code academy
CS301

Lab setup for Cybersecurity students (Defensive Security)

Author: Yagub Hajiyev
Contact: yagub.hajiyev.995@gmail.com

students

Table of Contents

Summary

This lab is designed for understanding the basic network security concept in **Cybersecurity**. The main component of the network security is firewall, and we will learn how to install, configure and manage a firewall. Before we start this lab, you need to have a fundamental knowledge of firewalls. I passed the most of theoretical information and focused on practice to keep the documentation short. You also need to deploy some machines on your own, and should have fundamental **Linux** and **Active Directory** skills for performing it. If you face any trouble while setup the lab, please contact with me.

This lab covers some key security methods that demonstrated according to **TCP/IP** model in the table,

Security methods	TCP/IP layers
WAF, web filtering, anti-virus security, checklist, Reverse Proxy, GeoIP blocking, Captive portal authentication, IDS/IPS	Application
Port filtering	Transport
IP address filtering	Internet
-	Datalink
-	Physical

As you see, most of the security preventions will be performed in **Application** layer. In following captures, we will examine and test all these methods step by step.

At the end of the lab, you will be able to perform,

- Firewall setup and general configuration
- Firewall policy management
- Installing **OPNsense** firewall plugins and configuration
- **IDS/IPS** setup and configuration in firewall
- Monitoring activities

Pre-requirements

In order to setup this lab you have to prepare several machines in virtual environment, which are listed below,

	DC	WEB	Jump	Client	Guest
OS	Windows server	Oracle Linux	Windows server	Windows	-
IP address	Static	Static	Static	Dynamic	Dynamic
Service	DC and DNS	Web server (DVWA)	-	-	-
Network adapter	VMNET1	VMNET1	VMNET3	VMNET2	VMNET4

- **DC**: Create a domain to use in your lab environment (which I used **cs301.local** domain in this lab). Use this machine as your **Domain Controller** and **DNS** server, its **IP** address will be also **DNS** address for all machines in our lab (except **Guest** machine because we will isolate it from our networks). I highly recommend to use the same parameters as I used in this lab for better troubleshooting.

  IP: 172.16.10.10
  Subnet: 255.255.255.0

Gateway: 172.16.10.2
DNS: 172.16.10.10 (or 127.0.0.1)
Domain: cs301.local
Hostname: DC

- **WEB**: This machine will serve as a web application server in our environment. You need to deploy the web application which named **DVWA** (Damn Vulnerable Web Application) on it. This application is vulnerable to cyberattacks and our task is to protect it from threats.
  IP: 172.16.10.11
  Subnet: 255.255.255.0
  Gateway: 172.16.10.2
  DNS: 172.16.10.10
  Domain: cs301.local
  Hostname: WEB
  DVWA setup: https://www.howtoforge.com/how-to-install-damn-vulnerable-web-application-on-centos-8/
  Join Oracle machine to the AD domain: https://blogs.oracle.com/cloud-infrastructure/post/joining-an-oracle-linux-client-to-an-active-directory-domain

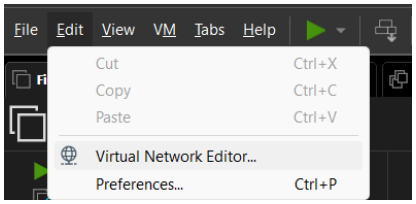
- **Jump**: We will talk about jump server logic in following captures, now just deploy a **Windows** server machine and give the parameters listed below,
  IP: 172.16.30.10
  Subnet: 255.255.255.0
  Gateway: 172.16.30.2
  DNS: 172.16.10.10
  Domain: cs301.local
  Hostname: Jump

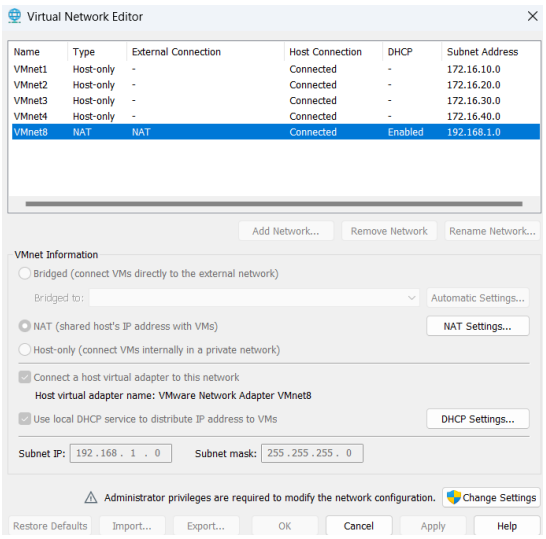
- **Client**: For simplicity, we will use only one client machine but you can use more. Join this machine to the domain and name it. We will deploy the **DHCP** service for **CLIENTS** interface in our firewall and all machines will get an **IP** from this service.

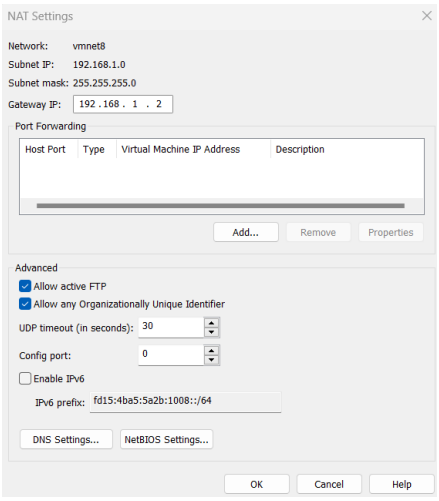
- **Guest**: It is depend on you to use whatever **OS** you want in guest machine, it also will obtain an **IP** address from the **DHCP** service deployed for **GUESTS** interface in firewall.

- **VMware**: **VMware** comes with 3 adapters (**VMNET0**, **VMNET1** and **VMNET8**) in default configuration but we need 5 network adapters in our lab. In order to prepare your **VM** (Virtual Machine) open **VMware** machine go to the **Virtual Network Editor** window (**Edit => Virtual Network Editor**) and make

sure networks parameters are set as required. In our lab, we will use one of the adapters (**VMNET8**) as **NAT** (**DHCP** enabled in **VMware** for other machines) and others as host only (**DHCP** disabled, because we need to give static **IP**s for our interfaces and machines in order to control and monitor their network traffic).



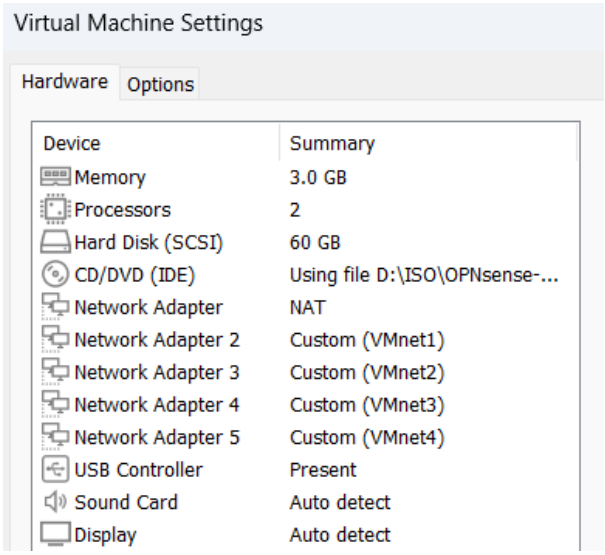




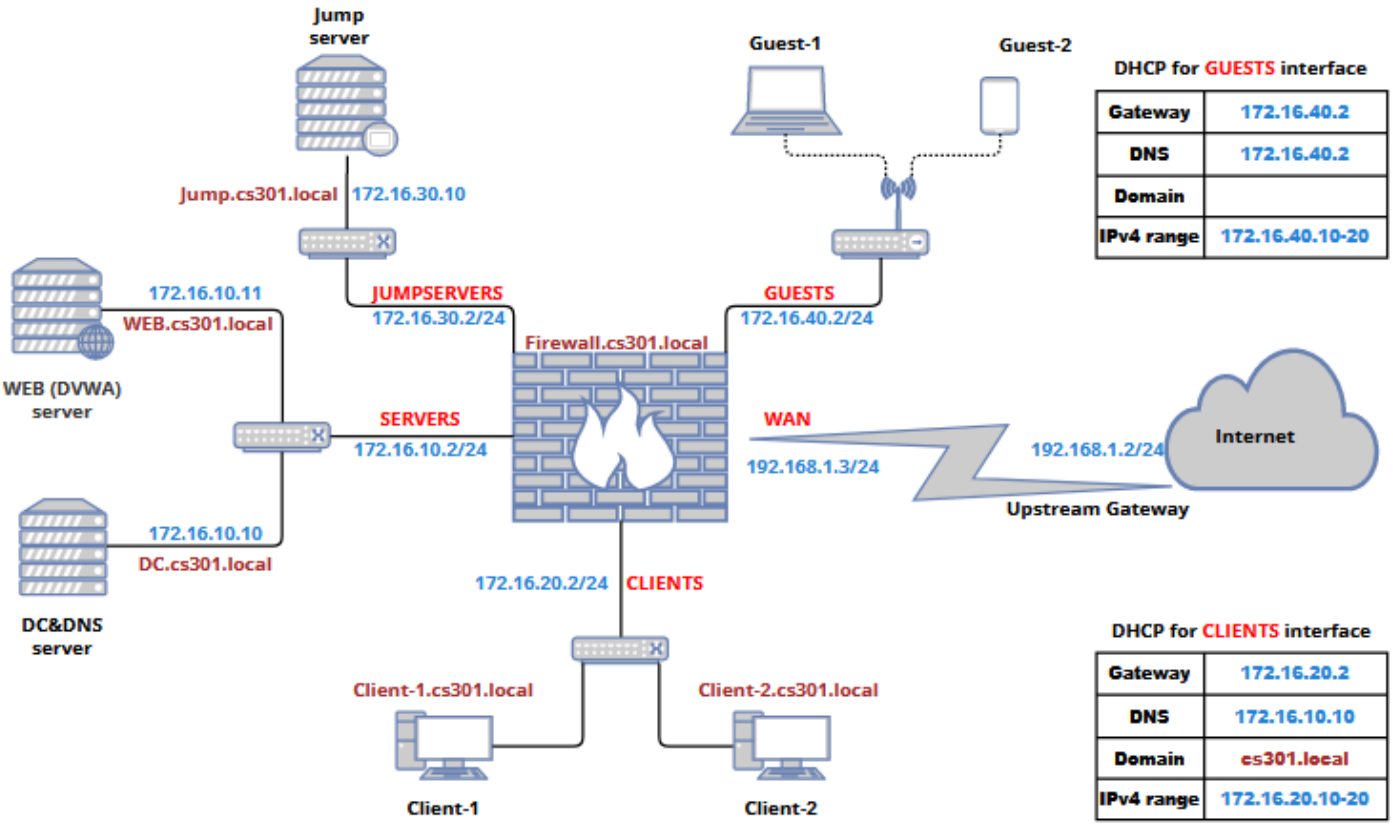
The upstream gateway address (**192.168.1.2**) will let us connect our machines to the Internet (don't confuse it with other gateway addresses).

- **Firewall**: We will use an open source firewall, **OPNsense**, in our lab environment. **OPNsense** has many features which enterprise firewalls offer us. You can download its **ISO** file from **OPNsense**'s official web site. Once you downloaded the **ISO** file, create new machine in **VMware** and add all adapters required. The figure displays the example to the configuration in **VMware** (firewall) **Settings** window.

  Link: https://mirror.ams1.nl.leaseweb.net/opnsense/releases/24.1/OPNsense-24.1-dvd-amd64.iso.bz2



The general scheme of our project is demonstrated below, for better understanding, I added 2 guests and 2 clients but we will use only one in each networks.

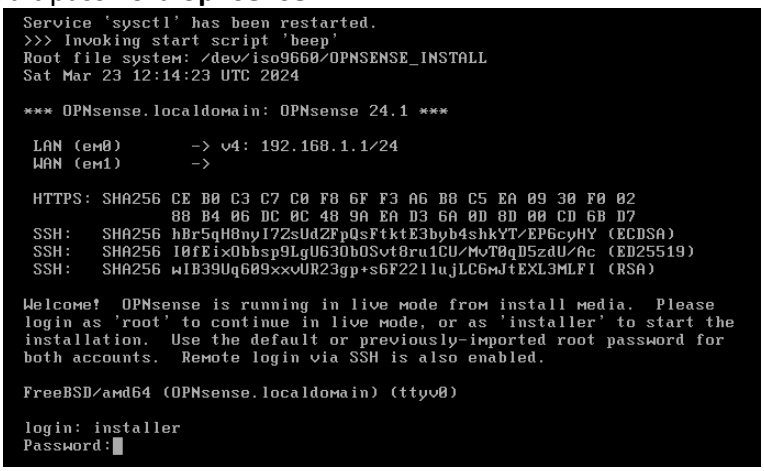


Adapter name	Interface name	Subnet IP	Host IP	Firewall IP
NAT	WAN	192.168.1.0/24	192.168.1.1	192.168.1.3
VMNET1	SERVERS	172.16.10.0/24	172.16.10.1	172.16.10.2
VMNET2	CLIENTS	172.16.20.0/24	172.16.20.1	172.16.20.2
VMNET3	JUMPSERVERS	172.16.30.0/24	172.16.30.1	172.16.30.2
VMNET4	GUESTS	172.16.40.0/24	172.16.40.1	172.16.40.2

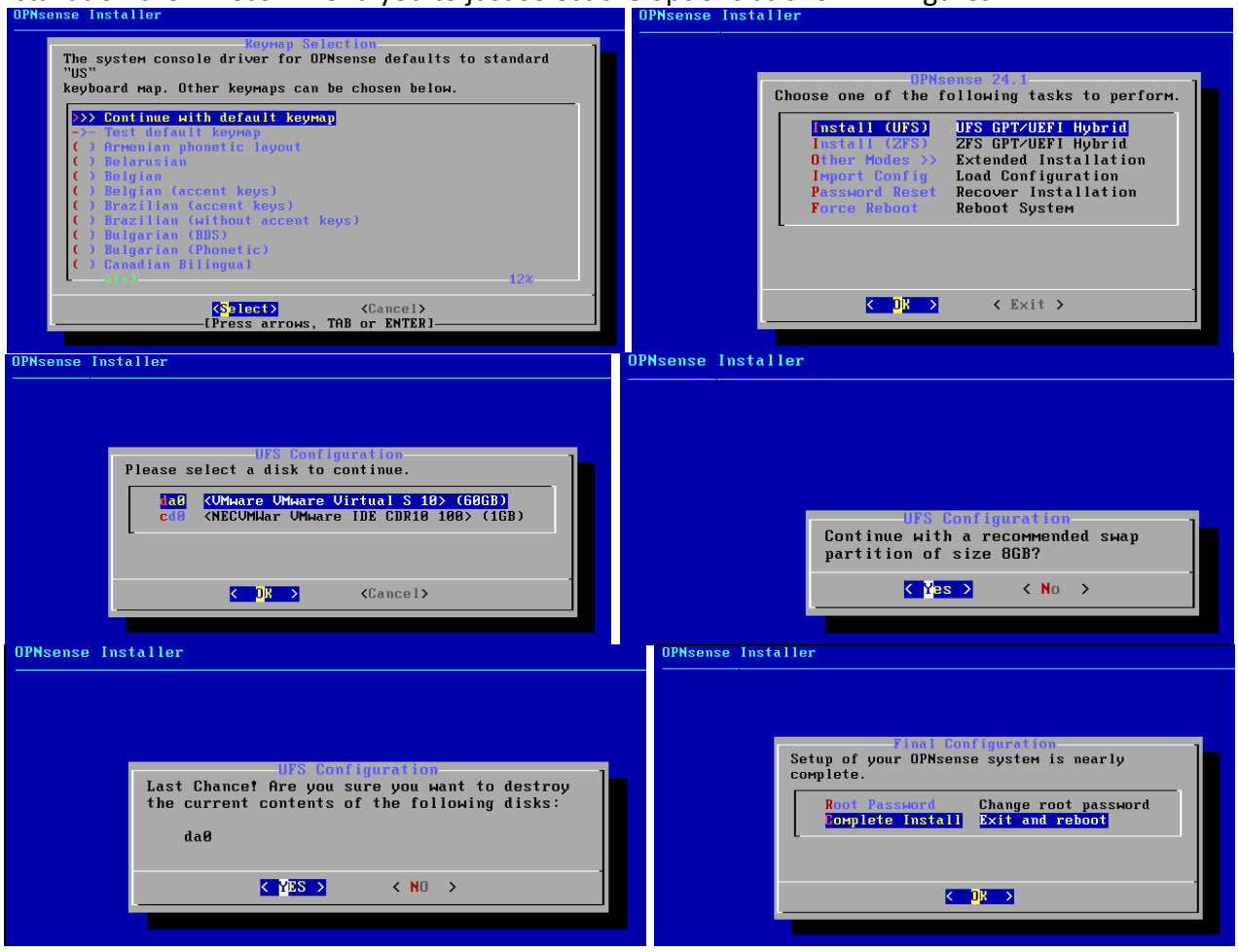
OPNsense setup and general configuration

1.1 Installing OPNsense firewall

Once we start installing process, the following window will be opened, we need to login with the **installer** user by entering the default password **opnsense**.

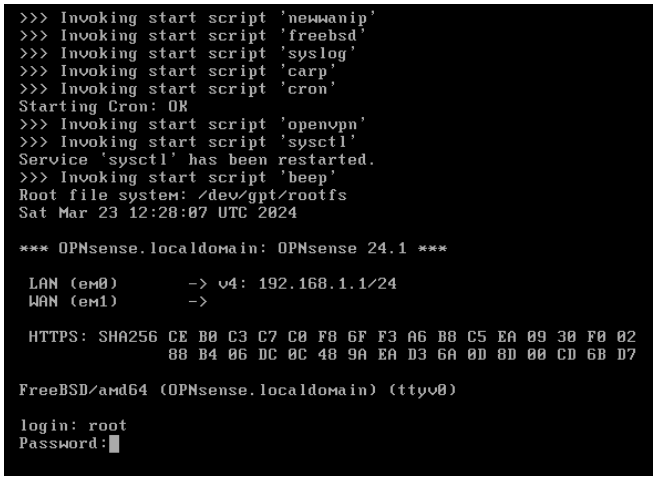


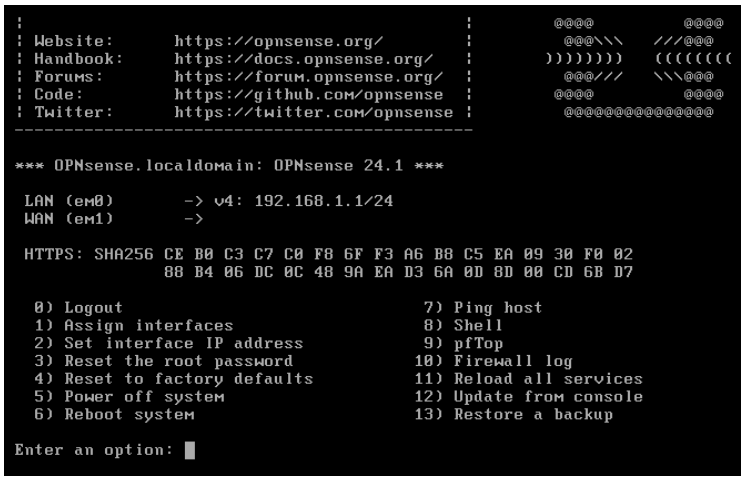
After successful login into the firewall the following windows will popped up, if you are not familiar with this installation then I recommend you to just select the options as shown in figures.



1.2 Initial configuration

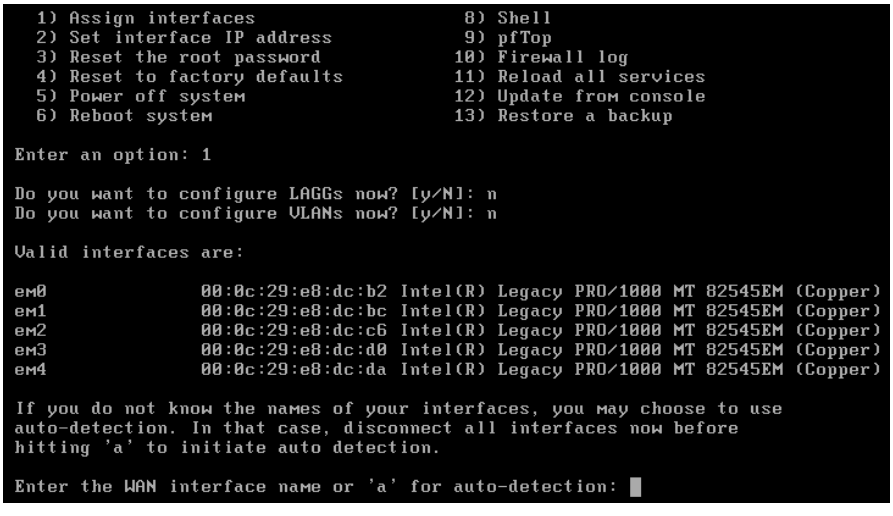
Rebooting the system will complete the installation and our firewall will be ready for configuration. Before we can use the firewall on **GUI** (Graphical User Interface), we need to login into **CLI** (Command Line Interface) with default credentials (username: **root** | password: **opnsense**) and configure at least two interfaces parameters, **NAT** and **LAN**, we will use **LAN** interface **IP** to open the **GUI**.



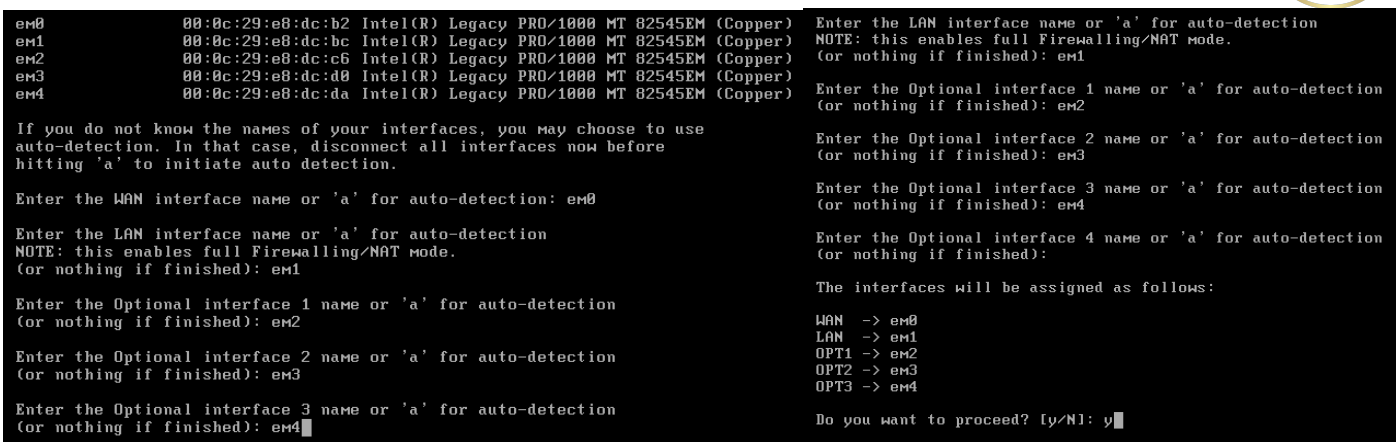


After login into firewall, we will start with assigning interfaces. In the command line type **1** and press **Enter** to open interfaces page, for the **LAGGs** and **WLANs** sections choose **No**. As we see in the figure, it will display valid interfaces for our firewall, the names (**em0**, **em1**, **em2** and **em3**) are default assigned names by **OPNsense**, and they are arranged from up to down as we entered in **VMware** firewall **VM** settings (right click on **VM** and go to **settings** in popped window), **em0**=**NAT**, **em1**=**VMNET1**, **em2**=**VMNET2**, **em3**=**VMNET3** and **em4**=**VMNET4**.

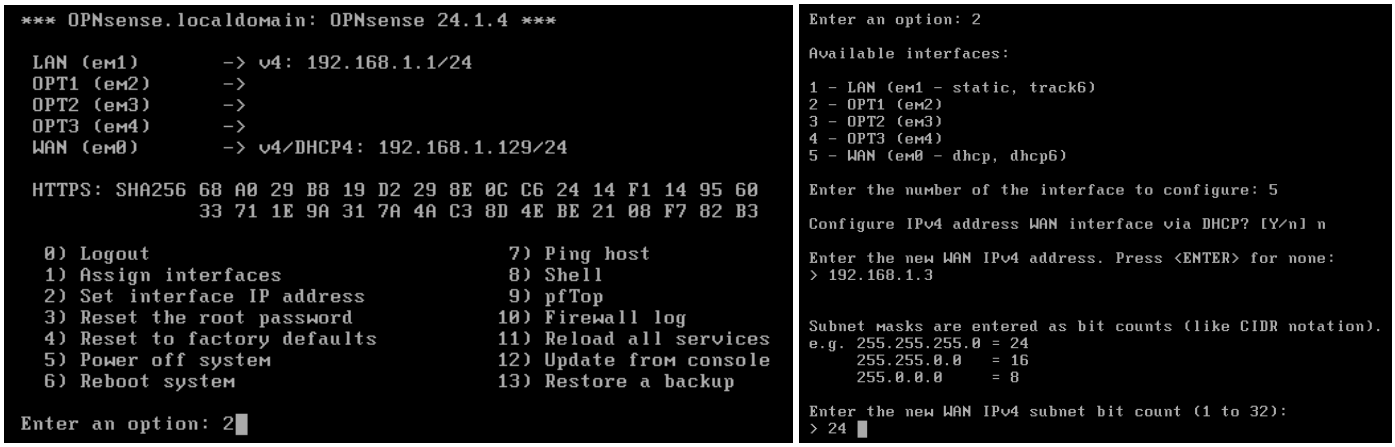
NOTE: Make sure that the **MAC** addresses of interfaces are same with adapters' **MAC** addresses in **VM** settings.



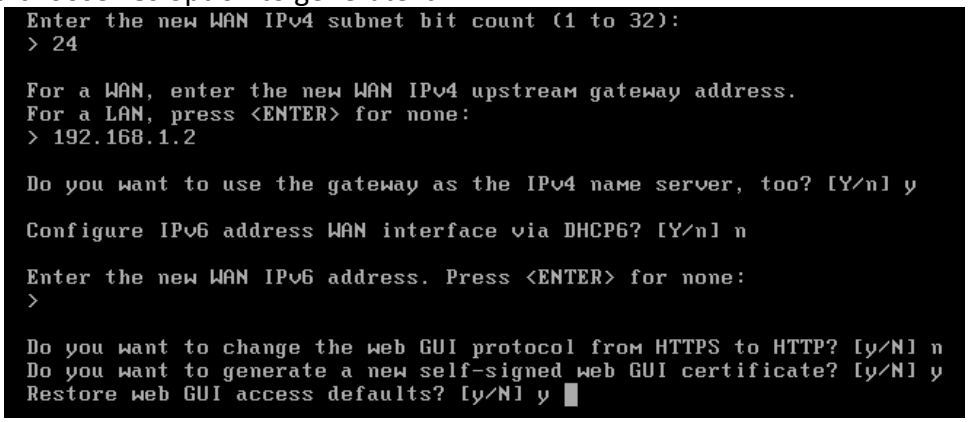
We will be asked to enter the name for **WAN** interface and later for **LAN**, **OPT1** and **OPT2**, add all names and leave empty when it asks **OPT3** name.



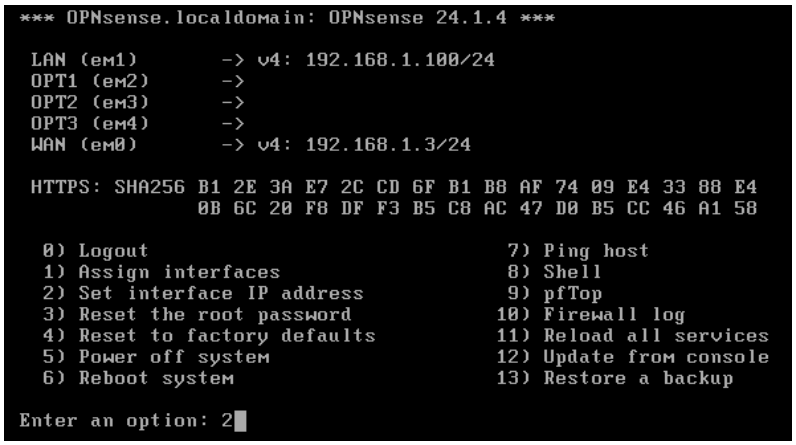
After defining interfaces, we will assign **IPv4** addresses for them. In the command line type **2** and press **Enter** to start assign **IP**s. In our lab we will configure only two interfaces, **WAN** and **LAN**, in **CLI** and others in **GUI**. Lets begin with **WAN** interface by choosing option **4** and enter the parameters as shown in figures below.

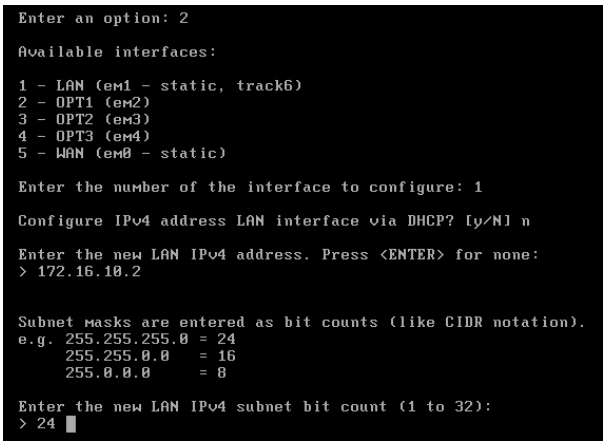


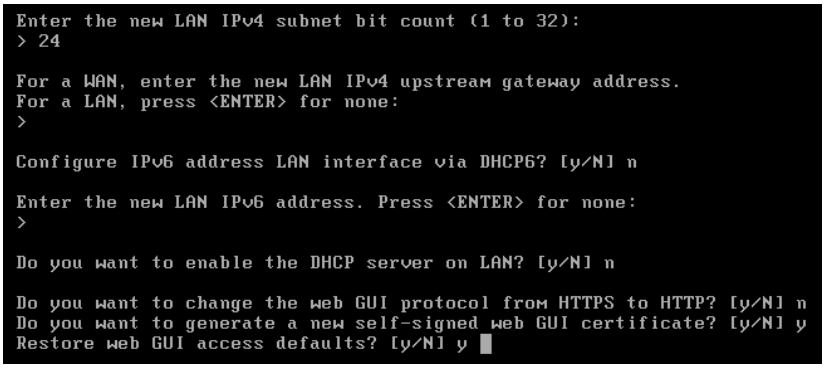
The upstream gateway address (which our firewall will use for connecting to the **Internet**) have to be the same with **NAT** adapter's gateway address as we assigned in **VMware, Virtual Network Editor** section above. In the last second question we will be asked to generate self-signed certificate for using **GUI** in our web browser and choose **Yes** option to generate it.



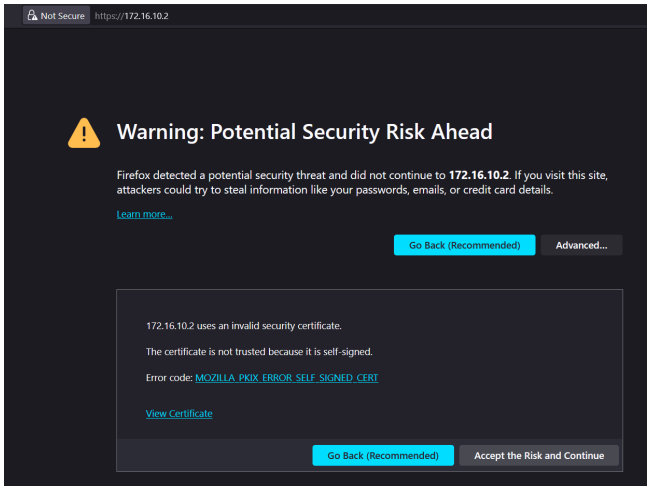
NOTE: in **VMware**, first **IP** address always automatically is assigned to the host itself, so we have to begin to use from the second address for our interfaces, but in **NAT** adapter the second **IP** (**192.168.1.2**) is given to the upstream gateway. This is why we used **192.168.1.3 IP** for **WAN** interface. In other interfaces will use the second **IP**s for our interfaces because in our firewall environment there will be only one upstream gateway address.
After configuring the **WAN**, we will assign the **LAN**'s parameters as shown in following figures.

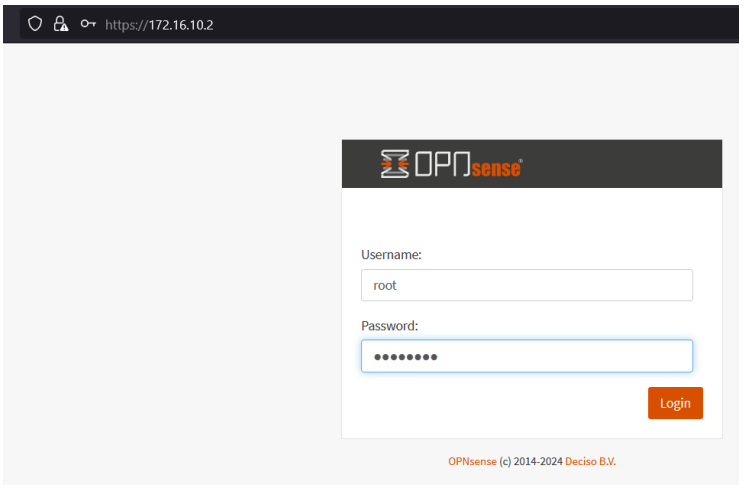




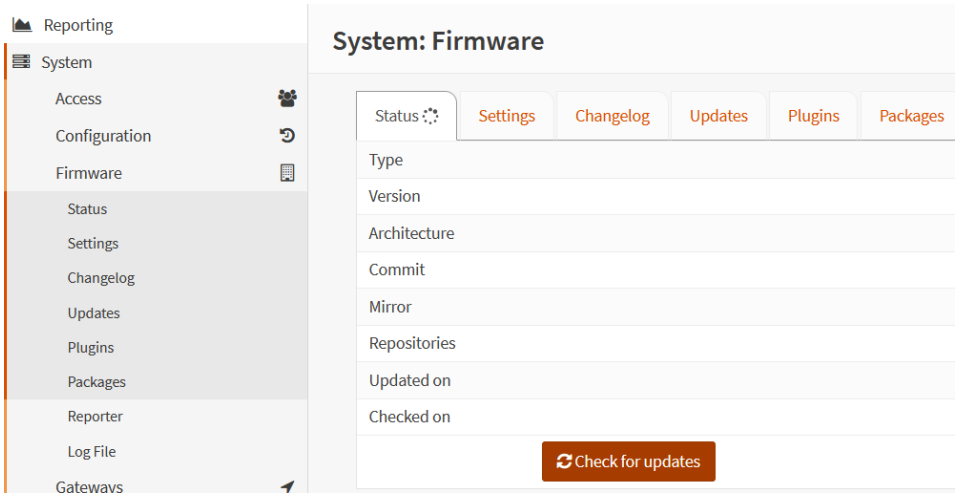


Now we are ready to use **GUI**, open your browser and go to the https://172.16.10.2 address. In the opening windows select **Advanced** and **"Accept the Risk and Continue"**. Later enter the default credentials (username: **root** | password: **opnsense**) and login.

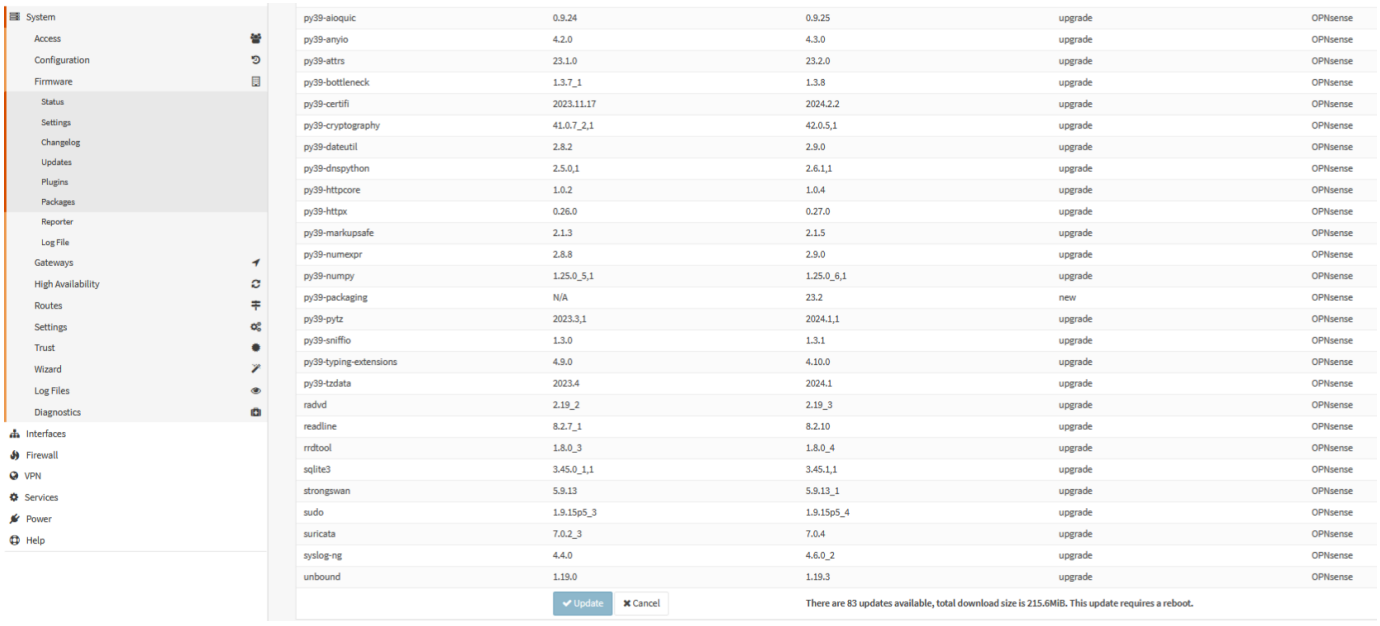




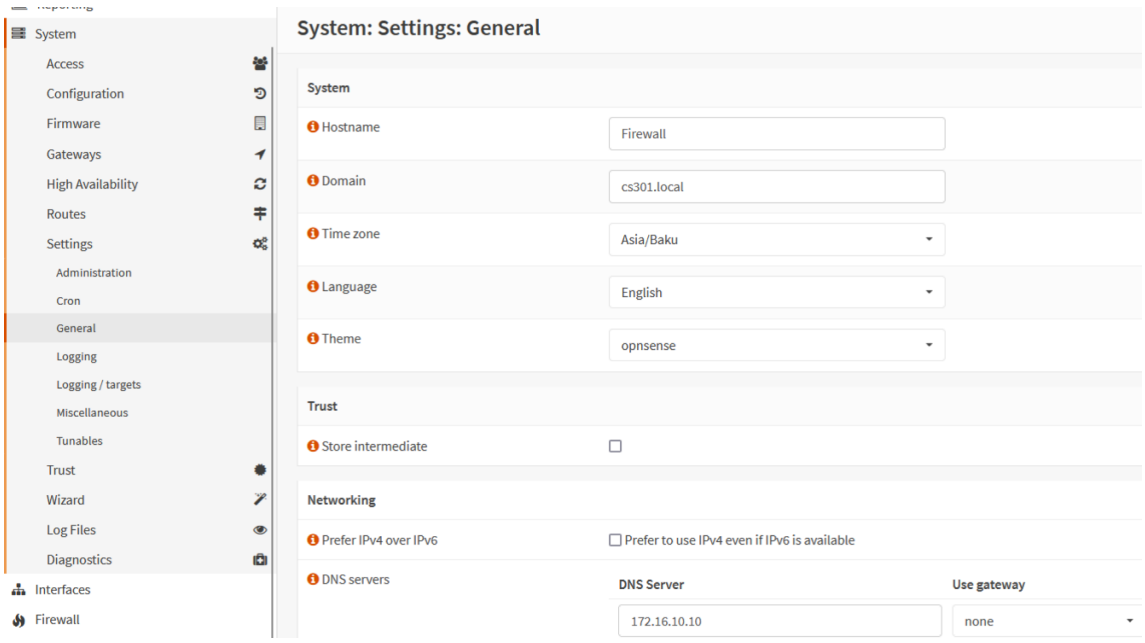
Once we enter the **GUI**, the early step should be updating system. First go to the **System => Firmware => Status** and **"Check for updates"**.



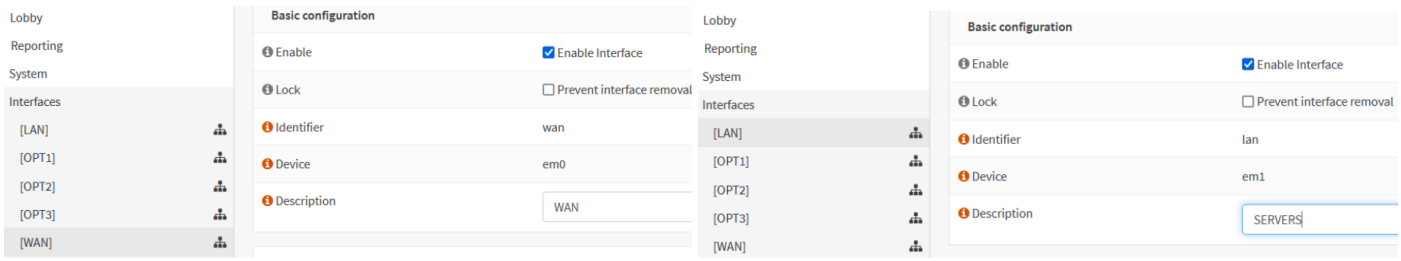
If there are available updates it will be displayed in **System => Firmware => Updates** section. Press **Update** button and confirm the action in popped windows. By the completion of update process, the firewall will reboot itself. After the reboot we will continue our firewall's configuration.



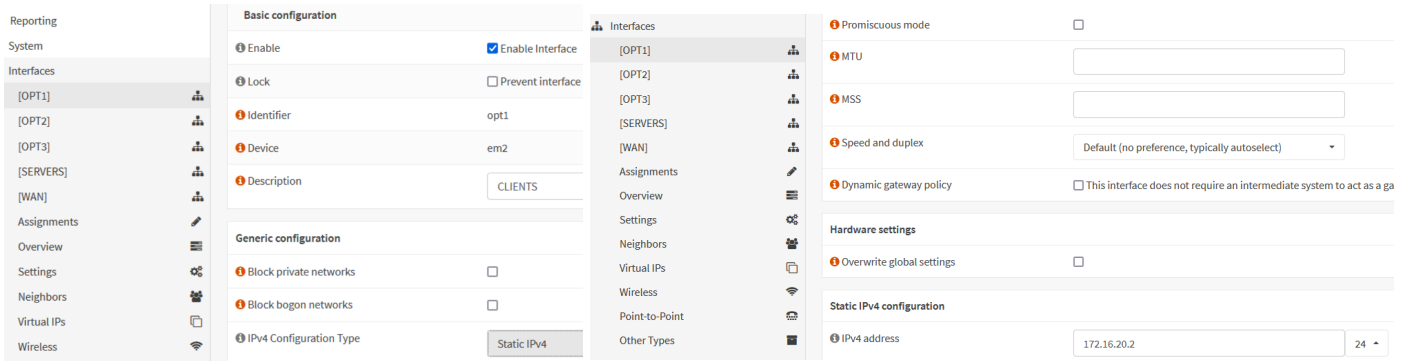
As we see in **Summary** capture, we choose the name **Firewall** for our **OPNsense** firewall and our domain is **cs301.local**, the time zone parameter has to be the same as we use in our other machines and it is also essential for logs because when there is an incident in our network, we have to know the exact time to take an action against it. We used our **DC** machine as our **DNS** server in the domain, so in this section we have to add **DC**'s **IPv4** address to synchronize our firewall with **DNS** service.



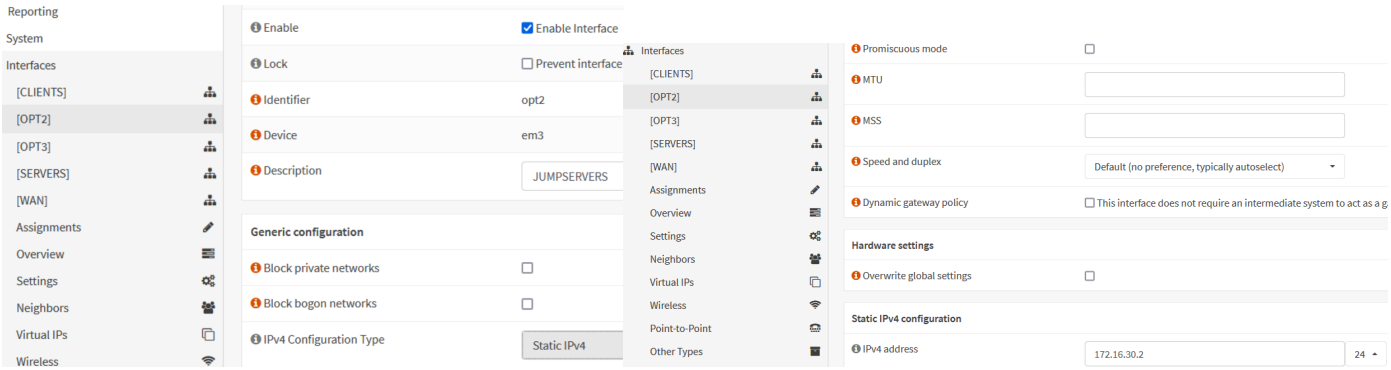
The next configuration will be assigning descriptive names and **IP** addresses (we didn't assign **IP**s to **OPT1**, **OPT2** and **OPT3** interfaces yet) to our firewall interfaces.
For the **WAN** interface we add **WAN** into the **Description** section, for the **LAN** interface we assign **SERVERS** name, which will be used for our servers in the following captures and leave rest of configurations as default.

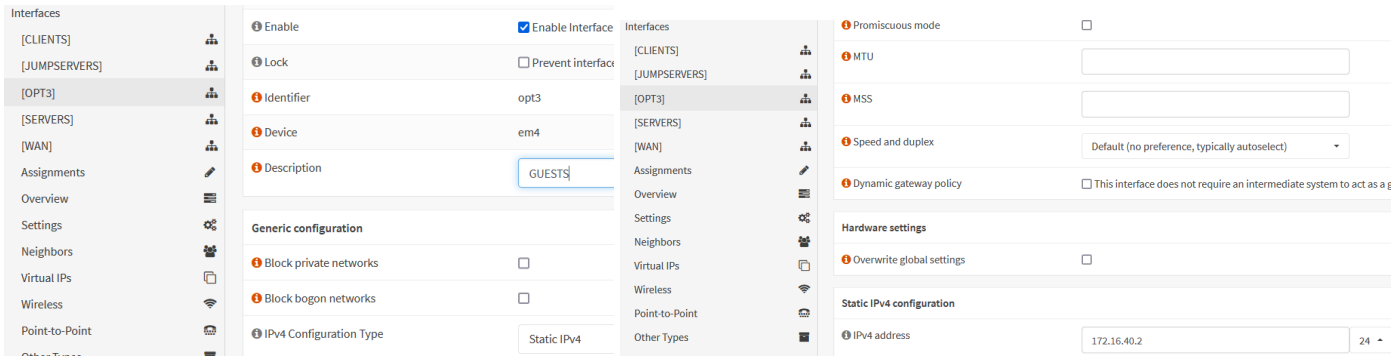


For **OPT1** type **CLIENTS** which will be used for our client machines and choose **"Static IPv4"** in **"IPv4 Configuration Type"** section, scroll down and in the **"IPv4 address"** section assign **172.16.20.2 IP** and set subnet **24** for **CLIENTS** interface. Later we will activate a **DHCP** service for this interface.

