SAE4.01B : Cassiopea

Livrable 5 : Rapport final

25 Mars 2025

Félix Grandet, Quentin Peguin, Tristan Fontanière, Manohisoa Mbelo Ndriamanampy, Hashem Belal
SAE4.01 : Parcours B

Table des matières

Table des figures

1 Introduction et contexte

Dans le cadre de notre projet, nous avons pour mission de mettre en place une infrastructure réseau. Nous avons choisi de contextualiser ce sujet en nous donnant pour mission de réaliser une architecture qui pourrait correspondre à celle d'un IUT au sein de l'UGA, destinée à des utilisateurs finaux tels que des enseignants et des étudiants. Cette infrastructure doit respecter des normes de structure, de choix de solutions et de sécurité, tout en utilisant des logiciels open-source. Les services à déployer incluent DNS, DHCP, authentification, accès à un intranet et accès externe, avec la possibilité de supporter différents systèmes d'exploitation. Côté administrateur, nous devons avoir une machine qui peut maintenir à distance les différents serveurs et un wiki qui contient toutes les spécifications techniques de notre infrastructure.

Pour le nom du projet, nous avons choisi “Cassiopea", en référence à la constellation Cassiopée, symbolisant les réseaux d'étoiles complexes, en continuité avec le service Leo de l'UGA.

Ce document présente l'architecture que nous avons pu réaliser au terme des semaines de SAE, nous revenons sur les détails techniques en terme de ressource, de configuration, des tests et des mesures de sécurité prises.

2 Rappel de l'architecture

La version finale de notre architecture est visible sur le schéma ci-dessous.

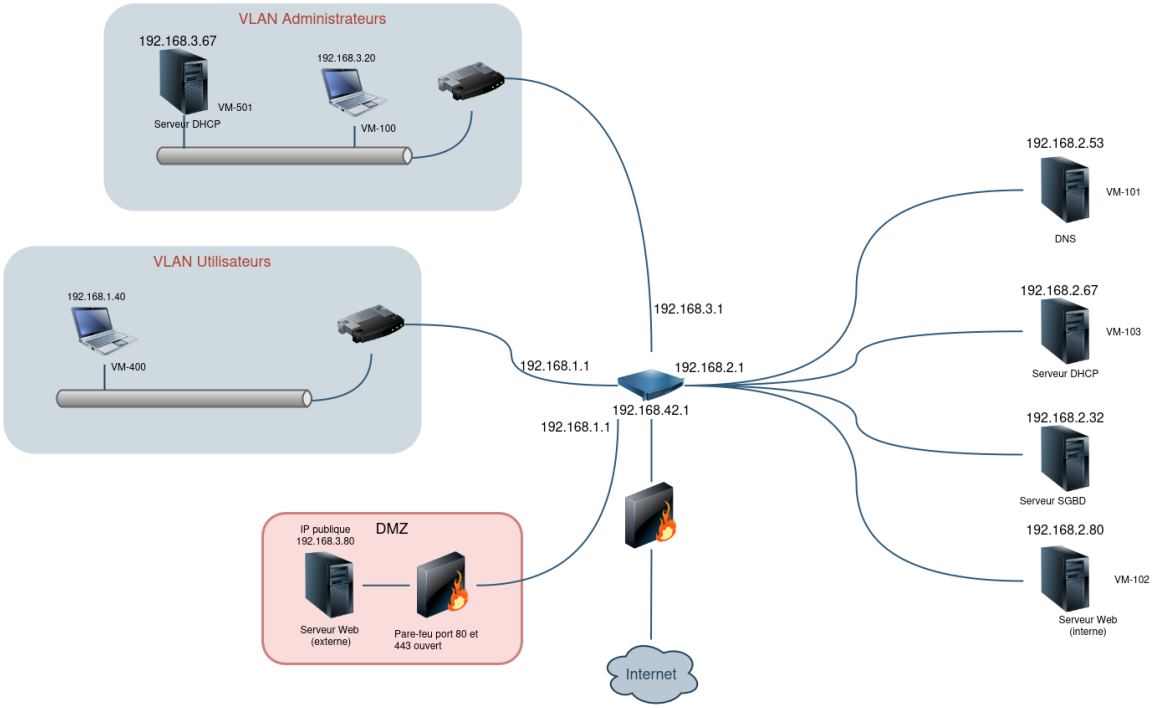

FIGURE 1 – Architecture réseau

Quelques changements sont notables concernant l'architecture depuis le livrable 1 :

- Le VLAN Windows n'a finalement pas été mis en place, cela n'était pas pertinent et nous avons préféré nous limiter à un réseau utilisateur utilisant des systèmes Linux.
- Le service DHCP a été divisé en deux : un dhcp pour les administrateurs et un pour les utilisateurs.
- Le logiciel utilisé pour le wiki n'est finalement pas Pmwiki mais Mediawiki. Pmwiki ne permettait pas de réaliser une authentification LDAP simplement, nous avons donc essayé avec les extensions proposées par Mediawiki. Au final, notre wiki utilise Apache pour réaliser l'authentification LDAP, la solution proposée par Apache est bien plus simple à mettre en place et offre une sécurité très satisfaisante car le client n'a jamais accès aux échanges entre le serveur web et le serveur LDAP.
  Nous avons donc gardé Mediawiki car son installation est plus simple et il offre de nombreuses fonctionnalités supplémentaires et est mieux documenté ce qui facilite sa maintenabilité.

