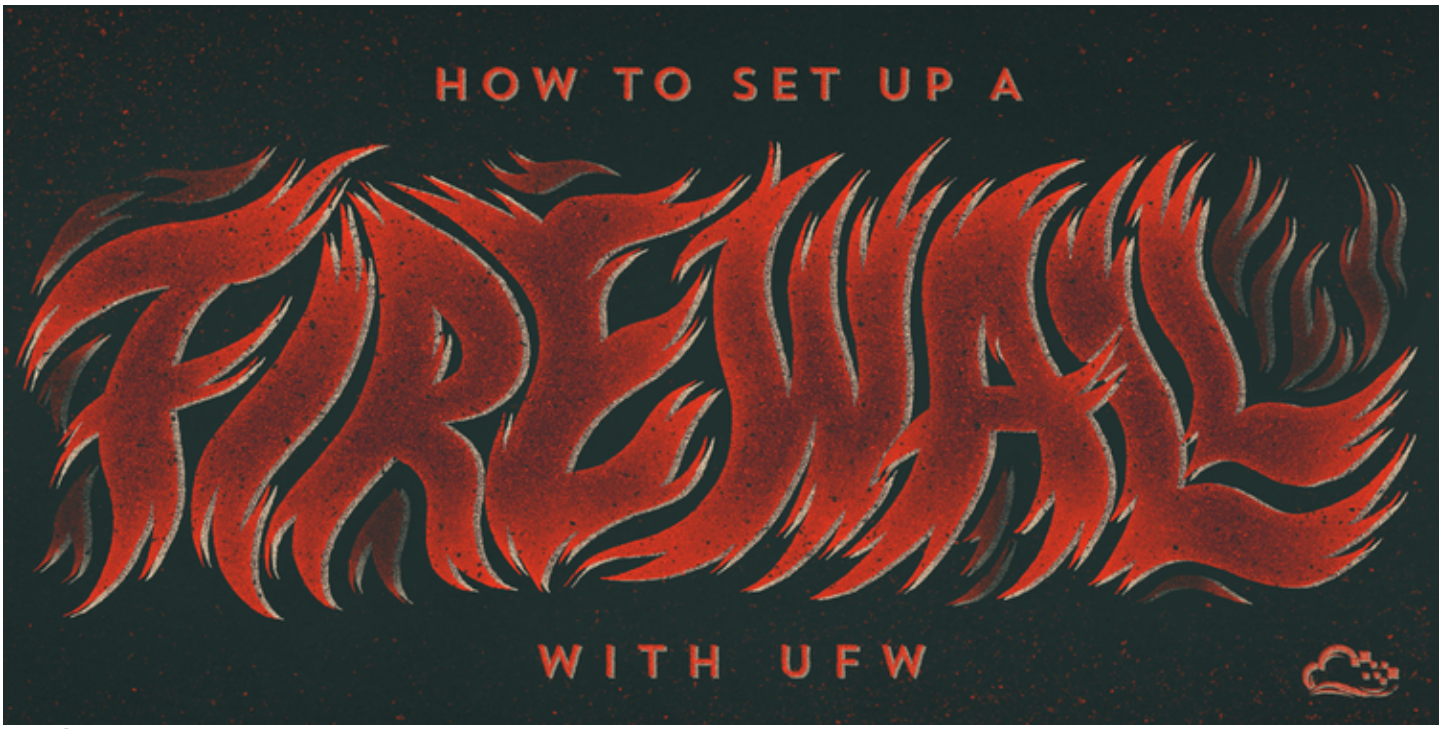
DigitalOcean

Subscribe



How To Set Up a Firewall with UFW on Ubuntu 14.04

87

Posted August 5, 2015 623.7k FIREWALL NETWORKING SECURITY UBUNTU

By: Mitchell Anicas

Not using **Ubuntu 14.04**? Choose a different version:

Introduction

UFW, or Uncomplicated Firewall, is an interface to `iptables` that is geared towards simplifying the process of configuring a firewall. While `iptables` is a solid and flexible tool, it can be difficult for beginners to learn how to use it to properly configure a firewall. If you're looking to get started securing your network, and you're not sure which tool to use, UFW may be the right choice for you.

SCROLL TO TOP

This tutorial will show you how to set up a firewall with UFW on Ubuntu 14.04.

Prerequisites

Before you start using this tutorial, you should have a separate, non-root superuser account—a user with sudo privileges—set up on your Ubuntu server. You can learn how to do this by completing at least steps 1-3 in the Initial Server Setup with Ubuntu 14.04 tutorial.

