

# 解析超经典的 Linux Fork 炸弹

Linux学习 今天

来自: Magic

链接: <https://blog.saymagic.cn/2015/03/25/fork-bomb.html>

Jaromil在2002年设计了最为精简的一个Linux Fork炸弹,整个代码只有13个字符,在shell中运行后几秒后系统就会宕机:

```
:() { :|:& };;:
```

这样看起来不是很好理解,我们可以更改下格式:

```
:()  
{  
  :|:&  
};  
:
```

更好理解一点的话就是这样:

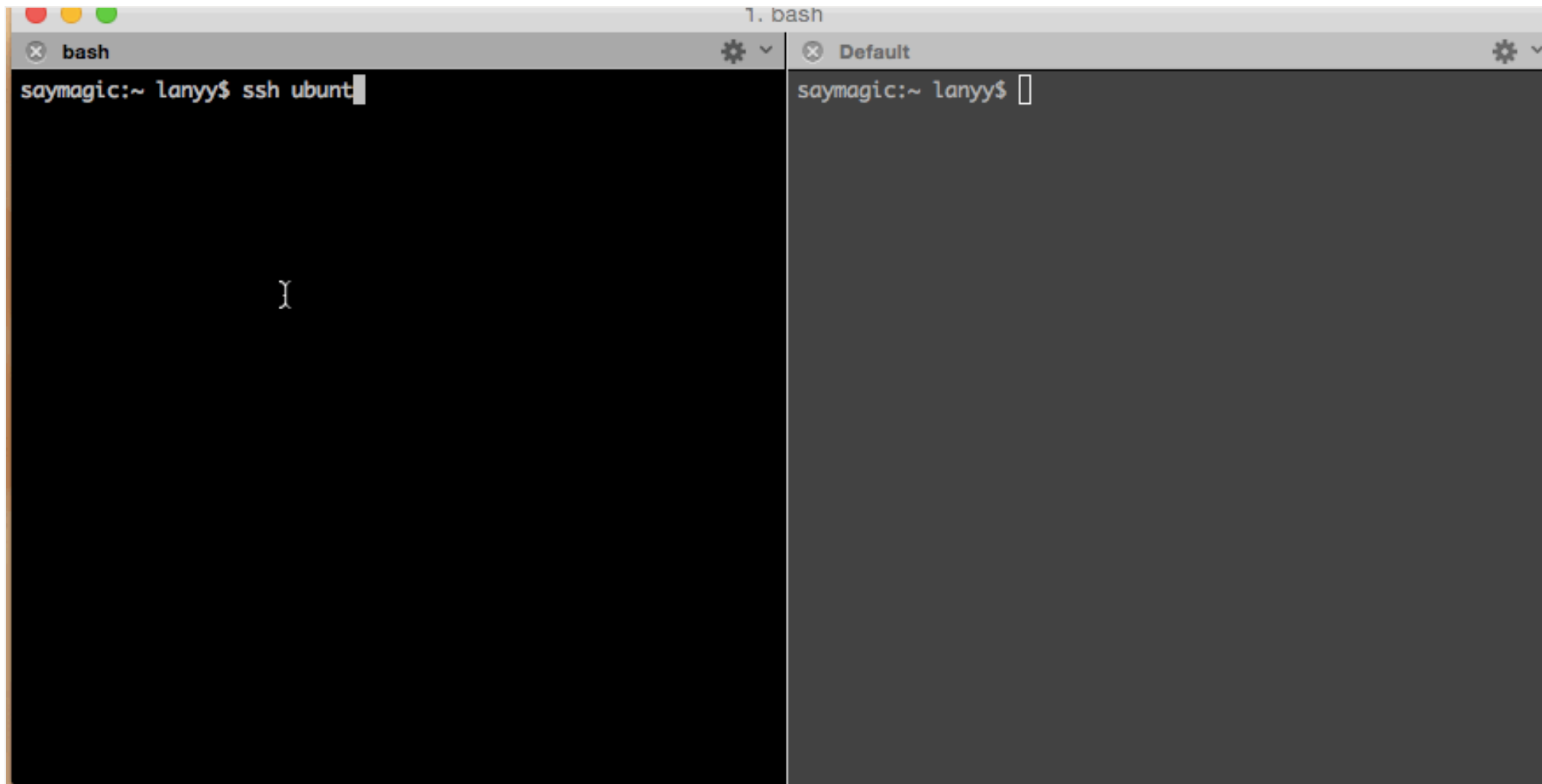
```
bomb()  
{  
  bomb|bomb&
```

```
};  
bomb
```

