HOW TO SECURE YOUR TRIXBOX SERVER

by Kerry Garrison | March 2009 | Linux Servers Networking & Telephony Open Source

Even though a trixbox system is a phone system, it is still a basic computer system like any other. One of the problems that we face is that extensions and VoIP service providers typically come into the system over the open Internet; this means that certain aspects of our system are wide open to the outside world. During the week that this article was written, several new scripts came out that allowed people to scan machines over the Internet, find systems that are running Asterisk, get the list of available extensions, and then hack the passwords. These tools allow a malicious hacker to get into your system and start making long-distance phone calls. There were numerous instances of companies with phone bills reaching into the thousands and even tens of thousands of dollars. Because of issues like this, it is more imperative than ever that you understand how to properly secure your trixbox server from the outside world. In this article by **Kerry Garrison**, we will focus on how to secure the trixbox server.

Start with a good firewall

Never have your trixbox system exposed completely on the open Internet; always make sure it is behind a good firewall. While many people think that because trixbox is running on Linux, it is totally secure, Linux, like anything else, has its share of vulnerabilities, and if things are not configured properly, is fairly simple for hackers to get into. There are really good open-source firewalls available, such as *pfSense*, *Viata*, and*M0n0Wall*. Any access to system services, such as HTTP or SSH, should only be done via a VPN or using a pseudo-VPN such as Hamachi. The best designed security starts with being exposed to the outside world as little as possible. If we have remote extensions that cannot use VPNs, then we will be forced to leave SIP ports open, and the next step will be to secure those as well.

Stopping unneeded services

Since trixbox CE is basically a stock installation of CentOS Linux, very little hardening has been done to the system to secure it. This lack of security is intentional as the first level of defence should always be a good firewall. Since there will be people who still insist on putting the system in a data center with no firewall, some care will need to be taken to ensure that the system is as secure as possible. The first step is to disable any services that are running that could be potential security vulnerabilities.

We can see the list of services that are used with the *chkconfig –list* command.

```
[trixbox1.localdomain rules]# chkconfig --list
anacron 0:off 1:off 2:on 3:on 4:on 5:on 6:off
asterisk 0:off 1:off 2:off 3:off 4:off 5:off 6:off
avahi-daemon 0:off 1:off 2:off 3:off 4:off 5:off 6:off
avahi-dnsconfd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
bgpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
capi 0:off 1:off 2:off 3:off 4:off 5:off 6:off
crond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
dc_client 0:off 1:off 2:off 3:off 4:off 5:off 6:off
dc_server 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

