



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Initial Process of Updating the Core of a Network

Proyecto Final de Carrera

Ingeniería Informática

Autor: Marcos Tortajada Rodríguez

Director: Juan Vicente Capella Hernández

Septiembre 2013

Initial Process of Updating the Core of a Network



Abstract

This report concerns my internship in the MI2S between the months of March and July of 2013, being employed by the Centre National de Recherche Scientifique. The CNRS is the most important public organization for scientific research, its aim is to coordinate the activities of the laboratories to make the scientific research more efficient.

As a part of my course of Computer Networks and Telecommunications specializing in Wireless Networks and Security (a French professional bachelor's degree) and my Erasmus student exchange programme, I finished my studies in Grenoble with this work experience in this public organization.

In this report I will explain how I developed my job to achieve the goal of my main project, the update of the core of the network and other tasks related with this one. This will include the hardware and software management of all the devices for this process and also a research of the appliances and improvements for the future network. Finally, I will conclude with a summary of my contribution to this project and a reflection on my work during the internship.

Key words: Cisco Catalyst 6506, firewall, Rancid, high availability, HSRP, VLAN.



Abstract

Ce rapport concerne mon stage dans les MI2S entre les mois de Mars et Juillet de 2013, étant employé par le Centre National de Recherche Scientifique. Le CNRS est l'organisme public le plus important pour la recherche scientifique, son objectif est coordonner l'activité des laboratoires en vue de tirer un rendement plus élevé de la recherche scientifique.

Dans le cadre de mon cours de Réseaux et Télécommunications spécialité en Réseaux Sans Fils et Sécurité (titulaire d'un Licence professionnelle) et mon programme d'échange d'étudiants Erasmus, j'ai terminé mes études à Grenoble avec cette expérience de travail à cet organisme publique.

Dans ce rapport, je vais expliquer comment j'ai développé mon travail pour atteindre l'objectif de mon projet principal, la mise à jour du noyau du réseau et d'autres tâches en rapport avec celui-ci. Il s'agira notamment de la gestion matérielle et logicielle de tous les dispositifs de ce processus et aussi une recherche des appareils et des améliorations pour le futur réseau. Enfin, je vais conclure avec un résumé de ma contribution à ce projet et une réflexion sur mon travail au cours du stage.

Resumen

Este informe trata de mis prácticas en el MI2S entre los meses de marzo y julio de 2013, siendo empleado por el Centre National de Recherche Scientifique de Francia. El CNRS es una de las organizaciones públicas para la investigación científica más importantes, su objetivo es coordinar las actividades de los laboratorios para hacer sus investigaciones más eficientes.

Como parte de mi curso de Redes y Telecomunicaciones especializado en Conexiones Inalámbricas y Seguridad (una licenciatura francesa) y mi programa de intercambio Erasmus, finalizo mis estudios en Grenoble con esta experiencia de trabajo en dicha organización pública.

En este informe explicaré como desarrollé mi trabajo para cumplir con la meta de mi proyecto principal, la actualización del núcleo de la red y otras tareas relacionadas con ésta. Esto incluirá la gestión del hardware y software de todos los dispositivos para este proceso y también la investigación sobre algunos nuevos y las mejoras para la futura red. Finalmente, concluiré con un sumario de mi contribución a dicho proyecto y una reflexión sobre mi trabajo durante las prácticas.

Table of contents

1.	Introduction	10
2.	Presentation of the MI2S.....	12
2.1.	Moyens Informatiques et Multimedia (MIM).....	12
2.2.	Information Scientifique (IS)	12
2.3.	Units served.....	12
2.4.	Organization	13
3.	Initial work environment	14
3.1.	General topology	14
3.2.	The chassis.....	14
3.3.	Internal topology the chassis.....	16
4.	Migration of VLANs.....	19
4.1.	Move the rules	19
4.2.	Prepare the files.....	21
4.3.	The migration	23
4.4.	Conclusion	23
5.	Rancid.....	25
5.1.	Brief explanation about Rancid.....	27
5.2.	Testing Rancid.....	27
5.2.1.	In the virtual machine	28
5.2.2.	Team meeting	35
5.3.	Introduce Rancid in the production system.....	36
5.3.1.	Configure the remote SVN	37
5.3.2.	Repositories on the company website	39
5.4.	Conclusion	41
6.	Replacement of the chassis	43
6.1.	Relocating the components	43



6.2.	Incompatibility of the transceivers.....	45
6.3.	Before the hardware replacement	46
6.4.	The replacement	47
6.5.	The day after	48
7.	Future	49
7.1.	High availability	49
7.1.1.	Early study	49
7.1.2.	Deeper in HSRP	53
7.2.	New security appliance.....	57
7.2.1.	Different possibilities	57
7.2.2.	Conclusion	58
8.	Conclusion	59
9.	Bibliography	62
10.	Glossary	65
11.	Appendix 1. Script: Check the migration files	67

1. Introduction

My project here is going to be focused on the update of the network core, initially a chassis Cisco Catalyst 6506, which includes the main router for some laboratories of the campus Saint-Martin-d'Hères and several modules between which there is a Firewall Service Module (FWSM).

Some components of the chassis are arriving to the date of end of support, being already the chassis itself and the fan rack out of warranty and without a maintenance contract. In front of this situation, the MI2S have decided to replace it by at least one Cat6506-E that will increase the throughput and scalability in the network too. Also, it's under consideration to purchase another Cat6506-E to configure both in a high availability mode; in this way, if one stops working or loose the connection, the other takes its tasks without any cut in the network. This duplicity is required in active/active mode, so there is a load balance between both routers.

To that aim, I'm in charge of a study of the different possibilities and protocols (HSRP, VRRP, GLBP, VSS) that allow us to reach this system, initially using the current Cat6506 together with the new Cat6506-E and, afterwards, dismissing the Cat6506 and working with two chassis of the new model.

As I already mentioned, facing the purchase of a chassis, that includes the supervisor card in charge of the routing task (we will refer to this card as a router), some service policies and the carrying out of several protocols like the Spanning Tree Protocol, etc., it is necessary to choose the new security hardware that will control the security as our current FWSM does. We have different options:

- A module integrated in the chassis.

- An external CISCO device.

- An external device of other brand (for instance JUNIPER).

In this respect, I will do an evaluation of the different options with which I will keep in mind features like the throughput, security options, price, organization needs, compatibility, etcetera.

Nevertheless, before all this, we should think in how to migrate all our system and configuration of the current devices to the new ones. Nowadays, most of the networks (VLANs) are directly connected to our router (Cat6056 supervisor card), without passing through the FWSM, so their security is managed by the access control lists (ACL) in the supervisor card. It is necessary to make a migration of the mentioned virtual LANs and their filters from the router to the FWSM.

The firewall module provides an improved management due to the distinct characteristics, like more advanced service policy rules, anti-spoofing, logs and customized graphs and the possibility to use a graphical interface (ASDM – Cisco Adaptive Security Device Manager) that speeds up the rules and network objects management.

The said interface transference is necessary within a short time due to the reasons previously mentioned. Thus, taking into account the future substitution of the current router and firewall, in case to choose a Cisco product to carry out the functionalities of the firewall, the migration of the configuration of the different interfaces and filters will be much faster to do it from the FWSM than to the router and its ACLs.

During this process, I will make use of several tools and protocols that will help me to monitor the state of the different network devices.

Therefore, my project will be focused on the beginning of the network update, starting by the migration of the virtual network from the router to the FWSM and, then, considering the different possibilities of development towards a network more flexible, scalable and productive.

Besides this, my position here includes diverse tasks around these devices: maintenance, management, errors correction, modification of filtering rules, helpdesk, etc.



2. Presentation of the MI2S

The MI2S is a *unité mixte de service* – UMS (combined service unit in French) created by the French National Centre for Scientific Research (CNRS), the National Polytechnical Institute of Grenoble (Grenoble-INP), and the Joseph Fourier University (UJF), with the purpose of providing a reliable service to the needs of the information, science and technology laboratories of Grenoble. This UMS can be divided in two parts: MIM and IS.

2.1. Moyens Informatiques et Multimedia (MIM)

The MIM is in charge of the acquisition and exploitation of the transverse devices intended to guarantee a good and common infrastructure for the research laboratories. This service specially offers the network infrastructure and services, the communications and the back-up of the data. Moreover, it provides technical mediums to the IS service with which they furnish multimedia support and put online and spread scientific events, and also, a video conference service. This is the department where I have been working as a trainee.

2.2. Information Scientifique (IS)

The goal of the IS is to give access and to the scientific information to the laboratories. It also works to spread, archive and manage the bibliography of the scientific results issued from the laboratories.

2.3. Units served

Currently, the service provided by the MI2S are extended to these units:

AGIM, CMP, GIPSA-lab, G-SCOP, ICA, LIG, LJK, TIMA, TIMC and Verimag.

2.4. Organization

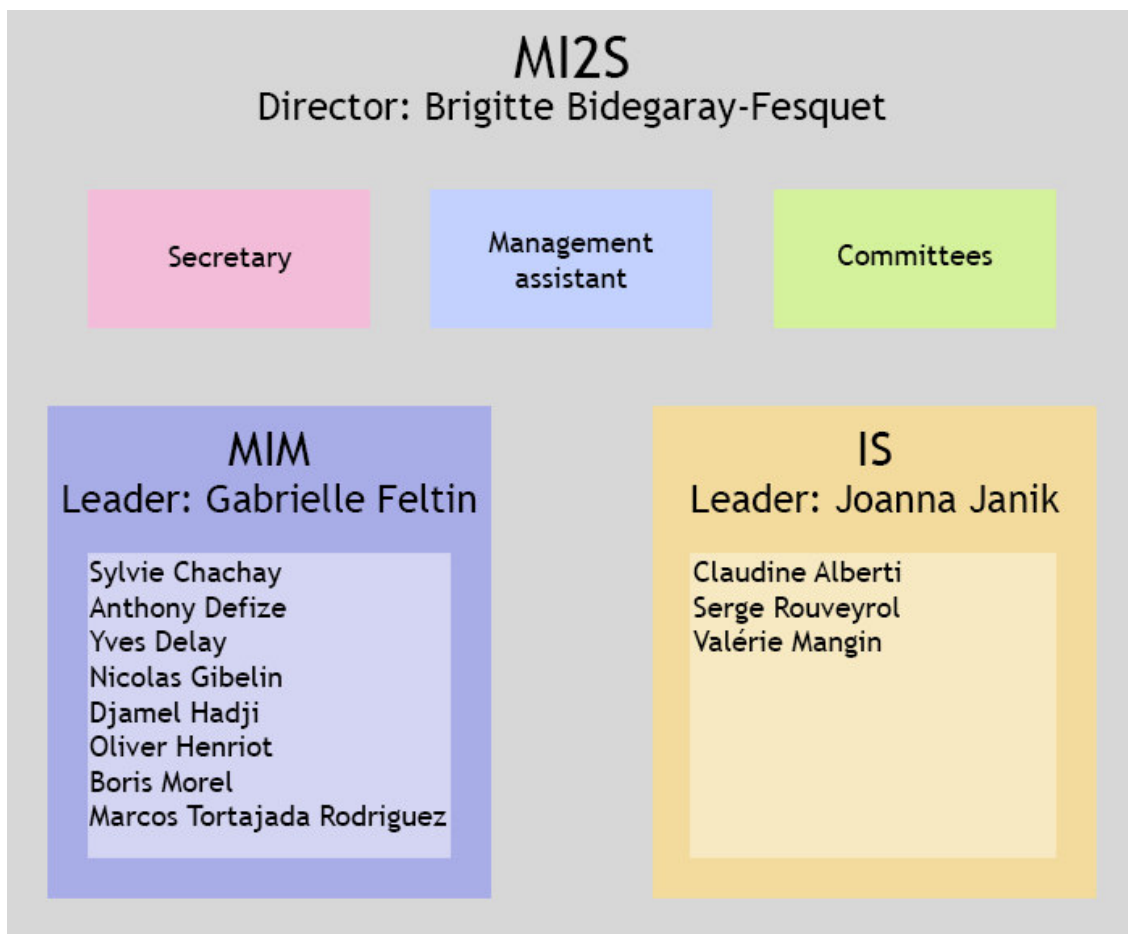


Illustration 1. Organization chart.

In this picture we see that the UMS is directed by Ms. Brigitte BIDEGARAY-FESQUET.

My direct boss is Ms. Gabrielle FELTIN and my supervisor is Mr. Yves DELAY. Most of the members of the MIM team work in the CETA building, the rest in the MJK; both buildings are centrally situated on the university campus Saint-Martin-d'Hères.

3. Initial work environment

3.1. General topology

As you can see in the next image, the MI2S is connected to different laboratories to which we provide our services and also to the RENATER backbone.

