Sécuriser un serveur SSH

Vous utilisez **ssh** pour vous connecter à votre serveur ou PC depuis l'extérieur et donc vous avez ouvert via une règle NAT/PAT le port de votre box pour la rendre accessible depuis l'extérieur. Ou vous avez un serveur (dédié ou vps) que vous administrez en ssh.

Voilà quelques règles de base à respecter pour limiter les intrus tapant à votre porte...

Bien que **fail2ban** utilise des règles **iptables**, nous n'aborderons pas **iptables** ici. Mais il est conseillé d'avoir un parefeu en plus qui gère la totalité des connexions au serveur / PC.

Le numéro du port ssh modifié donné en exemple est bien sûr à adapter ;)

1. Changer le port ssh par défaut

Sur le serveur

```
serveurProliant@serveur# nano /etc/ssh/sshd_config
Port 22
```

changer en

```
Port 6789
```

(port : tous les ports non utilisés de 1024 à 65535)

Redémarrer le service SSH

```
serveurProliant@serveur# systemctl restart ssh
```

Dès lors pour se connecter , on précisera le numéro du port

```
serveurProliant@serveur# ssh login@serveur.ext -p 6789
```

Si Box / Routeur En local, rien de plus à faire. Reste maintenant à rediriger le port 22 de votre box au port 6789 de votre serveur/PC. Pour cela, section administration, et modifier la règle comme dans l'image ci-dessous (pour une livebox pro, mais sensiblement identique pour toutes les box).

Dès lors se connecter depuis l'extérieur

```
serveurProliant@serveur# ssh login@serveur.ext -p 6789
```

2. S'identifier par clef SSH et non par mot de passe

Sur le client :

```
serveurProliant@serveur$ ssh-keygen
Your identification has been saved in /home/$USER/.ssh/id_rsa cat
~/.ssh/id_rsa.pub
```

Envoyer la clef au serveur

```
serveurProliant@serveur# cat ~/.ssh/id_rsa.pub | ssh login@serveur.ext -p
6789 “mkdir -p ~/.ssh && cat » ~/.ssh/authorized_keys”
```

Dès lors on se connectera avec le mot de passe du trousseau de clefs et non avec celui du compte **ssh**. Le mot de passe ssh ne transitera alors plus sur le réseau.

3. Supprimer l'identification par mot de passe

Lié inévitablement à l'étape précédente, on bloque les identifications par mot de passe et on accepte que celles via des trousseaux de clefs.

```
serveurProliant@serveur# nano /etc/ssh/sshd_config
# Modifiez ou ajoutez la ligne suivante
PasswordAuthentication no
```

Redémarrer le service ssh

```
serveurProliant@serveur# systemctl restart ssh
```

4. Limiter la connexion à certains users

On définira ici que les utilisateurs qui ont le droit à se connecter en ssh.

Éditer le fichier de configuration

```
serveurProliant@serveur# nano /etc/ssh/sshd_config
```

Et rajouter la ligne suivante (par défaut, elle est absente)

```
AllowUsers nom_utilisateur_autorisé
```

(si plusieurs utilisateurs, les séparer par un espace)

5. Installer et configurer fail2ban

Programme qui va analyser les logs (ssh, apache, nginx, ftp…) et rechercher les tentatives de connexions infructueuses afin de les bloquer/bannir (en ajoutant des règles au firewall). iptables). Il met donc “en prison” les services et va gérer leur connexion (les laisse passer ou les bannit)

Installer fail2ban

```
serveurProliant@serveur# apt-get install fail2ban
```

L'installer ne suffit pas, il faut dès lors, peaufiner sa configuration.

