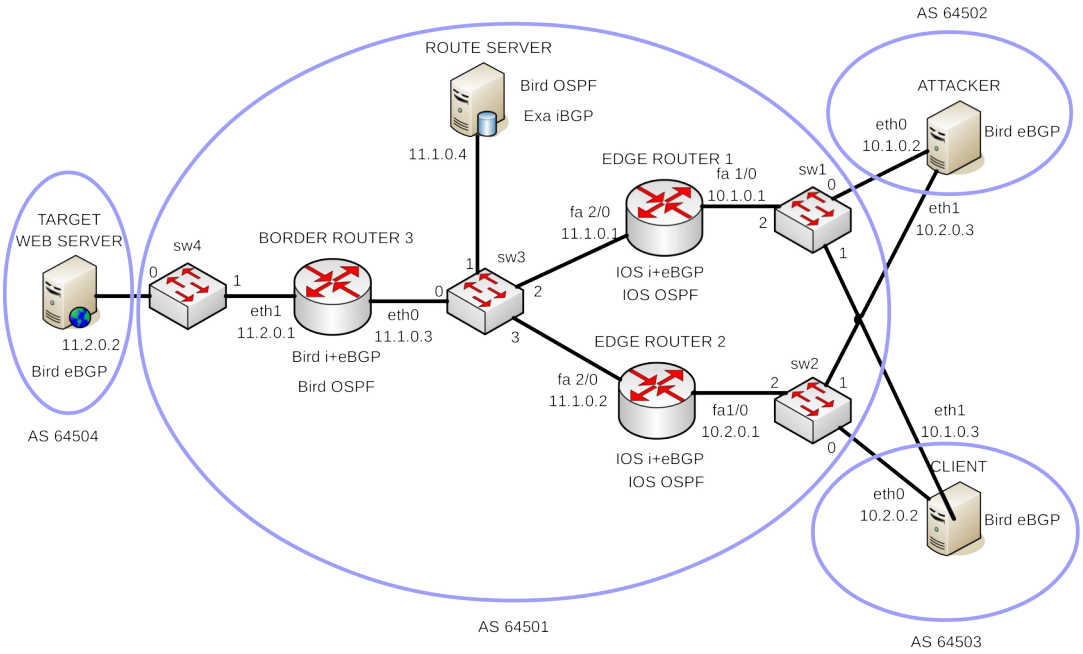
TD Remotely Triggered Black Hole Routing v2 - 20221020

L'objectif de ce TD est d'implémenter un serveur de routes capable de mettre à jour via BGP des serveurs Cisco. Ces mises à jour permettent de bloquer des routes menant à certaines machines (sous attaque par exemple) où bien des routes provenant d'attaquants identifiés. La méthodologie est expliquée dans le document `cisco-blackhole-whitepaper.pdf`.

1) Topologie du réseau



2) Mettre en place les deux routeurs Cisco avec l'émulateur Dynamips.

Déterminer la valeur Idle-PC de dynamips sur votre machine physique grâce à gns3. Copier la valeur trouvée dans le script ci-dessous à la place de `<0x6077b2bc>`.
Placer les commandes suivantes dans un script bash permettant de lancer deux instances ayant chacune deux interfaces :

```
dynamips -P 7200 –idle-pc=0x<6077b2bc> -i 1 -X -T 2001 \
 -p 1:PA-FE-TX -s 1:0:udp:10003:127.0.0.1:10002 \
 -p 2:PA-FE-TX -s 2:0:udp:10007:127.0.0.1:10006 \
 c7200-jk9s-mz.124-13b.image &

dynamips -P 7200 –idle-pc=<0x6077b2bc> -i 2 -X -T 2002 \
 -p 1:PA-FE-TX -s 1:0:udp:10005:127.0.0.1:10004 \
 -p 2:PA-FE-TX -s 2:0:udp:10009:127.0.0.1:10008 \
c7200-jk9s-mz.124-13b.image &
```

La documentation complète de cette commande est disponible dans `dynamips-cli-options.pdf`.

Les tunnels UDP seront reliés à des switches gérés par NEmu (cf. ci-dessous).

Lancez puis configurez les routeurs Cisco comme indiqué dans la figure (sans OSPF) et à l'aide du *white paper*.