3 Ressources utilisées

Dans son état final notre réseau utilise tous les hyperviseurs que nous avions à disposition en ayant presque atteint la limite de stockage. En faisant la somme des ressources utilisées par toutes les machines nous avons :

— 19 machines ayant presque toutes 3Go de disque (à l'exception de certains serveurs comme le wiki qui nécessite 6Go de disque).

— Au niveau des CPU nous avons laissé un coeur sur toutes les machines sauf celle en graphique qui nécessite plus de performances.

— 42 Go de RAM réservé au total pour les machines, à noter que toutes les machines n'utilisent pas leur mémoire à leur limite, par exemple les serveurs DHCP utilisent environ 800Mo de mémoire.

— Au total nous avions à disposition 100Go de stockage en additionnant nos hyperviseurs, au final nous utilisons 93Go pour nos machines, même si ce stockage n'est pas rempli il est réservé pour les machines.

4 Documentation technique

Nous allons revenir sur la configuration des différents éléments de notre réseau : les machines et la structure du réseau. Pour certains serveurs, la configuration est proposée automatiquement par des scripts qui sont disponibles dans le livrable 4.

4.1 Les sous réseaux

Actuellement quatre sous réseaux : réseau utilisateur, réseau serveurs, réseau administrateurs et réseau DMZ. Pour mieux comprendre leur configuration, se référer au tableau suivant.

Nom réseau	CIDR	Masque	Gateway	Type	Protocole
Utilisateurs	192.168.1.0/24	255.255.255.0	192.168.1.1	VLAN	IPv4
Serveurs	192.168.2.0/24	255.255.255.0	192.168.2.1	VLAN	IPv4
Administrateurs	192.168.3.0/24	255.255.255.0	192.168.3.1	VLAN	IPv4
DMZ	192.168.4.0/24	255.255.255.0	192.168.4.1	VLAN	IPv4
Routeurs	192.168.10.0/24	255.255.255.0	192.168.4.1	VLAN	IPv4

Côté Promox la configuration est divisé en deux parties : le vxlan et les vnet. Nous avons d'abord déclaré une zone globale appelé "Vxlan1" qui peut contenir des vnet. Ensuite, nous avons créé autant de vnets que nous avions besoin, un vnet représente

un sous réseau des routeurs (voir schéma réseau). Pour que tous les hyperviseurs puissent utiliser ces réseaux et faire communiquer les différentes machines virtuelles, nous déclarons l'adresse "publique" des hyperviseurs dans la section "peer address list" des hyperviseurs.
Par exemple, le serveur Web du wiki utilise le vnet1 (sous réseau 192.168.2.0), il contacte le routeur principal par son adresse 192.168.2.1 pour accéder à un internet .
La configuration à faire sur l'interface graphique est donc la suivante.

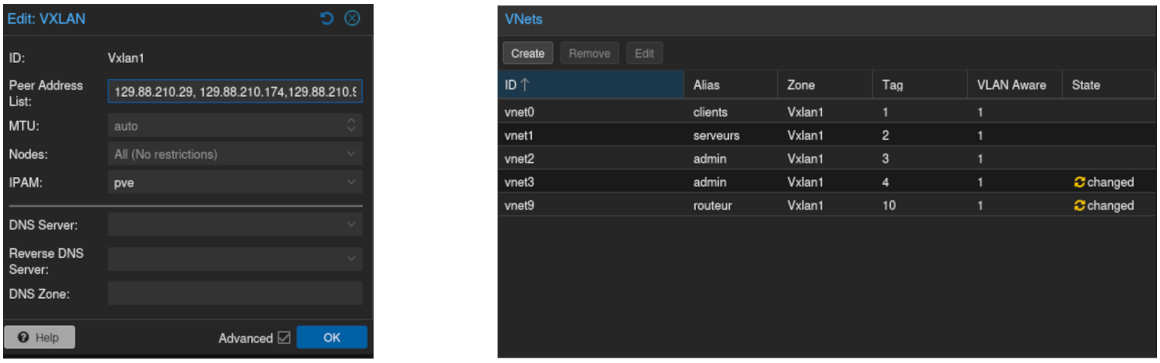

FIGURE 2 – Configuration Proxmox

4.2 Les routeurs

4.2.1 Le routeur principal

Notre routeur sert à relier tous les sous-réseaux, c'est lui qui met en place les règles de pare-feu les plus importantes et défini qui peut connecter avec qui et comment. Il a donc 5 interfaces réseau : 4 sous réseaux et une interface permettant d'accéder à internet.
Sa configuration est la suivante :

- 5 interfaces réseaux :
  - ens18 : 192.168.1.0/24 → réseau utilisateurs.
  - ens19 : 192.168.42.0/24 → chemin vers internet.
  - ens20 : 192.168.2.0/24 → réseau serveurs.
  - ens21 : 192.168.3.0/24 → réseau administrateurs.
  - ens22 : 192.168.10.0/24 → réseau routeurs.
- NAT activé pour permettre les échanges entre les sous réseaux
  - configuration du pare-feu pour accepter le nat : `/usr/local/bin/nat` et `/etc/nftables.conf` (fichiers qui seront disponibles sur le git à terme).

— activer l'ip forward de la machine : `/proc/sys/net/ipv4/ip_forward` (valeur 1 à activer).
— Activer avec la commande `sysctl -p`.

4.2.2 Le routeur DMZ

Le routeur que nous appelons DMZ est celui qui est relié à la DMZ, nous avons décidé de l'installer pour fournir une couche en plus de protection pour le réseau interne. De cette façon, les requêtes faites pour le serveur web de la DMZ n'atteignent jamais le réseau internet et les règles de pare-feux permettent de grandement limiter les attaques internes en passant par ce serveur Web.