Configurer fail2ban pour ssh <code bash>$ cat /etc/fail2ban/jail.d/mon_serveur.conf [DEFAULT] findtime = 3600 bantime = 86400 maxretry = 3 [sshd] port = 6789 [sshd-ddos] port = 6789</code> Explications findtime : on regarde dans les archives de log sur une période de 1 heure (3600 secondes). Si vous mettez une valeur trop haute, vous risquez une charge système très importante… Si on trouve dans ces logs l'IP 3 fois mal identifiée (maxretry=3), on la bannit (ici 86400 secondes soit 1 journée). Pendant donc une journée, cette IP sera donc rejetée automatiquement. Bien sûr, il est conseillé, si vous avez une IP fixe, d'ignorer cette règle sur cette IP (afin de vous éviter de mauvaises surprises (vous ne pourrez plus vous connecter si vous bous plantez…)) et d'ajouter cette directive dans la section [DEFAULT]. Faire de même bien sûr avec la boucle locale : 127.0.0.1 <code bash>ignoreip = 127.0.0.1 123.45.67.89</code> (remplacé 123.45.67.89 par votre IP ou vos IPs (dans ce cas séparées par un espace) Vérifier que la prison est bien effective <code bash>serveurProliant@serveur# fail2ban-client status Status |- Number of jail: 2 `- Jail list: sshd, sshd-ddos</code> Savoir si des IP ont été bannises <code bash>serveurProliant@serveur# fail2ban-client status sshd Status for the jail: sshd |- Filter | |- Currently failed: 0 | |- Total failed: 0 | `- File list: /var/log/auth.log `- Actions |- Currently banned: 41 |- Total banned: 41 `- Banned IP list: 103.110.89.148 103.85.18.99 109.110.52.77 111.223.73.20 122.195.200.148 153.36.236.35 153.36.240.126 153.36.242.114 159.65.144.233 159.65.7.56 163.172.106.114 164.132.225.250 167.99.200.84 167.99.75.174 176.31.253.55 180.250.183.154 180.96.14.98 181.99.48.120 183.131.82.100 183.131.82.99 187.60.97.209 190.119.190.122 192.168.1.10 193.32.163.182 200.0.236.210 206.189.197.48 206.248.181.122 210.206.179.221 222.76.119.165 23.101.133.58 46.105.244.1 5.135.223.35 51.75.247.13 52.172.44.97 54.39.196.199 61.216.15.225 73.144.161.209 83.147.102.62 87.99.77.104 91.134.227.180 92.91.60.249 serveurProliant@serveur# fail2ban-client status sshd-ddos Status for the jail: sshd-ddos |- Filter | |- Currently failed: 0 | |- Total failed: 0 | `- File list: /var/log/auth.log `- Actions |- Currently banned: 0 |- Total banned: 0 `- Banned IP list: </code> Bannir une IP manuellement fail2ban-client set sshd banip 12.34.56.78 Débannir une IP manuellement fail2ban-client set sshd unbanip 12.34.56.78 6. Gérer les notifications. 6.1. Avoir un rapport des IP bannis Voici un petit script à position sur un cron (ici envoyer 2 fois par jour) donnant le statut des jails. # Rapport de fail2ban 15 6,19 * * * /home/admin/scripts/fail2ban-status-ban.sh Et le script serveurProliant@serveur# cat fail2ban-status-ban.sh #!/bin/sh # Script de rapport fail2ban # Prend en compte tous les jails # dest=votre.login@domaine.ext msg=$(fail2ban-client status | sed -n 's/,*g;s/.*Jail list:*p' | xargs -n1 fail2ban-client status); echo "$msg" | mail -s "Rapport Fail2ban De $(hostname -s) $(date)" $dest 6.2 Avoir un rapport de connexion SSH Un autre script pour être notifié d'une connexion SSH sur son serveur (notification par mail ou par SMS (pour ceux ayant un forfait free, même à 2 €). Permet également dans la configuration par SMS d'ignorer une IP donnée (ou plusieurs) afin que le téléphone ne passe pas son temps à sonner. Etre notifié par mail d'une connexion serveurProliant@serveur# cd /etc/ssh/ ; ls moduli ssh_host_ecdsa_key ssh_host_ed25519_key.pub ssh_config ssh_host_ecdsa_key.pub ssh_host_rsa_key sshd_config ssh_host_ed25519_key ssh_host_rsa_key.pub Dans ce répertoire, créer le script suivant, remplacer $DEST par votre mail. serveurProliant@serveur# nano sshrc #!/bin/sh $DEST=votre.mail@domaine.ext DATE=$(date "+%d.%m.%Y-%Hh%Mm") IP=$(echo $SSH_CONNECTION | awk '{print $1}')