Pour cela, dans le routeur EDGE ROUTER 1, configurez les interfaces comme suit :

```
(config)#interface Loopback0
(config-if)#no ip address

(config)#interface Null0
(config-if)#no ip unreachables

(config)#interface FastEthernet0/0
(config-if)#no ip address
(config-if)#shutdown
(config-if)#duplex half

(config)#interface FastEthernet1/0
(config-if)#ip address 10.1.0.1 255.255.0.0
(config-if)#ip verify unicast source reachable-via any
(config-if)#duplex half

(config)#interface FastEthernet2/0
(config-if)#ip address 11.1.0.1 255.255.0.0
(config-if)#duplex half
```

Configurez le protocole BGP dans le routeur EDGE ROUTER 1 comme suit :

```
(config)#router bgp 64501
(config-router)#no synchronization
(config-router)#bgp log-neighbor-changes
(config-router)#timers bgp 20 60
(config-router)#redistribute connected
(config-router)#neighbor 10.1.0.2 remote-as 64502
(config-router)#neighbor 10.1.0.2 next-hop-self
(config-router)#neighbor 10.1.0.3 remote-as 64503
(config-router)#neighbor 10.1.0.3 next-hop-self
(config-router)#neighbor 11.1.0.3 remote-as 64501
(config-router)#neighbor 11.1.0.4 remote-as 64501
(config-router)#neighbor 11.1.0.4 route-map black-hole in
(config-router)#no auto-summary
```

Configurez une route statique qui mène vers nulle part (trou noir) :

```
(config)#ip route 192.0.2.1 255.255.255.255 Null0
```

Configurez les communautés BGP. On définit les communautés suivant la table ci-dessous :

BGP community *old format* 32 bits	BGP community *new format* 16 bits : 16 bits	Membres de la communauté
6553700	100:100	EDGE ROUTER 1
13107400	200:200	EDGE ROUTER 2
19661100	300:300	EDGE ROUTER 1 + EDGE ROUTER 2

```
(config)#ip community-list 1 permit 6553700
(config)#ip community-list 1 permit 19661100
```

```
(config)#ip community-list 2 permit 13107400
(config)#ip as-path access-list 1 permit ^$
```

Configurez les route-map :

```
(config)#route-map black-hole deny 5
(config-route-map)#match community 2

(config)#route-map black-hole permit 10
(config-route-map)#match as-path 1
(config-route-map)#match community 1
(config-route-map)#set community no-advertise
(config-route-map)#set ip next-hop 192.0.2.1

(config)#route-map black-hole permit 15
```

Configurez le EDGE ROUTER 2 de la même façon que le premier routeur ci-dessus en adaptant bien sûr les valeurs utilisées. Les différences concernent les adresses IP, les numéros d'AS et les communautés.

3) Mettre en place les VMs et le réseau virtuel avec NEmu.

Créez un dossier personnel dans le dossier `/espace` de votre machine.
Ne travaillez désormais que dans ce dossier !

Si NEmu n'est pas installé sur les machines, installez le dans le dossier ci-dessus en suivant la procédure décrite ici :
`https://gitlab.com/v-a/nemu/wikis/start/installation`

Récupérez la VM `debian8.img` située dans `/net/stockage/dmagoni`
Copiez la dans le dossier `/espace` de votre machine.
Elle possède un compte `user` (mdp : `user`).
Les autres VMs seront créées par le script NEmu.

Le script NEmu en python à adapter (i.e., remplacer les arguments entourés par < > par vos valeurs) est le suivant :

```
InitNemu(session='<ma-session>', workspace='/espace/<login>', hdcopy=True)

VHostConf('debian', accel="kvm", rtc="base=localtime", k="fr", m="4G")

VHost('attacker', conf='debian', hds=[VFs('debian8.img', 'cow',
tag='attacker.img')],
    nics=[
    VNic(hw='0a:0a:0a:00:01:01'),
    VNic(hw='0a:0a:0a:00:01:02'),
    VNic(hw='0c:0c:0c:00:01:01')])

VHost('border-router', conf='debian', hds=[VFs('debian8.img', 'cow', tag='ce-
bgp.img')],
    nics=[
    VNic(hw='0a:0a:0a:00:02:01'),
    VNic(hw='0a:0a:0a:00:02:02'),
    VNic(hw='0c:0c:0c:00:02:02')])

VHost('route-server', conf='debian', hds=[VFs('debian8.img', 'cow', tag='route-
server.img')],
```

```
    nics=[
    VNic(hw='0a:0a:0a:00:03:01'),
    VNic(hw='0c:0c:0c:00:03:03')])

VHost('target', conf='debian', hds=[VFs('debian8.img', 'cow', tag='web-
server.img')],
    nics=[
    VNic(hw='0a:0a:0a:00:04:01'),
    VNic(hw='0c:0c:0c:00:04:04')])

VHost('client', conf='debian', hds=[VFs('debian8.img', 'cow',
tag='client.img')],
    nics=[
    VNic(hw='0a:0a:0a:00:05:01'),
    VNic(hw='0a:0a:0a:00:05:02'),
    VNic(hw='0c:0c:0c:00:05:05')])

VSwitch('sw1', niface=3)
SetIface("sw1:0", proto='udp', port=11001, lport=11002)
SetIface("sw1:1", proto='udp', port=11003, lport=11004)
SetIface("sw1:2", proto='udp', port=10002, lport=10003)

VSwitch('sw2', niface=3)
SetIface("sw2:0", proto='udp', port=11005, lport=11006)
SetIface("sw2:1", proto='udp', port=11007, lport=11008)
SetIface("sw2:2", proto='udp', port=10004, lport=10005)

VSwitch('sw3', niface=4)
SetIface("sw3:0", proto='udp', port=11009, lport=11010)
SetIface("sw3:1", proto='udp', port=11011, lport=11012)

SetIface("sw3:2", proto='udp', port=10006, lport=10007)
SetIface("sw3:3", proto='udp', port=10008, lport=10009)

VSwitch('sw4', niface=2)
SetIface("sw4:0", proto='udp', port=11013, lport=11014)
SetIface("sw4:1", proto='udp', port=11015, lport=11016)

Link(client='attacker:0', core='sw1:0')
Link(client='attacker:1', core='sw2:1')
Link(client='client:0', core='sw2:0')
Link(client='client:1', core='sw1:1')

Link(client='border-router:0', core='sw3:0')
Link(client='route-server:0', core='sw3:1')
Link(client='target:0', core='sw4:0')
Link(client='border-router:1', core='sw4:1')

VSlirp('slirp1', net='192.168.1.0/24')
Link(client='attacker:2', core='slirp1')

VSlirp('slirp2', net='192.168.2.0/24')
Link(client='border-router:2', core='slirp2')

VSlirp('slirp3', net='192.168.3.0/24')
Link(client='route-server:1', core='slirp3')

VSlirp('slirp4', net='192.168.4.0/24')
Link(client='target:1', core='slirp4')

VSlirp('slirp5', net='192.168.5.0/24')
Link(client='client:2', core='slirp5')
```