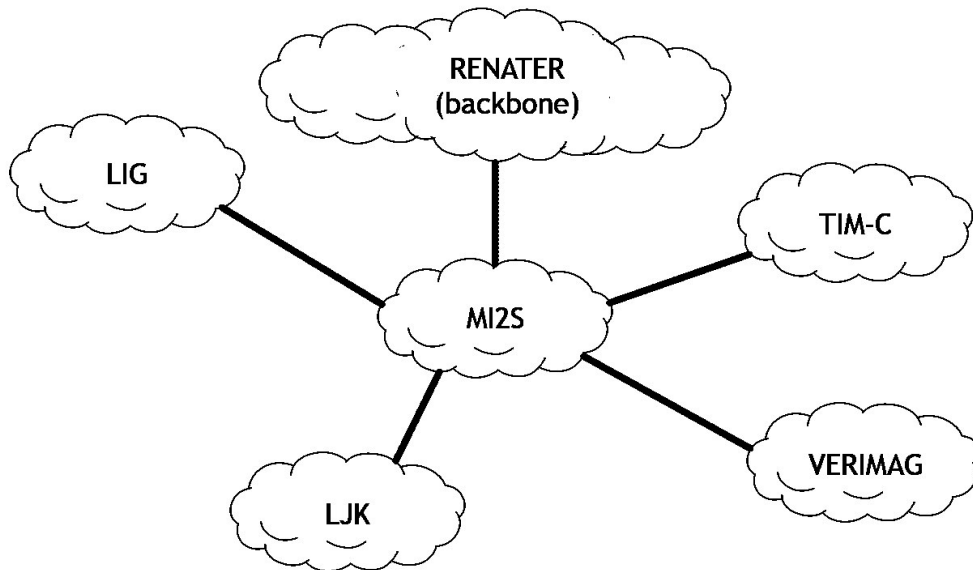


Illustration 2. Representation of MI2S connections. Not all the client laboratories are present in this picture.

Of course in our department we have many network devices among which there are five routers.

3.2. The chassis

My job is focused in a chassis Cisco Catalyst 6506. This device is a box where all its components are connected with the backplane, the most basic parts are the power supplies and the fan tray to refrigerate all the appliance; besides this, there are 6 slots to connect modules and provide different functionalities.

Our original case includes a supervisor card in charge of the routing task (in fact is a layer 3 switch), some service policies and the carrying out of several protocols like the Spanning Tree Protocol. Besides the slots to expand the memory and the console port, it has two ports of 1Gbit. The supervisor card is the brain of the chassis, we call it "the router" or with its domain name "rout-stmartin".

Also, our chassis includes another four modules, three of them are port modules (all able to manage speeds up to 1Gbit) to connect the different cables and the last one is a Firewall Service Module (FWSM) that provides an improved management due to some characteristics, like more advanced service policy rules, anti-spoofing, logs and customized graphs and the possibility to use a graphical interface that speeds up the rules and network objects management. The FWSM has no physical interfaces, they are virtual and all their traffic flows through the backplane and the router. The sixth slot is empty.



Illustration 3. Diagram of the main chassis of the campus Saint-Martin-d'Hères.

Some parts of the chassis are arriving to the end of date of support. In fact, the chassis itself and the fan tray are already out of warranty or maintenance contract. The supervisor and

interfaces card have less than two years of protection. The other components (FWSM and power supplies) are covered for three or more years.

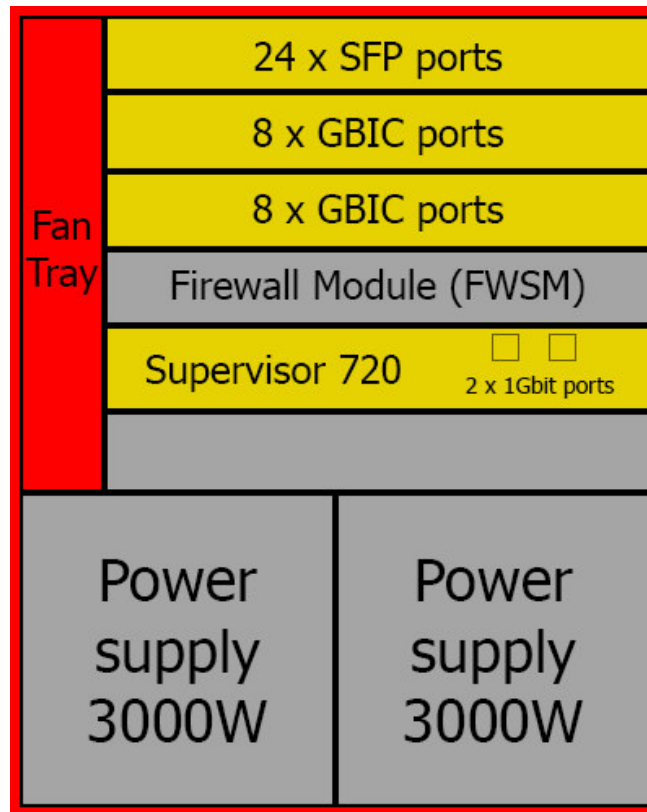


Illustration 4. Representation of the warranty or maintenance protection of the different components.

In front of this situation, the MI2S has decided to replace it by at least one Cat6506-E (a new version of our Cat6506) that also will increase the throughput and scalability in the network.

3.3. Internal topology the chassis

The logical representation of this device would be two different appliances, the router and the firewall. The first one is directly connected with the uplink to the core backbone of our network and to some other networks, these ones belongs to the different laboratories which we provide our services and also to ourselves. Besides the previous, we have a VLAN to connect with the firewall, this has a mask /30 (four IP addresses, one for each device, plus the network and broadcast addresses). Finally our FWSM has some VLANs connected to it.

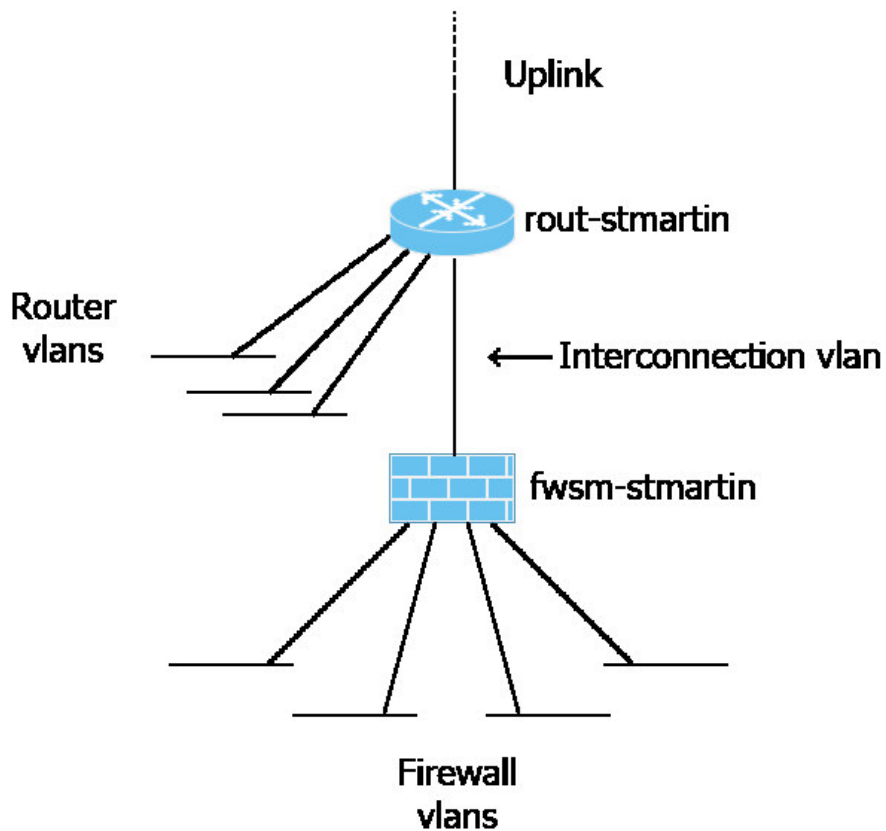


Illustration 5. Logical topology of the Cat6506.

We want to move all the VLANs from the router to the firewall, as their management is easier and improved in this one (for instance, this module is a stateful firewall, which means that if we open a rule in one way, it will remain open for the reply, with this we will save some rules).

Moreover, thinking in the future, it will be necessary to change the current FWSM by another more productive security device, and it's easier to migrate all the information to the new device from the FWSM than from the router.

Initial Process of Updating the Core of a Network

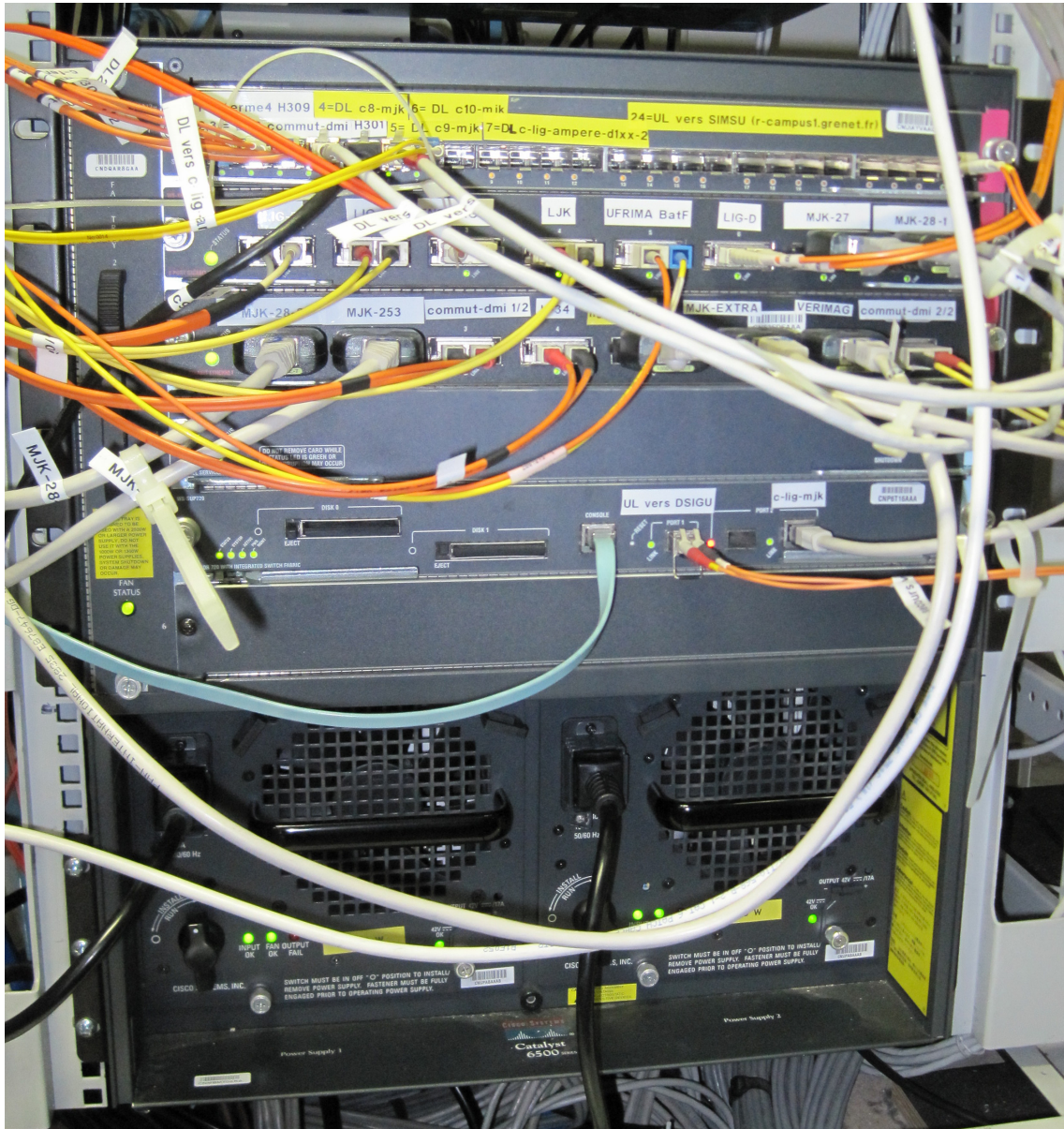


Illustration 6. Main Cat6506 chassis of the campus connected and working. Notice the fourth module (FWSM) without physical interfaces.

4. Migration of VLANs

My first task to start the evolution of the network process was to migrate the VLANs from the router to the FWSM. We did this, one by one, because the process for each migration is long and we have to be really careful, any mistake can cause problems to the laboratories or departments that use this VLAN or the services provided by its machines.

The only way to manage the supervisor card is with the command line interface (CLI); nevertheless our FWSM comes with an interesting tool, the Cisco Adaptive Security Device Manager (ASDM), a software with a graphical interface that makes easier the management of the rules, policies, and objects of the networks, of course we can still using a SSH connection to the firewall and use the CLI but, for instance, for the management of the access list, this application allow us to work faster.

The ASDM also shows us the logs in live and some basic graphs to control the CPU and traffic load.

4.1. Move the rules

Up to now, the filtering rules are configured in the router with Cisco Access Control Lists of level 3, these requires an intensive manual administration to avoid errors and it is poorly adapted to the applications that need dynamic port changes. The first step of every migration will be to move the access control lists associated to the VLAN from the router to the firewall.

As I said before, in the router we use the command line interface, so if we want to add a new rule we cannot just write the usual command to add the new rule because this is always at the end of the list and the order matters. The way we do the modifications is to write in a file the entire list of rules that concerns to the VLAN and make the changes in this file, then we put the file in a TFTP server which we can connect with the router to download it and configure the ACL with it. As you can guess, this is not a quick way if just want to add one rule.



Luckily with the ASDM we can modify a list of rules without the need of redefine it again.