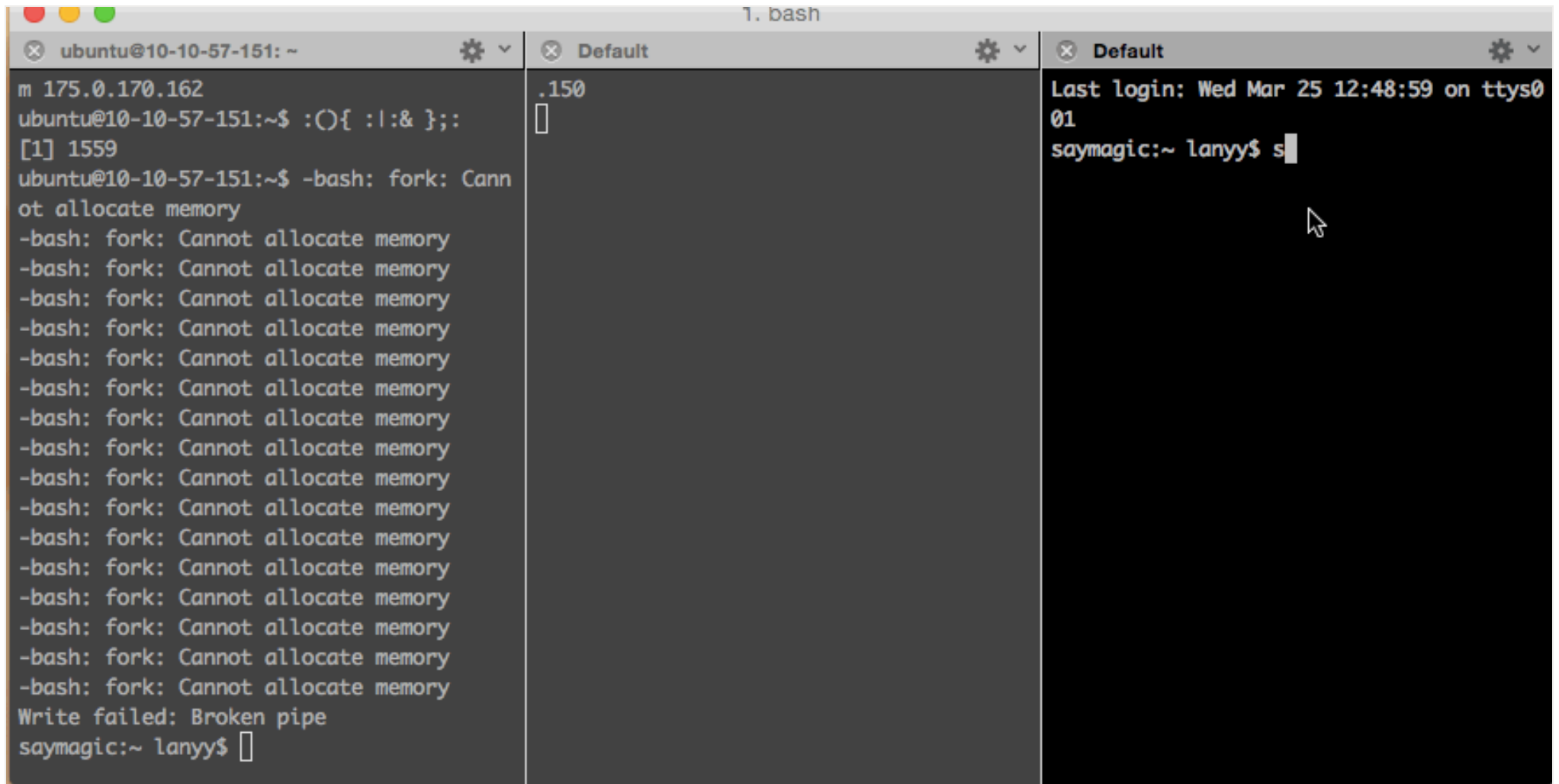
因为shell中函数可以省略 `function` 关键字，所以上面的十三个字符是功能是定义一个函数与调用这个函数，函数的名称为 `:`，主要的核心代码是 `:|:&`，可以看出这是一个函数本身的递归调用，通过 `&` 实现在后台开启新进程运行，通过管道实现进程呈几何形式增长，最后再通过 `:` 来调用函数引爆炸弹。因此，几秒钟系统就会因为处理不过来太多的进程而死机，解决的唯一办法就是重启。