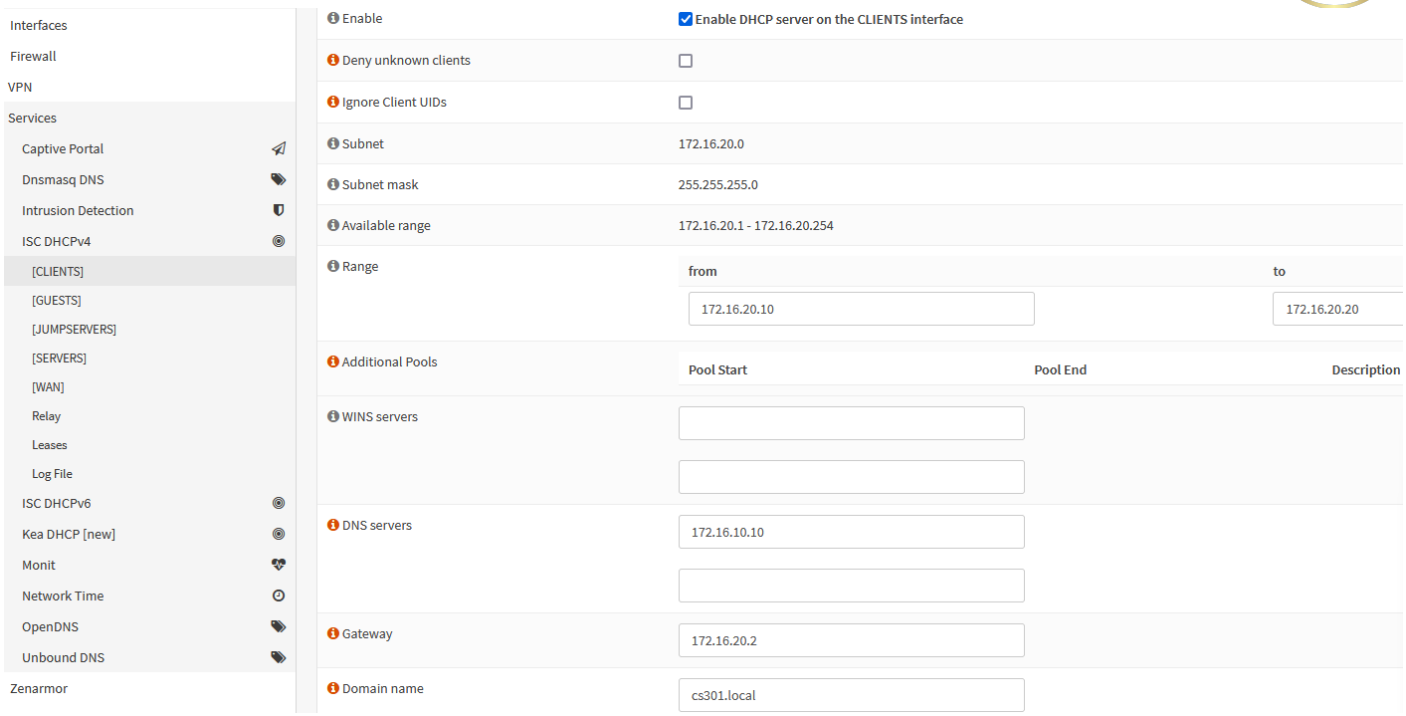


Repeat the same configuration for **OPT2** and **OPT3** interfaces, as we did for **OPT1**, by using the parameters which displayed in figures below. This interface will be used for controlling remotely our **DC** machine, which is located in **SERVERS** interface.

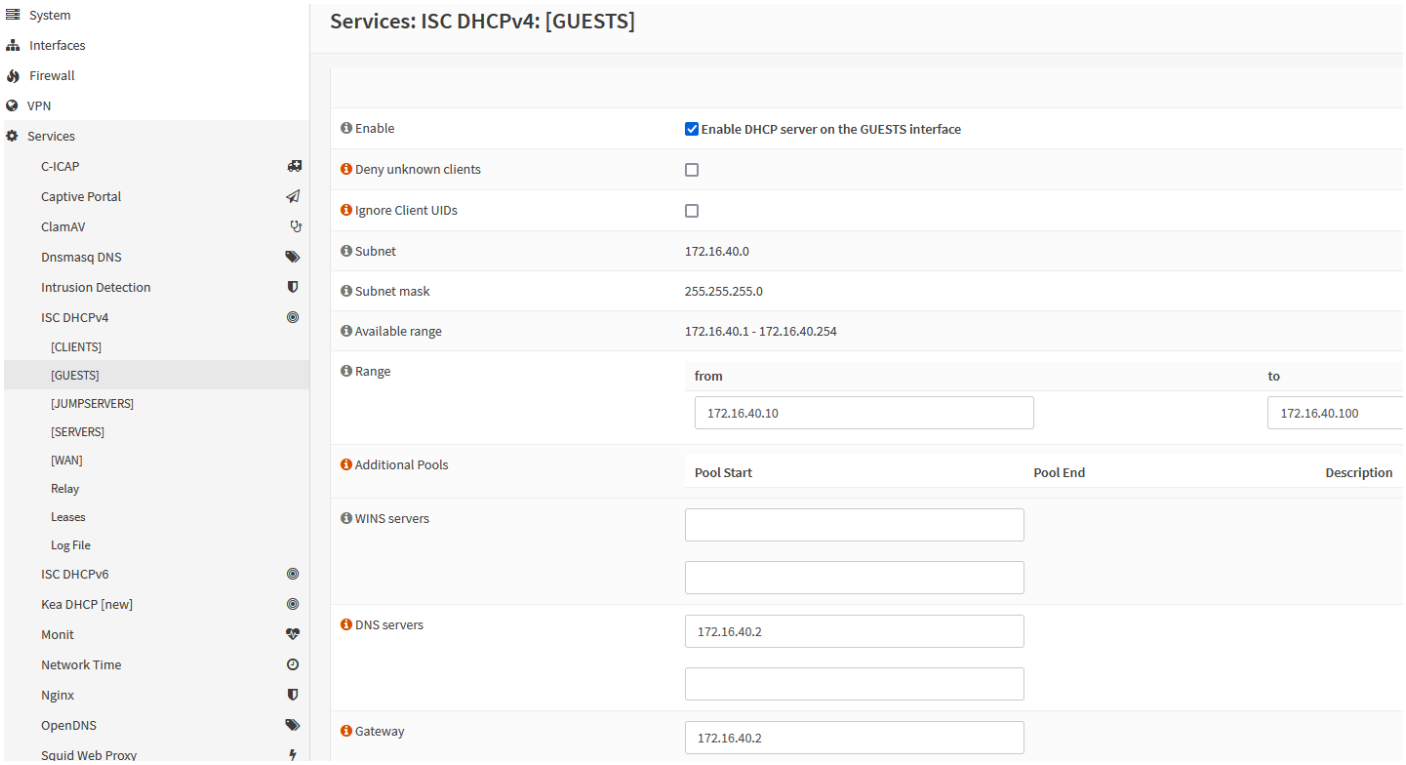




After successfully configuring all interfaces, we are ready to **DHCP** service for **CLIENTS** interface. This service will run in lab environment, so it is enough to give range short as I did, **172.16.20.10-20**, and **DNS** address have to be same in whole network, which is the **IPv4** address of **DC** server in our lab. The gateway address is the address of **CLIENTS** interface, but apart from upstream gateway (which we assigned for **WAN** interface), this gateway will serve only for local communication between subnets (**WAN**, **SERVERS** and **JUMPSERVERS**). Last we add our domain, **cs301.local**, and finish our configuration for the **DHCP** service. We can monitor and control our services on **Lobby => Dashboard => Services** section.

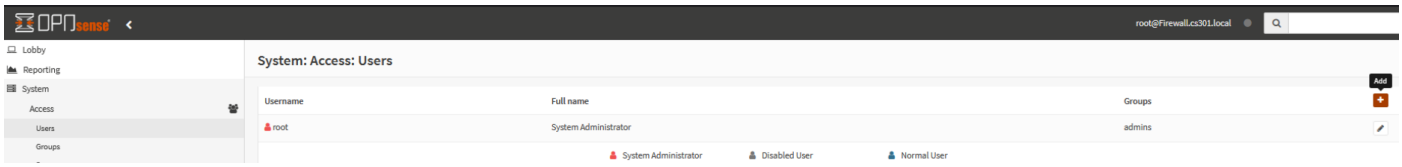


DHCP settings for **GUESTS** network.

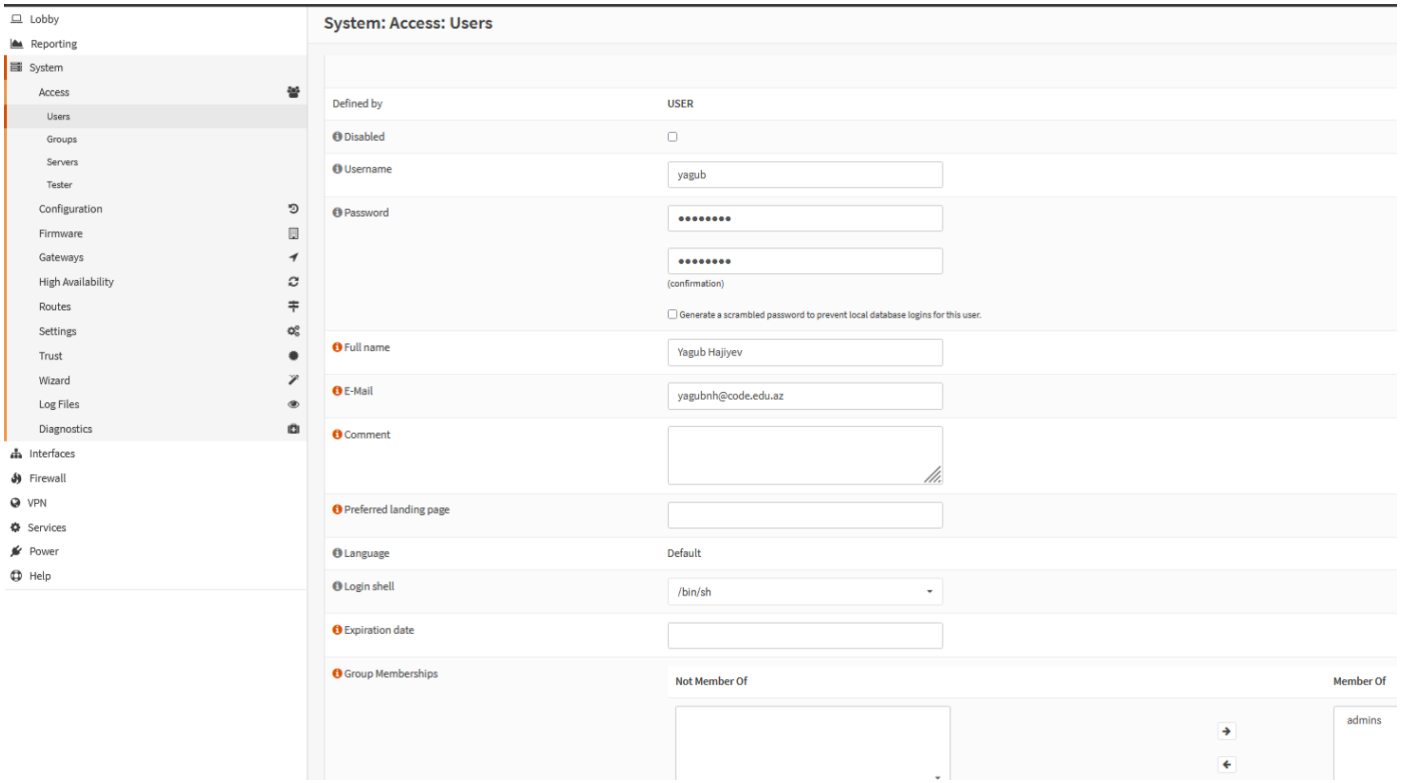


1.3 Firewall access configuration

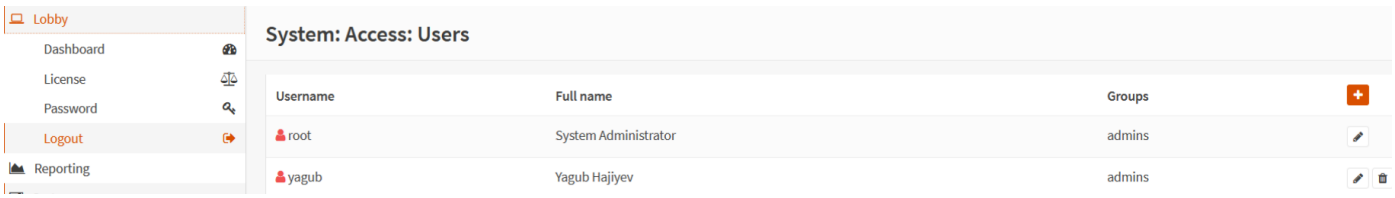
For the security purposes it is not recommended to use default users, credentials and ports. In this capture we will create a new user, disable the default **root** user and change access ports to make our firewall environment much more secure.
In order to create a new user, go to the **System => Access => Users** section and press plus (**+**) button to add new user.

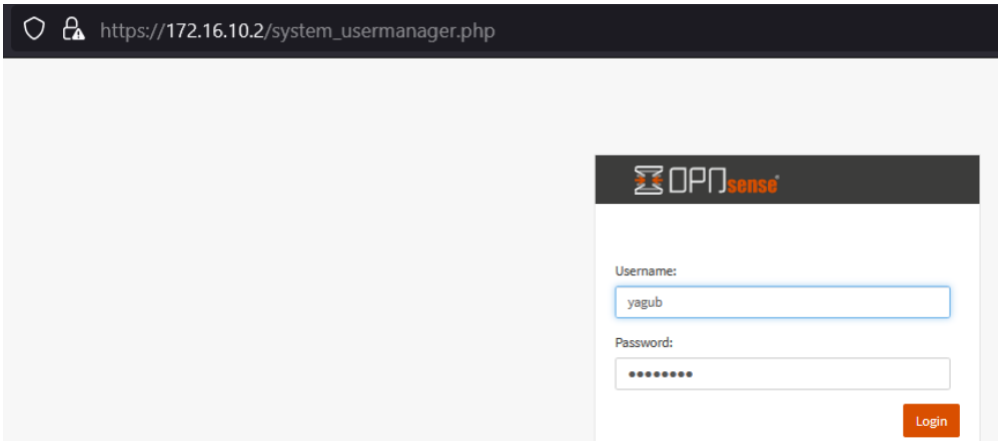


In the opening window, add your credentials, choose your **"Login shell"** and add your user to the **admins** group which will allow this new user to control the firewall with **root** privileges.



After creating the user logout from **root** user and login with the new user.



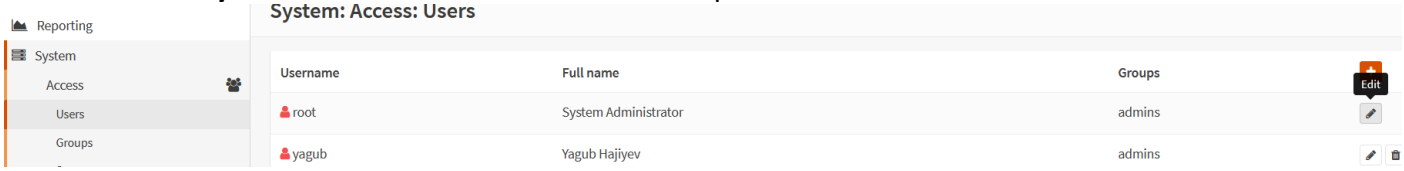


Before we disable the **root** user, we need to activate **wheel** group in our firewall, otherwise we will not be able to use **root** privileges. Open the console (**CLI**) and open the **sudoers** file which is located under **/usr/local/etc/sudoers** path and take the comment hash (**#**) in front of **"%wheel ALL=(ALL:ALL) ALL**" line. (you can install **nano** tool with **"pkg install nano"** command)