UFW is installed by default on Ubuntu. If it has been uninstalled for some reason, you can install it with `apt-get`:

```
$ sudo apt-get install ufw
```

Using IPv6 with UFW

If your Ubuntu server has IPv6 enabled, ensure that UFW is configured to support IPv6 so that it will manage firewall rules for IPv6 in addition to IPv4. To do this, open the UFW configuration with your favorite editor. We'll use nano:

```
$ sudo nano /etc/default/ufw
```

Then make sure the value of "IPV6" is to equal "yes". It should look like this:

/etc/default/ufw excerpt

```
...
IPV6=yes
...
```

Save and quit. Hit `Ctrl-X` to exit the file, then `Y` to save the changes that you made, then `ENTER` to confirm the file name.

When UFW is enabled, it will be configured to write both IPv4 and IPv6 firewall rules.

This tutorial is written with IPv4 in mind, but will work fine for IPv6 as long as you enable it.

Check UFW Status and Rules

At any time, you can check the status of UFW with this command:

```
$ sudo ufw status verbose
```

By default, UFW is disabled so you should see something like this:

```
Output:
Status: inactive
```

If UFW is active, the output will say that it's active, and it will list any rules that are set. For example, if the firewall is set to allow SSH (port 22) connections from anywhere, the output might look something like this:

```
Output:
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
```

As such, use the status command if you ever need to check how UFW has configured the firewall.

Before enabling UFW, we will want to ensure that your firewall is configured to allow you to connect via SSH. Let's start with setting the default policies.

Set Up Default Policies

If you're just getting started with your firewall, the first rules to define are your default policies. These rules control how to handle traffic that does not explicitly match any other rules. By default, UFW is set to deny all incoming connections and allow all outgoing connections. This means anyone trying to reach your cloud server would not be able to connect, while any application within the server would be able to reach the outside world.

Let's set your UFW rules back to the defaults so we can be sure that you'll be able to follow along with this tutorial. To set the defaults used by UFW, use these commands:

```
$ sudo ufw default deny incoming
$ sudo ufw default allow outgoing
```

As you might have guessed, these commands set the defaults to deny incoming and allow outgoing connections. These firewall defaults, by themselves, might suffice for a personal

computer but servers typically need to respond to incoming requests from outside users. We'll look into that next.

Allow SSH Connections

If we enabled our UFW firewall now, it would deny all incoming connections. This means that we will need to create rules that explicitly allow legitimate incoming connections—SSH or HTTP connections, for example—if we want our server to respond to those types of requests. If you're using a cloud server, you will probably want to allow incoming SSH connections so you can connect to and manage your server.

To configure your server to allow incoming SSH connections, you can use this UFW command:

```
$ sudo ufw allow ssh
```

This will create firewall rules that will allow all connections on port 22, which is the port that the SSH daemon listens on. UFW knows what "ssh", and a bunch of other service names, means because it's listed as a service that uses port 22 in the `/etc/services` file.

We can actually write the equivalent rule by specifying the **port** instead of the service name. For example, this command works the same as the one above:

```
$ sudo ufw allow 22
```

If you configured your SSH daemon to use a different port, you will have to specify the appropriate port. For example, if your SSH server is listening on port 2222, you can use this command to allow connections on that port:

```
$ sudo ufw allow 2222
```

Now that your firewall is configured to allow incoming SSH connections, we can enable it.

Enable UFW

To enable UFW, use this command:

```
$ sudo ufw enable
```

You will receive a warning that says the "command may disrupt existing ssh connections." We already set up a firewall rule that allows SSH connections so it should be fine to continue.

Respond to the prompt with `y`.

The firewall is now active. Feel free to run the `sudo ufw status verbose` command to see the rules that are set.

Allow Other Connections

Now you should allow all of the other connections that your server needs to respond to. The connections that you should allow depends your specific needs. Luckily, you already know how to write rules that allow connections based on a service name or port—we already did this for SSH on port 22.

We will show a few examples of very common services that you may need to allow. If you have any other services for which you want to allow all incoming connections, follow this format.

HTTP—port 80

HTTP connections, which is what unencrypted web servers use, can be allowed with this command:

```
$ sudo ufw allow http
```

If you'd rather use the port number, 80, use this command:

```
$ sudo ufw allow 80
```

HTTPS—port 443

HTTPS connections, which is what encrypted web servers use, can be allowed with this command:

```
$ sudo ufw allow https
```

If you'd rather use the port number, 443, use this command:

```
$ sudo ufw allow 443
```

FTP—port 21

FTP connections, which is used for unencrypted file transfers (which you probably shouldn't use anyway), can be allowed with this command:

```
$ sudo ufw allow ftp
```

If you'd rather use the port number, 21, use this command:

```
$ sudo ufw allow 21/tcp
```

Allow Specific Port Ranges

You can specify port ranges with UFW. Some applications use multiple ports, instead of a single port.