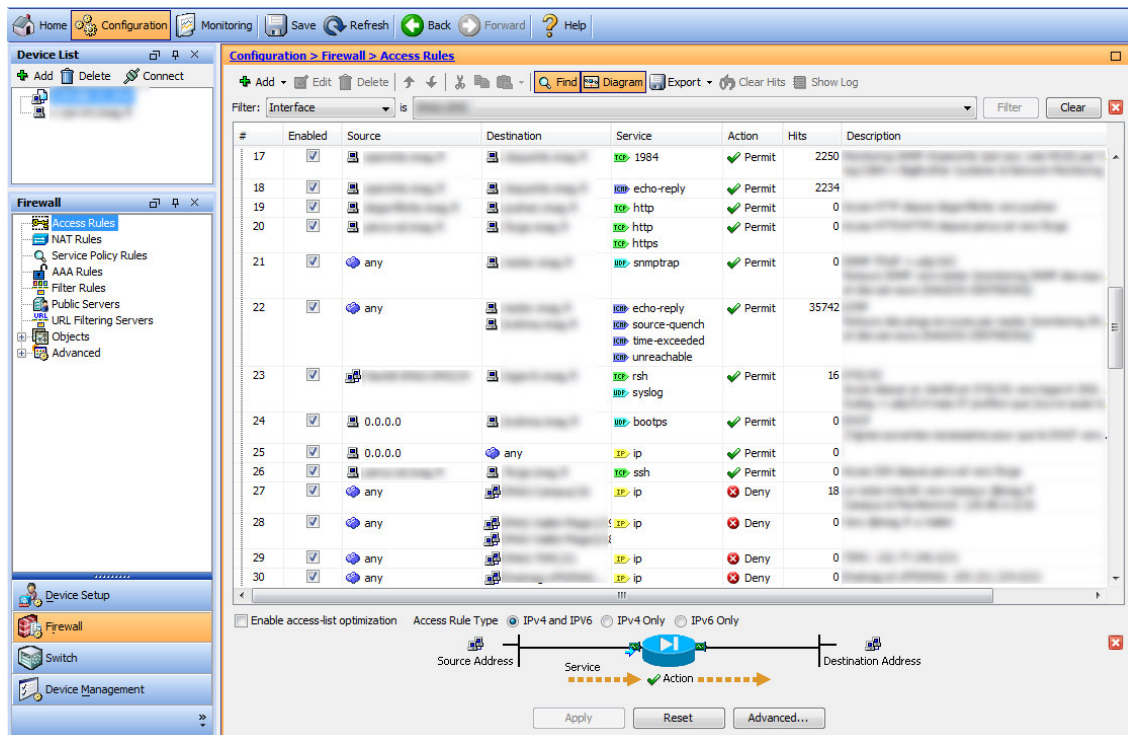


Illustration 7. ASDM, example of an access list.

As you can see in the picture, is simple to temporary enable or disable rules and of course is easy to add a new one in the middle of the list or to include a new machine in an existing rule.

The first thing I do to move the rules from the supervisor card to the FW5M is to print the ACLs, usually two (input/output), sometimes four (input/output x IPv4/IPv6). Then I check meticulously all rules to see if there is any inconsistency, sometimes happens after a lot of time of adding and removing rules, also I look for rules that are not used any more.

Once everything is clear, I can start to copy all of them to the firewall using the ASDM. At the same time I write the rules, I define the different network objects with their DNS names, with this, the future management of the rules will be really comfortable

4.2. Prepare the files

Then, we prepare two files, one contains the lines that we have to copy in both devices to move the VLAN from one place to another, the second file is prepared to do the step back in case of problems during the migration, so we introduce the commands to delete all new things in the firewall and to restore the previous situation of the interface on the router.

Example of a migration file, case of a DMZ VLAN which number is the 29:

On the router, we shut down the interface on the router and we specify that now will be located on the firewall:

```
conf t
interface Vlan29
shutdown
end
conf t
no interface Vlan29
end
conf t
firewall vlan-group 1 10, 29
firewall module 4 vlan-group 1
```

On the firewall, we create the new interface, always writing the same configuration that was written in the previous card, taking into account that some fields are different because the operative systems of the supervisor and the FWSM are similar but not equal. Then, we activate the access lists previously introduced using the graphical interface and also a command that prevent anti-spoofing attacks:

```
conf t
interface Vlan29
nameif DMZ1
description First DMZ
ip address 10.0.22.6 255.255.255.248
ospf cost 10
mtu DMZ1 1500
no shutdown
```

```
end
conf t
access-group DMZ1_in in interface DMZ1
access-group DMZ1_out out interface DMZ1
ip verify reverse-path interface DMZ1
end
```

Example of a step back file, of the same VLAN:

On the firewall, we delete the interface and its access lists:

```
conf t
no access-group DMZ1_in in interface DMZ1
no access-group DMZ1_out out interface DMZ1
end
conf t
no interface Vlan29
end
```

On the router, we create again the VLAN:

```
conf t
no firewall vlan-group 1 29
firewall vlan-group 1 10
firewall module 4 vlan-group 1
end
conf t
interface Vlan29
description DMZ1
ip address 129.88.27.6 255.255.255.248
ip access-group DMZ1_in in
ip access-group DMZ1_out out
ip verify unicast source reachable-via rx allow-default
no ip unreachable
no ip proxy-arp
no shutdown
end
```

I did a script (11. Appendix 1. Script: Check the migration files) to check these migration files and look for errors (for instance, some commands used in the router were not accepted by the firewall and sometimes we forgot this).

4.3. The migration

Finally, once everything is prepared and checked, we proceed to migrate the VLAN interface.

First we warn all laboratories that use the network with our schedule for the migration and we wait for their reply.

Then, at the planned time, we open Putty twice to connect via SSH to both devices and then we copy the corresponding code of the migration file in each one. Once this content has been copied, if there is not any error, the process is completed.

Since we are the security managers, we have access to some devices in the most of VLANs, connecting them, we can check the connections and see if everything is working properly and if the rules are blocking and allowing correctly. Also, we call to the involved to ask if everything is alright.

Sometimes, if there is a problem that cannot be solved in the same day, we do the “step back” copying the content of the file prepared for this case and the configuration of the router and the firewall will be restored, then we can start to investigate the problem thanks to our different logs and monitoring applications.

4.4. Conclusion

This work took a long time, it was started when I arrived to the company and it continues after I finished my internship since is a long task to move and check all the rules of our VLANs and we have a great number of networks that have not been migrated yet.



Initial Process of Updating the Core of a Network

In the future, as an upgrade of the management of our security and because it is arriving to the date of end of support of Cisco, the Firewall Service Module will be replaced by another device (7.2 New security appliance), with improved security policies and throughput and also a hardware handling of the IPv6 traffic, because the current one has not and controls it with software, that suppose an important growth in the CPU load. Then, if the new firewall is Cisco, we will take advantage of this task because there are tools to move the configuration from the FWSM to the new security appliances.



5. Rancid

During all the process of the update of the network core, some improvements are considered.

As a part of the network administration team, we modify usually the configuration of our routers and switches, for instance, to add or remove users or access lists. The point is that every time we change something, it is necessary to save the configuration in the start-up one of the device and also in the TFTP server to have the backup in case of breakdown.

We use different types of equipment, the main one are the routers (or level 3 switches) and the firewalls. In the case of the routers we do not modify the access lists directly in the device because the new rules are added at the end of the ACL and the order is significant, instead of this, we:

- Update the rules in the files stored in our TFTP, we have one file per virtual LAN.
- Backup the file in a different folder where we keep all the previous versions, we make the modifications and move this new version to the TFTP folder of the server.
- Connect to the router through SSH.
- Load the ACL file with the command *configure network*.

This manual method is really slow and prone to produce mistakes that will produce lost of information.

With the firewall and its graphical interface ASDM it's quite simple and we don't make the backup of independent ACL files, we just send the running configuration to the TFTP after each modification and do the security copy of the old one. The problem here is that we haven't any way to do go backwards in case of mistake easily, we have not a control of the different versions in the same management software.



Initial Process of Updating the Core of a Network

Also sometimes, because we are in a hurry, the change is small and not really significant or just because we forget to do it, sometimes the backup inside our TFTP is not done and this is a risk we should not accept. And also, because more than one person has access to these devices, if someone introduce a change or an error, it must be possible what and when has been modified.

So we need a tool able to get and manage the versions of the configuration of different kind of devices and also to do it automatically.

During my investigation I found different alternatives that could solve our necessities:

SolarWinds' Network Configuration Manager (NCM) – This complete tool suite offers a wide range of services to monitor via SNMP, send alerts, control the access lists, keep the version track of all devices and much more. Its price is over 2000€.

Cisco Prime LAN Management Solution (LMS) – Cisco provides an application to centralize and simplify the management of all our devices of this brand. In the same way of SolarWinds' software it includes tools to monitor the network, make audits and reports, etc.

Really Awesome New Cisco config Differ (RANCID) – Despite its name, this simple application is able to get the configuration of devices of different brands and keep a version track with CVS.

Finally I went for Rancid, being the most modest of the three studied, we don't need such a complete tool as NCM or LMS since we have their services covered with other applications. Also, Rancid is the only open source one, free and easy to configure or modify. The last thing that made me opt for my choice was the possibility of use it with not just the Cisco devices, because although right now the most of our routers and firewalls are Cisco we don't know yet if during the network renovation we will change some our devices to other brands like Juniper, Extreme Networks, etc.

5.1. Brief explanation about Rancid

Rancid is an open source software that allows to automate the backup of the network equipment and control the different revisions of their configurations with CVS (Concurrent Versions System). CVS is a software mainly used to manage projects of software development. It is based in repositories where all the files concerned to one project are located and the clients download a copy of the database to modify it and upload (commit) the changes to the repository. CVS also keeps a track of the versions of the files of the database making easy to restore a previous state of any file and to check the changes done by all the project users.

Rancid works with expect scripts which connect to the defined devices to get the running configurations and keep the track of all changes in them with a local repository.

This application allows to send an e-mail to the administrator every time a modification or problem is detected. Also, as it works with CVS, it is possible to create a web with web to control the revisions.

5.2. Testing Rancid

Some of these steps are used just in the case of the virtual machine, because the destination server had part of these features already installed or configured.

First of all, because the server where the software will be located (bragi.imag.fr) is using CentOS 6.4, I installed several virtual machines with the same Linux distribution using Oracle's Virtual Box in my desktop to test:

- The compatibility with this OS version.
- If the installation and configuration can be carried out easily and with the slightest impact on the environment, since it is going to be in a production system where other services are already installed and should not be interfered.

- Level of privileges needed by Rancid to run, as a high rights level can become a lack of security in our system.

Before start to configure the server we create a user in each router or switch we want to backup, this user with full privileges will allow the application to connect and get the information. Here the commands for the Cat6000 supervisor and FWSM:

```
rout-stmartin(config)# username rancid privilege 15 secret 0 password  
fws-smartin(config)# username rancid password password privilege 15
```

In a default installation we will have in Rancid's home the some new folders, including **bin**, **etc** and **var**:

- **bin** contains all the scripts used by the software, the main one is **rancid-run** which checks the devices.

- **etc** contains the configuration files, **rancid.conf** will allow to us to set a new group of routers and the destination of the software output.

- **var** contains the output of each execution and the repository used by rancid to show us the different versions, **var** also includes the logs where we can check if rancid had any problem to connect to the different destinations.

5.2.1. In the virtual machine

Check if the name of the machine is a valid DNS name:

```
prompt#hostname  
  
vBox  
  
prompt#hostname bruxelles.imag.fr
```

```
>vim /etc/hostname
```

- Edit file, replace *vBox* by *bruxelles.imag.fr*

Install packages required by rancid:

```
prompt#sudo apt-get install build-essential
```

```
#sudo apt-get install cvs cvsweb expect
```

Create rancid user:

```
prompt#sudo adduser rancid --home /home/rancid
```

Download rancid sources and install in rancid's home directory:

```
prompt#wget ftp://ftp.shrubbery.net/pub/rancid/rancid-2.3.8.tar.gz
```

```
>tar xvzf rancid-2.3.8.tar.gz
```

```
>cd rancid-2.3.8/
```

```
>./configure --prefix=/home/rancid/usr
```

```
>make install
```

Change the user and group of all software application to belong to rancid:

```
prompt#chown rancid:rancid /home/rancid -R
```

Configuration

```
prompt#su rancid

prompt>vim /home/rancid/usr/etc/rancid.conf

- Add /usr/sbin to the PATH variable

- Add LIST_OF_GROUPS="MI2S"
```

Now we initialize the CVS repository for the new group of devices "MI2S" with this command:

```
prompt>$HOME/usr/var/bin/rancid-cvs
```

Add the devices that we want to backup:

```
prompt>vim $HOME/usr/var/MI2S/router.db

- Write one line per device, first the DNS name or the IP, then the rancid ID for its operative system and then up or down depending if you want to include them in the process or not:

10.0.251.198:cisco:up

10.0.25.9:cisco:up

    >vim $HOME/.cloginrc

- Write the necessary information to access each device, in the method line we can add telnet if we want to use it. We must take into account if it is necessary to type "enable" in the device's console to get access or not, in these example we don't need to type this word in the router but we need it in the firewall module:

#configuration for the router

add password 10.0.251.198 password
```

```
add user 10.0.251.198 rancid

add method 10.0.251.198 ssh

add autoenable 10.0.251.198 1

#configuration for the FWSM

add password 10.0.25.9 password password_enable

add user 10.0.25.9 rancid

add method 10.0.25.9 ssh

add autoenable 10.0.25.9 0
```

Now to protect this file and the passwords:

```
prompt>chmod 600 $HOME/.cloginrc

>sudo chown rancid:rancid /home/rancid/.cloginrc
```

We can check if rancid is able to login on the devices with this command¹ as the rancid user:

```
prompt>$HOME/usr/bin/clogin -c "show version" -f $HOME/.cloginrc
10.0.251.198 \10.0.25.9 > test.txt
```

Open the text.txt file to see if the output of the show version command is there.

Up to here the basic configuration, now, to run rancid:

```
prompt>/home/rancid/usr/bin/rancid-run
```

¹ we use the binary clogin because we are working with a Cisco device, but rancid uses a different executable for each operative system.



