

# 2024.11.29

## web

### [SWPUCTF 2021 新生赛]easy\_md5

#### 1. 打开靶机，代码审计：

做题人:焦昱淇

题目url:<https://www.nssctf.cn/problem/386>

知识点:md5绕过、弱类型比较

将自身的源代码以HTML的形式输出到浏览器中，这样我们可以直接查看代码。

```
@include 'flag2.php';
```

它包含了一个名为 flag2.php 的文件

```
③if(isset($_GET['name'])&& isset($_POST['password'])){
    $name = $_GET['name'];
    $password = $_POST['password'];
```

a. 代码通过 `$_GET['name']` 和 `$_POST['password']` 获取了两个用户输入的值；

b. GET 用于从 URL 中获取参数，POST 用于从表单中获取参数，这里的 `$_GET['name']` 表示从 URL 中获取名为 "name" 的参数，而 `$_POST['password']` 表示从表单提交中获取名为 "password" 的参数；

```
<?php
highlight_file(__FILE__);
include 'flag2.php';

if (isset($_GET['name']) && isset($_POST['password'])) {
    $name = $_GET['name'];
    $password = $_POST['password'];
    if ($name != $password && md5($name) == md5($password)) {
        echo $flag;
    } else {
        echo "wrong!";
    }
} else {
    echo 'wrong!';
}
?>
wrong!
```

isset 函数：isset() 是一个 PHP 函数，用于检查变量是否已经设置并且非 NULL。

```

if($name != $password && md5($name) == md5($password)){
    echo $flag;
}
else {
    echo "wrong!";
}

```

如果\$name 和 \$password 不相等且是它们的MD5哈希值相等时，将输出flag,否则，将输出 "wrong!".

## 2. 审计代码得知：

- ①需要用get传递一个参数'name'，post传递一个参数'password'；
- ②传入的'name'不等于'password'且两个的MD5值相等，即可获得flag.

### 0e0跳过大法

PHP在进行“==”（弱类型比较）时，会先转换字符串类型，再进行字符串比较，而进行md5后以0e开头的都会被PHP识别为科学计数法，即0e被视作0的次方，结果都为0，故我们只需找到md5后为0e\*的字符串

字符串	对应md5值
240610708	0e462097431906509019562988736854
QLTHNDT	0e405967825401955372549139051580
QNKCZDZO	0e830400451993494058024219903391
PJNPDWY	0e291529052894702774557631701704
NWWKITQ	0e763082070976038347657360817689
NOOPCJF	0e818888003657176127862245791911
MMHUWUV	0e701732711630150438129209816536
MAUXXQC	0e478478466848439040434801845361

The screenshot shows a browser window with the URL `node7.anna.nssctf.cn:23006/?name=QLTHNDT`. The page displays the following PHP code:

```
<?php
highlight_file(__FILE__);
include 'flag2.php';
if (isset($_GET['name']) && isset($_POST['password'])) {
    $name = $_GET['name'];
    $password = $_POST['password'];
    if ($name != $password && md5($name) == md5($password)) {
        echo $flag;
    } else {
        echo "wrong!";
    }
} else {
    echo 'wrong!';
}
?>
```

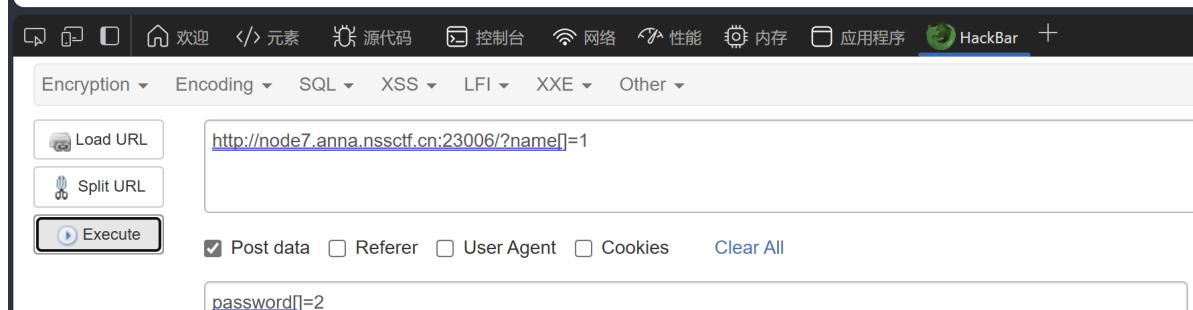
The output of the code is:  
NSSCTF{989f8e92-3fcf-48db-8136-2adc75bda20d}

The screenshot shows the HackBar extension interface. The main toolbar includes tabs for 欢迎 (Welcome), 元素 (Elements), 源代码 (Source Code), 控制台 (Console), 网络 (Network), 性能 (Performance), 内存 (Memory), 应用程序 (Applications), and HackBar. Below the toolbar, there are dropdown menus for Encryption, Encoding, SQL, XSS, LFI, XXE, and Other. The URL input field contains `http://node7.anna.nssctf.cn:23006/?name=QLTHNDT`. The Post data checkbox is checked, and the value `password=240610708` is entered in the text area below. The Execute button is highlighted with a red border.

