Filtrage d'accès (SSH) par « port knocking »

©
<bernard.perrot@univ-rennes1.fr>

Pourquoi ?

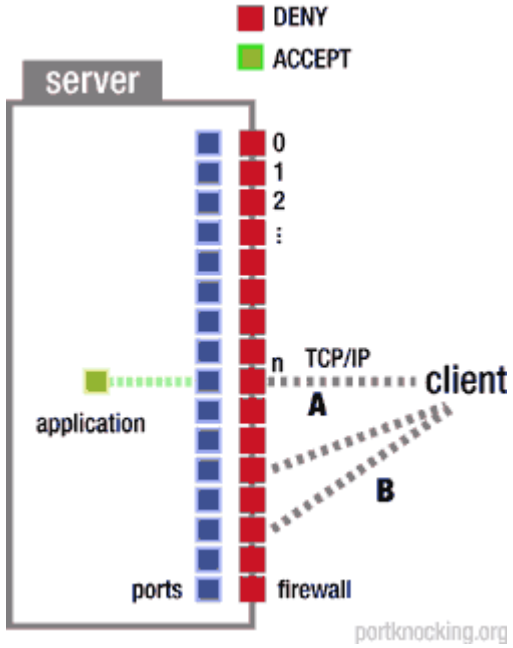
- Nombreuses attaques sur port SSH (22),
  ... et de plus en plus (>>1000/jours)
- Il apparaît que ces attaques sont faites par des automates,
  et qu'il s'agit d'attaques à dictionnaires
- Si :
  - Les mots de passe sont fiables
  - Le démon ne présente pas de faille (buffer overflow, ...)
  - Est correctement configuré (pas d'accès root, ...)
- ... on ne devrait rien craindre...
- Mais l'expérience semble démontrer que ces conditions peuvent ne pas être remplies...

Quoi faire ?

- N'autoriser que des machines connues : pas réaliste (liste impossible à stabiliser, grosse administration pour gérer cela)
- Interdire les attaquant connus : ça ne marche pas, ils changent sans cesse !
- Filtrer « dynamiquement » :
  - À postériori : système de blacklistage sur détection de tentative d'intrusion cette méthode ne préviendra pas une attaque réussie « de suite » (username = password, buffer overflow, ...)
  - À priori : rien n'est autorisé, sauf machines identifiées (idem filtrage statique, mais identifications automatisées => portknocking
- Chaque méthode a ses avantages et inconvénients, à chacun de faire en fonction de son cas particulier, de ses contraintes, de ses goûts...
- La méthode indiquée ici ne cherche à résoudre **que** ce problème (attaques par automates), pas à sécuriser SSH contre tous types d'attaques.

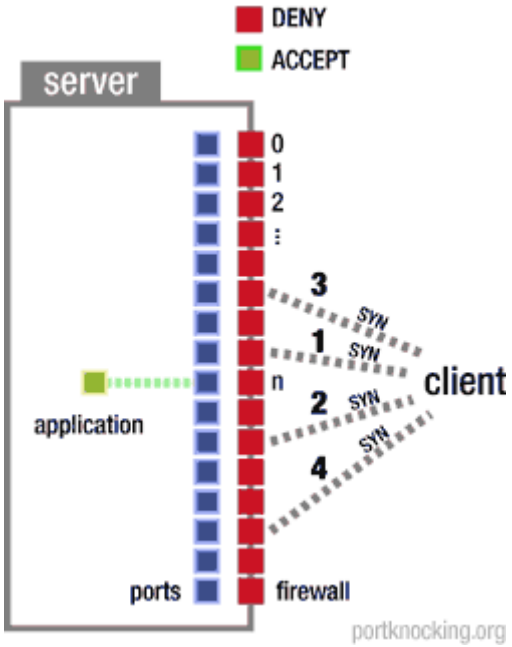
Port Knocking

- Une référence : http://www.portknocking.org/
- Un client veut se connecter à une application pour laquelle l'accès est fermé



