# Practical Network Defense

Third Assignment - University "La Sapienza"

**Group 27**: Nicola Bartoloni ******* - Valerio Trenta 1856471

2020-13-06

## 1 Scope and Initial Considerations

The scope of this assignment is to setup a **Virtual Private Network** through **Open-VPN** in **OPNSense** to allow three authenticated *Road Warriors* to access the internal subnetworks of **ACME Co.**, and to grant the usage of a **proxy server** on machine located at **100.100.6.3** - which corresponds to domain **proxy.acme.group27** to the very same authenticated internal clients of the network to reach the **WAN** outside the **Main Firewall**.

Since these tasks are going to allow connections from the outside - even if authenticated and thus probably secured - it is a good idea to make sure, as a first step, that every user on each of the internal machines is protected by a strong password, and we will probably also have to confirm that the SSH protocol on each machine behaves as we expect it to behave - some of the *root* accounts are already disabled by SSH on some machines, but some are still accepting connections based on basic authentication.

The next paragraphs will only deal with the two services we want to setup in the target network - **VPN** and **proxy server**.

## 2 VPN Setup

We can define the **road warriors**, in our case, as a set of three users - *Becca, Huck and Jim* - which we want to be able to authenticate and then connect to the internal subnetworks in the **ACME Co.** target network from the **WAN** interface on the **Main router** via a **Virtual Private Network**.

To this end, in order to setup a **VPN** on the **Main router**, we exploit the **OPNSense** administration panel and then navigate to **VPN** − > **OpenVPN** − > **Servers**, where we can **add a new server** to which the **road warriors** will refer to authenticate and configure their **VPN**. **Figure 1, 2, 3** show the server we created and all the parameters - protocol, port, cryptographic and tunnel settings - that we chose to adopt.



Figure 1: VPN Server's general settings with protocol and port.

Figure 2: VPN Server's crypto settings (1).

Figure 3: VPN Server's crypto settings (2) and IP tunneling address.

Server has been assigned to a **Server Certificate** signed by the authority - the server itself - which can be given to the clients to prove the server's identity, so to perform server-to-client authentication.

Also, under **System** − > **Servers** we setup a new *Acme CO Access Server* which will be exploited to actually access the **VPN Server**, exploiting an access strategy based on a *Local Database + Timebased One Time Password*, with the **OTP** being provided by *Google Authenticator* and thus having length 6. Before actually implementing the **VPN**, it is also necessary to define the authentication system and how the three users will be able to actually access, but this is discussed in the fourth paragraph.

In order to grant the **road warriors** to only access through **SSH** protocol the *internal machines* - i.e., machines on *Internal Servers* and *Clients* subnetworks - the rules at the **Main firewall** had to be changed as follows:

- at **WAN** interface, a **Pass** rule was added to accept every connection from any source on destination port **1194** for procotol **UDP**, so to enable connections with **OpenVPN Server**;

- a new interface, **OpenVPN**, was automatically generated by **OPNSense** once the **OpenVPN Server** was created: here, the firewall allows only connections with source IP address *192.168.0.248/29* (being the one specified in the **IP Tunneling** field of our server) to perform the users' authentication, and then connections with

same source IP address and destination IP address being either in the *Internal Servers* or in the *Clients* subnetworks on destination port 22.

Similar rules had also to be added on the interfaces of the **Internal router** to allow hosts in the two target subnetworks to accept connections with the **VPN**.

# 3 Proxy Setup

We want to setup a **proxy service** at **proxy.acme.group27** that can be exploited only by *authenticated* users located in the *Clients* subnetwork. For the sake of simplicity, the users we will accept and authenticate are the same we have defined previously for the **road warriors** set.

The setup is carried out entirely in the **proxy.acme.group27** machine, and even if this machine gives us the chance to setup an *HTTP Squid Proxy service* through **zentyal** administration panel, we decide to setup **Squid** directly via the *squid.conf* file as pictured in **Figure 4** (next page).

We decide to implement a **non-transparent proxy** given the requirements for the service, so that the users in the *Clients* subnet will have to manually configure their browsers. The service can be accessed, as anticipated in the previous assignment, on port **3128** via **HTTP**. Apart from defining several default **acl**s for different safe ports, we define an **acl** named **Clients_net** to identify the target *Clients* subnetwork, and an **acl** named **authenticated** which identifies users that have been successfully **authenticated** through module **basic_ncsa_auth** (**Figure 5**).

Notice that every user-password pair is stored under the file */etc/squid/passwd*, this and the whole authentication procedure will be discussed on the next paragraph.

