How to Restrict a Login Shell Using Linux Namespaces

Firejail is a SUID sandbox program that reduces the risk of security breaches by restricting the running environment of untrusted applications using Linux namespaces. It allows a process and all its descendants to have their own private view of the globally shared kernel resources, such as the network stack, process table, mount table.

Started as a simple sandbox for Mozilla Firefox, Firejail was expanded to work on any type of executable, such as servers, graphic programs, and even as login shell.

The program is written in C and only needs libc and POSIX threads (libpthreads), available by default on any Linux platform. The download page provides source code (*./configure && make && sudo make install*), deb (*dpkg -i firejail.deb*) and rpm (*rpm -i firejail.rpm*) packages. Once installed, you can start a program in sandbox as:

```
$ firejail [options] program and arguments
Example:
$ firejail --debug firefox
```

Default sandbox

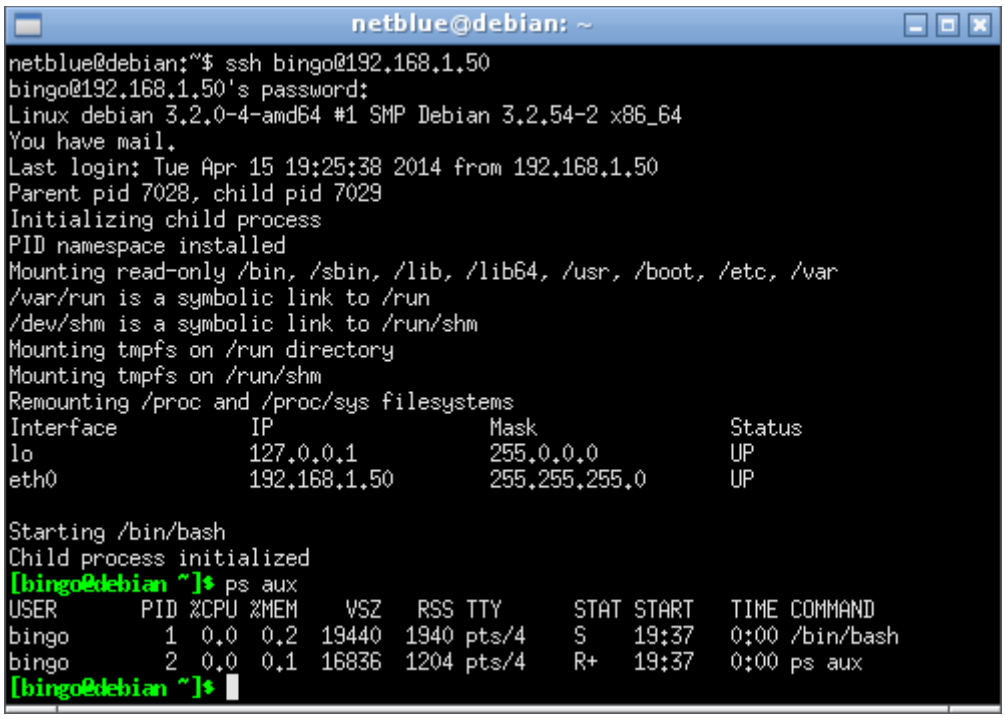
To login into a Firejail sandbox, you need to set */usr/bin/firejail* as user shell in */etc/passwd*. You can change the shell for an existing user with chsh command:

```
# chsh --shell /usr/bin/firejail
```

Another option is to define the shell when the user account is created:

```
# adduser --shell /usr/bin/firejail username
```

Below is a ssh login session into a sandboxed account:



SSH login into a default Firejail sandbox

This is a default sandbox. The following directories are mounted read-only: */bin, /sbin, /lib, /lib64, /usr, /boot, /etc*, and */var*. Only */home* and */tmp* directory are mounted read-write. The sandbox also starts a new process namespace, where our bash shell becomes PID 1. No external processes are visibile in the sandbox.