4.3 Les serveurs DHCP

Les deux serveurs DHCP de notre infrastructure ont un fonctionnement assez différent, revenons d'abord sur le serveur distribuant des IP aux clients.
Sa configuration :

- Adresse : 192.168.2.67 (en rappel au port utilisé).
- Sur 192.168.1.0/24 : 192.168.1.50 → 192.168.1.200, le réseau utilisateur doit contenir un grand nombre de machines nous gardons donc toutes les adresses.
- Sur 192.168.2.0/24 : Pas de DHCP, les serveurs ont besoin d'adresses fixes pour garantir le fonctionnement des services constamment.
- Sur 192.168.3.0/24 : aucune IP, un autre serveur s'en charge.

Procédure de configuration :

- Sur le serveur installer `kea-dhcp`, le contenu du fichier de configuration `/etc/-kea/kea-dhcp4.conf` est le suivant :

kea-dhcp4 configuration

```
{
  "Dhcp4": {
        "interfaces-config": {
           "interfaces": ["ens18"]
         },
         "lease-database": {
            "type": "memfile",
            "lfc-interval": 3600
         },
         "subnet4": [
           {
             "subnet": "192.168.1.0/24",
             "pools": [
                { "pool": "192.168.1.50 - 192.168.1.200" }
             ],
              "relay": {
                "ip-addresses": ["192.168.1.1"]
              },
              "option-data": [
                {
                  "name": "routers",
                  "data": "192.168.1.1"
                },
                {
                  "name": "domain-name-servers",
                  "data": "192.168.2.53, 152.77.1.22"
                }
              ]
            }
          ]
        }
      }
```

- Sur le routeur, installer isc-dhcp-relay cela permet au routeur de redistribuer les requêtes discover au serveur que l'on souhaite, encore une fois la configuration de /etc/default/isc-dhcp-relay est la suivante :

isc-dhcp configuration

```
#
# This is a POSIX shell fragment
#

# IP of the DHCP server you want to focus
SERVERS="192.168.2.67"

# Interface that correspond to the network of the DHCP server
INTERFACES="ens18 ens20"

# Additional options
OPTIONS=""
```

- Enfin, il faut configurer les machines clients de telle sorte qu'elle récupère leur IP à travers un serveur DHCP. Pour cela, il faut éditer le fichier `/etc/network/interfaces` et appliquer une configuration similaire à celle-ci :

configuration interfaces

```
# The loopback network interface
auto lo
iface lo inet loopback

# The network interface on 192.168.1.0/24
auto ens18
iface ens18 inet dhcp
```

Nous avons également un DHCP spécifique au réseau administrateur qui se trouve dans le même réseau que les machines qu'il sert.
Sa configuration :

- Adresse : 192.168.3.67
- Sur 192.168.3.0 : 192.168.3.20 → 192.168.3.50, un réseau avec moins de machines a besoin de moins d'adresses.

Procédure de configuration :

- Même structure que le serveur des clients en adaptant les adresses et interfaces.

4.4 Les serveurs DNS

4.4.1 Le serveur DNS interne

Le serveur DNS interne est utile à la résolution de noms des machines auxquelles nous accédons à l'intérieur du réseau à savoir l'intranet, le serveur base de données et le wiki.
Sa configuration :

- Adresse : 192.168.2.53 (en rappel au port utilisé).
- Utilise le logiciel bind9 intégré aux systèmes Linux de base.

Procédure de configuration :

- Avant toute chose il faut configurer le fichier named.conf.local qui indique au serveur DNS quelles zones il doit gérer et dans quels fichiers il doit trouver cette configuration. La configuration utilisée ici est la suivante :

named.conf.local

```
//
// Do any local configuration here
//

zone "cassiopea.fr" {
    type master;
    file "/etc/bind/db.cassiopea.fr";
};
```

- Créer et configurer un fichier `/etc/bind/cb.cassiopea.fr` qui définit les différentes machines du domaine cassiopea.fr de la façon suivante :

db.cassiopea.fr

```
TTL    604800
@       IN      SOA     ns1.cassiopea.fr. admin.cassiopea.fr. (
                          2023101001         ; Serial
                          604800             ; Refresh
                          86400              ; Retry
                          2419200            ; Expire
                          604800 )           ; Negative Cache TTL
;
@       IN      NS      ns1.cassiopea.fr.
@       IN      A       192.168.2.53         ; Adresse DNS
ns1     IN      A       192.168.2.53         ; Adresse DNS
wiki    IN      A       192.168.2.43         ; Adresse  wiki
web     IN      A       192.168.2.80         ; Adresse intranet
pgsql   IN      A       192.168.2.32         ; Adresse  BDD
```

- En cas d'ajout d'une machine importante comme un autre service web il suffit de rajouter une ligne qui suit le schéma de celle qui indique le wiki ou la base de données.

4.4.2 Le serveur DNS DMZ

À l'instar du DNS interne, le DNS de la DMZ définit les noms des machines du domaine public, dans notre situation cela pourrait être le site de présentation du département informatique. Sa configuration est identique à celle du DNS interne, seules les IP changent. La machine web qui héberge le site public est appelé extern.cassiopea.fr.

4.5 Les serveurs Web

Au sein de notre réseau interne nous avons deux serveurs Web avec chacun des droits spécifiques que nous allons préciser. L'utilité de ces serveurs dépend des utilisateurs : l'un est réservé aux administrateurs du réseau tandis que l'autre est pour les utilisateurs. En se référant à notre situation, le wiki pourrait être un site spécifique à une équipe de support informatique et l'intranet une sorte de chamilo à la simple exception qu'il n'est pas accessible depuis l'extérieur.

