How To Protect SSH with fail2ban on Debian 7

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: *http://faq.asphosthelpdesk.com/article.php?id=195*

Introduction

Having a server or computer connected to a network comes with a certain amount of risk. Any machine, including a server, connected to the internet is a potential target for malicious attacks.

While having a well-configured firewall will prevent many kinds of illegitimate access, you still need to open up certain services to allow yourself the ability to log in and administer the server. SSH is the service most commonly used to log into remote systems, and so it also is one of the most frequently targeted.

Fortunately, there is a tool available that can mitigate this attack vector, called fail2ban. This can be configured to allow legitimate logins using SSH, but ban IP addresses after they have failed to authenticate correctly after a set number of times.

We will be installing and configuring this software on a Debian 7 server.

Step One â€“â€“ Install fail2ban

Debian includes fail2ban in its default repositories. We can download and install it with the following set of commands:

```
sudo apt-get update
sudo apt-get install fail2ban
```

This will not only install fail2ban, it will also start the service with the default settings.

Step Two â€“â€“ Configure fail2ban

The fail2ban configuration is kept in the `/etc/fail2ban` directory. The configuration file that specifies the default banning rules is called `jail.conf`.

Because of the way that fail2ban updates its configuration files when the program has a new version, we should not edit the default configuration file.

Instead, we should copy it to a new location and edit it there:

```
cd /etc/fail2ban
sudo cp jail.conf jail.local
sudo nano jail.local
```

Here, we can change any settings that we do not like that were set in the default configuration.

Default Configuration

The section that begins `[DEFAULT]` configures defaults that may be overridden in more specific contexts

later in the configuration. It is a good idea to have strong defaults.

Most of the settings that have been given are good selections for default options. However, there are some areas that would benefit from configuration.

Ban Defaults

We can configure the way that fail2ban implements its banning by modifying a few parameters. Here are some of the more important ones:

- **ignoreip: This parameter takes a list of IP addresses that should be excluded from fail2ban rules. The IP addresses or blocks listed here will not have restrictions placed on them, so choose them wisely and specifically.**
  - IP addresses and ranges are separated by white space.
  - You should add your home or work IP address to the end of the list so that you won't be blocked if you are having trouble logging in.
  - This will look like: "ignoreip = 127.0.0.1/8 `YOUR_IP_ADDRESS`"
- **bantime: This lists the amount of time a ban will last if the client fails to authenticate correctly. It is given in seconds.**
  - The default value bans clients for 10 minutes.
- **maxretry: This parameter specifies the number of attempts are allowed before a ban is instituted.**

Define Ban Actions

When a ban is needed, fail2ban can procede in quite a few different ways. It decides on the necessary actions by looking at the following parameters:

- **banaction: This setting specifies the configuration file that will be used when a ban is needed.**
  - The value of this parameter refers to a file in the `/etc/fail2ban/action.d` directory, which will handle the actual banning process.
  - The default value uses iptables (a firewall) to ban an IP on all ports when it fails authentication. We will look at the specific banning rules later.
- **action: This parameter specifies one of the action shortcuts that are listed above it. It basically calls a `banaction` script (as mentioned above), and then assigns appropriate information to variables and passes them to the script.**
  - The default action is `action_`, which calls the script and passes the name, port, protocol, and chain to the script. It does not send an email address or log lines as some of the other actions do.

Configure Email Alerts

If you would like to configure fail2ban to email you when it institutes a ban, you can configure that in the default section as well.

If you have configured a mail server on your machine, you can configure fail2ban to send email to an external address. Otherwise, you can have it delivered to a local unix account.

There are two relevant parameters:

- **destemail: This option sets the email address that will be notified in the event of a banning.**
  - The default value, "root@localhost", will deliver mail to the root account of the current computer.
  - If you have a mail server configured, feel free to change this to an external mailing address.
- **mta: This specifies the mail agent that will be used to deliver mail.**
  - If you have a mail server configured with sendmail, leave the default option (sendmail), as is.
  - If you do not have a mail server configured, but want local mail delivered to a user account, you can change "sendmail" to "mail".

If you do wish to configure email, you will have to edit the `action` parameter as mentioned above. Change the action to either "action_mw" or "action_mwl" to have the email information passed to the banning script.