At this point, with line *http_access allow Clients_net authenticated* we grant access to users that are in the desired internal subnet and that have been successfully authenticated. Every other connection is denied through default line *http_access deny all*, and reply access is allowed if not denied before as default.

As pictured in the detail on **Figure 6**, as requested by the assignment the **proxy service** holds a **cache** and **log file** for it, as well as an **access.log** file to log all the accesses and operations performed by the authenticated users.

The firewall rules were already setup so that this service could be available from the *Clients* subnetwork and so that its machine had Internet access. So, in order to visit one of white-listed websites - **cybersecurity.uniroma1.it** which has IP address **151.100.17.12** - we only need to start the daemon of the **Squid** service via command *systemctl start squid* after having it enabled, and then setup the *Firefox Browser* on the Kali machine so that it knows how to contact the service: *Firefox − > Preferences − > Network Settings − > Settings − > Manual proxy configuration* with the very well known values (100.100.6.3 and 3128 as port) to be exploited with all protocols.
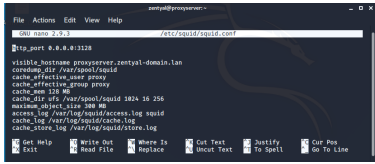
Results are showed in the sixth paragraph.

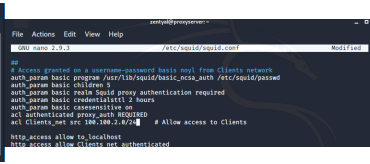Figure 4: Beginning of the squid.conf file with the 3128 port specified.

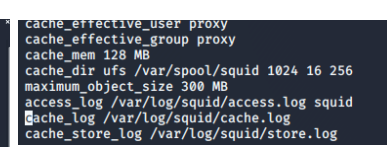Figure 5: acls defined to restrict the access to the service.

Figure 6: Configuring log files for cache and accesses to the service.

# 4 Authentication

Both services we implemented in this assignment require a *basic authentication* in order to be accessed. Users to be authenticated are the three we already mentioned in the previous paragraphs, belonging to the **road warriors** set: *Becca, Huck and Jim.* Each of them has two unique passwords, different from each other, one for the **VPN service** and one for the **Proxy service**.

## 4.1 VPN Authentication

Access rules for the **VPN service** have been defined on the **OPNSense** administration panel located at the **Main router**, under the **System** tab. The three users have been assigned to the **ROADWARRIORS** group and, for each of them, an **RSA Public Key** has been created and assigned for **TLS client-to-server Authentication** process of **OpenVPN**. Moreover, each of the three users holds a unique password and a **One Time Password** - a six-characters token provided by **Google Authenticator**.
This way, in order for one of the users to authenticate to the **OpenVPN service**, it must necessarily own *three requirements*:

- its own unique **.ovpn client exported file** located under the tab **OpenVPN** − > **Clien Export**, which must be exploited by the client machine's **openvpn** tool to retrieve the server's location, port, parameters and perform the **TLS Authentication**;

- its own unique and static **password** defined under its own profile on the **OPNSense** administration panel;

- its own **OTP** provided by **Google Authenticator**

when these three requirements are met, the user can easily access to the **OpenVPN** service and reach the internal machines as desired.

## 4.2 Proxy Authentication

As already mentioned, the **Proxy Service** is only reachable when the client's IP address is found to belong to the **Clients subnetwork** (*100.100.2.0/24*).

Moreover, we perform once again **basic authentication** so that only internal clients owning a *username-password* pair that is known to the service can actually authenticate and exploit it. Once again, the three authorized users to authenticate to the service are *Becca, Huck and Jim*. Each of them holds a unique password, different from the ones we defined for the **OpenVPN service**, whose *hash* has been stored through the **htpasswd** tool under the file */etc/squid/passwd* located at **proxy.acme.group27**, so that the passwords are not stored in clear.

This way, each time an user from the *internal Clients subnetwork* accesses a web browser having the **Proxy service** set, a dialog pops up requesting username and password to authenticate: if the authentication fails, the service won't be reached and the user will not be able to exploit it and thus surf the Internet.

Details about the basic authentication exploited in the **Squid** service were also given in the previous paragraph: passwords are *case sensitive* and the service only remembers an authenticated user for *2 hours* if he doesn't disconnect from the service itself. After 2 hours, the service will require the user once again to provide their credentials.

# 5 Tests

## 5.1 Testing the VPN

## 5.2 Testing the Proxy service

# 6  Final remarks