## 空数组

```
<?php
highlight_file(__FILE__);
include 'flag2.php';

if (isset($_GET['name']) && isset($_POST['password'])) {
    $name = $_GET['name'];
    $password = $_POST['password'];
    if ($name != $password && md5($name) == md5($password)) {
        echo $flag;
    }
    else {
        echo "wrong!";
    }
}
else {
    echo 'wrong!';
}
?>
NSSCTF{989f8e92-3fcf-48db-8136-2adc75bda20d}
```



## pwn

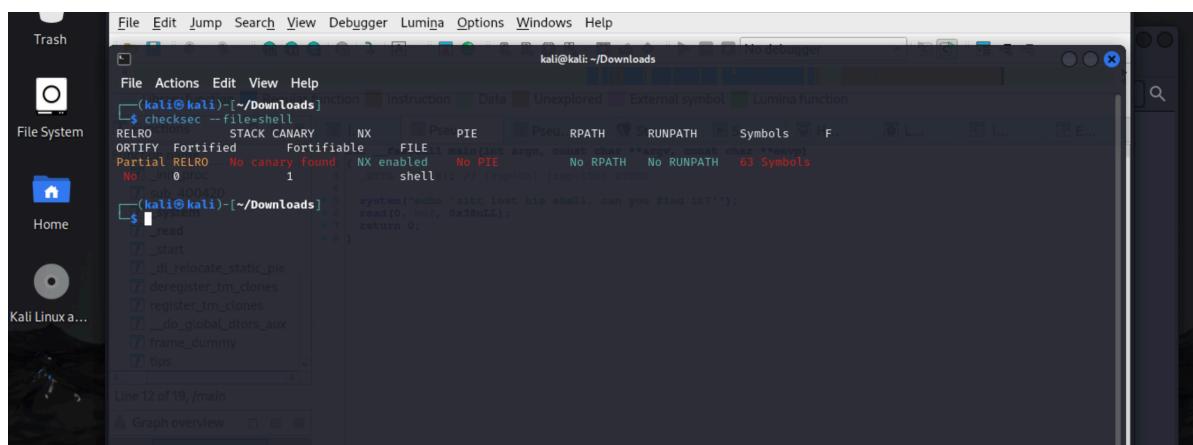
### [GFCTF 2021]where\_is\_shell

做题人:焦昱淇

题目url:<https://www.nssctf.cn/problem/889>

知识点:shell执行、exp、栈溢出

### checksec查看

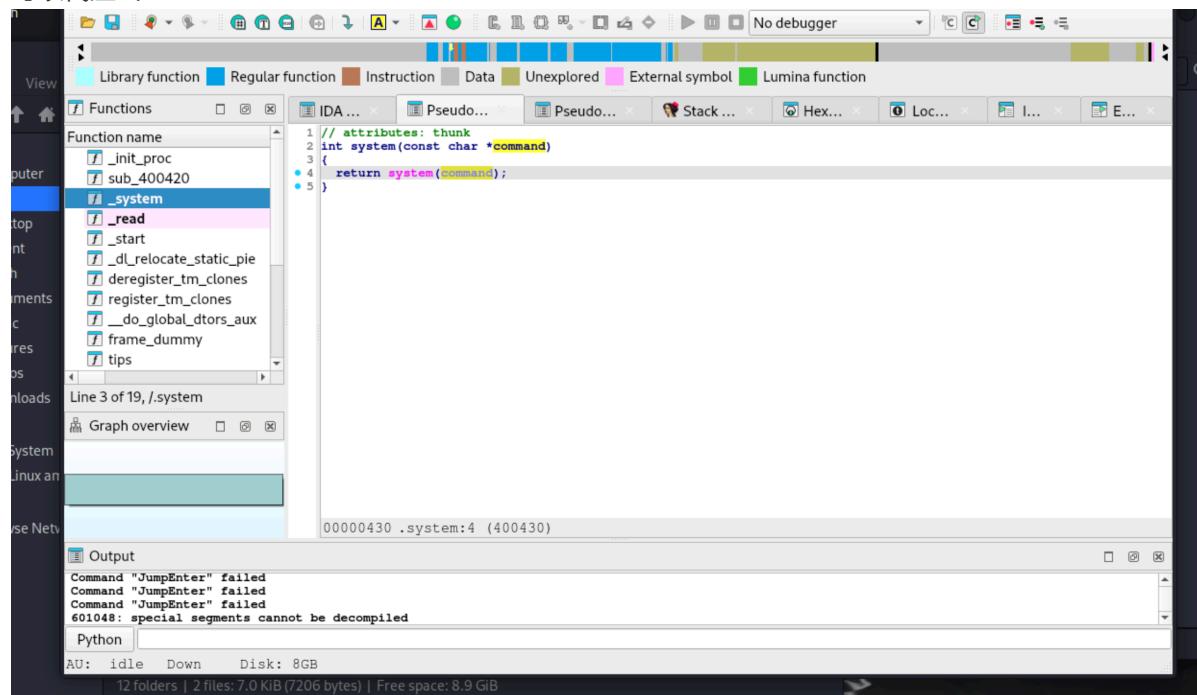


没保护、64位

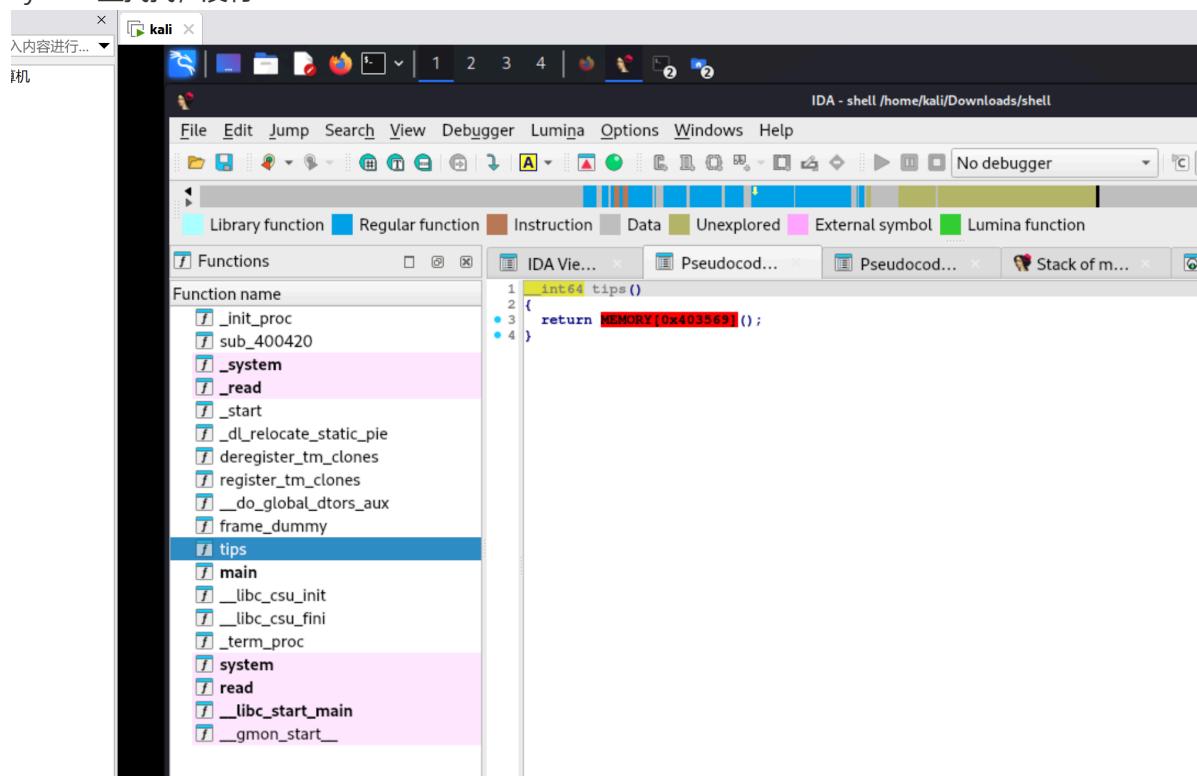
## 发现漏洞

```
_BYTE buf[16]; // [rsp+0h] [rbp-10h] BYREF  
  
system("echo 'zltt lost his shell, can you find it?'");  
read(0, buf, 0x38uLL);  
return 0;
```