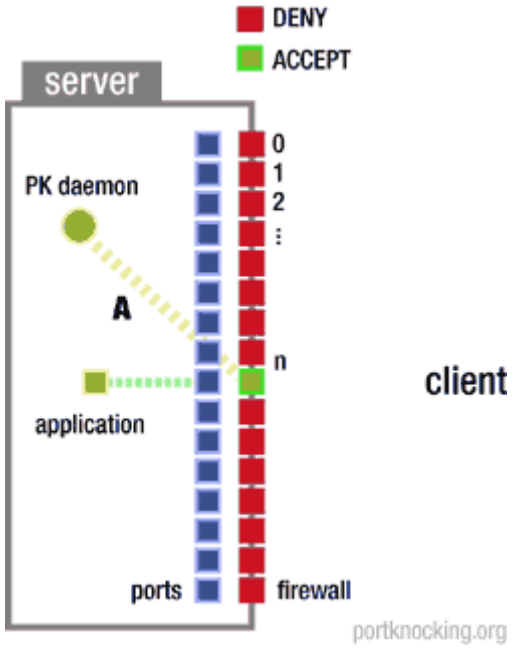
Port Knocking

- Le client, qui a connaissance du « secret » (existence d'un système de knocking, connaissance d'une séquence « clé »), envoi une séquence de paquets SYN (TCP ou UDP) pour s'identifier (déclarer son adresse IP comme fiable). La séquence peut être simple ou très complexe, c'est un choix de configuration.



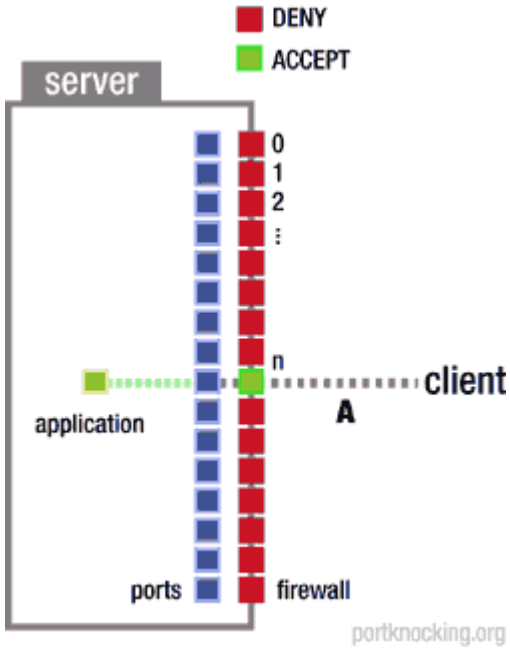
Port Knocking

- Le démon installé sur le serveur reconnaît la séquence de signalisation, et agit pour libérer l'accès au service demandé pour le client (@IP) identifié (libération temporaire ou définitive, par reconfiguration de règles de filtrage par exemple).



Port Knocking

- Le client peut accèder à l'application souhaitée
  NB: client = @IP, s'il s'agit d'un serveur, cela peut faire du monde.
  Dans le cas traité ici (SSH), cela n'est un problème que si ce serveur ne devient plus de confiance (cas très rare).



Port Knocking

- Il existe de très nombreuses implémentations, voir le site de référence indiqué.
- Pour le cas que je voulais traiter (au plus simple), j'ai retenu le logiciel « knockd » disponible ici :

  http://www.zeroflux.org/knock
- Contrainte : il faut que les ports utilisés pour le knocking soient ouverts au niveau réseau d'accès au site hébergeant le serveur (il faut demander au gestionnaire, CRI par exemple, de les laisser passer)
- Contrainte : il faut que les ports utilisés pour le knocking soient ouverts en sortie du site du client
- Nb : le système fonctionne très bien s'il y a du PAT/NAT en route.

knockd

- Voir ci-après le script de lancement au démarrage de la machine
- Les principales options sont :
  - **`-i, --interface <int>`**
    spécifie l'interface ethernet utilisée
  - **`-d, --daemon`**
    pour démarrer en démon
  - **`-c, --config <file>`**
    indique le fichier de configuration
  - **`-D, --debug`**
  - **`-v, --verbose`**
  - **`-h, --help`**

knockd.conf

- Le programme utilise un fichier de configuration, dont voici les directives:

- Directives globales
  (dans section **`[options]`**)
  - **`UseSyslog`** : utilisation de syslogd pour le journal
  - **`LogFile = /path`** : indique un fichier journal spécifique
  - **`PidFile = path`** : fichier contenant le pid du process en mode démon
  - **`Interface = iface-name`** : nom de l'interface réseau

knockd.conf

- Directives liées aux évènements
  (dans sections **[*nom_evt*]**) :
  - **Sequence = <port1>[<tcp:udp>][,<port2>[<tcp:udp>],...]**
    la séquence à détecter
  - **seq_timeout = <timeout>**
    durée pour réaliser la séquence
  - **start_command = <command>**
    la commande à exécuter quand la séquence a été reconnue
    (**command** est un alias pour **start_command**)
  - **cmd_timeout = <timeout>**
    durée d'attente entre la commande de début et la commande de fin
    (facultatif)
  - **stop_command = <command>**
    commande de fin si **cmd_timeout** indiqué

knockd.conf

- Autres directives :
  - **`TCPflags = fin|syn|rst|psh|ack|urg`**
    ne traite que les paquets ayant ces flags positionnés
  - **`One_Time_Seq=ences = <file>`**
    fichier contenant des séquences qui ne seront utilisées qu'uen seule fois, successivement.
- Notez que « knockd » ne connait rien au filtrage proprement dit, il ne fait qu'exécuter des commandes externes. Il est donc indépendant sur logiciel de filtrage utilisé, voire même de l'exact usage qui sera fait
- On peut aussi bien le faire tourner sur la machine ayant le service SSH que sur un garde-barrière situé en amont
  - Dans l'exemple décrit ci-après, il tourne sur la même machine que le serveur SSH (serveur Linux, filtrage Iptables, seule cette machine est visible de l'extérieur en SSH)

Misee en oeuvre

- Pour ma mise en production, le choix a été fait de faire au plus simple :
  - Deux ports seulement utilisés, en TCP
  - Ceci est documenté sur une page Web publique
  - Cela ne perturbe donc personne, sauf les automates qui ne lisent pas la page Web en question...
  - Pour le moment, efficacité à 100%, et pas d'abus du procédé (si cette séquence devait être abusée par des automates, il faudrait la changer ou la compliquer).
  - Comme il s'agit de « knock » TCP, il est possible d'utiliser n'importe quel logiciel qui tente d'ouvrir une connexion (donc envoie des SYN sur les ports), par exemple :
    `ssh -p 1228 serveur; ssh -p 1227 serveur; ssh -p 22 serveur`
  - Mais on peut utiliser aussi telnet ou un navigateur Web !
  - Knockd est fourni avec un utilitaire (Unix) pour transmettre la séquence

Mise en oeuvre

- Entre le « start_command » et le « stop_command », l'automate reste en attente
- Pour éviter cette situation, dans la mise en oeuvre présente, seule une commande « start » est exécutée par le démon knockd à la validation de séquence, puis ce sont des scripts « maison » qui gère le filtrage et sa durée de vie
- Il a été retenu :
  - 15 secondes pour réaliser la séquence (deux ports seulement...)
  - Ouverture du port (pour le client donné) pendant 5mn
    - NB: il s'agit de la durée pour ouvrir la session, une fois celle-ci ouverte, elle peut rester active aussi longtemps que nécessaire bien entendu.
- Il a été implémenté une « liste blanche » pour autoriser à demeure les machines considérées comme fiables (résoud les pbs pour sites distants filtrant en sortie les ports utilisés)

/etc/knockd/knockd.conf (1)

```
[options]
  usesyslog
  pidfile = /var/run/knockd.pid



[opencloseSSH]
  sequence    = 1228,1227
  seq_timeout = 15
  tcpflags    = syn
  command     = /etc/rc.d/init.d/knockd accept %IP%


#NB : les noms de machines, adresses IP et numéros de ports
#     indiqués dans cette présentation sont bien entendu fictifs...
```