REVERSE=$(dig -x $IP +short) MSG="Connexion de $(echo $USER) sur $(hostname -s) IP: $IP ReverseDNS: $REVERSE Date: $DATE" echo "$MSG" | mail -s "$(echo $DATE) : Connexion de $(echo $USER) sur $(hostname -s)" $DEST Ce script doit apartenir à root mais être accessible en lecture à tous. serveurProliant@serveur# ls -la … -rw-r–r– 1 root root 54 Jul 4 11:02 sshrc Etre notifié par mail ET SMS (free seulement) On devra créer 3 fichiers (les deux derniers sont positionnés dans un répertoire /home/TOTO/scripts/ ; à adapter à votre configuration) - /etc/ssh/sshrc - /home/TOTO/scripts/send-notification-data.txt - /home/TOTO/scripts/send-notification.sh Pour des raisons de sécurité, le fichier send-notification-data.txt ne devra être lisible que par root, il contient les données d'identification free et les données nécessaires au script. Si vous ne souhaitez pas recevoir de notification depuis une ou des IP(s) précise(s), mettre cela dans la variable $IP_AUTHORIZED. La variable $DEST contient le mail de notification. serveurProliant@serveur# cat send-notification-data.txt ## ## IP A AUTORISER SANS ALERTE SMS ## # Décommenter cette variable et saisir l'IP de connexion permise sans alerte # SMS. Si plusieurs IP, les séparer d'un espace IP_AUTHORIZED='12.345.67.890'; ## ## NOTIFICATION SMS ## # Login utilisateur / identifiant Free Mobile (celui utilisé pour accéder à # l'Espace Abonné) USER_LOGIN="123456789" # Clé d'identification (générée et fournie par Free Mobile via l'Espace Abonné, # "Mes Options" : https://mobile.free.fr/moncompte/index.php?page=options) API_KEY="aBcDeFgHiJkL" ## ## NOTIFICATION MAIL ## # Nom du destinaire de la notification par mail DEST=mon.mail@domaine.com Adpater les droits, très important : serveurProliant@serveur# chmod 600 send-notification-data.txt serveurProliant@serveur# ls -la … -rw——- 1 admin 1007 644 Jul 4 10:57 send-notification-data.txt On incluera ce fichier dans le script bash afin qu'on ne puisse pas lire son contenu. Enfin le script La fonction d'envoi via l'API de free est à l'origine ici : https://github.com/C-Duv/freemobile-smsapi-client [DUVERGIER Claude (http://claude.duvergier.fr)] Modifiée pour les besoins. serveurProliant@serveur# cat send-notification.sh #!/bin/sh # Données utilisateur . /home/admin/TOTO/send-notification-data.txt # ========== DATE=$(date "+%d.%m.%Y-%Hh%Mm") IP=$(echo $SSH_CONNECTION | awk '{print $1}') REVERSE=$(dig -x $IP +short) MSG="Connexion de $(echo $USER) sur $(hostname -s) IP: $IP ReverseDNS: $REVERSE Date: $DATE" ## ## La fonction d'envoi SMS ## fctEnvoiSms() { # Script d'envoi de notification SMS via l'API Free Mobile # https://github.com/C-Duv/freemobile-smsapi-client # Auteur:v # modification crust@crust.ovh readonly PROGNAME=$(basename $0) readonly PROGDIR=$(readlink -m $(dirname $0)) usage_error () { echo "ERROR: ${1}" >&2 echo "" usage_help exit 1 } usage_help () { echo "Possible usages:" echo "* ${PROGNAME} [options] [message]" echo "* echo \"All your base are belong to us\" | ${PROGNAME} [options]" echo "" echo "Options:" echo "* -c file specify configuration file" echo "* -h display this help" } CONFIG_FILE="" while getopts "c:h" option; do case "$option" in c) CONFIG_FILE=${OPTARG} ;; :) usage_error "Invalid arguments" ;; h) usage_help ; exit 0 ;; esac done shift $[1)] ## ## Configuration système ## # Caractère de fin de ligne # (http://en.wikipedia.org/wiki/Percent-encoding#Character_data) NEWLINE_CHAR="%0D" # Valeurs possibles : %0A, %0D et %0D%0A # URL d'accès à l'API SMSAPI_BASEURL="https://smsapi.free-mobile.fr" # Action d'envoi de notification SMSAPI_SEND_ACTION="sendmsg" # Texte qui sera ajouté AVANT chaque message envoyé MESSAGE_HEADER="$(date "+%d.%m.%Y-%Hh%Mm"): " # Texte qui sera ajouté APRÈS chaque message envoyé MESSAGE_FOOTER=" - $(hostname -s)" ## ## Fichier de configuration ## if [-n "${CONFIG_FILE}"]; then if [-e "${CONFIG_FILE}"]; then . "${CONFIG_FILE}" else echo "ERROR: Configuration file \"${CONFIG_FILE}\" does not