可以栈溢出



system里找找，没有/bin/sh



发现tips函数里的红色提示

```

Functions          Pseudocode...    Pseudocode...    Stack of...    St...    Hex Vi...
Function name
  _init_proc
  sub_400420
  _system
  _read
  _start
  _dl_relocate_static_pie
  deregister_tm_clones
  register_tm_clones
  __do_global_dtors_aux
  frame_dummy
  tips
    main
    __libc_csu_init
    __libc_csu_fini
    _term_proc
    system
    read
    __libc_start_main
    __gmon_start_
Line 11 of 19, /tips
Output
Command "JumpEnter" failed
Command "...JumpEnter" failed
  _read
  _start
  _dl_relocate_static_pie
  deregister_tm_clones
  register_tm_clones
  __do_global_dtors_aux
  frame_dummy
  tips
    main
    __libc_csu_init
    __libc_csu_fini
    _term_proc
    system
    read
    __libc_start_main
    __gmon_start_

```

.text:0000000000400520 rep ret  
 .text:0000000000400520 \_\_do\_global\_dtors\_aux endp  
 .text:0000000000400520 ; ===== align 10h  
 .text:0000000000400522  
 .text:0000000000400530 ; ===== S U B R O U T I N E =====  
 .text:0000000000400530 ; ===== Attributes: bp-based frame  
 .text:0000000000400530 frame\_dummy proc near ; CODE XREF: \_\_libc\_csu\_init+49+p  
 .text:0000000000400530 ; DATA XREF: .init\_array:\_frame\_  
 .text:0000000000400530 push rbp  
 .text:0000000000400530 mov rbp, rsp  
 .text:0000000000400534 pop rbp  
 .text:0000000000400535 jmp short register\_tm\_clones  
 .text:0000000000400535 frame\_dummy endp  
 .text:0000000000400535  
 .text:0000000000400537 ; ===== S U B R O U T I N E =====  
 .text:0000000000400537 ; ===== Attributes: bp-based frame  
 .text:0000000000400537 ; ===== public tips =====  
 .text:0000000000400537 tips proc near  
 .text:0000000000400537 ; \_ unwind {  
 .text:0000000000400537 push rbp  
 .text:0000000000400538 mov rbp, rsp  
 .text:0000000000400538 mov eax, 0  
 .text:0000000000400540 call near ptr 403569  
 .text:0000000000400545 nop  
 .text:0000000000400546 pop rbp  
 .text:0000000000400547 ret  
 .text:0000000000400547 ; } // starts at 400537  
 .text:0000000000400547 tips endp  
 .text:00000537 0000000000400537: tips

24 30对应\$0也可以ssssssssshell执行

取出shell的地址是0x400541

还要加上ret 和 pop|rdi|ret这个语句

ret 是一个汇编指令，用于函数的返回操作。当程序执行到 ret 指令时，它会从栈中弹出一个地址，并跳转到该地址处，继续执行代码。在函数调用过程中，当一个函数执行完毕时，它会通过 ret 指令返回到调用该函数的位置。这个返回地址通常是在函数调用时被压入栈中的。

"pop rdi; ret" 是一种常见的ROP (Return Oriented Programming) gadget，用于构建ROP攻击时的 payload。在x86\_64架构中，这个gadget的作用是从栈中弹出一个数值，并将其赋值给RDI寄存器，然后进行返回。

在ROP攻击中，攻击者利用程序中存在的这种gadget，通过精心构造的数据将程序控制流引导到这个 gadget，从而实现对程序行为的控制。通常情况下，攻击者会将需要的参数放置在栈上，然后利用ROP 链将控制流引导到 "pop rdi; ret" 这样的gadget，将参数加载到合适的寄存器中，然后再执行调用目标函数的ret指令

```

$ ROPgadget --binary shell --only "pop|ret"
Gadgets information
=====
[+] _start          usage: ROPgadget [-h] [-v] [-c] [--binary <binary>] [--opcode <pcodes>] [--string <string>] [--memstr <memstr>] [--rawArch <arch>] [--rawEndian <endian>] [--offset <hexaddr>] [--ropchain] [--thumb] [--conservative] [--callPpreceded] [--nosys] [--multibr] [--all] [--noinstr] [--dump] [--silent] [--align]
[+] _dl_relocate    ROPgadget: error: unrecognized arguments: --binaryshell
[+] deregister_tm_clones
[+] register_tm_clones
[+] _do_global
[+] frame_dumper
[+] tips
[+] main
[+] __libc_csu_init
[+] __libc_csu_fini
[+] _termproc
[+] system
[+] read
[+] __libc_start_main
[+] __gmon_start

Unique gadgets found: 11
$ 

```

找到白色部分的地址

写脚本，运行