You can find the logs on `/home/rancid/usr/var/logs`.

Scheduler

Now, to automate the execution of rancid, we use the utility crontab included in Linux:

```
prompt>su rancid

    >crontab -e

- Here, we are asked which text editor do we want to use, choose your favorite.

- For the schedule you have to follow a pattern explained in the comments of the file, for
example:

10 * * * * /home/rancid/usr/bin/rancid-run

30 23 * * * /usr/bin/find /home/rancid/usr/var/logs -type f-mtime +7 \
-exec rm {} \;

- The first line will execute rancid every hour at minute 10. The second line, every day at 23:30,
will look for the logs older than 7 days to delete them.
```

CVS (web)

```
prompt>sudo vim /etc/cvsweb/cvsweb.conf

- Modify the @CVSrepositories variable:

@CVSrepositories = (

'Local' => ['Local Repository', '/var/lib/cvs'],

'MI2S' => ['MI2S', '/home/rancid/usr/var/cvs'],

);

- /home/rancid/usr/var/cvs is the directory where the repository is placed by default.
```


Create a symbolic link for the icons and style sheets:

```
prompt>sudo ln -s /usr/share/cvsweb/ /var/www/cvsweb
```

You can use know the web interface in <http://bruxelles.imag.fr/cgi-bin/cvsweb/MI2S/configs/>

Diff for /MI2S/configs/10.0.25.9		between versions 1.7 and 1.8	
version 1.7, 2013/06/06 08:17:04		version 1.8, 2013/06/10 14:09:42	
Line 16		Line 16	
!Flash: file 0: origin: 0 length:6390272		!Flash: file 0: origin: 0 length:6390272	
!Flash: file 1: origin: 6390272 length:12747700		!Flash: file 1: origin: 6390272 length:12747700	
!Flash: file 2: origin:19138048 length:5021		!Flash: file 2: origin:19138048 length:5021	
!Flash: file 3: origin:19143168 length:381329		!Flash: file 3: origin:19143168 length:381117	
!Flash: file 4: origin:21085696 length:280		!Flash: file 4: origin:21085696 length:280	
!		!	
!		!	
Line 1424 object-group network DM_INLINE_NETWORK_7		Line 1424 object-group network DM_INLINE_NETWORK_7	
object-group service DM_INLINE_TCP_33 tcp		object-group service DM_INLINE_TCP_33 tcp	
group-object VMware_Remote_Console		group-object VMware_Remote_Console	
group-object VMware_Server_Console		group-object VMware_Server_Console	
object-group network DM_INLINE_NETWORK_72			
network-object host .imag.fr			
network-object host .imag.fr			
object-group service DM_INLINE_SERVICE_2		object-group service DM_INLINE_SERVICE_2	
service-object tcp eq netbios-ssn		service-object tcp eq netbios-ssn	
service-object udp eq netbios-dgm		service-object udp eq netbios-dgm	

Illustration 8. Differences between two configurations on cvsweb.

E-Mail

If we want Rancid to send an email each time we have a modification in a device configuration or a problem:

```
prompt>vim /etc/aliases
```

- Add the next lines:

```
rancid-MI2S: bruxelles@imag.fr
```

```
rancid-admin-MI2S: bruxelles@imag.fr
```

- The first address will receive the report with the changes if there is anyone, the second address will receive the errors.

As a Mail Transfer Agent, I prefer to use EXIM, here the installation and configuration:

```
prompt>sudo apt-get install mailutils
```

```
>sudo apt-get install exim4
```

```
>dpkg-reconfigure exim4-config
```

This command starts the configuration of Exim, follow this steps:

OK;

Mail sent by smarthost: no local mail

*System mail name: **bruxelles.imag.fr***

OK;

127.0.0.1 ; ::1

*Other destinations for which mail is accepted: **(Erase the default, let it in blank);***

*Visible domain name for local users: **imag.fr***

*IP address or hostname of the outgoing smarthost: **mailserver.imag.fr***

OK;

NO;

NO;

OK;

*Root and mostmaster mail recipient: **your.email@imag.fr***

Try to send an email with:

```
prompt>mail -s 'testSubject' your.email@imag.fr < /etc/passwd
```

You can check the mail logs on /var/log/exim4/mainlog

To check if the port 25 is already in use:

```
prompt>sudo netstat -taupen | grep 25
```

After run rancid, if it detects a change in any configuration, we will receive an e-mail like this one received after add a new user to the router:

```
Index: configs/10.0.251.198
=====
retrieving          revision          1.12
diff -u -4 -r1.12 10.0.251.198
@@ -592,8 +592,9 @@
+          username          userTestR
aaa          new-model
aaa authentication login default local
aaa authorization exec default local
!
```

5.2.2. Team meeting

After test the application and get the total understanding of its functioning I made a presentation to the team and bosses with a live demonstration to show Rancid's functionalities, how it works and the installation process.

Two main questions were formulated:



- Is it possible to use a versioning control system different than CVS? Subversion in fact, as it's the one used by the company.

- Is it possible to configure Rancid use a remote repository? Because we want to install the software in one machine and we have another server which is use to store all the repositories.

Then I started to work in these requirements, for this reason, I created new virtual machines to test the new configurations.

To use Subversion instead of CVS was not difficult, Rancid is already prepared for that, but to configure the remote repository required more investigation than expected. I will explain the process directly in the section **4.3. Introduce Rancid in the production system** since the tests I did were close to the real environment as I already used the real machine which stores the repositories to check that everything was working properly.

5.3. Introduce Rancid in the production system

To install this software in the company network we had to do several modifications to the previous guide. It was a requirement of our chief to use Subversion, often abbreviated SVN, instead of CVS. Subversion is an open source software that solve some inconsistency problems of CVS and adds new features.

In the MI2S we have already a server where we store all the repositories, this machine (freyja.imag.fr) will keep Rancid's repositories while the software itself is installed on bragi.imag.fr.

So, to achieve this, we will do some variations in the software before run the command "rancid-cvs" which prepares the local repositories:

```
prompt#su - rancid
```

```
prompt>vim $HOME/usr/etc/rancid.conf
```

- Modify the **RCSSYS** and **CVSROOT** variables:

```
RCSSYS=svn; export RCSSYS
```

```
CVSROOT=$BASEDIR/SVN; export CVSROOT
```

- **CVSROOT** is the address of the Subversion repository, the problem is that Rancid does not allow remote addresses, we will have to do a small trick to solve this.

We run now the command `$HOME/usr/var/bin/rancid-cvs` and we follow the guide of the test until the end.

5.3.1. *Configure the remote SVN*

Since Rancid is not able to manage natively remote addresses for the repository and the company has a server for this purpose, a deeper study of this software and SVN work was required to finally invent this trick and make it work.

As we already know, when we run `rancid-run` it stores the devices configuration in `/home/rancid/usr/var/MI2S/`, this is the working directory of the Subversion repository that right now is locally located in `/home/rancid/usr/var/SVN`. The point here is to modify the working directory to make it commit against our corporative repository, we cannot just make a checkout of this one from our local working directory because it already is linked to the local deposit. Lets see step by step how to do it.

In `freyja.imag.fr` we create a new repository:

```
prompt#svnadmin create /home/web/subversion/rancid
```

We give the ownership to apache since we want to access using HTTPS:

```
prompt#chown -R apache:apache /home/web/subversion/rancid
```

Also we make the convenient set up for the web (do it accessible, add user and password, restart service...).



To access to it with the protocol HTTPS (<https://freyja.imag.fr/svn/rancid>), we will open the ports to allow the access of from bragi (rancid) to freyja (repository).

Firewall:





Enabled	Source	Destination	Service	Action	Description
<input checked="" type="checkbox"/>	 freyja.imag.fr	 bragi.imag.fr	 https	 Permit	Allows rancid communication with the SVN repository

Illustration 9. Rule on the ASDM allowing the access from our working directory to the repository server.

Iptables on freyja.imag.fr, allowing bragi's access:

```
-A INPUT -s 10.0.25.10 -p tcp -m state --NEW -m tcp -dport 443 -j ACCEPT
```

Now we just have to redirect the commits of Rancid to our remote repository on freyja.imag.fr. To do this, we must do `/home/rancid/usr/var/MI2S` to be the working directory of `https://freyja.imag.fr/svn/rancid`:

```
prompt#cd /tmp
```

- Bring the information stored in our remote repository which is "empty" to `/tmp/rancid`, our interest is to bring its svn configuration:

```
#svn checkout http://freyja.imag.fr/svn/rancid
```

Now we replace the contents of our current Rancid working directory by the files inside `/tmp/rancid`, taking care to not forget the hidden files which carry the configuration to connect to freyja.imag.fr. In this way, we also change the configuration of our svn working directory, it will be pointing to the corporative repository.

```
prompt#rm -Rf /home/rancid/usr/var/MI2S
```

```
#mv /tmp/rancid /home/rancid/usr/var/MI2S
```

Remember to put rancid as an owner of its files.

```
#chown -R rancid:rancid /home/rancid
```

Each execution of rancid-run gets the running configuration of all devices, stores them on /home/rancid/usr/var/MI2S and commits. From now on, its commits will be directed to our remote repository.

We must do the first commit manually, because it's done using the protocol https, so we will be asked for the password and to accept the certificate, this has to be done manually.

```
prompt#su - rancid
```

```
prompt>cd /home/rancid/usr/var/MI2S
```

```
>svn commit -m "Initial rancid repository"
```

- Accept the certificate and to store the password as plain text.

- Change the permissions of the directory /home/rancid/usr/var/MI2S/.svn to protect the password.

```
>chmod -R 600 .svn
```

With that, we have already our remote repository configured, currently is empty but, with the first execution of rancid-run we will commit the entire configuration of our devices, from this point on, next time we run Rancid, we will start to register the differences on our routers' configuration.

5.3.2. Repositories on the company website

In the guide above we explained how to install cvsweb, now to do SVN accessible we could use the online browser websvn but, instead of this, we are going to take advantage of the current Redmine web project manager already installed in freyja.imag.fr.



Initial Process of Updating the Core of a Network

Create a new Redmine project and include in it the Rancid repository is the easiest solution to make the versions of the devices configurations to the team members.

Network for Mi2s » Rancid for Mi2s

Search: » Rancid for Mi2s

Overview Activity Issues New Issue Calendar Documents Wiki **Repository** Settings

rancid Statistics | Revision:

Name	Size	Revision	Age	Author	Comment
└─ mjk		14	about 4 hours	rancid	updates
└─┬─ configs		14	about 4 hours	rancid	updates
└─┬─┬─ 10.0.251.198	405.983 KB	14	about 4 hours	rancid	updates
└─┬─┬─ 10.0.25.9	374.615 KB	11	about 6 hours	rancid	updates
└─┬─┬─ router.db	43 Bytes	11	about 6 hours	rancid	updates
└─┬─┬─ routers.all	37 Bytes	11	about 6 hours	rancid	updates
└─┬─┬─ routers.down	0 Bytes	4	3 days	rancid	restore the initiale version
└─┬─┬─ routers.up	37 Bytes	11	about 6 hours	rancid	updates

Latest revisions

#	Date	Author	Comment
14	01/07/2013 12:45 pm	rancid	updates
13	01/07/2013 12:35 pm	rancid	updates
12	01/07/2013 12:15 pm	rancid	updates
11	01/07/2013 10:45 am	rancid	updates
10	01/07/2013 10:04 am	rancid	updates
9	01/07/2013 10:00 am	rancid	updates
8	28/06/2013 04:59 pm	rancid	updates
7	28/06/2013 04:51 pm	rancid	updates
6	28/06/2013 04:51 pm	rancid	new router
5	28/06/2013 04:51 pm	rancid	set svn:ignores

[View differences](#)

[View all revisions](#)

Illustration 10. Rancid's repository in Redmine.

Network for Mi2s » Rancid for Mi2s

Search: » Rancid for Mi2s

Overview Activity Issues New Issue Calendar Documents Wiki **Repository** Settings

Revision 8:10

View differences: inline side by side

```
mjk/configs/10.0.251.198
591 username fcltzn privilege 15 secret 5
592 username rancid privilege 15 secret 5
593 username lortajal privilege 15 secret 5
594 username userTest privilege 0
595 username userTest2 privilege 0
596 aaa new-model
597 aaa authentication login default local
598 aaa authorization exec default local

591 username fcltzn privilege 15 secret 5
592 username rancid privilege 15 secret 5
593 username lortajal privilege 15 secret 5
594 aaa new-model
595 aaa authentication login default local
596 aaa authorization exec default local
```

Also available in: [Unified diff](#)

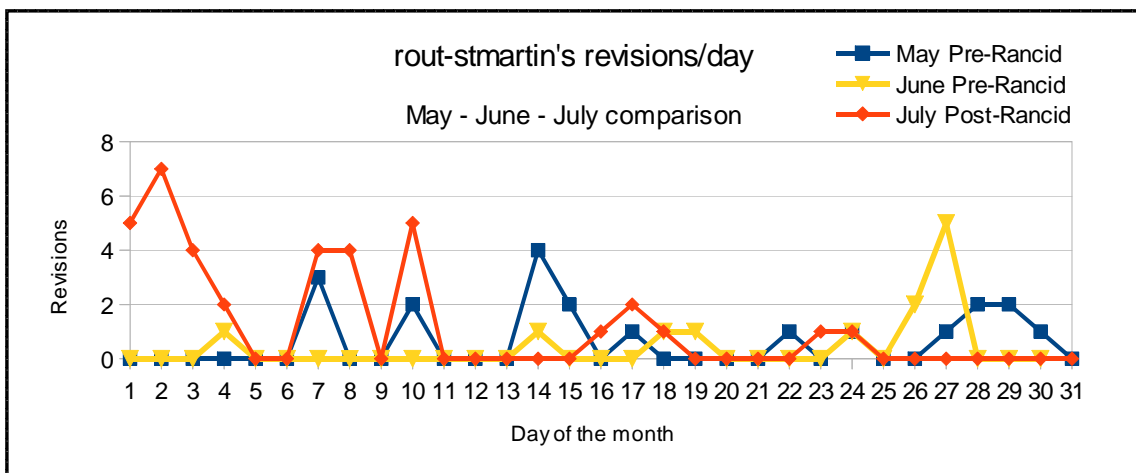
Illustration 11. Differences between two configurations of the router.

