看我如何玩转PHP代码加密与解密

转载

写回 于 2019-08-20 10:43:13 发布 1363 收藏 3
分类专栏： php 代码加密 文章标签： php 代码加密
原文链接：https://xz.aliyun.com/t/2403
版权


php 同时被 2 个专栏收录
60 篇文章 0 订阅
订阅专栏


代码加密
1 篇文章 0 订阅
订阅专栏

参考文献：

https://xz.aliyun.com/t/2403

前言

两次比赛，两个题目，两种方式，两个程序。

一切PHP的代码终究是要到**Zend Engine**上走一走的，因此一切PHP的源码加密都是可以被解密的。（不包括OpCode混淆-VMP）

代码混淆

比较恶心人的一种处理方式，也不太算是加密。

单独拿出来是为了说明代码混淆和代码加密是两种方式。

本质是是对变量进行乱七八糟的修改，多用动态函数处理。处理应该没什么难度，就是比较复杂，浪费时间精力。

混淆方式是按照套路随机生成相关动态函数，替换明文函数，然后批量修改变量名。

该法常与代码加密联合使用。

代码加密解密

用PHP代码进行PHP代码的加密，套了层壳。大多数代码加密都进行了一定的代码混淆，不同的加密工具也有不同的混淆。

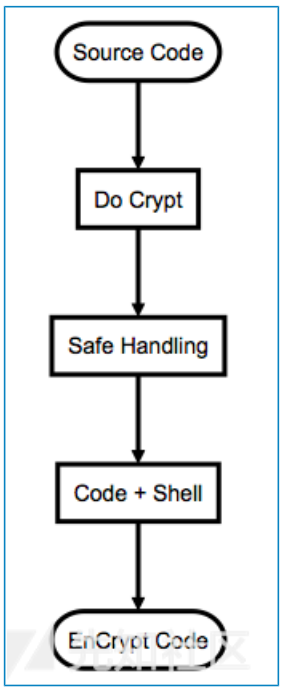
- 壳混淆
- 代码混淆
- 壳和代码都分别混淆

常见加密工具

- phpjiami

加密

源码 -> 加密处理（压缩，替换，BASE64，转义）-> 安全处理（验证文件 MD5 值，限制 IP、限域名、限时间、防破解、防命令行调试）-> 加密程序成品，再简单的说：密文源码 + 自解密外壳 == 密文代码



加密方式

1. 独立加密程序统一对明文代码进行加密处理

解密

加密也好，混淆也罢，终归是要变成**Zend Engine**能处理的源码，该“加密”方法的的根本是通过壳把代码解密并通过`eval`等函数执行代码。

因此，只要用**HOOK EVAL**大法，将相关可执行代码的函数Hook住就能拿到其中需要执行的数据，也就是我们想要得到的源码。

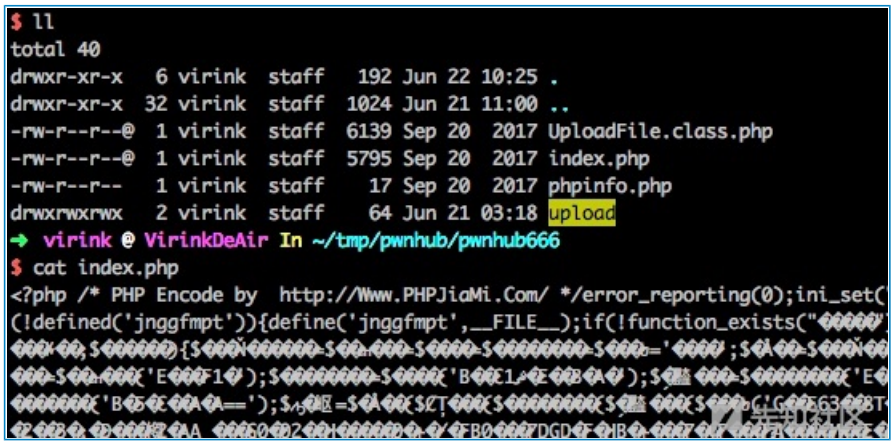
调用`eval`等代码执行的函数，最终会调用**PHP内核**的`zend_compile_string`函数。