4.5.1 L'intranet

Tout d'abord le serveur web intranet qui par exemple donne aux étudiants l'accès à leurs données personelles.
Sa configuration :

- Adresse fixe : 192.168.2.80.
- Protocole utilisé : https avec certificat auto-signé.

Procédure de configuration :

- Il faut tout d'abord installer Apache sur le serveur.
- Il faut activer le support du https avec `a2enmod ssl` et activer le site de base qui supporte la connexion sécurisée avec `a2ensite default-ssl.conf`.
- Ensuite il suffit d'installer les pages que l'on souhaite, dans notre cas nous avons laissé le fichier par défaut à titre d'exemple.
- Aucune configuration spécifique ici puisqu'il s'agit d'un serveur assez basique. Les évolutions de ce serveur seront précisées dans la partie 7.2.

4.5.2 Le wiki

Le wiki est utilisé uniquement par les administrateurs du système, c'est lui qui a pour vocation de documenter la configuration réseau actuelle et qui sert de support aux nouveaux administrateurs pour prendre la main sur l'infrastructure. Le logiciel utilisé est Mediawiki, comme dit précédemment, c'est le logiciel qui nous semblait le plus complet et le plus simple d'utilisation, il a également une très bonne réputation sur internet ce qui nous aide à faire confiance à ce support.

Afin de restreindre l'accès aux administrateurs deux mesures sont mises en place : une authentification liée à un serveur LDAP et un pare-feu qui autorise les connexions seulement depuis le réseau administrateur.

Sa configuration :

- Adresse fixe : 192.168.2.43
- Protocole utilisé : https avec certificat créé et auto-signé (contrairement à l'intranet qui utilise le certificat par défaut de Apache).
- Mediawiki 1.39 avec seulement les extensions par défaut.
- Apache avec les mods `auth_cas` et `authn_basic`

Procédure de configuration :

- Il faut répéter les étapes de configuration du serveur intranet.
- Ensuite il faut installer son wiki en suivant les étapes décrites sur ce guide. Le guide est assez bien guidé. Il existe cependant deux alternatives au téléchargement d'archive :

1. Cloner le dépôt git : dans ce cas il faut choisir la branche REL1_43 qui est la branche stable et déclarée en LTS (long time support) par les développeurs Mediawiki. Les branches Mediawiki encore mises à jour par les développeurs sont visibles sur ce tableau à l'exception de la 1.40 et 1.41 qui ont arrêté d'être suivies.

Info page	Release notes		Branch points	
	Wiki	Git	Branch	Date branched from master
MediaWiki 1.44	Wiki	Git	master	N/A
MediaWiki 1.43 LTS	Wiki	Git	REL1_43	22 October 2024
MediaWiki 1.42	Wiki	Git	REL1_42	9 April 2024
MediaWiki 1.41	Wiki	Git	REL1_41	9 October 2023
MediaWiki 1.40	Wiki	Git	REL1_40	13 March 2023
MediaWiki 1.39 LTS	Wiki	Git	REL1_39	6 September 2022

FIGURE 3 – Branches Mediawiki

2. Télécharger depuis les packages Debian : dans ce cas seules les versions 1.38 et 1.39 sont disponibles sur bookworm. Il suffit donc d'utiliser la commande `apt install mediawiki`

- Une fois les packages présents dans le dossier `/var/www/html/mediawiki` il suffit d'ouvrir le site en graphique et suivre les étapes demandées. La base de données utilisée sera détaillée plus tard.
- Concernant la configuration du site côté Apache il faut préciser deux choses : rediriger automatiquement wiki.cassiopea.fr vers `/var/www/html/mediawiki/index.php` et permettre l'utilisation de .htaccess. Pour cela il faut créer une copie de `default-ssl.conf` que l'on nommera `cassiopeawiki.conf`. Dans ce fichier il faut alors modifier/ajouter les lignes suivantes :

cassiopeawiki.conf

```
    RedirectMatch ^/$ /mediawiki/
    <Directory /var/www/html/mediawiki>
                AllowOverride All
                Require all denied
        </Directory>
  SSLCertificateFile      /etc/ssl/wiki.crt
  SSLCertificateKeyFile   /etc/ssl/wiki.key
```

La redirection est faite par RedirectMatch et AllowOverride All permet d'utiliser le .htaccess. Les fichiers SSLCertificate sont des certificats que nous avons généré mais nous pouvons très bien utiliser les certificats par défaut de Apache.

- Le fichier .htaccess est la dernière étape car c'est ce qui permet à Apache d'interroger le serveur LDAP. La configuration que nous utilisons est la suivante :

htaccess du wiki

```
Options Indexes FollowSymlinks
AuthType Basic
AuthName "Apache LDAP authentication"
AuthBasicAuthoritative Off
AuthBasicProvider ldap
AuthLDAPURL "ldap://192.168.2.70/ou=users,dc=web,dc=interne,dc=fr \
?uid?sub"
AuthLDAPBindDN "uid=cas-reader,ou=users,dc=web,dc=interne,dc=fr"
AuthLDAPBindPassword "UserC@s35."
Require valid-user
```

4.6 Les clients et administrateurs

Nous avons 3 machines qui permettent de tester les serveurs : une machine administrateur en CLI, une machine étudiant en CLI et une machine administrateur graphique. Leur configuration est assez simple car elle permet de réaliser les scripts de tests de disponibilité, de pare-feu et l'accès aux différents sites.

Configuration administrateur CLI :

- Adresse dynamique obtenue grâce au DHCP administrateur.
- Différents packages sont installés comme nettools, tshark ou nmap.