exists.” >&2 exit 2 fi else if [-e “${PROGDIR}/.freemobile-smsapi”]; then . “${PROGDIR}/.freemobile-smsapi” elif [-e “${HOME}/.freemobile-smsapi”]; then . “${HOME}/.freemobile-smsapi” fi fi ## ## Vérifications des paramètres requis ## if [-z “${USER_LOGIN}”] \ || [-z “${API_KEY}”] \ || [-z “${SMSAPI_BASEURL}”] \ || [-z “${SMSAPI_SEND_ACTION}”] \ ; then echo “ERROR: Either USER_LOGIN, API_KEY, SMSAPI_BASEURL or ” \ “SMSAPI_SEND_ACTION is not set” >&2 exit 2 fi ## ## Traitement du message ## MESSAGE_TO_SEND=“” if [“${1}”]; then # Message en tant qu'argument de la ligne de commande MESSAGE_TO_SEND=“${1}” else # Message lu de STDIN while read line do MESSAGE_TO_SEND=“${MESSAGE_TO_SEND}${line}\n” done MESSAGE_TO_SEND=${MESSAGE_TO_SEND%“\n”} # Retire le dernier saut de ligne fi # Assemble header, message et footer FINAL_MESSAGE_TO_SEND=“${MESSAGE_HEADER}${MESSAGE_TO_SEND}${MESSAGE_FOOTER}” ## ## Appel à l'API (envoi) ## # echo “Will send the following to ${USER_LOGIN}:” #DEBUG # echo “${FINAL_MESSAGE_TO_SEND}” #DEBUG # Converts newlines to $NEWLINE_CHAR FINAL_MESSAGE_TO_SEND=$(\ echo -n “${FINAL_MESSAGE_TO_SEND}” | \ sed '{:q;N;s/\n/'${NEWLINE_CHAR}'/g;t q}'\ ) # echo “Newline encoded message:” #DEBUG # echo “${FINAL_MESSAGE_TO_SEND}” #DEBUG # Particularités de l'appel de curl et la/les options associées : # * Renvoi le code réponse HTTP uniquement : # –write-out “%{http_code}” –silent –output /dev/null # HTTP_STATUS_CODE=$(\ curl \ –write-out “%{http_code}” \ –silent \ –output /dev/null \ –get “${SMSAPI_BASEURL}/${SMSAPI_SEND_ACTION}” \ –data “user=${USER_LOGIN}” \ –data “pass=${API_KEY}” \ –data “msg=${FINAL_MESSAGE_TO_SEND}” \ ) # Codes réponse HTTP possibles # 200 : Le SMS a été envoyé sur votre mobile. # 400 : Un des paramètres obligatoires est manquant. # 402 : Trop de SMS ont été envoyés en trop peu de temps. # 403 : Le service n'est pas activé sur l'espace abonné, ou login / clé # incorrect. # 500 : Erreur côté serveur. Veuillez réessayez ultérieurement. if [“${HTTP_STATUS_CODE}” -eq 200]; then # echo “API responded with 200: exiting with 0” #DEBUG exit 0 echo “Error: API responded with ${HTTP_STATUS_CODE}” else exit 1 fi } ## ## ENVOI SMS / MAIL ## if echo “$IP” | egrep $IP_AUTHORIZED ; then echo “MATCH NO SEND SMS”; else fctEnvoiSms “Connexion SSH de $(echo $USER) ; IP : $(echo $IP)” fi # QUOI QU'IL EN SOIT ON ENVOIE UN MAIL echo “$MSG” | mail -s “$(echo $DATE) : Connexion de $(echo $USER) sur $(hostname -s)” $DEST Après tout cela, vous devriez à a voir quelque chose qui tient la route… Par contre, un iptable en toile de fond reste bien sûr nécessaire….

1)

OPTIND-1