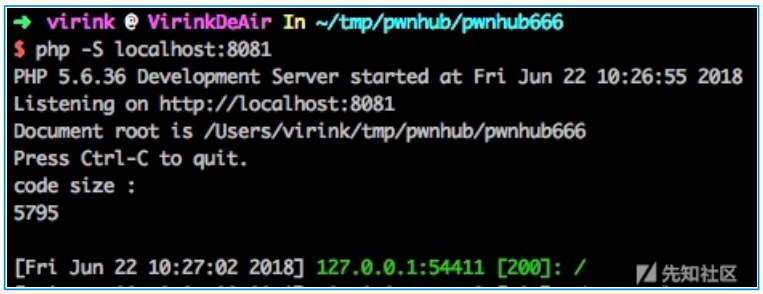
通过PHP本身提供的一个HOOK机制，写个插件轻松搞定。

```
// 声明一个临时的 compile_string 函数
static zend_op_array *(*orig_compile_string)(zval *source_string, char *filename TSRMLS_DC);
// 在 PHP_MINIT_FUNCTION 中替换
orig_compile_string = zend_compile_string;
zend_compile_string = phpjiami_decode_compile_string;
// 在 PHP_MSHUTDOWN_FUNCTION 中恢复
zend_compile_string = orig_compile_string;
// 提取 compile_string 中的代码并保存
static zend_op_array *phpjiami_decode_compile_string(zval *source_string, char *filename TSRMLS_DC)
{
    int c, len, yes;
    char *content;
    FILE *fp = NULL;
    char fn[512];
    if (Z_TYPE_P(source_string) == IS_STRING) {
        len  = Z_STRLEN_P(source_string);
        content = estrndup(Z_STRVAL_P(source_string), len);
        if (len > strlen(content))
            for (c=0; c<len; c++)
                if (content[c] == 0)
                    content[c] = '?';
        sprintf(fn, "/tmp/%s.php", zend_get_executed_filename(TSRMLS_C));
        fp = fopen(fn,"a+");
        if (fp!=NULL)
            fprintf(fp, "<?php\n%s\n?>\n\n", content);
        fclose(fp);
    }
    return orig_compile_string(source_string, filename TSRMLS_CC);
}
```

案例

Challenge: PWNHUB 公开赛 / 傻 fufu 的工作日 Writeup





```
$ ll
total 56
drwxr-xr-x   7 virink  staff   224 Jun 22 10:27 .
drwxr-xr-x  32 virink  staff  1024 Jun 21 11:00 ..
-rw-r--r--@  1 virink  staff  6139 Sep 20  2017 UploadFile.class.php
-rw-r--r--   1 virink  staff  4511 Jun 22 10:27 index.eval.php
-rw-r--r--@  1 virink  staff  5795 Sep 20  2017 index.php
-rw-r--r--   1 virink  staff    17 Sep 20  2017 phpinfo.php
drwxrwxrwx   2 virink  staff    64 Jun 21 03:18 upload
➜  virink @ VirinkDeAir In ~/tmp/pwnhub/pwnhub666
$ cat index.eval.php
<?php

?>

<?php
if(strpos(__FILE__, jnqgmpt) !== 0){$exitfunc();}
?>

<?php
eval(base64_decode($[illegible]));
?>

<?php
?><?php @eval("//Encode by  phpjiami.com,Free user."); ?><?php
if($_FILES) {
    include 'UploadFile.class.php';
    $dist = 'upload';
    $upload = new UploadFile($dist, 'upfile');
    $data = $upload->upload();
}
?><!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-wid[illegible]
    <title>pwnhub6669</title>
    <link rel="stylesheet" href="assets/bootstrap/css/bootstrap.min.css">
```