- Cette machine possède une clé permettant de se connecter à tous les serveurs du réseau, c'est la seule machine qui en est capable.

Configuration adminstrateur graphique :

- Adresse IP dynamique également.
- Possède des logiciels graphiques comme Firefox pour réaliser la maintenance du wiki ou wireshark en cas de besoin.

Configuration machine étudiant CLI :

- Adresse IP dynamique obtenu depuis le DHCP client.
- Possède les mêmes scripts de tests que la machine administrateur pour vérifier que les tests fonctionnent quelque soit le réseau.

4.7 Serveur NFS

4.8 Serveur Postgres

4.9 Serveur de journalisation global

Le serveur de journalisation permet de centraliser tous les journaux (logs) système des machines du réseau. Cela facilite la supervision, le diagnostic des erreurs, et la traçabilité des événements. En cas d'incident de sécurité ou de dysfonctionnement, il devient plus simple de retracer les actions effectuées sur l'ensemble du réseau à partir d'un seul point de consultation.
Configuration :

- Adresse : 192.168.2.11
- port utilisé : le port TCP 19532
- Utilise le logiciel systemd-journal-upload intégré aux systèmes Linux de base.
- Il faut vérifier l'installation de systemd-journal-upload sur les machines et les serveurs.
- Il faut ensuite configurer le port par lequel passeront les logs
- par faute de temps les logs sont en http et non en https
- Amélioration future : Avoir un certificat ssl pour passer les échanges en HTTPS.

4.10 Serveur LDAP/CAS

5 Bilan des tests

Nous avons pu réaliser différents tests sur notre infrastructure qui ont tous été donnés dans le livrable 3. Durant toute la durée du projet nous avons réalisé régulièrement les tests de disponibilité que ce soit aux serveurs web, la résolution de nom ou l'attribution d'adresses par DHCP. Nous avons également testé les pare-feux, par exemple nous avons vérifié que le réseau client ne pouvait pas interroger le serveur web du wiki ou que seuls les administrateurs pouvaient se connecter en ssh quelque soit la destination.
Nous allons donc revenir sur le résultat des différents aux termes du projet à travers des traces d'éxecution correspondant aux scripts du livrable 3 :

- Test de réponses ICMP

```
root@cassiopeaadmin:~/test/Connectivity/connectivity_test# ./connectivity_ping_test.sh
Test de ping vers 192.168.3.21...
✓ Connectivité OK avec 192.168.3.21
  - Temps moyen: 0.611 ms
  - Paquets reçus: 4
  - Paquets perdus: 0%
----------------------------------------
Test de ping vers 192.168.2.80...
✓ Connectivité OK avec 192.168.2.80
  - Temps moyen: 2.242 ms
  - Paquets reçus: 4
  - Paquets perdus: 0%
----------------------------------------
Test de ping vers 192.168.2.32...
✓ Connectivité OK avec 192.168.2.32
  - Temps moyen: 2.099 ms
  - Paquets reçus: 4
  - Paquets perdus: 0%
----------------------------------------
Test de ping vers 192.168.1.255...
✗ Échec de la connectivité avec 192.168.1.255
Aucun paquet reçu.
----------------------------------------
Tests de connectivité terminés.
```

FIGURE 4 – Test ping

- Test de routage

```
root@cassiopeaadmin:~/test/Disponibility/disponibility# ./../../Connectivity/connectivity_test/connectivity_routing_test.sh
===          Routage           ===
---------------------------------
Test de routage vers 192.168.2.43...
Routage OK avec 192.168.2.43
  - Nombre de sauts: 3
  - Temps moyen: 128.112 ms
---------------------------------
Test de routage vers 192.168.2.80...
Routage OK avec 192.168.2.80
  - Nombre de sauts: 3
  - Temps moyen: 128.112 ms
---------------------------------
Tests de Routage terminé
```

FIGURE 5 – Test routage

- Test d'accès au wiki

```
root@cassiopeaadmin:~/test/Disponibility/disponibility# ./disponibility_server_wiki_test.sh
===        Test de disponibilité du serveur wiki       ===
------------------------------------------------------------
Serveur wiki disponible.
Tests serveur wiki terminé.
```

FIGURE 6 – Test wiki

- Test de résolution DNS

```
root@cassiopeaadmin:~/test/Connectivity/connectivity_test# ./connectivity_dns_test.sh
=== TEST DNS MULTIPLE DOMAINS ===
DNS interrogé : 192.168.2.53
---------------------------------
Nom de domaine testé : web.cassiopea.fr

Résultat dig :
192.168.2.80

Résultat nslookup :
Server:         192.168.2.53
Address:        192.168.2.53#53

Name:   web.cassiopea.fr
Address: 192.168.2.80

---------------------------------
---------------------------------
Nom de domaine testé : google.com

Résultat dig :

Résultat nslookup :
Server:         192.168.2.53
Address:        192.168.2.53#53

** server can't find google.com: REFUSED
```

FIGURE 7 – Test DNS

- Test d'attribution d'adresse par DHCP

```
root@cassiopeaadmin:~/test/Connectivity/connectivity_test# ./connectivity_dhcp_test.sh
=== TEST DU SERVEUR DHCP ===
Killed old client process
Adresse IP relâchée.

Adresse IP actuelle :
    inet 127.0.0.1/8 scope host lo
    inet 192.168.3.20/24 brd 192.168.3.255 scope global dynamic ens18
```