```
dhcpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
dhcrelay 0:off 1:off 2:off 3:off 4:off 5:off 6:off
ez-ipupdate 0:off 1:off 2:off 3:off 4:off 5:off 6:off
haldaemon 0:off 1:off 2:off 3:on 4:on 5:on 6:off
httpd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
ip6tables 0:off 1:off 2:off 3:off 4:off 5:off 6:off
iptables 0:off 1:off 2:off 3:off 4:off 5:off 6:off
isdn 0:off 1:off 2:off 3:off 4:off 5:off 6:off
kudzu 0:off 1:off 2:off 3:on 4:on 5:on 6:off
lm_sensors 0:off 1:off 2:on 3:on 4:on 5:on 6:off
lvm2-monitor 0:off 1:on 2:on 3:on 4:on 5:on 6:off
mDNSResponder 0:off 1:off 2:off 3:on 4:on 5:on 6:off
mcstrans 0:off 1:off 2:off 3:off 4:off 5:off 6:off
mdmonitor 0:off 1:off 2:on 3:on 4:on 5:on 6:off
mdmpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
memcached 0:off 1:off 2:on 3:on 4:on 5:on 6:off
messagebus 0:off 1:off 2:off 3:on 4:on 5:on 6:off
multipathd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
mysqld 0:off 1:off 2:off 3:on 4:on 5:on 6:off
named 0:off 1:off 2:off 3:off 4:off 5:off 6:off
netconsole 0:off 1:off 2:off 3:off 4:off 5:off 6:off
netfs 0:off 1:off 2:off 3:on 4:on 5:on 6:off
netplugd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
network 0:off 1:off 2:on 3:on 4:on 5:on 6:off
nfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off
nfslock 0:off 1:off 2:off 3:on 4:on 5:on 6:off
ntpd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
ospf6d 0:off 1:off 2:off 3:off 4:off 5:off 6:off
ospfd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
portmap 0:off 1:off 2:off 3:on 4:on 5:on 6:off
postfix 0:off 1:off 2:on 3:on 4:on 5:on 6:off
rdisc 0:off 1:off 2:off 3:off 4:off 5:off 6:off
restorecond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
ripd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
ripngd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

```
rpcgssd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
rpcidmapd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
rpcsvcgssd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
saslauthd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
snmpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
snmptrapd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
sshd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
syslog 0:off 1:off 2:on 3:on 4:on 5:on 6:off
vsftpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
xinetd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
zaptel 0:off 1:off 2:on 3:on 4:on 5:on 6:off
zebra 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

The highlighted lines are services that are started automatically on system startup. The following list of services is required by trixbox CE and should not be disabled:

- Anacron
- crond
- haldaemon
- httpd
- kudzu
- lm_sensors
- lvm2-monitor
- mDNSResponder
- mdmonitor
- memcached
- messagebus
- mysqld
- network
- ntpd
- postfix
- sshd
- syslog
- xinetd
- zaptel

To disable a service, we use the command *chkconfig <servicename> off*. We can now turn off some of the services that are not needed:

```
chkconfig ircd off
chkconfig netfs off
chkconfig nfslock off
chkconfig openibd off
chkconfig portmap off
chkconfig restorecond off
chkconfig rpcgssd off
chkconfig rpcidmapd off
chkconfig vsftpd off
```

We can also stop the services immediately without having to reboot:

```
service ircd stop
service netfs stop
service nfslock stop
service openibd stop
service portmap stop
service restorecond stop
service rpcgssd stop
service rpcidmapd stop
service vsftpd stop
```

Securing SSH

A very large misconception is that by using SSH to access your system, you are safe from outside attacks. The security of SSH access is only as good as the security you have used to secure SSH access. Far too often, we see systems that have been hacked because their root password is very simple to guess (things like *password* or *trixbox* are not safe passwords). Any dictionary word is not safe at all, and substituting numbers for letters is very poor practice as well. So, as long as SSH is exposed to the outside, it is vulnerable. The best thing to do, if you absolutely have to have SSH running on the open Internet, is to change the port number used to access SSH. This section will detail the best methods of securing your SSH connections.

Create a remote login account

First off, we should create a user on the system and only allow SSH connections from it. The username should be something that only you know and is not easily guessed. Here, we will create a user called trixuser and assign a password to it. The password should be something with letters, numbers, symbols, and not based on a dictionary word. Also, try to string it into a sentence making sure to use the letters, numbers, and symbols. Spaces in passwords work well too, and are hard to add in scripts that might try to break into your server.

A nice and simple tool for creating hard-to-guess passwords can be found at http://www.pctools.com/guides/password/.

```
[trixbox1.localdomain init.d]# useradd trixuser
[trixbox1.localdomain init.d]# passwd trixuser
```

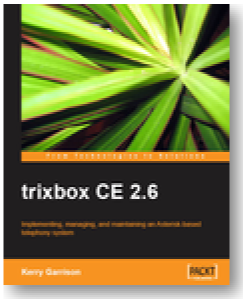
Now, ensure that the new account works by using SSH to log in to the trixbox CE server with this new account. If it does not let you in, make sure the password is correct or try to reset it. If it works, continue on.

Only allowing one account access to the system over SSH is a great way to lock out most brute force attacks. To do this, we need to edit the file in */etc/ssh/sshd_config* and add the following to the file.

```
AllowUsers trixuser
```

The *PermitRootLogin* setting can be edited so that root can't log in over SSH. Remove the # from in front of the setting and change the *yes* to *no*.