扩展加密解密

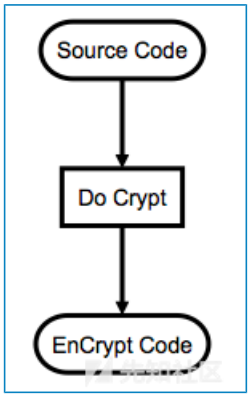
将文本源码进行加密存储，在使用的时候通过扩展实现解密。

常见加密工具

- pm9screw
- pm9screw_plus

加密

源码 -> 加密处理（对称/非对称加密、自定义加密）-> 加密成品：密文代码



加密方式

1. 独立加密程序统一对明文代码进行加密处理

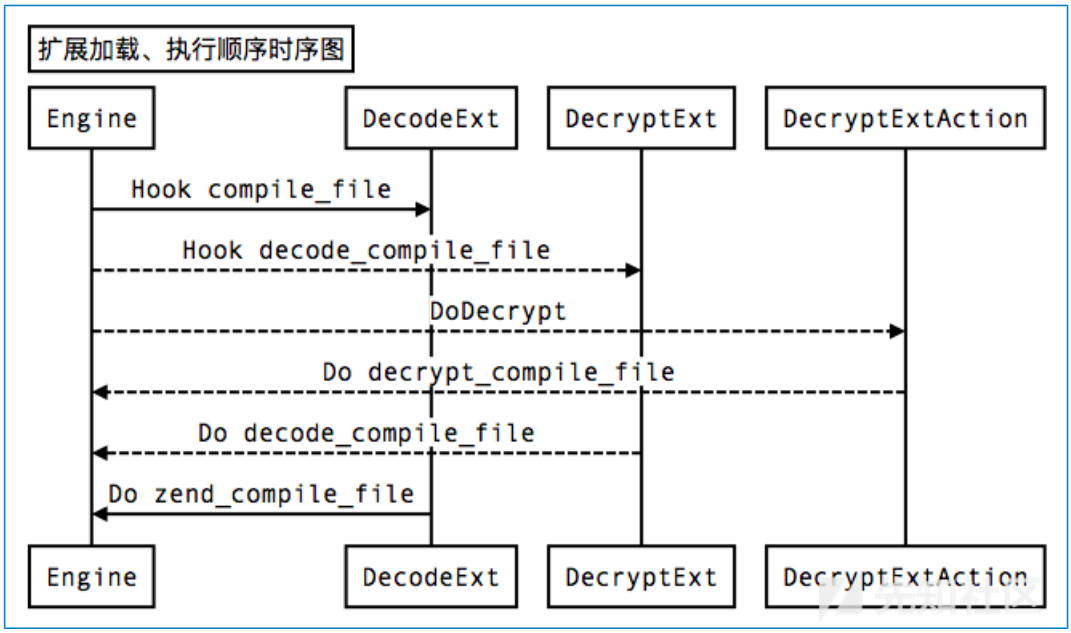
2. 扩展存在加密解密功能，执行前判断源码是否经过加密处理，如果没有就进行加密

解密

还是那句话，一切的源码都要到**Zend Engine**上执行，密文也得解密了再执行。那么在最终的执行之前，提取出来就可以了。

因此Hook住`zend_compile_file`函数就可以了。

但是其中有一个坑点，PHP的扩展是“栈”加载的，也就是先加载的先Hook，后执行。我们需要获取到解密之后的内容，所以需要让“加密”插件先执行，也就是我们的解密插件要先加载。



要实现这个操作，只需要在INI配置文件中先写我们的插件。(不保证)

```
extension="decode.so"
extension="encrypt.so"
```

这个方式需要能够加载执行**encrypt.so**，我觉得这个还是可以实现的。通过一定手段获取到**encrypt.so**和密文源码以及服务器，中间件相关信息（版本等）。