5.4. Conclusion

After some time, it has been proved that is really comfortable not to have to do the backup of the devices manually every time we modified something. Moreover, to store the revisions in the repository integrated in Redmine has been a good idea, doing faster to consult the information and see precisely what has been changed in the configuration each time.

This tools safe time, not just because save automatically the data, but also because make the information more accessible and clear.

Moreover, after one month with this application in our production environment we have experienced an increase in the number of revisions. For instance, or the machine rout-smartin, we have more updates registered in this month than than in the two previous ones together. That is more significant if we take into account that the main administrators of this device were on holidays the last week of July.



Total of May: 20 revisions

Total of June: 12 revisions

Total of July: 37 revisions

Initial Process of Updating the Core of a Network

After the positive experience with few appliances, we included the rest. Right now we have more than one hundred devices, most of them switches, been checked by Rancid every 30 minutes. Each execution takes 15 minutes.

Finally, for the future, we wrote the necessary documentation to manage this tool in the “Wiki” section in the Redmine project, here we explain the installation process and how to add new devices to be backuped.



6. Replacement of the chassis

As we saw in the introduction, the end of warranty or maintenance contract of some of the components of our chassis Cat6506 and an increasing traffic load in the network, our manager decide to buy a new chassis to replace the current one.

When I arrived to the company, this new device was already chosen and ordered to our provider. The choice was the Cisco chassis Cat6506-E, the latest version of our appliance. It was requested with three components: a fan tray for the refrigeration, one port card with 48 SFP interfaces and the supervisor.

In this case the supervisor card is the model 2T with an improved processor and two interfaces of 10Gbit/s, ideal for the uplink.

6.1. Relocating the components

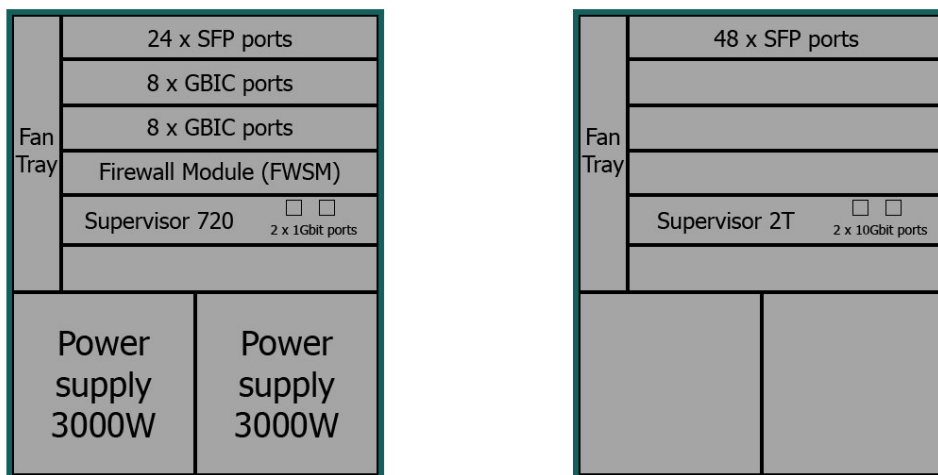


Illustration 12. Representation of the modules in both chassis, on the left the Cat6506 and on the right the Cat6506-E.

Looking to the previous illustration it's possible to realize that for the replacement of the old chassis by the new one we have a lack of components.

Initial Process of Updating the Core of a Network

The port module:

With the new port module (48 ports), we have enough interfaces to include all the already existing in the three cards of the Cat6506 (24+8+8=40), so we don't need any other port module.

The firewall:

We didn't buy a new security module or appliance, so we will use the FWSM.

The power:

About the power supplies, we have two old ones in spare of 2500W but, because for the moment, after we replace the chassis, we are not going to use the Cat6506, it is more convenient to use the two units of 3000W that are being used in it.

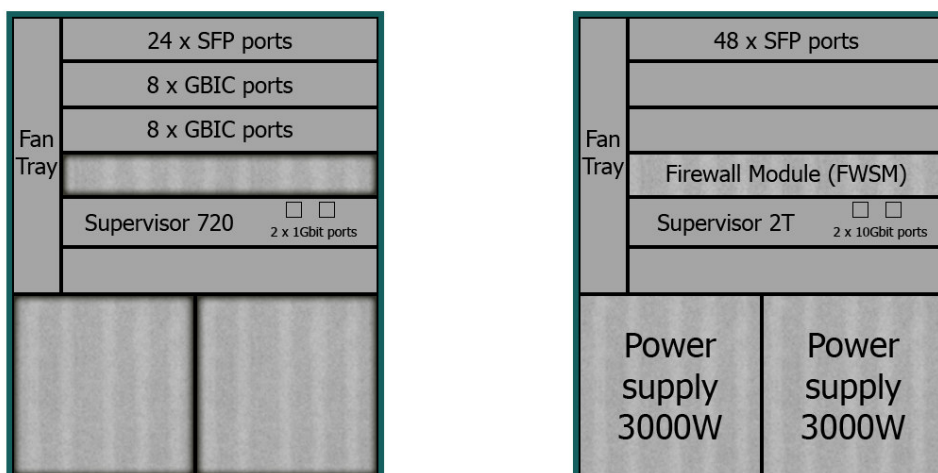


Illustration 13. New representation with the modules already relocated.

6.2. Incompatibility of the transceivers

Now, going one step ahead, thinking in the physical links, even if our current number of cables and connectors fits in the new SFP module, there is an incompatibility of technologies.

There is no problem to move the 24 SFP transceivers and their cables (LC fibre optics) but, the GBIC transceivers of the two GBIC cards cannot be added to our new card and we cannot move the entire cards to the new chassis because they are not compatible with the new Supervisor 2T. Moreover, the type of cable used in GBIC (SC fibre optics) will have to be replaced by the LC ones.

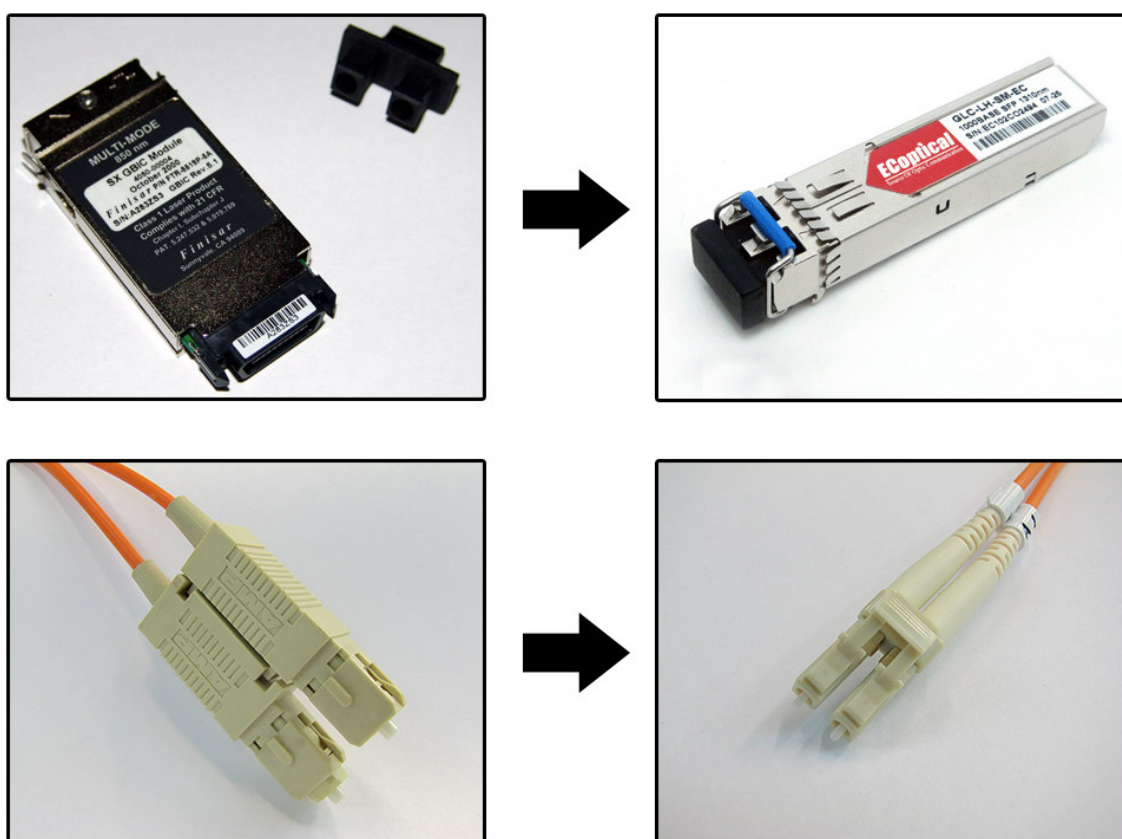


Illustration 14. On the left: GBIC transceiver and SC fibre optic. On the right: SFP transceiver and LC fibre optic.

6.3. Before the hardware replacement

Once our provider delivered the new chassis we could to start the real preparation of the replacement.

First, using the two power supplies we have in spare, we connected the Cat6506-E to the electrical network to check if it was working correctly. Then, we had to prepare it, we followed the next steps:

Update the operative system

Because it was not delivered with the latest version of Cisco IOS, we had to install it. For this, we connected a computer with the console port to the supervisor card. With this connection we were able to configure an interface to connect it with an RJ45 cable to the same computer where we have an FTP server, in this server we put the latest version of Cisco IOS for our supervisor (the 15.1(1)SY1). Of course, this small network was built choosing IPs in the same network, etc. the addresses are not important, since this configuration was temporary.

We checked the size of the new image and the free space on the flash memory of the supervisor. Then we downloaded from the FTP to the card the OS and replace the boot variable to load the device with the new operative system. The last thing to finish the software upgrade is to restart the device.

Configure the supervisor

To make the transition as fluent as possible, we must configure our router before introduce it in the production environment, for this reason, we copied the configuration from the supervisor that was already working to this one. This took some time because since they don't use the same operative systems, some commands had changed or disappeared.

Besides this, all the VLANs information are stored in a file called vlans.dat. We downloaded this file to the Cat6506-E to get the current list of VLANs.

Prepare the wiring

Because we were going to move a big amount of wires and to put new ones (the new LC fibre optics) we labelled all of them to keep the order.

Warn

Finally we warned by mail all the laboratories, organizations, departments that are connected to us about our intention to replace this central device of our network. We agree on a date and time to carry on this work.

6.4. The replacement

Finally, we started the change on June 2, at 8 PM, at this time, nobody was working in our offices and everybody knew about this operation.

Because all the previous preparations, the process was not too long.

First of all, we disconnected some monitoring systems because to disconnect one of our main routers would trigger a big amount of warnings and alarm messages.

Then we proceeded to disconnect the Cat6506 and extract it from the rails of the rack. Immediately, we introduced the new Cat6506-E in the frame.

We connected one computer to the console port before start the device, in this way, we were able to read all the boot information. We turned on the new chassis and started to plug all the cables following the labels, the loading process was longer than expected because the hardware started to check all the interfaces.

Looking the console output we realised that almost everything was working fine but we detected two main problems: one VLAN and one mail server didn't have network connection. Luckily we found the problems quickly, one transceiver was damaged and one cable was connected to the wrong port.

Initial Process of Updating the Core of a Network

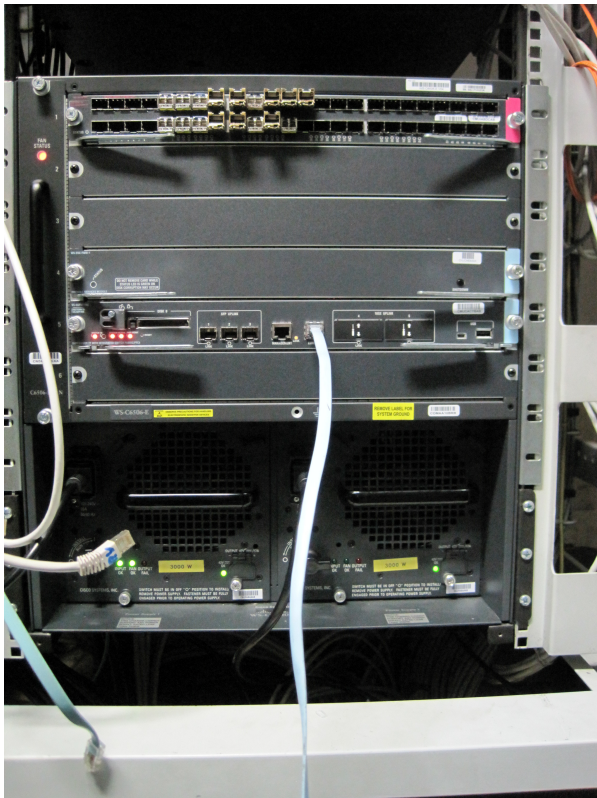


Illustration 15. Cat6506-E in the rack, loading with the console port connected.