```
PermitRootLogin no
```


From Technologies to Solutions
trixbox CE 2.6
Implementing, managing, and maintaining an Asterisk-based telephony system
Kerry Garrison
PACKT

Change the SSH port

Finally, it's recommended that the *Port* setting from the standard *22* is changed, which everyone knows as SSH, to something else. Be careful what you change it to; you don't want the port to conflict with a port in use or that might become in use. You can also attract more attention to the server if you put it on another known port than if you left it at *22*. In this example, we will use *2222*. Please decide your own port number to use on your system. The setting we edit is *Port 22* in */etc/ssh/sshd_config*.

Remove the # from in front of the setting and change *22* to *2222*.

```
Port 2222
```

We need to restart *sshd* for the changes to take effect. Please use caution when changing these settings on a remote system that you can't easily get to. If there is an error in the *config*, it could cause *sshd* to not restart. To restart the SSH service for the new settings to take effect, use the following command:

```
service sshd restart
```

Now, test to make sure that you can get into the server over SSH. The root user should be denied access and only the user we created should be allowed to get in. Don't forget to change your SSH port to 2222 when connecting. In Putty, it is listed next to the IP address; on the command line, the flag is *-p port*.

Extension security

Although, in the examples you've seen throughout this article, the extensions use the same secret as the extension number, in practice this is a very big security hole as several scripts that are available look for exactly this setup when trying to attack Asterisk-based systems. Make sure that you use a very strong password as your secret for each extension. In the next section, we will look at a set of tools that can be used to protect your system against extension attacks.

Additional security

With the advent of hacking scripts, you really cannot be too careful; if you have any remote extensions or VoIP trunks, it is now recommended that you set up tools to capture illegitimate login requests and block those IP addresses from getting into your system. One popular tool among trixbox CE users is **fail2ban**, and there is quite a bit of information in the trixbox forums about how to set it up. For the purpose of this article, we are going to look at APF and BFD as a more robust solution.

The following information is provided courtesy of Tim Yardley, the trixbox CE Build Engineer. Tim's recommendation is to use R-fx Networks, APF, and BFD for firewalling trixbox CE systems.

Links to their software can be found here.

APF: **http://rfxnetworks.com/apf.php**

BFD: **http://rfxnetworks.com/bfd.php**

APF stands for **Advanced Policy Firewall**. This is used to control iptables on the system to allow or disallow ports to be open. APF has additional features that make it stand out above the rest. **Reactive Address Blocking** (**RAB**), QoS (TOS), direct integration with BFD, and much more—see its site for full details.

BFD stands for **Brute Force Detection**. This is used to monitor any failed logins and block IP addresses from getting in. This runs as a cron daemon and works perfectly with APF.

Installing both of these applications is very simple. You can download both of them from the R-fx Networks links, uncompress them, and then run the *install.sh* script. Tim has also created an installer script that can be downloaded to your machine and run. This will install the latest and greatest APF/BFD. To get this script, you will need to use *wget* or another method to pull it off a web server. You will want to be logged into your system as root to use these commands:

- wget http://engineertim.com/install_apf_bfd.sh
- chmod 755 install_apf_bfd.sh
- ./install_apf_bfd.sh

This will start the installation process for both APF and BFD. Once the scripts complete, you will be returned to a command prompt.

APF

Configuring APF is pretty easy, and we will look at few of the *config* file options in this section. Two of the options are covered in great detail on its web site and well-commented in the *conf.apf* file.

The *config* file for APF lives in */etc/apf* and is called *conf.apf*.

We will need to edit the *conf.apf* file. If you have multiple network interfaces on your trixbox setup, you will want to set the IFACE_IN and IFACE_OUT to your external interface. This is the untrusted network interface that is connected to the Internet. If you have a second card, eth1, that is used for internal, trusted network, you can set the IFACE_TRUSTED to this interface.

To begin editing the file, use the following command:

```
nano /etc/apf/conf.apf
```

Please see the comments in the *conf.apf* if you are uncertain.

The setup script will try to properly determine which interface is used for the untrusted network and place it in the appropriate field. It is recommended to set the value of *SET_TRIM* to *0*. This value sets the total number of rules allowed inside of the *deny trust* system. It is designed to save memory and start time. With the default value of 50, the system will start to purge old rules once this number is met. With the inclusion of BFD, this number will generally climb past 50.

Setting this value to *0* will disable this feature.

