// Tutorial //

How To Protect SSH With Fail2Ban on CentOS 7

Published on January 28, 2016

Security Firewall Networking CentOS

By Brennen Bearnes

Introduction

While connecting to your server through SSH can be very secure, the SSH daemon itself is a service that must be exposed to the Internet to function properly. This comes with some inherent risk and offers a vector of attack for would-be assailants.

Any service that is exposed to the network is a potential target in this way. If you pay attention to application logs for these services, you will often see repeated, systematic login attempts that represent brute-force attacks by users and bots alike.

A service called **Fail2ban** can mitigate this problem by creating rules that automatically alter your iptables firewall configuration based on a predefined number of unsuccessful login attempts. This will allow your server to respond to illegitimate access attempts without intervention from you.

In this guide, we'll cover how to install and use Fail2ban on a CentOS 7 server.

Install Fail2ban on CentOS 7

While Fail2ban is not available in the official CentOS package repository, it is packaged for the EPEL project. EPEL, standing for Extra Packages for Enterprise Linux, can be installed with a release package that *is* available from CentOS:

```
$ sudo yum install epel-release
```

Copy

You will be prompted to continue—press **y**, followed by **Enter**:

```
yum prompt
Transaction Summary
================================================================================
Install  1 Package
```

```
Total download size: 14 k
Installed size: 24 k
Is this ok [y/d/N]: y
```

Now we should be able to install the `fail2ban` package:

```
$ sudo yum install fail2ban
```
Copy

Again, press **y** and **Enter** when prompted to continue.

Once the installation has finished, use `systemctl` to enable the `fail2ban` service:

```
$ sudo systemctl enable fail2ban
```
Copy

Configure Local Settings

The Fail2ban service keeps its configuration files in the `/etc/fail2ban` directory. There, you can find a file with default values called `jail.conf`. Since this file may be overwritten by package upgrades, we shouldn't edit it in-place. Instead, we'll write a new file called `jail.local`. Any values defined in `jail.local` will override those in `jail.conf`.

`jail.conf` contains a `[DEFAULT]` section, followed by sections for individual services. `jail.local` may override any of these values. Additionally, files in `/etc/fail2ban/jail.d/` can be used to override settings in both of these files. Files are applied in the following order:

1. `/etc/fail2ban/jail.conf`
2. `/etc/fail2ban/jail.d/*.conf`, alphabetically
3. `/etc/fail2ban/jail.local`
4. `/etc/fail2ban/jail.d/*.local`, alphabetically

Any file may contain a `[DEFAULT]` section, executed first, and may also contain sections for individual jails. The last vavalue set for a given parameter takes precedence.

Let's begin by writing a very simple version of `jail.local`. Open a new file using `nano` (or your editor of choice):

```
$ sudo nano /etc/fail2ban/jail.local
```
Copy

Paste the following:

/etc/fail2ban/jail.local

```
[DEFAULT]
# Ban hosts for one hour:
bantime = 3600

# Override /etc/fail2ban/jail.d/00-firewalld.conf:
banaction = iptables-multiport

[sshd]
enabled = true
```

This overrides three settings: It sets a new default `bantime` for all services, makes sure we're using `iptables` for firewall configuration, and enables the `sshd` jail.

Exit and save the new file (in `nano`, press **Ctrl-X** to exit, **y** to save, and **Enter** to confirm the filename). Now we can restart the `fail2ban` service using `systemctl`:

```
$ sudo systemctl restart fail2ban
```
Copy

The `systemctl` command should finish without any output. In order to check that the service is running, we can use `fail2ban-client`:

```
$ sudo fail2ban-client status
```
Copy

```
Output
Status
|- Number of jail:      1
`- Jail list:   sshd
```

You can also get more detailed information about a specific jail:

```
$ sudo fail2ban-client status sshd
```
Copy

Explore Available Settings

The version of `jail.local` we defined above is a good start, but you may want to adjust a number of other settings. Open `jail.conf`, and we'll examine some of the defaults. If you decide to change any of these values, remember that they should be copied to the appropriate section of `jail.local` and adjusted there, rather than modified in-place.

```
$ sudo nano /etc/fail2ban/jail.conf
```
Copy

Default Settings for All Jails

First, scroll through the `[DEFAULT]` section.

```
ignoreip = 127.0.0.1/8
```

You can adjust the source addresses that Fail2ban ignores by adding a value to the `ignoreip` parameter. Currently, it is configured not to ban any traffic coming from the local machine. You can include additional addresses to ignore by appending them to the end of the parameter, separated by a space.

```
bantime = 600
```

The `bantime` parameter sets the length of time that a client will be banned when they have failed to authenticate correctly. This is measured in seconds. By default, this is set to 600 seconds, or 10 minutes.

```
findtime = 600
maxretry = 3
```

The next two parameters that you want to pay attention to are `findtime` and `maxretry`. These work together to establish the conditions under which a client should be banned.