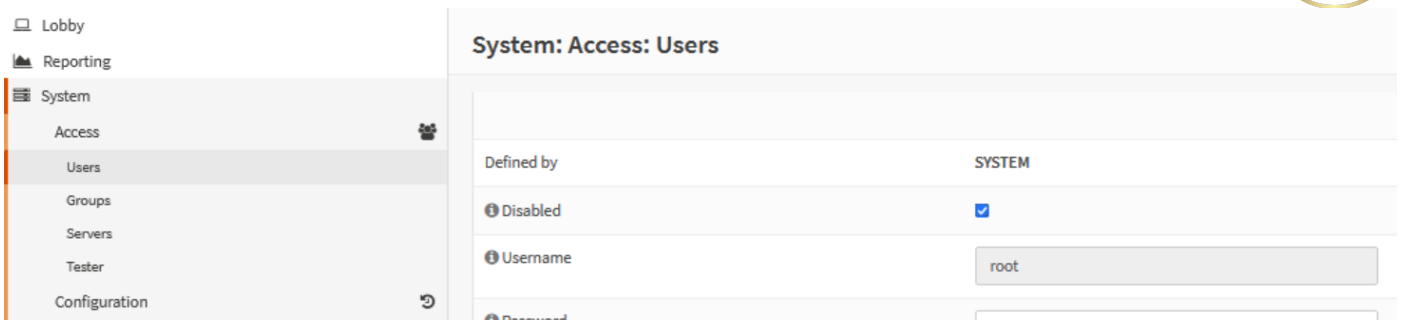
```
root@Firewall:~ # nano /usr/local/etc/sudoers
```

```
 GNU nano 7.2                    /usr/local/etc/sudoers

## Uncomment to allow members of group wheel to execute any command
# %wheel ALL=(ALL:ALL) ALL

## Same thing without a password
# %wheel ALL=(ALL:ALL) NOPASSWD: ALL

## Uncomment to allow members of group sudo to execute any command
#%sudo ALL=(ALL:ALL) ALL

## Uncomment to allow any user to run sudo if they know the password
## of the user they are running the command as (root by default).
# Defaults targetpw  # Ask for the password of the target user
# ALL ALL=(ALL:ALL) ALL  # WARNING: only use this together with 'Defaults targe

## Uncomment to show on password prompt which users' password is being expected
# Defaults passprompt="%p's password:"

## Read drop-in files from /usr/local/etc/sudoers.d
@includedir /usr/local/etc/sudoers.d

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

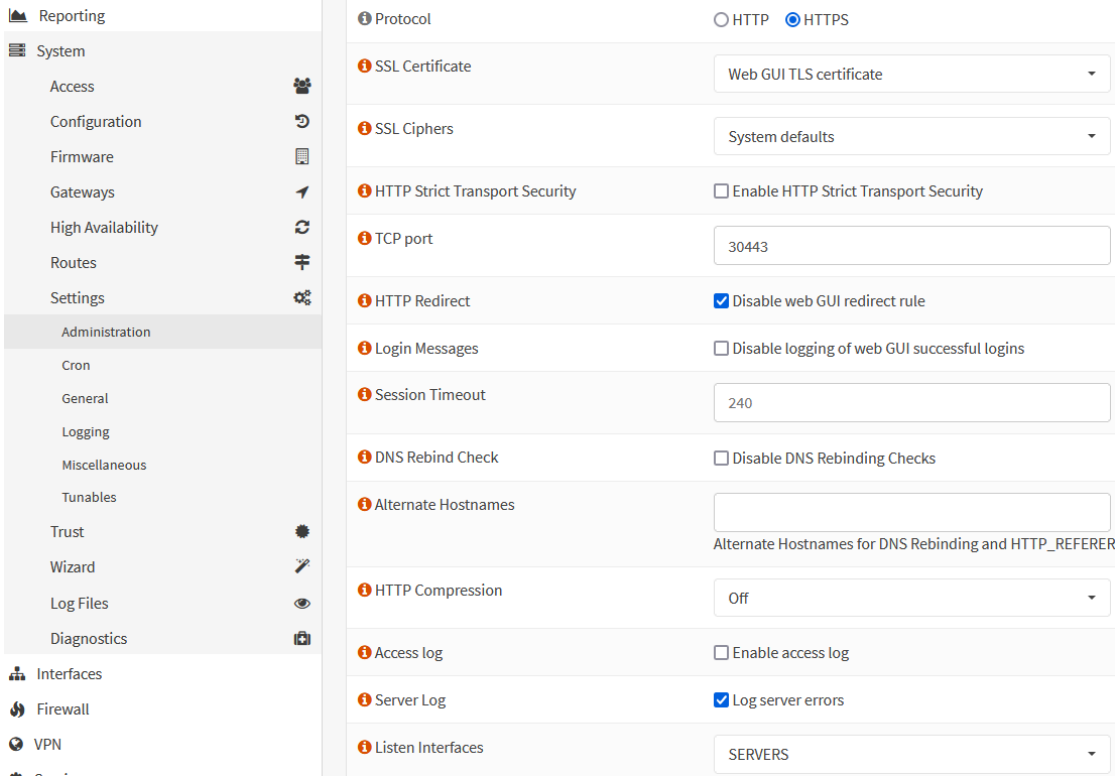
Go back to the **System => Access => Users** section and press **Edit** button in **root** user's line.



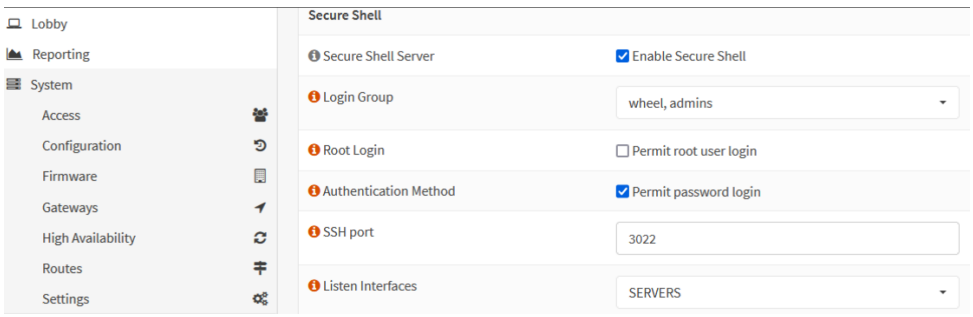
In the opened window, disable (put the tick in **Disabled** box) the root user and save. Now you can't use the **root** user login to the firewall until you enable it again.



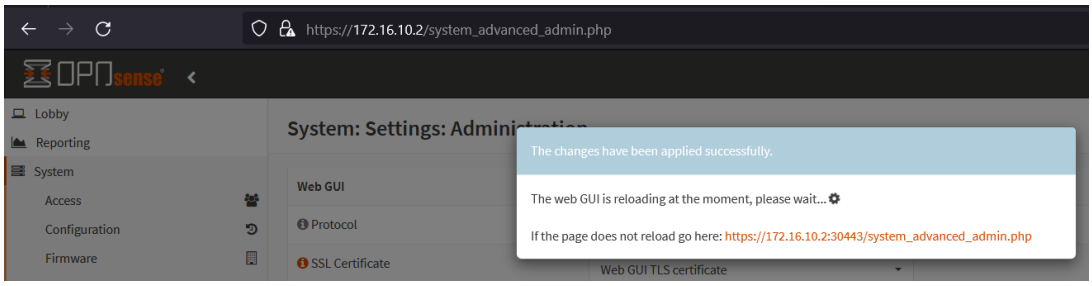
In the next step we have to change the **HTTPS** and **SSH** access ports. Go to the **System => Setting => Administration** section and change the default **"TCP port"** from **443** to whatever number you want for **HTTPS** access, I used **30443** in this lab, and set the **SERVERS** interface in the **"Listen Interfaces"** section, which will allow only **SERVERS** and block other interfaces for entering to the **GUI** interface.



After changing **HTTPS** port, scroll down, enable **Secure Shell**, change the **"SSH port"** and set **SERVERS** in the **"Listen interfaces"** and save configuration changes.

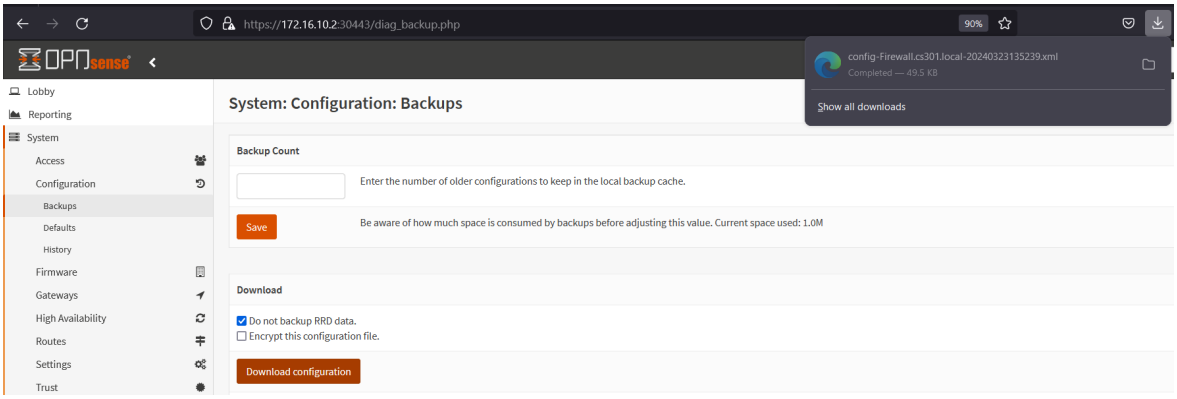


Once you hit the Save button, the system will pop up the window displayed in the figure, and will require to go new address, just click the **URL** and go to the new page. This is happed because we changed the default **HTTPS** port number and this will add extra security layer to our firewall, which means if an attacker wants to gain access our GUI interface then he/she has to find the correct port number to load login page.



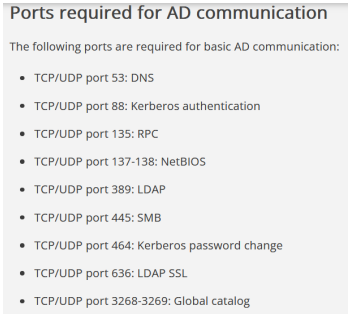
1.4 Configuration backup

When we have to make a critical configuration changes or updates the system, it sometimes causes system crash problems. One of the recommended step is to always taking backups of the system just before doing any operation. Go to the **System** => **Configuration** => **Backups** section and download the current configurations, later you can use this **XML** file to restore your configuration in this section.

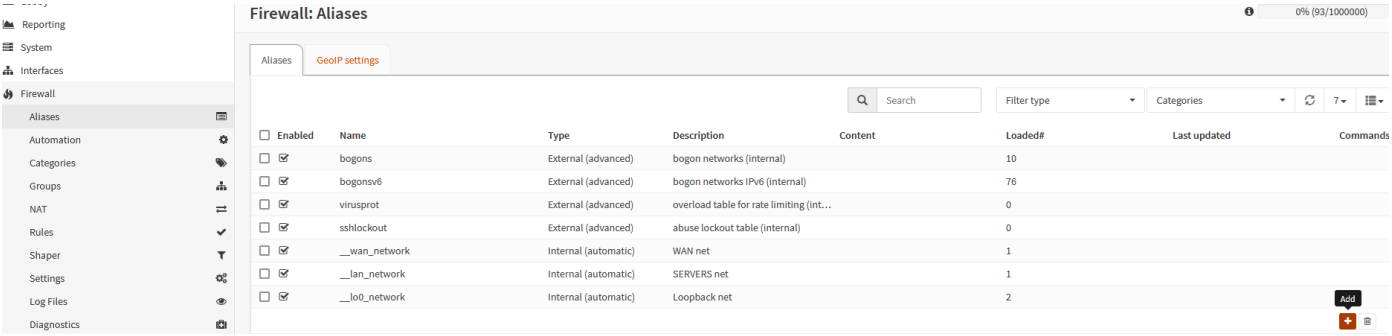


1.5 Aliases

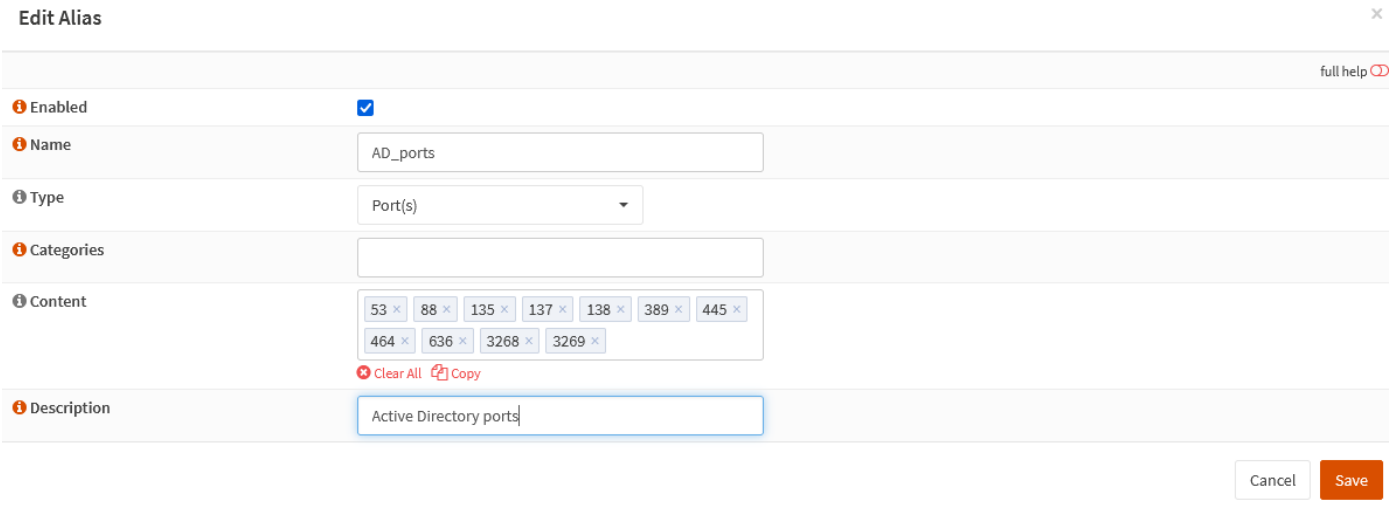
In firewall environment, aliases help us to list ports, hosts and networks under a single name, which we can use just calling the alias's name in various entities of firewall, such as firewall rule. Below in the figures, we will create the alias of **Active Directory** ports for using it in different firewall rules. This will reduce the effort we spend on each rule, because we will call just the alias name in different rules, instead of adding ports into the rules each time.



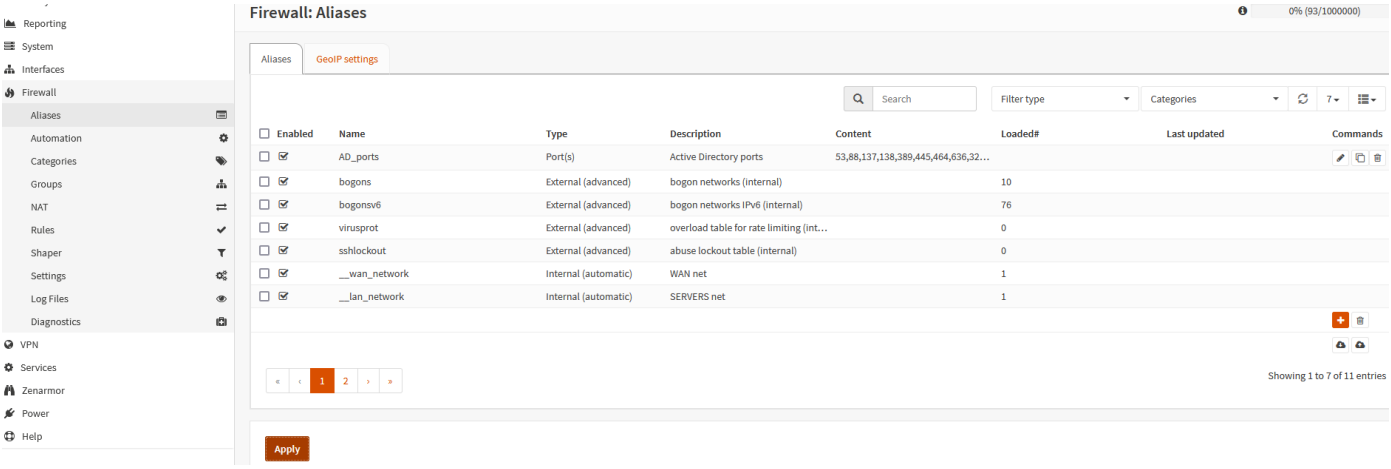
There are a few default aliases in **OPNsense** firewall, which can be found in **Firewall => Aliases** path. Press the **Add** button to add new alias.



In the **"Edit Alias"** window, type the name of alias in the **Name**, select **Port(s)** option in the **Type**, add the ports to the **Content** section and type a short description whatever you want.



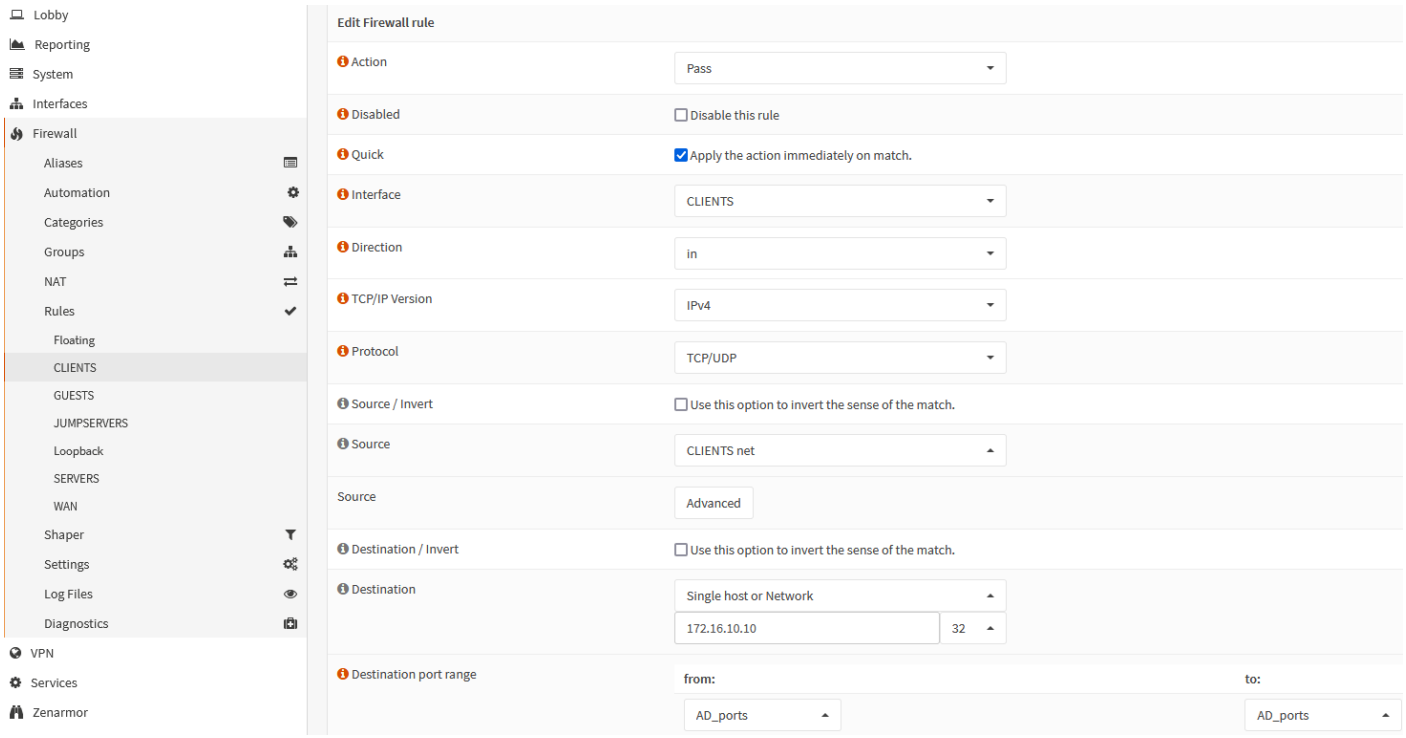
Once you **Save** and close the popped up window, your new alias will be displayed in **Firewall => Aliases** section, then press the **Apply** button to make it available in firewall environment. And it is done, we are ready to use it wherever we want in our firewall.



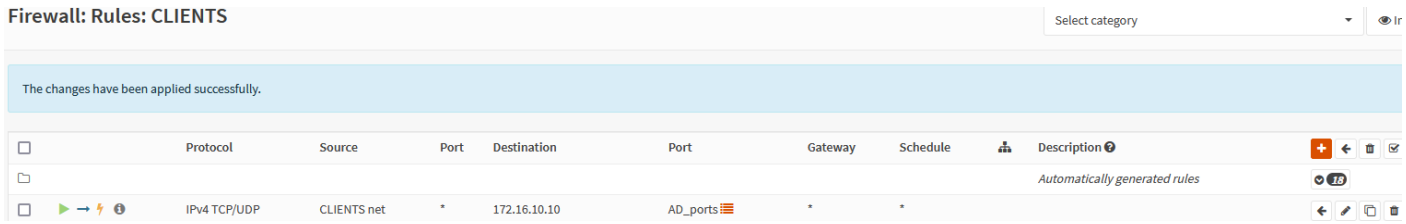
1.6 Writing initial rules

1.6.1 CLIENTS rules

In this capture we will write a few rules for **CLIENTS** interface to allow and block some actions. The short summary will be like this: for **CLIENTS** interface machines, allow **AD** ports in **DC** and block others, allow **RDP** port in **JUMPSERVERS** interface and block others. Lets begin with allowing **AD** ports in **DC**.
Go to the **Firewall => Rules => CLIENTS**, add new rule, in the rule page, select Pass for **Action**, select **CLIENTS** in Interface, select **TCP/UDP** for **Protocol**, select **"CLIENTS net"** as **Source**, select **"Single host or Network"** in **Destination** and type the **DC** address with **CIDR 32**, select your alias name for **AD** ports (which is **AD_ports** in this lab) in **"Destination port range"** section and other options as default.

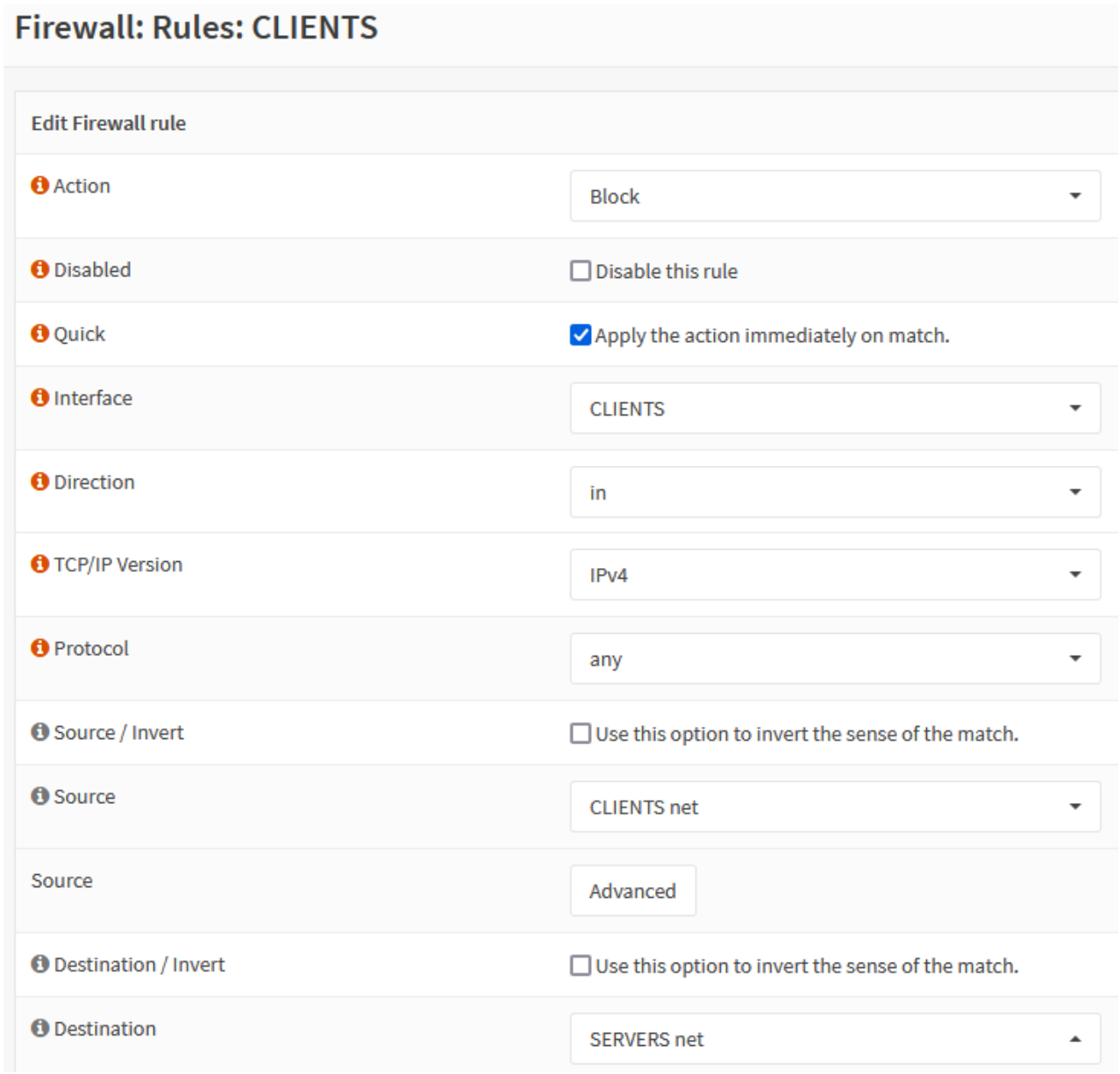


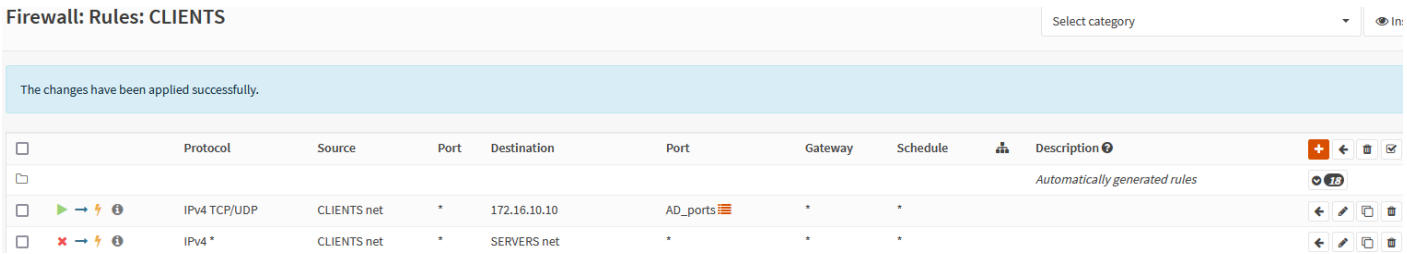
Our rule will be added in **Firewall => Rules => CLIENTS** section at the bottom of existing rule. Every new created rule locates at the bottom of all other existing rules, we will arrange them in this at the end of this capture.



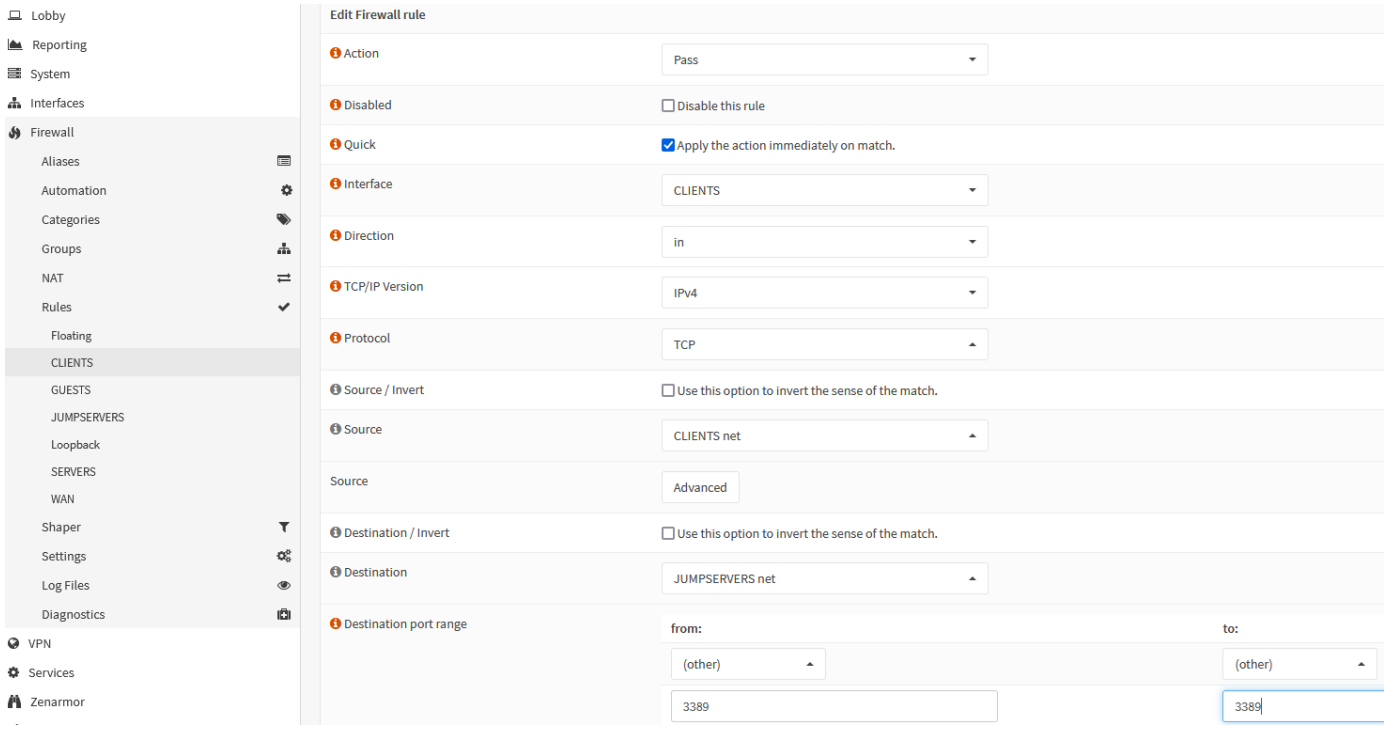
Now we need to add new rule to block all other connections with **SERVERS** network. Again add new rule

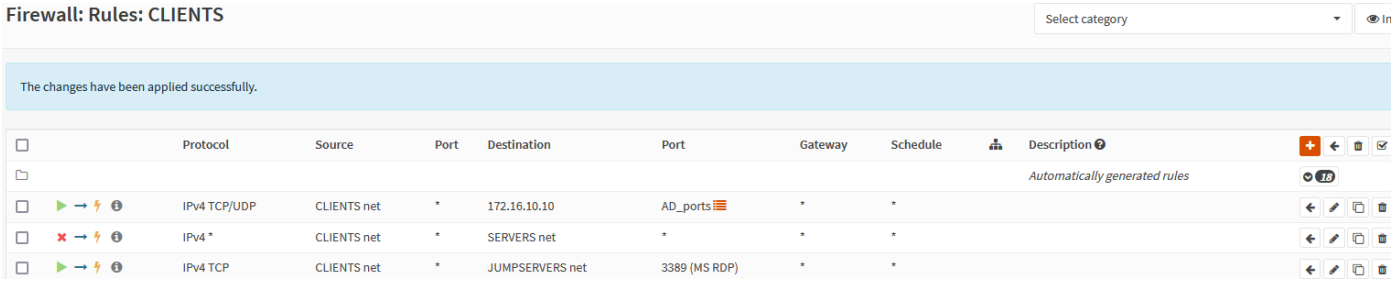
and in the rule window, select **Block** for **Action**, select **CLIENTS** for Interface, select **"CLIENTS net"** as source, select **"SERVERS net"** option as the **Destination** and leave all other options as default. This rule will block all traffic goes from **CLIENTS** network to the **SERVEFRS** network.



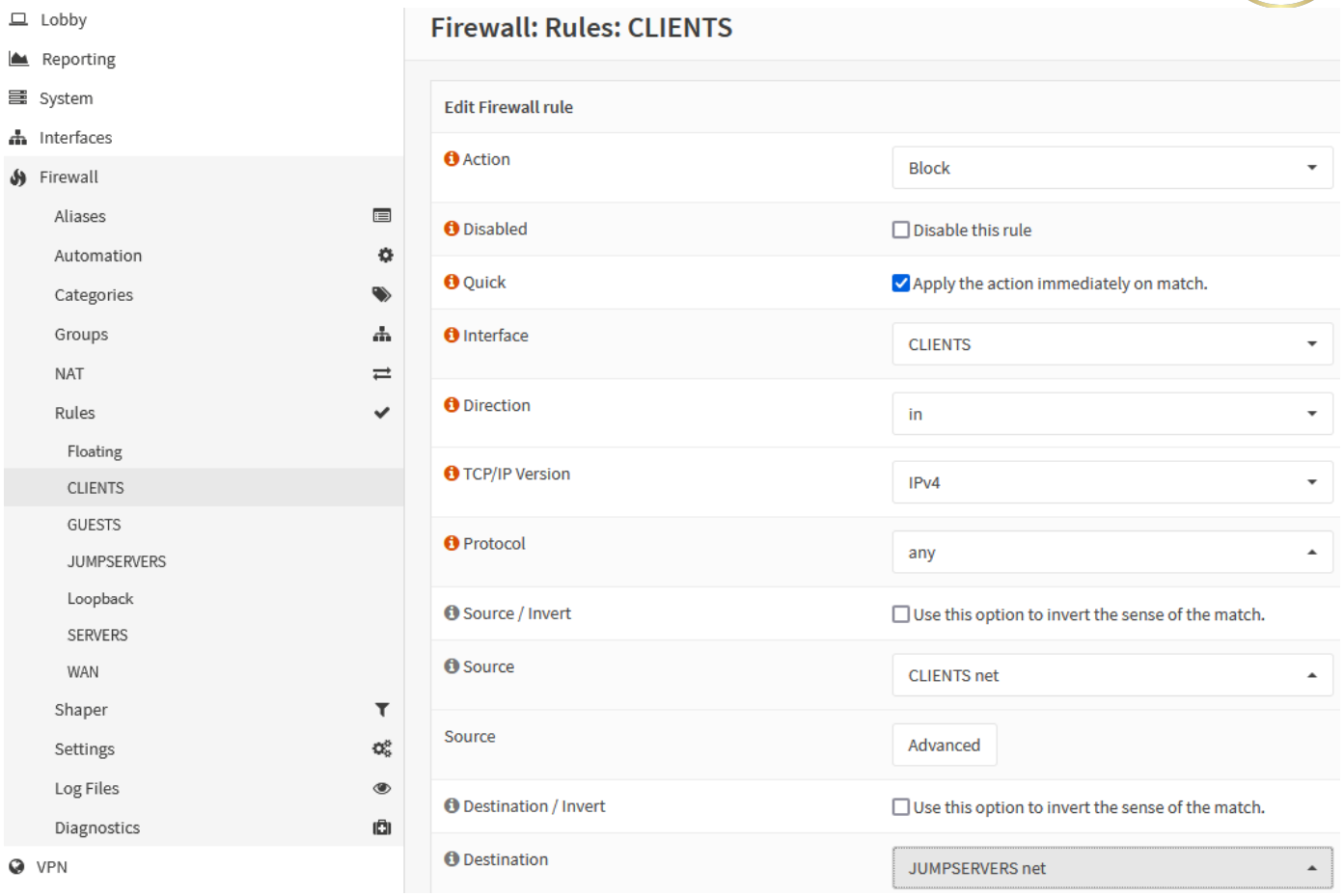


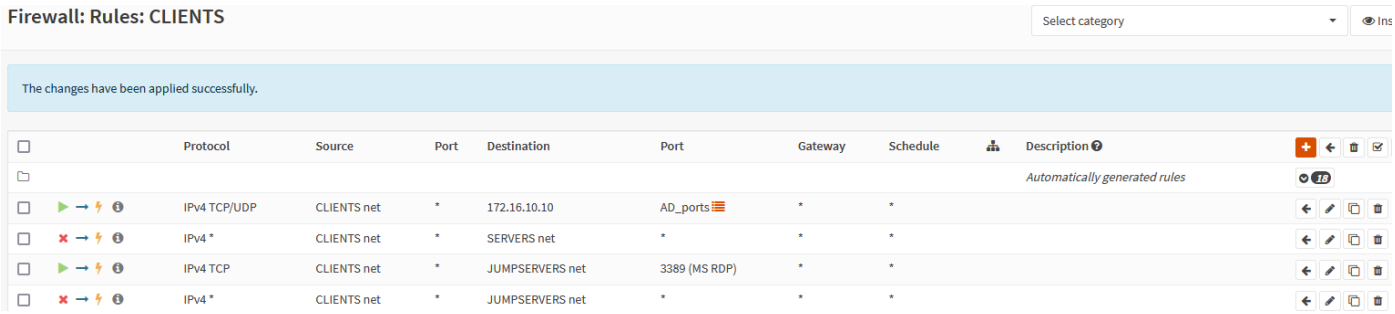
In the next rule we will allow only **RDP** port (**TCP 3389**) access from **CLIENTS** to **JUMPSERVERS**. In the rule, select **Pass** for **Action**, select **CLIENTS** for interface, select **TCP** as **Protocol**, select **"CLIENTS net"** as **Source**, select **JUMPSERVERS** as **Destination**, select "**(other)**" option in **"Destination port range"** and enter the **RDP** port in both, **from** and **to** section, then leave all other options as default and **Save** the rule.



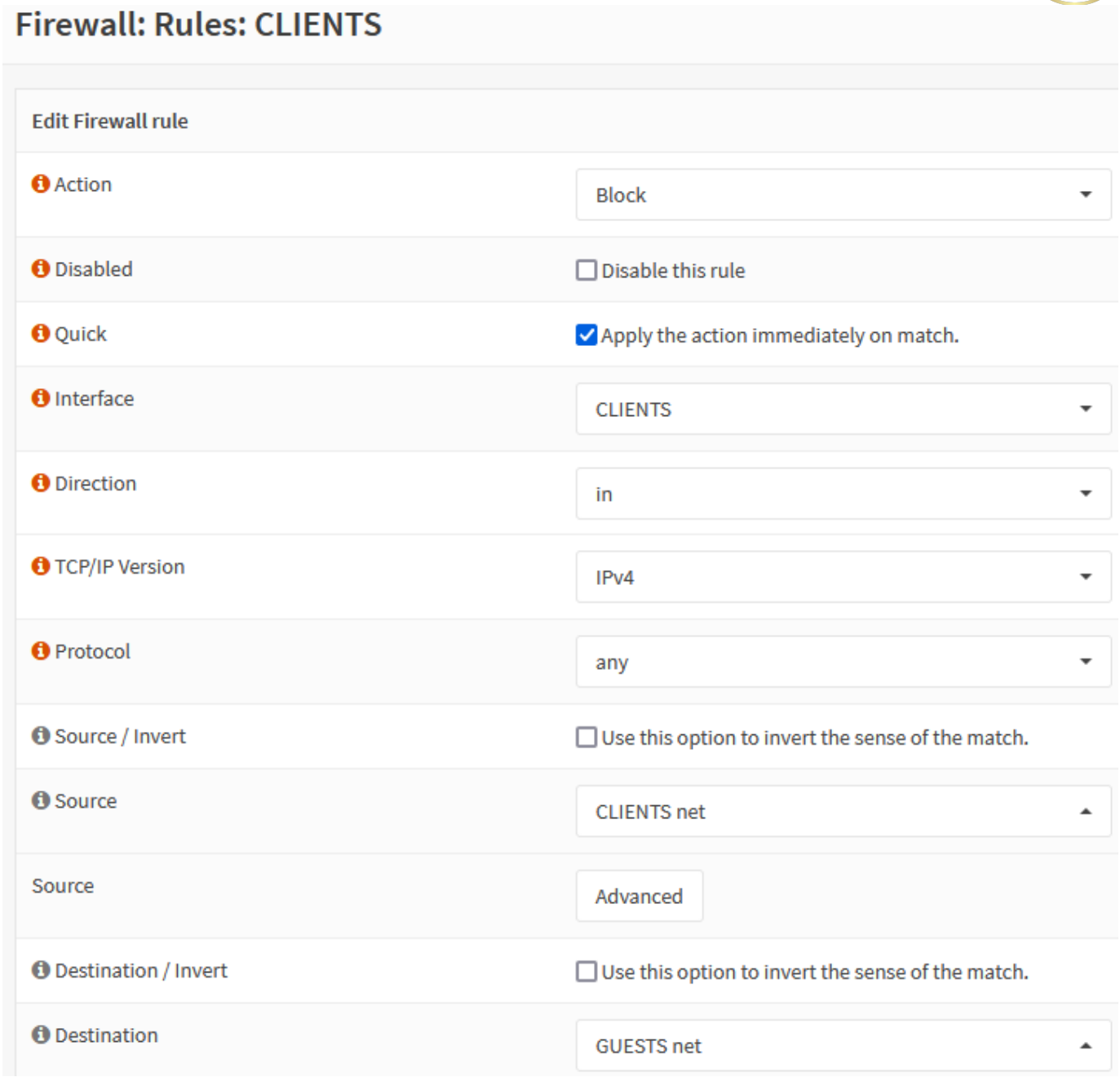


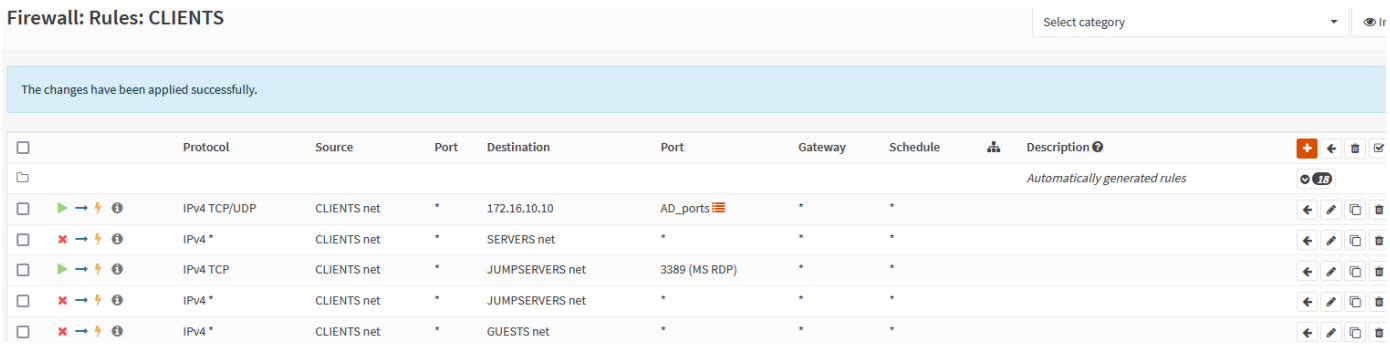
Now we need to block all other connections with **JUMPSERVERS** network. Add new rule, select **Block** for **Action**, select **CLIENTS** for **Interface**, select **CLIENTS** as **Source**, select **JUMPSERVERS** as **Destination** and leave all other options as default.



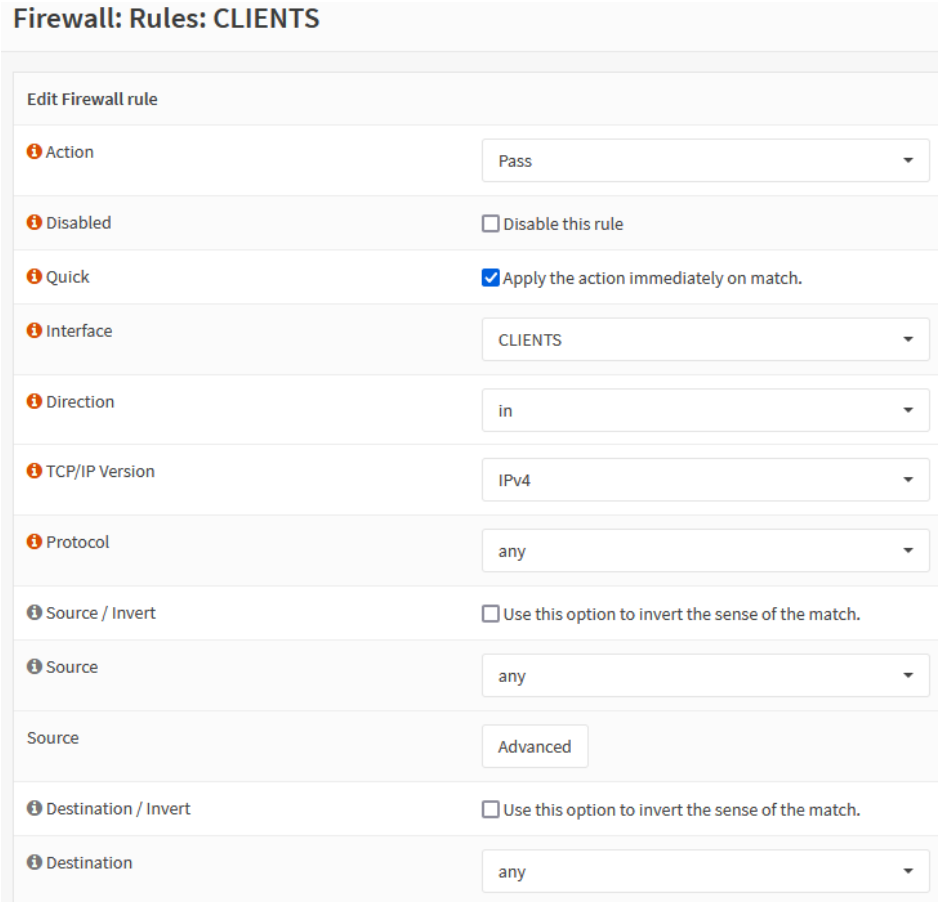


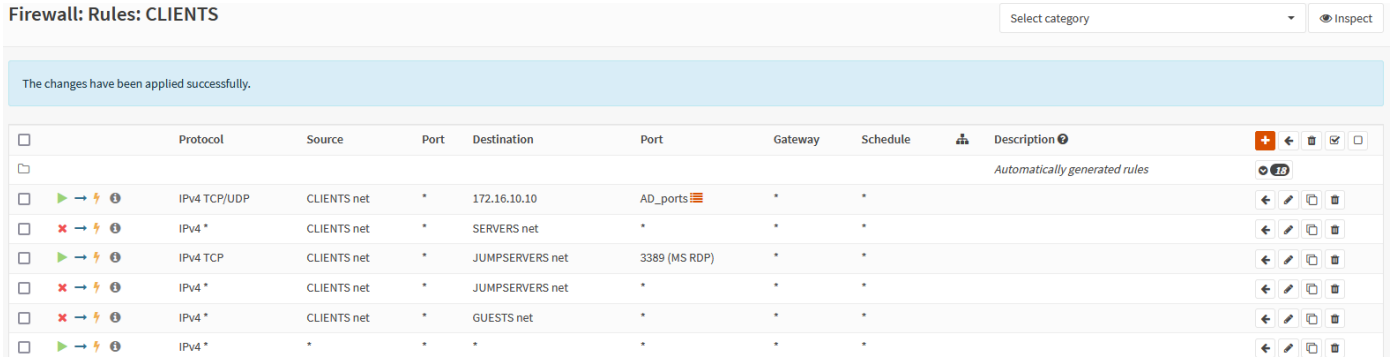
In the next rule we will block all traffic with the **GUESTS** network, add new rule, select **Block** for **Action**, select **"CLIENTS net"** as **Source**, select **"GUESTS net"** as **Destination** and leave all other options default.


Firewall: Rules: CLIENTS
Edit Firewall rule
Action
Block
Disabled
Disable this rule
Quick
Apply the action immediately on match.
Interface
CLIENTS
Direction
in
TCP/IP Version
IPv4
Protocol
any
Source / Invert
Use this option to invert the sense of the match.
Source
CLIENTS net
Source
Advanced
Destination / Invert
Use this option to invert the sense of the match.
Destination
GUESTS net


Firewall: Rules: CLIENTS
Select category
The changes have been applied successfully.
Protocol
Source
Port
Destination
Port
Gateway
Schedule
Description
Automatically generated rules
IPv4 TCP/UDP
CLIENTS net
*
172.16.10.10
AD_ports
IPv4 *
CLIENTS net
SERVERS net
IPv4 TCP
CLIENTS net
JUMPSERVERS net
3389 (MS RDP)
IPv4 *
CLIENTS net
JUMPSERVERS net
IPv4 *
CLIENTS net
GUESTS net

In the last rule, we give full access for **CLIETNS** network.



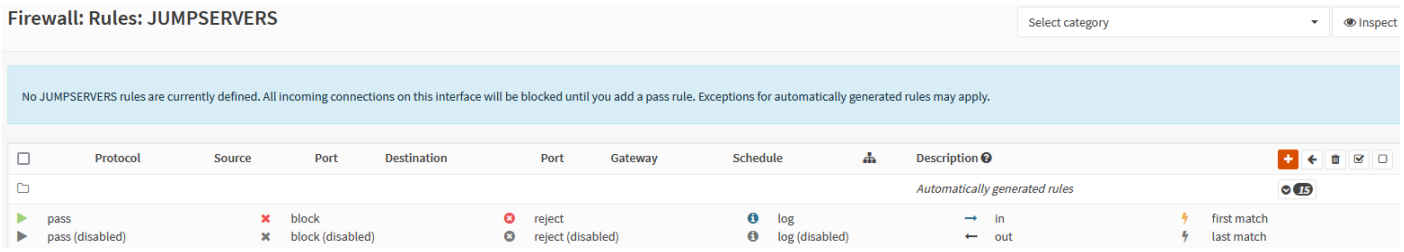


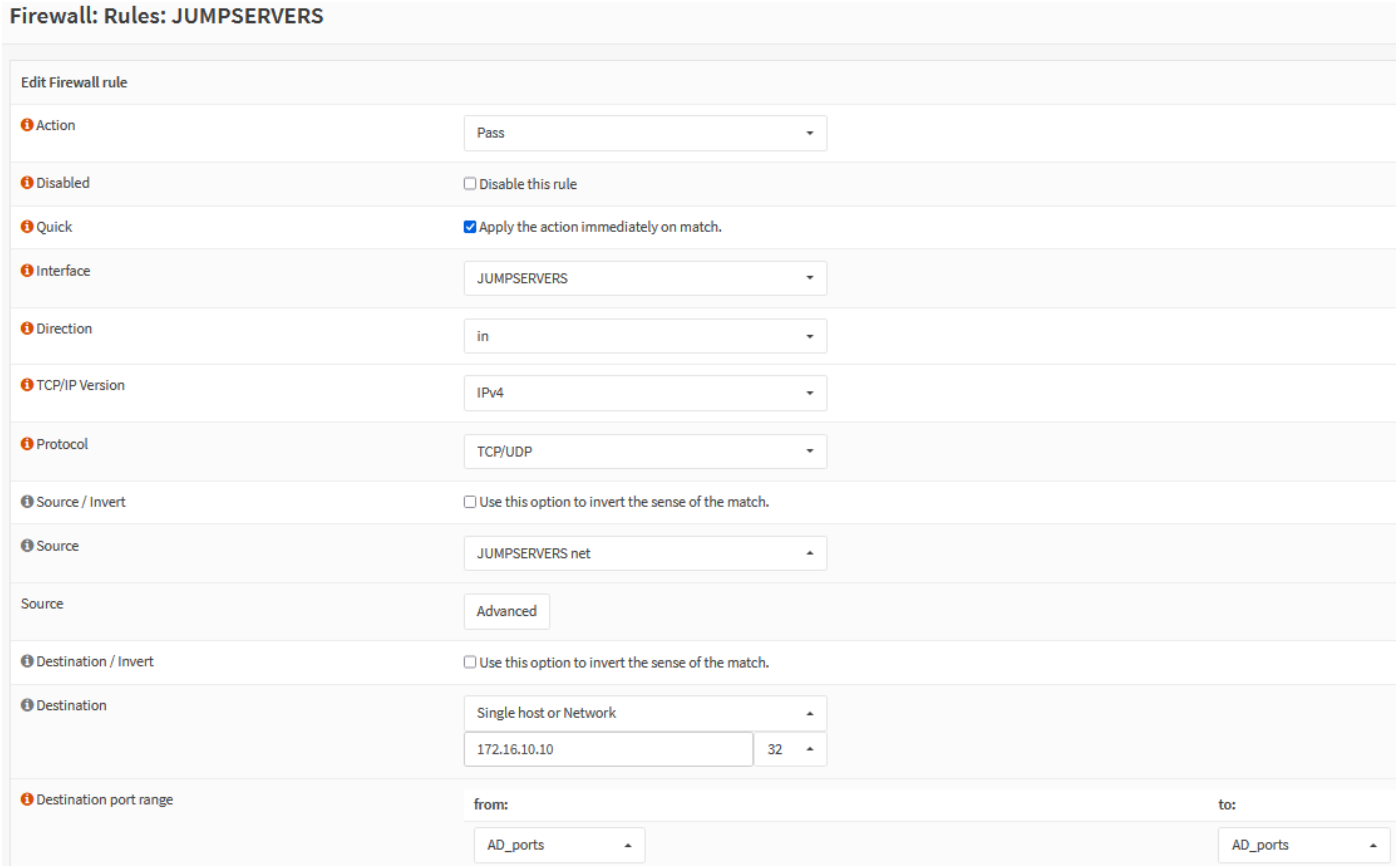
As you see we have 6 rules in **CLIENTS** interface, and they have to be arranged like shown in the figure below. In order to understand the logic of firewall rules, lets read them from left to right and from top to bottom.

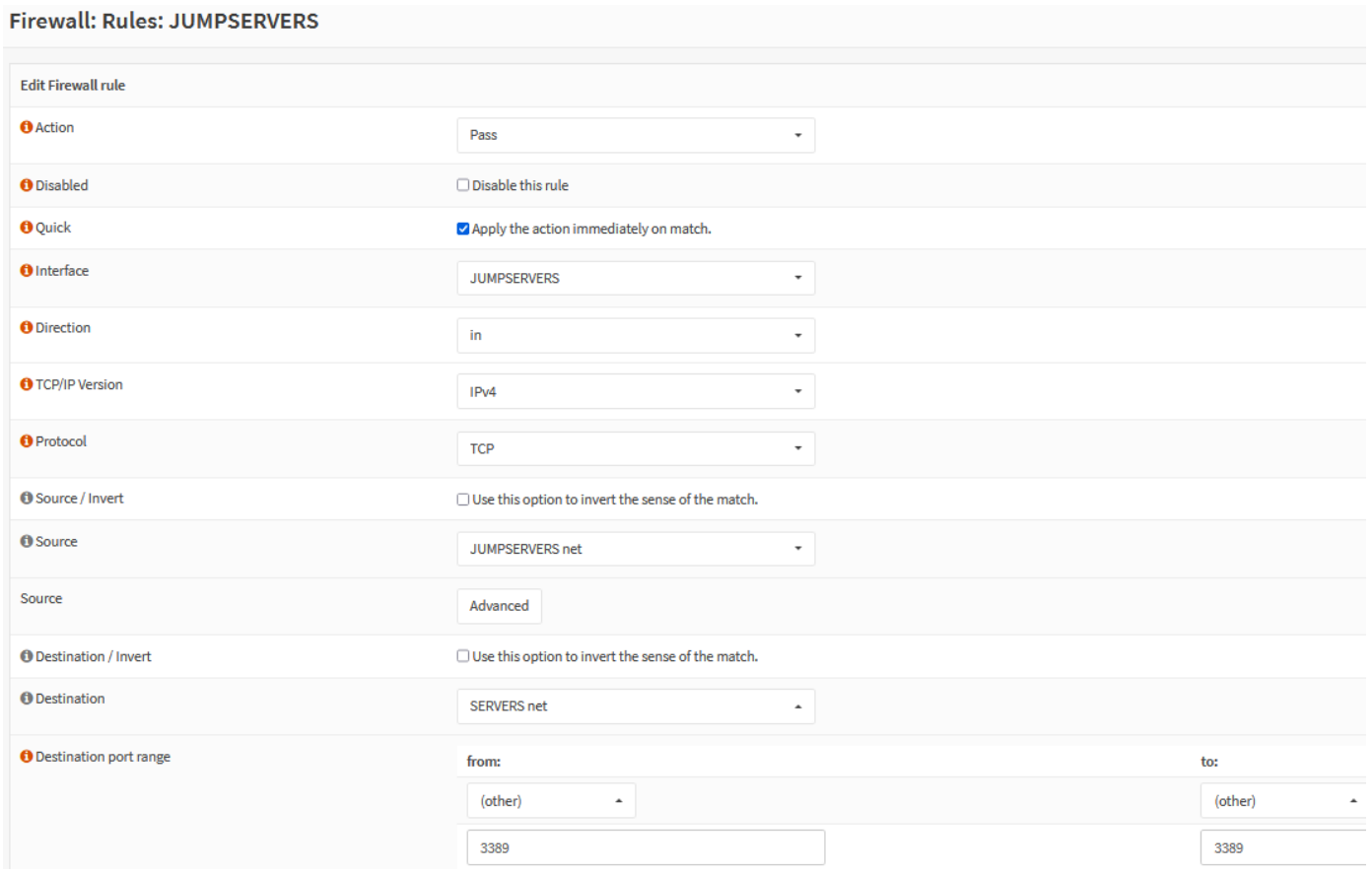
1-Allow **TCP** connection from **CLIENTS** network to **JUMPSERVERS** network's port **3389**
2-Block any connection from **CLIENTS** network to **JUMPSERVERS** network
3-Allow **TCP/UDP** connections from **CLIENTS** network to **172.16.10.10** (**DC**) host's **AD** ports
4-Block any connection from **CLIENTS** network to **SERVERS** network