## Bomb一下

秉着不作不死的心态，我们也来运行一下，于是我将矛头指向云主机，我使用了国内的一个2G内存的云主机，首先在本地开启两个终端，在一个终端连接云主机后运行炸弹，几秒后再尝试用另外一个终端登录，效果可以看下面Gif图：



看，运行一段时间后直接报出了 `-bash: fork: Cannot allocate memory`，说明内存不足了。并且我在二号终端上尝试连接也没有任何反应。因为是虚拟的云主机，所以我只能通过主机服务商的后台来给主机断电重启。然后才能重新登录：



```
ubuntu@10-10-57-151: ~  
m 175.0.170.162  
ubuntu@10-10-57-151:~$ :(){ :|:& };;  
[1] 1559  
ubuntu@10-10-57-151:~$ -bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
-bash: fork: Cannot allocate memory  
Write failed: Broken pipe  
saymagic:~ lanyy$  
  
1. bash  
.150  
  
Last login: Wed Mar 25 12:48:59 on ttys001  
saymagic:~ lanyy$ s
```

## 炸弹危害

Fork炸弹带来的后果就是耗尽服务器资源，使服务器不能正常的对外提供服务，也就是常说的DoS(Denial of Service)。与传统1v1、通过不断向服务器发送请求造成服务器崩溃不同，Fork炸弹有种坐山观虎斗，不费一兵一卒斩敌人于马下的感觉。更吓人的是这个函数是不需要root权限就可以运行的。看到网上有帖子说某些人将个性签名改为Fork炸弹，结果果真好奇之人中枪，试想如果中枪的人是在公司服务器上运行的话，oh，！

## 预防方式

当然，Fork炸弹没有那么可怕，用其它语言也可以分分钟写出来一个，例如，python版：

```
import os
while True:
    os.fork()
```

Fork炸弹的本质无非就是靠创建进程来抢占系统资源，在Linux中，我们可以通过 `ulimit` 命令来限制用户的某些行为，运行 `ulimit -a` 可以查看我们能做哪些限制：

```
ubuntu@10-10-57-151:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 7782
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 7782
```

```
virtual memory      (kbytes, -v) unlimited
file locks          (-x) unlimited
```

可以看到，`-u` 参数可以限制用户创建进程数，因此，我们可以使用 `ulimit -u 20` 来允许用户最多创建20个进程。这样就可以预防 `bomb` 炸弹。但这样是不彻底的，关闭终端后这个命令就失效了。我们可以通过修改 `/etc/security/limits.conf` 文件来进行更深层次的预防，在文件里添加如下一行（`ubuntu`需更换为你的用户名）：

```
ubuntu - nproc 20
```

这样，退出后重新登录，就会发现最大进程数已经更改为20了，

```
ubuntu@10-10-57-151:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 7782
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 20
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

这个时候我们再次运行炸弹就不会报内存不足了，而是提示 `-bash: fork: retry: No child processes`，很棒，此时说明Linux限制了炸弹创建线程。

## 参考

[http://en.wikipedia.org/wiki/Fork\\_bomb](http://en.wikipedia.org/wiki/Fork_bomb)

--- EOF ---

推荐↓↓↓



运维

分享网络管理、网络运维、运维规划、运维开发、Python运维、Linux运维等知识，推广围绕DevOps理念的自动化运维、精益运维、... >  
1篇原创内容

公众号

阅读原文

喜欢此内容的人还喜欢

腾讯 Code Review 规范出炉!

码农有道



百度C++工程师的那些极限优化（内存篇）

百度Geek说



这才是牛逼程序员的标配!

程序员数学之美