/etc/knockd/knockd.conf (2)

```
[banned1]
  sequence    = 1220,1221
  seq_timeout = 15
  tcpflags    = syn
  command     = /etc/rc.d/init.d/knockd drop %IP%

[banned2]
  sequence    = 1224,1225
...
[banned3]
  sequence    = 1226,1226
...
[banned4]
  sequence    = 1223,1224
...
[banned5]
  sequence    = 1222,1222
...
```

/etc/rc.d/init.d/knockd (1)

```
#!/bin/sh
# Startup script for knockd
#
# chkconfig: 2345 97 24
# description: Port Knocking daemon

# Source function library.
. /etc/rc.d/init.d/functions

knockd="/usr/local/sbin/knockd"
[ -f $knockd ] || exit 0

srvsshname="ssh.maths.univ.fr"
accept_timeout=300

sshport=22
ipt="/sbin/iptables"
chain_filter="ssh_filter"
chain_drop="ssh_banned"
chain_policy="ssh_policy"
chain_trusted="ssh_trusted"

...
```

```
...

srvsship=$(dig +short $srvsshname)
srvssheth=$(grep "$srvsship" \
  /etc/sysconfig/network-scripts/ifcfg-eth*)
srvssheth=${srvssheth:37}
idx=$(expr index "$srvssheth" "IPADDR=" - 2 )
srvssheth=${srvssheth:0:$idx}

k_if=${srvssheth} # interface de srv-ssh

k_conf="/etc/knockd/knockd.conf"
k_trusted="/etc/knockd/knockd.trusted"
k_opts="-d -i $k_if -c $k_conf"

prog="knockd"

...
```

/etc/rc.d/init.d/knockd (2)

```
...

case "$1" in
   start)
        setup_iptables
        start
        ;;

   stop)
        stop
        ;;

   status)
        status $knockd
        ;;

   restart)
        stop
        start
        ;;

   condrestart)
        if test "x`pidof $knockd`" != x; then
      stop
      start
        fi
        ;;

   restartifdead)
        status $knockd > /dev/null 2>&1
        RETVAL=$?
        if [ "$RETVAL" = "1" ]; then
          echo $"Knockd dead... restarting..."
          start
        fi
        ;;
   accept)
        accept $2
        ;;
   flush)
        flush $2
        ;;
   drop)
        drop $2
        ;;
   trust)
        trust $2
        ;;
   setup)
        setup_iptables
        ;;
   *)
        echo $"Usage: ...
        exit 1
esac
exit 0
```

/etc/rc.d/init.d/knockd (2)

```
start() {
    echo -n $"Starting $prog: "
    daemon $knockd $k_opts
    RETVAL=$?
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/$prog
    echo
    return $RETVAL
}

stop() {
    if test "x`pidof $knockd`" != x; then
  echo -n $"Stopping $prog: "
  killproc $knockd
  echo
    fi
    RETVAL=$?
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/$prog
    return $RETVAL
}
```

/etc/rc.d/init.d/knockd (3)

```
setup_iptables() {

    $ipt -F $chain_filter
    $ipt -F $chain_trusted
    $ipt -F $chain_drop
    $ipt -F $chain_policy

    $ipt -D INPUT -p tcp -d $srvsship --dport $sshport -m state -state \
            ESTABLISHED,RELATED -j ACCEPT >/dev/null 2>&1
    $ipt -D INPUT -p tcp -d $srvsship --dport $sshport -j $chain_policy >/dev/null 2>&1

    $ipt -X $chain_filter
    $ipt -X $chain_trusted
    $ipt -X $chain_drop
    $ipt -X $chain_policy

    $ipt -N $chain_filter
    $ipt -N $chain_trusted
    $ipt -N $chain_drop
    $ipt -N $chain_policy


...
```