FIGURE 8 – Test DHCP

- Test d'accès refusé à la base de données

```
root@cassiopeaadmin:~/test/Security/security# ./security_control_access_test.sh
===       Test de sécurité accès SGBD      ===
-------------------------------------------------
Test des règles SGBD :
Password for user arthur:
psql: error: connection to server at "localhost" (::1), port 5432 failed: FATAL:  password authentication failed for user "arthur"
connection to server at "localhost" (::1), port 5432 failed: FATAL:  password authentication failed for user "arthur"
Échec de la connexion SGBD, le SGBD est bien réglé.
Tests de sécurité terminés.
-------------------------------------------------
```

FIGURE 9 – Test SGBD

6 Bilan de la sécurité

6.1 Pare-feux

La sécurité de notre infrastructure passe avant tout par la configuration précise de pare-feux spécifiques à chaque réseau et machine. Nous avons deux niveaux de pare-feux, d'abord le routeur vérifie tous les échanges qui lui parviennent et autorise ou non en fonction de la provenance, de la destination et du service demandé. Ensuite, chaque serveur traite les requêtes qu'il reçoit et décide ou non de les accepter selon les mêmes critères. Bien que cela créé une répétition de certaines règles, c'est aussi un moyen de proposer une sécurité plus élevée. En cas d'intrusion sur le routeur par exemple, l'attaquant ne pourra pas se connecter sur les autres serveurs sans passer par le réseau administrateur.

Par exemple, le réseau administrateur a le droit de demander des connexions ssh vers n'importe où mais toute autre demande de connexion ssh est refusée que ce soit en entrée ou sortie.
Nous allons maintenant détailler les pare-feux mis en place en présentant les règles, bien sûr toutes les machines ont le droit d'utiliser le https et le dns pour réaliser les mises à jour de sécurité. Les serveurs ne peuvent initialiser aucune connexion.

- Règle par défaut indépendamment des serveurs : refuser tout en entrée et sortie.

règles par défaut

```
chain input {
        type filter hook input priority 0; policy drop;
        }
chain output {
        type filter hook output priority 0; policy drop;
        }
```

Il y aussi des règles communes aux serveurs du réseau 192.168.2.0/24 qui permet aux administrateurs d'accéder à ces machines :

règles par défaut

```
chain input {
    # Autoriser les connexions établies
    ct state established,related accept

    # Autoriser les pings administrateurs
    icmp type echo-request ip saddr 192.168.3.0/24 \
        accept

    # Autoriser les nouvelles connexions ssh
    tcp dport 22 ip saddr 192.168.3.0/24 \
        tcp flags == syn ct state new accept
}
chain output {
    # Autoriser les connexions établies
    ct state established,related accept

    # Autoriser les réponses icmp
    icmp type echo-reply ip daddr 192.168.3.0/24 accept
}
```

- Routeur interne : il a pour "mission" de gérer la chaîne forward et donc de vérifier la provenance et la destination des paquets qu'il reçoit. Les seules règles input et output concernent l'accès en ssh par les administrateur.

pare-feu routeur interne

```
chain input {
        # Autoriser la boucle locale et les connexions
        # établies/connexes
        iif "lo" accept
        ct state established,related accept

        # Autoriser le SSH sur l'interface interne (ens18)
        # en provenance du routeur principal
        tcp dport 22 iifname "ens21" accept
        icmp type echo-request iifname "ens21" \
            ip saddr 192.168.3.0/24 accept

        udp dport { 53, 67 } accept
  }
  chain forward {
    type filter hook forward priority filter; policy drop;
        ct state established,related accept
        icmp type echo-request iifname "ens21" \
            ip saddr 192.168.3.0/24 accept
        icmp type echo-reply iifname "ens20" \
            ip daddr 192.168.3.0/24 accept
        icmp type echo-reply iifname "ens19" \
            ip daddr 192.168.3.0/24 accept
        icmp type echo-reply iifname "ens18" \
            ip daddr 192.168.3.0/24 accept

        # Drop les autres requêtes ping
        icmp type echo-request drop

        ip saddr 192.168.3.0/24 tcp dport 22 accept
        ip daddr 192.168.3.0/24 tcp sport 22 accept

        # Autoriser DNS (UDP/TCP 53) pour la résolution de noms
        udp dport 53 accept
        udp sport 53 accept

        #Port wazuh uniquement pour les clients pour le moment

        # Autoriser HTTP/HTTPS (TCP 80/443) pour télécharger
        # les paquets
        tcp dport { 80, 443 } accept
  }
    chain output {
        oif "lo" accept
        ct state established,related accept
        tcp sport 22 oifname ens21 accept
        icmp type { echo-request, destination-unreachable, \
            time-exceeded } accept
        udp sport { 53, 67 } accept
  }
```

- Routeur DMZ : il doit gérer la chaîne forward en provenance de l'extérieur du réseau, c'est lui qui s'occupe des demandes d'accès au site public du département. Aucun paquet qu'il reçoit depuis l'extérieur ne doit être redistribué vers le réseau interne et les seuls accès autorisés sont des services web. Les seules règles input et output concernent l'accès en ssh par les administrateur.