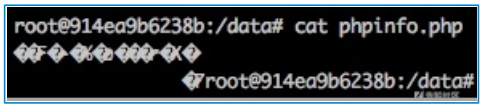
利用Docker运行一个基本相同的环境应该是可以做到的。

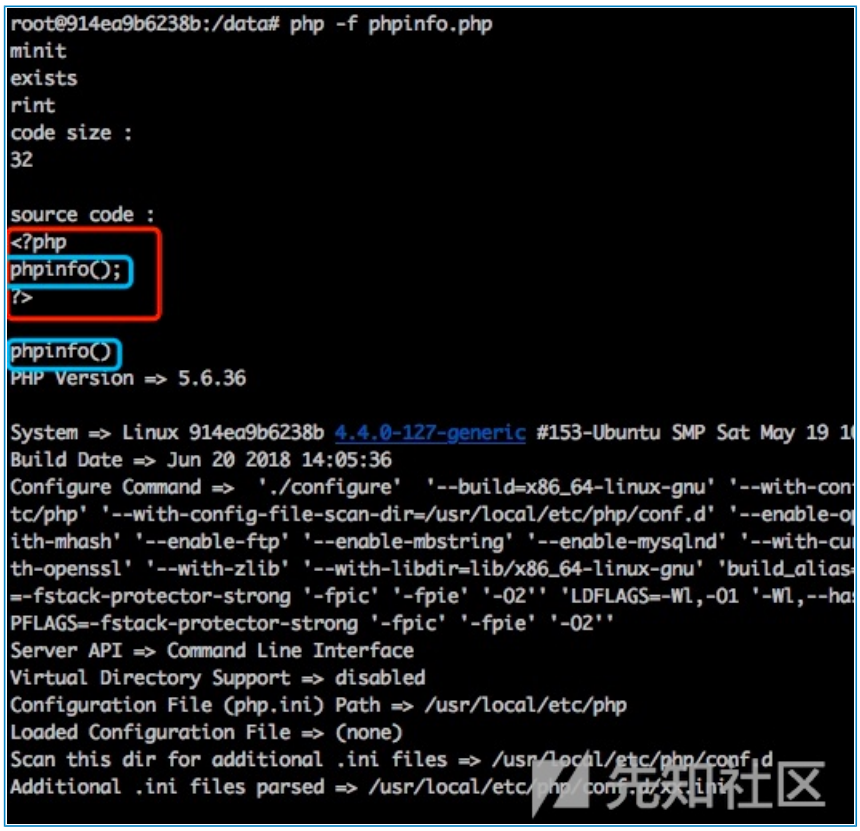
```
// 声明一个临时的 compile_file 函数
static zend_op_array *(*orig_compile_file)(zend_file_handle *file_handle, int type TSRMLS_DC);
// 在 PHP_MINIT_FUNCTION 中替换
orig_compile_file = zend_compile_file;
zend_compile_file = phpjiami_decode_compile_file;
// 在 PHP_MSHUTDOWN_FUNCTION 中恢复
zend_compile_file = orig_compile_file;
// 提取 compile_file 中的代码并保存
static zend_op_array *phpjiami_decode_compile_file(zend_file_handle *file_handle, int type TSRMLS_DC){
    char *buf;
    size_t size;
    if (zend_stream_fixup(file_handle, &buf, &size TSRMLS_CC) == SUCCESS) {
        FILE *ff = NULL;
        int i=0;
        php_printf("code size :\n%d\n\nsource code :\n%s\n\n", size, buf);
        ff = fopen("/tmp/decode.php","a+");
        if (ff!=NULL)
            for(i = 0; i <= size; i++)
                fprintf(ff, "%c", buf[i]);
        fclose(ff);
    }
    return orig_compile_file(file_handle,type TSRMLS_DC);
}
```