For example, to allow X11 connections, which use ports 6000-6007, use these commands:

```
$ sudo ufw allow 6000:6007/tcp
$ sudo ufw allow 6000:6007/udp
```

When specifying port ranges with UFW, you must specify the protocol (`tcp` or `udp`) that the rules should apply to. We haven't mentioned this before because not specifying the protocol simply allows both protocols, which is OK in most cases.

Allow Specific IP Addresses

When working with UFW, you can also specify IP addresses. For example, if you want to allow connections from a specific IP address, such as a work or home IP address of `15.15.15.51`, you need to specify "from" then the IP address:

```
$ sudo ufw allow from 15.15.15.51
```

You can also specify a specific port that the IP address is allowed to connect to by adding "to any port" followed by the port number. For example, If you want to allow `15.15.15.51` to connect to port 22 (SSH), use this command:

```
sudo ufw allow from 15.15.15.51 to any port 22
```

Allow Subnets

If you want to allow a subnet of IP addresses, you can do so using CIDR notation to specify a netmask. For example, if you want to allow all of the IP addresses ranging from `15.15.15.1` to `15.15.15.254` you could use this command:

```
$ sudo ufw allow from 15.15.15.0/24
```

Likewise, you may also specify the destination port that the subnet `15.15.15.0/24` is allowed to connect to. Again, we'll use port 22 (SSH) as an example:

```
sudo ufw allow from 15.15.15.0/24 to any port 22
```

Allow Connections to a Specific Network Interface

If you want to create a firewall rule that only applies to a specific network interface, you can do so by specifying "allow in on" followed by the name of the network interface.

You may want to look up your network interfaces before continuing. To do so, use this command:

```
$ ip addr
```

```
Output Excerpt:
...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
...
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
...
```

The highlighted output indicates the network interface names. They are typically named something like "eth0" or "eth1".

So, if your server has a public network interface called `eth0`, you could allow HTTP traffic (port 80) to it with this command:

```
$ sudo ufw allow in on eth0 to any port 80
```

Doing so would allow your server to receive HTTP requests from the public Internet.

Or, if you want your MySQL database server (port 3306) to listen for connections on the private network interface `eth1`, for example, you could use this command:

```
$ sudo ufw allow in on eth1 to any port 3306
```

This would allow other servers on your private network to connect to your MySQL database.

Deny Connections

If you haven't changed the default policy for incoming connections, UFW is configured to deny all incoming connections. Generally, this simplifies the process of creating a secure firewall policy by requiring you to create rules that explicitly allow specific ports and IP addresses through. However, sometimes you will want to deny specific connections based on the source IP address or subnet, perhaps because you know that your server is being attacked from there. Also, if you want change your default incoming policy to **allow** (which isn't recommended in the interest of security), you would need to create **deny** rules for any services or IP addresses that you don't want to allow connections for.

To write **deny** rules, you can use the commands that we described above except you need to replace "allow" with "deny".

For example to deny HTTP connections, you could use this command:

```
$ sudo ufw deny http
```

Or if you want to deny all connections from `15.15.15.51` you could use this command:

```
sudo ufw deny from 15.15.15.51
```

If you need help writing any other **deny** rules, just look at the previous **allow** rules and update them accordingly.

Now let's take a look at how to delete rules.

Delete Rules

Knowing how to delete firewall rules is just as important as knowing how to create them. There are two different ways specify which rules to delete: by rule number or by the actual rule (similar to how the rules were specified when they were created). We'll start with the **delete by rule number** method because it is easier, compared to writing the actual rules to delete, if you're new to UFW.

By Rule Number

If you're using the rule number to delete firewall rules, the first thing you'll want to do is get a list of your firewall rules. The UFW status command has an option to display numbers next to each rule, as demonstrated here:

```
$ sudo ufw status numbered
```

```
Numbered Output:
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    15.15.15.0/24
[ 2] 80                         ALLOW IN    Anywhere
```

If we decide that we want to delete rule 2, the one that allows port 80 (HTTP) connections, we can specify it in a UFW delete command like this:

```
$ sudo ufw delete 2
```

This would show a confirmation prompt then delete rule 2, which allows HTTP connections. Note that if you have IPv6 enabled, you would want to delete the corresponding IPv6 rule as well.