If you have configured local mail delivery, you can check the mail by typing:

```
sudo nano /var/mail/mail
```

Configure Application-Specific Jails

Further down in the file, you should see sections marked like this:

```
[application_name]
```

You should be able to decipher most of the parameters.

The `filter` parameter specifies a file within the `/etc/fail2ban/filter.d` directory. This tells fail2ban how to parse the log file of the program for failed authentication attemts.

The `logpath` variable holds the path to the service's log file, which fail2ban will parse for failures.

You can override any of the other default parameters here. For instance, the `maxretry` option is different for SSH than the dault option in a Debian install.

Step Three â€“â€“ Configure iptables

We will not actually be doing much configuration of iptables, but we will take a look at the configuration file that implements its behavior. This will help us understand how fail2ban implements its banning policies.

Open the file that was specified in our jail configuration under the `banaction` parameter:

```
sudo nano /etc/fail2ban/action.d/iptables-multiport.conf
```

Here, we can see what actually happens when fail2ban calls to have an IP banned. It uses the iptables firewall software to implement rules.

When fail2ban begins, it calls these lines:

```
actionstart = iptables -N fail2ban-<name>
              iptables -A fail2ban-<name> -j RETURN   # questionable usefulness
              iptables -I <chain> -p <protocol> -m multiport --dports <port> -j
fail2ban-<name>
```

This initializes the environment to pass traffic through a filtering chain.

The iptables software controls traffic based on "funnels", or "chains". Each of these funnels applies rules on all traffic that is given to it in order to decide if it is acceptable or not.

The first line, `iptables -N fail2ban-<name>`, creates a new chain named "fail2ban-" with the name of the service following. This will hold the rules that ban certain IP addresses.

The next line, `iptables -A fail2ban-<name> -j RETURN`, adds a rule to the chain we just created, which tells iptables to return control to the chain that called this chain.

The last line, `iptables -I <chain> -p <protocol> -m multiport --dports <port> -j fail2ban-<name>`, adds a rule to the INPUT chain (specified in our jail.local file) which immediately passes control to our new fail2ban chain.

So, the current flow is that incoming traffic is handled by our INPUT chain. At this point, it hits the rule passing control to the fail2ban chain. The first rule in this chain passes control back to the chain that called it, the INPUT chain.

So, at this point, control is just passed back and forth with nothing actually happening. However, we have set up a control flow that will accomodate additional rules. When we need to ban an IP address, we can add another rule to the fail2ban chain right above where it passes control back to the INPUT chain.

We can see the complementary actions for tearing down the fail2ban rules, for when the service stops, here:

```
actionstop = iptables -D <chain> -p <protocol> -m multiport --dports <port> -j
fail2ban-<name>
             iptables -F fail2ban-<name>
             iptables -X fail2ban-<name>
```

This basically just reverses all of the rules that we've just built up.

When you ban a client, this rule is implemented:

```
actionban = iptables -I fail2ban-<name> 1 -s <ip> -j DROP
```

This tells iptables to drop any packets from that IP address, which effectively bans them from even attempting to authenticate again.

When the bantime has elapsed, this rule reverses the ban:

```
actionunban = iptables -D fail2ban-<name> -s <ip> -j DROP
```

If you would like to see which rules are implemented and which IP addresses are currently banned, you can check the current iptables rules by typing:

```
sudo iptables -L
```

If any clients are banned, they will be in the bottom chain.

Step Four â€“â€“ Restart fail2ban

When you've made any changes to your configuration, you need to restart fail2ban to implement the new rules.

You can do this by typing this command:

```
sudo service fail2ban restart
```

To test your new rules, you can create another server instance and authenticate incorrectly on purpose enough times from that machine to trigger the ban rule. After that, your SSH call will not even return a password prompt.

If you look at the iptable rules on the host you configured, you will see a new rule:

```
sudo iptables -L

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
fail2ban-ssh  tcp  --  anywhere             anywhere             multiport
dports ssh
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Chain fail2ban-ssh (1 references)
target     prot opt source               destination
DROP       all  --  xxx-xxxxxxxx.dyn.xxxxxxxxx.net  anywhere
RETURN     all  --  anywhere             anywhere
```

You can see the new rule second from the bottom.

Conclusion

You should now have some additional security by making your server a harder target to brute force. While this is a great start, a more complete solution would be disabling password authentication completely and allowing only key-based authentication.