Comment accéder à son serveur Domoticz depuis le web sans l'exposer directement depuis sa box?

Cet article propose une solution (parmi d'autres) pour rendre possible l'accès à votre serveur Domoticz depuis le Web sans l'exposer directement depuis votre Box ?

1 Le NAT c'est pratique mais....

Dans les différents forums sur Domoticz, la solution la plus souvent proposée pour accéder à son serveur Domoticz depuis le web consiste à faire du Nat depuis sa box internet.

1.1 Exemple de NAT

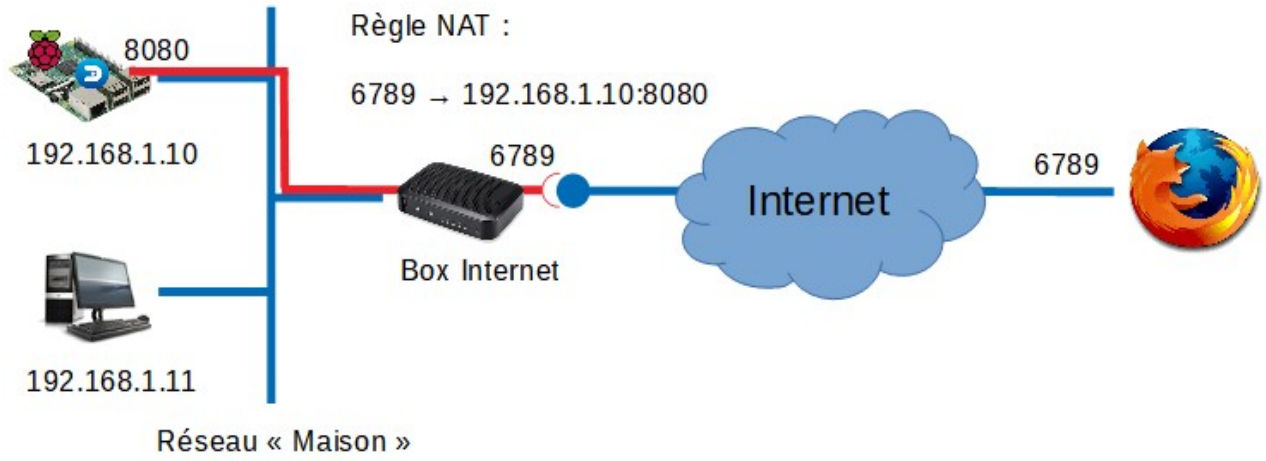

Illustration 1: Exemple de Nat 6789 -> 192.168.0.10:8080

Généralement, on conseil de faire du « NAT » depuis sa box internet. En quoi cela consiste ?

Simplement à ouvrir un port sur la Box Internet accessible depuis le Web et d'envoyer l'ensemble des flux sur une adresse ip:port précise. Par exemple, sur la figure 1, on a fait une règle Nat sur la Box Internet du port 6789 vers le raspi en 192.168.0.10. Les flux TCP se connectant sur ce port sont ainsi redirigés sur le serveur Web de Domoticz en 8080. On peut ainsi se connecter depuis un Browser sur votre Domoticz.

C'est quand même bien pratique !

1.2 Oui mais ce n'est pas très secure...

Bon, maintenant imaginons le scénario suivant... une personne malveillante trouve que votre port 6789 est ouvert et qu'il peut contacter le serveur web Domoticz. Même s'il n'a pas le mot de passe, il peut déjà savoir que c'est un Domoticz et connaître sa version.

S'il existe une CVE (Common Vulnerabilties and Exposures), ou en clair, une vulnérabilité connue, alors avec un tout petit peu de talent et beaucoup d'internet, il devrait pouvoir exploiter cette vulnérabilité pour... ben je sais pas, prendre le contrôle des périphériques domotiques de votre maison et allumer le

chauffage, ouvrir les volets, regarder les caméras, prendre des photos, etc. Mais comme il arrive directement sur votre réseau "maison", alors il peut également choisir de l'explorer et arriver sur votre NAS ou votre ordinateur personnel...

1.3 Je risque rien, il le trouvera jamais mon port "6789" !

On pourrait croire que ce scénario est improbable et que franchement, perdu dans ma Bretagne profonde, personne n'ira récupérer l'adresse ip de ma box et ne pourra pas trouver mon port secret ?

Je le pensais avant... mais c'était avant. En fait, si vous regardez vos logs de box (si vous y avez accès), vous allez vous apercevoir que chaque jour des scans de ports sont effectués. De toute façon, votre box à fort probablement été référencée par le moteur de recherche shodan (https://www.shodan.io). Vous pouvez même y faire une recherche sur "Domoticz" et trouver directement les ports ouvert comme le 8081 en Figure 2.

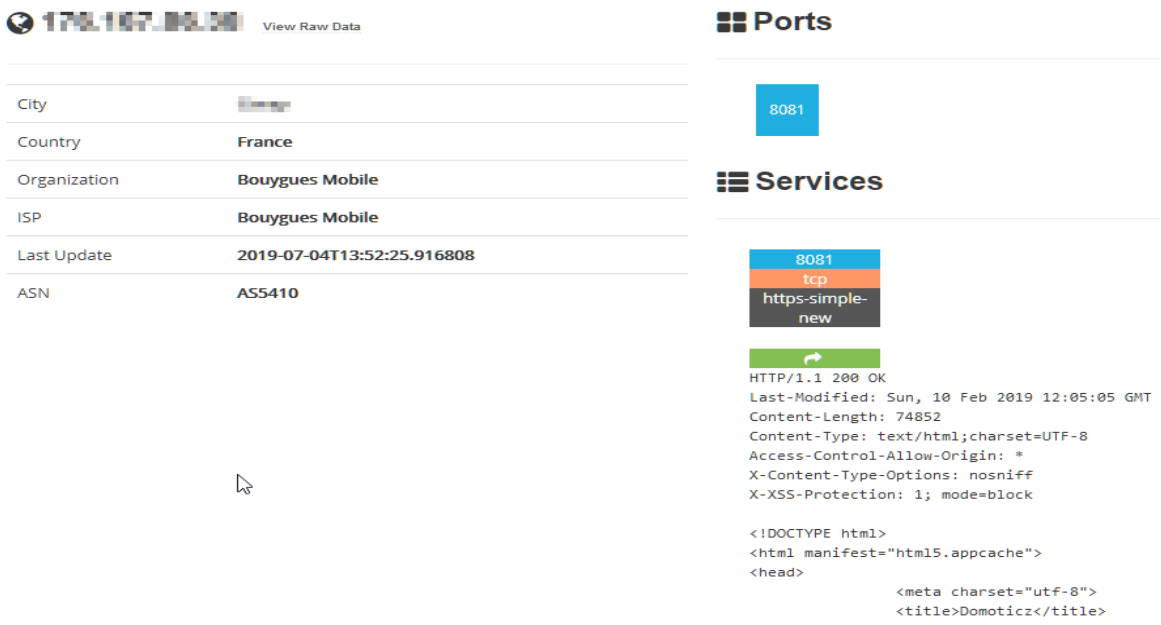

Illustration 2: Exemple de scan Shodan avec le mot clé "Domoticz"

Ici, Shodan ne propose pas de CVE, mais ce n'est pas le cas pour tout le monde... Voici une autre réponse concernant une personne qui aime ouvrir beaucoup de ports sur son accès internet (cf. Figure 3) et aussi toute une ribambelle de CVE directement exploitables...

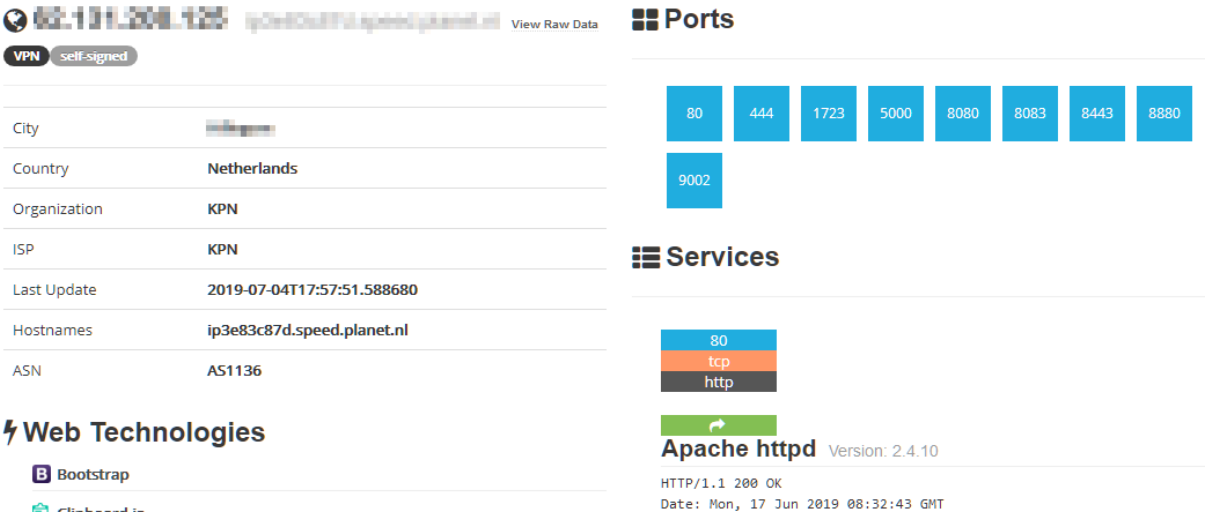

Illustration 3: Une recherche particulièrement intéressante...

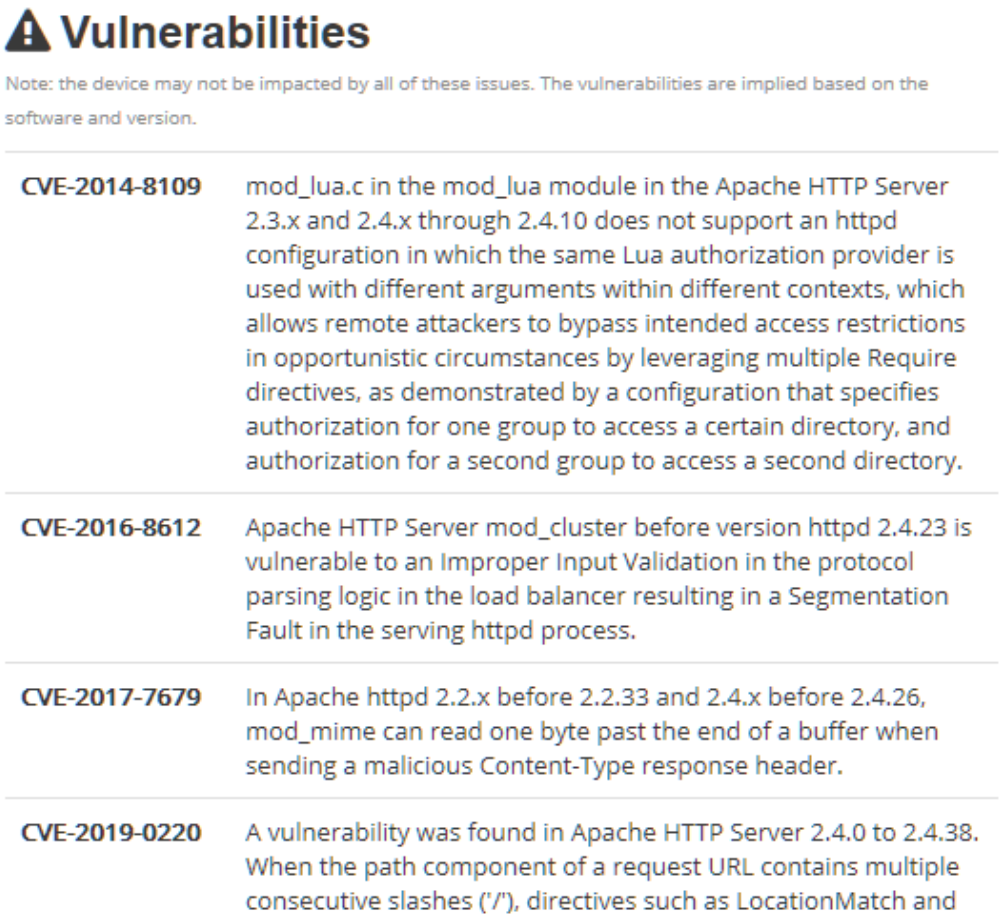

Illustration 4: Avec une vraie boite à outil de CVE..

Et même sans avoir tous vos ports sur Internet, le simple fait d'exposer votre Domoticz sur le Net vous expose aux CVE connues (et non encore connues). Une recherche sur un moteur de recherche de CVE comme https://www.cvedetails.com propose la CVE-2019-10789 avec :

- l'explication de la CVE
- un POC montrant l'exploit
- et éventuellement (mais pas ici) le module Metasploit permettant de l'utiliser !

Note : le moteur de recherche shodan ne scan pas l'ensemble des ports possibles, il teste uniquement un ensemble connu de ports. Un port ouvert 6789 ne devrait pas être visible depuis ce moteur de recherche...

1.4 Tout ça pour dire...

Tout d'abord, examinons les fragilités de la solution d'un Domoticz "Naté" :

1. Comme nous l'avons vu, exposer Domoticz directement sur le web permet à un attaquant de savoir qu'il est là.
2. Domoticz ne repose pas sur un serveur Web "mainstream" comme Apache ou Nginx. Le code du serveur Web est spécifique à Domoticz. Aussi, il y a relativement peu de développeurs s'occupant de ce code et relativement peu de testeurs également (comparativement à Apache ou Nginx). Il est fort probable que ce code souffre de vulnérabilités non connues.
3. Au-delà de la confiance que l'on peut avoir sur la partie Web de Domoticz, on peut se dire qu'il suffit de sécuriser son accès en https. Mais non en fait, car dans son utilisation la plus courante, Https permet juste à un utilisateur :
    - d'avoir confiance dans le serveur qu'il contact car le certificat utilisé est reconnu par le Browser et est dit de confiance,

- de chiffrer les interactions entre le serveur et le Browser afin de ne pas laisser quelqu'un observer les échanges.

4. Oui mais voilà, Domoticz ne dispose pas d'un certificat reconnu, aussi vous aurez systématiquement une demande d'autorisation pour certificat invalide à chaque connexion. Du coup, si on n'y fait pas gaffe, il peut être facile de faire un Man In The Middle pour regarder vos logins de connexions... (ok ce n'est pas aussi facile)
5. Et puis pour finir, l'usage de HTTPS qui en est fait ne permet en aucun cas pour Domoticz que vous avez bien les droits d'accès dessus... (en dehors du mot de passe qui aurait pu être sniffé avec un MiTN).

Pourtant, depuis que j'ai associé à mon Domoticz un système de surveillance, j'ai besoin de venir le contrôler de temps en temps pour voir ce qui a bien pu déclencher l'alarme et éviter de voir mes voisins m'accueillir avec un fusil sous prétexte que l'alarme a sonnée toute la nuit pour rien !?

2 Autre Solution... passons par un VPS

Aussi voici une solution différente qui devrait résoudre les différents points de fragilité énoncés dans la section précédente. Elle repose sur l'utilisation d'un VPS comme point d'accès extérieur (cf. Figure 5). Celui-ci doit offrir deux services :

- un serveur ssh qui va permettre à notre solution Domoticz de se connecter dessus
- un serveur Web (Apache ou Nginx) configuré en reverse proxy afin de servir de portail d'accès à notre Domoticz

L'idée est la suivante :

- Tout d'abord, le serveur Domoticz établi une connexion ssh vers le VPS pour établir un tunnel de communication sécurisé.
- Le point d'accès extérieur va être réalisé par un serveur Web configuré en double authentification https afin de garantir l'accès uniquement aux personnes autorisées
- Une fois la double authentification réalisée (entre le serveur Web et votre Browser), le flux https déchiffré (donc http) va être redirigé en local sur le VPS à l'adresse 127.0.0.1:6789
- Le tunnel ssh va ensuite transporter l'ensemble des communications arrivant en 127.0.0.1:6789 vers votre serveur Domoticz en 8080, bien au chaud derrière votre box.

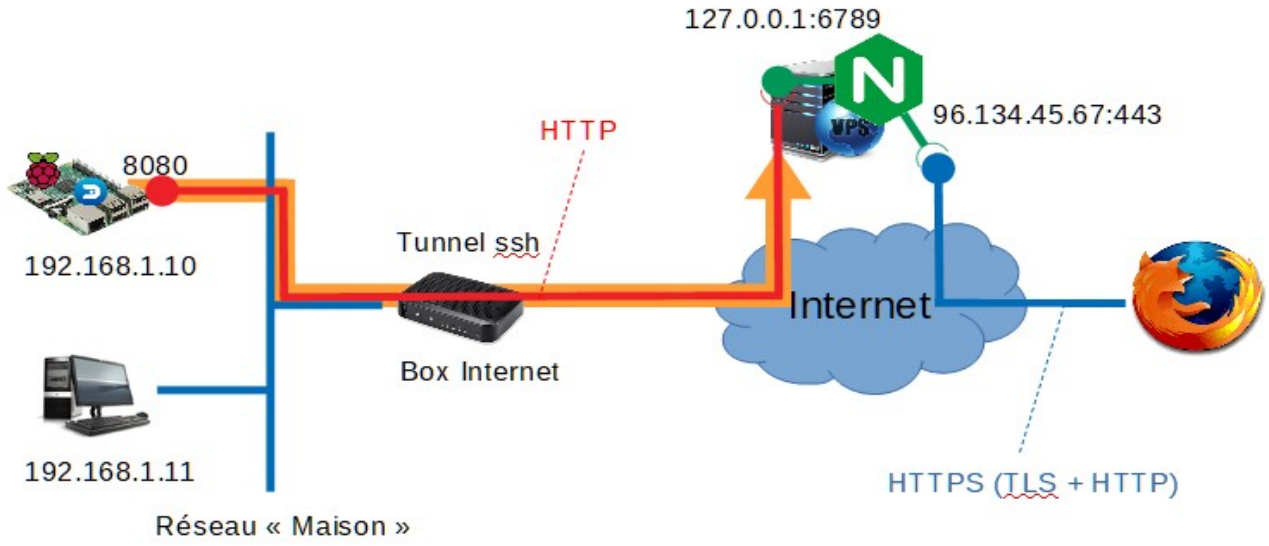

Illustration 5: Accès depuis un VPS

Ainsi :

1. Domoticz n'est pas identifiable depuis l'extérieur. Un scan de votre Box Internet n'offre plus aucun port ouvert, et un scan de votre VPS ne montrera que le port HTTPS et le port du serveur SSH ouvert (que nous vous conseillons de changer).
2. Les services exposés sur votre VPS sont supportés par une vaste communauté. Les mises à jour pourront suivre les nouvelles CVE.
3. La double authentification mise en place au niveau du serveur Web empêchera les curieux de venir explorer votre VPS.
4. Votre serveur Domoticz ainsi que votre LAN n'est plus exposé en première ligne. Pour y accéder, il faudra que le Hacker pawn d'abord votre VPS pour ensuite rebondir sur votre serveur Domoticz.
5. Le tunnel SSH est réalisé à la discrétion de votre serveur Domoticz. Il n'est pas utile de le maintenir tout le temps. Il pourrait également être activé après envoie d'un sms sur votre Domoticz par exemple.

3 Le Tunneling SSH

Nous allons tout d'abord réaliser le tunneling SSH entre notre serveur Domoticz et notre VPS. Pour cela, nous allons devoir :

- créer un couple de clé privée / publique lié à un utilisateur "domoticz"
- distribuer la clé publique sur le VPS
- tester la connexion,
- faire une connexion sans shell mais avec un forward de port
- tester le forward de port

3.1 Création d'une paire de clés SSH

Le protocole SSH peut utiliser des clés asymétriques pour permettre une connexion sécurisée. Aussi, depuis le serveur Domoticz nous allons générer une paire de clés SSH pour le user "pi" (utilisateur sous domoticz) avec la commande suivante :

```
pi@domobox:~ $ ssh-keygen -b 4096
```

Vous allez ensuite avoir le message suivant qui vous propose de générer les clés dans le répertoire .ssh de l'utilisateur courant (je conseille de garder ce répertoire).

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pi/.ssh/id_rsa):
```

Puis il vous est demandé un mot de passe... Normalement, il est fortement conseillé d'en mettre une afin de protéger votre clé privée. Nous n'allons pas en mettre ici...

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Voilà c'est fini :

```
Your identification has been saved in /home/pi/.ssh/id_rsa.
Your public key has been saved in /home/pi/.ssh/id_rsa.pub.
The key fingerprint is:
a4:eb:04:ff:5e:fb:57:80:8d:66:d7:25:84:6b:d1:77 pi@domobox
The key's randomart image is:
+---[RSA 4096]----+
|             +.  |
```

```
|           o o E|
|        .    * +o|
|       o    B + .|
|    . . S  + . . |
|     o .        .|
|      +   .    . |
|     o . . .  .  |
|      ..o ....   |
+-----------------+
```

La commande a produit deux fichiers :

- /home/pi/.ssh/id_rsa qui est la clé privé (à garder très précieusement)
- /home/pi/.ssh/id_rsa.pub qui est la clé publique que nous allons diffuser sur le serveur VPS.

Note : pourquoi ne pas utiliser une passphrase ? Normalement une passphrase est fortement conseillée avec l'utilisation de l'authentification par clé ssh. Par contre, celle-ci devra être entrée par l'utilisateur à chaque connexion. Il existe des solutions avec ssh-cache/ssh-ident mais cela va tout de même nécessiter de saisir la passphrase à chaque reboot. Dans certains blogs, il est proposé de contourner cette limitation en passant la passphrase au directement au processus, mais cela implique que le mot de passe soit en clair présent sur la même machine que la clé privée... ce qui limite l'intérêt de protéger sa clé avec une passphrase. Par contre, j'ai configuré le serveur SSH sur le VPS pour n'accepter que des connexions utilisateur (et pas route) avec certificat uniquement.

3.2 Diffusion de la clé publique sur le VPS

Nous allons utiliser la commande ssh-copy-id pour diffuser notre clé publique sur le VPS. Depuis le serveur domoticz, nous écrivons la commande suivante :

```
pi@domobox:~ $ ssh-copy-id vpsUser@96.134.45.67
```

Le message va ensuite apparaître :

```
The authenticity of host '96.134.45.67 (96.134.45.67)' can't be established.
ECDSA key fingerprint is df:ff:1e:c5:70:f7:de:03:ae:72:c2:df:d5:6a:d7:00.
Are you sure you want to continue connecting (yes/no)? yes
```

Tapez **yes** dans la ligne de commande et appuyez sur Entrée. Ce message apparaît uniquement à la première connexion. Puis le message suivant s'affichera :

```
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
```

Puis vous demandera le mot de passe du compte utilisateur sur votre vps:

```
vpsUser@96.134.45.67's password:
```

Le prompt suivant vous indique qu'une clé vient d'être rajoutée sur le VPS (ajout de la clé dans le fichier /home/vpsUser/.ssh/authorized_keys)

```
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'vpsUser@96.134.45.67'"
and check to make sure that only the key(s) you wanted were added.
```

Comme indiqué, vous pouvez vérifier que vous pouvez vous connecter sur le vps depuis votre serveur

domoticz:

```
pi@domobox:~ $ ssh vpsUser@96.134.45.67
```

Et enfin, vous être connecté sur votre VPS !

```
Linux 96.134.45.67 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u1 (2019-07-19) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have mail.
Last login: Tue Jul 30 22:42:45 2019 from xxx.xxx.xxx.xxx
```

Note : vous avez du remarqué que la connexion sur le VPS n'a pas nécessité le mot de passe utilisateur. C'est grâce à la clé publique que vous avez copié ; au moment de la connexion, un "challenge" a été chiffré grâce à la clé publique et envoyé à votre serveur Domoticz. Grâce à sa clé privée, celui-ci à pu déchiffrer le challenge et certifier de la bonne authentification.

Note : le principe ici consiste à envoyer uniquement une paire de clé du serveur Domoticz vers votre VPS (et non de façon bidirectionnelle). Cela permet d'autoriser uniquement les connexions Domoticz vers votre VPS (et non l'inverse).

3.3 Mettre en place le forward de port

Nous avons maintenant la possibilité d'ouvrir un shell distant sur notre VPS depuis notre box Domoticz. Mais ce n'est pas vraiment notre objectif car nous voulons juste rediriger le flux http émis localement sur le VPS en 127.0.0.1 sur le port 6789 vers notre Box. Pour cela, nous allons utiliser la fonctionnalité "forward de port" offerte par le protocole SSH.

Le forward de port (cf. Figure 6) permet de transporter des paquets TCP au travers un tunnel d'un port TCP à un autre. Dans notre figure, un tunnel SSH est réalisé à l'initiative de Domoticz pour qu'un serveur du côté du VPS puisse envoyer des paquets TCP sur l'adresse de la loopback (127.0.0.1) sur le port 6789. Quand un paquet arrive à cette adresse, il est pris en charge par le tunnel SSH qui le transporte jusqu'à l'arrivée (le port 8080). Bien entendue, la communication se fait dans les deux sens, mais doit avoir pour origine le VPS.

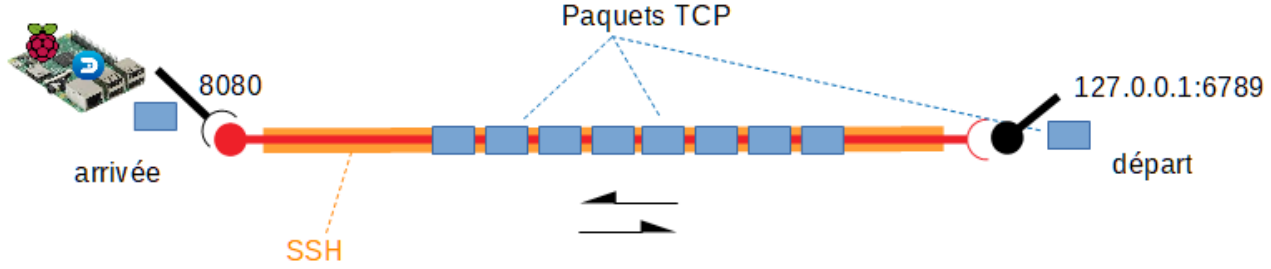

Illustration 6: Principe du forward de port

La création du tunnel SSH se fait avec la commande suivante (depuis notre box Domoticz) :

```
pi@domobox:~ $ ssh -N -R 6789:127.0.0.1:8080 vpsUser@96.134.45.67
```

avec pour arguments :

- -N ne pas exécuter de commande distante
- -R 6789:127.0.0.1:8080 : demande de création d'un forward de port sur le serveur distant de 127.0.0.1:6789 vers le serveur local (domoticz) en 8080
- vpsUser@96.134.45.67 : utilisateur et adresse du vps

Une fois la commande exécutée, on peut voir depuis le VPS que le tunnel est bien présent et qu'il a une connexion en écoute en localhost sur le port 6789 :

```
vpsUser@96.134.45.67:~/.ssh$ netstat -tnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:6789          0.0.0.0:*               LISTEN
```

Vous pouvez également obtenir la page html de Domoticz depuis votre VPS. Soit directement par un browser si X est installé soit par netcat comme ceci:

```
vpsUser@96.134.45.67:~/.ssh$ printf "GET /index.html HTTP/1.1\r\nUser-Agent: nc/0.0.1\
r\nHost: 127.0.0.1\r\nAccept: */*\r\n\r\n" | nc 127.0.0.1 6789 | grep "Domoticz"
```

et si tout fonctionne bien, on obtient une partie de la page HTML de Domoticz

```
<title>Domoticz</title>
var msgtxt=$.t('A new version of Domoticz is Available!...');
window.location.href = data.DomoticzUpdateURL;
ShowUpdateNotification(data.Revision,data.SystemName,data.DomoticzUpdateURL);
```

Note : il est très important d'ouvrir le forward de port sur l'adresse local du VPS (127.0.0.1). Ainsi, seule une connexion venant localement du VPS pourra emprunter le tunnel. Si vous utilisez l'adresse IP de votre VPS comme adresse d'accès, alors l'ensemble du Web pourra l'emprunter… ce qui n'était pas vraiment l'effet voulu...

3.4 Automatisation du lancement du tunnel ssh

Nous avons quasiment terminé cette première partie, il ne nous reste plus qu'à automatiser le lancement (ou le re-lancement du tunnel) en cas de coupure ou de reboot de votre Domoticz. Pour cela, nous proposons un script ('watchdog-tunnel.sh') lancé au boot, qui va superviser le processus ssh et le relancer au besoin :

```
#!/bin/bash
cmd="sudo -u pi ssh -N -R 6789:127.0.0.1:8080 vpsUser@96.134.45.67"
until $cmd; do
    echo "Tunnel 'myVPS' crashed with exit code $?.  Respawning.." >&2
    sleep 60
done
```

Note : comme les droits au boot son root, nous lançons le tunnel par un sudo afin de lancer en tant que utilisateur "pi".

Nous allons ensuite installer ce script comme un service. Pour cela, copions le dans /usr/local/bin

```
pi@domobox:~ $ sudo cp watchdog-tunnel.sh /usr/local/bin/
```

On lui donne les droits lecture/écriture/exécution uniquement pour le root

```
pi@domobox:~ $ sudo chmod 700 /usr/local/bin/watchdog-tunnel.sh
```

Maintenant, préparons le service qui va lancer notre script. On écrit un fichier 'run-tunnel.service' dans /etc/systemd/system

```
[Unit]
Description=Open Tunnel with VPS
After=network.target
[Service]
Type=oneshot
ExecStart=/usr/local/bin/watchdog-tunnel.sh
[Install]
WantedBy=multi-user.target
```

Puis on l'active :

```
pi@domobox:~ $ sudo systemctl daemon-reload
pi@domobox:~ $ sudo systemctl enable run-tunnel.service
Created symlink from /etc/systemd/system/multi-user.target.wants/run-tunnel.service
to /etc/systemd/system/run-tunnel.service.
```

Note : l'utilisation de la crontab au boot aurait pu permettre de lancer notre script, mais dans mes essais, celui-ci était parfois lancé avant l'initialisation du réseau... d'où mon choix pour la solution 'service'.

3.5 Détection de coupure du tunnel

Si le tunnel n'est pas utilisé et qu'il se retrouve coupé, le protocole TCP n'a pas la possibilité de détecter la rupture de la communication. Aussi, nous allons ajouter des options pour maintenir le flux ssh ouvert en envoyant périodiquement des paquets.

Pour cela, coté client (serveur Domoticz) nous ajoutons un fichier de configuration dans ~/.ssh/config :

```
Host *
ServerAliveInterval 300
ServerAliveCountMax 2
```

Et du côté serveur (votre VPS) dans /etc/ssh/sshd_config:

```
ClientAliveInterval 300
ClientAliveCountMax 2
```

Avec ces options, des paquets vides vont être échangés toutes les 300 secondes. En cas d'absence de réponse et après 2 essais, la connexion va être considérée comme abortée (ce qui sera détecté par le script watchdog qui essaiera de relancer le tunnel).

Pour vérifier que le tout fonctionne, on reboot !

```
pi@domobox:~ $ sudo reboot
```

Le tunnel doit être lancé après le reboot :

```
pi@domobox:~ $ ps -aef | grep "ssh"
root       827     1  0 08:39 ?        00:00:00 /usr/sbin/sshd -D
root       938   783  0 08:39 ?        00:00:00 sudo -u pi ssh -N -R
6789:127.0.0.1:8080 vpsUser@96.134.45.67
pi         942   938  0 08:39 ?        00:00:00 ssh -N -R 6789:127.0.0.1:8080
vpsUser@96.134.45.67
root       943   827  0 08:39 ?        00:00:00 sshd: pi [priv]
pi         972   943  0 08:40 ?        00:00:00 sshd: pi@pts/0
pi        1022   975  0 08:42 pts/0    00:00:00 grep --color=auto ssh
```

4 Accès Web avec double authentification

Nous allons maintenant nous occuper de la dernière partie... Le front-end web qui va nous permettre de rebondir au niveau du forward de port que nous venons de créer. Pour cela nous allons :

- installer un serveur web nginx
- configurer ce serveur en https et en reverse proxy
- créer une paire de certificat pour l'authentification
- ainsi qu'une autorité de certification

4.1 Installation du serveur nginx

Tout d'abord nous avons besoin d'un serveur Http qui va gérer les connexions depuis le Web. J'ai personnellement une préférence pour Nginx... mais Apache fait également très bien l'affaire.

```
root@96.134.45.67:/home/vpsUser# apt install nginx
Reading package lists... Done
Building dependency tree
Reading state information... Done
...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for systemd (241-5) ...
```

On peut maintenant vérifier l'installation avec une commande nginx:

```
root@96.134.45.67:/home/vpsUser# printf "GET / HTTP/1.1\r\nUser-Agent: nc/0.0.1\r\
nHost: 96.134.45.67\r\nAccept: */*\r\n\r\n" | nc 96.134.45.67 80
```

afin d'obtenir la page http d'invite de Nginx :

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 07 Aug 2019 07:16:19 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Wed, 07 Aug 2019 07:11:50 GMT
Connection: keep-alive
ETag: "5d4a79b6-264"
Accept-Ranges: bytes
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
</body>
</html>
```

Les communications Http étant visibles par tous (mot de passe compris) nous allons maintenant configurer Nginx pour utiliser Http(s) (S: pour sécurisé, et surtout chiffré).

4.2 Création du certificat serveur

La première étape consiste en la création d'un certificat permettant d'authentifier votre serveur (votre VPS). Il faut tout d'abord générer une clé privée:

```
vpsUser@96.134.45.67:$ openssl genrsa -out vpsServer.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
...++++
.....................................................................................
.....................................++++
e is 65537 (0x010001)
```

Il faut ensuite faire une demande de certificat à partir de la clé privée :

```
vpsUser@96.134.45.67:$ openssl req -new -sha256 -key vpsServer.key -out vpsServer.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:96.134.45.67
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Note : Common Name doit contenir le nom de domaine de votre serveur ou son adresse IP si vous n'en avez pas

Cette commande produit un fichier vspServer.csr (csr pour Certificat Signing Request). Il contient la clé publique issue de votre clé privée.

Maintenant, vous avez 2 possibilités :

- Soit vous envoyez cette demande de certificat à un organisme (autorité de certification) afin d'obtenir un certificat valide signé par la clé privée de l'organisme. Cette option est obligatoire dans le cas où votre serveur Web doit être utilisé par un large publique. (votre certificat sera reconnu par les Browser Web du fait de la certification officielle).
- Comme notre "site Web" ne devra être accessible que par nous-même, nous choisissons d'auto signer notre certificat (option choisie pour cet article).

4.3 Création d'une autorité de certification

Du coup, pour répondre à notre demande de certification, nous allons devoir créer notre propre autorité de certification. Nous allons tout d'abord créer la clé privée de notre autorité (mais avec un mot de passe pour la protéger, car une personne malveillante en possession de cette clé pourra produit n'importe quel certificat).

```
vpsUser@96.134.45.67:$ openssl genrsa -aes256 -out myCA.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....................................................................++++
..........++++
e is 65537 (0x010001)
Enter pass phrase:
Verifying - Enter pass phrase:
```

A partir de cette clé privée, nous produisons un certificat x509 pour une durée d'un an.

```
VpsUser@96.134.45.67:$ openssl req -new -x509 -sha256 -days 360 -key myCA.key -out
myCA.crt
Enter pass phrase for myCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyLittleCorp
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:MyLittleCorp
Email Address []:
```

Note: comme nous avons ajouté un mot de passe à la clé privée, celui-ci est demandé à la création du certificat.

Note: le Common Name doit être différent du Common Name de la demande de certificat du serveur.

4.4 Signature de la demande de certificat du VPS

Nous pouvons maintenant utiliser notre propre autorité de certification pour signer la demande de certificat de notre VPS (vpsServer.crt) :

```
vpsUser@96.134.45.67:$ openssl x509 -req -sha256 -in vpsServer.csr -out vpsServer.crt -
CA myCA.crt -CAkey myCA.key -CAcreateserial -CAserial myCA.srl -out vpsServer.crt
```

Ce qui donne :

```
Signature ok
subject=C = AU, ST = Some-State, O = Internet Widgits Pty Ltd, CN = MyLittleVPS
Getting CA Private Key
Enter pass phrase for myCA.key:
```

Le certificat vpsServer.crt vient d'être créé. Nous pouvons en vérifier la validité avec la commande suivante :

```
vpsUser@96.134.45.67:$ openssl verify -CAfile myCA.crt vpsServer.crt
vpsServer.crt: OK
```

4.5 Double authentification

Grâce à l'autorité de certification que nous avons créée et le certificat vpsServer.crt nous sommes maintenant en mesure de réaliser une connexion HTTPS entre votre Browser et le serveur Nginx de votre VPS. Cela permet de protéger l'ensemble des communications entre ses deux entités. Par contre, toutes les connexions seront acceptées quel que soit leurs provenance..

Aussi nous allons maintenant mettre en œuvre un mécanisme permet au serveur Nginx de valider que le Browser qui est en train de se connecter en a bien le droit. Pour cela, nous allons demander une double authentification entre le serveur Web et votre Browser (et ainsi exclure toutes les autres connexions ne montrant pas 'pattes blanches').

Nous avons besoin de créer un certificat client afin de vous authentifier. Pour cela, on va suivre le même principe que précédemment :

- création d'une clé privée
- création d'une demande de certificat à partir de cette clé
- validation de cette demande à partir de notre autorité de certification

Commençons par créer une clé privée :

```
vpsUser@96.134.45.67:$ openssl genrsa -out user.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....................................................................................
.....................................................................................
.........................++++
.....................................................................................
.................++++
e is 65537 (0x010001)
```

Ensuite, produisons une demande de certification :

```
vpsUser@96.134.45.67:$ openssl req -new -key user.key -out user.csr
```

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:user
Common Name (e.g. server FQDN or YOUR name) []:Browser
Email Address []:
```

(Inutile d'ajouter d'extra)

Puis nous authentifions cette demande avec notre propre autorité :

```
vpsUser@96.134.45.67:$ openssl x509 -req -days 365 -sha256 -in user.csr -CA myCA.crt -
CAkey myCA.key -set_serial 2 -out user.crt
Signature ok
subject=C = US, ST = Some-State, O = Internet Widgits Pty Ltd, OU = user, CN = browser
Getting CA Private Key
Enter pass phrase for myCA.key
```

Pour résumer, nous avons produit :

- notre propre autorité de certification représentée par une clé privée (myCA.key) à garder bien au chaud (dans un conteneur chiffré c'est pas mal)
- ainsi que le certificat proprement dit (myCA.crt). Il va devoir être connu de notre Browser favoris afin que celui-ci le considère comme un tiers de confiance. Dans le cas de la double authentification, le serveur va également avoir besoin de ce certificat afin de vérifier que le certificat de l'utilisateur a bien été validé par cette autorité.
- une "carte d'identité" de notre serveur VPS sous la forme d'une clé privée (myVPS.key) et d'un certificat (myVPS.crt). Ces deux éléments vont devoir être copiés sur le VPS au niveau de la configuration de Nginx.
- une "carte d'identité" du client afin que le serveur Web puisse vérifier que le Browser qui est en train de se connecter possède bien cette autorisation.

4.6 Configuration de Nginx

Sur notre VPS, commençons par créer une arborescence permettant de recueillir les certificats dans Nginx :

```
root@96.134.45.67:/# mkdir -p /etc/nginx/certs/server
root@96.134.45.67:/# mkdir -p /etc/nginx/certs/ca
```

afin d'y copier nos certificats :

```
root@96.134.45.67:/# cp /home/vpsUser/work/certificats/myCA.crt /etc/nginx/certs/ca/
root@96.134.45.67:/# chmod 400 /etc/nginx/certs/ca/myCA.crt
root@96.134.45.67:/# cp /home/vpsUser/work/certificats/vpsServer.crt
/etc/nginx/certs/server/
root@96.134.45.67:/# chmod 400 /etc/nginx/certs/server/vpsServer.crt
root@96.134.45.67:/# cp /home/vpsUser/work/certificats/vpsServer.key
/etc/nginx/certs/server/
root@96.134.45.67:/# chmod 400 /etc/nginx/certs/server/vpsServer.key
```

Puis nous allons maintenant créer l'accès à notre serveur Domoticz depuis le Nginx installé sur notre VPS. Pour cela, nous allons faire une règle de redirection des urls /domoticz vers notre forward de port en 127.0.0.1:6789. Pour faire simple, on va s'inspirer de la configuration par défaut

```
root@96.134.45.67:/# cp /etc/nginx/sites-available/default
/etc/nginx/sites-available/domoticz
```

Nous éditons ensuite ce fichier pour :

- écouter uniquement sur le port 443 (https)

```
listen 443 ssl default_server;
listen [::]:443 ssl default_server;
```

- donner à Nginx le chemin vers le certificat du server ainsi que sa clé privée

```
ssl_certificate /etc/nginx/certs/server/vpsServer.crt;
ssl_certificate_key /etc/nginx/certs/server/vpsServer.key;
```

- activer la double authentification et donner le certificat de notre autorité de certification

```
ssl_client_certificate /etc/nginx/certs/ca/myCA.crt;
ssl_verify_client on;
```

- créer 2 règles de redirections permettant de rediriger l'url https://96.134.45.67/domoticz/* -> http://127.0.0.1:6789

```
location /domoticz/ {
    proxy_pass http://127.0.0.1:6789/;
        proxy_redirect off;
        ...
}
location /domoticz/(.*) {
    proxy_pass http://127.0.0.1:6789/$1;
    proxy_redirect off;
        ...
}
```

Le fichier complet est disponible ici :

https://github.com/Hagrou/Domoticz-Remote-Access/blob/master/domoticz_nginx_conf

Une fois la configuration effectuée, on supprime le lien de la configuration par défaut des sites actifs:

```
root@96.134.45.67:/# unlink /etc/nginx/sites-enabled/default
```

Puis on créé un lien sur notre nouvelle configuration

```
root@96.134.45.67:/# ln -s /etc/nginx/sites-available/domoticz /etc/nginx/sites-enabled/domoticz
```

Il vous suffit ensuite de redémarrer Nginx pour qu'il prenne en compte la configuration.

4.7 Installation de votre CA dans le Browser

La partie serveur est maintenant terminée, nous devons nous occuper maintenant de la partie Browser. Comme notre autorité de certification n'est connue que de nous-même puisque nous l'avons auto signée, il faut explicitement l'importer dans notre Browser pour que celui-ci considère la communication de confiance.

Avec Firefox, l'import se fait via Preferences->Advanced->Certificates->View Certificates->Authorities->Import

Puis charger le fichier vpsServer.crt

Si vous rechargez le site, vous allez avoir maintenant une erreur :

```
400 Bad Request, No required SSL certificate was sent.
```

C'est normal, le serveur Nginx demande maintenant le certificat utilisateur afin de vérifier que celui-ci a bien été validé par votre autorité de certification.

4.8 Installation du certificat utilisateur

L'installation d'un certificat utilisateur nécessite un fichier au format PKCS#12 qui va contenir à la fois le certificat et la clé privée de l'utilisateur. On va le générer à partir de la commande suivante :

```
# openssl pkcs12 -export -clcerts -in vpsUser.crt -inkey vpsUser.key -out vpsUser.p12
Enter Export Password:
Verifying - Enter Export Password:
```

(Un mot de passe est nécessaire pour protéger ce certificat).

5 Conclusion

Ouf ! Voilà c'est fait. Je ne peux pas garantir que cette solution vous offre une sécurité maximale, mais en tout cas vous n'avez plus de point d'entrée à votre réseau local depuis votre box et Domoticz n'est plus en première ligne, d'ailleurs il n'apparaîtra plus dans Shodan.

Personnellement j'utilise cette solution depuis quasiment une année et j'accède à mon serveur Domoticz depuis mon smartphone sans problème.

2 derniers conseils pour la route.. Ajouter un mot de passe HTTP à domoticz histoire de ralentir un éventuel intrus c'est toujours rassurant, et surtout **SURTOUT** restez toujours à jour histoire de ne pas prêter le flanc à une CVE qui traîne…. .

Quelques liens utiles :

https://www.hostinger.fr/tutoriels/generer-cle-ssh/ : tuto sur l'échange de clés

https://docs.ovh.com/fr/public-cloud/creation-des-cles-ssh/ : petit tuto pour la création de clés ssh depuis un linux ou un windows

http://www.unixwiz.net/techtips/ssh-agent-forwarding.html : si vous êtes curieux voici une explication illustrée des mécanismes d'authentification ssh

https://fr.wikipedia.org/wiki/Certificat_électronique : une explication sur le fonctionnement des certificats

http://linux-france.org/prj/edu/archinet/systeme/ch24s03.html : un article bien fait sur la création des certificats pour l'HTTPS (dont je me suis inspiré)

https://blog.codeship.com/how-to-set-up-mutual-tls-authentication/ : article sur l'authentification mutuelle HTTPS ainsi que la configuration du Nginx avec une règle de redirection d'url.