5-Block any connection from **CLIENTS** network to **GUESTS** network
6-Allow any connection from **CLIENTS** network to any network

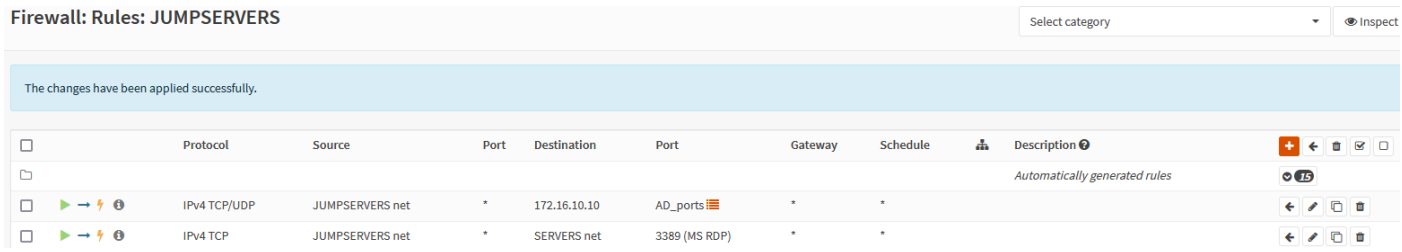
1.6.2 JUMPSERVERS rules

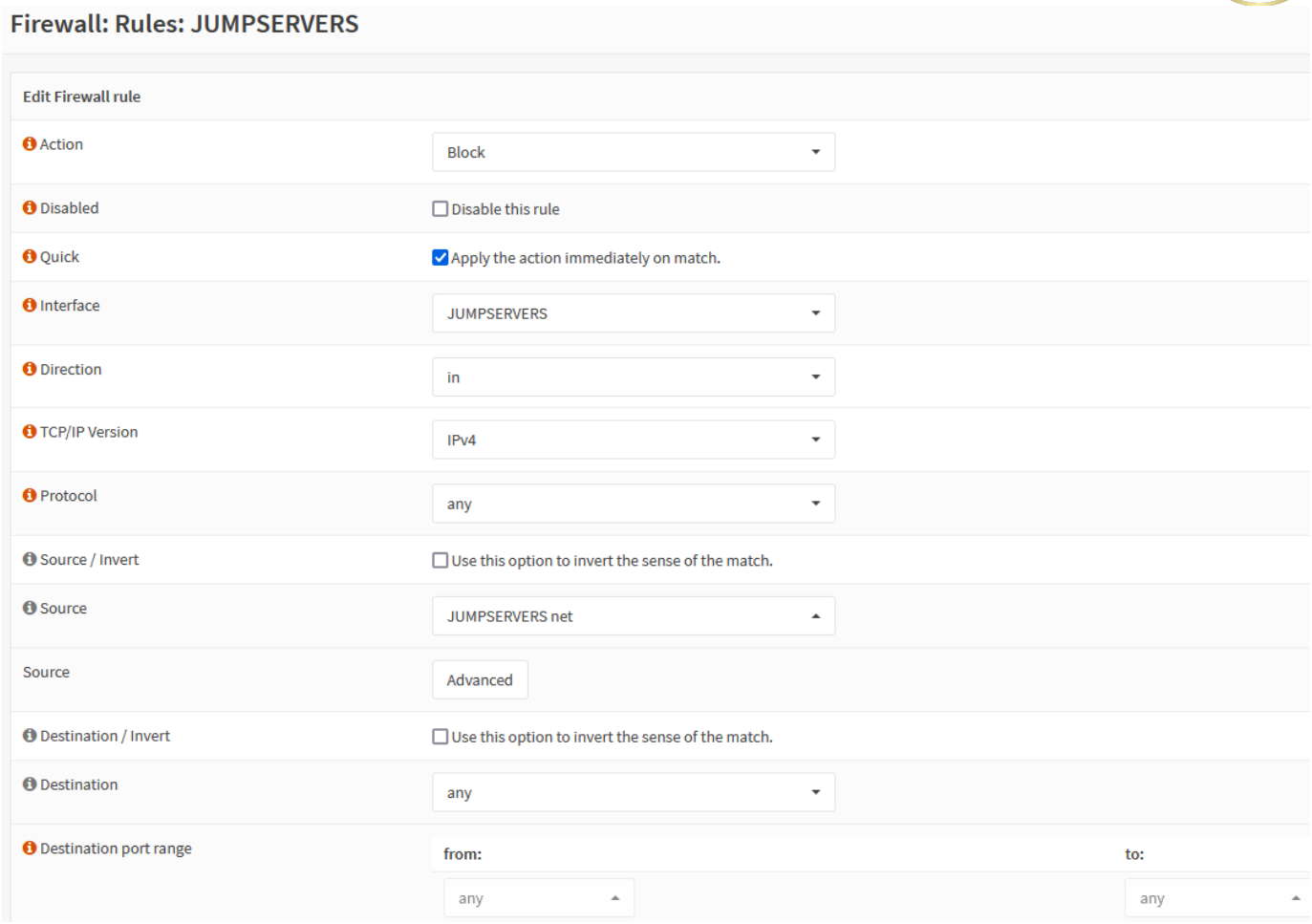
In order to keep the instruction short, I will not write all steps for **JUMPSERVERS** interface because it is similar with the **CLIENTS** rules. So it will be enough to understand its logic just by analyzing the following figures.

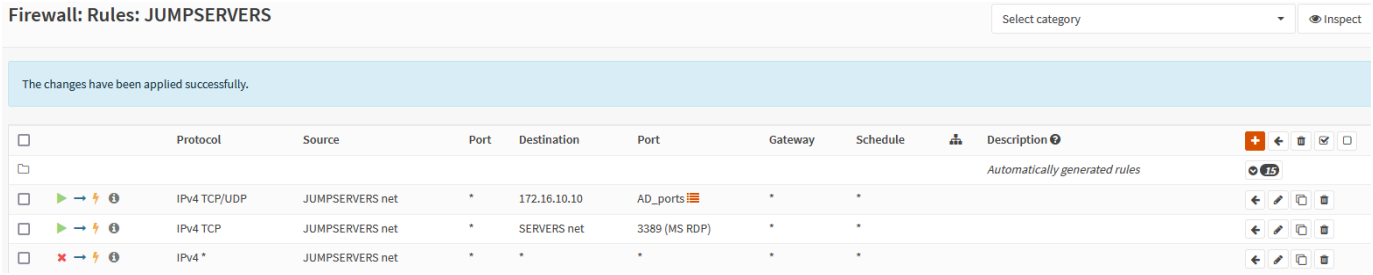





Firewall: Rules: JUMPSERVERS
Edit Firewall rule
Action
Pass
Disabled
Disable this rule
Quick
Apply the action immediately on match.
Interface
JUMPSERVERS
Direction
in
TCP/IP Version
IPv4
Protocol
TCP
Source / Invert
Use this option to invert the sense of the match.
Source
JUMPSERVERS net
Source
Advanced
Destination / Invert
Use this option to invert the sense of the match.
Destination
SERVERS net
Destination port range
from:
to:
(other)
(other)
3389
3389


Firewall: Rules: JUMPSERVERS
Select category
Inspect
The changes have been applied successfully.
Protocol
Source
Port
Destination
Port
Gateway
Schedule
Description
Automatically generated rules
IPv4 TCP/UDP
JUMPSERVERS net
*
172.16.10.10
AD_ports
*
*
IPv4 TCP
JUMPSERVERS net
*
SERVERS net
3389 (MS RDP)
*
*

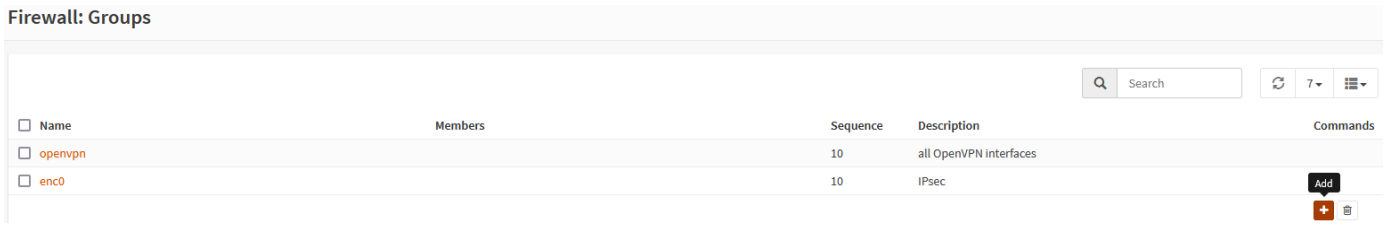




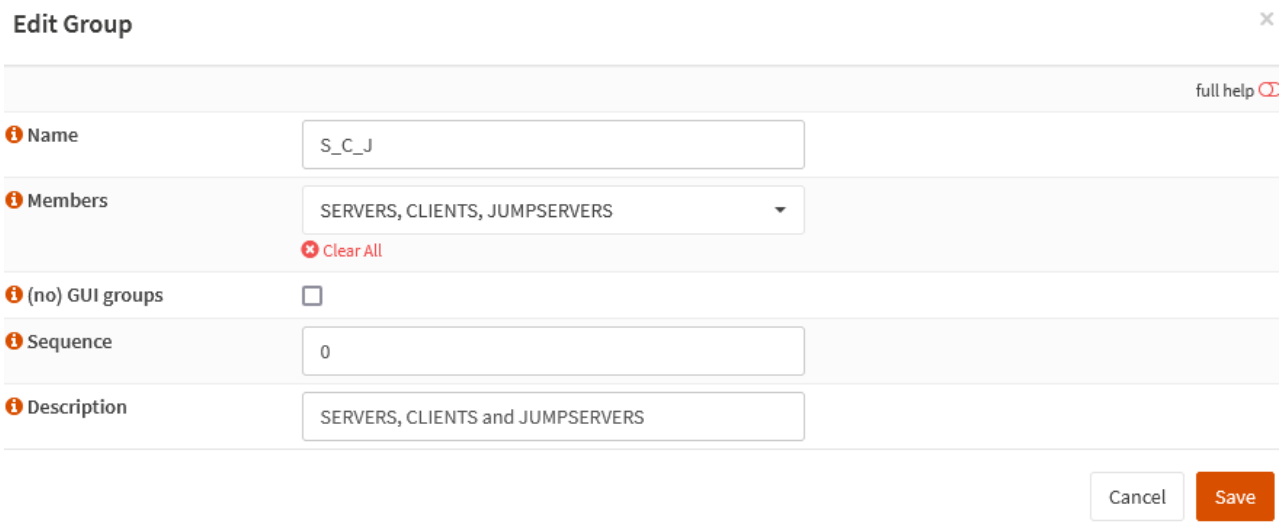
The reading of rules for **JUMPSERVERS** interface is like this,
1-Allow **TCP/UDP** connections from **JUMPSERVERS** network to 172.16.10.10 (**DC**) host's **AD** ports
2-Allow **TCP** connections from **JUMPSERVERS** network to **SERVERS** network's **RDP** ports
3-Block any connection from **JUMPSERVERS** network to any network

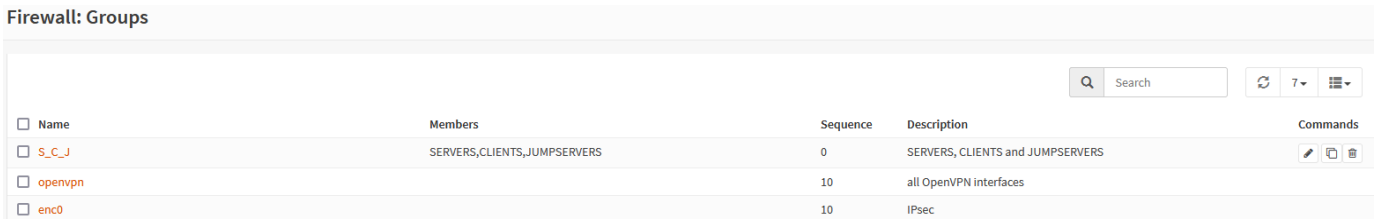
1.6.3 GUESTS rules

In our project we consider **GUESTS** network for unauthorized persons, so there is no need **GUESTS** to communicate with **SERVERS**, **JUMPSERVERS** and **CLIENTS** networks, it will be enough for them being able to connect to the **Internet** under our control. In order to realize it, first we have to write rules for each of this interfaces to block network traffic outgoing from **GUESTS** network, but instead of writing same rules separately for each interface, we can gather all of these 3 interfaces under one group and write just one rule for this group. Rules are read from left to right and applied from top to bottom. Group rules' priorities are higher than individual rules always. Lets create a group firstly.
Go to the **Firewall => Groups** and click the **Add** button to create a group.

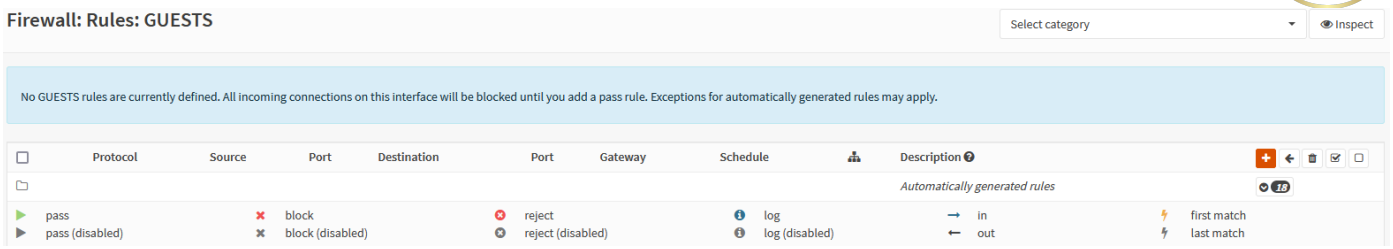


In the **"Edit Group"** window, give a name for the group and select the interfaces.

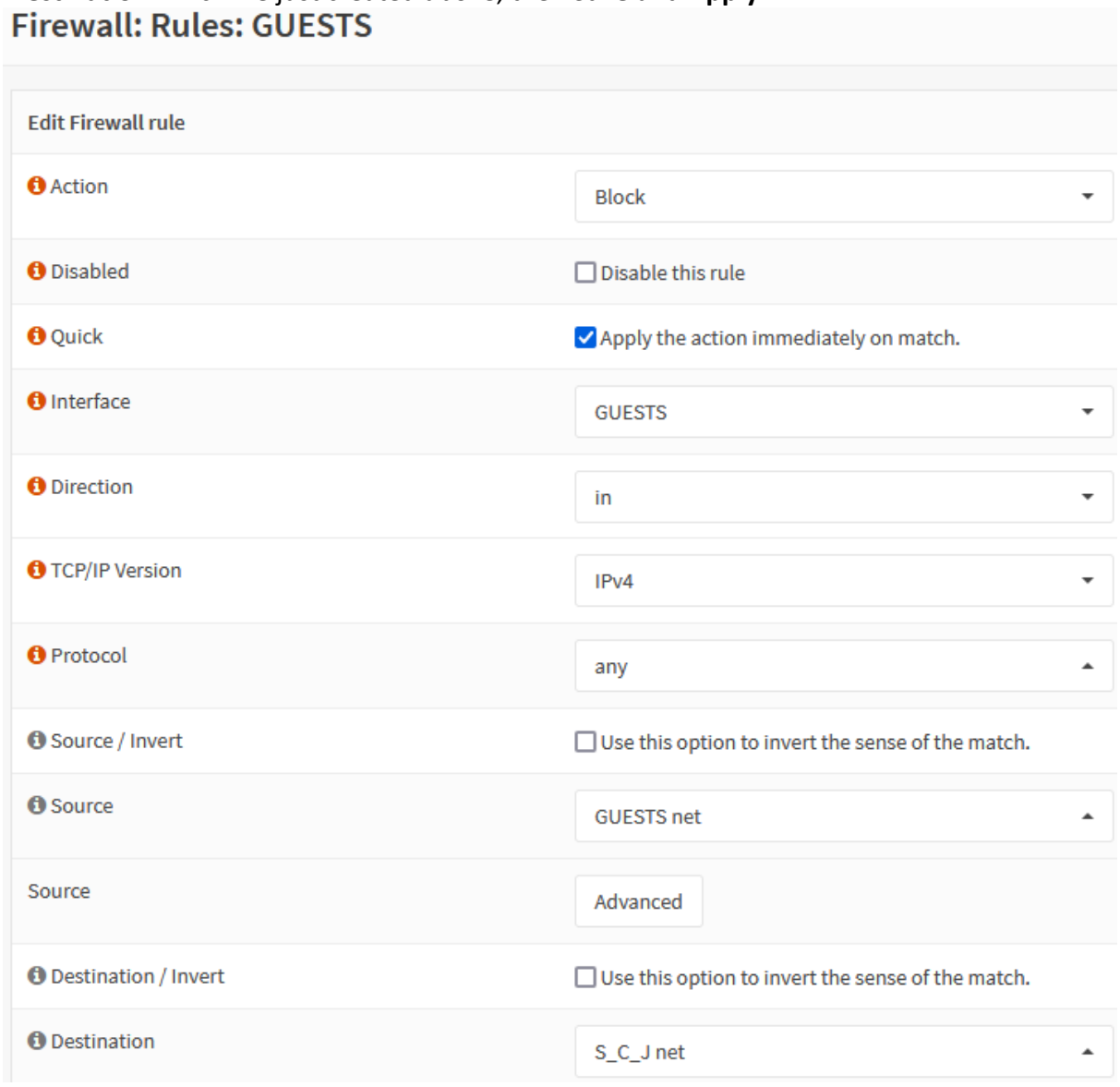


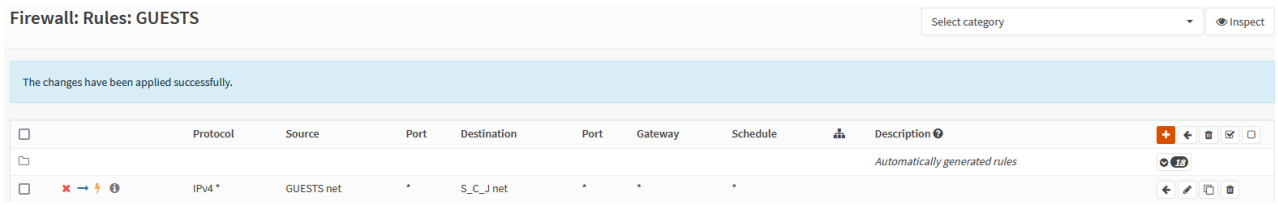


After creating a group, go to the **Firewall => Rules => GUESTS** and click **Add** button.

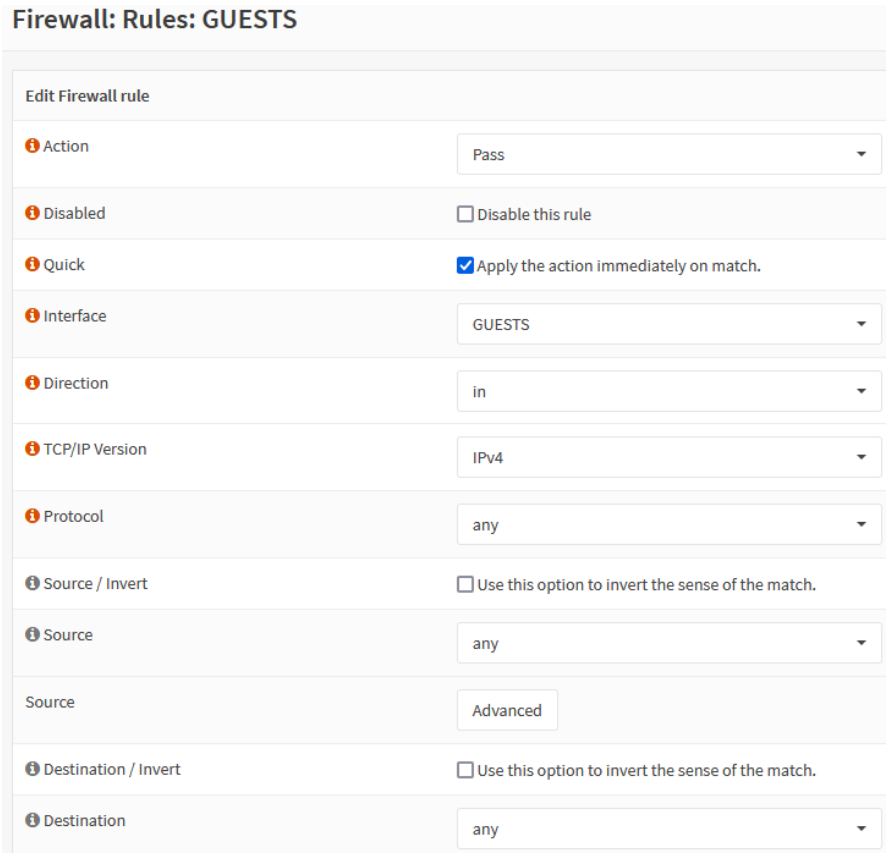


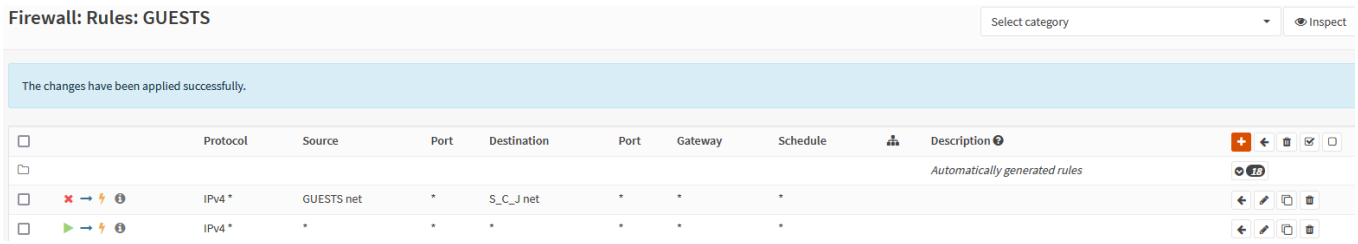
Inside the rule, select **Block** for **Action**, select **"GUESTS net"** as **Source** and select the group name**, "S_C_J net"** as **Destination** which we just created above, then **Save** and **Apply**.





After blocking outgoing traffic to **SERVERS**, **CLIENTS** and **JUMPSERVERS** networks, we need to write new rule to allow **GUESTS** interface to connect **Internet**.

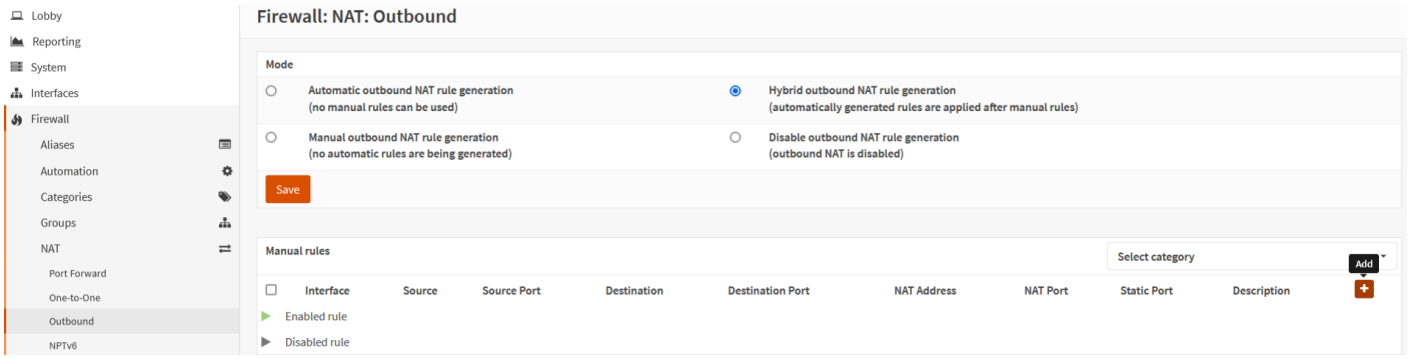




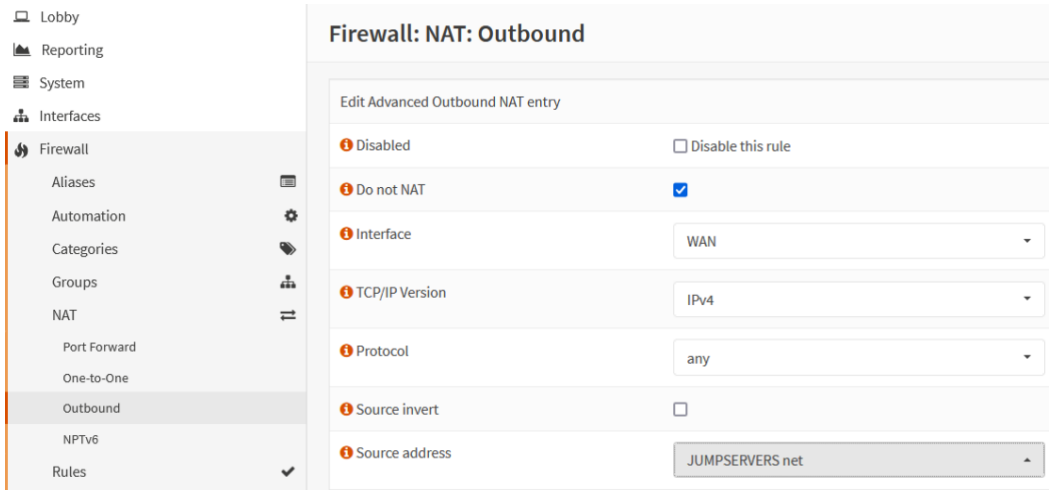
The reading of rules for **GUESTS** is like this,
1-Block any connection from **GUESTS** network to **S_C_J** group interfaces
2-Allow any connection from **GUESTS** network to any network

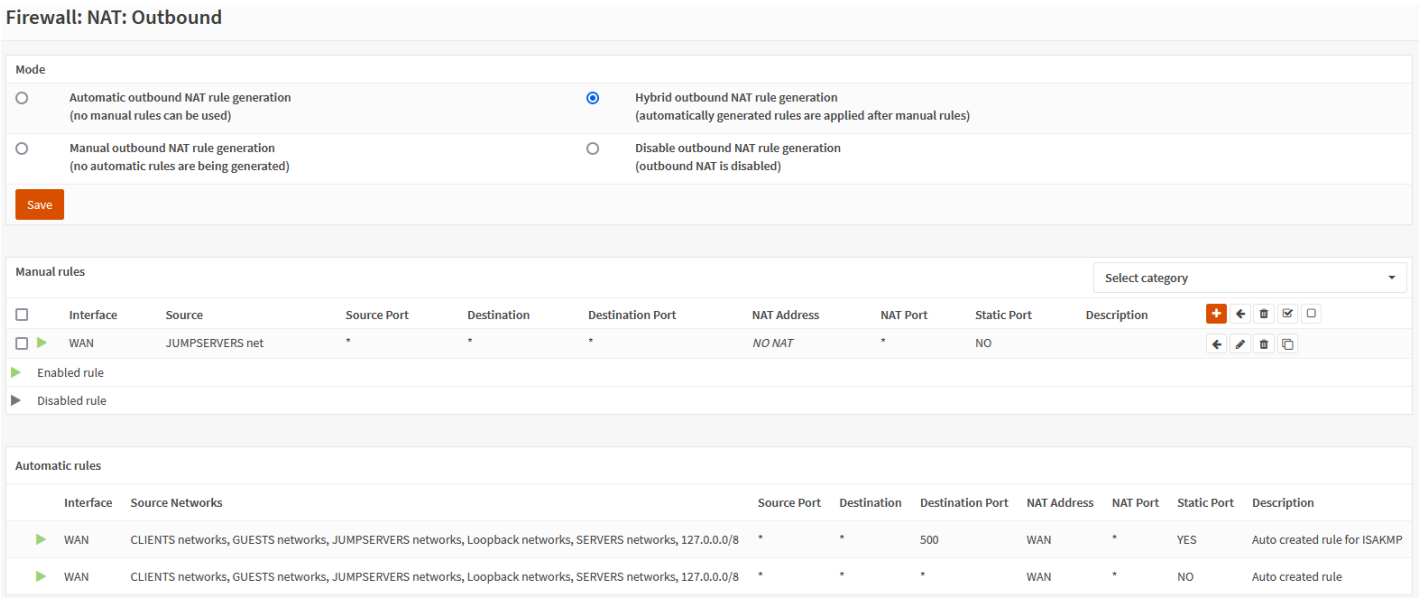
1.7 NAT configuration

In this capture we are going to block Internet access for **JUMSERVERS** interface and allow all other interfaces. Default, firewall comes in **Automatic** mode which allow all interfaces to access to the internet over the **WAN** interface, but we have to choose **Hybrid** mode to control permissions manually ourselves. This is similar with firewall rules, and the logic behind the process is same.
Go to the **Firewall => NAT => Outbound** section and hit the add button to open new rule sheet.



In opened window, put tic in the **"Do not NAT"** box to disable **NAT** function and in the **"Source address"** choose **JUMPSERVERS** network. Without **NAT**, it is not possible to communicate with other devices on Internet because **NAT** allows us to map our **private IP** addresses with **public IP** addresses.



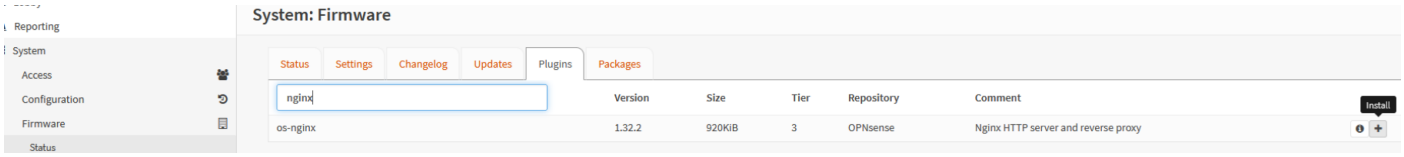


OPNsense plugins

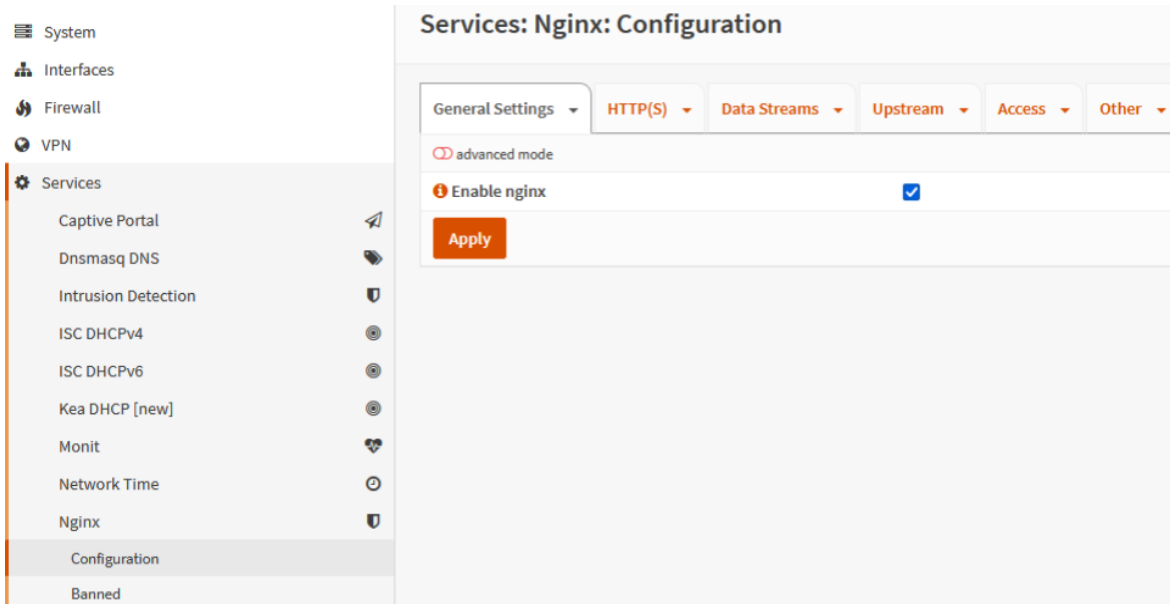
2.1 Nginx

Interacting directly with the web server itself can be dangerous for the security reason, in order to prevent this and add an additional security layer mostly **RP** (Reverse Proxy) logic is used in practice. **RP** server stands between web servers and clients and forwards requests from clients to web servers. Another important security feature is **WAF** (Web Application Firewall) for web servers, which allow us to filter the **HTTP** requests between web applications and the **Internet**. The **OPNsense Nginx** plugin combines these two features in it.

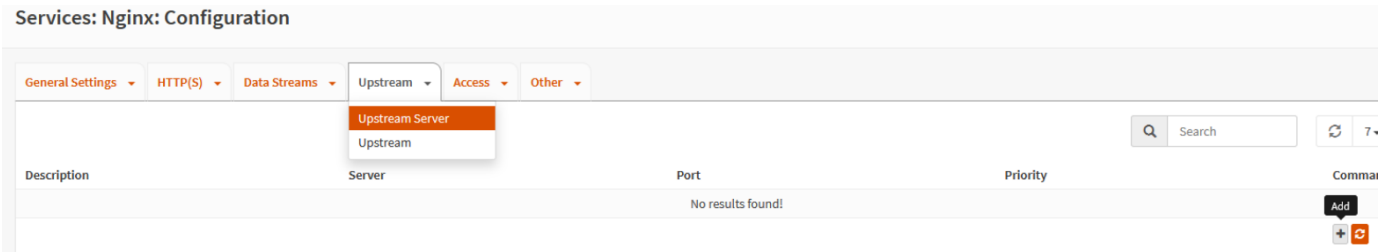
First we need to download **Nginx** plugin, go to the **System => Firmware => Plugins**, type **nginx** and search for the plugin. Once you confirm that it is right plugin, hit install button as displayed in the figure.



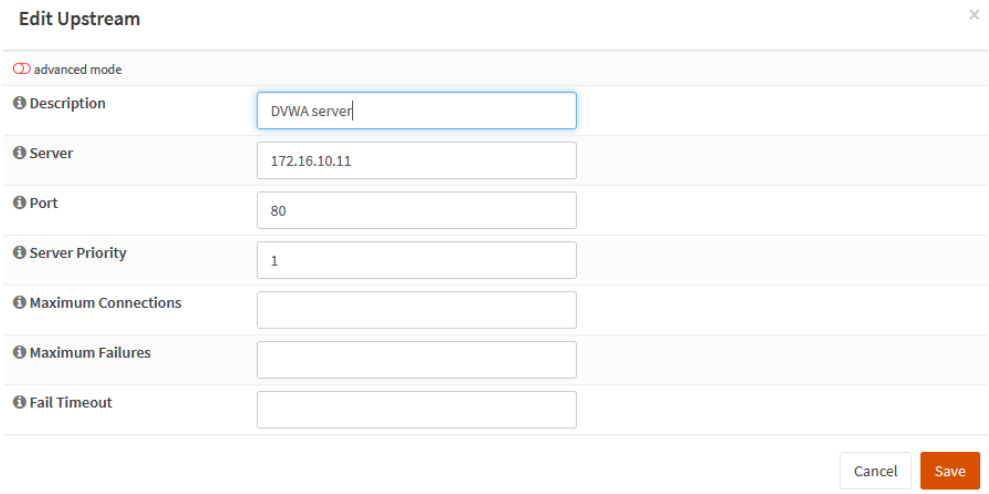
After completing installation go to the **Services => Nginx => Configuration** and enable the service.



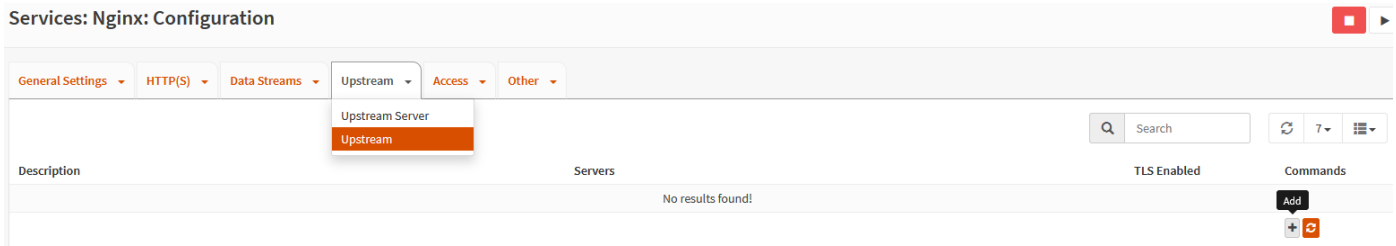
In this section choose the **"Upstream Server"** and hit the add button.



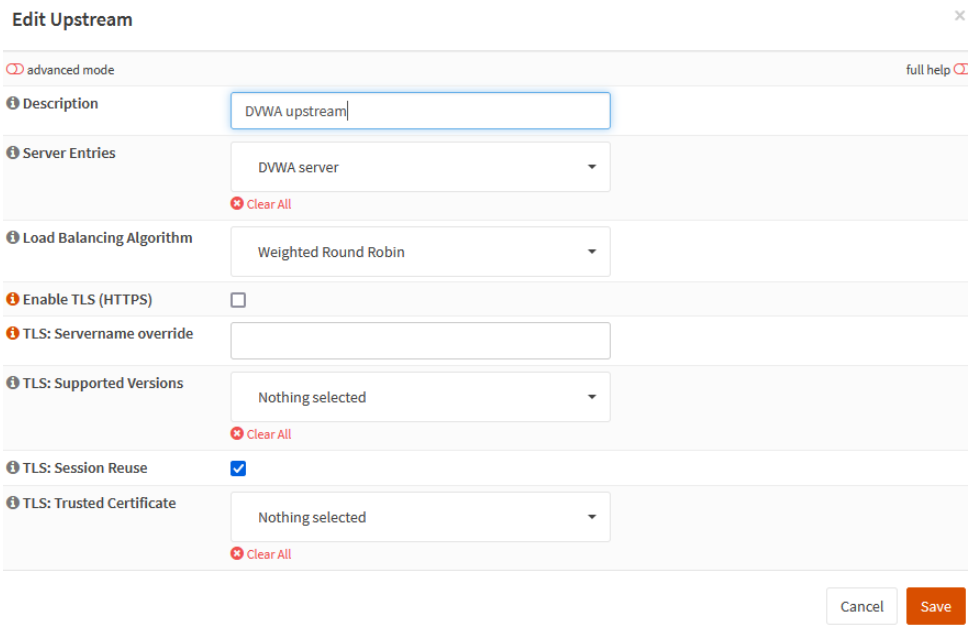
In the **"Edit Upstream"** window give a descriptive name to the server in the **Description** section, assign your web server's **IPv4** address (172.16.10.11), **Port** number (our web server runs on port **80**) and the **"Server Priority"** value (if there are a few running servers we can define their priorities manually) which is **1** in this lab.



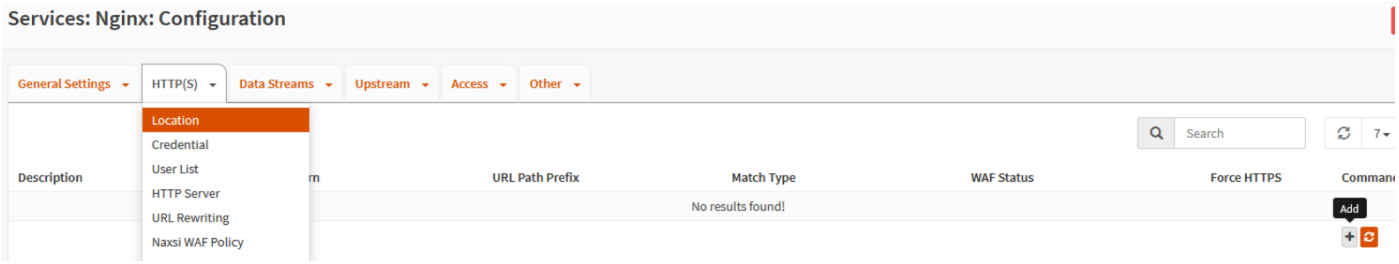
Later go to the **Upstream** section and add configuration as shown in the figure.



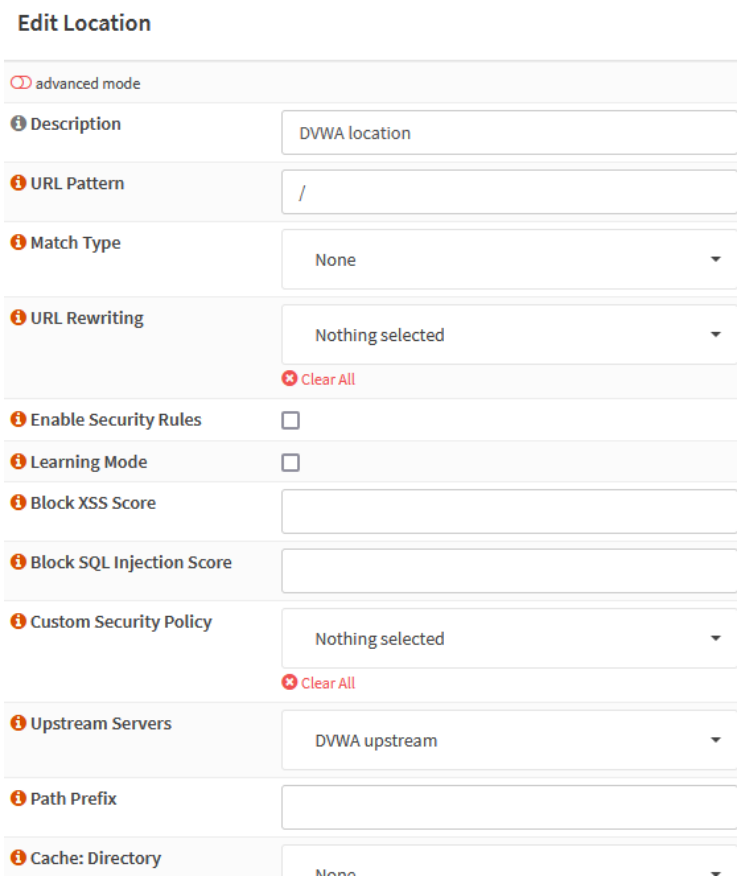
In popped window type new descriptive name for **Upstream** and choose the server we created just above in **"Server Entries"**.



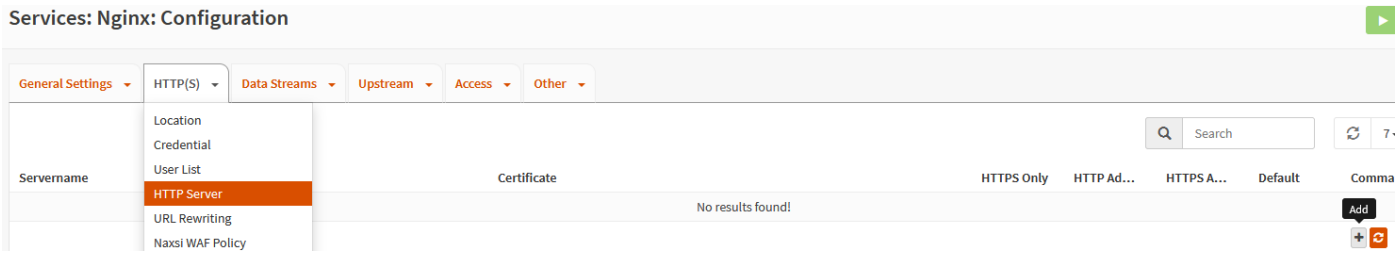
In the next step we have to define the location for our **Upstream Server**.



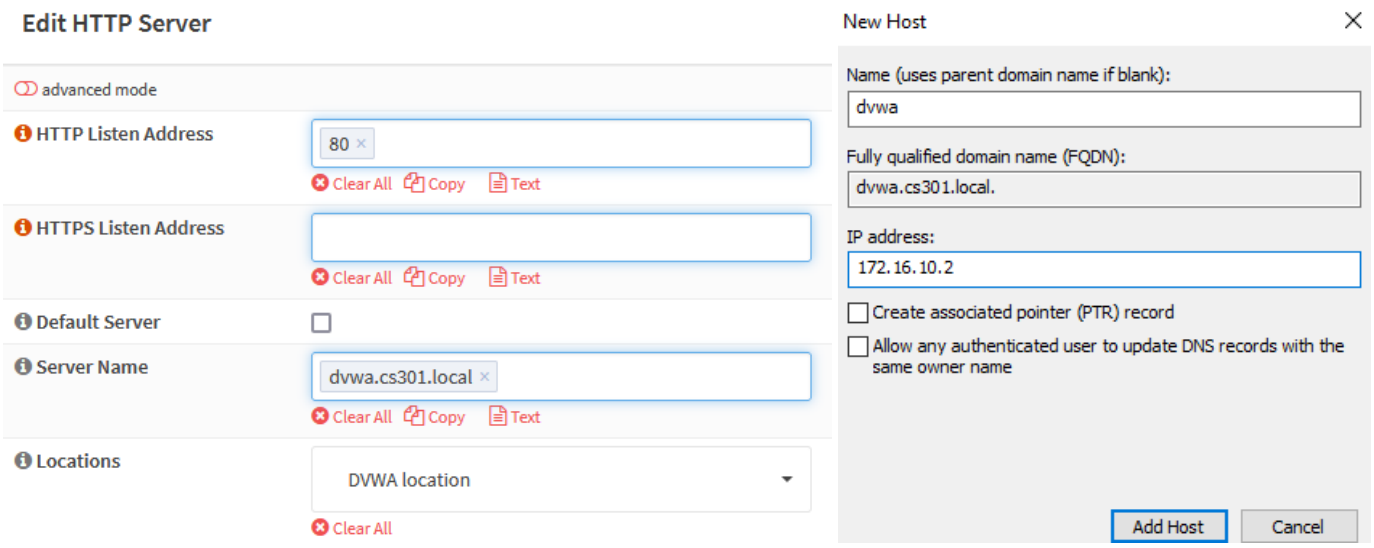
In the opened window add **Description** for our location, in the **"URL Patters"** section add the "**/**", and last in the **"Upstream Servers"** section choose the **Upstream** which we created above.



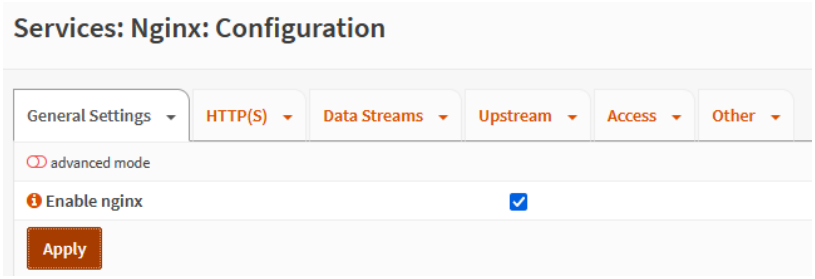
Back to the configuration page and add new configuration in **HTTP(S) => HTTP Server**.



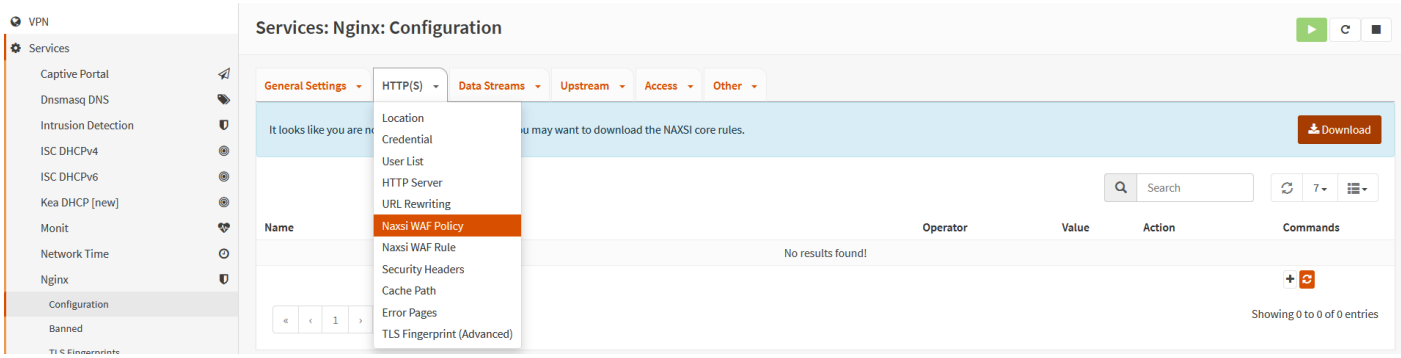
In the **"HTTP Listen Address"** type the web application port number (it is **80** in our lab) and in the **"Server Name"** section type your domain name for proxy (we have to add new **DNS** record for this domain name in **DC** server).



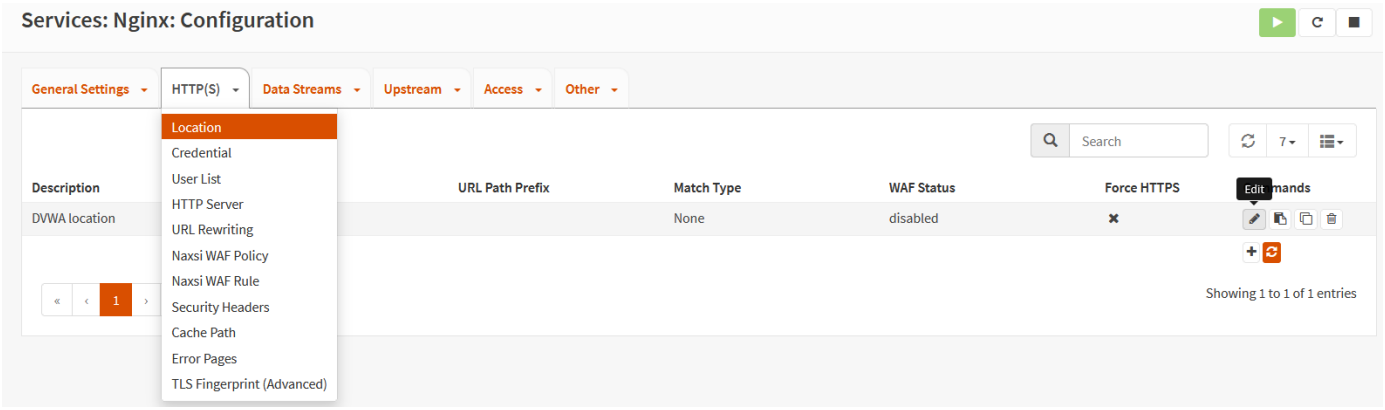
Later back to the **"General Settings"** in configuration and press **Apply** button to save configuration changes.



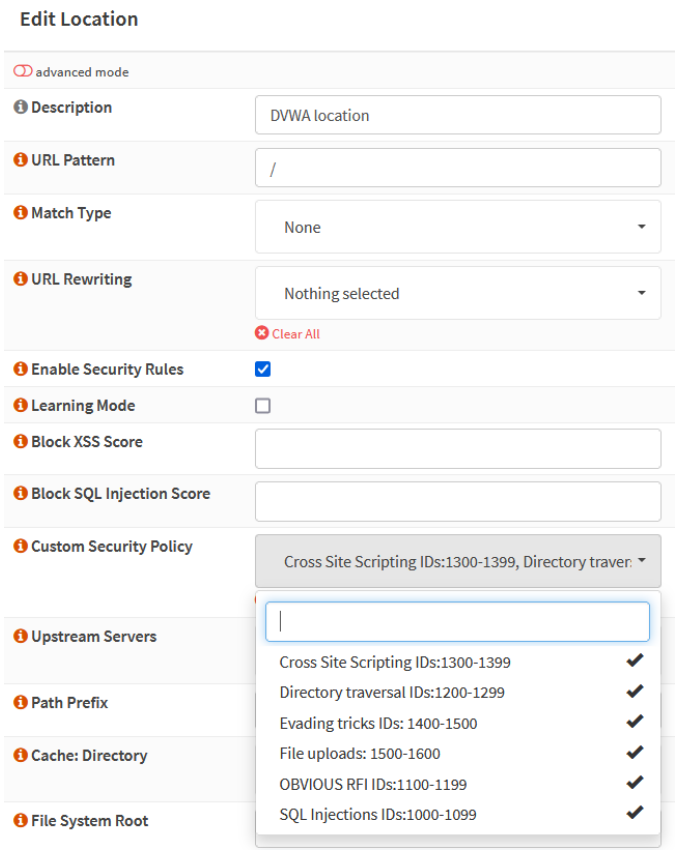
Now we are ready for **WAF** configuration, firstly go to the **HTTP(S) => Naxsi WAF Policy** and press **Download** button to download policies for **WAF**.



When downloading is finished go back to the **HTTP(S) => Location** and edit the configuration we created above.

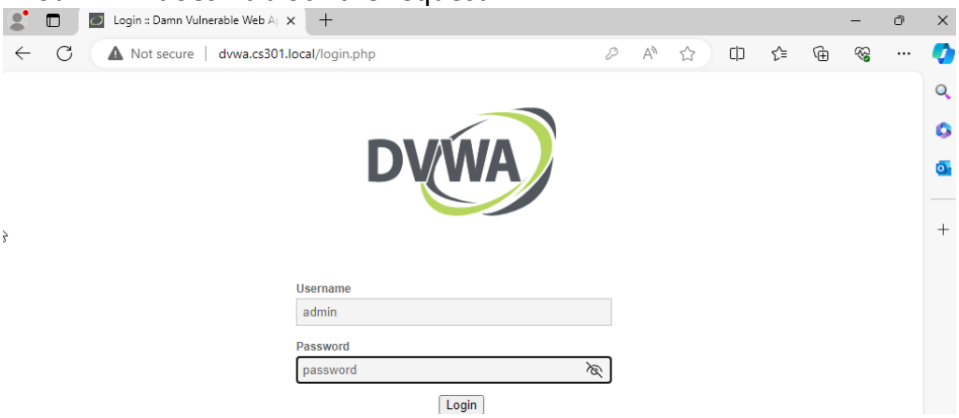


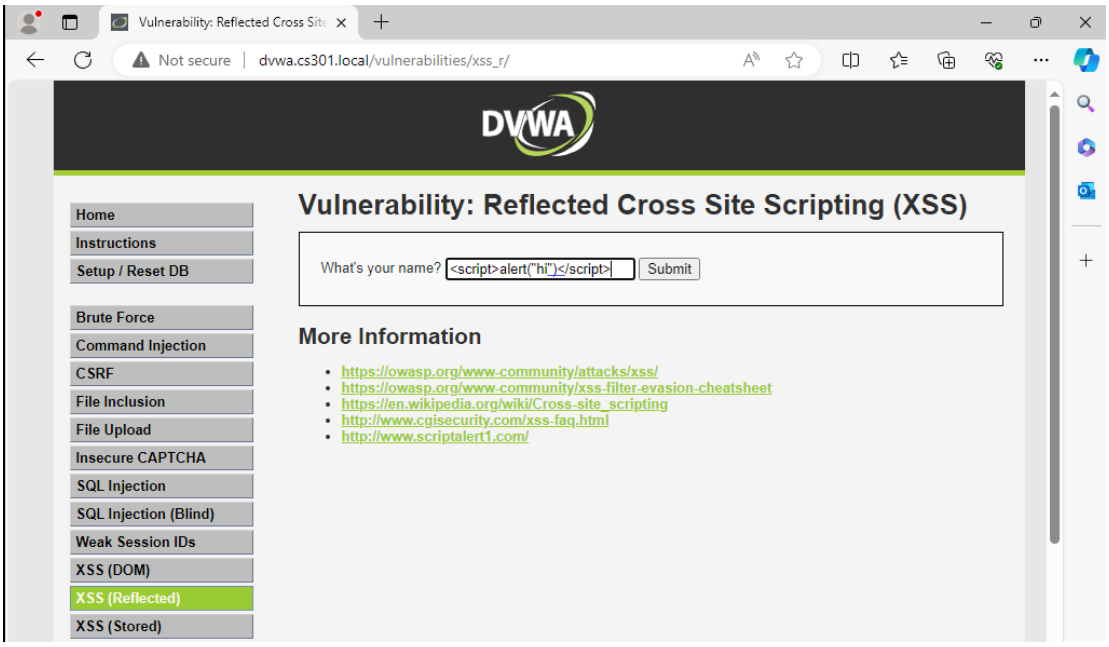
In the **"Costume Security Policy"** section you will see the all downloaded policies for **WAF**, choose whichever you want to apply your **WAF**.



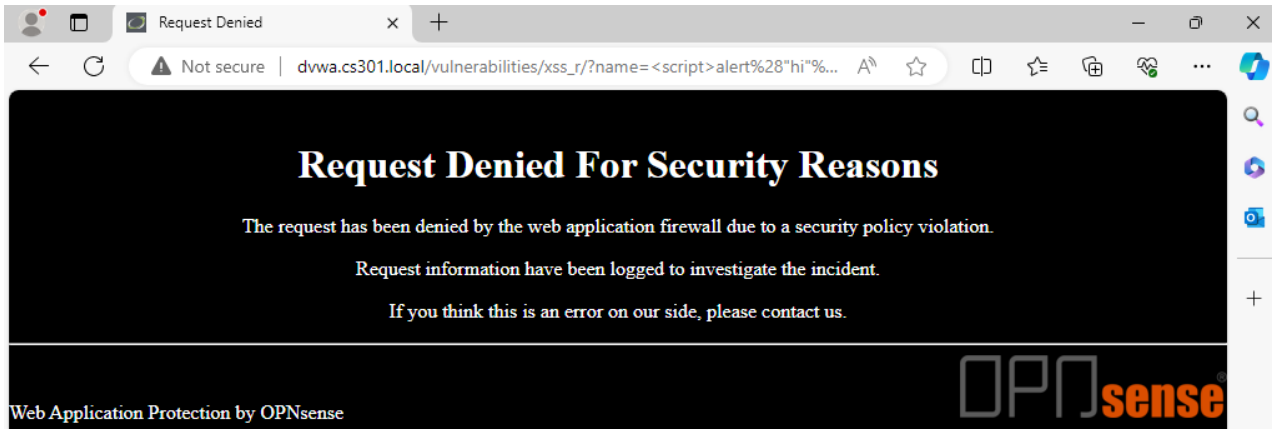
We successfully installed **Nginx** plugin for **OPNsense** firewall, now we can test it. As you remember we installed **DVWA** (Damn Vulnerable Web Application) in our **Kali Linux** machine and add this machines **IP** address in our **Nginx** configuration (in **HTTP(S) => HTTP server => Server** section) for redirecting requests and generated a **DNS** record for our proxy. Open your browser, go to the http://dvwa.cs301.local address and login with default credentials (username: **admin** | password: **password**) for **DVWA**. After login to the application select any vulnerable section, for better understanding the **WAF** logic I choose **XSS (Reflected)**

