R4-Cyber-11 – Sécurisation de services réseaux

HAProxy

Étudiant : GRONDIN Benjamin
Formation : BUT 2 TP 1 Réseaux & Télécommunications
Professeur : REPUSSEAU Willy

Contents

1 Introduction **2**

2 Plan d'adressage **2**

3 Installation d'Apache **2**
3.1 Choix du serveur . . . 2
3.2 Installation d'Apache . . . 2
3.3 Vérification du port d'écoute . . . 3

4 Modification de la page d'accueil du site **3**
4.1 Quel(s) fichier(s) ce dossier contient-il ? . . . 3

5 Sécurisation d'un site en HTTPS **4**
5.1 Génération d'un certificat auto-signé . . . 4
5.2 Création d'un site sécurisé et activation de HTTPS . . . 4
5.3 Configuration du site sécurisé . . . 5
5.4 Test . . . 5

6 Mise en place de la haute disponibilité HTTP **5**
6.1 Installation d'un second serveur Apache . . . 5
6.2 Installation de HAProxy . . . 6
6.3 Configuration de HAProxy . . . 6
6.3.1 Configuration du frontend . . . 6
6.3.2 Configuration du backend . . . 6
6.3.3 Configuration réseau des serveurs . . . 7
6.4 Vérification de l'accès aux sites . . . 7
6.5 Persistance basée sur des cookies . . . 8
6.6 Visualisation des statistiques . . . 8

7 Mise en place de la haute disponibilité HTTPS **8**
7.1 Configuration et vérification . . . 8
7.1.1 Génération du certificat pour HAProxy . . . 8
7.1.2 Création du fichier PEM . . . 9

7.1.3 Configuration du frontend HTTPS 9
7.2 Configuration supplémentaire 9
7.2.1 Redirection HTTP vers HTTPS 9

8 Conclusion **9**

1 Introduction

Objectifs du TP

Mettre en place un équilibrage de charge avec HAProxy pour distribuer les requêtes entre plusieurs serveurs web Apache. L'infrastructure inclut la configuration de HTTPS, la redirection automatique de HTTP vers HTTPS et l'activation des statistiques HAProxy. Une analyse de la communication TLS est également effectuée.

L'objectif de ce TP est de mettre en place un équilibrage de charge avec HAProxy pour distribuer les requêtes entre plusieurs serveurs web Apache. L'infrastructure inclut la configuration de HTTPS, la redirection automatique de HTTP vers HTTPS et l'activation des statistiques HAProxy. Une analyse de la communication TLS est également effectuée.

2 Plan d'adressage

Machines	Adresses IP
Serveur web 1	`192.168.10.1/24`
Serveur web 2	`192.168.10.2/24`
2*Load balancer HAProxy	`192.168.1.9/24`
	`192.168.10.10/24`

Table 1: Plan d'adressage de l'infrastructure

3 Installation d'Apache

3.1 Choix du serveur

Pour débuter ce TP sur HAProxy, il faut d'abord une distribution Linux pour héberger les sites web. J'ai choisi Ubuntu, car j'ai déjà installé et configuré des serveurs de répartition de charge (load balancing) sur ce système par le passé.

3.2 Installation d'Apache

Définition

Apache HTTP Server, ou simplement Apache, est l'un des serveurs web les plus connus et les plus utilisés au monde. Gratuit et open-source, il est à la fois puissant et très flexible. Son rôle principal est de gérer et diffuser des sites web, des applications et du contenu en ligne via le protocole HTTP. Il s'adapte à différents systèmes d'exploitation comme Linux, Windows et macOS.

Pour installer un serveur Apache sur une distribution Ubuntu, il faut taper les commandes suivantes :

Listing 1: Installation d'Apache2

```
sudo apt update
sudo apt upgrade
sudo apt install apache2
```

Avant d'installer Apache, il est important de s'assurer que le système d'exploitation est à jour. La commande `apt update` permet au système de récupérer la liste des dernières versions des paquets disponibles. Ensuite, la commande `apt upgrade` met à jour les paquets existants en installant les versions les plus récentes.