The `maxretry` variable sets the number of tries a client has to authenticate within a window of time defined by `findtime`, before being banned. With the default settings, Fail2ban will ban a client that unsuccessfully attempts to log in 3 times within a 10 minute window.

```
destemail = root@localhost
sendername = Fail2Ban
mta = sendmail
```

If you wish to configure email alerts, you may need to override the `destemail`, `sendername`, and `mta` settings. The `destemail` parameter sets the email address that should receive ban messages. The `sendername` sets the value of the “From” field in the email. The `mta` parameter configures what mail service will be used to send mail.

```
action = $(action_)s
```

This parameter configures the action that Fail2ban takes when it wants to institute a ban. The value `action_` is defined in the file shortly before this parameter. The default action is to simply configure the firewall to reject traffic from the offending host until the ban time elapses.

If you would like to configure email alerts, you can override this value from `action_` to `action_mw`. If you want the email to include the relevant log lines, you can change it to `action_mwl`. You'll want to make sure you have the appropriate mail settings configured if you choose to use mail alerts.

Settings for Individual Jails

After `[DEFAULT]`, we'll encounter sections configuring individual jails for different services. These will typically include a `port` to be banned and a `logpath` to monitor for malicious access attempts. For example, the SSH jail we already enabled in `jail.local` has the following settings:

/etc/fail2ban/jail.local

```
[sshd]

port    = ssh
logpath = %(sshd_log)s
```

In this case, `ssh` is a pre-defined variable for the standard SSH port, and `%(sshd_log)s` uses a value defined elsewhere in Fail2ban's standard configuration (this helps keep `jail.conf` portable between different operating systems).

Another setting you may encounter is the `filter` that will be used to decide whether a line in a log indicates a failed authentication.

The `filter` value is actually a reference to a file located in the `/etc/fail2ban/filter.d` directory, with its `.conf` extension removed. This file contains the regular expressions that determine whether a line in the log is bad. We won't be covering this file in-depth in this guide, because it is fairly complex and the predefined settings match appropriate lines well.

However, you can see what kind of filters are available by looking into that directory:

```
$ ls /etc/fail2ban/filter.d
```

Copy

If you see a file that looks to be related to a service you are using, you should open it with a text editor. Most of the files are fairly well commented and you should be able to tell what type of condition the script was designed to guard against. Most of these filters have appropriate (disabled) sections in `jail.conf` that we can enable in `jail.local` if desired.

For instance, pretend that we are serving a website using Nginx and realize that a password-protected portion of our site is getting slammed with login attempts. We can

tell Fail2ban to use the `nginx-http-auth.conf` file to check for this condition within the `/var/log/nginx/error.log` file.

This is actually already set up in a section called `[nginx-http-auth]` in our `/etc/fail2ban/jail.conf` file. We would just need to add an `enabled` parameter for the `nginx-http-auth` jail to `jail.local`:

/etc/fail2ban/jail.local

```
[DEFAULT]
# Ban hosts for one hour:
bantime = 3600

# Override /etc/fail2ban/jail.d/00-firewalld.conf:
banaction = iptables-multiport

[sshd]
enabled = true

[nginx-http-auth]
enabled = true
```

And restart the `fail2ban` service:

```
$ sudo systemctl restart fail2ban
```

Copy

Monitor Fail2ban Logs and Firewall Configuration

It's important to know that a service like Fail2ban is working as-intended. Start by using `systemctl` to check the status of the service:

```
$ sudo systemctl status fail2ban
```

Copy

If something seems amiss here, you can troubleshoot by checking logs for the `fail2ban` unit since the last boot:

```
$ sudo journalctl -b -u fail2ban
```

Copy

Next, use `fail2ban-client` to query the overall status of `fail2ban-server`, or any individual jail:

```
$ sudo fail2ban-client status
$ sudo fail2ban-client status jail_name
```

Copy

Follow Fail2ban's log for a record of recent actions (press **Ctrl-C** to exit):

```
$ sudo tail -F /var/log/fail2ban.log
```
Copy

List the current rules configured for iptables:

```
$ sudo iptables -L
```
Copy

Show iptables rules in a format that reflects the commands necessary to enable each rule:

```
$ sudo iptables -S
```
Copy

Conclusion

You should now be able to configure some basic banning policies for your services. Fail2ban is very easy to set up, and is a great way to protect any kind of service that uses authentication.

If you want to learn more about how Fail2ban works, you can check out our tutorial on how fail2ban rules and files work.