section and typed the piece of script, '**<script>alert("hi")</script>**', and submit, which should pop up a window says "**hi**" if our **WAF** doesn't block the request.



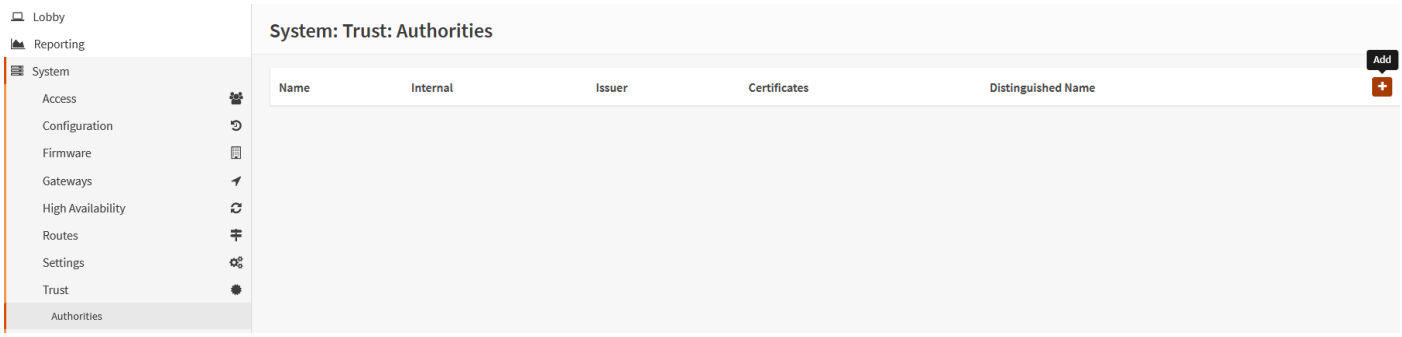


But when I pressed the submit button firewall blocked my request and displayed the default error page. This means that our firewall works properly and we can modify it by adding new strict policies to make it much more secure.

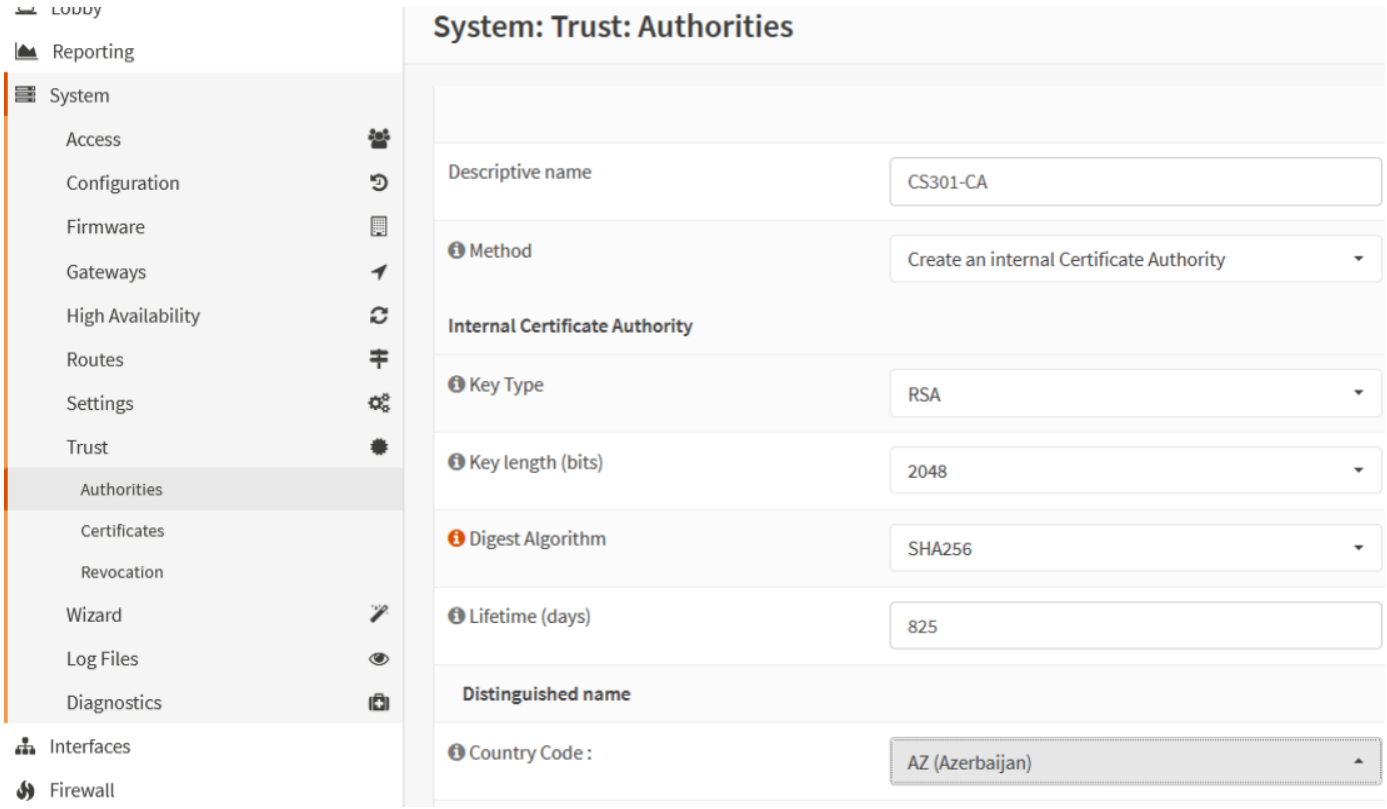


2.2 Squid Web Proxy plugin

The **Squid** is a forward caching (transparent) proxy plugin available in **OPNsense** which allows us to control web requests by apply whitelist-blacklist logic. We can block unwanted web traffic with the help of this web filtering feature.
Before setup the **Squid** plugin, we have to create a **self-signed certificate** for allowing **HTTPS** traffic to flow over our proxy service. To generate a new certificate, go to the **System => Trust => Authorities** and hit the add button on the right corner of the page.

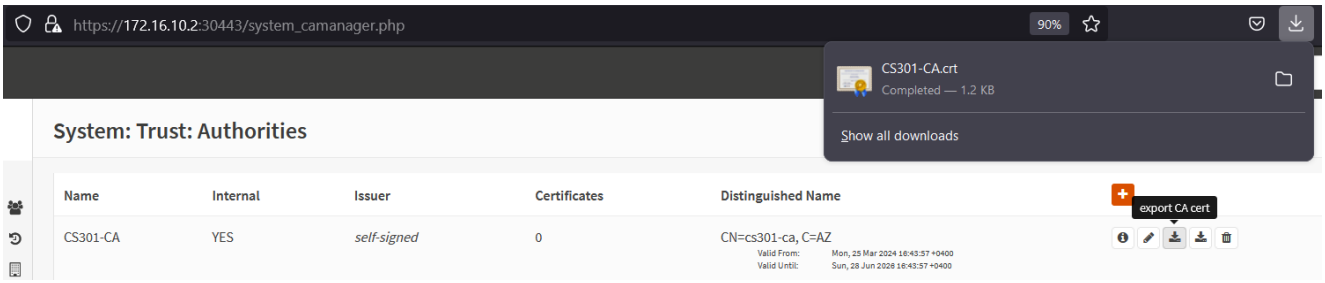


In opened window, give a **"Descriptive name"** for your certificate, choose the **Method** as **"Create an internal Certificate Authority"** and define the **"Country Code"** where you are located, leave other parameters default and **Save**.

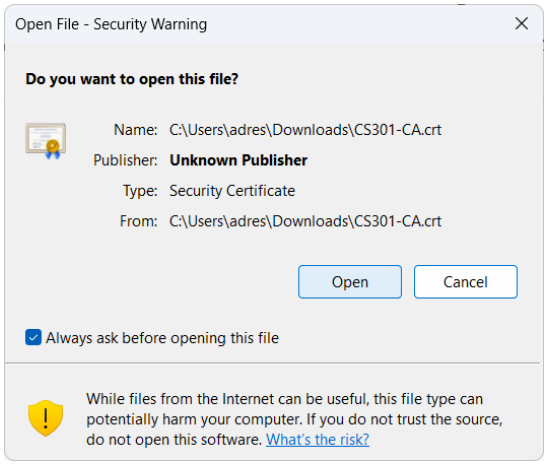


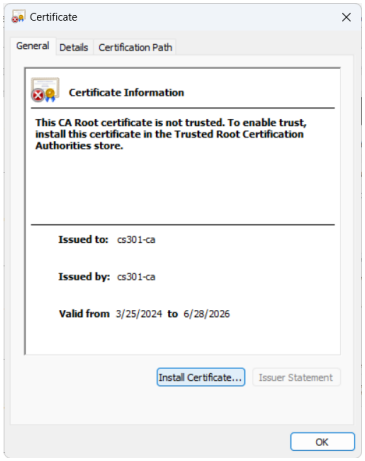
Now download the certificate into your machine as shown in the figure below and copy it to your machine

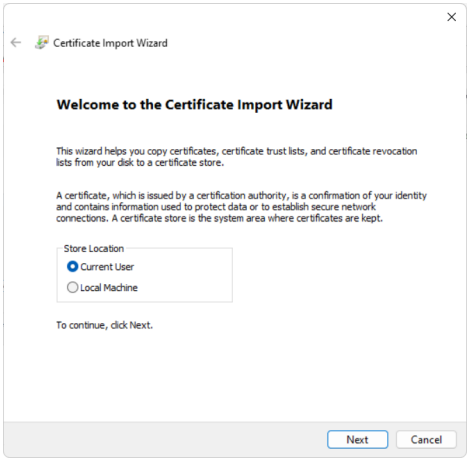
whichever you want to connect to the **Internet** over the **Squid Web Proxy**. (I copied it to my **Windows 10** client machine in **VMware**, because I applied web filtering only to my **CLIENTS** interface)

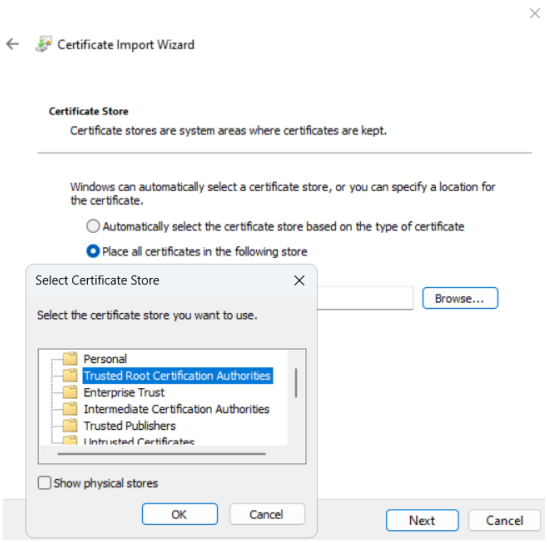


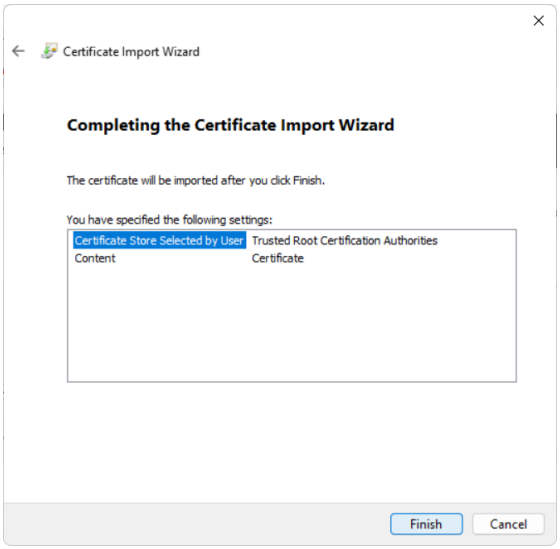
Double click on the certificate and install it to your machine where you want to use web filtering feature. Below, the figures demonstrate whole installation process step by step, just follow them if you are not familiar with certificates.







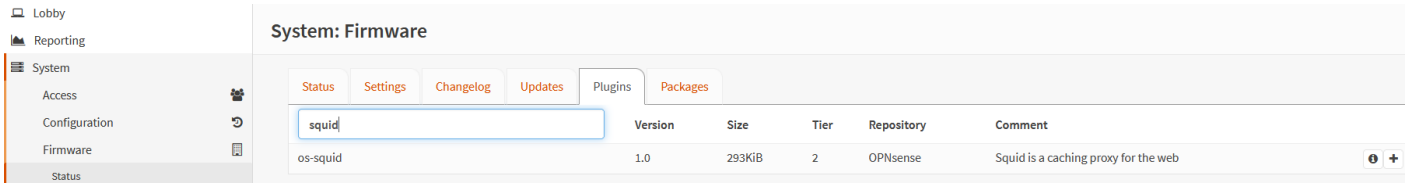




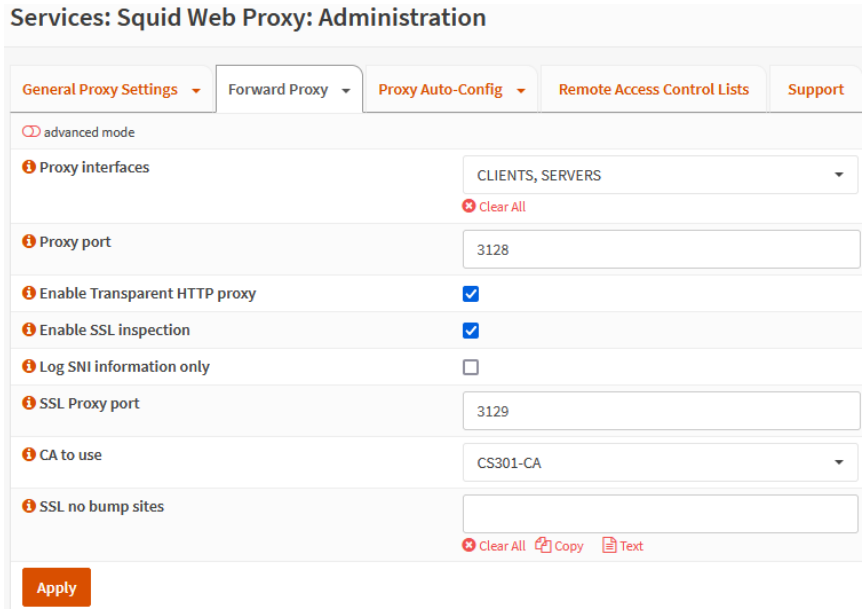
NOTE: we couldn't use the **HTTPS** websites without this certificate if we want to apply web filtering feature to our machine, because in following steps we will block default port (**443**) for **HTTPS** traffic in order to prevent proxy bypassing, so don't forget to install.

Now we are ready to install the Squid proxy, as other plugins, also the Squid have to be installed from

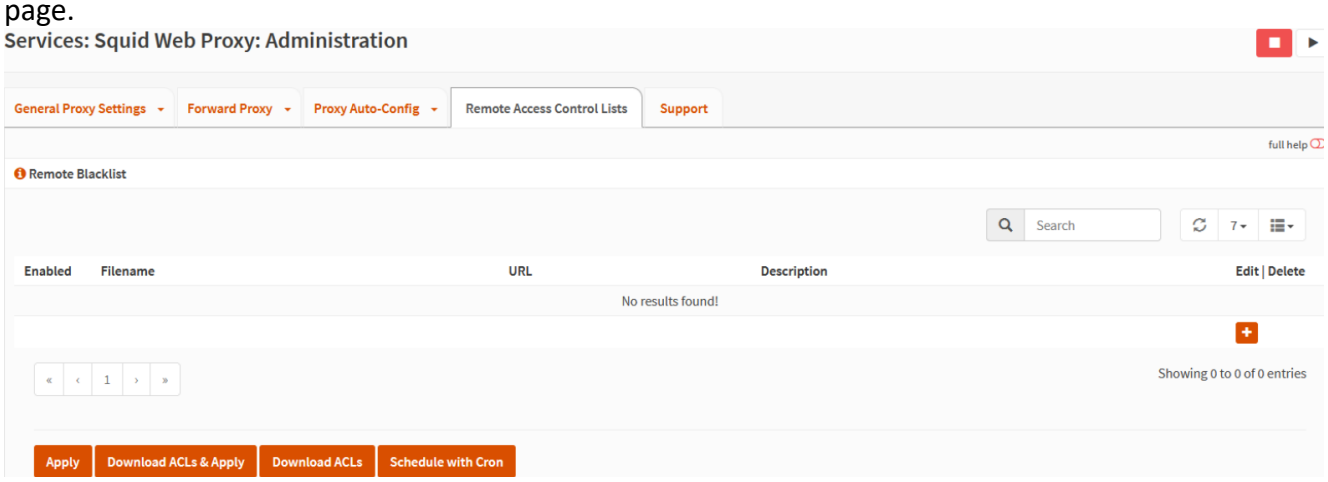
System => Firmware => Plugins section.



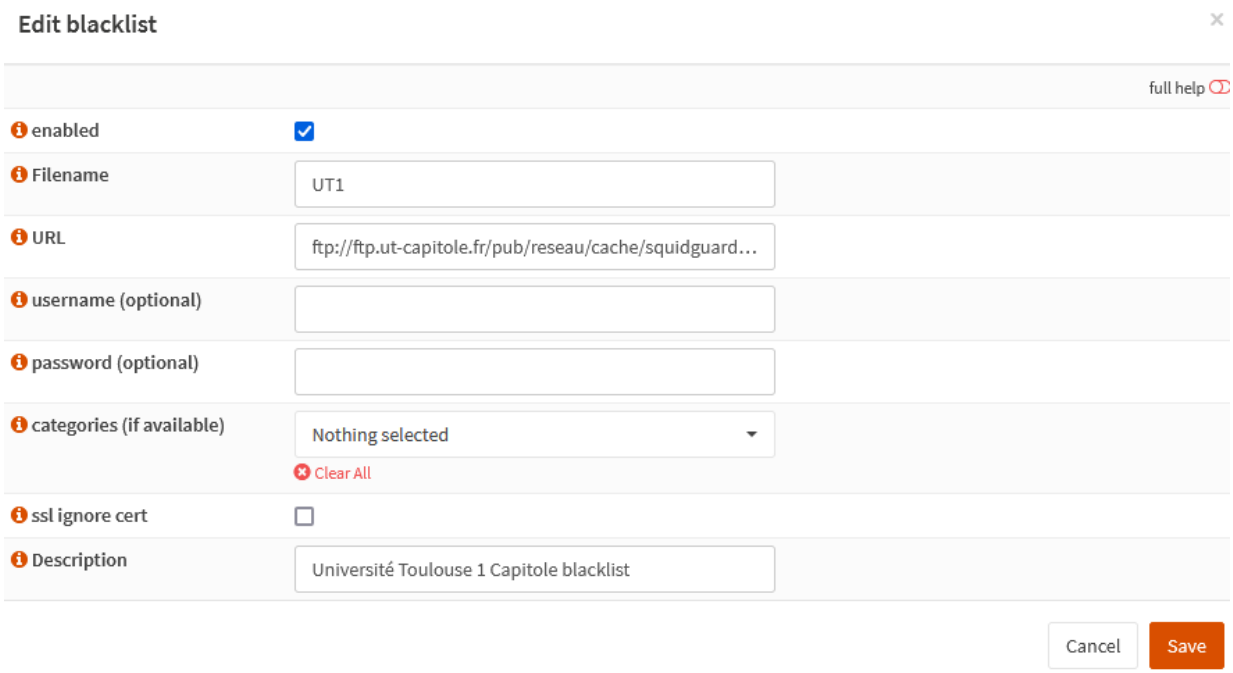
After installing the **Squid** plugin, we can find it in the **Services** section. Here enter to the **Administration** settings of **Squid** service and fill the required boxes as displayed in the figure below.
Firstly choose the interface you want to apply web filtering (as I choose **CLIENTS** interface), put tics in **"Enable Transparent HTTP proxy"** and **"Enable SSL inspection"**, and choose the certificate we created above for **"CA to use"**.



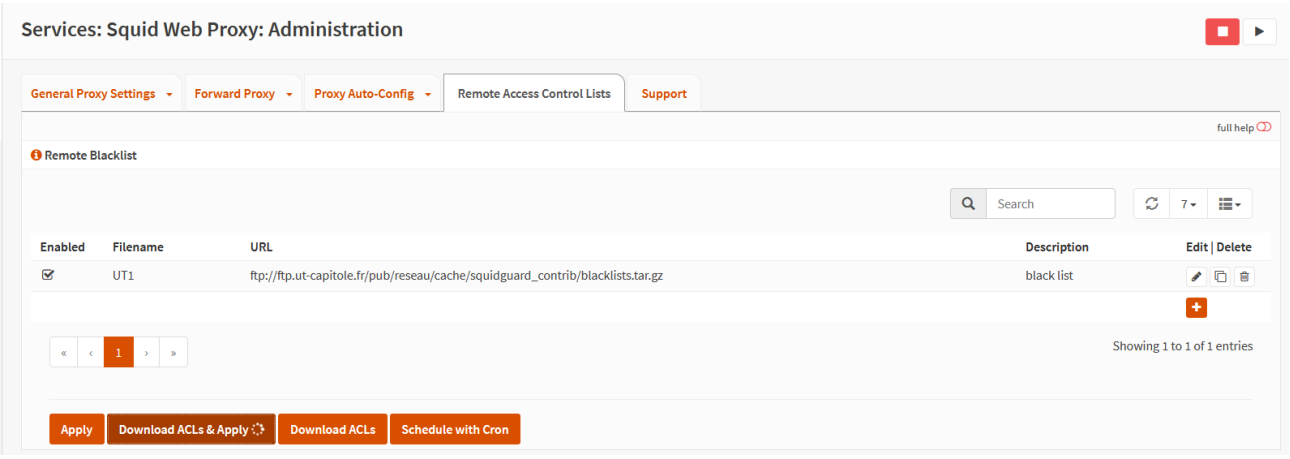
After defining proxy ports and enabling proxy methods (for HTTP or HTTPS traffic), in this step we can apply checklist to our proxy for web filtering, go to the **Services => Squid Web Proxy => Administration => Remote Access Control Lists** section and add new blacklist by pressing add (**plus**) button at the right of page.



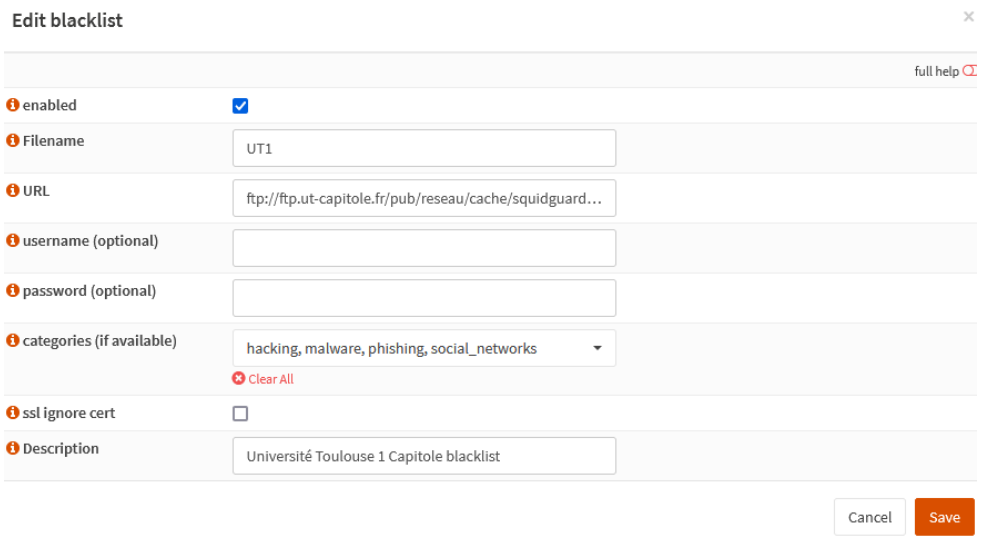
In the configuration page enable it, define the **Filename** whatever you want, paste the source address for blacklist into the **URL** section (feel free to use other source, although I used ftp://ftp.ut-capitole.fr/pub/reseau/cache/squidguard_contrib/blacklists.tar.gz) and type the **Description** for it. The another important section is the **"categories (if available)"** in this configuration. But it will be available when we **Save** this current configuration and download in the next step.



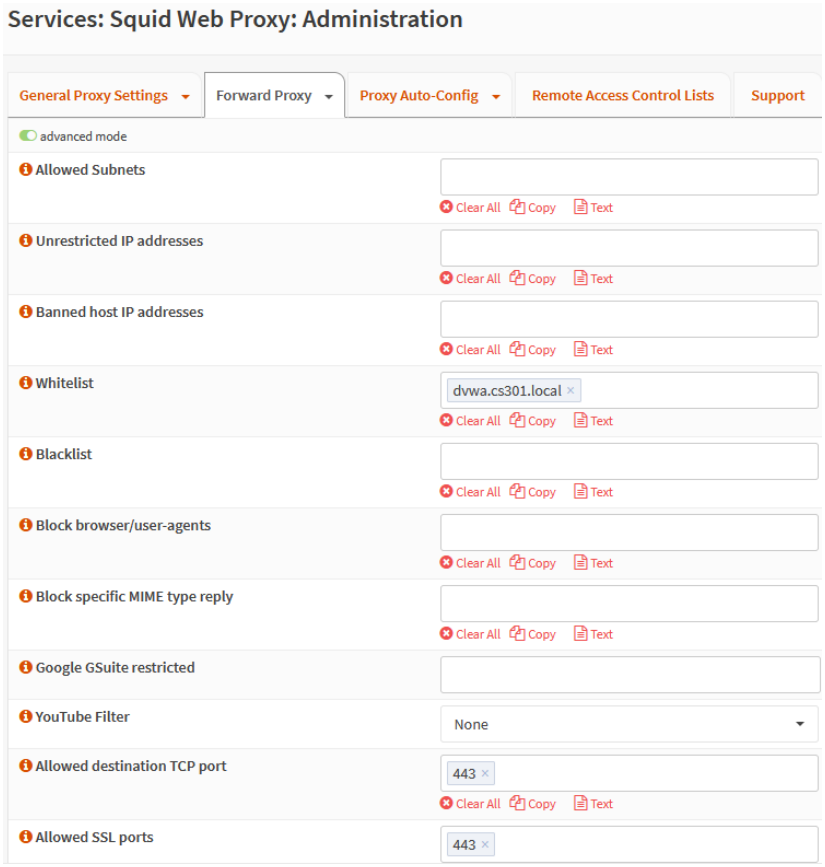
Once we **Save** the configuration, it will appear in **Services => Squid Web Proxy => Administration => Remote Access Control Lists**. Now we have to download this blacklist from the source **URL**, for this press the **"Download ACLs & Apply"** button and wait for a few minutes until it is downloaded into your firewall.



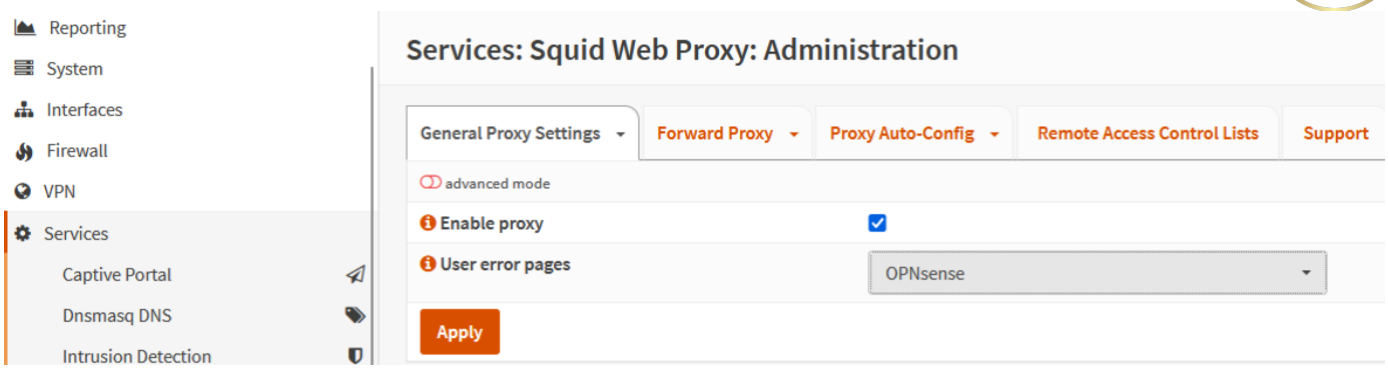
When the downloading and applying are finished, edit the configuration we created above. Now we can see that the **"categories (if available)"** section is available with multiple choices. This kind of blacklist is called category based where we are free to use whatever category we want for blocking in our selected machine. In this lab I leaved this section as default to block all included categories.



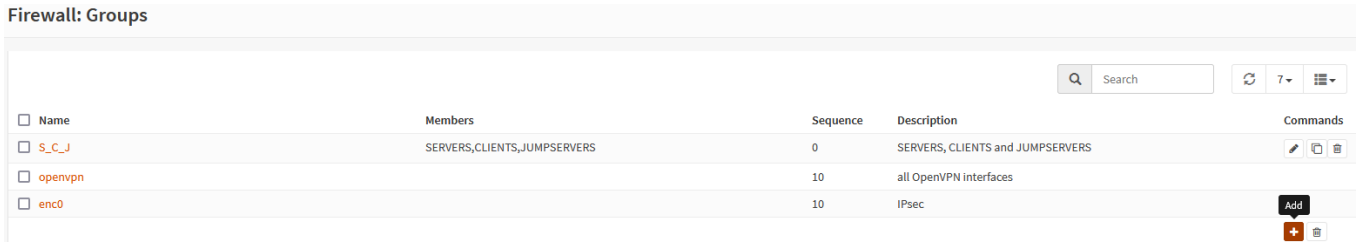
After choosing categories, go to the **Services => Squid Web Proxy => Administration => Forward Proxy => Access Control List** settings and activate the **"advanced mode"** on the left corner of configuration window. Then add our web application domain address (**dvwa.cs301.local**) into the Whitelist and type **HTTPS** port (**443**) number into **"Allowed destination TCP port"** and **"Allowed SSL ports"** sections.



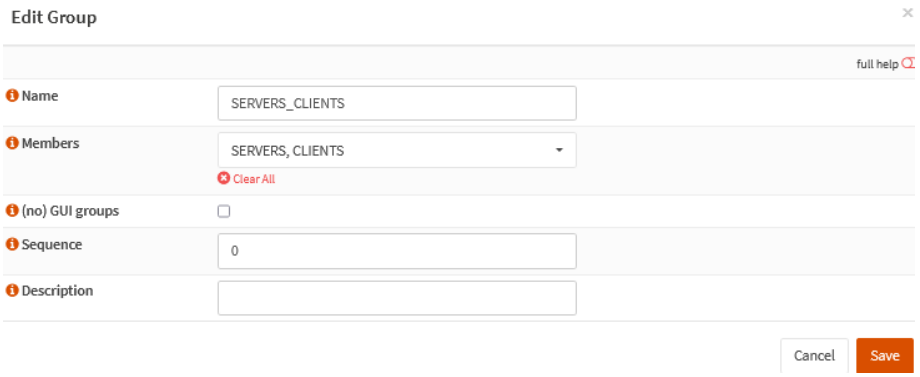
After finishing the whole configuration in **Squid** service, go to the **Services => Squid Web Proxy => Administration => General Proxy Settings** enable the service, choose **OPNsense** in the **"User error pages"** for displaying default error page to the client when any action is blocked by firewall (**Squid** service) because of web filtering and **Apply** changes.

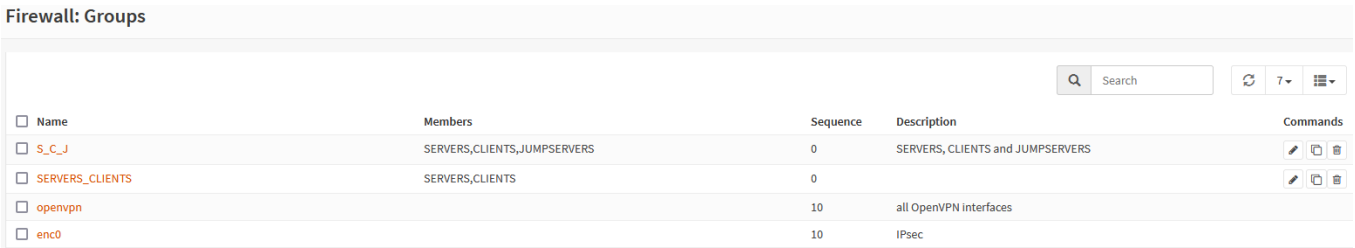


As you remember we choose two interfaces, **CLIENTS** and **SERVERS**, for web filtering, now we have to write a few separate rules for both networks. In a short way, we can add these both interfaces into the one group and write common rules for that group.
First we need to create a group, go to the **Firewall => Groups** section and add new group.



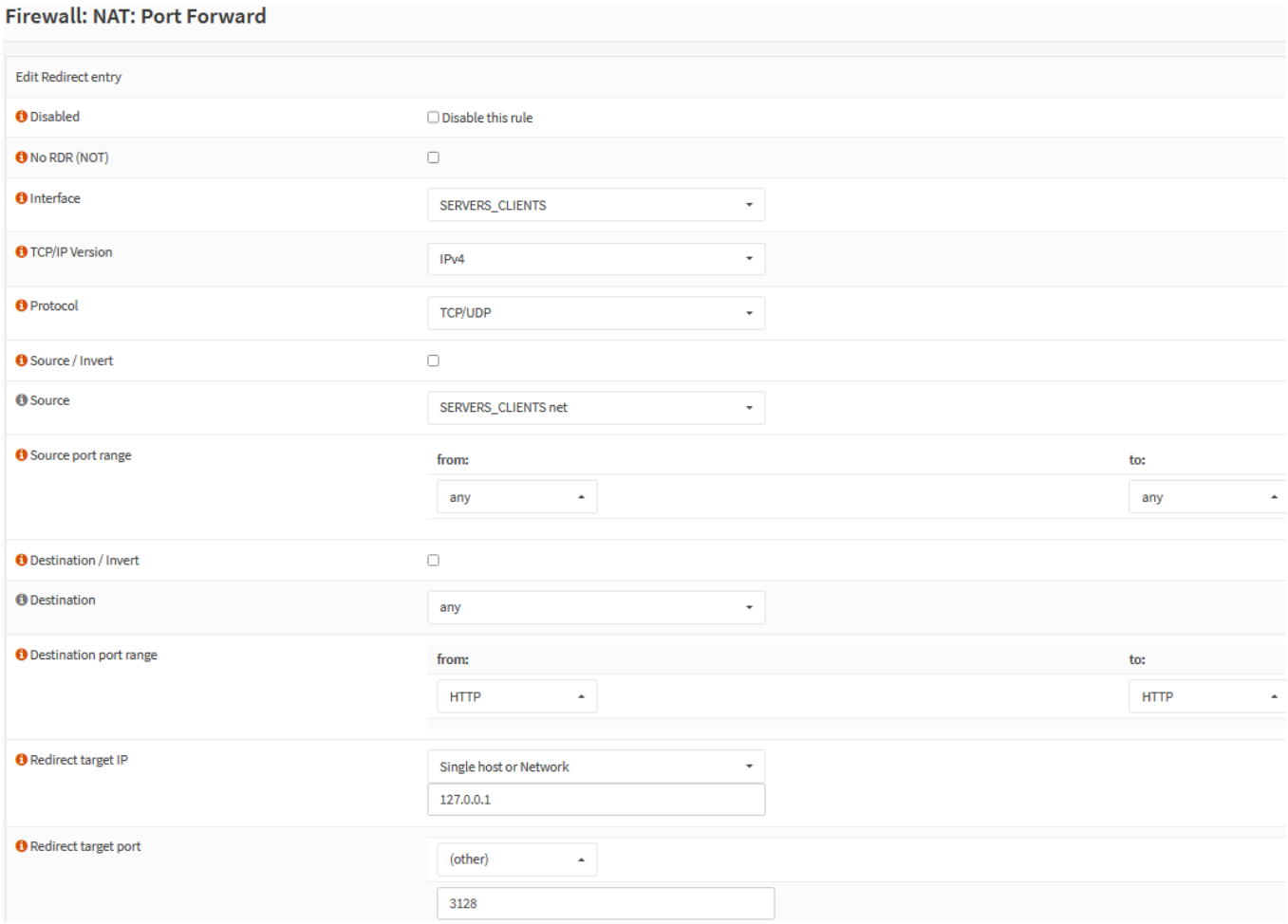
Enter the name for group and select the interfaces in the **"Edit Group"** page.

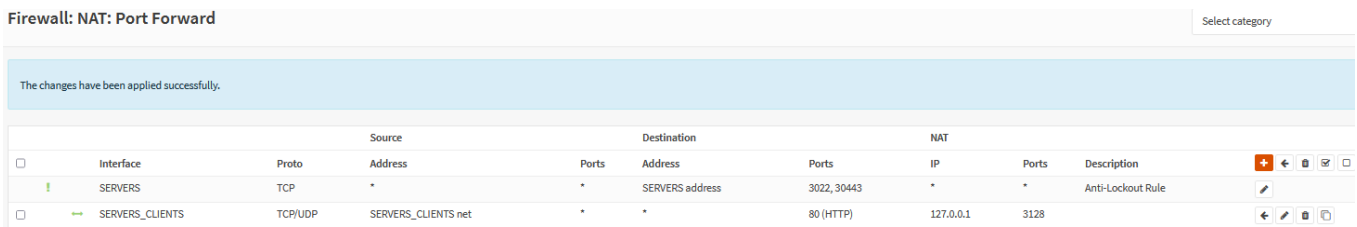




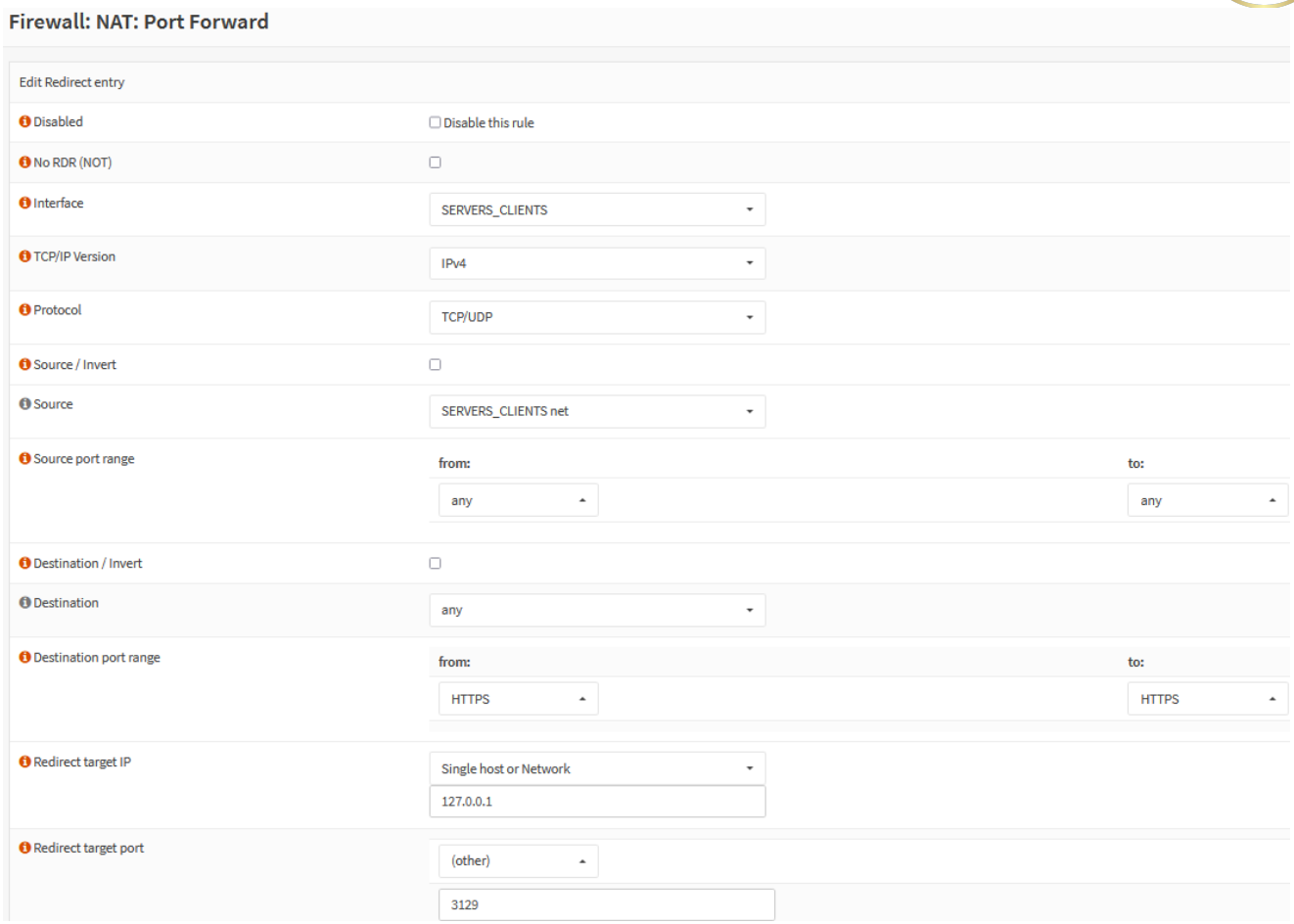
That was the basic configuration of the **Squid Web Proxy** service which will serve us as a web filer. But as you remember, in the first step, we defined two ports (**3128** for **HTTP** and **3129** for **HTTPS**) to flow traffic over our proxy service, and now we have to forward the port **80** to **3128** for **HTTP** requests and port **443** to **3129** for **HTTPS** requests. In order to doing it we have to write two new rules in firewall **NAT** configuration,

one for **HTTP** and the other one for **HTTPS**.
Go to the **Firewall => NAT => Port Forward** section and here, choose the **Interface** which you want to apply port forwarding (**SERVERS_CLIENTS** group for this lab, because I enabled web filtering only for this interfaces), define **Source** (again **SERVERS_CLIENTS** group for this lab), **"Destination port range"** as **HTTP**, **"Redirect target IP"** choose the option **"Single host or Network"** and type your firewall's **IP** address (or just loopback address as shown in the figure), **"Redirect target port"** choose the "**(other)**" option and enter the port address for **HTTP** which we defined in the earliest step of the **Squid** configuration and enter the redirection port **3128**.

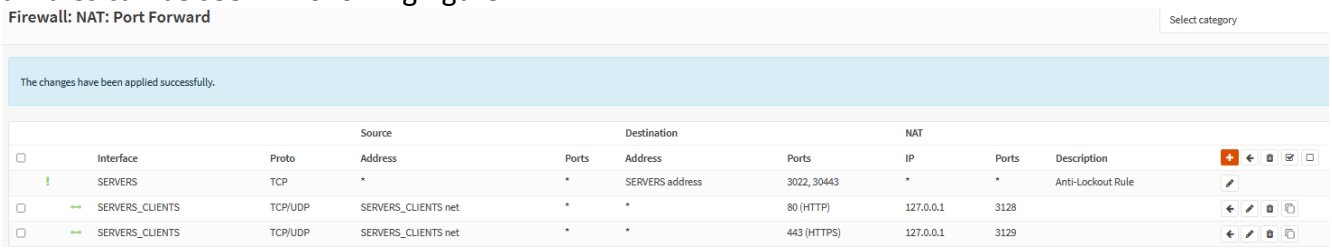




In the second rule we have to follow all steps we did above, but the only difference will be the **"Destination port range"** (**HTTPS** instead of **HTTP**) and the **"Redirect target port"** value (**3129** instead of **3128**).

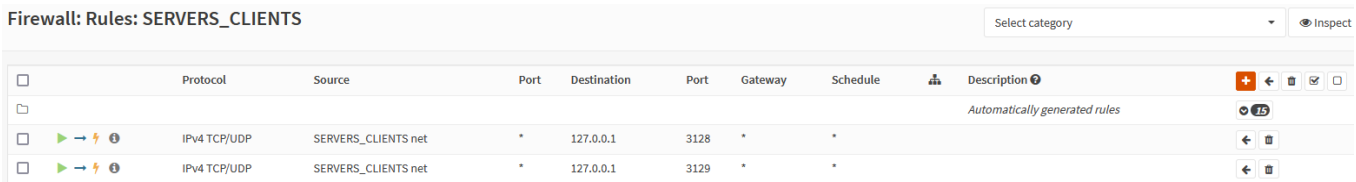


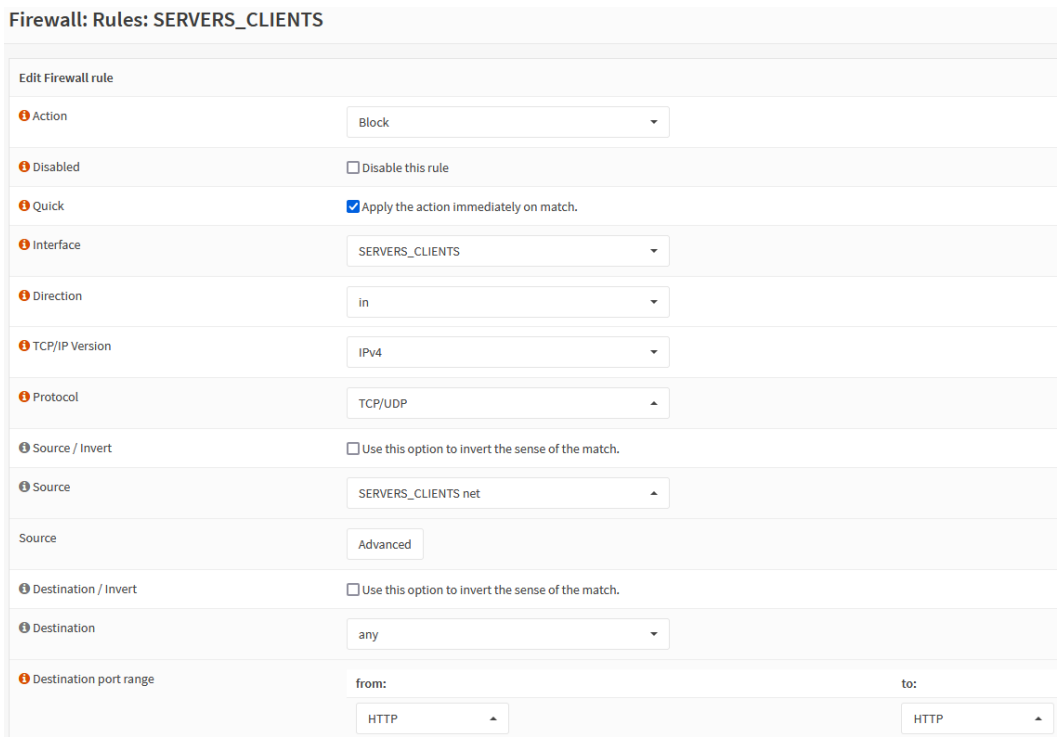
Both rules can be seen in following figure.

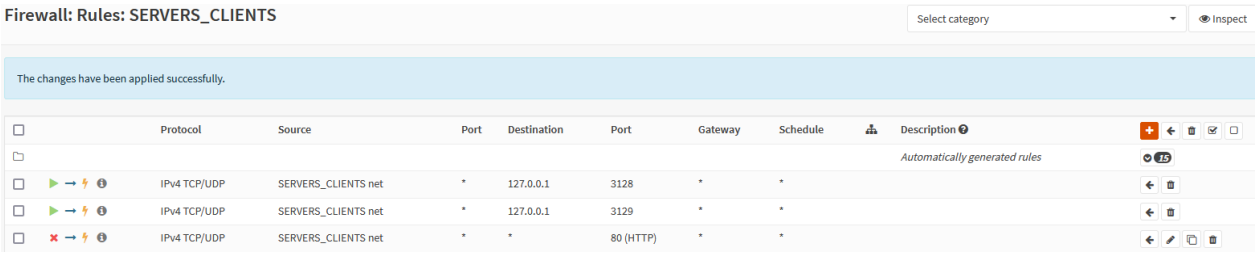


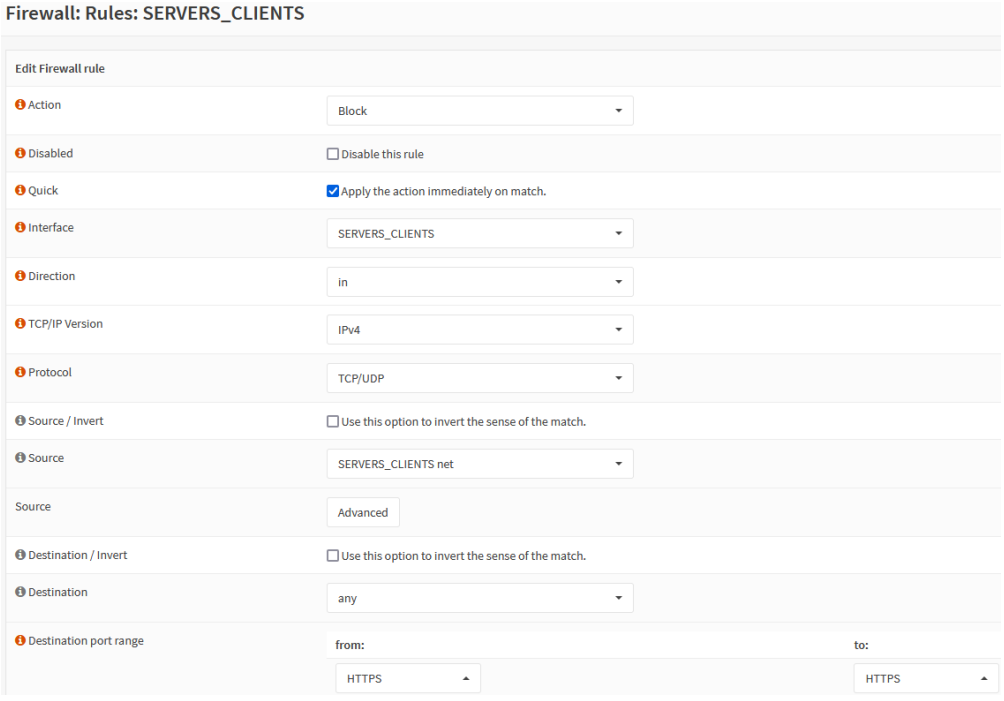
In the final step we need to block port **80** and **443** for web requests, otherwise the whole traffic will pass by our proxy and no web filtering will be applied. For this purpose, we have to write two rules in our firewall, one for blocking **HTTP** requests on port **80**, and the other one for blocking **HTTPS** requests on port **443**.

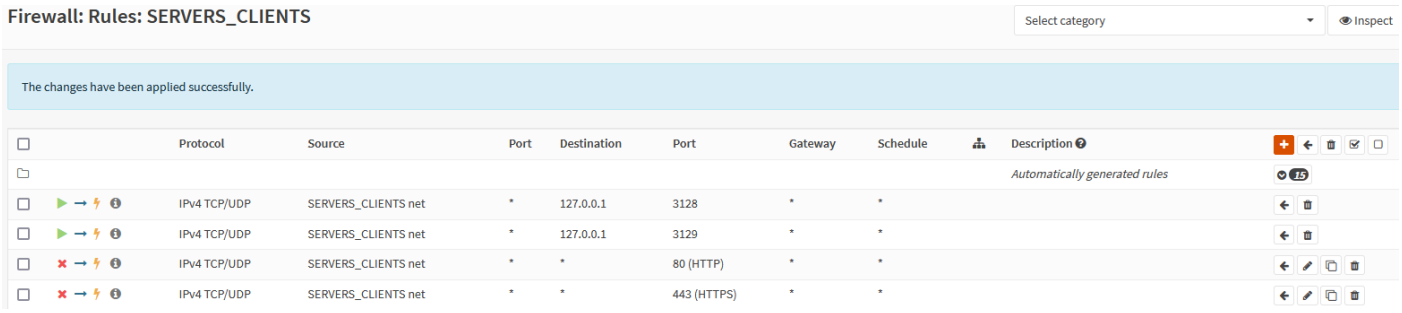
Go to the **Firewall => Rules** section in firewall and choose the interface (**SERVERS_CLIENTS** for this lab) which you want to add a new rule. In the content of the rule choose the **Action Block**, **Interface** whatever you want (I choose **SERVERS_CLIENTS**), **Protocol TCP/UDP**, Source the interface or specific **IP** (I choose whole **SERVERS_CLIENTS** interfaces), **"Destination port range"** from **HTTP** (**HTTPS** in second rule) to **HTTP** (**HTTPS** in second rule), **Category "Block proxy bypass"** and **Description** whatever you want.