Illustration 16. Cat6506 already out of the rack, without the power supplies nor the FWSM since they are in the new chassis.

6.5. The day after

The next morning we paid special attention to our ticket-tracking system (Request Tracker), to see if there was any incidence reported by any lab.

We just detected one problem with the logs of the supervisor, it was emitting and receiving some error messages related with the PIM protocol, after investigate the protocol and the logs I found that the reason of the problem were some missing lines in the configuration of the device. Other routers were trying to register r-campus as a source of some multicast groups and r-campus was not properly configured to accept. After add the needed lines, the problem disappeared.

7. Future

Until here the progress we have done for the update of the network core, but this is far from over. The new chassis has been already installed, but we are looking for some improvements in our network. My managers asked me to investigate two of them: a redundancy system for the router and the different options to replace our current firewall by a newer one.

7.1. High availability

In the halfway through my internship, my supervisors asked me to investigate the different possibilities to establish a high availability system for our chassis. The suggested a scenario with two chassis connected to the network and carrying out the same task, in case of failure of one of them, the other would overtake all the work without any cut of the network services.

7.1.1. *Early study*

After this generic and initial requirement I started my study of all the options, also checking the compatibility with our Cat6506 and the Cat6506-E with the Cisco Feature Navigator².

Gateway redundancy

HSRP (Hot Standby Routing Protocol)

It is an active/passive Cisco proprietary redundancy protocol for establishing a fault-tolerant default gateway.

It's necessary to create a virtual IP and a virtual MAC address to make the redundancy.

With this configuration we have:

An active router: the one that receives the traffic and redirects it to its destination.

² <http://tools.cisco.com/ITDIT/CFN/jsp/by-feature-technology.jsp>



Initial Process of Updating the Core of a Network

A standby router: the one that wait the failure of the active one to overtake its traffic.

Virtual router: It's not a router, but it represents the HSRP group as a virtual router and it's the gateway for the hosts.

Both routers send "Hello" messages every 3 seconds, the active one to know their neighbours and the standby to know if the active router is still alive (using the multicast address 224.0.0.2 and the port 1985 UDP).

If the active router doesn't reply in 10 seconds, the standby router takes over as the active one.

HSRP doesn't support load balancing, but it's possible to use a trick that is divide the different servers/machines in two groups and make each group to use a different router, so each router is the active router for its group and the standby for the other and vice versa. If one of both routers fails, the other one assume both groups.

It is not compulsory to have the same model of devices to configure HSRP between them. It's compatible with our Cat6506 IOS.

VRRP (Virtual Router Redundancy Protocol)

It's a standard appeared as an alternative to HSRP and it's described in RFC 2338. They are almost equal but VRRP operates on multiple vendors' routers.

The timing intervals on VRRP are faster than HSRP and it uses the multicast address 224.0.0.18.

It's compatible with our Cat6506 IOS.

GLBP (Gateway Load Balancing Protocol)

Cisco's proprietary protocol designed to provide a load balancing. Unlike HSRP or VRRP, all the routers are active.



One router is chosen as the Active Virtual Gateway (AVG). This router will reply to all ARP request coming from the different servers to the virtual router, then it will reply with the MAC of the corresponding router. It also uses the same method with the "Hello" messages to know if their neighbours still alive.

The load balancing can be configured with different behaviours:

- Round robin (by default): Same load for all routers.
- Weighted: We can use weights to distribute the load with the percentages that we want.
- Host dependent: Each client always uses the same router

It's compatible with our Cat6506 IOS.

Supervisor redundancy

The three previous methods are used to reach the redundancy of the gateway, but we can also get the redundancy of the supervisor. There are three modes of redundancy:

RPR (Route Processor Redundancy)

The redundant supervisor it's just initialized in part, and it loads the modules of the switch and the rest of functions when the active one fails.

Dynamic routing information is lost during the fail-over because it is not synchronised.

RPR+

The redundant supervisor is loaded almost completely (but not the layer 2 and 3 functions).

SSO (Stateful Switchover)

The redundant supervisor is totally initialized and the configurations running and startup of both engines are synchronised.

During the fail-over the switching hardware still is working because the layer 2 information is in both supervisors. Links aren't lost and the spanning tree protocol is not affected. It's just necessary to rebuild the ARP tables.

If we also use **NSF (NonStop Forwarding)** we get a fast regeneration of routing information. With SSO & NFS almost 0 packets are lost during the transition.

RPR+ and SSO/NSF are supported by our Cat6506 IOS, but I proposed to check with our provider if the combination of SSO/NSF + HSRP or SSO/NSF + VRRP was also supported.

VSS (Virtual Switching System)

In the future, if we buy another Catalyst 6506-E an Active/Active after buy another Chassis Cat6506-E, this will allow us to balance the load between the routers and also to manage both as a single virtual device. All looks like a single chassis.

It requires two Catalyst 6506 Series Switches with the same supervisor card (sup720-10GE or sup2T) and a Virtual Switch Link (connections 10Gb Ethernet between both devices).

The IOS version of our old Cat6506 is not compatible with VSS³, this system will be possible just if we get another Cat6506-E.

The advantages are:

- With this, there is no need of VRRP, HSRP, GLBP.
- No need to have Spanning Tree Protocol between the devices.

³ See "Table 1" at:
http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps9336/prod_qas0900aecd806ed74b.html

- It is better for the management because it's only necessary to administer one device (one routing point, one place for the rules...).
- The only thing in standby is the console of the slave chassis because now both are merged and we just need one console.
- To keep both synchronised, VSS uses SSO/NSF.

7.1.2. Deeper in HSRP

After this study and facing the fact that the purchase of a new Cat6506-E was not going to be soon, I was asked to go deeper in the HSRP system, with an Active/Passive configuration.

I extended the previous information about HSRP and added an example to configure one of our network with this system.

Hot Standby Routing Protocol

HSRP is a Cisco protocol that allow several routers or layer 3 switches to achieve an active/passive redundancy system. From now on, I will use the term "router".

All devices inside this system (CAT6506 and CAT6506-E in our case) will seem to the network as a unique gateway. They look like a single virtual router for the hosts.

In this protocol, one router is active and another is in standby mode, waiting to take over the tasks of the active one in case of need.

To reach this behaviour it is necessary to configure all the devices we want in the same HSRP group and to assign a number between 0 and 255 to it. The active router will be the only one that forward packets.

In case of failure of the active device, the standby becomes the active and, if we have more routers in the group, one of those will be elected as the new standby router.

To communicate the status and existence of all the group members, they exchange "HELLO" multicast messages (address 224.0.0.2, port 1985 UDP). Even if the routers look as a single



virtual gateway to the host, for this purpose the different members of the HSRP group use their real IP address to identify themselves. By default, the active router sends a "hello" message every 3 seconds, if during 10 seconds no message is received, it is considered that the active router has failed and the standby one takes over the active role. After this, if the router with the failure is recovered it will not become the active again until the current one fails, but we can force the *preempt* command.

In the same way that every router has its own IP and all of them share a virtual IP for the gateway, this gateway has also a virtual MAC address that follows this pattern:

00 – 00 – 0C – 07 – AC - **XX**

Where XX is the HSRP group number.

Configuration commands

These are the main commands to configure HSRP in one vlan.

Remember to put rancid as an owner of its files.

```
#chown -R rancid:rancid /home/rancid
```

Define the virtual gateway address:

```
router(config-if)#standby group-number ip ip [secondary]
```

Set the priority to be active (from 0 to 255):

```
router(config-if)#standby group-number priority priority
```

Hello and hold timers can be changed (optional):

```
router(config-if)#standby group-number timers hello-seconds holdtime-seconds
```

Force a device to be the active one if its priority is the highest, without this command, even if the standby router has a higher priority, it waits until the active falls:

```
router(config-if)#standby group-number preempt
```

Change the priority in case of fail in the uplink goes down:

```
router(config-if)#standby group-number track interface new-priority
```

Example for VLAN 19 (DMZ1)

In this case, we have a network 10.0.29.192/26 which gateway is 10.0.29.254. To configure HSRP we will give a new IP to each device and we will keep the current one as the virtual one, in this way, we don't have to modify any host.

On the Cat6506-E:

```
rout-stmartin-E# conf t
rout-stmartin-E(config)# interface vlan19
rout-stmartin-E(config-if)# ip address 10.0.29.253
rout-stmartin-E(config-if)# standby 19 ip 10.0.29.254
rout-stmartin-E(config-if)# standby 19 priority 120
rout-stmartin-E(config-if)# standby 19 preempt
rout-stmartin-E(config-if)# standby 19 track GigabitEthernet1/24 50
```



On the Cat6506:

```
rout-stmartin#conf t
rout-stmartin(config)# interface 19
rout-stmartin(config-if)# ip address 10.0.29.252
rout-stmartin(config-if)# standby 19 ip 10.0.29.254
rout-stmartin(config-if)# standby 19 preempt
```

Once these commands has been written in the consoles, HSRP is running, there is nothing else to do.

Hardware requirements

One of the issues that is necessary to solve is to have enough cables and transceivers to duplicate all the interfaces. Besides the interfaces, during the replacement of the chassis we took some components from the old one, luckily we have two power supplies and a FWSM in spare.

Finally, we checked that the power consumption of the two chassis together manageable by the electricity network of our building.

Conclusion

When I finished my internship, the HSRP was not implemented yet, but I already wrote part of the configuration of several VLANs that will guide my supervisors to finish the tasks themselves.

7.2. New security appliance

With the growing size of the network and the new services, our Firewall Service Module is becoming unable to manage correctly our traffic and it is arriving to the date of end of support by the Cisco technical assistance centre. Some of our machines and services work with IPv6 and the FWSM can handle it but not natively, this cause an important rise of the CPU load each time that we connect to the firewall a new machine which uses IPv6.

This means that every time that we want to migrate a VLAN from the router to the FWSM, we have to check if this VLAN uses IPv6 or not, we already underwent high load rises and if we keep including more of this services in the FWSM it will fail.

This is not a trivial problem and it is the main reason to look for a new security appliance.

7.2.1. *Different possibilities*

I was asked to do a pre-study of the different possibilities:

- A CISCO module integrated in the chassis.
- An external CISCO device.
- An external device of other brand (for instance JUNIPER).

In this respect, I started to do an evaluation of the different options with which I will keep in mind features like the throughput, security options, price, organization needs, compatibility, etcetera.

At first I took a device of each alternative to compare their features.

Cisco module

ASA-SM with a throughput of 20Gbps, able to manage up to 300.000 connections per second and 10 millions of total connections.

Cisco external device

ASA 5585-X, there are different models of this appliance, the most similar to the ASA-SM is the SSP40, but supports less connection rate and total connections. We can also consider the SSP60 with 40Gbps of throughput, the number of total connections supported than the ASA-SM and a similar connection rate (350.000 conn./sec). These devices are significantly more expensive than the ASA-SM

Other brand device

I took Juniper from my supervisors as a reference brand to study the competitor of Cisco devices. I discovered the **SRX3400** and the **SRX3600**, both with more throughputs, but a smaller amount of total connections and, in case of the SRX3400, less connection rate. The price of these devices is around a 30% lower than the ASA-SM.

7.2.2. Conclusion

But, being simultaneously involved in the Rancid project, the high availability studies and the replacement of the chassis, I had to put off this task to focus on the others.

Finally, after discuss with our provided, my managers and me opted for the Cisco module ASA-SM, having similar features than the external appliance, the price is much lower. The main reasons to take this choice are:

- Easy migration of rules from our former security appliance.
- Same operative system. It is and advantage to keep the same or similar environment and not to spend resources to learn and get used and experienced in a new one.

Also, as a part of the purchase, they included the applications to achieve the migration of all the information from the FWSM to this new device.

8. Conclusion

Concerning my main works here, I want to represent the effort that I have dedicated to each one with the next pie graph:

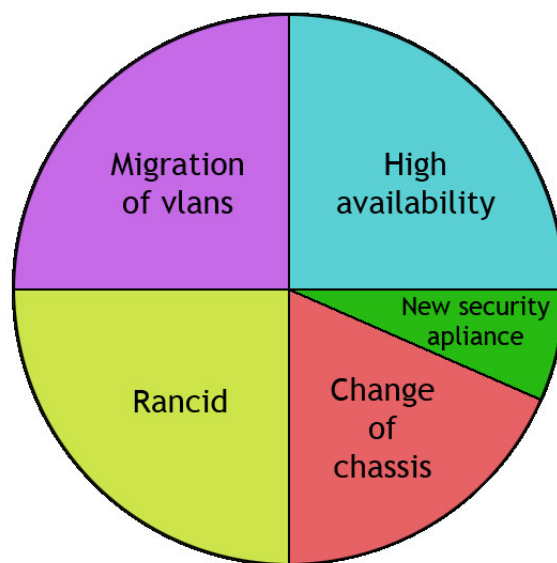


Illustration 17. Proportion of time dedicated to each project.

I spent a lot of time with the migration of the VLANs, besides the preparation and checking of the files, the hardest part was to check all the rules of each network and move them to the firewall.

In Rancid, because at first I didn't know in which machine it was going to be installed, I did a lot of test in virtual machines with different Linux distributions. Moreover, I had to prepare the live presentation in front of my team and, after this, I faced the new requirements that go one step further from the basic configuration.

In the case of the high availability, I dedicate a lot of time to get a comprehensive knowledge about the different options, more specifically in HSRP and VSS, the first one because is the

Initial Process of Updating the Core of a Network

required for our network and the second one because it the most interesting looking to the future.

In the case of the chassis I didn't have to dedicate so much time, since my supervisor took a more active presence in this project due to its delicacy.