/etc/rc.d/init.d/knockd (4)

```
...


    $ipt -A $chain_policy -j DROP
    $ipt -I $chain_policy -j LOG --log-prefix="[knockd,drop] "
    $ipt -I $chain_policy -j $chain_filter
    $ipt -I $chain_policy -j $chain_trusted
    $ipt -I $chain_policy -j $chain_drop

    $ipt -A INPUT -p tcp -d $srvsship --dport $sshport -m state -state \
            ESTABLISHED,RELATED -j ACCEPT
    $ipt -A INPUT -p tcp -d $srvsship --dport $sshport -j $chain_policy

    if [ -f $k_trusted ]; then

      while read addrip dummy; do
        if [ "$addrip" != "" ] && [ "${addrip:0:1}" != "#" ]; then
           $ipt -A $chain_trusted -s $addrip -p tcp --dport $sshport \
                    -m state --state NEW -j ACCEPT
        fi
      done < $k_trusted

    fi

}
```

/etc/rc.d/init.d/knockd (5)

```
accept() {
    addrip="$1"
    if [ "$addrip" = "" ]; then
      echo "Usage: $0 accept addr-ip"
      return 1
    fi

    $ipt -D $chain_filter -s $addrip -p tcp --dport $sshport -m state --state NEW \
            -j ACCEPT >/dev/null 2>&1
    RETVAL=$?

    if [ "$RETVAL" = "1" ]; then
       (sleep $accept_timeout; $ipt -D $chain_filter -s $addrip -p tcp \
                    --dport $sshport -m state --state NEW -j ACCEPT >/dev/null 2>&1) &
    fi

    $ipt -I $chain_filter -s $addrip -p tcp --dport $sshport -m state --state NEW \
            -j ACCEPT

    RETVAL=$?
    return $RETVAL
}
```

/etc/rc.d/init.d/knockd (6)

```
trust() {
    addrip="$1"
    if [ "$addrip" = "" ]; then
      echo "Usage: $0 trust addr-ip"
      return 1
    fi

    $ipt -D $chain_trusted -s $addrip -p tcp --dport $sshport \
            -m state --state NEW -j ACCEPT >/dev/null 2>&1

    $ipt -I $chain_trusted -s $addrip -p tcp --dport $sshport \
            -m state --state NEW -j ACCEPT

    RETVAL=$?
    return $RETVAL
}
```

/etc/rc.d/init.d/knockd (7)

```
flush() {
    addrip="$1"
    if [ "$addrip" = "" ]; then
      echo "Usage: $0 flush addr-ip|filter|banned"
      return 1
    fi
    if [ "$addrip" = "filter" ]; then
      $ipt -F $chain_filter
      RETVAL=$?
    elif [ "$addrip" = "banned" ]; then
      $ipt -F $chain_drop
      RETVAL=$?
    else
      $ipt -D $chain_filter -s $addrip -p tcp --dport $sshport \
               -m state --state NEW -j ACCEPT
      RETVAL=$?
    fi
    return $RETVAL
}
```

/etc/rc.d/init.d/knockd (8)

```
drop() {
    addrip=$1

    if [ "$addrip" = "" ]; then
      echo "Usage: $0 drop addr-ip"
      return 1
    fi

    $ipt -I $chain_drop -s $addrip -p tcp --dport $sshport \
            -m state --state NEW -j DROP

    RETVAL=$?
    return $RETVAL
}
```

/etc/knockd/knockd.trusted

```
# machines autorisées sans contrôle de
# knocking sur le serveur ssh :

ssh.maths.fr  # i.e. ssh.maths.fr
29.123.4.0/24      #
29.123.21.0/24     #
30.28.121.188      #
```

Relance automatique

- Il arrive que le démon crashe...
- Je n'ai pas identifié pourquoi,
  j'ai été au plus court, un cron surveille le démon et le relance si nécessaire
  - En fait, ce cron, lancé toutes les minutes fait un simple :
    `/etc/rc.d/init.d/knockd restartifdead`

En conclusion

- C'est somme toute très simple
- La mise en oeuvre est facile
- Cas exotiques (clients sur sites filtrants en sortie les ports du knocking) résolus avec la liste blanche.
- La gestion de ceci une fois installé est nulle, sauf ajout/retrait d'adresse dans la « liste blanche »
- Et ça marche !
- Alors, on arrête de dire qu'on ne peut rien faire contre les attaques actuelles (à dictionnaire) contre les serveurs SSH...
- Voir **http://www.portknocking.org/** pour d'autres implémentations.

Des questions ?
Commentaires ?