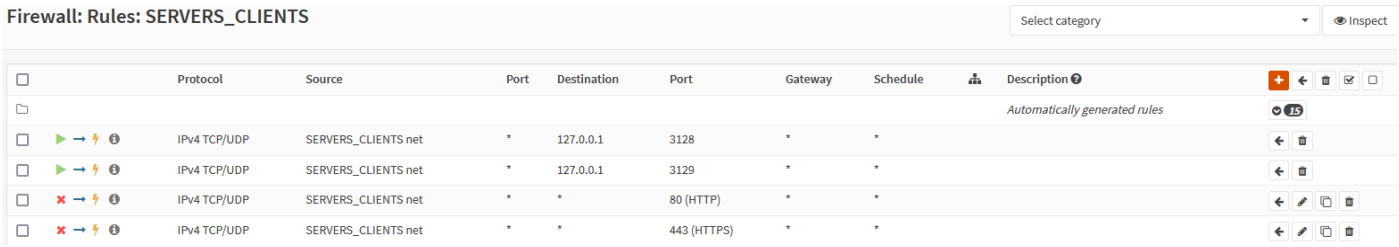

Firewall: Rules: SERVERS_CLIENTS
Edit Firewall rule
Action
Block
Disabled
Disable this rule
Quick
Apply the action immediately on match.
Interface
SERVERS_CLIENTS
Direction
in
TCP/IP Version
IPv4
Protocol
TCP/UDP
Source / Invert
Use this option to invert the sense of the match.
Source
SERVERS_CLIENTS net
Source
Advanced
Destination / Invert
Use this option to invert the sense of the match.
Destination
any
Destination port range
from:
to:
HTTP
HTTP


Firewall: Rules: SERVERS_CLIENTS
Select category
Inspect
The changes have been applied successfully.
Protocol
Source
Port
Destination
Port
Gateway
Schedule
Description
Automatically generated rules
IPv4 TCP/UDP
SERVERS_CLIENTS net
127.0.0.1
3128
IPv4 TCP/UDP
SERVERS_CLIENTS net
127.0.0.1
3129
IPv4 TCP/UDP
SERVERS_CLIENTS net
80 (HTTP)

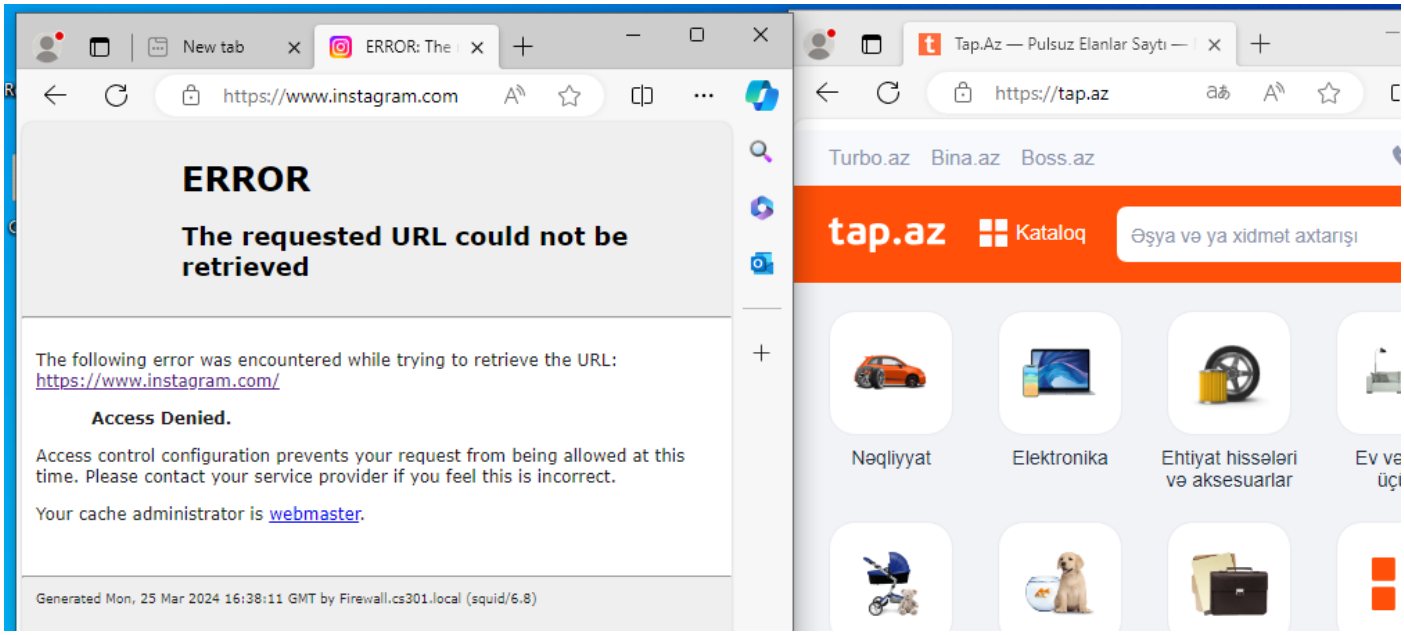

Firewall: Rules: SERVERS_CLIENTS
Edit Firewall rule
Action
Block
Disabled
Disable this rule
Quick
Apply the action immediately on match.
Interface
SERVERS_CLIENTS
Direction
in
TCP/IP Version
IPv4
Protocol
TCP/UDP
Source / Invert
Use this option to invert the sense of the match.
Source
SERVERS_CLIENTS net
Source
Advanced
Destination / Invert
Use this option to invert the sense of the match.
Destination
any
Destination port range
from:
to:
HTTPS
HTTPS



All rules (2 port forwarding rules and 2 port blocking rules) are displayed in the figure below.



It is time to test, go to the client machine, open browser and try to go somewhere is blocked. In the figure I tried to go https://www.instagram.com and https://www.tap.az web sites, but my first request was blocked by firewall because **Instagram** is a social media platform and we had blocked social media platforms in **Services => Squid Web Proxy => Administration => Remote Access Control Lists => categories** section, and the second one is allowed because it is a shopping platform.



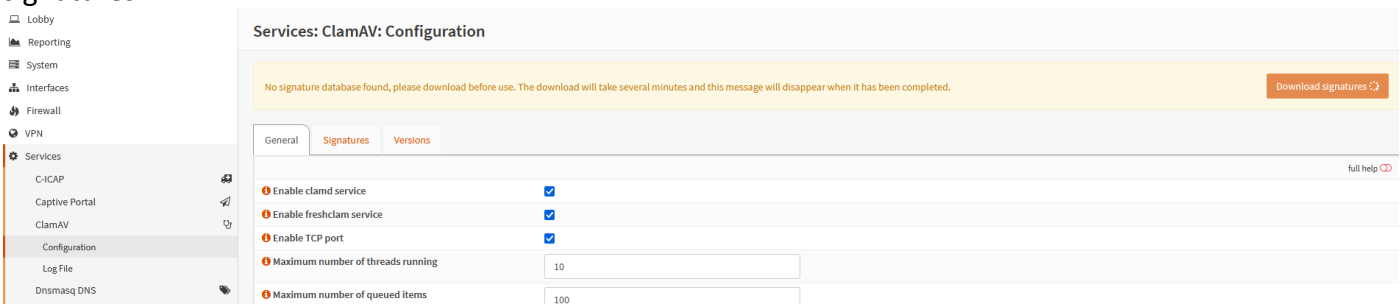
2.3 ClamAV and C-ICAP plugins

In the previous capture we tested web filtering feature in **Squid Web Proxy** plugin, which blocked our unwanted **HTTP** and **HTTPS** requests, but it doesn't mean that we are protected from harmful actions such as downloading malicious files (ransomwares, trojans, viruses etc.) because with a simple web proxy configuration we can't scan the content of traffic. In this point we need to add **ClamAV** and **C-ICAP** plugins' features to our web proxy.

ClamAV is an open source antivirus software and the **OPNsense** is able to use it as a plugin. We need an additional protocol as a glue between **Squid Web Proxy** and **ClamAV** which is called **ICAP** and **OPNsense** has its plugin, **C-ICAP**, too.

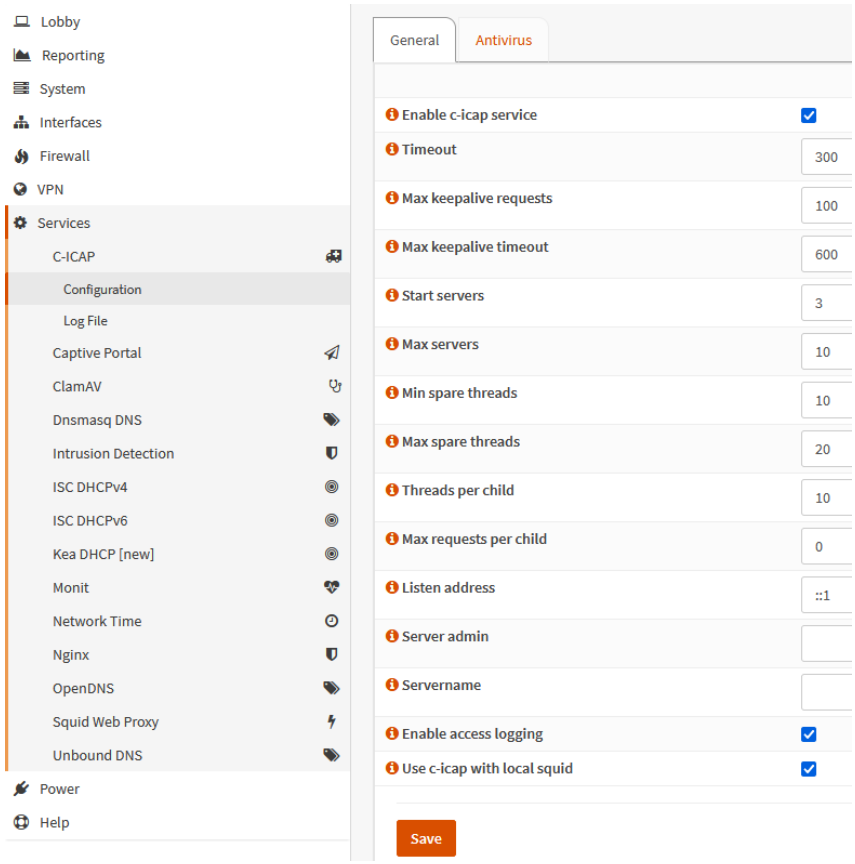
First, download **ClamAV** plugin, go to the **System => Firmware => Plugins** section, in the search bar type "**os-clamav**" and install the result.

Open **Services => ClamAV => Configuration**, enable **"Enable clamd service"**, **"Enable freshclam service"** and **"Enable TCP port"** options and leave rest of them as default. At the right corner of the page, click the **"Download signatures"** button and **Save** changes. Our antivirus plugin will work with the help of these signatures.

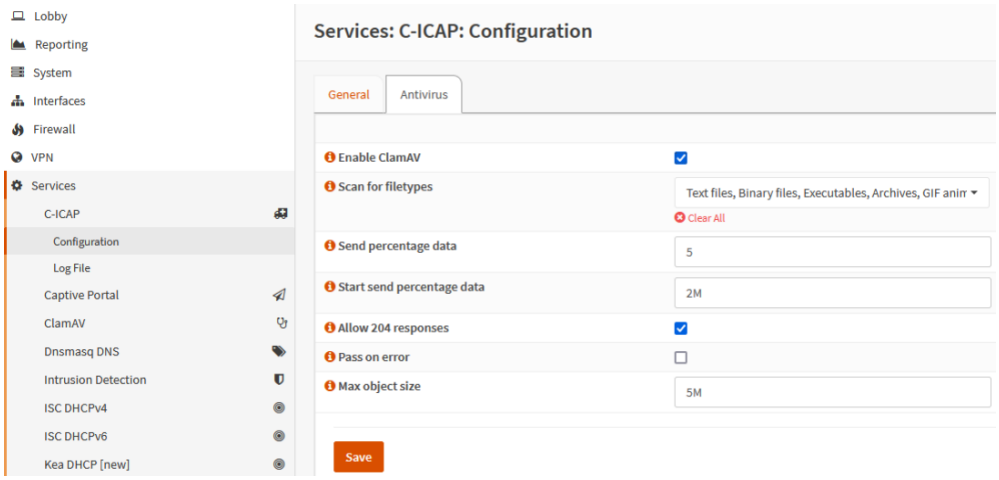


Once the **ClamAV** plugin setup finished, install the **C-ICAP** plugin with the **"os-c-icap"** keyword.

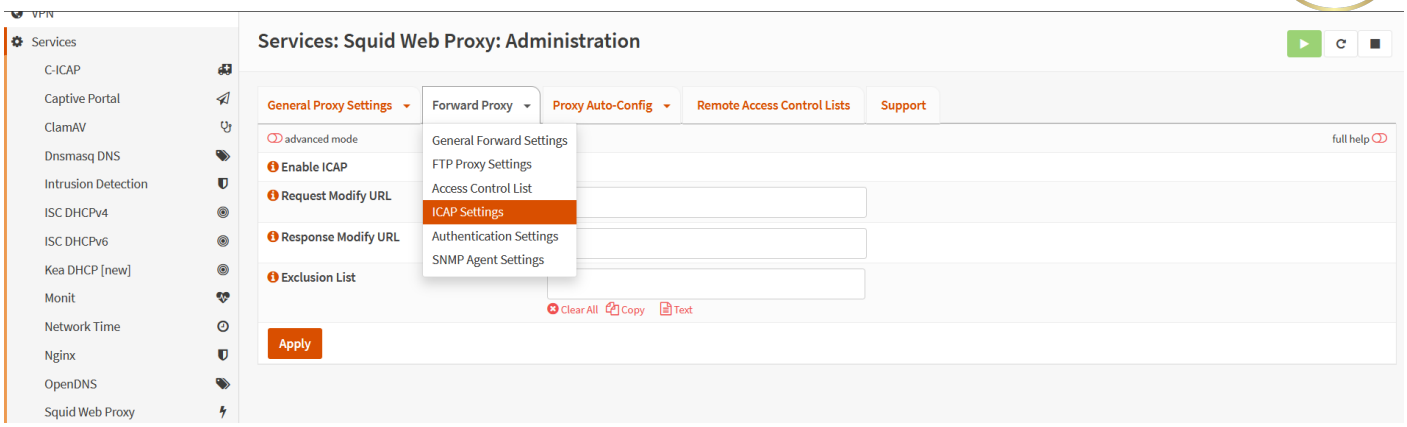
After the installation, **C-ICAP** plugin will be located in **Services** section. Go to the **Services => C-ICAP => Configuration => General** section and configure parameters like displayed in the figure.

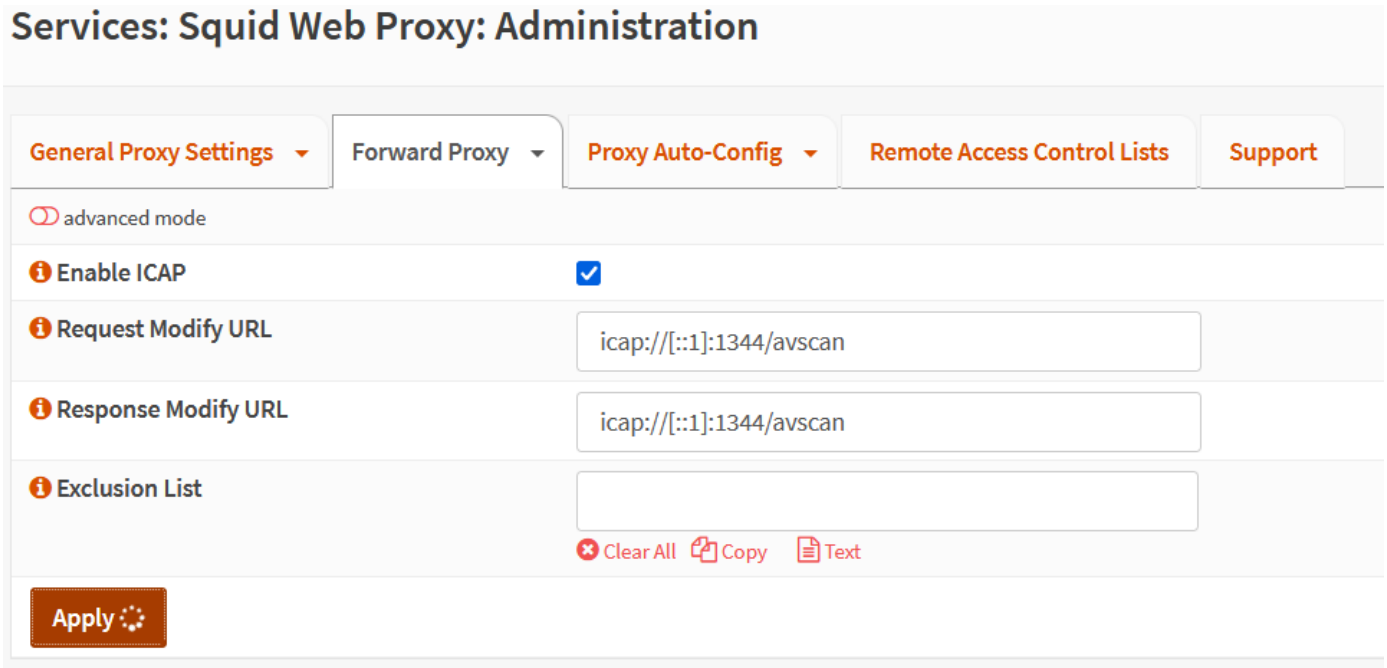


Later open the **Services => C-ICAP => Configuration => Antivirus** section and enable the **"Enable ClamAV"** option and leave other options default. The most interesting part of this configuration is **"Scan of filetypes"** option which lets us to control whatever we want to scan in our antivirus plugin.

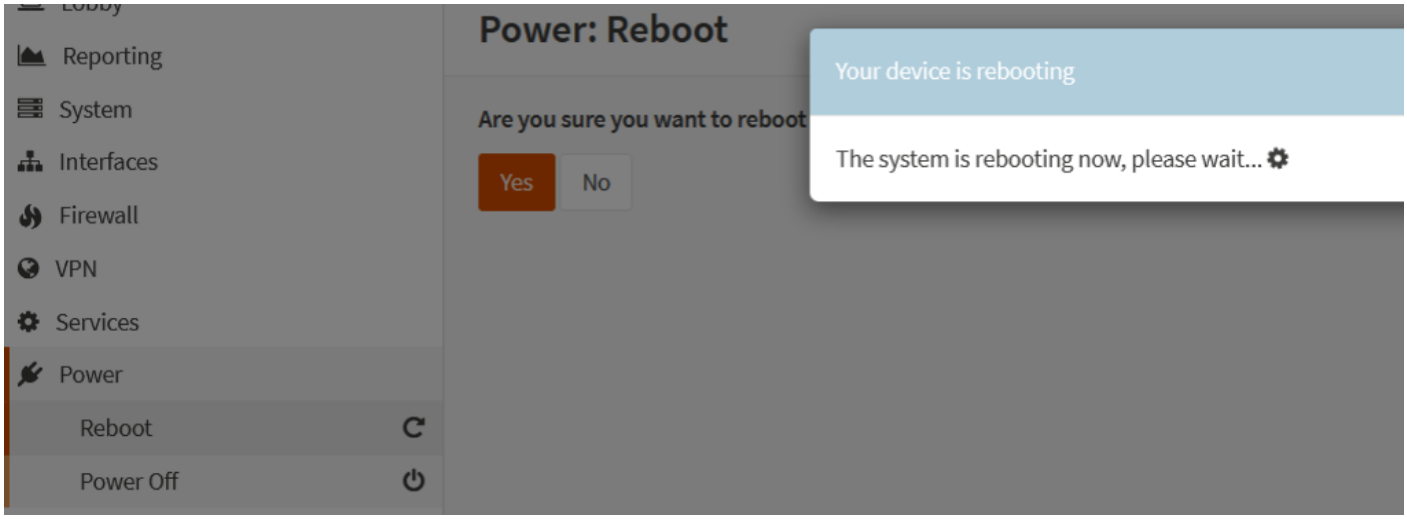


Now we need to enable **ICAP** option in our web proxy, go to the **Service => Squid Web Proxy => Administration => Forward Proxy => ICAP** Settings and enable the **"Enable ICAP"** option and type **"icap://[::1]:1344/avscan"** to both **"Request Modify URL"** and **"Response Modify URL"** sections.

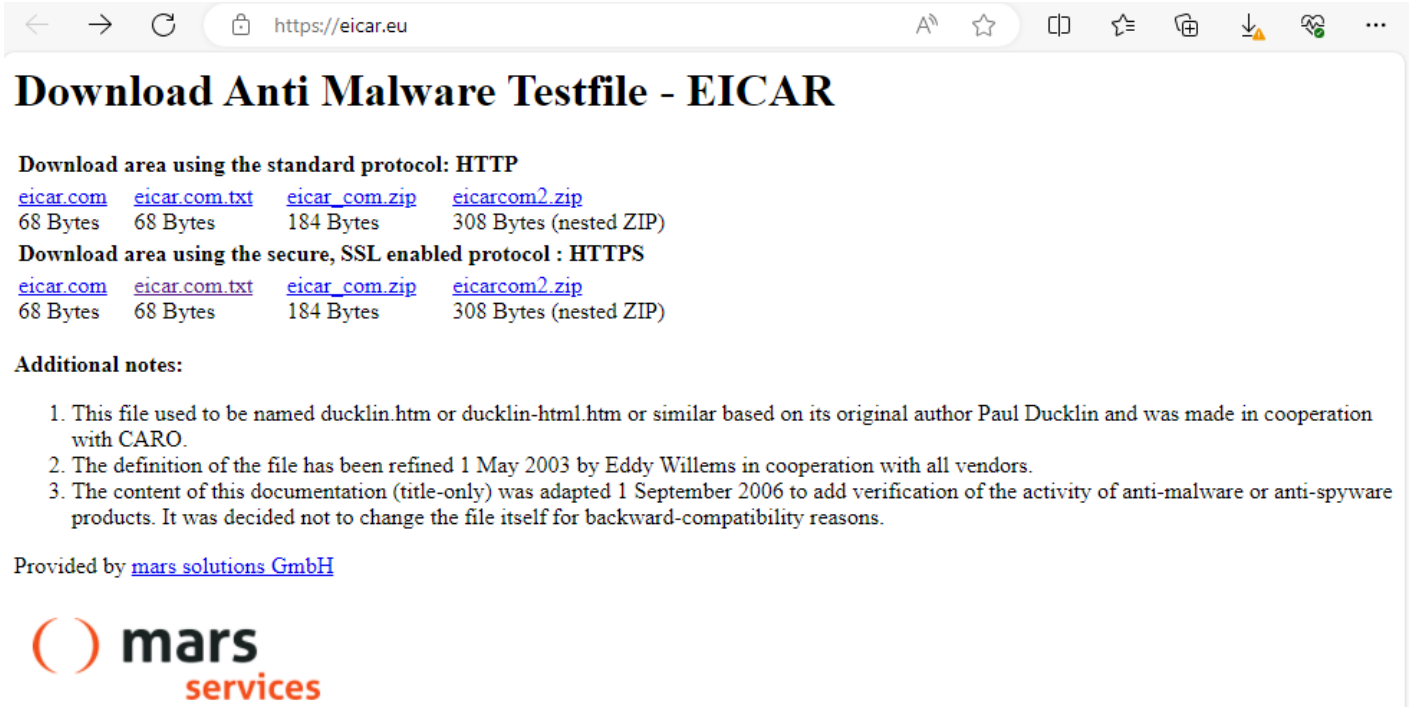




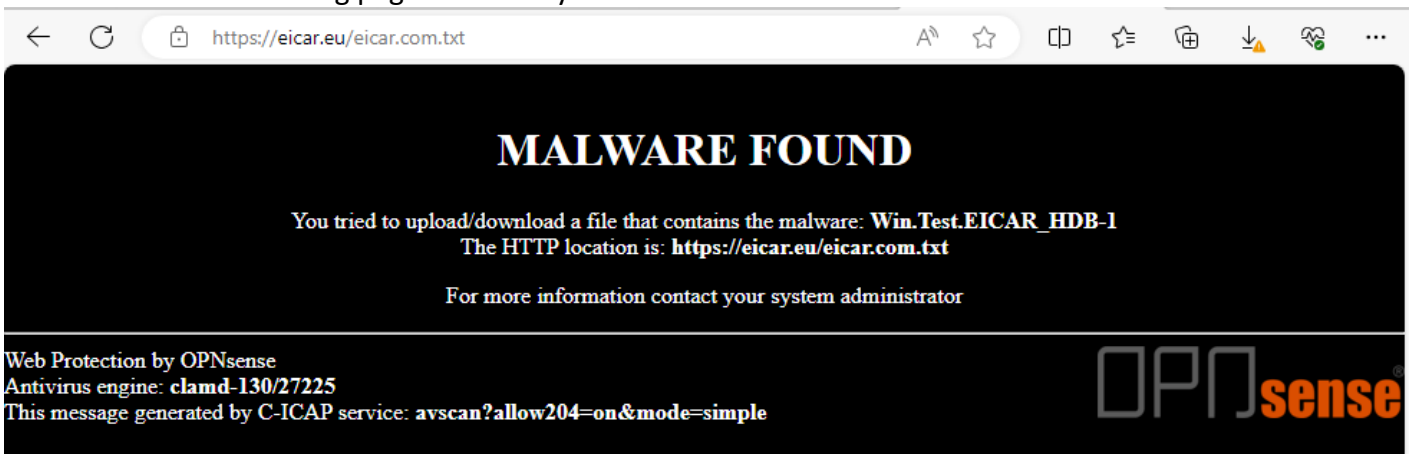
At the end of the whole installations and configurations, we must reboot the system for applying configurations, go to the **Power => Reboot** and wait for rebooting firewall.



After the reboot, our firewall is ready for caching malicious traffic with antivirus plugin. In order to test if it works, go to the machine where we installed our certificate and open the browser. In the search bar type the https://eicar.eu and open the web page. You will meet a few anti malware test files what we will try to download.

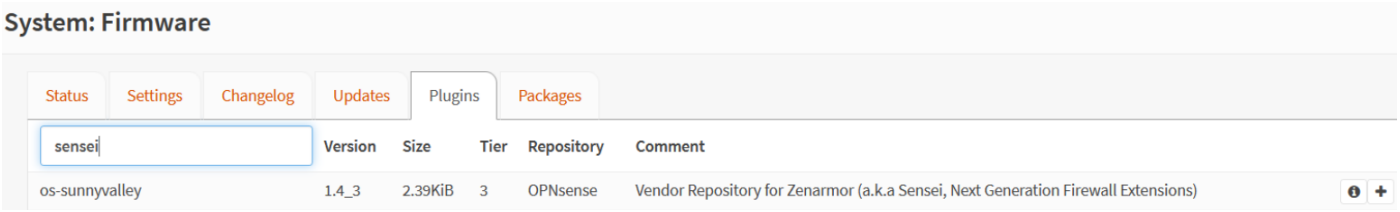


Whenever you click on the download link in the page it will be blocked by our antivirus and will display the **OPNsense** default warning page to inform you about the accident.

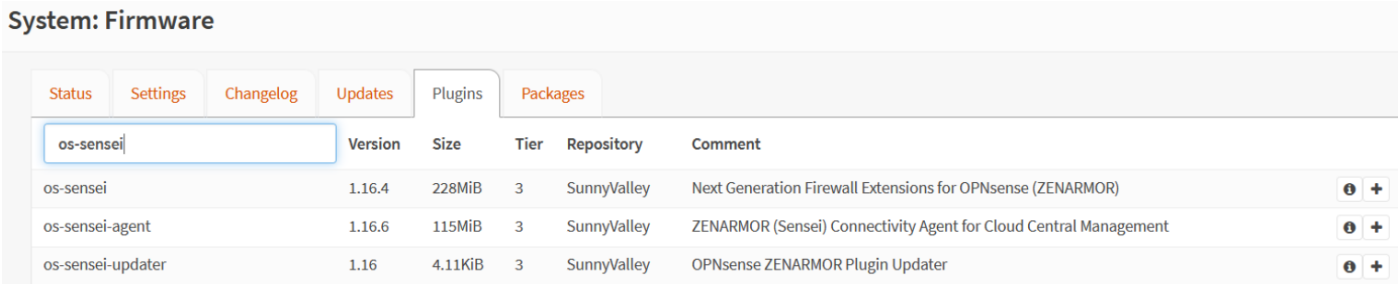


2.4 Zenarmor plugin

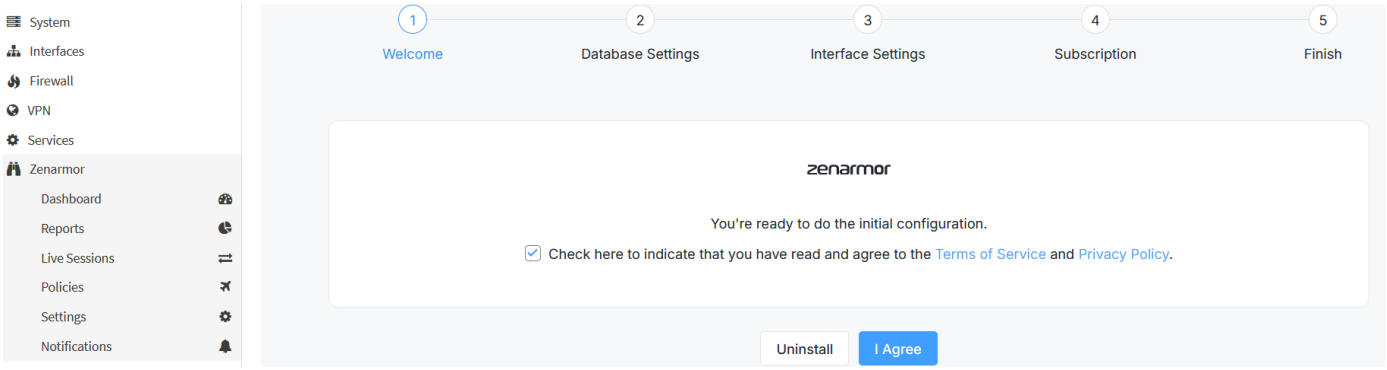
Like other open source firewalls, the **OPNsense** works in **L4**, which means in a traditional mode we can't control the **Application** layer. In this point, the **Zenarmor** plugin helps us to gain some next generation firewall features such as **Application Control**, **TLS Inspection**, **Network Analytics** and more.
Before we download **Zenarmor** packages, we have to install its repository what is called **"os-sunnywalley"**.



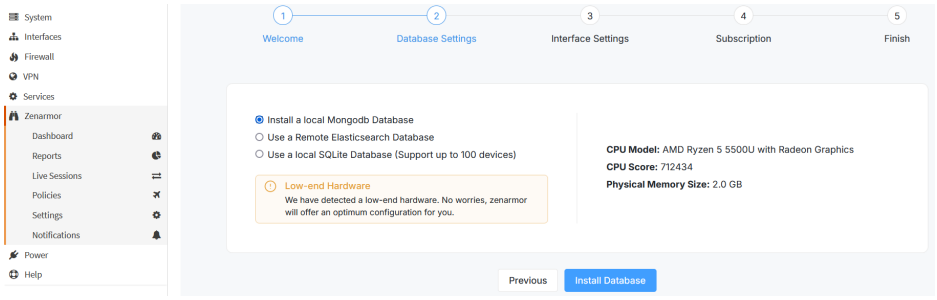
After downloading the repository, we can install the three packages, **"os-sensei"** (**Zenarmor** plugin) and **"os-sensei-update"** (updater for **Zenarmor**).



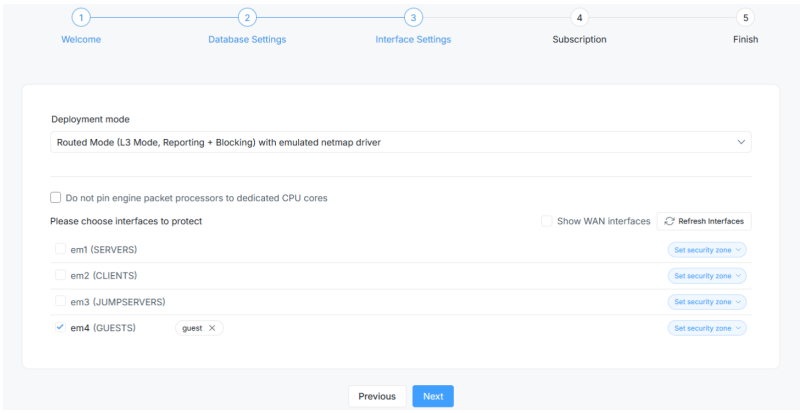
Once we finish the installation of packages, the **Zenarmor** will be located in settings column with **Zenarmor** name. Apart from other plugins we installed in previous captures, this one has its own environment to work on it, that is why it is not in the Services.
When we click on the **Zenarmor => Dashboard**, the installation page will meet us. The installation process is very simple, just follow the directives step by step.
1-Accept the **Terms of Services**, **Privacy Policies** and go to next.



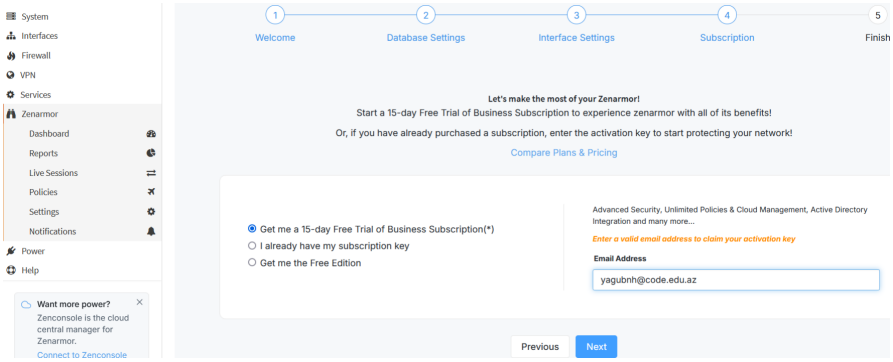
2-Cooshe the **"Install a local Mongodb Database"** and **Install Database**. After the installation is finished click **Next**.



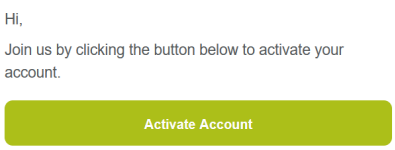
3-Choose **“Deployment mode”** as **“Routed Mode (L3 Mode, Reporting + Blocking) with emulated netmap driver”**, define the interfaces which you want to control and monitor via the **Zenarmor** and set the security zone for each interface (I choose only **GUESTS** interface because we already have protection on others).



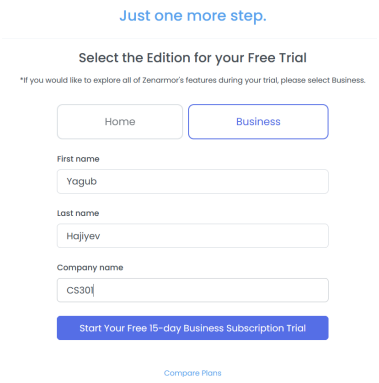
4-Choose the **“Get me a 15-day Free Trial of Business Subscription(*)”** and type your email for subscribing and click Next button.

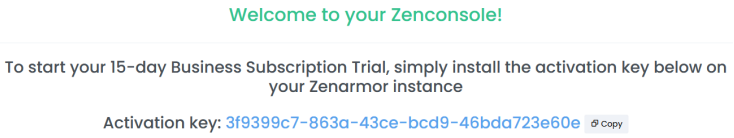


It will send an activation email for creating new account, open it and click on the **“Activate Account”**.

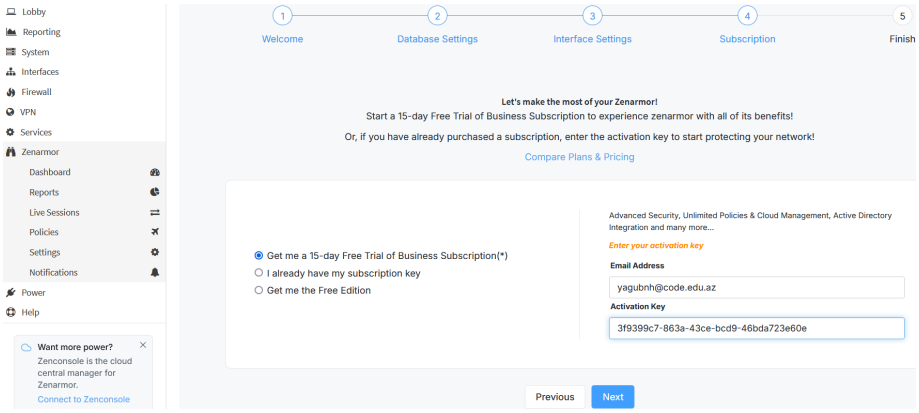


In the new page choose the **Business** or **Home** (it doesn’t matter) version and fil the boxes. When you sign up successfully, it will give you an activation key, copy it.

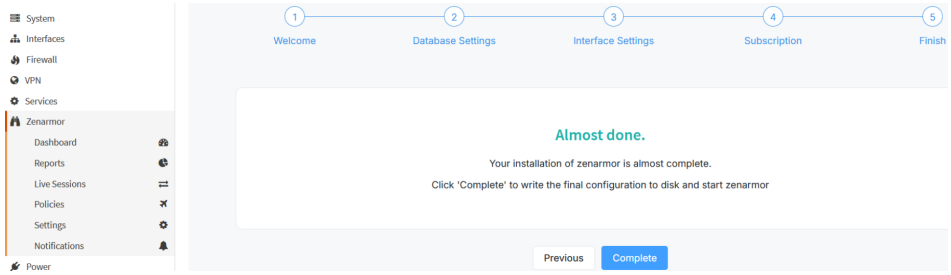




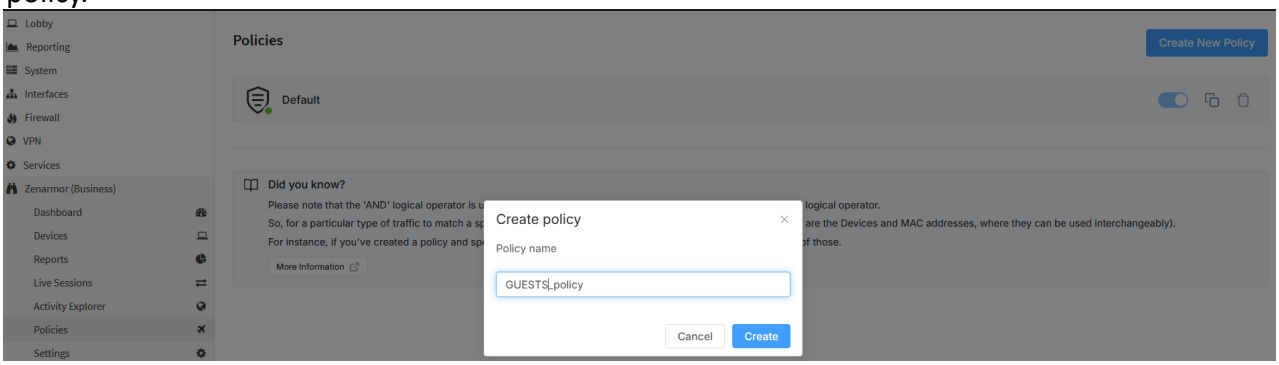
Paste your activation key and go on the next step.



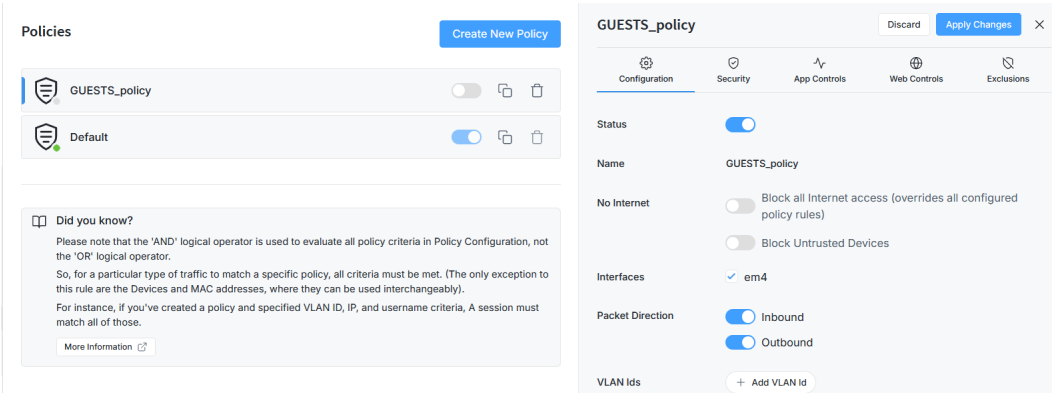
5-**Complete** the installation and our **Zenarmor** is ready to use.



Now our plugin is ready to use, go to the **Zenarmor => Policies** section and click on the **"Create New Policy"** button on the right corner of the page and in the popped up window, type the name you want to give to your policy.

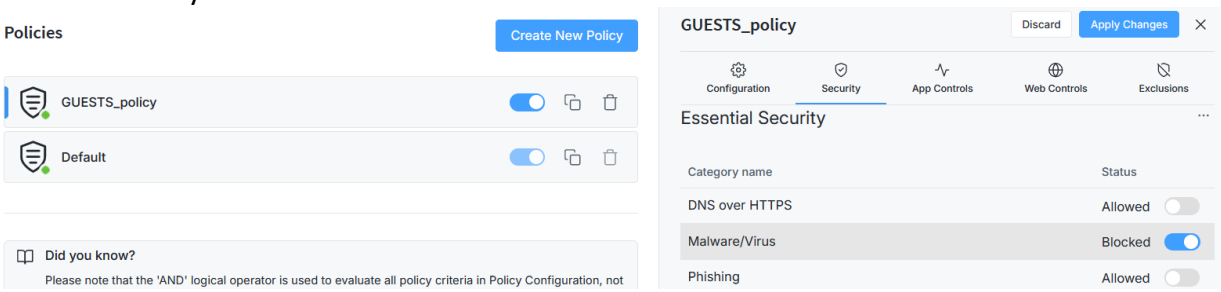


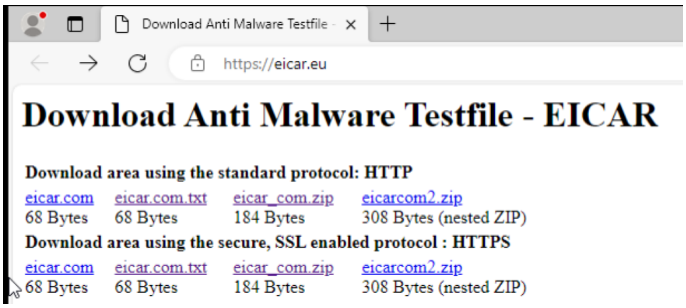
After creating the policy click on the policy name for editing it. This will open the editing page, from the **Interfaces** section choose the interface (**em4**) to apply your policy and enable the **Status**.



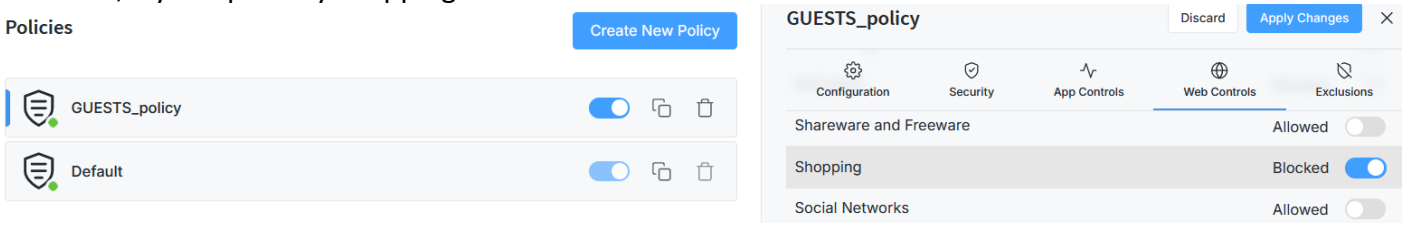