pare-feu routeur dmz

```
    table inet nat {
  chain postrouting {
    type nat hook postrouting priority 100; policy accept;
    oifname "ens19" masquerade
  }
}
table inet filter{
    chain input {
        type filter hook input priority 0; policy drop;
        # Autoriser la boucle locale et les
        # connexions établies/connexes
        iif "lo" accept
        ct state established,related accept

        # Autoriser le SSH sur l'interface interne (ens18) en
        # provenance du routeur principal
        tcp dport 22 iifname "ens18" accept
        icmp type echo-request iifname "ens18" \
            ip saddr 192.168.3.0/24 accept
  }

  chain forward {
    type filter hook forward priority filter; policy drop;
        ct state established,related accept
        icmp type echo-request iifname "ens18" ip saddr \
            192.168.3.0/24 accept
        icmp type echo-reply iifname "ens20" ip daddr \
            192.168.3.0/24 accept
        # Drop les autres requêtes ping
        icmp type echo-request drop

        ip saddr 192.168.3.0/24 tcp dport 22 tcp flags == syn \
            ct state new accept

        tcp dport 22 drop

        # Autoriser DNS (UDP/TCP 53) pour la résolution de noms
        iifname { "ens18", "ens20", "ens21" } oifname "ens19" \
            meta l4proto { tcp, udp } th dport 53 ct state new accept

        # Autoriser HTTP/HTTPS (TCP 80/443) pour
        # télécharger les paquets
        iifname { "ens18", "ens20", "ens21" } oifname "ens19" \
            tcp dport { 80, 443 } ct state new accept

        # Optionnel : NTP (UDP 123) pour la synchronisation du temps
        iifname { "ens18", "ens20", "ens21" } oifname "ens19" \
            udp dport 123 ct state new accept

  }
```

pare-feu dhcp client

```
      chain output {
        type filter hook output priority filter; policy drop;
        oif "lo" accept
         ct state established,related accept
        tcp sport 22 accept
        icmp type { echo-request, destination-unreachable, \
            time-exceeded } accept
        oifname "ens19" meta l4proto { tcp, udp } th dport \
            { 53, 80, 443, 123 } ct state new accept
  }
}
```

- Le DHCP client : il s'occupe des requêtes DHCP en provenance du réseau 192.168.1.0/24. Il acepte également les connexions ssh des administrateurs.

pare-feu dhcp client

```
        chain input {
                # Autoriser DHCP
                udp dport 67 ip saddr 192.168.1.0/24 accept
        }
        chain output {
                # Autoriser les réponses dhcp
                udp sport 67 ip daddr 192.168.1.0/24 accept

                # Autoriser le téléchargement sur internet
                udp dport 53 accept
                tcp dport 80 accept
                tcp dport 443 accept
        }
```

- Le DHCP administrateur : il s'occupe des requêtes DHCP en provenance de son propre réseau 192.168.3.0/24. Il acepte également les connexions ssh des administrateurs. Aucune requête en provenance des autres réseaux ne doit être acceptée.

pare-feu dhcp administrateur

```
        chain input {
                # Autoriser DHCP
                udp dport 67 ip saddr 192.168.3.0/24 accept
        }
        chain output {
                # Autoriser les réponses dhcp
                udp sport 67 ip daddr 192.168.3.0/24 accept

                # Autoriser le téléchargement sur internet
                udp dport 53 accept
                tcp dport 80 accept
                tcp dport 443 accept
        }
```

- Le DNS interne : il se contente de répondre aux requêtes DNS des différents réseaux internes, il ne peut donc répondre qu'aux demandes en provenance de 192.168.1/.2/.3.0/24. Il accepte aussi les connexions ssh des adminstrateurs.

pare-feu dns interne

```
chain input {
        # DNS
        udp dport 53 ip saddr \
        { 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24 } accept
        }
chain output{
        # DNS
        udp sport 53 accept

        # Autoriser le téléchargement sur internet
        udp dport 53 accept
        tcp dport 80 accept
        tcp dport 443 accept
}
```

6.2 Failles logicielles

Afin d'éviter les failles logicielles qui peuvent apparaître à tout moment, nous avons sur chaque machine un script qui tourne tous les jours à 08h00 automatiquement grâce aux timers de systemd. Ce script est assez simple et se contente de mettre à jour les packages debian en intégralité. Ce script est le suivant :

Script màj sécurité

```
#!/bin/bash
# Ce script télécharge, installe et clean les packages debian
apt update && apt upgrade -y && apt clean && apt autoclean
```

Cette mise à jour automatique nous permet de quasiment garantir l'absence de failles logiciels dans les pakages Debian car nous partons du principe que nous pouvons faire confiance aux mises à jours Debian.

Le seul logiciel auquel nous ne pouvons pas garantir l'absence de failles et Mediawiki puisque nous l'avons installé manuellement. La version 1.39 que nous utilisons est mise à jour jusqu'à Novembre 2025, dans une situation réelle il nous faudrait donc prévoir une date à laquelle nous réaliserions la migration vers la version 1.45 qui est la plus récente, elle sera publiée en décembre 2025 et sera maintenue jusqu'à décembre 2026.
Pour réaliser les mises à jour en cas de failles nous devons : enregistrer la configuration actuelle de LocalSettings.php qui est le fichier le plus important de l'application ainsi que la base de donnée par précaution. Ensuite, il suffit de réaliser un git pull de la nouvelle version de la branche où nous sommes ou de réextraire les archives téléchargées. Si besoin, nous réstituons les fichiers sauvegardés. Cette mise à jour doit être réalisée manuellement il faut donc prévoir des crénaux où l'accès au wiki n'est pas important. De plus, afin de veiller à ces failles nous avons inscrit une adresse e-mail à mediawiki-announce@lists.wikimedia.org qui est la liste d'annonce des mises à jour et de la sécurité.

6.3 Bonnes pratiques

De manière globale nous avons respecté des pratiques que nous jugeons de "bonnes habitudes" concernant la sécurité, que ce soit la surveillance des algorithmes de chiffrement utilisés ou l'application des moindres privilèges sur les pare-feux. Nous allons revenir sur les différentes pratiques et mesures que nous avons mis en place.