3.3 Vérification du port d'écoute

Le port d'écoute est un point d'entrée pour les requêtes. Lorsqu'un utilisateur souhaite accéder à une page web, l'adresse IP sera associée à un port afin de permettre l'échange des requêtes. Dans le cadre d'un serveur Apache, il est important de vérifier que le port 80 est bien actif en utilisant la commande `netstat -ntl`.

Listing 2: Vérification du port d'écoute

```
tcp    0    0 192.168.1.9:80    0.0.0.0:*    LISTEN
```

Remarque

Cette capture montre que le serveur écoute bien sur le port 80 après TCP (protocole utilisé par Apache). L'adresse IP de la machine suivie de `:80` indique le port d'écoute, et la ligne se termine par `LISTEN`, confirmant que la machine est en écoute et peut recevoir des requêtes HTTP.

4 Modification de la page d'accueil du site

Nous allons maintenant modifier la page web de notre serveur Apache afin de la rendre identifiable lorsque le load balancing sera opérationnel. Pour ce faire, nous accédons au fichier *000-default.conf*, situé dans */etc/apache2/sites-available/*.

4.1 Quel(s) fichier(s) ce dossier contient-il ?

Le fichier *000-default.conf* définit la configuration par défaut d'Apache. La directive `DocumentRoot` indique le dossier où sont stockés les fichiers du site web, généralement */var/www/html* sur Ubuntu et Debian. Ce dossier contient par défaut un fichier *index.html*, qui sert de page de test après l'installation d'Apache.

Remarque

Pour personnaliser la page d'accueil, il suffit de modifier le fichier */var/www/html/index.html* et d'y insérer le contenu souhaité, par exemple : "Vous êtes sur le serveur web 1 de GRONDIN Benjamin".

5 Sécurisation d'un site en HTTPS

Pour garantir que les données échangées entre le client et le serveur ne puissent être lues ou altérées par des tiers, il est essentiel d'utiliser HTTPS afin de chiffrer les informations. Bien que l'ANSSI ne rende pas HTTPS obligatoire pour tous les sites, son utilisation est fortement recommandée pour protéger les données sensibles et se conformer aux bonnes pratiques de sécurité. Certaines réglementations, comme le RGPD, exigent des mesures de protection adaptées, et HTTPS est une solution standard pour sécuriser les échanges de données.

Dans ce TP, nous aborderons ce principe de chiffrement afin d'acquérir les bonnes pratiques. Les sites nécessitant un chiffrement des échanges devront obtenir un certificat reconnu délivré par un organisme de confiance. Ici, nous opterons pour un certificat auto-signé.

5.1 Génération d'un certificat auto-signé

Pour générer un certificat auto-signé, il faut taper la commande suivante sur le serveur web :

Listing 3: Génération d'un certificat auto-signé

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -out /etc/apache2/server.pem \
    -keyout /etc/apache2/server.key
```

Remarque

Le fichier *server.pem* contient le certificat du serveur, tandis que *server.key* stocke la clé privée associée. Ensemble, ils permettent d'établir une connexion sécurisée via HTTPS en assurant le chiffrement et l'authentification des échanges.

5.2 Création d'un site sécurisé et activation de HTTPS

Nous allons créer un nouveau dossier *html_ssl*, qui contiendra le fichier *index.html*, accessible par les clients via des requêtes HTTPS.

Listing 4: Création du dossier et activation HTTPS

```
mkdir html_ssl
cd html_ssl
nano index.html
```

Maintenant, nous pouvons activer les connexions HTTPS :

Listing 5: Activation du module SSL

```
sudo a2enmod ssl
sudo a2ensite default-ssl
sudo systemctl restart apache2
```

La commande `a2enmod ssl` active le module SSL d'Apache pour permettre les connexions sécurisées. La commande `a2ensite default-ssl` active la configuration du site sécurisé. Un redémarrage d'Apache est nécessaire pour appliquer les modifications.

5.3 Configuration du site sécurisé

Nous devons maintenant modifier le fichier *default-ssl.conf* pour que la page affichée soit celle en HTTPS.

Listing 6: Configuration du VirtualHost HTTPS

```
<VirtualHost *:443>
    DocumentRoot /var/www/html_ssl
    ServerName www.site-de-benjamin.com
    SSLEngine on
    SSLCertificateFile /etc/apache2/server.pem
    SSLCertificateKeyFile /etc/apache2/server.key
</VirtualHost>
```

Remarque

Pourquoi n'est-il pas nécessaire de modifier le fichier *ports.conf* ?
Il n'est pas nécessaire de modifier ce fichier car il est déjà configuré pour permettre l'écoute des connexions HTTPS sur le port 443, qui est le port standard utilisé pour les communications sécurisées avec SSL/TLS.

5.4 Test

Puisque le site écoute désormais sur le port 443 pour HTTPS, on peut vérifier que celui-ci est bien actif comme dans la partie 1.3.

Remarque

Le message d'erreur affiché par le navigateur indique que le certificat SSL utilisé n'est pas signé par une autorité de certification reconnue (`NET::ERR_CERT_AUTHORITY_INVALID`), ce qui est courant pour un certificat auto-signé. Pour accéder au site, il faut accepter une exception de sécurité dans le navigateur. Pour rendre le site digne de confiance, il faudrait utiliser un certificat signé par une autorité de certification reconnue comme Let's Encrypt ou DigiCert.

6 Mise en place de la haute disponibilité HTTP

Pour assurer une haute disponibilité (HA), il faut au moins deux serveurs qui hébergent les mêmes services. J'ai cloné la première machine et modifié la page web afin que l'utilisateur sache qu'il est sur le serveur web 2. Grâce à HAProxy, on peut répartir le trafic entre ces serveurs, garantissant un service toujours accessible même en cas de panne d'un serveur, tout en améliorant les performances et l'expérience utilisateur.

6.1 Installation d'un second serveur Apache

Pour obtenir un serveur avec les mêmes configurations que le précédent, il suffit de le cloner sous VirtualBox en suivant ces étapes :

1. Faire un clic droit sur la machine, puis cliquer sur **Cloner**

2. Choisir le nom de la machine ainsi que le dossier de stockage
3. Dans le menu déroulant, choisir **Générer de nouvelles adresses MAC pour toutes les interfaces réseau** pour éviter les conflits d'adressage

6.2 Installation de HAProxy

Pour installer HAProxy, j'ai utilisé une troisième machine Ubuntu qui servira de load balancer.

Listing 7: Installation de HAProxy

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install haproxy
```

6.3 Configuration de HAProxy

Pour que HAProxy puisse fonctionner, voici les paramètres à ajouter dans le fichier */etc/haproxy/haproxy.cfg* :

6.3.1 Configuration du frontend

Listing 8: Configuration du frontend HTTP

```
1 frontend http_front
2     bind 192.168.1.9:80
3     bind 192.168.10.10:80
4     default_backend web_servers
```

Ces lignes configurent le frontend nommé `http_front`. Elles indiquent à HAProxy d'écouter les connexions HTTP sur deux adresses IP différentes, chacune sur le port 80. Cela permet à HAProxy de recevoir les requêtes HTTP provenant de ces deux interfaces réseau et de les acheminer vers les serveurs backend configurés.

6.3.2 Configuration du backend

Listing 9: Configuration du backend

```
1 backend web_servers
2     balance roundrobin
3     cookie SERVERUSED insert indirect nocache
4     server web1 192.168.10.1:80 cookie serveur1 check inter 500 rise 2 fall 3
5     server web2 192.168.10.2:80 cookie serveur2 check inter 500 rise 2 fall 3
```

Dans cette configuration, deux serveurs web (web1 et web2) sont spécifiés avec leurs adresses IP respectives. Les paramètres définissent des options de suivi de session et de santé :

- **balance roundrobin** : Répartition équitable du trafic
- **cookie** : Persistance de session basée sur les cookies
- **check** : Vérification de la disponibilité des serveurs

- **inter 500** : Intervalle de vérification (500ms)
- **rise 2** : Nombre de vérifications réussies avant de considérer le serveur opérationnel
- **fall 3** : Nombre d'échecs avant de considérer le serveur hors service

6.3.3 Configuration réseau des serveurs

Pour que l'infrastructure fonctionne correctement, il faut configurer les interfaces réseau avec des adresses IP statiques.

Modification du fichier netplan sur les serveurs web Pour commencer, rendez-vous dans */etc/netplan/*. Modifiez le fichier de configuration *50-cloud-init.yaml* :

Listing 10: Configuration netplan serveur web 1

```
network:
    ethernets:
        enp0s3:
            addresses: [192.168.10.1/24]
            dhcp4: no
            optional: true
    version: 2
```

Remarque

Dans ce fichier, on spécifie l'adresse IP statique du serveur web. Il faut appliquer la même configuration pour le serveur web 2 en changeant l'adresse IP. Une fois les modifications effectuées, tapez `netplan try`, puis appuyez sur Entrée si la configuration est correcte.

Modification du fichier netplan sur le serveur HAProxy Pour que le serveur HAProxy fonctionne correctement, il faut deux cartes réseau. L'une sera accessible depuis l'extérieur pour les machines envoyant leurs requêtes, et la deuxième servira d'intermédiaire entre les serveurs web et le réseau Internet.

Pour ajouter une carte réseau dans VirtualBox : cliquer sur la machine cible, puis Configuration > Réseau. Ajoutez une interface réseau en cliquant sur "Activer l'interface réseau" et choisissez le paramètre "Réseau interne" dans le menu déroulant.

6.4 Vérification de l'accès aux sites

Dès qu'un utilisateur souhaite se connecter au site, il doit maintenant taper l'adresse IP du load balancer (`192.168.1.9`), qui se chargera de répartir les requêtes en fonction de la disponibilité des serveurs.

Remarque

Pour vérifier le fonctionnement du load balancer, on peut désactiver l'interface réseau sur le serveur 1 avec la commande `ip link set enp0s3 down`. Le trafic sera alors automatiquement redirigé vers le serveur 2.

Le load balancer fonctionne car roundrobin répartit le trafic de manière équitable entre les serveurs, envoyant chaque nouvelle requête à un serveur différent dans la liste, permettant à chaque serveur de recevoir une part égale du trafic au fil du temps.

6.5 Persistance basée sur des cookies

La persistance basée sur des cookies (aussi appelée stickiness ou affinité de session) permet de garantir qu'un utilisateur soit toujours dirigé vers le même serveur backend pendant toute la durée de sa session. Dans cette section, nous configurons HAProxy pour utiliser les cookies comme mécanisme de persistance.

La configuration des cookies a déjà été ajoutée dans la section backend précédente :

Listing 11: Configuration des cookies pour la persistance

```
cookie SERVERUSED insert indirect nocache
server web1 192.168.10.1:80 cookie serveur1 check inter 500 rise 2 fall 3
server web2 192.168.10.2:80 cookie serveur2 check inter 500 rise 2 fall 3
```

Le paramètre `cookie SERVERUSED` permet à HAProxy de définir un cookie qui assure la persistance de la session entre le client et le serveur backend. Le cookie est ajouté dans les réponses des serveurs, avec les options `insert`, `indirect` et `nocache` pour garantir qu'il soit bien envoyé et non mis en cache.