案例

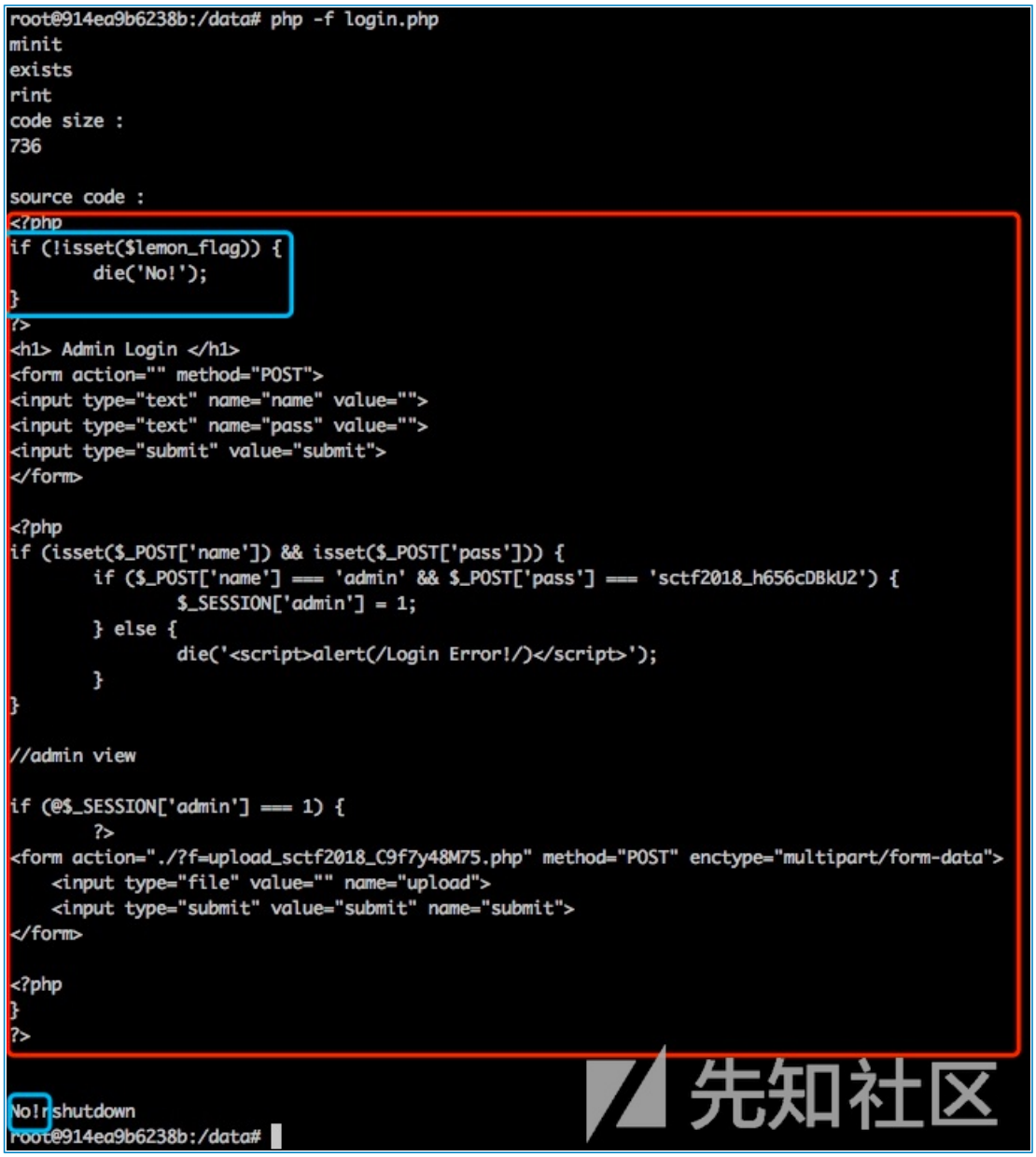
Challenge: SCTF2018 BabySyc - Simple PHP Web Writeup - L3m0n

- phpinfo





- login.php



OpCode混淆

一种是比如**Swoole Compile**的方式，部分脱离了zend虚拟机，对opcode做了混淆，这就比较像是vmp的一种方式。

加密方式

1. 独立加密程序统一对明文代码进行加密处理（猜测）

解密

我不会啊！emmmmmmm

参考

1. phpjiami 数种解密方法 - PHITHON
2. Decrypt php VoiceStar encryption extension - 小鹿师傅
3. PHPDecode 在线解密工具 - Medici.Yan
4. Decoding a User Space Encoded PHP Script - Stefan Esser
5. PHP代码加密技术 郭新华 PHPCON2018 - swoole郭新华