By Actual Rule

The alternative to rule numbers is to specify the actual rule to delete. For example, if you want to remove the "allow http" rule, you could write it like this:

```
$ sudo ufw delete allow http
```

You could also specify the rule by "allow 80", instead of by service name:

```
$ sudo ufw delete allow 80
```

This method will delete both IPv4 and IPv6 rules, if they exist.

How To Disable UFW (optional)

If you decide you don't want to use UFW for whatever reason, you can disable it with this command:

```
$ sudo ufw disable
```

Any rules that you created with UFW will no longer be active. You can always run `sudo ufw enable` if you need to activate it later.

Reset UFW Rules (optional)

If you already have UFW rules configured but you decide that you want to start over, you can use the reset command:

```
$ sudo ufw reset
```

This will disable UFW and delete any rules that were previously defined. Keep in mind that the default policies won't change to their original settings, if you modified them at any point. This should give you a fresh start with UFW.

Conclusion

Your firewall should now be configured to allow (at least) SSH connections. Be sure to allow any other incoming connections that your server, while limiting any unnecessary connections, so your server will be functional and secure.

To learn about more common UFW configurations, check out this tutorial: UFW Essentials: Common Firewall Rules and Commands

Good luck!

By: Mitchell Anicas

♡ Upvote (87) Subscribe

Introducing Projects on DigitalOcean

Organize your resources according to how you work.

READ MORE

Related Tutorials

How To Migrate Iptables Firewall Rules to a New Server

How To Set Up a Firewall with UFW on Debian 9

How To Set Up a Firewall with UFW on Ubuntu 18.04

How To Secure Nginx with Let's Encrypt on FreeBSD

How to Block Unwanted SSH Login Attempts with PyFilter on Ubuntu 16.04

12 Comments

Leave a comment...

Log In to Comment

jatsrt *August 6, 2015*

2 Something I find myself doing often is allowing certain traffic on private interfaces but not on public. Might be worth an example:

```
ufw allow in on eth1 to any port 22 proto tcp
```

If you had a droplet with private networking this would allow ssh only to the private interface.

```
ufw allow in on eth0 to any port 80 proto tcp
```

This would allow HTTP to the external interface

---

manicas MOD *August 6, 2015*

0 True. I'll add that in soon. Thanks!

---

diveyez *December 25, 2017*

0 ufw allow tcp/22
ufw allow 80/tcp
ufw allow 443/tcp
Is all you need. Checkout this script.
https://github.com/diveyez/ufw-dnsbl-rules-set/blob/master/dofw.sh

---

Linza *August 10, 2015*

0 Does it matter if it was recently updated or not? The latest ufw release was version 0.33 on 2012-08-17. Would it be safe to use ufw?

---

manicas MOD *August 10, 2015*

0 I believe the latest version of UFW is 0.34. It should be fine to use older versions for the most part, as long as it supports the iptables features that you need.

---

shad *October 9, 2015*

0 Great tutorial. Thanks a lot

---

yaroslava95e681 *October 31, 2015*

0 Will changes be persisted?

do3649 *February 11, 2016*

1 I will add in that if you choose to do a deny all, and then allow exceptions, make sure you put the deny rule after (or below, or higher numbered rule) the rules that allow. For instance, below would still not allow connections from 15.15.15.0/24, you will have to reverse the order, and don't forget to reload once you change the rules, else it won't take effect yet until you do: ufw reload

this would deny everyone, even those on the subnet (due to the order)
[1] 80 DENY IN Anywhere
[2] 80 ALLOW IN 15.15.15.0/24

this would work better, denying everyone, but an IP on that subnet:
[1] 80 ALLOW IN 15.15.15.0/24
[2]80 DENY IN Anywhere

gr44bli *May 17, 2016*

0 How could i get ip's from a list (from a CMS) and only allow these ip's and also to always update the ip's from the list, is this possible?

Thanks

jimmyaneja *August 5, 2016*

0 I am having trouble accessing my machine via local IP. I set up an openvpn server along with ufw. I am able to access the openvpn server machine via private IP just fine but not able to access any other machine via private IP in the same network. Is there a rule I can add that will solve my problem or is this unrelated?? Appreciate the help and great article btw

MGV *July 11, 2017*

0 Thanks! this was great. i followed all the steps and the https works perfectly,
when I run the test though, I only get a B.
This server supports weak Diffie-Hellman (DH) key exchange parameters. Grade capped to B.

Any thoughts on how to improve it?

diveyez *December 25, 2017*

0 This script will help. https://github.com/diveyez/ufw-dnsbl-rules-set/blob/master/dofw.sh

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Copyright © 2018 DigitalOcean™ Inc.