6db6db6249249254924924924924924625bd7a09bec4ca81bcdd9f8fd6b63cc"))

try:
  public_key.verify(signature, data, ec.ECDSA(hashes.SHA256()))

except cryptography.exceptions.InvalidSignature:
  print("Result fast-ecdsa:", "False")
else:
  print("Result cryptography.io:", "True")
```

AntonKueltz commented on Apr 14, 2020 • edited — Owner

Thanks for raising this issue. Will have to look at intermediate values as to why this is failing, which means debugging the C extensions. Any further debug info you found will be helpful in fixing this.

Thanks,
Anton

AntonKueltz commented on Apr 14, 2020 — Owner

Found the issue, had to do with a case where the point at infinity was not handled correctly in the C code. Fix is in release `v2.1.2`.

```
➜  Desktop pyenv shell 3.7.5
➜  Desktop pip install fastecdsa==2.1.2
Collecting fastecdsa==2.1.2
  Downloading fastecdsa-2.1.2-cp37-cp37m-macosx_10_14_x86_64.whl (53 kB)
     |████████████████████████████████| 53 kB 1.6 MB/s
Installing collected packages: fastecdsa
Successfully installed fastecdsa-2.1.2
➜  Desktop ./poc.py
Result fast-ecdsa: True
Result cryptography.io: True
```

AntonKueltz closed this as completed on Apr 14, 2020

adelapie commented on May 23, 2020 • edited — Author

Hello Anton,

I got this bug assigned to CVE-2020-12607, if you want to use it, with the following description: An issue was discovered in fastecdsa before 2.1.2. When using the NIST P-256 curve in the ECDSA implementation, the point at infinity is mishandled. This means that for an extreme value in k and s^-1, the signature verification fails even if the signature is correct. This behavior is not solely a usability problem. There are some threat models where an attacker can benefit by successfully guessing users for whom signature verification will fail.

Best regards,
Antonio

AntonKueltz commented on May 26, 2020 — Owner

Hey Antonio,

Thanks for getting the CVE assigned. Just looked it up in the database and saw that it's reserved. It's probably worth fixing this via a patch release for recent older versions as well. Shouldn't take long to make those updates.

-Anton

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants