

Situation n°2 : Sécuriser l'accès à un service web



MISSION A : Mise en place d'une résolution de nom pour les tests

Commençons par installer les paquets nécessaires au fonctionnement du futur service DNS tournant sous Linux : bind9.

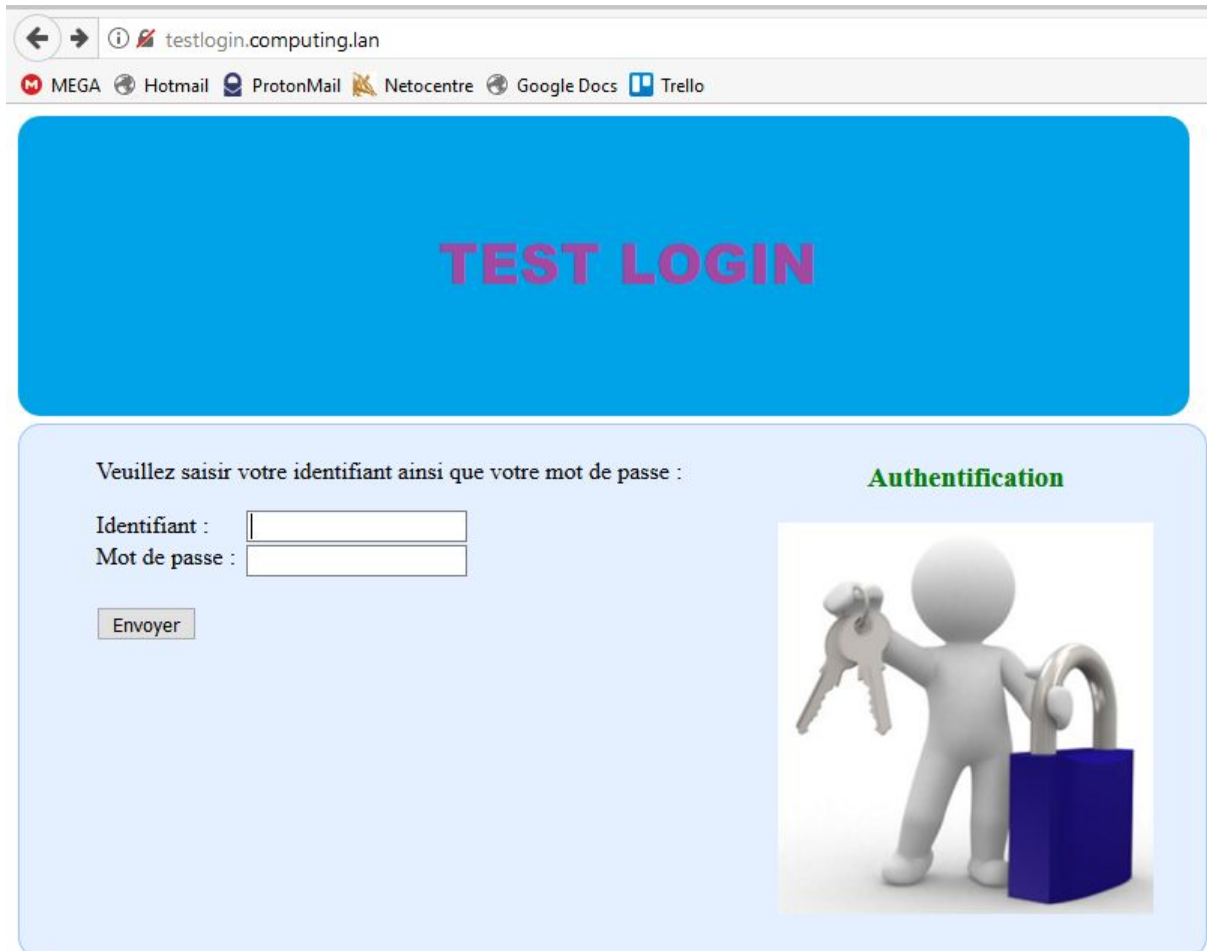
Nous configurons la zone "computing.lan", ainsi que les fichiers de zone direct, de zone inverse de la façon suivante :

```
;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      DOCKER.computing.lan. root.computing.lan. (
                                2          ; Serial
                                604800     ; Refresh
                                86400      ; Retry
                                2419200    ; Expire
                                604800 )   ; Negative Cache TTL
;
@         IN      NS       DOCKER.computing.lan.
docker    IN      A        172.16.55.2
@         IN      A        127.0.0.1

testlogin IN CNAME docker
```

```
;
; BIND reverse data file for local loopback interface
;
$TTL      604800
@         IN      SOA      DOCKER.computing.lan. root.computing.lan. (
                                1          ; Serial
                                604800     ; Refresh
                                86400      ; Retry
                                2419200    ; Expire
                                604800 )   ; Negative Cache TTL
;
@         IN      NS       DOCKER.computing.lan.
2         IN      PTR      DOCKER.computing.lan.
```

De cette façon, un “*ping docker.computing.lan*” ou un “*ping testlogin.computing.lan*”. Nous pouvons maintenant naviguer jusqu'à l'URI “testlogin.computing.lan” pour accéder à notre application :



testlogin.computing.lan

MEGA Hotmail ProtonMail Netocentre Google Docs Trello

TEST LOGIN


Veuillez saisir votre identifiant ainsi que votre mot de passe :

Identifiant :

Mot de passe :

Envoyer

Authentification



A2. Proposer un réaliser un test permettant de vérifier que le service DNS est opérationnel et que la résolution de nom pour notre application web est activé.

Il suffit d'essayer d'accéder à l'application via le FQDN “testlogin.computing.lan”, et de vérifier que les interactions avec les applications fonctionnent correctement.

MISSION B : Comprendre un service modulaire

B.1 Vérifier les ports d'écoute des conteneurs du service web et du service de base de données.

Pour ce faire, nous entrons dans les conteneurs concernés (“*docker exec -it <numero> bash*”), nous installons le paquet “*net-tools*” au sein des conteneurs, et nous exécutons la commande permettant de lister les ports d'écoutes : “*netstat -paunt -a*”.

Le résultat de la commande sur le container apache est :

Sur le container base de donnée, le résultat sera :

Nous confirmons donc que la redirection de port docker est conforme aux ports d'écoutes des containers.

B.2 Lister les fichiers principaux liés au service web "apache2". Préciser le fichier de configuration qui va être lu en priorité lors du lancement du service.

Ces fichiers sont présents sous le répertoire `/etc/apache2`, soit "*apache2.conf*, *conf-enabled*, *magic*, *mods-enabled*, *sites-available*, *conf-available*, *envvars*, *mods-available*, *ports.conf*, *sites-enabled*".

Le fichier qui sera lu en priorité est "*apache2.conf*". En effet, celui-ci est le fichier principal de configuration. C'est à partir de lui que tous les autres fichiers de configuration sont chargés.

B.3 indiquer la hiérarchie qui existe entre les principaux fichiers cités précédemment. En quoi cette information pourrait-être utile en cas de problème ou d'incident avec le service web ?

La hiérarchie entre les fichiers de configuration est illustrée dans le fichier principal "*apache2.conf*" :

En cas d'incident, il peut être intéressant de localiser la source du problème grâce au rôle de chacun de ces fichiers de configurations. En connaissant l'arborescence des fichiers de configurations apache, il est alors possible de déterminer quel fichier est mal chargé par le service.

B.4 Retrouver et traduire les valeurs indiqués dans votre fichier de configuration “*apache2.conf*” pour les directives suivantes :

- **Timeout**
- **KeepAlive**
- **MaxKeepAliveRequests**
- **KeepAliveTimeout**

Dans le fichier “*apache2.conf*” du container, les valeurs sont les suivantes :

- Timeout 300 : la durée en secondes pendant laquelle le service apache va attendre la réception ou l’émission d’informations en cours de communications.
- KeepAlive On : indique que le serveur autorise plusieurs demandes par connexions.
- MaxKeepAliveRequests 100 : Nombre maximum de connexion persistantes à autoriser sur le serveur apache2.
- MaxKeepAliveTimeout 5 : Nombre de secondes à attendre entre plusieurs requêtes du même client. Si le délais est dépassé, la connexion sera terminée par le serveur.

B.5 Indiquer la directive qui définit le port d’écoute du serveur.

La directive “Listen” permet de définir les ports d’écoutes du serveur apache :

B.6 remplacer le port d’écoute du protocole HTTP par le port 880. Préciser les modifications à opérer dans votre environnement docker et le test à réaliser pour valider votre changement.

Pour ce faire, nous allons éditer le fichier */etc/apache2/ports.conf* en modifiant le port d’écoute :

Ensuite, il ne restera qu’à stopper le container apache, et le relancer en adaptant la commande de redirection de ports :

“docker run -d -p880:880 --name apache-bis apache”.

Pour vérifier que tout fonctionne bien avec les nouveaux ports, essayons d’accéder à l’URL de notre serveur docker via le port 880 :

B.7 Comparer la liste des modules dans le dossier “*mods-available*” avec celle du module “*mods-enable*”. Citer un module disponible mais non-chargé.

En comparant la liste des modules, nous observons les différences de modules :

Par exemple, le module “*dump_io.load*” est disponible mais non chargé sur le serveur.

B.8 Quelle est l'utilité des modules pour le service web “apache2” ? Pourquoi peut-on parler d'un service modulaire.

Les modules permettent d'ajouter de nouvelles fonctionnalités au serveur web. Le service web est donc modulaire puisqu'il accepte différents modules susceptibles de modifier les fonctionnalités du serveur, ou d'en ajouter de nouveaux.

MISSION C : Sécuriser l'accès au service web

C.1 Installer et paramétrer sur le serveur DOCKER un accès distant via les protocoles TELNET et SSH.

le service ssh ayant déjà été installé à la création de la machine, il ne reste plus qu'à installer un client TELNET. Un "*apt-cache search telnet*" nous indique que le paquet responsable du serveur TELNET s'appelle "telnetd". il ne reste qu'à l'installer avec notre gestionnaire de paquet apt. Nous pouvons désormais nous connecter en ssh et via telnet depuis l'utilitaire putty.

C.2 Utilisez un logiciel d'écoute de trames afin de visualiser le contenu des messages lors d'une connexion en TELNET puis en SSH. Commenter sur le protocole à conserver pour une connexion distante sécurisée et désactiver l'autre protocole.

La connexion TELNET est complètement visible en clair via un logiciel de capture de trames:

Alors que la connexion distante via SSH est complètement illisible :

En définitif, il est plus sécurisé d'utiliser SSH pour les connexions à distances. Pour terminer, nous allons supprimer et purger les paquets telnet afin de ne plus pouvoir héberger de connexions telnets distantes.

C.3 Configurer votre site pour n'accepter que des requêtes d'une adresse IP d'un poste client de votre réseau, toutes les requêtes d'autres postes doivent-êtré rejetées.

Pour effectuer un filtrage par adresses IP, il suffit de modifier le fichier

"*/etc/apache2/apache2.conf*", et d'ajouter dans la balise <Directory> les lignes suivantes :

```
< Directory "/var/www/html/" >  
    Options Indexes FollowSymLink  
    AllowOverride none  
    Require IP *mon ip*  
    Require all denied  
< /Directory >
```

De cette façon, seul la machine dont l'ip est indiquée dans le fichier est autorisée à voir le contenu des pages sur le serveur web. les autres hôtes qui vont essayer de se connecter tomberont sur la page suivante :

C.4 Configurer un espace protégé “administration” sur votre site accessible avec le login “docker” et le mot de passe “admin” afin d’y héberger les scripts les plus sensibles.

Pour configurer une espace d'administration protégé sur le serveur, nous allons commencer par créer un dossier “Administration” sous /var/www/html. Nous ajouterons un fichier de test “index.html” dedans, comportant un simple message de bienvenu sur l'espace d'administration”.

Pour protéger le dossier, nous allons commencer par créer le fichier caché .htpasswd contenant le hash du mot de passe d'accès à l'espace web d'administration.

La commande htpasswd -c /etc/apache2/passwords guillaume.

“guillaume” sera le nom d'utilisateur pour se connecter. le mot de passe nous sera demandé, avant d'être hashé dans le fichier passwords sous le répertoire apache2. il faudra ensuite éditer le fichier “apache2.conf” pour y ajouter les instructions :

Pour terminer et appliquer les changement, nous rechargeons la configuration apache.

Lorsque nous allons essayer d'accéder à notre espace d'administration, nous tombons alors sur le formulaire de connexion :