Later open the Security options page and choose the security policies for selected interface input and output traffic and **"Apply Changes"**. In our lab we will test a few features of **Zenarmor** such as malware blocking, web filtering and creating whitelist.

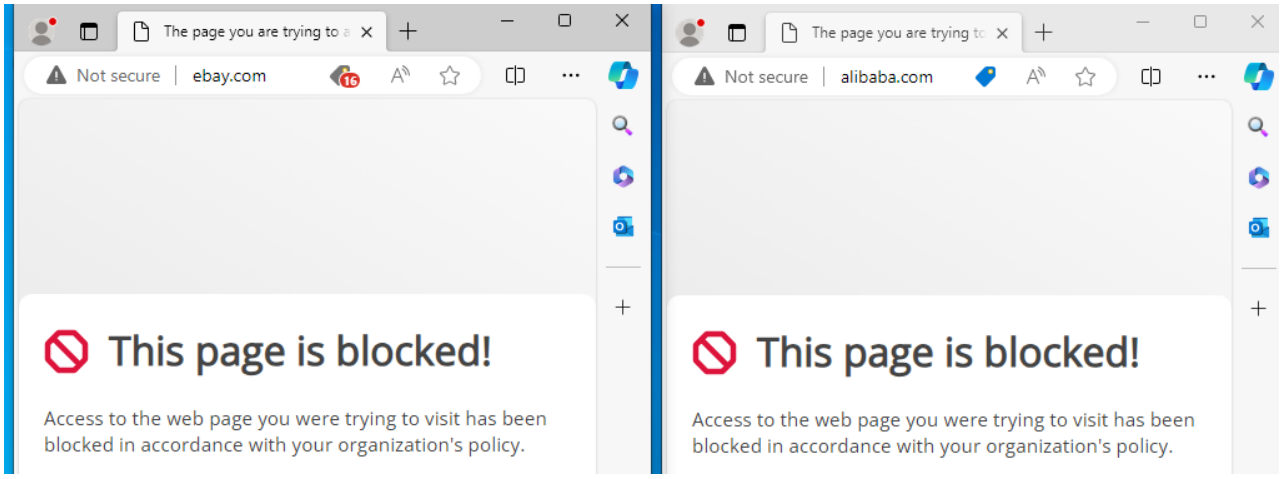
Malware blocking: go to the **Security** section and enable the **Malware/Virus** option in your policy. Later open any machine in **GUESTS** network and search for www.eicar.eu website. In this site try to open the eicar.com.txt file and you will see it.



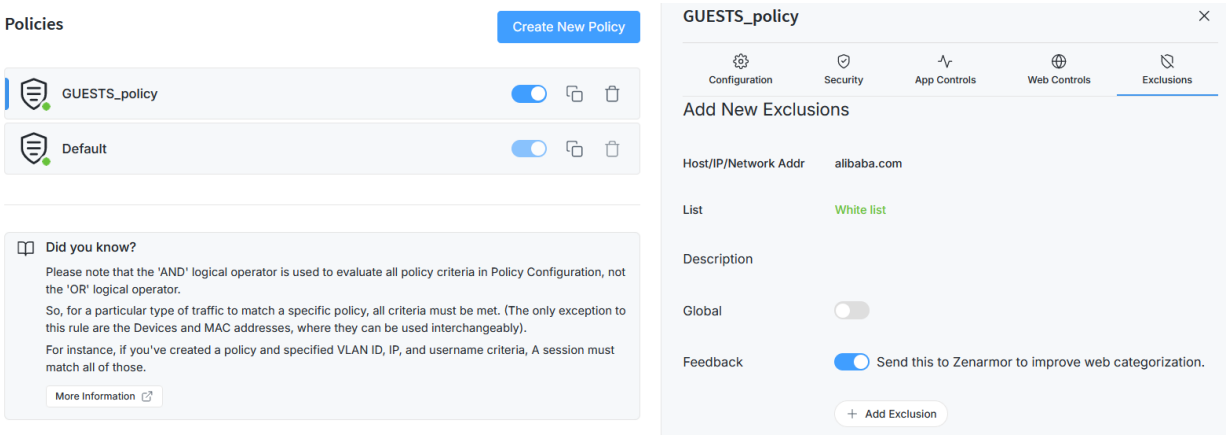


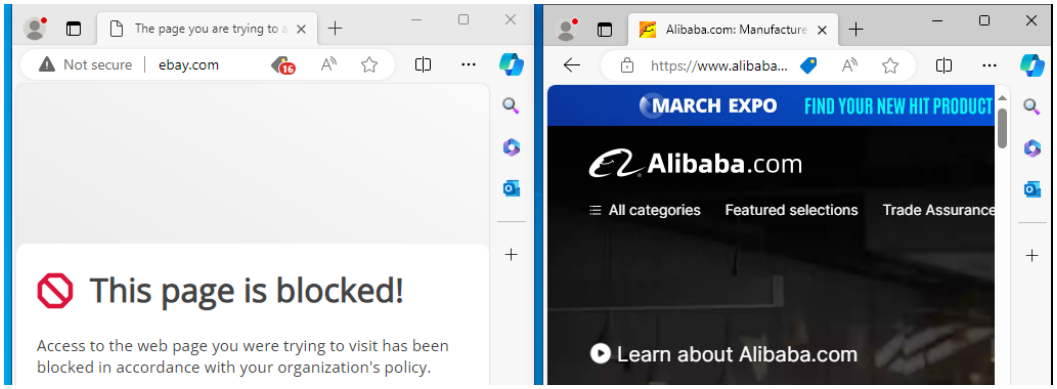
Web filtering: again go to the **"Web Controls"** in the policy and enable the **Shopping** option. In **GUESTS** interface, try to open any shopping site.





Whitelist: we saw that the all shopping sites are blocked by **Zenarmor** plugin, now we will add **alibaba.com** to the whitelist. Open your policy and in the **Exclusions** section, add the domain you want to add into the whitelist (or black list, you can change the mode in **List** option) and click on the **"Add Exclusion"** button and apply changes. When you visit www.alibaba.com address, firewall will let your browser to load page.



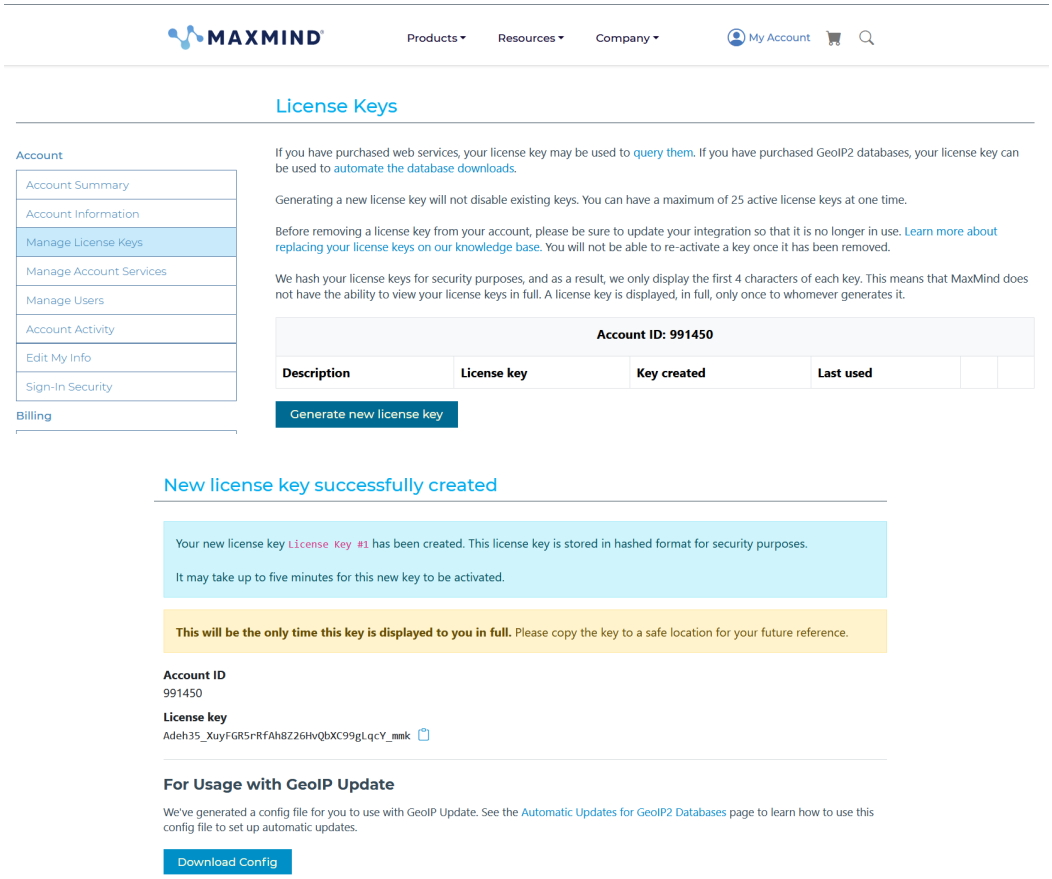


OPNsense additional features

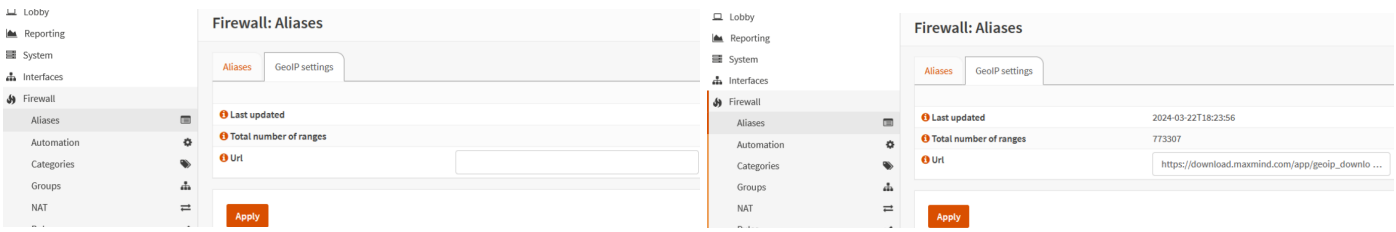
3.1 GeoIP blocking

One of the most essential feature of firewalls' is blocking **IP**s in whole geographical region such as city, country. In **OPNsense** to enable **GeoIP** blocking function, go to the **Firewall => Aliases => GeoIP settings**. In the **URL** section we need to add the source to download the **IP** ranges for regions. There numerous sources available on the **Internet** and we will use **Maxmind** platform in this lab.

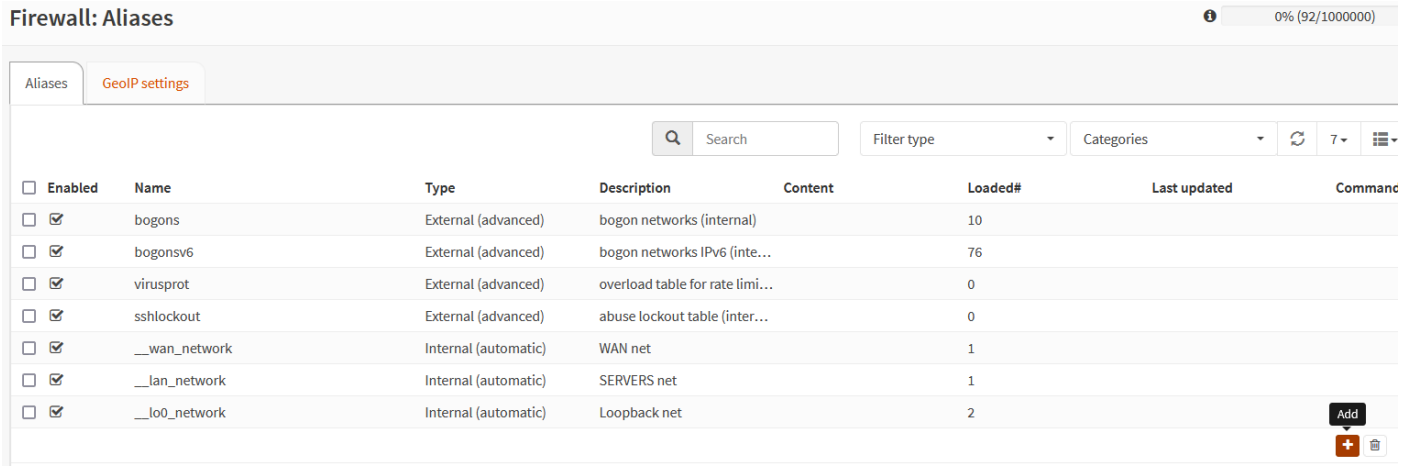
Go to the https://www.maxmind.com address and create an account, next enter to your profile and generate new license key.



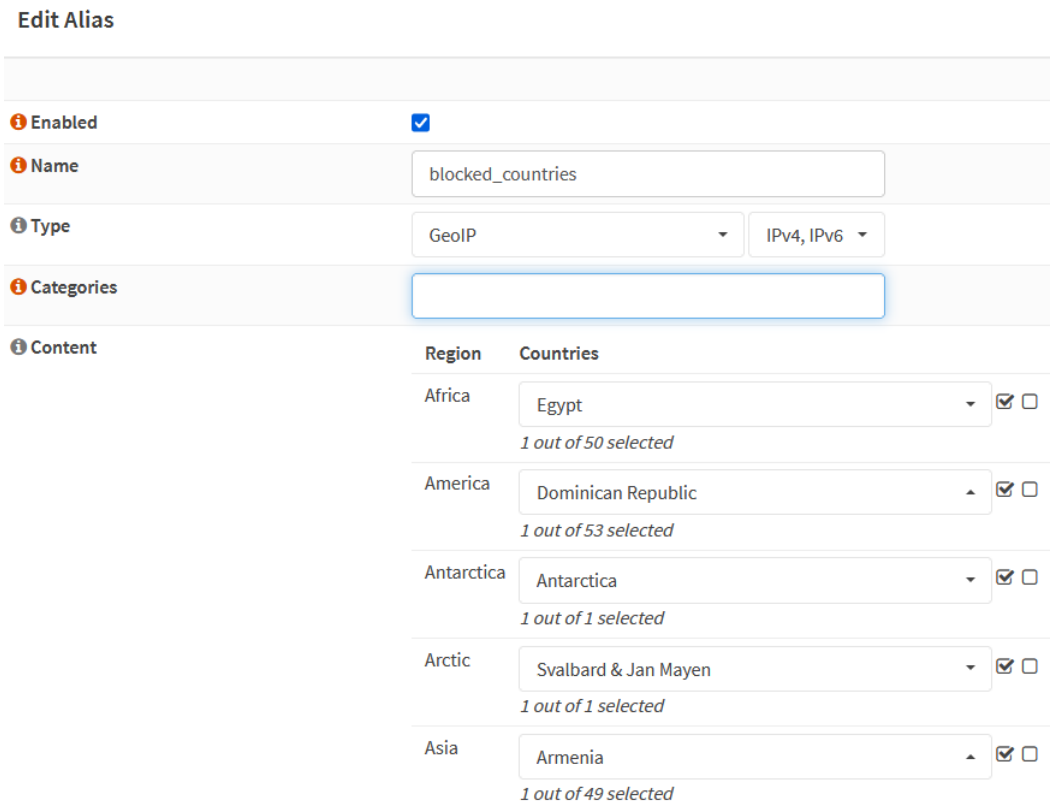
After getting our license key we must create the **URL**. For the **Maxmind**, we can use a template **URL**: https://download.maxmind.com/app/geoip_download?edition_id=GeoLite2-Country-CSV&license_key=YOUR_KEY&suffix=zip Replace **YOUR_KEY** with the key you created and paste the new **URL** into **Firewall => Aliases => GeoIP settings => URL** section and **Apply**.



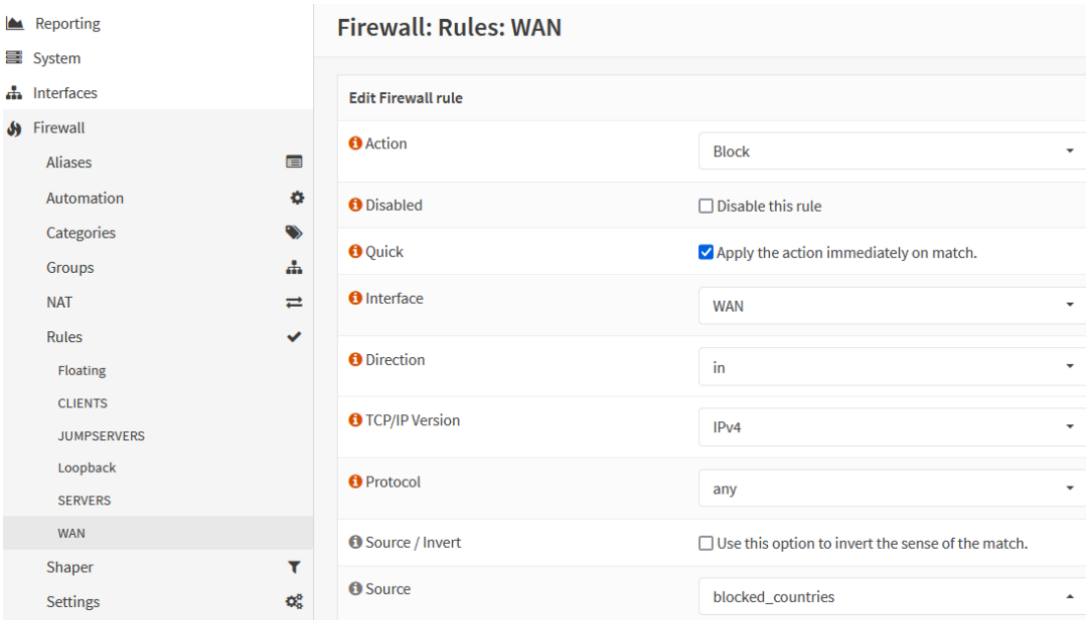
Now we have the **GeoIP** source and the next step is to create an aliases to in our rule to block input/output traffic for selected regions. Open **Firewall => Aliases** section and add new one.



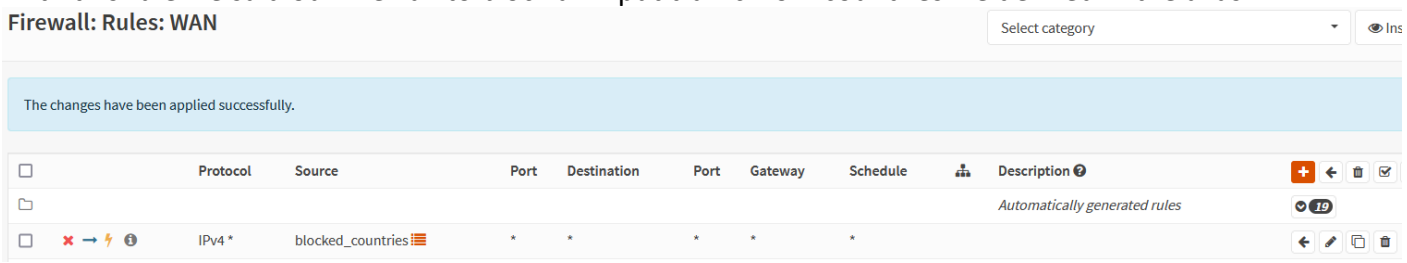
In the alias editing window, give the to your alias in the **Name** section, choose the **GeoIP** option in **Type** section and choose countries you want to block.



In the last step we will create a new rule for blocking **IP**s in region. In order to block this traffic, we must append this rule to our **WAN** interface, because only it is able to communicate with outside of our networks (**Internet**). Go to the **Firewall => Rules => WAN**, set the **Action** as **Block** and choose the alias we created above in the **Source** section.



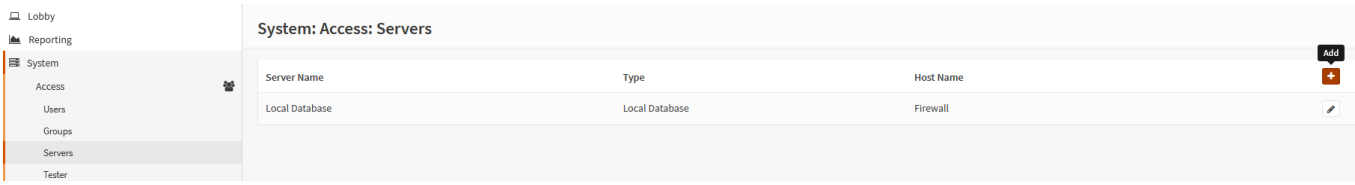
With this rule we said our firewall to block all input traffic from countries we defined in the alias.



3.2 Captive Portal

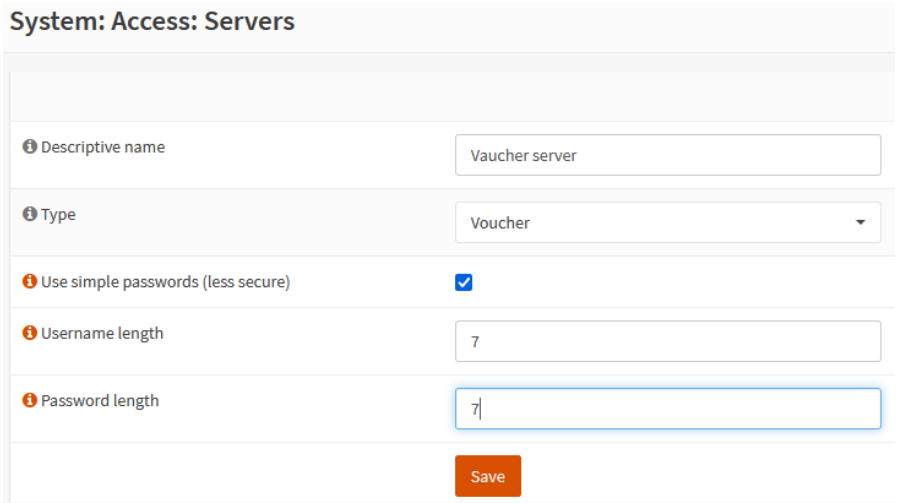
As you remember, in previous captures, we mentioned that the **GUESTS** network should isolated from our environment. We fulfilled it with firewall rules, and in this capture we will setup a specific service to authenticate our guests to give an access to the **Internet** over our **GUESTS** network. This service is called **Captive Portal**, which is a customized login page that guests must address before connecting to the **Internet**. This service has additional features to use for authenticating and limiting network traffic in it, but we will use only one method, **Vouchers**, to control authentication process. In this method we will create a login page and login credentials for guests, what is called voucher. Each voucher will have a validity time limit and we will provide our guests with this vouchers whenever they ask for credentials to connect to the Internet.

Before we start configuring **Captive Portal**, we need to create an access server for using it in our service. Go to the **System => Access => Servers** and add new server.

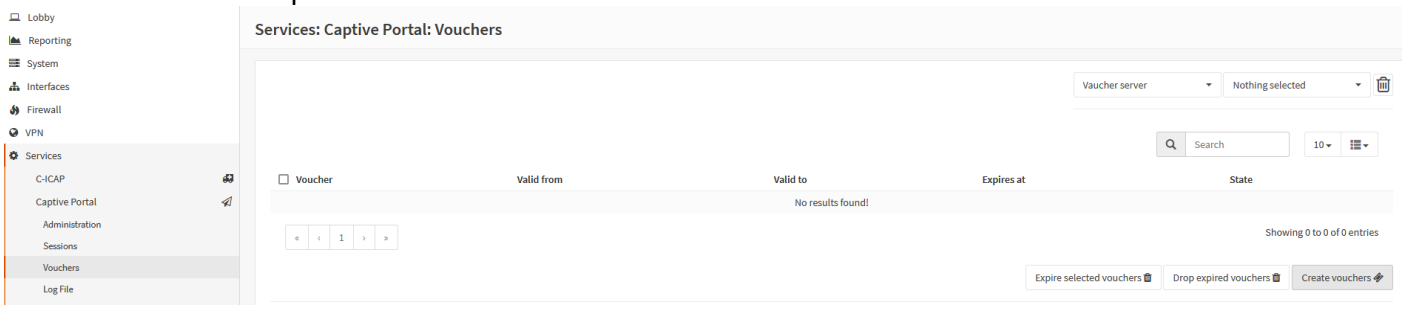


In the configuration window, type a **"Descriptive name",** choose **Voucher** option in **Type**, put the tic in **"Use**

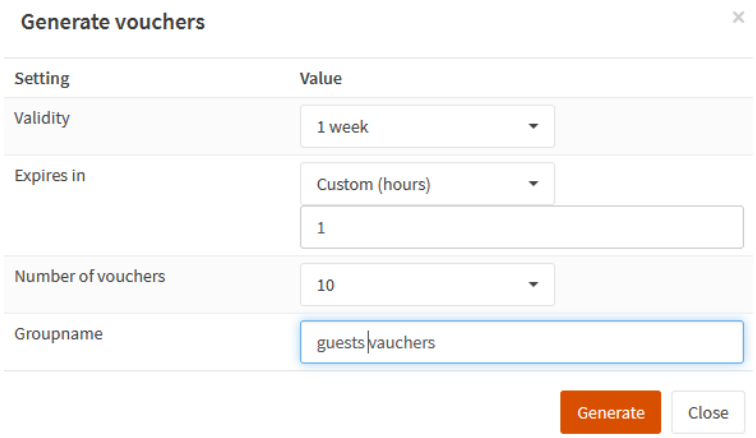
simple passwords (less secure)" box (it is a little bit user friendly way to creating passwords) and define the credentials' lengthes.



In the next step, we should generate our vouchers for guests, go to the **Services => Captive Portal => Vouchers** section and press the **"Create Vouchers"** button.

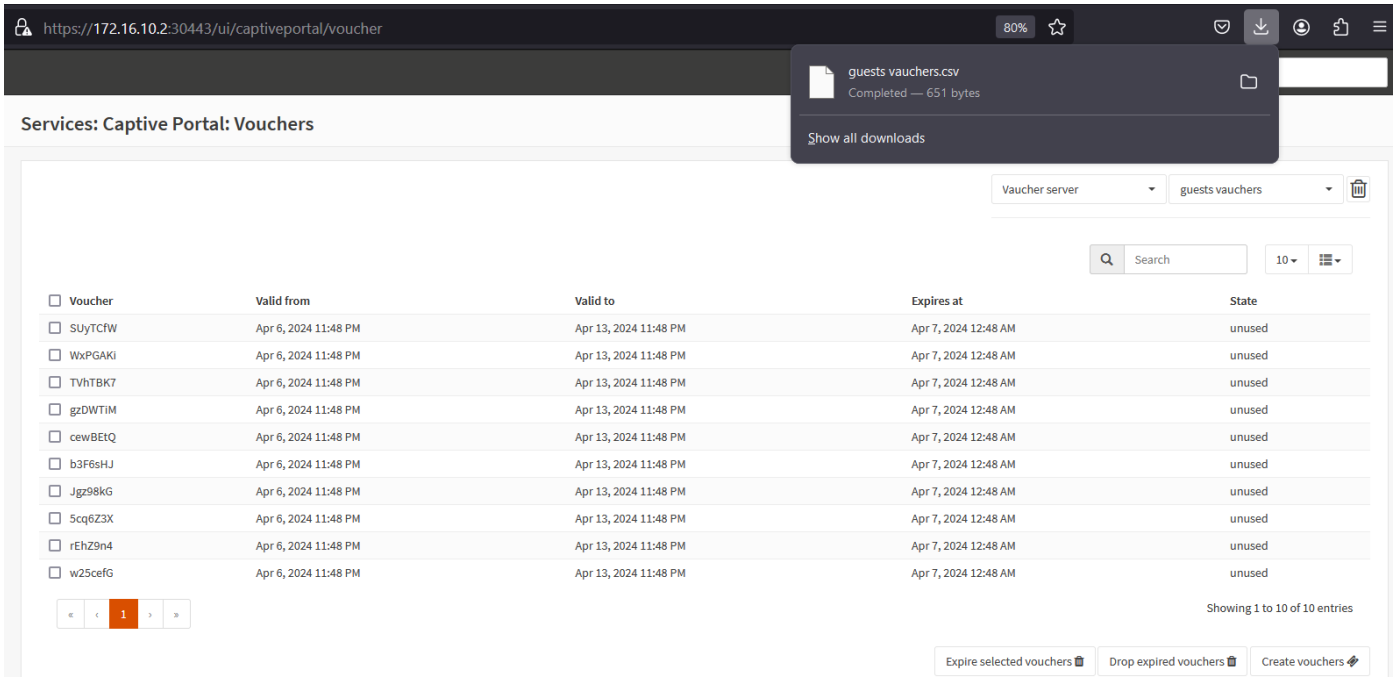


In the **"Generate vouchers"** window, select the options for your need and enter the **Groupname**.

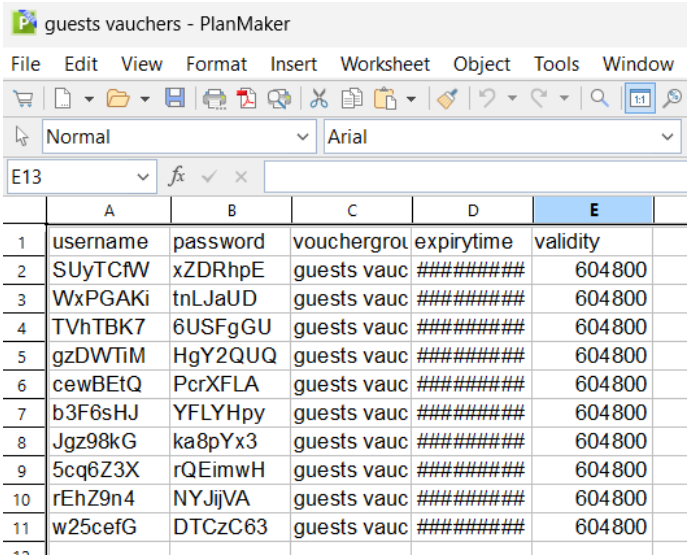


Once you press the **Generate** button, vouchers will be in the **Services => Captive Portal => Vouchers** page

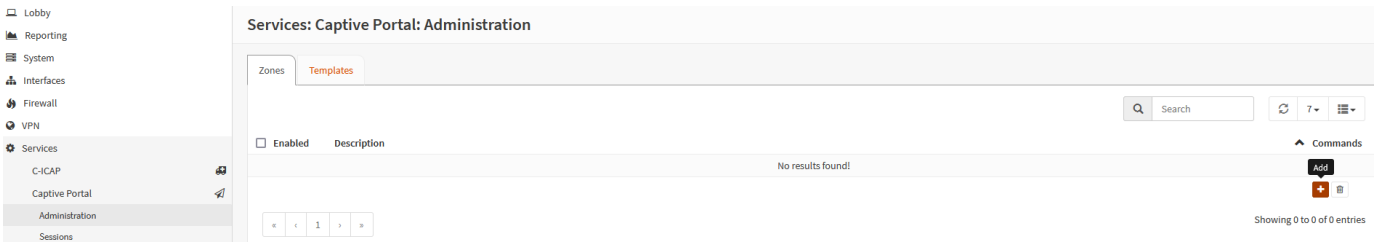
and will be downloaded into your machine in **CSV** file format.



You can see the login credentials, username and password, in this file.

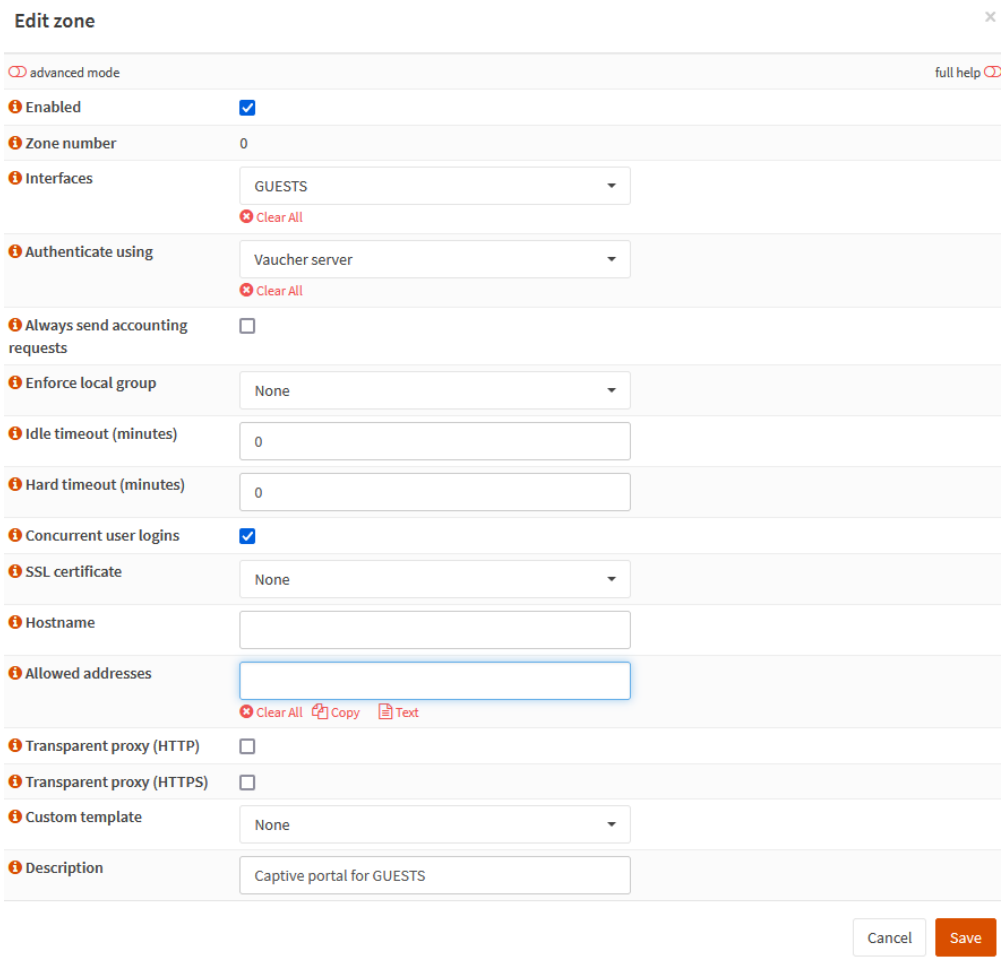


After creating vouchers, go to the **Services => Captive Portal => Administration** page and press add button.

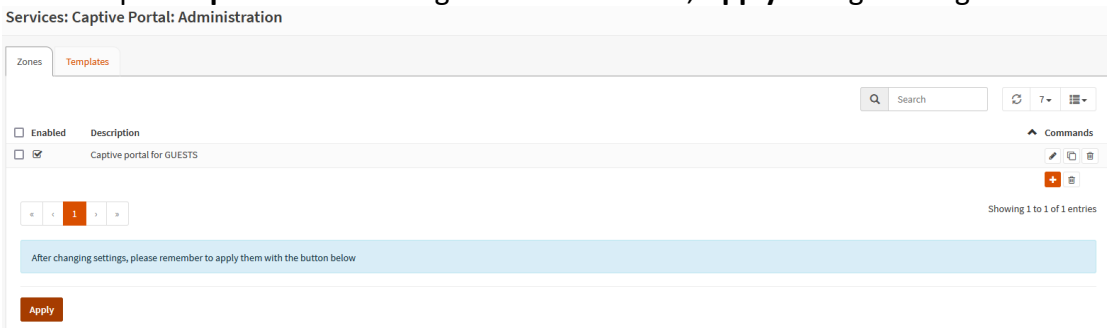


In the **“Edit zone”** window, put the tic in **Enabled** box, choose the interface in **Interfaces**, in **“Authentication**

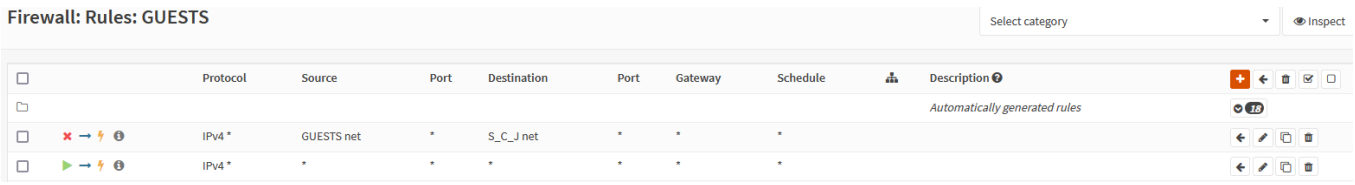
using” section, select the authentication server name which we created, give a description and **Save**.



That was the last step of **Captive Portal** configuration for our lab, **Apply** changes and go to the next setup.

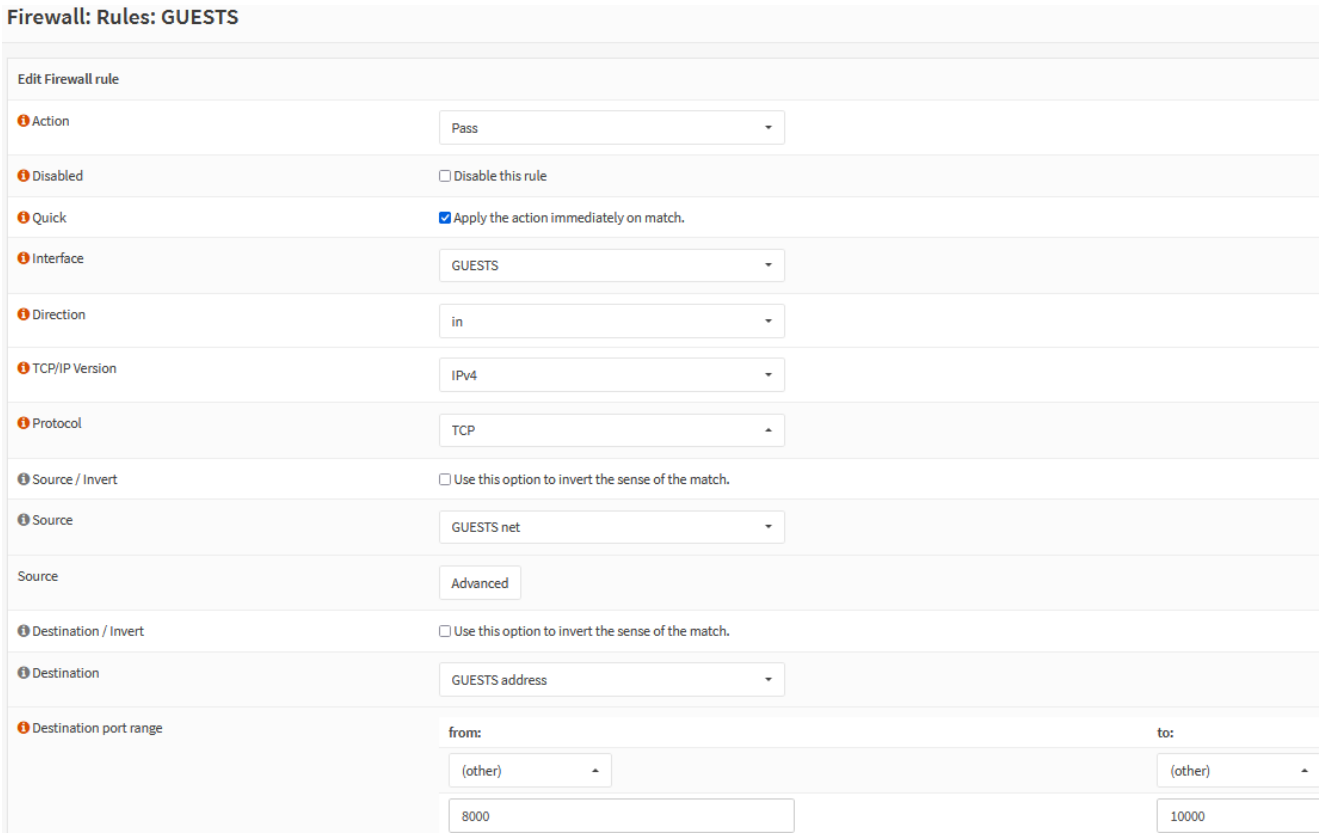


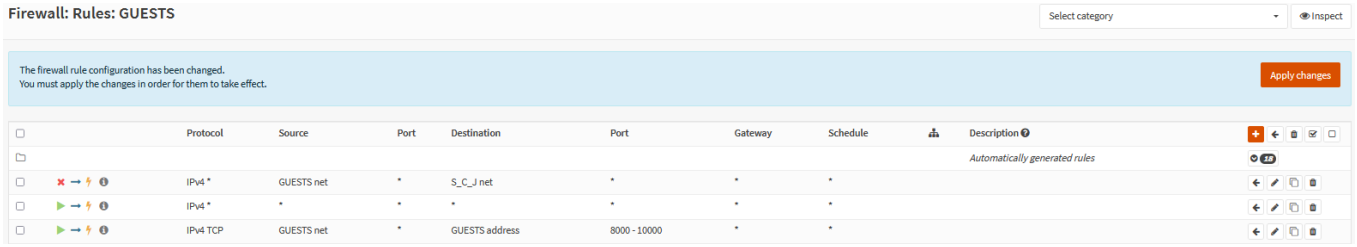
Now we must write new rules for **GUESTS** interface to allow network to access DNS and Captive Portal ports. Go to the **Firewall => Rules => GUESTS** and add new rule.



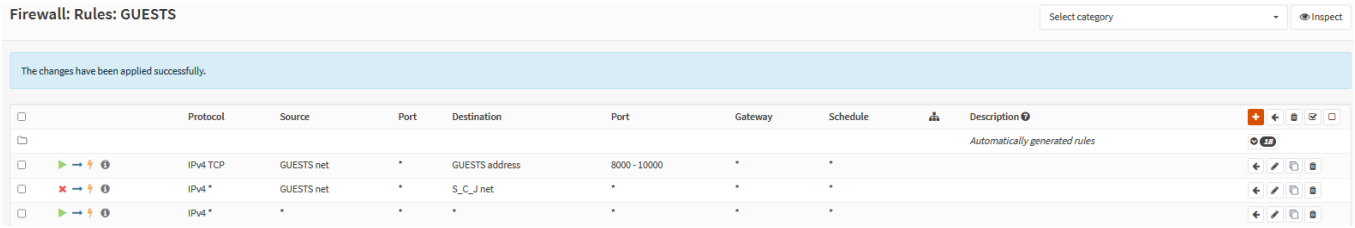
In the rule, select **Pass** for **Action**, select **GUESTS** in Interfaces, select **TCP** as **Protocol**, select **“GUESTS net”**

as **Source**, select **“GUESTS address”** as **Destination**, select **“(other)”** option in **“Destination port range”** and type **from: 8000**, **to:10000** (**Captive Portal** works in this port range in **OPNsense** firewall).



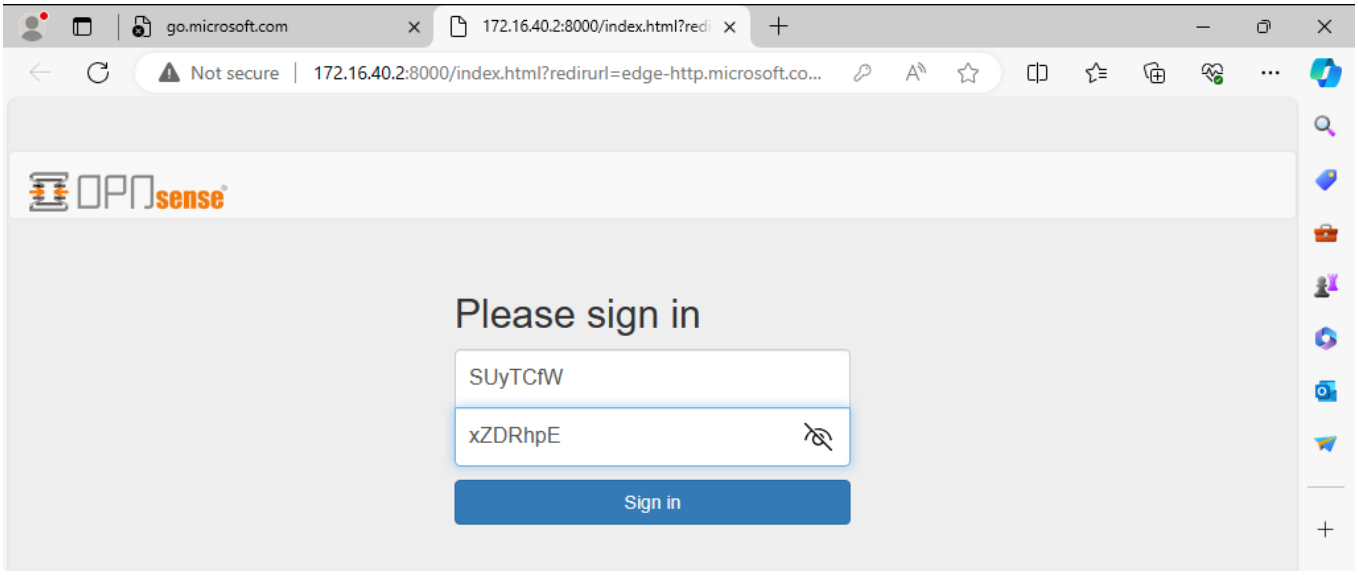


After creatin new rule, arrange all rules as shown in the figure below.

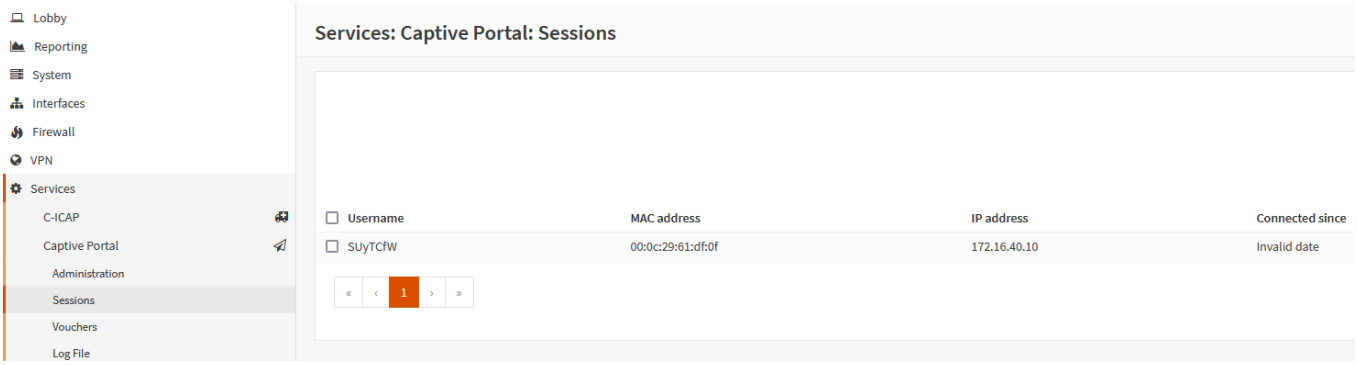


Now it is time to test our setup, go to the guest machine which you connect to the **GUESTS** interface (for our lab, **VMNET4** adapter), then try to go any address, and the default login page will meet you. Enter one of the vouchers in it and **“Sign up”**. Our guest machine could access to the **Internet** until the voucher

expires.



We can observe and finish the sessions in the **Services => Captive Portal => Sessions** section.



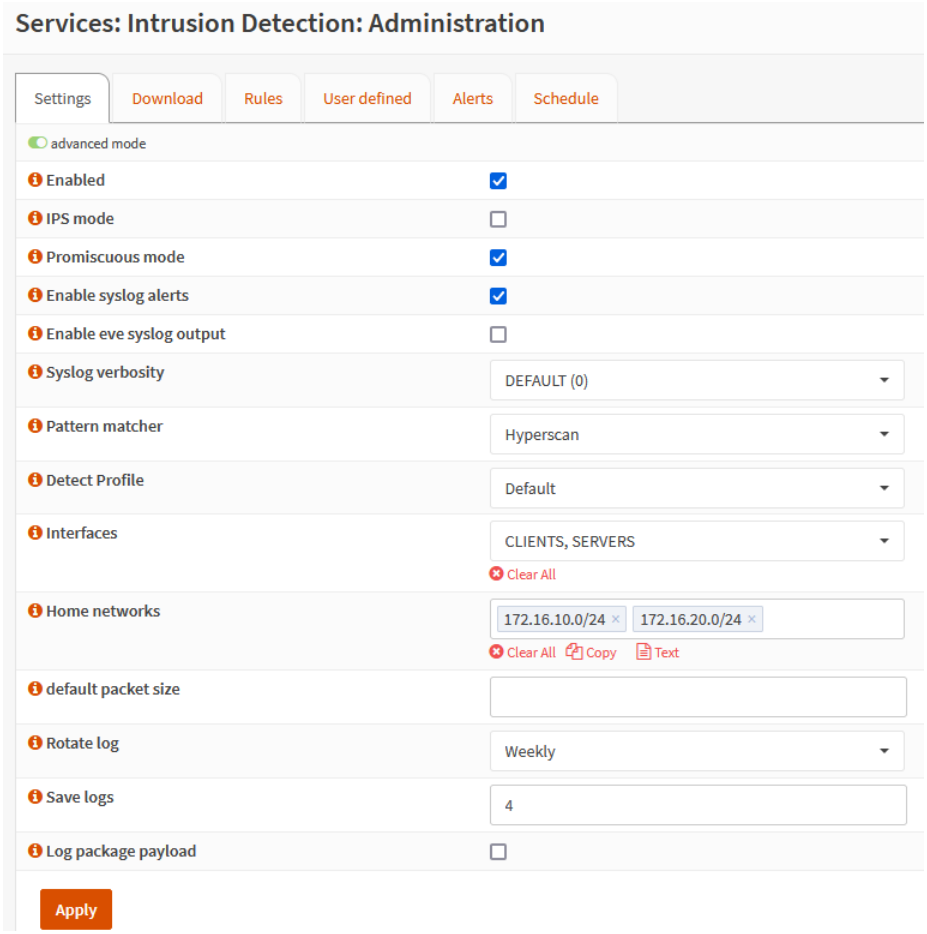
3.3 IDS/IPS configuration

If you analyze the firewall rules and proxies' configurations carefully, you will notice that we didn't fully protect **SERVERS** and **CLIENTS** networks. If bad actors try to scan these networks and machines inside them, they will be successful because our infrastructure is vulnerable to recon tools, such as **Nmap**, **Metasploit** etc. In this capture, we will try to build an extra security layer by configure IDS/IPS feature of **OPNsense** firewall.

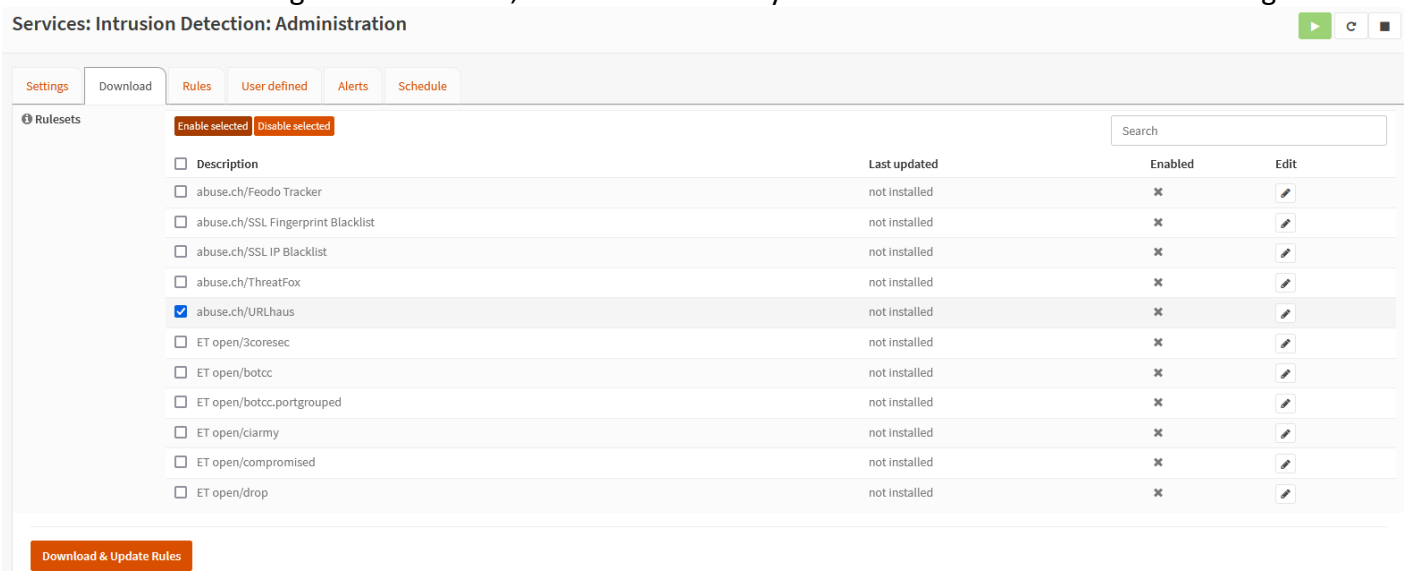
IDS (Intrusion Detection System) listen the network traffic and search for suspicious patters to match with signatures (rules) in its database and gives an alert if they match. **IDS** doesn't intervene any action, but **IPS** (Intrusion Prevention System) can drop the requests if it is suspicious.

In the earliest step, we will configure the **IDS**, go to the **Services => Intrusion Detection => Administration** section and toggle the **"advanced mode"**, then put tics in **Enabled**, **"Promiscuous mode"**, **"Enable syslog alerts"** boxes, select **Hyperscan** in **"Pattern matcher"** section, select **SERVERS** and **CLIENTS** in **Interfaces**, type these interfaces subnet **IP**s in **"Home networks"** section and **Apply**. (you can get detailed information

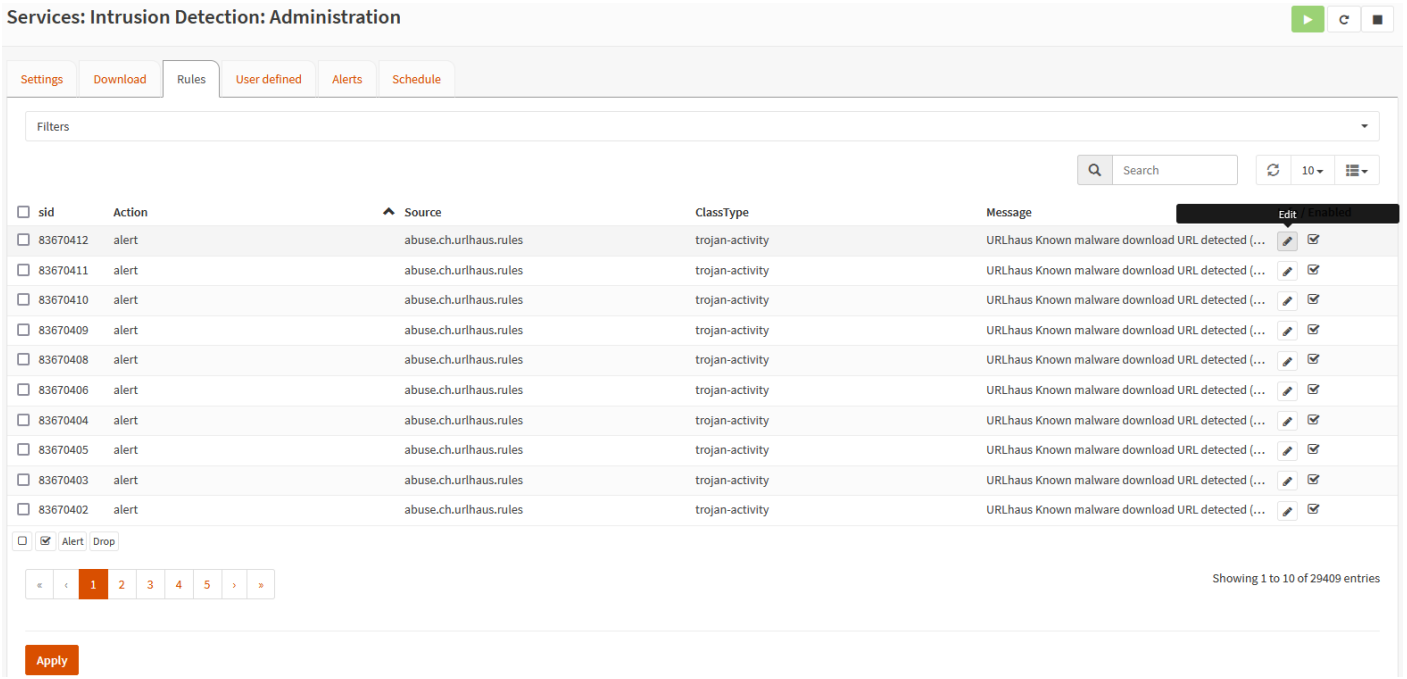
