Uncomplicated Firewall for ampr.org VPN’s

Mark Phillips, NI2O 03/2025 V1.0

Version control

Mark Phillips, NI2O	Initial Release	04/2025

Contents

Abstract

Uncomplicated Firewall (**UFW**) is a program for managing a netfilter firewall designed to be easy to use. It uses a command-line interface consisting of a small number of simple commands, and uses iptables for configuration. UFW has been available by default in all Ubuntu installations since 8.04 LTS. UFW has been available by default in all Debian installations since 10.

In my experience the UFW is anything but uncomplicated. This document aims to somewhat demystify the deployment of the UFW and aid in securing your ampr.org networks.

This document speaks explicitly to ampr.org Wireguard VPN connections hosted on Debian/Ubuntu Linux however it may have parallels and similarities to other distro's. It is also NOT a guide to the operation of UFW itself.

Install UFW

As stated above, UFW comes installed by default in Ubuntu and Debian installations. To test your system for its presence find a command line or terminal session and type "`ufw`". If the application is installed it should return something like "Status: inactive". If you get "`command not found`" then you will need to install it as below.

```
sudo apt install ufw
```

Best practice

Best practice dictates that we should have a firewall device installed ahead of any public network interface. This is often called the "boundary firewall". This should be the first thing that

incoming network connections see on their way to your applications. There is also some strong support for firewalls to be enabled on each device that will accept alien network connections. It is without the scope of this document to discuss the pro's and con's of your network requirements. Our assumption is that this UFW configuration will be installed on a boundary firewall device of some sort.

Planning

Now that we have UFW installed on our system we need to establish the access requirements of the system. There is no "one size fits all" solution to the firewall needs of any system. UFW tackles this issue with 2 basic rules; DENY everything coming in and ALLOW everything going out. If all you require is a simple firewall such as this then our work is done. Otherwise, read on.

Create a list

Get a notepad going and create a list of applications ports, protocols and addresses. An example list is created below with some common access requirements.

telnet/23/tcp/everywhere
ftp/21/tcp/44net only
smtp/25/tcp/everywhere
http/80/tcp/my house
sip/5060/udp/hamshack hotline
rtp/10000-12000/udp/hamshack hotline
snmp/161/udp/my house
ssh/22/tcp/my house

In our example we will create a set of UFW rules that will allow incoming access to applications from some networks and not others. Because of the recent ARDC IP address sell off, we will have to create 2 rules for every one application where 44net (44.x.x.x) only access is required. This may seem like duplication but it is necessary as the entire 44net is split into 2 subnets; 44.0.0.0/9 and 44.128.0.0/10. We also have to allow ourselves access to our own system. While console access from the attached monitor/keyboard is always available, network access is also desirable.

Below is the actual UFW rule set used here at ni2o.ampr.org. I run a /28 network here (16 addresses). Bear in mind that your firewall device should be equipped with 2 network interfaces; at least one of them will be a real Ethernet type connected to your home network and the other could be a virtual interface of some sort such as a TUN interface created by the AMPR-RIP application. It does not matter as long as we know what traffic will travel over which interfaces.

```
root@AMPRVPN:/etc/ufw# ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] Anywhere                   ALLOW IN    44.56.66.0/28
[ 2] Anywhere                   ALLOW OUT   44.56.66.0/28
[ 3] Anywhere                   ALLOW FWD   44.56.66.0/28
[ 4] 44.56.66.0/28              ALLOW IN    192.168.8.0/24
[ 5] 44.56.66.0/28              ALLOW IN    192.168.161.0/28
[ 6] 44.56.66.0/28              ALLOW IN    172.16.10.0/28
[ 7] 44.56.66.0/28              ALLOW FWD   192.168.8.0/24
[ 8] 44.56.66.2 80/tcp          ALLOW FWD   Anywhere
[ 9] 44.56.66.5 23/tcp          ALLOW FWD   Anywhere
[10] 44.56.66.5 2300/tcp        ALLOW FWD   Anywhere
[11] 44.56.66.5 119/tcp         ALLOW FWD   Anywhere
[12] 44.56.66.5 80/tcp          ALLOW FWD   Anywhere
[13] 44.56.66.5 21/tcp          ALLOW FWD   Anywhere
[14] 44.56.66.6 80/tcp          ALLOW FWD   Anywhere
[15] 44.56.66.6 666/tcp         ALLOW FWD   Anywhere
[16] 44.56.66.6 4000/tcp        ALLOW FWD   Anywhere
[17] 44.56.66.6 8082/tcp        ALLOW FWD   Anywhere
```

The rules above are configured in a manner so as to allow my 44 subnet to come and go without hindrance, allow my local LAN networks to go only to my personal 44net subnet and then allow global access to certain applications running on my various ampr.org systems.