Lancez le script avec la commande : `nemu -f bhre.py -i`

Puis dans la console NEmu (un shell python) qui vient de se lancer grâce à la commande ci-dessus, tapez la commande `StartNemu()`
Cela va lancer les 5 VMs.

Vérifiez la connectivité IP entre toutes les paires de voisins.
La connectivité globale n'est pas encore active.

Configurez l'IP forwarding sur la VM *border-router*.

```
sysctl -w net.ipv4.ip_forward=1
```

Et afin qu'il soit permanent modifier la ligne suivante dans `/etc/sysctl.conf` :

```
net.ipv4.ip_forward = 1
```

Puis tapez : `sysctl -p /etc/sysctl.conf`

4) Installez le démon de routage multi-protocole BIRD dans les VMs comme indiqué sur la figure.

Pour cela, obtenez une IP sur la dernière interface eth* :

```
dhclient eth1 (ou eth2 si elle existe)
```

Modifiez le fichier `/etc/apt/source.lists` pour trouver les paquets Debian *Jessie* en remplaçant la ligne `deb` par la ligne ci-dessous :

```
deb http://archive.debian.org/debian/ jessie contrib main non-free
```

Puis :

```
apt update
apt install bird
```

Ne mettez surtout pas à jour tous les paquets !

Configurez le démon de routage BIRD via le fichier `/etc/bird/bird.conf`.
Il contient bcp de commentaires explicatifs.

Filtrer les routes qui mènent à l'internet via les *slirp* :

```
filter fnat {
    if net ~ [ 0.0.0.0/0, 192.168.0.0/16+ ] then
    {
        #print net;
        reject;# "route rejected through fnat";
    }
    else
    {
        #print net;
        accept;# "route accepted through fnat";
    }
}
```

Configurez les modules *direct* et *kernel* :

```
protocol direct {
    interface "eth*";
}

protocol kernel {
    learn off;        # Learn all alien routes from the kernel
    persist off;         # Don't remove routes on bird shutdown
    scan time 10;    # Scan kernel routing table every 20 seconds
    import none;           #filter fnat;   # Default is import all
    export filter fnat;      # Default is export none
}
```

Configurez la route statique vers nulle part :

```
protocol static {
    preference 1000;    # Default preference of routes
    route 192.0.2.1/32 unreachable;
}
```

Ce qui suit se fait dans le *border-router*.

Configurez la connexion iBGP vers le routeur 1 :

```
protocol bgp iBGP1 {
        description "iBGP1";
        local 11.1.0.3 as 64501;
        neighbor 11.1.0.1 as 64501;
        multihop;
        hold time 60;
        startup hold time 60;
        connect retry time 30;
        keepalive time 20;      # defaults to hold time / 3
        start delay time 5;     # How long do we wait before initial connect
        error wait time 60, 300;
    # Minimum and maximum time we wait after an error (when consecutive
    # errors occur, we increase the delay exponentially ...
        error forget time 300;  # ... until this timeout expires)
        export filter fnat;
        import filter fnat;
}
```

Faites de même pour les connexions vers le routeur 2 et le *route-server*.