Adding a network namespace to the sandbox

The default sandbox uses the same TCP/IP network stack as the host. We can configure the sandbox to use a different networking stack using *–net* option. Firejail grabs extra arguments for the specific user from */etc/firejail/login.users* file:

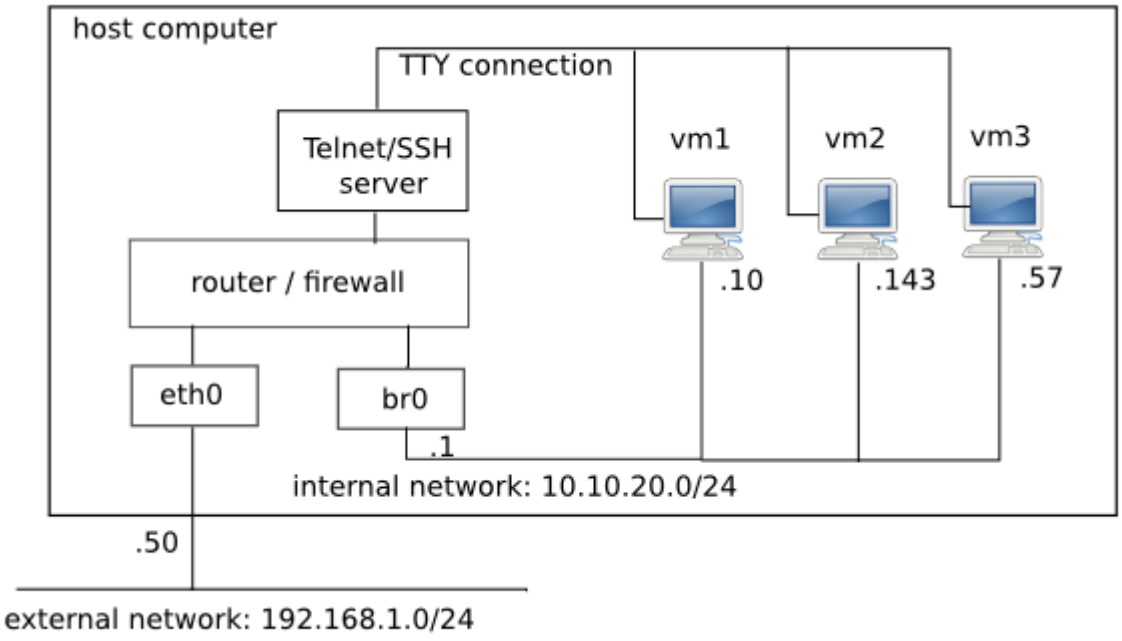
```
$ cat /etc/firejail/login.users
# Each user entry consists of a user name and firejail
# program arguments:
#
#       user name: arguments
#
# The extra arguments are inserted into program command line if firejail
```

```
# was started as a login shell.
bingo: --debug --net=br0
```

br0 is a standard Linux bridge device. It has to be created and initialized in order for login to succeed.

```
# brctl addbr br0
# ifconfig br0 10.10.20.1/24
```

Users can create as many sandboxes as needed, one for each SSH login. Each sandbox has its own TCP/IP network stack connected to the host on *br0* device. IP addresses can be assigned manually or automatically. A regular /24 network allows 253 independent IP addresses. The addresses reported by *ifconfig* in the sandbox are different than the addresses used for SSH login.



Network setup

Controlling network traffic

The traffic between 10.10.20.0/24 network and 192.168.1.0/24 is routed by the host networking stack. The router and the firewall in Linux kernel provide a mechanism to control this traffic.

If we need our users not to be able to go out from the internal network, all we have to do is to disable routing in the host:

```
# echo "0" > /proc/sys/net/ipv4/ip_forward
#
```

If instead, we need to give our users full network access, we enable routing and network address translation (NAT) on our host:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o eth0 -s 10.10.20.0/24  -j MASQUERADE
#
```

Conclusion

This is by far the simplest form of virtualization available on Linux platform. Instead of assigning users fully fledged virtual machines, we assign them Telnet or SSH login shells restricted using Linux namespaces. It is cheap on memory resources and it is easy to configure.

This is a comparison of a regular SSH login and a restricted SSH login:

	Regular login	Restricted login
Filesystem	Read-write	Configurable, mostly read-only
Process table	Access to all running processes	Access only to processes started in this login session
Network	Full local network access	Controlled local network access

The same solid server security practices are required in both the regular and the restricted login case. An attacker gaining unauthorized root access is bad in both cases.

Source : http://l3net.wordpress.com/2014/04/16/how-to-restrict-a-login-shell-using-linux-namespaces/