about these options in **OPNsense** documentation in its web page)



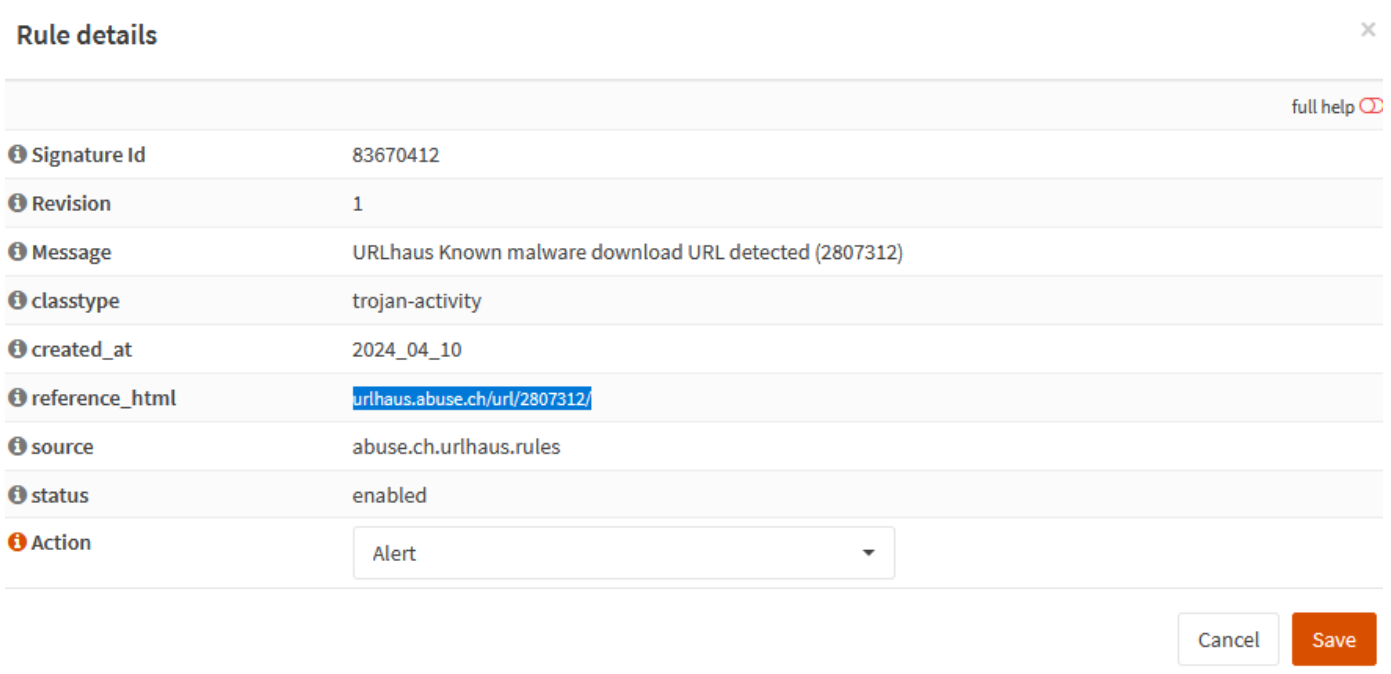
Later go to the **Download** page and put the tic in **"abuse.ch/URLhaus"** option and press **"Enable selected"** button and press **"Download & Update Rules"** button. Each of these option is a **IDS/IPS** database and they contain hundreds of signatures in them, but we will use only one of them for better understanding.



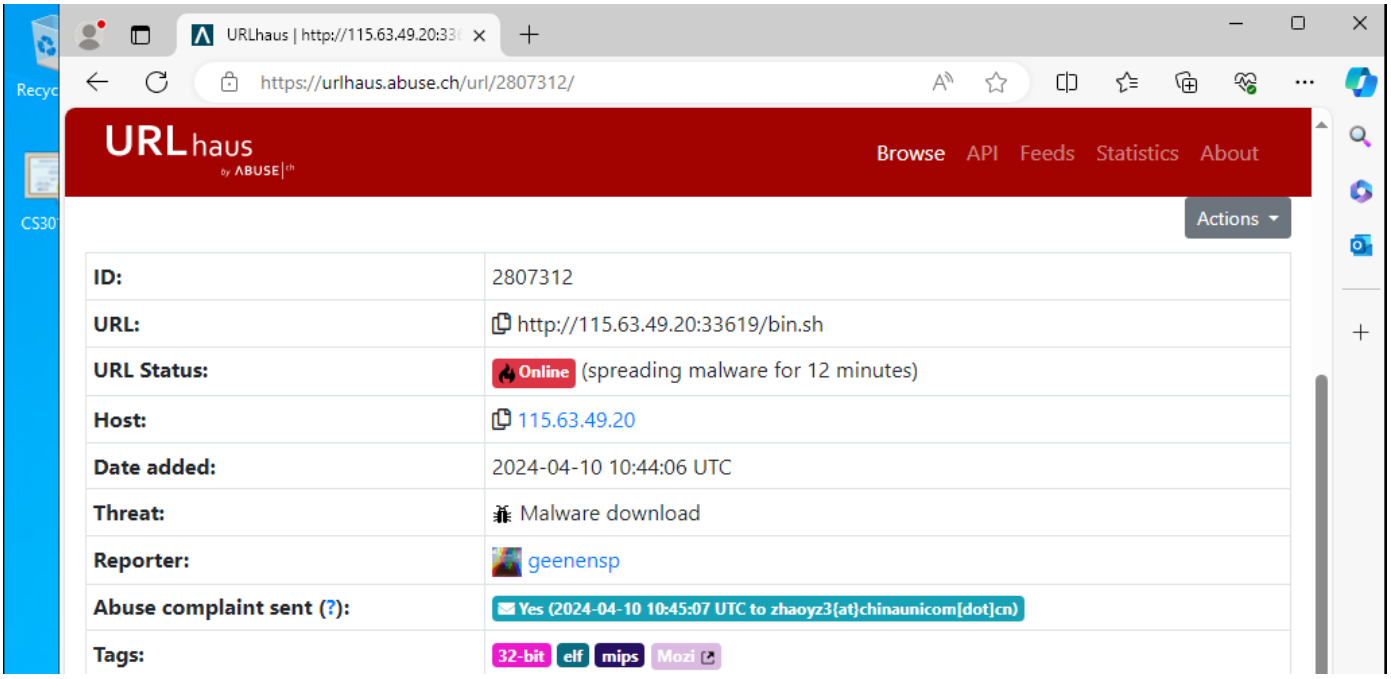
After we successfully download the checked database's rules, they will appear in **Rules** section. In order to test our **IDS**, choose a rule and press **Edit** icon.

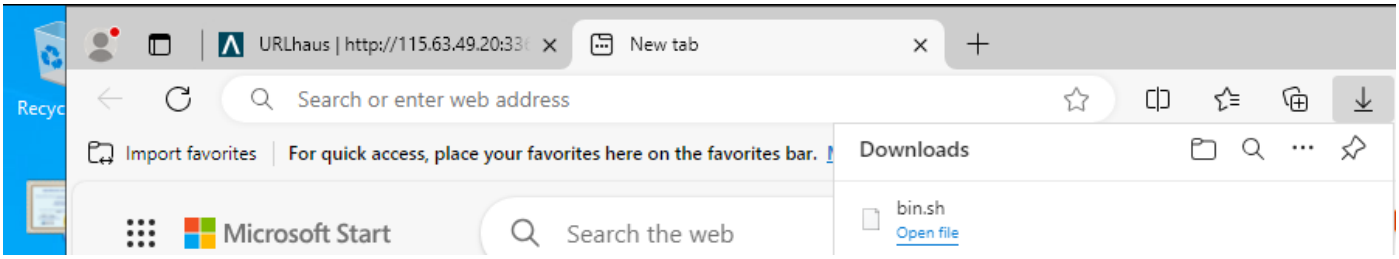


In the popped window, copy the **URL** address in **referance_html** section.

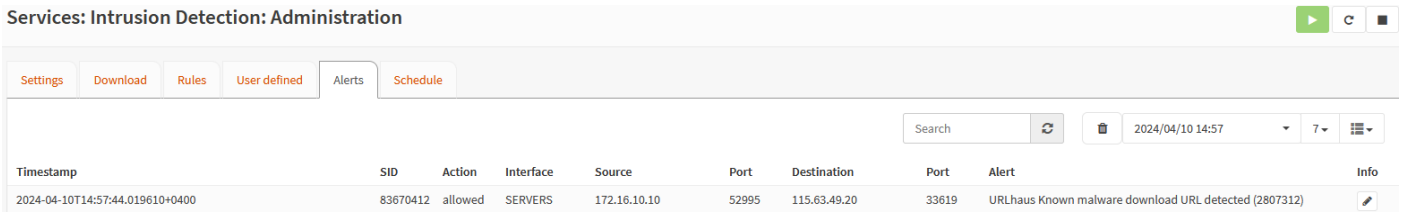


Open your browser in a virtual machine which belongs to one of **SERVERS** or **CLIENTS** networks, then in the search bar, paste and go to that address. In the opened web page, again copy the address in **URL:** section and go to that address. It will download a file whose signature is exist in our **IDS**'s database.

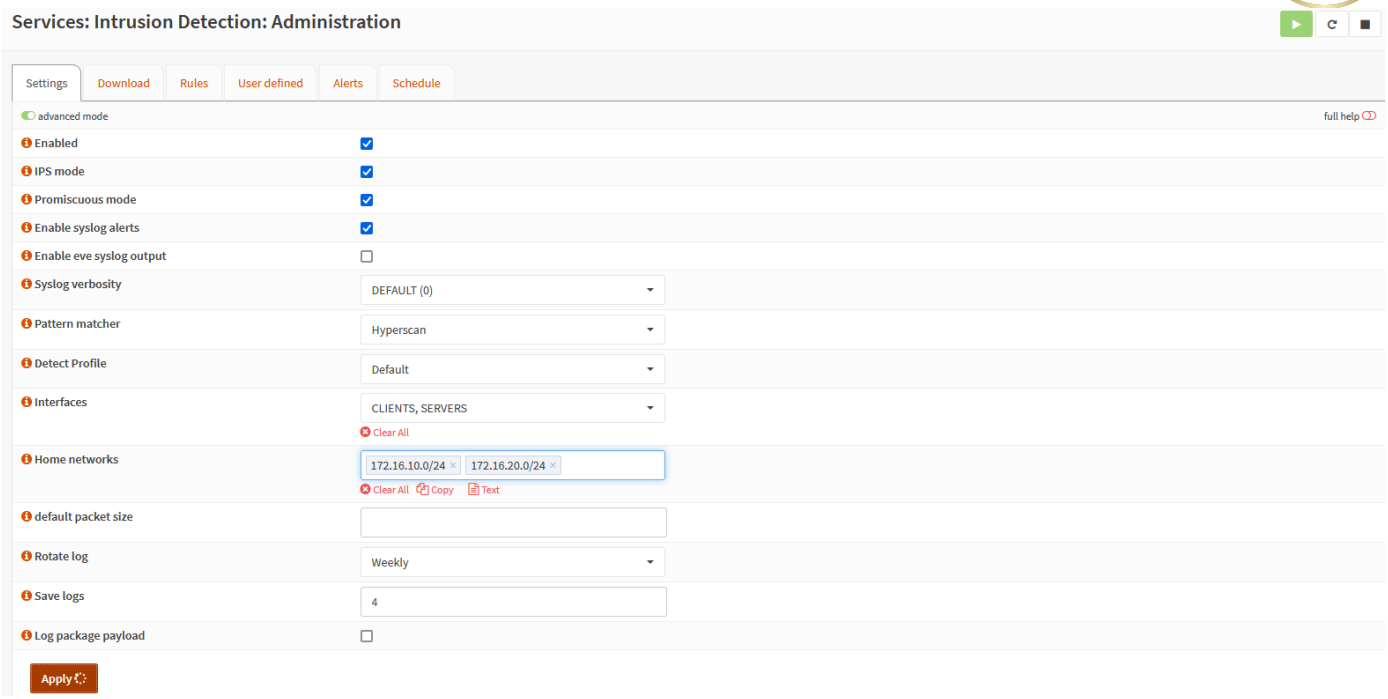




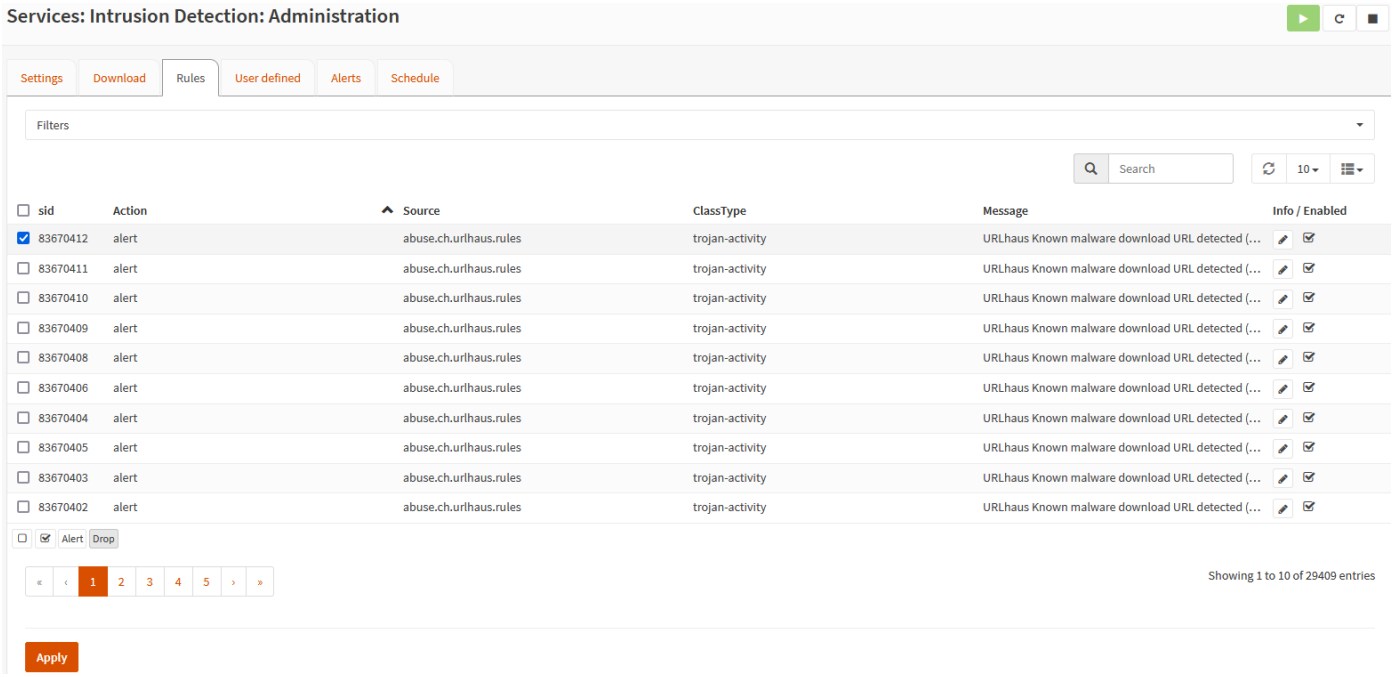
Now back to the **Services => Intrusion Detection => Administration => Alert** section in **OPNsense GUI**, here is our alert. When we downloaded this malicious test file from **URLhause** web page, **IDS** matched this file's patters with its existing rule and gave an alert to us.



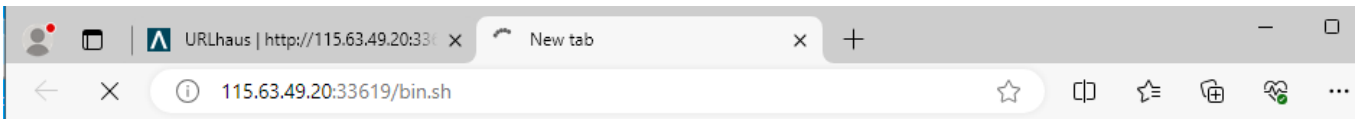
We will try it again but this time IPS will be active. Go to the **Services => Intrusion Detection => Administration => Settings** section and put tic in **"IPS mode"** option and **Apply**.



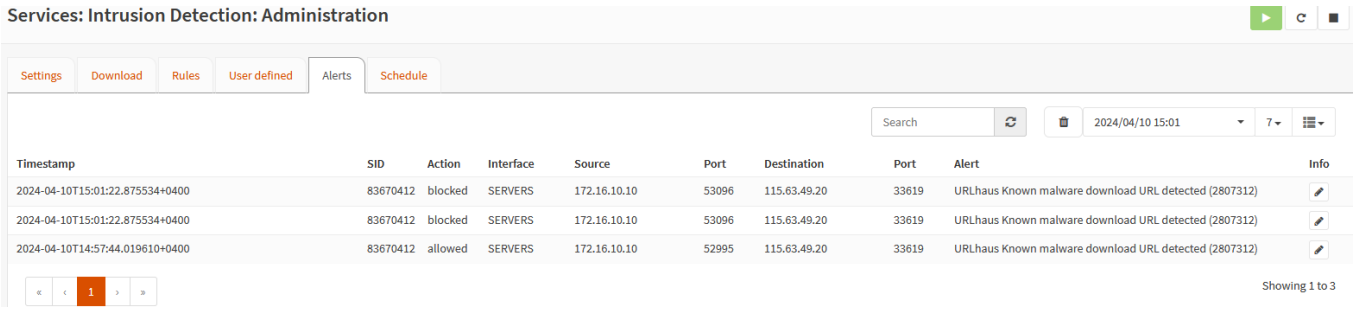
After activating **IPS** mode, go to the Rules section and choose the same rule we checked above and press the **Drop** button (or choose **Action** as **Drop** in **"Rules Details"** window) and **Apply**.



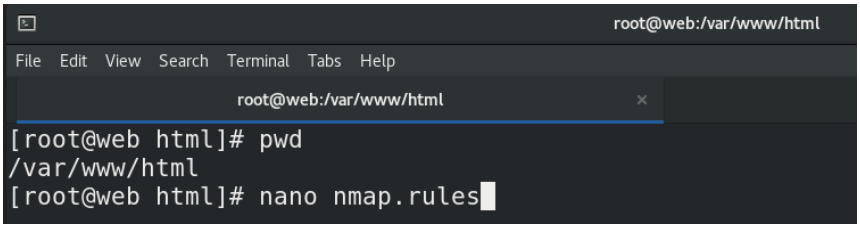
Back to the browser and retry to go the address to download the same file again.



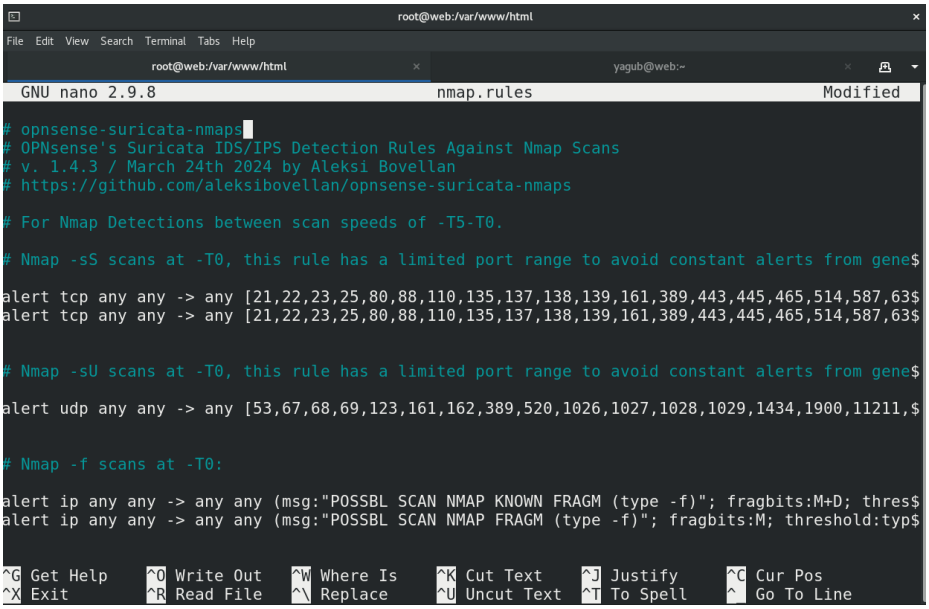
You'll see that the action will be blocked and when you check the **Alert** section there will be new alert with action **blocked**. This is the main difference between **IDS** and **IPS**. **IDS** just detect the suspicious traffic and give alert but **IPS** detect and block it.

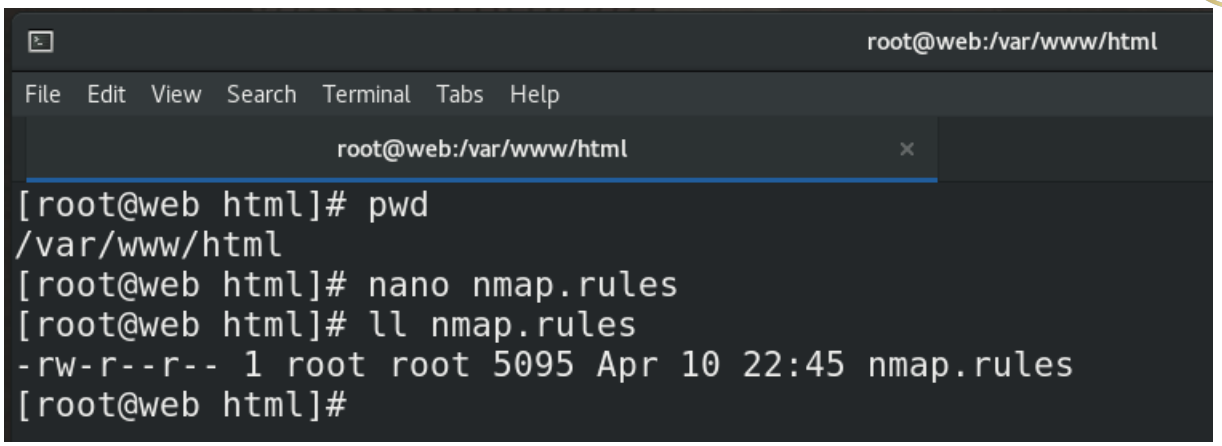


Although **OPNsense IDS** comes with thousands of default rules, you may want to add your own custom rules in it. Until here we did almost every configuration in **GUI** interface but this time we obliged to use the **CLI**. But before we start to configuration in firewall, we need to prepare a database for our custom rules.
As you know we have a **WEB** server in our infrastructure and we deployed the web application (**DVWA**) in it. We will use this **WEB** server as our database in this task. Open your **Oracle Linux** machine and go to the **/var/www/html** path and create new file named **nmap.rules** (or whatever you want for name) in this path.

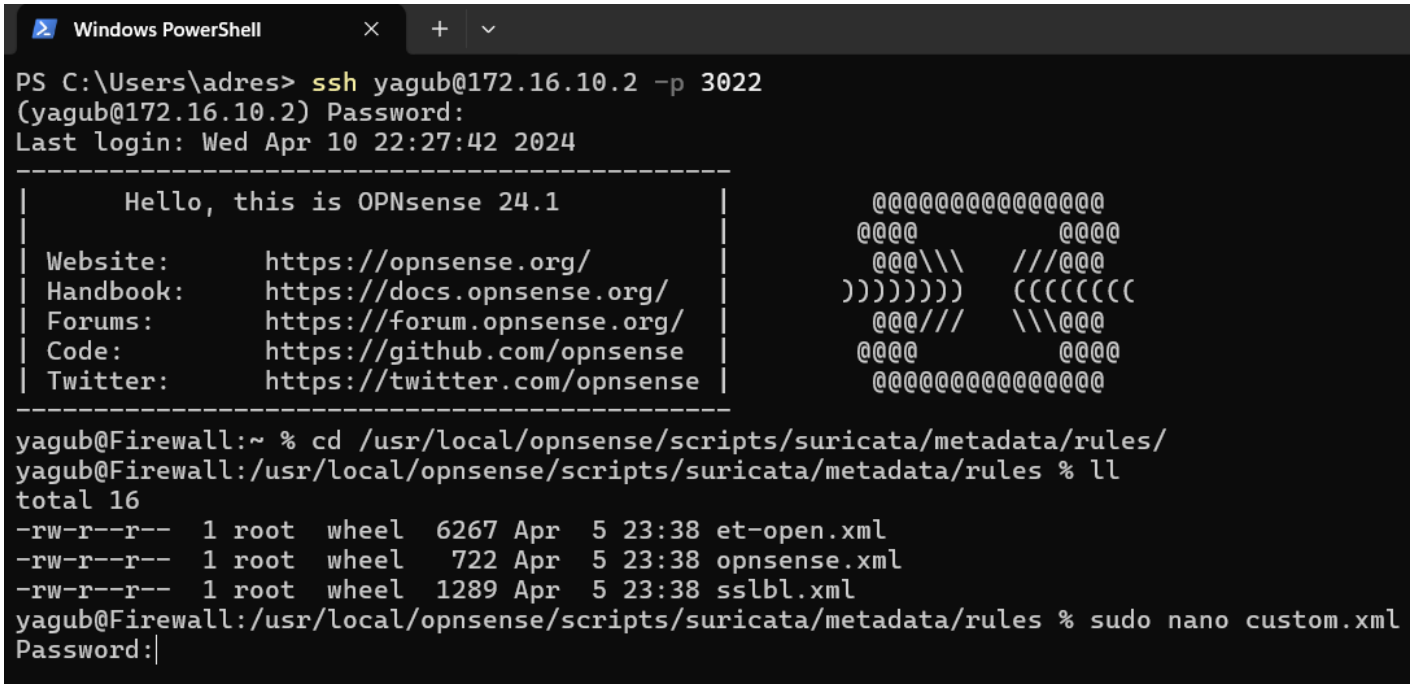


Now we have to add our custom rules in this file. Instead of typing all rules one by one, I used the ready **Github** repository for detecting **Nmap** scanning activities. Open the link below, copy and paste all rules in it to your **nmap.rules** file.
Nmap rules: https://github.com/aleksibovellan/opnsense-suricata-nmaps/blob/main/local.rules

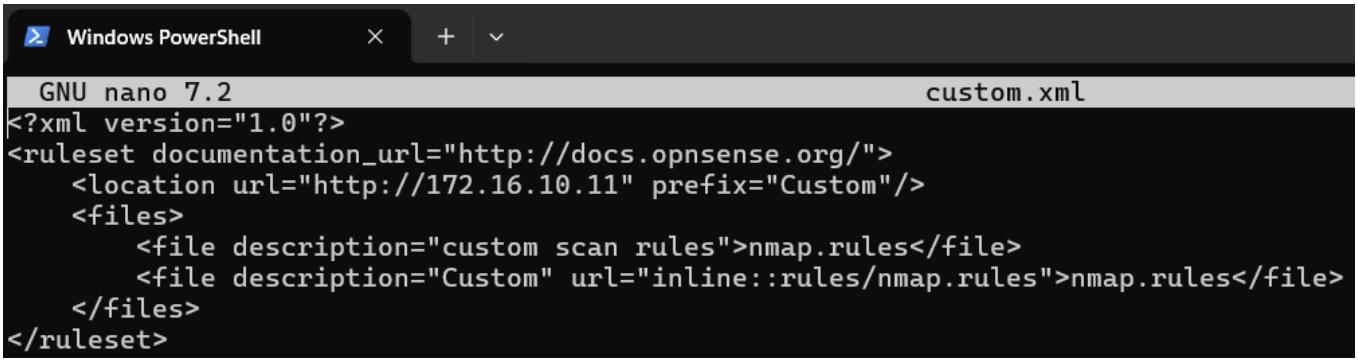


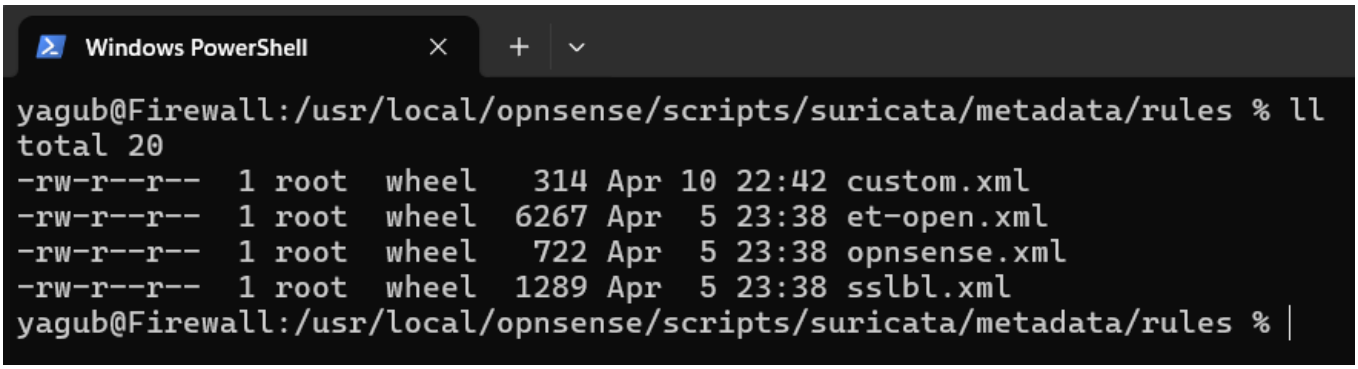


After creating our rules database, we are ready to configure **Suricata IDS** (**OPNsense** use **Suricata** as **IDS/IPS** service as default) to pull rules from this database.
As you remember we enabled **SSH** service for connecting our firewall over the **SERVERS** network and defined port as **3022** for security reason. For easy management, I recommend to connect firewall with **SSH**, how I did in this task (you can use its console too).
Go to the **/usr/local/opnsense/scripts/suricata/metadata/rules** path and create new file named **custom.xml**.

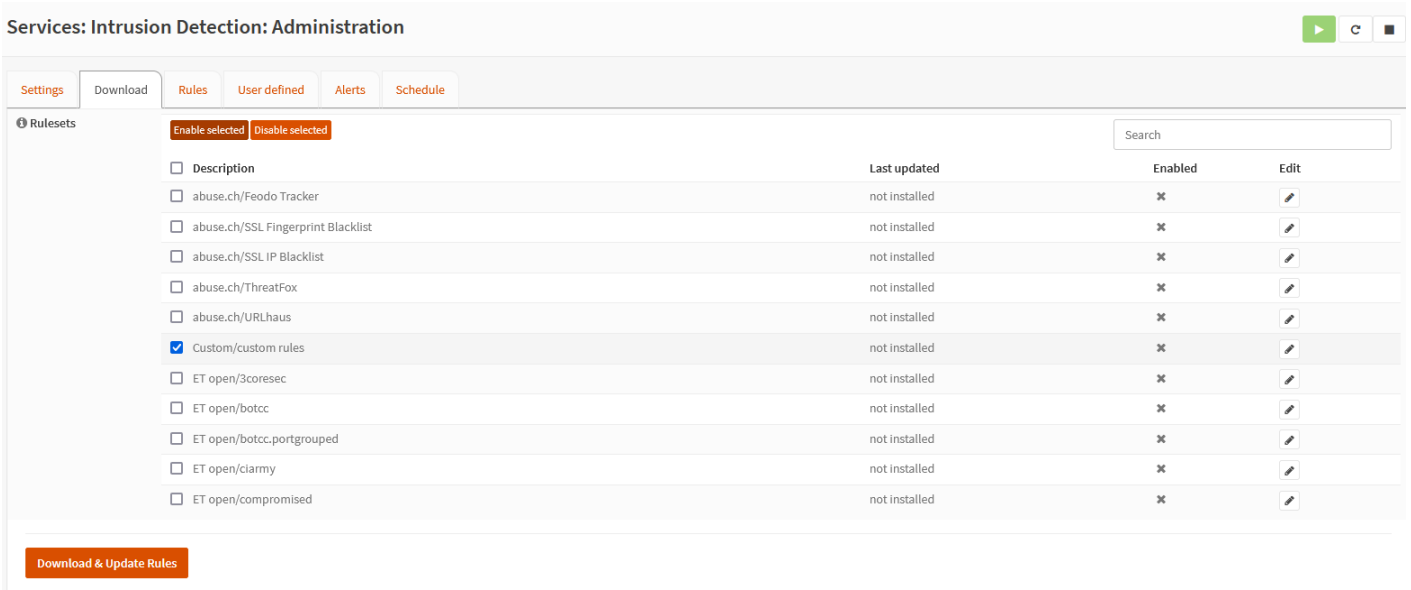


Inside this file we have to define our database's location **URL** (which is our **WEB** server's **IP** address) and its name (**nmap.rules**). You can get the syntax of configuration from this source below.
https://forum.opnsense.org/index.php?topic=7209.0 .

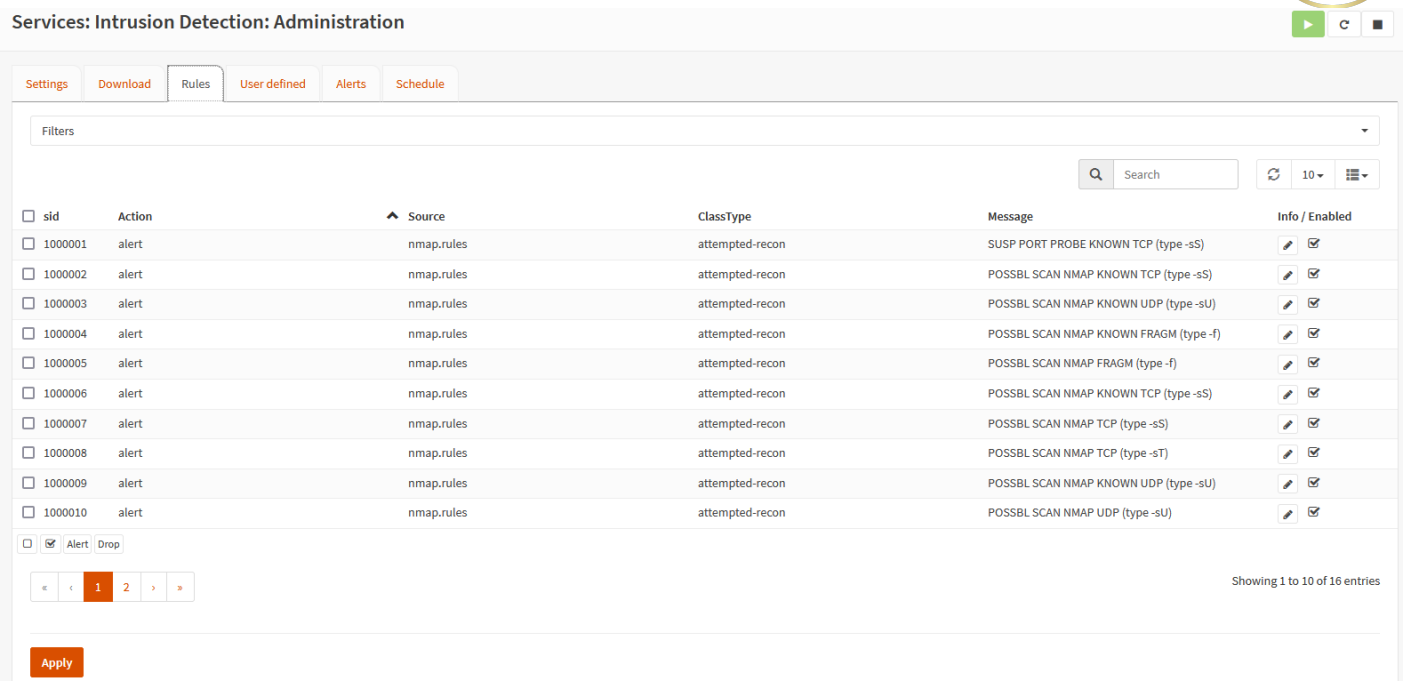




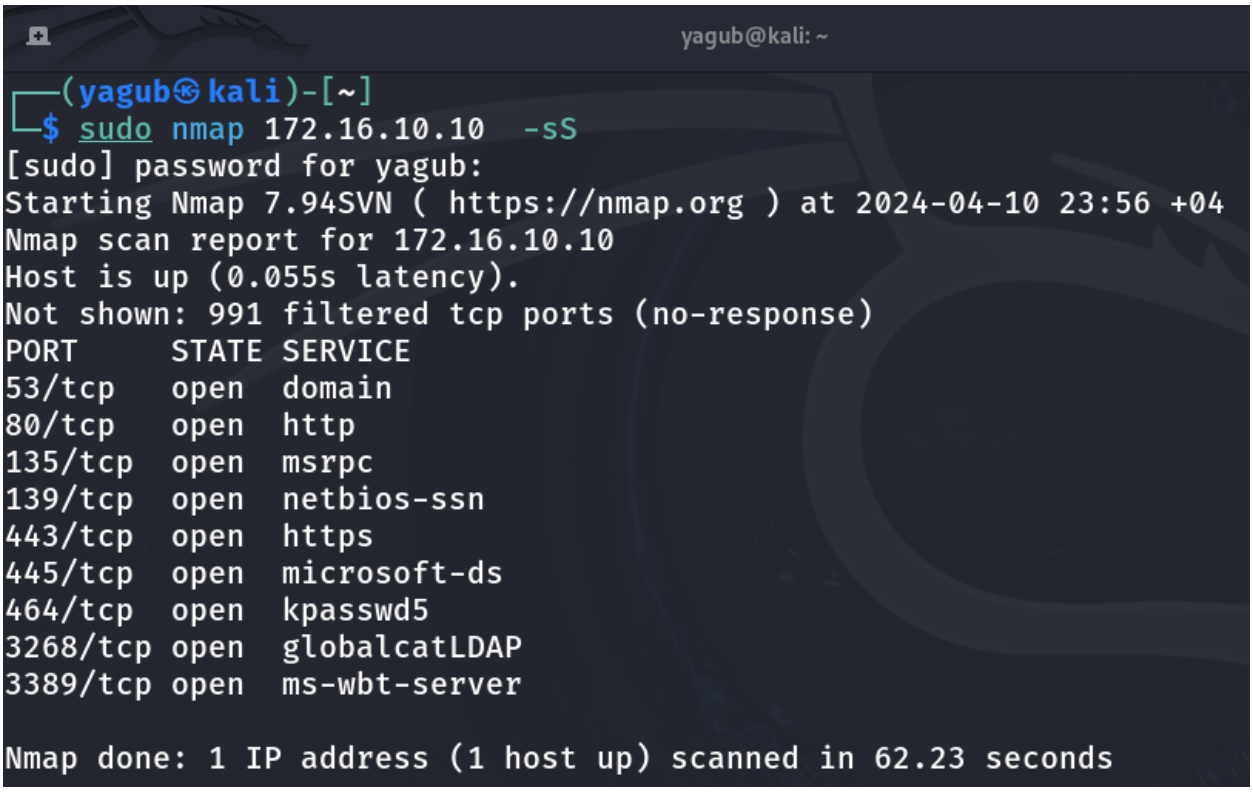
After configuring **Suricata**, back to the **GUI** and restart the **IDS** service. Go to the **Download** section, find your custom rules, enable and download.



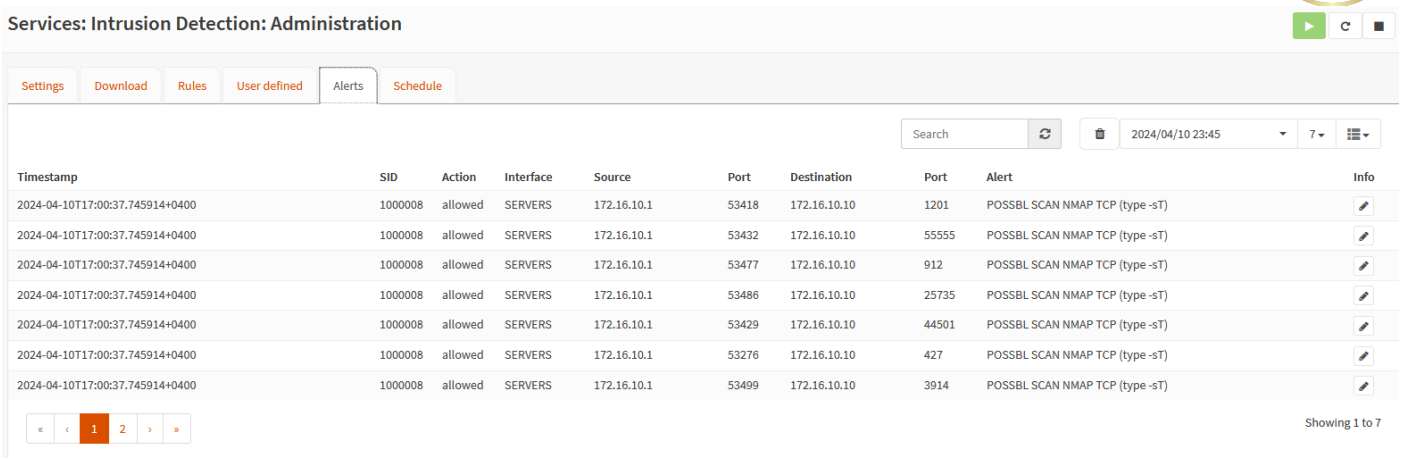
In the **Rules** section you will find all your custom rules which are available.



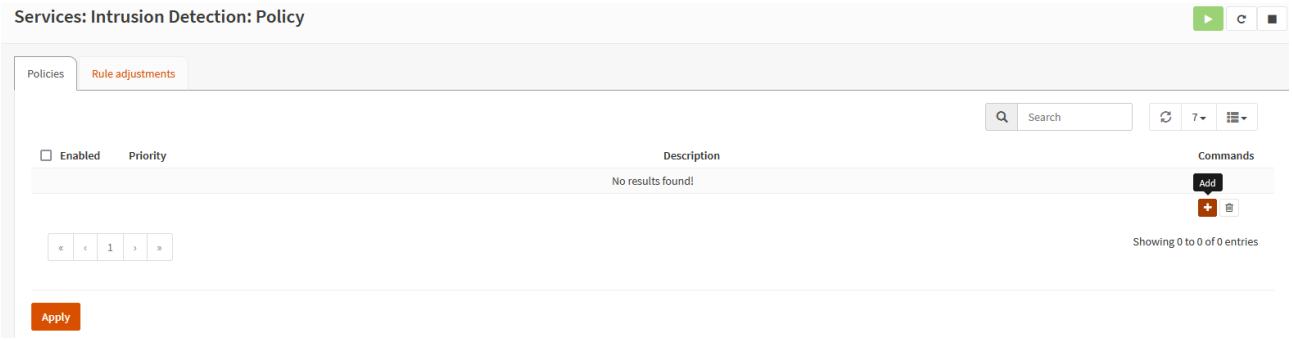
In order to test your rules, go to your **Kali** machine and run a scan on your **DC** (or any other machine in **SERVERS** or **CLIENTS** network).



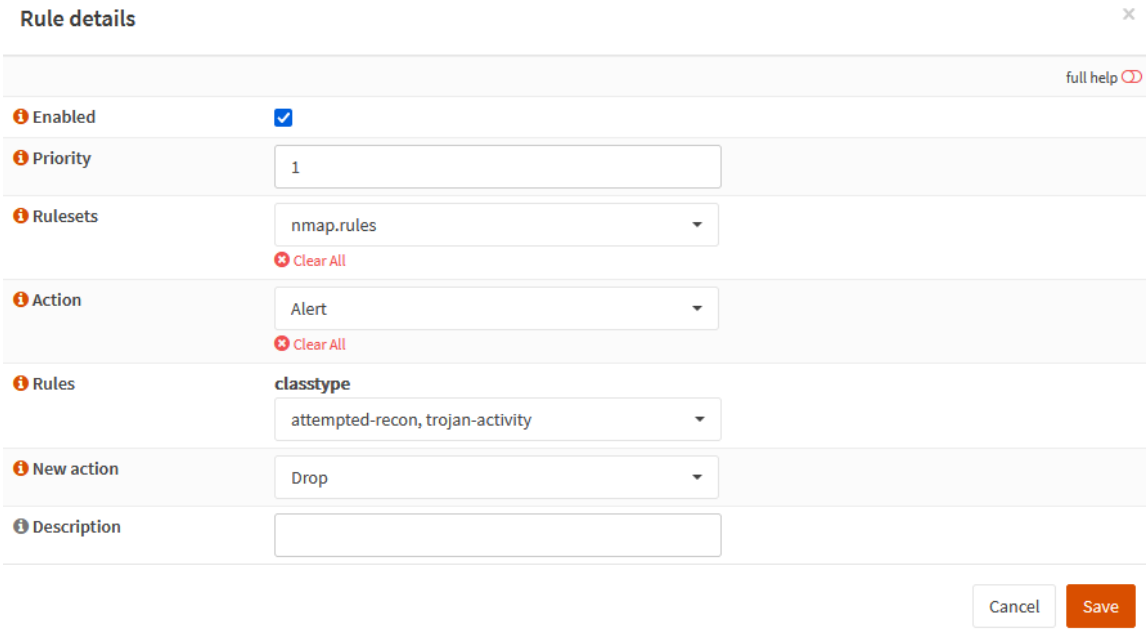
After completing **Nmap** scan, back to the **Alerts** section and you will see the tens of alerts generated by **IDS** and all of them are allowed to pass.



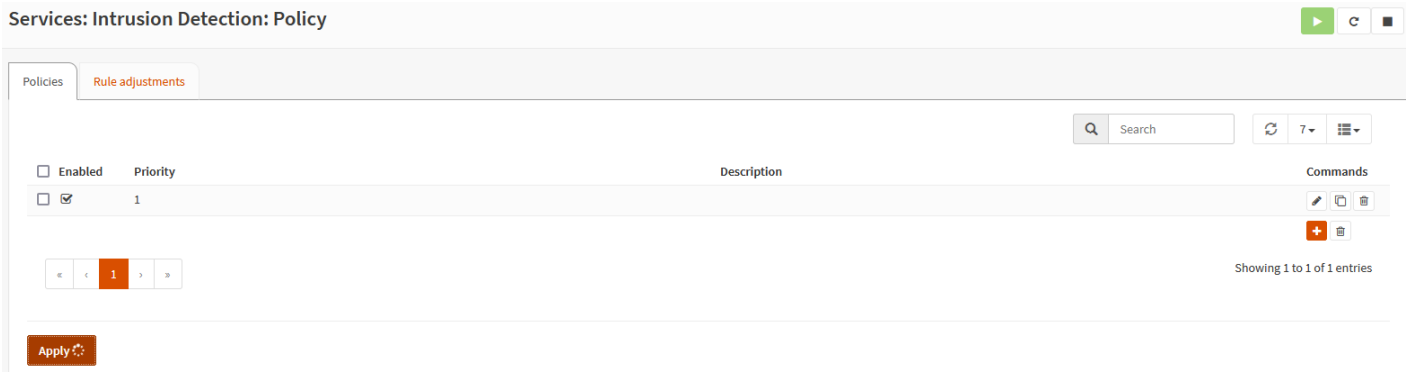
If you remember we enabled **IPS** mode for this service and tested it on **URLhaus**, this time we will test it on **Nmap** scans. All we need to do is to change the rules actions from alert to drop. Instead of doing it one by one manually, **OPNsense** lets us to modify whole database at the same time. In order to do it, go to the **Services => Intrusion Detection => Policy** section and add new policy.



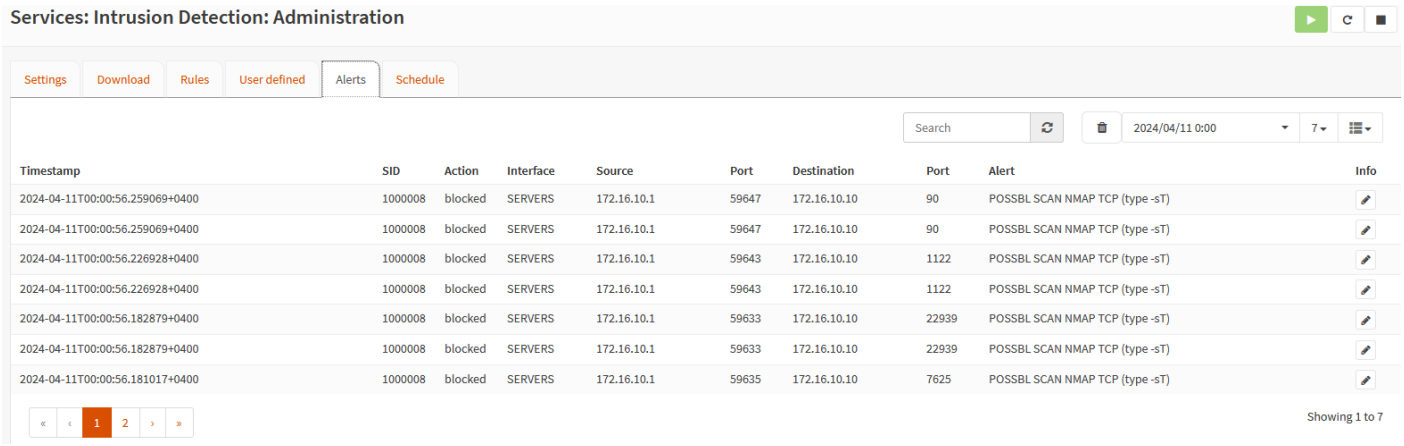
In the popped window, put tic in **Enabled** box, assign a **Priority**, select the custom rule in **Rulesets**, select the rule types in **Rules**, select **Drop** as **"New Action"** and **Save**. This will change all rules' conditions from **allowed** to **blocked**.



After creating new policy, press **Apply** button.



Now back to your **Kali** machine and try to scan again. You will see that new alerts appear in the **Alerts** section, but this time their condition become **blocked**.



NOTE: sometimes **IPS** mode doesn't block actions, this is because we are using virtual environment and open source tool, but the important thing is to understand the logic of **IDS/IPS** and their usage with next generation firewalls.

Conclusion

This lab was designed to understand fundamental security concepts. During the lab we used different methods to protect our infrastructure. Although we spent a lot of effort to make as possible as secure our infrastructure, still there are some weaknesses. Bad virtualization and open source solutions sometimes don't work properly and cause unpredictable errors and bugs. But despite all of these problems, we gained an ability to work with firewalls and design a basic infrastructure.
Now lets analyze our project and give a general summarize what we did:

- We prepared a foundation for our lab in pre-requirements capture
- Installed **OPNsense** firewall, added 5 different networks adapters and configured them
- For the security reason, we created a new user with root privileges and disabled **root** user
- Changed defaults access ports for **HTTPS** and **SSH** access, and defined only **SERVERS** network for firewall access and blocked for all other networks
- Isolated all networks from each other with firewall rules and applied Jump server logic to access our servers (although we didn't mention about this method and complete fully installation and testing it, you can find numerous sources to learn its logic)
- Installed a reverse proxy (**Nginx**) to hide our **WEB** server behind it and applied **WAF** function of **Nginx** plugin for protecting our web application (**DVWA**) against cyberattacks by using **Naxsi** policies
- Installed transparent **Squid Web Proxy** in our firewall and applied web filtering on **SERVERS** and **CLIENTS** networks to protect them from malicious files and web sites
- Installed **Zenarmor** plugin for web filtering **GUESTS** network's web traffic
- Configured **GeoIP** feature of firewall to block internet traffic with some selected countries
- Configured **Captive Portal** to authenticate and control the guests' **Internet** access in **GUESTS** network
- Configured **IDS/IPS** and added our custom rules to protect our **SERVERS** and **CLIENTS** network against suspicious network traffic

I hope this documentation became useful for you. Your feedbacks are precious for me, feel free to contact with me if you have a question or comment about pros and cons of this lab.