```
SET_TRIM="0"
```

APF has the ability to do QoS on packets; this is defined with the TOS values in the *conf.apf* file. For SIP and IAX, you can set the following:

```
TOS_8="21,20,80,4569,5060,10000_20000"
```

This also requires a small tweak to one of the *config* files, which we will discuss later in this article, in order to tag UDP packets

If you changed the SSH port to a different number, we have to edit the *conf.apf* file to match this new port.

```
HELPER_SSH_PORT="2222"
```

Make sure to replace 2222 with the correct port number on which you decided to run SSH.

Ingress filtering is used to open inbound ports for access; both TCP and UDP have separate settings. For a trixbox setup, the following ports should be open; both TCP and UDP are listed. If you are not using TFTP, then do not have port 69 open. Do not forget to change the SSH port from 22, to the port you choose to run SSH on. Otherwise, you will be locked out; here we are using port 2222 from our last example. We have not included IAX ports in this setup. There is an easy way to ensure that only specific hosts can use IAX, which we will cover later. This is handy if you use IAX to do interoffice trunks, as I do, but don't want IAX ports open for the world to see.

```
IG_TCP_CPORTS="2222,69,80,5060,6600,10000_20000"
IG_UDP_CPORTS="69,5060,10000_20000"
```

Egress filtering is used to allow outbound filtering. I don't use egress filtering, and it will not be covered in this article. It is set to *EGF="0"*, or disabled by default. In the section of the *conf.apf* file called Imported Rules, there are settings for various feeds. Feeds are used so that many people can get information about malicious IP addresses as soon as one system reports them; this way if a script from a certain IP is attacking systems, often before the script gets a chance to get to you, your system has already blocked that IP address. Some of these feeds are very handy and I use them all. You can even set up your own custom feed that would allow you to adjust all of your servers with global deny rules. You can disable or enable this feature with the *USE_DS* setting—a *1* is enabled, a *0* is disabled.

We are now ready to start APF for the first time. If you start APF right now and something is wrong, it will disable itself in 5 minutes. This is called *DEVEL_MODE* and is the first setting in the *conf.apf* file. Leave this set to *1* until you are certain you can get in via SSH and things are working.

To save the configuration file, hit *Ctrl+O* to save and *Ctrl+X* to exit.

To see a list of command-line options, run *apf* without any flags.

```
[trixbox1.localdomain apf]# apf
apf(3402): {glob} status log not found, created
APF version 9.6 <apf@r-fx.org>
Copyright (C) 1999-2007, R-fx Networks <proj@r-fx.org>
Copyright (C) 2007, Ryan MacDonald <ryan@r-fx.org>
This program may be freely redistributed under the terms of the
GNU GPL
usage /usr/local/sbin/apf [OPTION]
-s|--start ......................... load all firewall rules
-r|--restart ....................... stop (flush) & reload
firewall rules
-f|--stop........ .................. stop (flush) all firewall
rules
-l|--list .......................... list all firewall rules
-t|--status ........................ output firewall status log
-e|--refresh ....................... refresh & resolve dns names
in trust
rules
-a HOST CMT|--allow HOST COMMENT ... add host (IP/FQDN) to
allow_hosts.rules and immediately load new rule into firewall
-d HOST CMT|--deny HOST COMMENT .... add host (IP/FQDN) to
```

```
deny_hosts.rules and immediately load new rule into firewall
-u|--remove HOST ................... remove host from
[glob]*_hosts.rules and immediately remove rule from firewall
-o|--ovars ......................... output all configuration
options
```

To start APF, we issue the following command:

```
[trixbox1.localdomain apf]# apf -s
apf(3445): {glob} activating firewall
apf(3489): {glob} determined (IFACE_IN) eth0 has address
192.168.1.31
apf(3489): {glob} determined (IFACE_OUT) eth0 has address
192.168.1.31
apf(3489): {glob} loading preroute.rules
apf(3489): {resnet} downloading http://r-fx.ca/downloads/reserved.
networks
apf(3489): {resnet} parsing reserved.networks into
/etc/apf/internals/reserved.networks
apf(3489): {glob} loading reserved.networks
apf(3489): {glob} SET_REFRESH is set to 10 minutes
apf(3489): {glob} loading bt.rules
apf(3489): {dshield} downloading http://feeds.dshield.org/top10-
2.txt
apf(3489): {dshield} parsing top10-2.txt into
/etc/apf/ds_hosts.rules
apf(3489): {dshield} loading ds_hosts.rules
apf(3489): {sdrop}
downloading http://www.spamhaus.org/drop/drop.lasso
apf(3489): {sdrop} parsing drop.lasso into
/etc/apf/sdrop_hosts.rules
apf(3489): {sdrop} loading sdrop_hosts.rules
apf(3489): {glob} loading common drop ports
...........trimmed for this document.........
apf(3489): {glob} default (ingress) input drop
apf(3445): {glob} firewall initalized
apf(3445): {glob} !!DEVELOPMENT MODE ENABLED!! - firewall will
```

```
flush
every 5 minutes.
```

We can see that APF has started, downloaded some rules from *dshield.org* and *spamhaus.org*, and then told us it is in *DEVELOPMENT MODE*. Now, test connecting to your server over SSH to ensure that you have set up the correct port number ingress. If you can't connect, you will have to wait 5 minutes and then APF will shutdown. Once you are sure you can get in with SSH, we can change the *conf.apf* file from*DEVEL_MODE="1"* to *DEVEL_MODE="0"* and restart/start APF. APF will start and not warn you about being in *DEVELOPMENT MODE*; your firewall should be good to go.

APF additional tweaks

This setup might not be ideal for everyone. If you connect to your provider over IAX, then you will definitely want to add the IAX ports to the *conf.apf*. However, if you have two or more systems that you connect to each other over IAX for interoffice connections, this is the way to go. This will work with static IP addresses and DynDNS setups alike. You can use a fully qualified DNS hostname or IP address. One of the flags for the *apf* command is *-a*, which is allow. This will globally allow a host to connect to this system, bypassing the firewall rules. It can't be stressed that how handy this is. Some examples are allowing an SNMP query, IAX connections, or other ports that you do not want open, but need to allow specific hosts to connect to. To do this, just issue the following command and substitute your remote system IP address for the one we have here.

```
apf -a 192.168.1.216
```

This will allow the system *192.168.1.216* to connect to any port on the firewalled server, thereby bypassing the firewall rules. If you are running APF on both systems, be sure to do the same thing on the other host using the correct IP address.

APF also allows a system admin to block a host or a complete subnet. This is handy if you see someone attempting to connect to your machine over FTP, Telnet, SSH, and so on. To block a specific host, use the following; be sure to use the IP address you want to block.

```
apf -d 192.168.1.216
```

To block a complete subnet (CIDR), the command is very similar:

```
apf -d 202.86.128.0/24
```

This will block the entire subnet. You can sometimes get the subnet (CIDR) listing using a WHOIS on the IP address. You can also look up a CIDR by IP on Google or ripe.net. Be sure that the subnet that you are blocking is not the one you are using or you could lock yourself out.

TOS for UDP packets are not defined for APF. Only TCP packets have the TOS bit set. There is an easy way to fix this. In the */etc/apf/internals* folder, there is a file called *functions.apf*. We need to edit this file manually. It is pretty straightforward as to what we need to change, so don't worry. There are several places where we have to add a single line. Look for the *TOS_* section in the *functions.apf* file. It will look like this:

```
if [ ! "$TOS_0" == "" ]; then
for i in `echo $TOS_0 | tr ',' ' '`; do
i=`echo $i | tr '_' ':'`
$IPT -t mangle -A PREROUTING -p tcp --sport $i -j TOS --set-tos 0
done
fi
```

We have to add the settings for UDP. We copy one line and change *tcp* to *udp*. A sample is below, highlighted.