6.6 Visualisation des statistiques

Pour visualiser les statistiques d'utilisation du load balancer, on peut activer la page stats dans les configurations en ajoutant les lignes suivantes :

Listing 12: Configuration de la page de statistiques

```
frontend stats
    bind 192.168.1.9:8080
    stats uri /stats
    stats auth admin:password
```

Ces lignes configurent un frontend stats qui écoute sur l'adresse `192.168.1.9:8080`, permet d'accéder aux statistiques via l'URL `/stats`, et protège l'accès avec une authentification `admin:password`.

7 Mise en place de la haute disponibilité HTTPS

Afin de pouvoir échanger des informations en HTTPS entre le serveur HAProxy et les clients, nous devons mettre en place une sécurisation HTTPS via un certificat auto-signé.

7.1 Configuration et vérification

7.1.1 Génération du certificat pour HAProxy

Listing 13: Génération du certificat auto-signé pour HAProxy

```
sudo openssl req -x509 -newkey rsa:2048 \
    -keyout /etc/ssl/certs/haproxy.key \
    -out /etc/ssl/certs/haproxy.crt \
```

```
    -days 365 -nodes
```

Cette commande génère un certificat auto-signé RSA de 2048 bits, avec une clé privée stockée dans */etc/ssl/certs/haproxy.key* et un certificat public dans */etc/ssl/certs/haproxy.crt* valable pendant 365 jours. L'option `-nodes` empêche la création d'un mot de passe pour la clé privée.

7.1.2 Création du fichier PEM

Ensuite, il faut copier le certificat et la clé privée dans le même fichier :

Listing 14: Création du fichier PEM

```
sudo cat /etc/ssl/certs/server.crt /etc/ssl/private/server.key > \
    /etc/ssl/certs/haproxy.pem
```

7.1.3 Configuration du frontend HTTPS

Pour finir, ajoutez la ligne suivante dans *haproxy.cfg* pour que les utilisateurs puissent se connecter via HTTPS :

Listing 15: Configuration du bind HTTPS

```
1 bind 192.168.1.9:443 ssl crt /etc/ssl/certs/haproxy.pem
```

Cette ligne configure HAProxy pour écouter sur l'adresse `192.168.1.9` et le port 443 en HTTPS, en utilisant le certificat *haproxy.pem* pour sécuriser les connexions.

7.2 Configuration supplémentaire

7.2.1 Redirection HTTP vers HTTPS

Afin de garantir une sécurité supplémentaire pour l'utilisateur, on peut rediriger toutes les requêtes HTTP vers HTTPS. Pour ce faire, il faut ajouter cette ligne dans *haproxy.cfg* :

Listing 16: Redirection automatique HTTP vers HTTPS

```
1 http-request redirect scheme https if !{ ssl_fc }
```

Cette ligne redirige toutes les requêtes HTTP vers HTTPS si la connexion n'est pas déjà sécurisée en SSL/TLS.

8 Conclusion

Ce TP nous a permis de mettre en place un équilibrage de charge avec HAProxy et d'expérimenter différentes techniques pour assurer la disponibilité, la répartition du trafic et la sécurisation des connexions web.

Compétences acquises :

- Optimisation de la disponibilité d'un site en répartissant la charge entre plusieurs serveurs

- Redirection automatique du trafic HTTP vers HTTPS pour garantir une connexion sécurisée
- Gestion des certificats SSL en imposant l'utilisation de TLS 1.2 au minimum
- Supervision de HAProxy à l'aide de son interface de statistiques
- Analyse des échanges réseau avec Wireshark, notamment les différentes étapes du handshake TLS

Ce projet nous a permis de comprendre l'importance de HAProxy dans les architectures modernes et de manipuler des concepts essentiels en réseau et en cybersécurité. Les compétences acquises au cours de ce TP seront précieuses pour la mise en place et la sécurisation de services web en production.