You will notice 2 statements in the list above; “ALLOW IN” and “ALLOW FWD”. As I have more than one system equipped with a 44 address my firewall device has to forward connections to those devices.

Firewall rules should go from the most explicit to the most generic. They should also go in numerical IP address order (the router doesn’t care about this but it’s human readable). Exact rules should come first. If you wish to allow only your desktop machines to connect via SSH you should define your desktop’s unique IP address in that rule.

FWD vs IN

There are 2 main rules that we will need to understand. FWD (forward) and IN (accept inbound).

FWD tells UFW to accept connections that will pass through it on their way to another device. Bear in mind that we are not doing NAT (a whole other nightmare) and so for this rule our

firewall device acts as a router. It will send incoming data to the allowed recipient only on the specified port using the specified protocol (TCP/UDP/ICMP/IP).

IN tells UFW to accept connections to itself on the specified port and protocol.

You can of course, add other rules such as DENY (useful if you are getting bothersome interference from an external source)

Create the rulebase

At this point you should have a list of all the ports and IP addresses you wish to expose to the greater Internet. Let's add a rule to allow access to my RTLSDR. Because of rule 3 I am able to see the interface from my local LAN but it is not available externally. Let's fix this by adding a rule as follows;

```
sudo ufw route allow from any to 44.56.66.2/32 port 1234 proto tcp
```

This rule instructs UFW to accept connections that will be forwarded to 44.56.66.2 on port 1234 using the TCP protocol. Note that the IP address has been defined including its subnet. This rule could be slightly modified to allow forwarding of traffic to all of the subnet by amending the /32 to (in my case) /28. While the assumption of UFW is that IP addresses are specifically /32 unless otherwise stated you should ensure to add the /32 so as to remove any possible confusion in the human readable form.

It is worth noting that "route" makes the UFW forward connections and "allow" accepts connections only on the device UFW is running on. So had we written this rule:

```
sudo ufw allow from any to 44.56.66.2/32 port 1234 proto tcp
```

The UFW firewall would have accepted that rule but would NOT forward connection on to 44.56.66.2. The connections would simply "blackhole" right there at the UFW.

Edit the config files

While it is true that UFW will write all your manually entered configs to its own files it is useful to know where the relevant files are. One of the reasons we might want access to the main rules file is to change the order in which things appear. Using the UFW INSERT and DELETE commands can become a bit cumbersome when trying to order the chaos.

```
sudo nano /etc/ufw/user.rules
```

This command will open the user.rules file in the nano text editor. If you need to move things around this is where you do it.

Other files pertinent to the operation of UFW can be found in /etc/ufw. We will not discuss them here as they are without the scope of this document. If you choose to edit them, ensure to make a backup of your original file beforehand.

Start the UFW

Until now we have done nothing but edit the rulebase. Our firewall is not actually running yet. Starting it up is as simple as:

```
sudo ufw enable
```

This command will both start the firewall and also ensure it automatically starts when the host system reboots. You can check that the firewall rules are indeed loaded by doing:

```
sudo ufw status numbered
```

Whereupon UFW will print a numbered list of the current firewall rules.

Reload the rules

If you have edited the /etc/ufw/user.rules file while the UFW is running you will need to reload the rulebase to action your changes:

```
sudo ufw reload
sudo ufw status numbered
```

These commands will reload the UFW rulebase and then display the rules in a numerical list.

Delete rules on the fly

Did you note that we always asked UFW to display its status in a numbered fashion? This is especially important when we want to delete, insert or move a rule.

```
sudo delete 7
```

This command deletes rule 7 from the list and then re-orders the rules. If you continue to issue the same command you will continue to delete rule 7 which was previously rule 8 and so on. USE WITH CARE.

Inserting a rule into a specific slot in the rulebase is as simply as;

```
sudo ufw insert 7 route allow from any to 44.56.66.2/32 port http
```

This will insert the rule into the number 7 slot in the rulebase.

Careful use of the UFW will ensure a significant increase in your system security. I would recommend that you read some of the many online tutorials. This primer covers only what you will need to get going.