```
if [ ! "$TOS_0" == "" ]; then
for i in `echo $TOS_0 | tr ',' ' '`; do
i=`echo $i | tr '_' ':'`
$IPT -t mangle -A PREROUTING -p tcp --sport $i -j TOS --set-tos 0
$IPT -t mangle -A PREROUTING -p udp --sport $i -j TOS --set-tos 0
done
fi
```

This additional line has to be done for all the TOS bits you are using. If you are using only TOS_8, then only worry about doing it for those. Make sure you do the *tospostroute* and *tospreroute* sections.

BFD

Brute Force Detection is used to capture illegitimate login attempts for services on the system. We see quite often a large number of SSH attempts into servers that haven't had the SSH port changed. These attempts are often an outside attempt to gain access by running dictionary attacks against common user names. These can now easily be stopped by using BFD.

If you ran the *install_apf_bfd.sh*, then BFD should be installed. The configuration file for BFD is located in*/usr/local/bfd* and is called *conf.bfd*. This file, like the one for APF, is heavily commented and covered in great detail on the R-fx Networks web site. This section will just cover some of the settings. First, this must be stated that you can become locked out of your own server if you fail to type your own password correctly.

This is another good reason to add a trusted system using the *apf -a* command. You can also add a host to specifically block by adding the IP address to the */usr/local/bfd/ignore.hosts* file.

The *ban* command that BFD uses is tied directly to APF. The command is *apf -d*, which is the same as we saw to manually ban addresses and subnets. The first configuration variable we will look at is TRIG; this is the number of failed attempts before becoming banned. The default is 15, and is pretty good. Keep in mind that this is per IP address connections, not account. So if 1 IP address fails 15 times using multiple accounts, it will be banned. Feel free to change this value if you want; I recommend not setting this above 5 to reduce the number of attempts that are allowed.

BFD has the ability to send emails out to alert of brute force attempts. This is a good idea as it will give you notice when attempts to access your system are occurring. To enable email alerts, set the value of*EMAIL_ALERTS* to *1*; then set the address you want emails to be sent to using *EMAIL_ADRESS*. You can define the subject for the email as well. This makes for easy flagging/filtering in email applications.

BFD runs from cron and places a cron entry in */etc/cron.d* called *bfd*. This runs BFD every 3 minutes. This should be acceptable for almost anyone. You can get a list of offending IP addresses using *bfd* on the command line. This is useful for looking at specific IP subnets that you might want to start blocking, if you see a pattern starting. To get this list, use the following command:

```
bfd -a
```

To start BFD, use the following command:

```
bfd -s
```

Summary

While there are other ways to help ensure the security of your system, we have covered some of the most important in this article. Besides a good firewall, changing access to the SSH service and adding login attempt protection to your extensions is going to go a long way in keeping hackers out of your system. Do not underestimate the importance of security; these steps can mean the difference between being secured and having someone log in and start making thousands of phone calls around the country from your phone system.

Director of trixbox, Kerry Garrison

About the Author :

Kerry Garrison

Kerry Garrison has been in the IT industry for over 20 years with positions ranging from IT Director of a large multi-site distribution company to developing a large hosted web server platform for a major ISP, to finally running his own IT consulting business in Southern California. Kerry was introduced to the world of Asterisk by a friend and began running his own business on it. After about a year of working with it and writing some articles that became extremely popular on the net, he felt it was time to start putting clients onto Asterisk-based systems. Today, Asterisk PBX systems represent a significant portion of his business revenue. Kerry has spoken at Astricon and does a regular seminar series in California. He is also the publisher of both **http://www.voipspeak.net** and **http://www.asterisktutorials.com**. He is very active with the Asterisk and FreePBX community and has even contributed modules to the FreePBX project.