The task which I dedicated less time as I explained in the corresponding section is the search of a new security appliance.

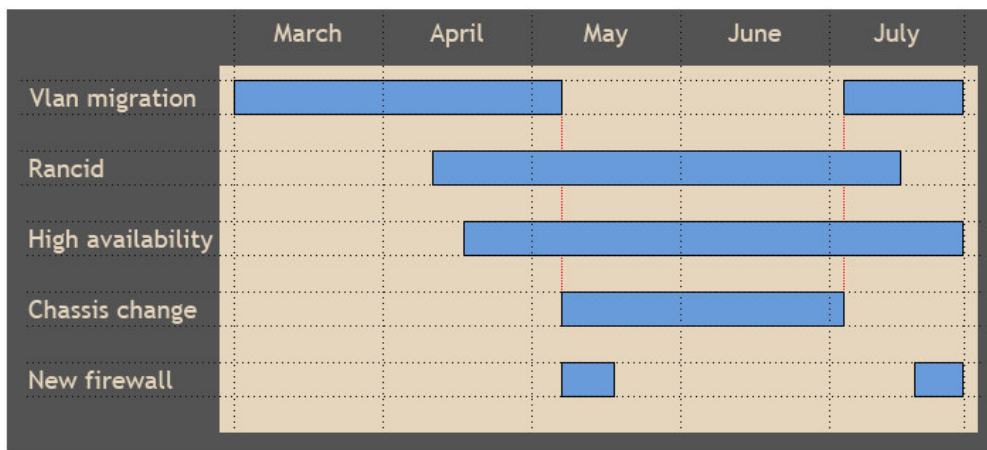


Illustration 18. Gantt chart of the main tasks.

In the Gantt chart there is room to emphasise the first work was already started when I arrived here and it will continue after I leave. Also, as a detail, we can see how during the replacement of the chassis, we stopped migrating VLANs because we wanted to keep the configuration of the chassis as much stable as possible.

During my five months stay in this company I have combined these projects with some routine tasks, like the firewall management, helpdesk, opening and closing accesses depending of the need of the workers of my department or other laboratories. Sometimes I dedicated my work to troubleshooting, with the ticket-tracker website Request Tracker of my department, which has been really interesting, since the problems belonged to different fields: DNS, SNMP, logging servers, mail servers, etc. and tools like Cacti to monitor all the routers, firewalls and switches and their interfaces. These tasks drove me to study some protocols and services that I

had never investigated so deeply before, providing me new knowledges that I am sure will be really useful for my professional future.

Besides my knowledge about the Cisco environment which I could widely extended thanks to an entire year working in its devices, I have seen improved my understanding not just about the different network services and the technical aspects but also about how the public laboratories and organizations work.

Working in the MI2S has been an enriching adventure where I have been able to work in a real environment in a big network, to know the day-to-day work, the experience to be part a project already started and also to complete one from the beginning to the end. And all of this with a good team but also with autonomy, this helped me to grow both professionally and personally.



9. Bibliography

Cisco

Besides the documentation of our provider of the public market, the most common encyclopedias and the internal wiki of the MI2S, I have made an exhaustive use of the official website of Cisco (<http://www.cisco.com>), their official forum mainly for troubleshooting (<https://supportforums.cisco.com>) and more specifically the site of the Catalyst 6500:

<http://www.cisco.com/en/US/products/hw/switches/ps708/index.html>

Rancid

HARRISON Peter,

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO:_Ch1:_Network_Backups_With_Rancid#.Uej2DKySD85

OpenManiak, http://openmaniak.com/rancid_tutorial.php

GIL Victor, <http://virtcommunity.blogspot.fr/2012/08/instalacion-y-configuracion-de-rancid.html>

MAAS Rob, <http://www.progob.nl/robmaaseu/index.php/backing-up-network-devices-with-rancid-opensuse-11-2/>

MAAS Rob, <http://www.progob.nl/robmaaseu/index.php/switch-rancid-to-svn-and-view-your-svn-db-with-websvn/>

KUFEL Szymon, <http://blog.skufel.net/2012/03/how-to-using-websvn-as-rancid-repository-access-tool/>

KUFEL Szymon, <http://blog.skufel.net/2012/01/how-to-adding-devices-to-rancid/>

Subversion:

BRIANO Fernando, <http://picandocodigo.net/downloads/docs/subversion-presentacion-01.pdf>

BRIANO Fernando, <http://picandocodigo.net/downloads/docs/subversion-presentacion-02.pdf>

Centos, <http://wiki.centos.org/es/HowTos/Subversion>

CyberHades, <http://www.cyberhades.com/2010/07/24/instalando-subversion-en-centos/>

Cisco Prime LAN Management Solution:

Cisco, <http://www.cisco.com/en/US/products/ps11200/index.html>

Solarwinds' Network Configuration Manager:

Solarwinds, <http://www.solarwinds.com/network-configuration-manager.aspx>

Replacement of the chassis

Cisco,

http://www.cisco.com/en/US/products/hw/switches/ps5213/products_tech_note09186a0080a49dbf.shtml

Future

HSRP:

RFC2281, <http://www.ietf.org/rfc/rfc2281.txt>

New device:

Juniper, <http://www.juniper.net>

Ugap, <http://www.ugap.fr>

Cisco, <http://www.cisco.com/en/US/products/ps11061/index.html>

Cisco, <http://www.cisco.com/en/US/products/ps11621/index.html>



Others

List of well known ports:

IANA, <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>

Wikipedia, http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Access-lists:

Brocade,

http://www.brocade.com/support/Product_Manuals/ServerIron_SecurityGuide/acls.3.8.html

Sed utility:

BARNETT Bruce, <http://www.grymoire.com/Unix/Sed.html>

Cisco Feature Navigator:

Cisco, <http://tools.cisco.com/ITDIT/CFN/jsp/by-feature-technology.jsp>

10. Glossary

Access Control List (ACL): List of rules to set access permissions to a particular object or network. Normally used to filter traffic in network devices

Demilitarized Zone (DMZ): Network used by the companies and organizations put in it all the services that can be accessed from the internet The security in this region is more opened and it is the network between the public internet and the private and secure network.

Domain Name System (DNS): Hierarchical way to name the machines of a network. Used to call the network objects in a comprehensive way for the people and not be forced to use the Internet Protocol addresses which are made up by a string of four numbers between 0 and 255. This system also provides the way to translate the DNS names into IP addresses.

Putty: Open source software to stablish diverse kind of connections between two machines. It supports SSH and Telnet between others.

Protocol Independent Multicast (PIM): Multicast routing protocol to structure a tree distribution between the multicast clients creating domains.

Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche (RENATER): Computer network that links the different universities and research centres of France. It is the French equivalent to the Spanish REDIRIS network.

Secure Shell (SSH): Protocol and software used to connect to remote machines through a network in a safe way.



Trivial File Transfer Protocol (TFTP): Simply version of the File Transfer Protocol. Useful to exchange files between machines in a network.

Uplink: Connection between equipments of the core of a network.

Virtual Local Area Network (VLAN): LAN where the traffic is tagged with a number that identifies it with a network. With this, the machines in the same local network don't need to be connected to the same switch and we can have more than one LAN working in a single switch.

11. Appendix 1. Script: Check the migration files

```
#!/bin/bash

# Marcos Tortajada Rodriguez: MI2S-MIM, office: bat. CETA, 308

# 05/04/2013

# This script finds the next errors in the preparation files for the VLANs
migration:

# On Migration files:

# - There's not "ip verify unicast ..."

# - There's not "no ip proxy-arp"

# - There's not "no ip unreachable"

# - There's not "ip pim ..."

# - There's not "ip multicast ..."

# - There's not "ntp multicast ..."

# - There's not "mls rp ..."

# - There's not "ip access-group"

# - There is "mtu"

# - There is "ospf cost"

# - There is "security-level"

# - There is "! A la fin, si la migration s'est bien..."

# no ip access-list ..."

# - If there is "ip helper-address <IP>" in r-campus then, in the migration
file:

# + There is the same number of "dhcprelay server" than "helper-address"

# + There is "dhcprelay server <IP>"

# + The IP's of "helper-address" and "dhcprelay server" are equal

# + There is "dhcp enable"
```



```
#
# - If there is "ipv6 ..." in r-campus the, in the migration file:
# + There is "ipv6 address ..."
# + There is "ipv6 enable"
# + There is "ipv6 nd previx"
# ` + At least one "access-group" command contains at least one token "ipv6"
# + There is "no ipv6 access-list ..."
#
#
#
#
# On RetourEnArriere files:
# - There are lines "ip access-group ..."
#
# On both files:
# - There's not "appletalk ..."
# - There's not any # symbol
# - There is "no shutdown"
#
# If there is any problem, an "err.txt" file with information is created
#
VLANSDIR=/tftpboot/routeurs/Parefeux/Campus/temp_Marcos
RCAMPUSFILE=/tftpboot/routeurs/r-campus*
if test $# == 0; then
echo -e "\e[00;34mTesting default directory:\e[00m $VLANSDIR"
```

```

echo -e "\e[00;34mRouter file:\e[00m $RCAMPUSFILE "

elif test "$#" -eq "1"; then

VLANSDIR="$1"

echo -e "\e[00;34mTesting \e[00m $VLANSDIR"

elif test "$#" -gt "1" ; then

echo "Usage: ./failSeeker [DIRECTORY]"

echo This script test all files under the specified directory and its
subdirectories.

echo Default directory:

echo $VLANSDIR

echo Router file:

echo $RCAMPUSFILE

exit

fi

echo " " > err.txt

#####

##### Migration files #####

#####

# Test: There's not "ip verify unicast ..."

VAR_UNICAST=$(grep -Rl "ip verify unicast" $VLANSDIR/* | grep -v "Retour")

if [ -n "$VAR_UNICAST" ]; then

echo ERROR, here you have an \"ip verify unicast\": >> err.txt

echo $VAR_UNICAST | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt

# grep put the name of all the files in a single string, sed split them to
show one file per line

```



```

# this awk avoid to show the entire path of the file

FLAGALL=1

echo " " >> err.txt

echo -e "\e[00;31mERROR in \"ip verify unicast\" test.\e[00m"

else

echo Correct \"ip verify unicast\" test.

fi

# Test: There's not "no ip proxy-arp"
VAR_PROXYARP=$(grep -Rl "no ip proxy-arp" $VLANSRDIR/* | grep -v "Retour")
if [ -n "$VAR_PROXYARP" ]; then
echo ERROR, here you have a \"no ip proxy-arp\": >> err.txt
echo $VAR_PROXYARP | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt
FLAGALL=1
echo " " >> err.txt
echo -e "\e[00;31mERROR in \"no ip proxy-arp\" test.\e[00m"
else
echo Correct \"no ip proxy-arp\" test.
fi

# Test: There's not "no ip unreachable"
VAR_UNREACH=$(grep -Rl "no ip unreachable" $VLANSRDIR/* | grep -v "Retour")
if [ -n "$VAR_UNREACH" ]; then
echo ERROR, here you have a \"no ip unreachable\": >> err.txt
echo $VAR_UNREACH | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt
FLAGALL=1

```

```

echo " " >> err.txt

echo -e "\e[00;31mERROR in \"no ip unreachable\" test.\e[00m"

else

echo Correct \"no ip unreachable\" test.

fi

# Test: There's not "ip pim ..."

VAR_IPPIM=$(grep -Rl "ip pim" $VLANSRDIR/* | grep -v "Retour")

if [ -n "$VAR_IPPIM" ]; then

echo ERROR, here you have a \"ip pim\": >> err.txt

echo $VAR_IPPIM | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt

FLAGALL=1

echo " " >> err.txt

echo -e "\e[00;31mERROR in \"ip pim\" test.\e[00m"

else

echo Correct \"ip pim\" test.

fi

# Test: There's not "ip multicast ..."

VAR_IPMULTICAST=$(grep -Rl "ip multicast" $VLANSRDIR/* | grep -v "Retour")

if [ -n "$VAR_IPMULTICAST" ]; then

echo ERROR, here you have a \"ip multicast\": >> err.txt

echo $VAR_IPMULTICAST | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>>
err.txt

FLAGALL=1

echo " " >> err.txt

echo -e "\e[00;31mERROR in \"ip multicast\" test.\e[00m"

else

```

```

echo Correct \"ip multicast\" test.

fi

# Test: There's not "ntp multicast ..."
VAR_NTPMULTICAST=$(grep -Rl "ntp multicast" $VLANSRDIR/* | grep -v "Retour")
if [ -n "$VAR_NTPMULTICAST" ]; then
echo ERROR, here you have a \"ntp multicast\": >> err.txt

echo $VAR_NTPMULTICAST | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>>
err.txt

FLAGALL=1

echo " " >> err.txt

echo -e "\e[00;31mERROR in \"ntp multicast\" test.\e[00m"

else
echo Correct \"ntp multicast\" test.

fi

# Test: There's not "mls rp ..."
VAR_MLSRP=$(grep -Rl "mls rp" $VLANSRDIR/* | grep -v "Retour")
if [ -n "$VAR_MLSRP" ]; then
echo ERROR, here you have a \"mls rp\": >> err.txt

echo $VAR_MLSRP | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt

FLAGALL=1

echo " " >> err.txt

echo -e "\e[00;31mERROR in \"mls rp\" test.\e[00m"

else
echo Correct \"mls rp\" test.

fi