— Sécurité des mots de passe : tous les mots de passes utilisés ont été générés automatiquement par Bitwarden (pour rappel, ici Bitwarden utilise Argon2id avec 1000 itérations et 64Mo de mémoire)

— Accès aux serveurs par clés ssh seulement : nous avons généré une paire de clé pour l'administrateur principal qui a ensuite transféré sa clé publique sur tous les serveurs nécessitant d'être administrés. Toutes les clés que nous utilisons sont générées avec l'algorithme rsa et 4096 bits.
— Sécurité des communications : l'échange des mots de passe a été fait par un envoi de mail signé et chiffré sur Thunderbird
— Application des moindres privilèges : tous les pare-feux ont une politique par défaut de refus que nous adaptons ensuite pour autoriser seulement les services nécessaires sur chaque serveur. Par exemple, le wiki n'autorise des demandes de connexions que sur le port 80, 443 et 22 depuis le réseau 192.168.3.0/24.
— Limitation des accès : les connexions ssh ne sont autorisées que depuis le réseau des administrateurs afin d'ajouter de la sûretée par dessus l'accès par clés ssh.

7 Conclusion

7.1 Retour d'expérience

Durant cette SAE, le groupe a le sentiment d'avoir appris beaucoup de choses d'un point de vue technique, nous avons pu tous prendre en main des nouveaux logiciels et services que nous n'avions jamais eu à installer avant. Notamment l'authentification par un serveur LDAP utilisée dans un serveur Web qui était très intéressante pour les personnes ayant travaillé dessus. De plus, nous avons pu apprendre à lire des documentations assez complexes que nous devions adapter à nos besoins. La mise en place des pare-feux était également un point assez complexe et le schéma que nous avions défini nous a beaucoup aidé à identifier quels réseaux pouvait utiliser quel service.
L'équipe a aussi aimé avoir un projet complètement concentré sur le réseau sans avoir besoin de réaliser du développement, cela nous a donné l'impression de vraiment nous spécialiser.

Nous trouvons cela assez dommage que la prise en main de Proxmox ait pris autant de temps, ce temps que nous aurions pu utiliser pour aller plus loin dans la complexité de notre réseau par exemple avec Kerberos, authentification CAS sur le wiki, meilleure définition des pare-feux, ajout du TLS sur les requêtes DNS. Au final nous avons perdu les deux premières semaines à configurer un réseau qui, dans la théorie, est très simple mais en pratique est difficile à mettre en place sur Proxmox.

Bien que notre temps ait été limité nous avons pu finir l'infrastructure que nous souhaitions dans les temps que nous avions prévus. Vendredi 4 avril nous n'avions plus d'installation à réaliser et nous nous sommes contentés de vérifier nos pare-feux et le fonctionnement des services une dernière fois. Nous avons donc pu finaliser les

rendus dans l'après-midi de samedi et en avons conclu que notre planning était bien défini.

7.2 Pistes d'amélioration

Bien que nous soyons satisfaits de l'état final de notre infrastructure et des services qu'elle propose, nous pourrions l'améliorer à la fois d'un point de vue technique sur les logiciels mais aussi sur l'infrastructure elle-même. Nous allons revenir point par point sur le type d'améliorations que nous aurions pu apporter.

— Renforcement des pare-feux : même si nous considérons que nos pare-feux répondent aux problématiques de disponibilité de notre infrastructure, nous n'avons pas eu le temps de considérer les attaques pouvant avoir lieu. Par exemple, une attaque *ping flood* qui ciblerait notre DMZ serait assez facile à réaliser. Pour éviter cela nous aurions pu limiter le nombre de paquets et/ou créer des *blacklist* d'adresses IP.

— Authentification plus poussée : actuellement l'authentification sur le wiki des administrateurs passe par un annuaire LDAP et nous avons un serveur CAS qui fonctionne avec ce serveur LDAP. Nous aurions voulu mettre en place l'authentification par le serveur CAS directement mais faute de temps nous n'avons pas pu. De plus, Mediawiki propose des extensions pour réaliser cette authentification alors que nous utilisons actuellement celles de Apache. Il aurait été intéressant de passer par les outils fournis par Mediawiki directement.

— Ajout de services : nous avons pu remplir les attendus du niveau 1 et 2 et quelques détails du niveau 3 mais nous pourrions encore ajouter des services qui contriburaient à la sécurité. Par exemple, l'utilisation de DNS over TLS, l'ajout de serveurs DHCP "de secours" qui prennent le relais sur nos serveurs DHCP actuels en cas de panne pour que les clients puissent continuer à fonctionner, ou encore la sécurisation du NFS avec Kerberos.

— Outils de surveillance : nous n'avons pas d'outils de supervision du réseau que ce soit les performances ou bien qui surveille les éventuelles attaques. Nous pourrions donc ajouter un outil de supervision ou un SIEM. Bien que nous ayons essayé de mettre en place des clients avec Wazuh, nous n'avons pas eu le temps de finaliser son fonctionnement.

— Automatisation plus poussée : nous aurions pu réaliser des scripts généraux qui, depuis un hyperviseur, réaliserait l'installation complète des différentes machines en éxecutant pour chacune des scripts spécifiques aux logiciels souhaités. Cela simplifierait la migration de l'infrastructure et pourrait servir de plan de reprise en cas de destruction d'un hyperviseur.

— Crash test : nous aurions voulu réaliser des tests de résistance sur nos serveurs, par exemple voir le nombre de requêtes qu'un serveur web pouvait supporter et définir les mesures à prendre pour éviter une panne.