```



```

# Test: There's not "ip access-group"

VAR_ACCGRP=$(grep -Rl "ip access-group" $VLANSRDIR/* | grep -v "Retour")

if [ -n "$VAR_ACCGRP" ]; then

echo ERROR, here you have a \"ip access-group\": >> err.txt

echo $VAR_ACCGRP | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt

FLAGALL=1

echo " " >> err.txt

echo -e "\e[00;31mERROR in \"ip access-group\" test.\e[00m"

else

echo Correct \"ip access-group\" test.

fi

# Test: There is "mtu"

VAR_MTU=$(find $VLANSRDIR -type f \! -exec grep -q "mtu" {} \; -print | grep -v "Retour")

# In the line below I use "find" because "grep -v" only works on line level
and

# all files have lines without the word "mtu", so all files were matching

if [ -n "$VAR_MTU" ]; then

echo WARNING, here you DON'T HAVE a \"mtu\": >> err.txt

echo $VAR_MTU | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt

FLAGALL=1

echo -e "\e[00;31mERROR in \"mtu\" test.\e[00m"

else

echo Correct \"mtu\" test.

fi

```



```

# Test: There is "ospf cost"

VAR_OSPF=$(find $VLANSRDIR -type f \! -exec grep -q "ospf cost" {} \; -print |
grep -v "Retour")

# In the line below I use "find" because "grep -v" only works on line level
and

# all files have lines without the word "ospf cost", so all files were
matching

if [ -n "$VAR_OSPF" ]; then

echo WARNING, here you DON'T HAVE an \"ospf cost\": >> err.txt

echo $VAR_OSPF | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt

FLAGALL=1

echo -e "\e[00;31mERROR in \"ospf cost\" test.\e[00m"

else

echo Correct \"ospf cost\" test.

fi

# Test: There is "security-level"

VAR_SECURITYLVL=$(find $VLANSRDIR -type f \! -exec grep -q "security-level" {}
\; -print | grep -v "Retour")

# In the line below I use "find" because "grep -v" only works on line level
and

# all files have lines without the word "security-level", so all files were
matching

if [ -n "$VAR_SECURITYLVL" ]; then

echo WARNING, here you DON'T HAVE a \"security-level\": >> err.txt

echo $VAR_SECURITYLVL | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>>
err.txt

FLAGALL=1

echo -e "\e[00;31mERROR in \"security-level\" test.\e[00m"

```

```

else

echo Correct \"security-level\" test.

fi

# Test: There is \"! A la fin, si la migration s'est bien...\"

VAR_BIEN=$(find $VLANS DIR -type f \\! -exec grep -q \"! A la fin, si la
migration \" {} \\; -print | grep -v \"Retour\")

VAR_NOIPACL=$(find $VLANS DIR -type f \\! -exec grep -q \"no ip access-list\" {}
\\; -print | grep -v \"Retour\")

if [ -n \"$VAR_BIEN\" ] && [ -n VAR_NOIPACL ]; then

echo WARNING, here you DON'T HAVE an \"! A la fin, si la migration s'est
bien...\": or there isn't any \"no ip access-list\": >> err.txt

echo $VAR_BIEN | sed 's/ /\n/g' | awk -F/ '{print \" - \" $NF}'>> err.txt

FLAGALL=1

echo -e \"\e[00;31mERROR in \"!A la fin, si la migration s'est bien
passee...\" test.\e[00m\"

else

echo Correct \"!A la fin, si la migration s'est bien passee \" test.

fi

#####

##### RetourEnArriere files #####

#####

#Test: There is \"ip access-group\"

VAR_ACCESSGROUP=$(find $VLANS DIR -type f \\! -exec grep -q \"ip access-group\"
{} \\; -print | grep \"Retour\")

```



```

if [ -n "$VAR_ACCESSGROUP" ]; then

echo WARNING, here you DON'T HAVE an \"ip access-group\": >> err.txt

echo $VAR_ACCESSGROUP | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>>
err.txt

FLAGALL=1

echo -e "\e[00;31mERROR in \"ip access-group\" test.\e[00m"

else

echo Correct \"ip access-group\" test.

fi

#####

##### Both files #####

#####

# Test: There's not "appletalk..."
VAR_APPLE=$(grep -Rl "appletalk" $VLANSRDIR/*)

if [ -n "$VAR_APPLE" ]; then

echo ERROR, here you have an \"appletalk\": >> err.txt

echo $VAR_APPLE | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt

FLAGALL=1

echo -e "\e[00;31mERROR in \"appletalk\" test.\e[00m"

else

echo Correct \"appletalk\" test.

fi

```

```

# Test: There's not sharp/hash/number symbol (#)

VAR_HASH=$(grep -Rl "#" $VLANSDIR/*)

if [ -n "$VAR_HASH" ]; then

echo ERROR, here you have a "\"#\": >> err.txt

echo $VAR_HASH | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>> err.txt

FLAGALL=1

echo -e "\e[00;31mERROR in "\"#\\" test.\e[00m"

else

echo Correct "\"#\\" test.

fi

# Test: There is "no shutdown"

VAR_NOSHUTDOWN=$(find $VLANSDIR -type f \! -exec grep -q "no shutdown" {} \;
-print)

if [ -n "$VAR_NOSHUTDOWN" ]; then

echo WARNING, here you DON'T HAVE an "\"no shutdown\": >> err.txt

echo $VAR_NOSHUTDOWN | sed 's/ /\n/g' | awk -F"/" '{print " - " $NF}'>>
err.txt

FLAGALL=1

echo -e "\e[00;31mERROR in "\"no shutdown\" test.\e[00m"

else

echo Correct "\"no shutdown\" test.

fi

```



```
#####
##### Check Network IP #####
#####

echo Checking IP... >> err.txt

for f in $(ls $VLANSDIR)
do
echo "Processing $f..."
echo "Processing $f..." >> err.txt

#Get the vlan number (for instance: Vlan3)
VLANNUM=$(echo $f | awk -F "-" '{print $2}' | awk -F "_" '{print $1}' | grep -o '[1-9].*\{2,\}')

VLANNUM=Vlan$VLANNUM

#Get the IP of the GW and the MASK of a Vlan from the r-campus file
LINERou=$(grep $VLANNUM$ $RCAMPUSFILE -A2 -m1 | tail -n1)
GWADDrou=$(echo $LINERou | awk -F " " '{print $3}' | tr -d [:cntrl:])
MASKrou=$(echo $LINERou | awk -F " " '{print $4}' | tr -d [:cntrl:])
EXTENSION=".txt"

#Get the IP of the GW and the MASK of a Vlan from the Migration file
MIGRATIONFILE=$VLANSDIR/$f/$f$EXTENSION
LINEmig=$(grep 'ip address' $MIGRATIONFILE )
GWADDmig=$(echo $LINEmig | awk -F " " '{print $3}' | tr -d [:cntrl:])
MASKmig=$(echo $LINEmig | awk -F " " '{print $4}' | tr -d [:cntrl:])

#Get the IP of the GW and the MASK of a Vlan from the RetourEnArrier file
REAFILE=$VLANSDIR/$f/RetourEnArriere-$f$EXTENSION
```

```

LINERet=$(grep 'ip address' $REAFILE )
GWADDret=$(echo $LINERet | awk -F" " '{print $3}' | tr -d [:cntrl:])
MASKret=$(echo $LINERet | awk -F" " '{print $4}' | tr -d [:cntrl:])

if [[ "$GWADDrou" != "$GWADDmig" ]]; then

echo $VLANNUM: ERROR, the IP of the router file and the migration file is
different >> err.txt

echo router gw $GWADDrou >> err.txt

echo migration gw $GWADDmig >> err.txt

FLAGALL=1

fi

if [[ "$MASKrou" != "$MASKmig" ]]; then

echo $VLANNUM: ERROR, the MASK of the router file and the migration file is
different >> err.txt

echo mask router $MASKrou >> err.txt

echo mask migration $MASKmig >> err.txt

FLAGALL=1

fi

if [[ "$GWADDrou" != "$GWADDret" ]]; then

echo $VLANNUM: ERROR, the IP of the router file and the RetourEnArriere file
is different >> err.txt

echo router gw $GWADDrou >> err.txt

echo retour en arriere gw $GWADDret >> err.txt

FLAGALL=1

fi

```



```
if [[ "$MASKrou" != "$MASKret" ]]; then

echo $VLANNUM: ERROR, the MASK of the router file and the RetourEnArriere
file is different >> err.txt

echo mask router $MASKrou >> err.txt

echo mask retour en arriere $MASKret >> err.txt

FLAGALL=1

fi

# echo ipRou $GWADDrou
# echo ipMig $GWADDmig
# echo ipRet $GWADDret
# echo maskRou $MASKrou
# echo maskMIG $MASKmig
# echo maskRet $MASKret

##### Test: IP HELPER

# Check that there is the same number of "helper-address" on the r-campus
file than

# "dhcprelay server" on the migration file

#sed -n '/Vlan69/,/!/p' /tftpboot/routeurs/r-campus-20130403-01 | grep
helper-address

numHelper=$(sed -n '/'$VLANNUM'$/,/!/p' $RCAMPUSFILE | grep "helper-address"
| wc -l)

# if [[ -n "$numHelper" ]]; then

# FLAGHELPER=1

# else

# FLAGHELPER=0

# fi
```



```

numRelay=$(grep "dhcprelay server" $MIGRATIONFILE | wc -l)

if [[ "$numHelper" != "$numRelay" ]]; then

echo $VLANNUM: ERROR, the number of helper-address and dhcprelay is not the
same >> err.txt

echo Helper address commands: $numHelper >> err.txt

echo Dhcp relay commands: $numRelay >> err.txt

FLAGALL=1

fi

# If there is any dhcprelay, we must check that there is also a "dhcprelay
enable":

enableDHCP=$(grep "dhcprelay enable" $MIGRATIONFILE)

if [[ "$numRelay" != "0" ]] && [[ -z "$enableDHCP" ]]; then

echo $VLANNUM: WARNING, dhcprelay is not being enabled >> err.txt

FLAGALL=1

fi

dhcpLine=$(sed -n '/'$VLANNUM'$/,/!/p' $RCAMPUSFILE | grep "helper-address" )

dhcpAddresses=$(echo $dhcpLine| awk -F" " '{for(x=1;x<=NF;x++)if(x % 3 ==
0)print $x}')

for dhcpAddress in $dhcpAddresses

do

testAddress=$(grep $dhcpAddress $MIGRATIONFILE)

if [[ -z $testAddress ]]; then

echo $VLANNUM: ERROR, one helper-address is not in the migration file or it is
wrong >> err.txt

echo Helper address: $dhcpAddress >> err.txt

FLAGALL=1

fi

done

##### Fin Test: IP HELPER

```



```

##### Test: IPv6

# There's the same number of "traffic-filter" than "access-group" with the
token "ipv6"

numTraffic=$(sed -n '/'$VLANNUM'$/,/!/p' $RCAMPUSFILE | grep "ipv6 traffic-
filter" | wc -l)

numAclIpv6=$(grep "access-group" $MIGRATIONFILE | grep -i "ipv6" | wc -l) #
grep -i -> ignore upper/lower case

if [[ "$numTraffic" != "$numAclIpv6" ]]; then

echo $VLANNUM: ERROR, the number of traffic-filter and access-group with the
token "ipv6" is not the same >> err.txt

echo Traffic-filter commands: $numTraffic >> err.txt

echo Access-group with "ipv6": $numAclIpv6 >> err.txt

FLAGALL=1

fi

# and there is the same number of no ipv6 access-list ..."

numNoAclIpv6=$(grep "no ipv6 access-list" $MIGRATIONFILE | wc -l)

if [[ "$numNoAclIpv6" != "$numAclIpv6" ]]; then

echo $VLANNUM: ERROR, the number of access-group and "no ipv6 access-list
..." is not the same >> err.txt

echo Access-group with "ipv6": $numAclIpv6 >> err.txt

echo "no ipv6 access-list ...": $numNoAclIpv6 >> err.txt

FLAGALL=1

fi

# If there is any ipv6, we must check that there is also "ipv6 address",
"ipv6 enable", "ipv6 nd prefix":

addressIpv6=$(grep "ipv6 address" $MIGRATIONFILE)

enableIpv6=$(grep "ipv6 enable" $MIGRATIONFILE)

prefixIpv6=$(grep "ipv6 nd prefix" $MIGRATIONFILE)

if [[ "$numAclIpv6" != "0" ]] && [[ -z "$addressIpv6" ]] && [[ -z
"$enableIpv6" ]] && [[ -z "$prefixIpv6" ]]; then

```

```

echo $VLANNUM: WARNING, one of the next commands is not declared: "ip
address", "ipv6 enable" or "ipv6 nd prefix" >> err.txt

FLAGALL=1

fi

# The IPv6 address is the same in both files

ipv6Line=$(sed -n '/'$VLANNUM'$/,/!/p' $RCAMPUSFILE | grep "ipv6 address" )

rcampusIpv6=$(echo $ipv6Line | awk -F" " '{print$3}')

testIpv6Address=$(echo $addressIpv6 | grep $rcampusIpv6 $MIGRATIONFILE)

if [[ -n $ipv6Line ]] && [[ -z $testIpv6Address ]]; then # There's IPv6 on r-
campus && It's the same IP no migration file

echo $VLANNUM: ERROR, one ipv6 address is not in the migration file or it is
wrong >> err.txt

echo ipv6Line: $ipv6Line >> err.txt

echo IPv6 address: $rcampusIpv6 >> err.txt

FLAGALL=1

fi

##### Fin Test: IPv6

done

if [ $FLAGALL ]; then

echo -e "\n""\e[00;31mThere is at least one problem, check the err.txt
file.\e[00m ""\n"

else

echo -e "\n""\e[00;32mAll correct.\e[00m""\n"

rm err.txt

fi

echo Execution concluded

```