Ce qui suit se fait dans le *border-router*, *l'attacker*, le *client* et la *target*.

Configurez la connexion eBGP du *border-router* vers la *target* comme suit :

```
protocol bgp eBGP {
    description "eBGP";
    local 11.2.0.1 as 64501;
    neighbor 11.2.0.2 as 64504;
    hold time 60;
    startup hold time 60;
    connect retry time 30;
    keepalive time 20;   # defaults to hold time / 3
    start delay time 5;  # How long do we wait before initial connect
    error wait time 60, 300;
    # Minimum and maximum time we wait after an error (when consecutive
    # errors occur, we increase the delay exponentially ...
    error forget time 300;   # ... until this timeout expires)
```

```
    #  next hop self;
    # Disable next hop processing and always
    # advertise our local address as nexthop
    source address 10.1.0.1; # What local address we use for the TCP connection
    export filter fnat;
    import filter fnat;
}
```

Faites les autres VMs en modifiant le code ci-dessus.
Afin de pouvoir lancer le démon, tapez :

```
mkdir /run/bird; bird;
```

Le client en ligne de commande qui permet de contrôle BIRD est `birdc`.
Les commandes utiles sont `show protocol` et `show route`.

5) Installez la version 3.4.x d'ExaBGP dans la VM *route-server*.
Pour cela, tapez les commandes suivantes :

```
git clone https://github.com/Exa-Networks/exabgp.git
git checkout 3.4
git pull
python2.7 setup.py build
python2.7 setup.py install
```

L'exécutable se trouve dans `/usr/local/bin`.

Configurez ExaBGP comme suit.
Afin d'utiliser l'API RESTful, installez Flask par la commande :

```
apt install python-flask
```

Puis créez le script `/etc/exabgp/app` comme suit en respectant scrupuleusement la syntaxe :

```
from flask import Flask, request
from sys import stdout

app = Flask(__name__)

# Setup a command route to listen for prefix advertisements
@app.route('/', methods=['POST'])
def command():

        command = request.form['command']
        stdout.write('%s\n' % command)
        stdout.flush()

        return '%s\n' % command

if __name__ == '__main__':
    app.run()
```

Configurez le fichier `/etc/exabgp/exabgp.conf` comme suit en plaçant les accolades exactement au même endroit :

```
group my_peers {

  router-id 11.1.0.4;
  local-address 11.1.0.4;
  local-as 64501;
  peer-as 64501;

  static {
    #ERCO CONTROLLED PART
    #END OF ERCO CONTROLLED PART
  }

  process httpapi {
    run "/usr/bin/python /etc/exabgp/app";
  }

#  process socket {
#    run "/var/www/erco/utilities/bin/exabgp-ws-server";
#  }

  neighbor 11.1.0.1 {
    description "Cisco 1 BGP";
  }

  neighbor 11.1.0.2 {
    description "Cisco 2 BGP";
  }

  neighbor 11.1.0.3 {
    description "Bird BGP";
  }
```

Configurez ExaBGP comme service pour *SystemD* en créant le fichier `exabgp.service` suivant :

```
root@debian8:/usr/local/bin# cat /etc/systemd/system/exabgp.se
[Unit]
Description=ExaBGP
Requires=network.target
After=network.target
ConditionPathExists=/etc/exabgp/exabgp.conf

[Service]
Type=simple
User=nobody
Environment=exabgp_daemon_daemonize=false
Environment=ETC=/etc
WorkingDirectory=/tmp
ExecStart=/usr/local/bin/exabgp /etc/exabgp/exabgp.conf
ExecStartPost=/bin/bash -c 'pgrep exabgp > /tmp/exabgp.pid'
ExecStop=/bin/kill -SIGTERM $MAINPID
ExecReload=/bin/kill -SIGUSR1 $MAINPID

[Install]
```

Lancez ExaBGP avec la commande : `systemctl start exabgp`

Vérifiez les échanges BGP avec *Wireshark*.

Attention : n'utilisez pas *Wireshark* dans les VMs pendant plus de qlq (dizaines de) minutes sinon il va saturer la mémoire et freezer la VM.

6) Installez Ostinato dans la VM *attacker* afin de générer du trafic réseau vers la VM *target*.

7) Effectuez un *destination-based blackhole routing* à l'aide d'ExaBGP sur le *route-server* en utilisant l'API RESTful :

```
curl --form "command=announce route 11.2.0.2/32 next-hop 192.0.2.1" \
    http://localhost:5000/
```

Vérifiez que la cible n'est plus joignable.
On suppose que l'attaque est terminée. Réinstallez la route :

```
curl --form "command=withdraw route 11.2.0.2/32 next-hop 192.0.2.1" \
    http://localhost:5000/
```

L'API pour commander ExaBGP est la suivante :

```
    shutdown    (Terminates ExaBGP. > shutdown in progress)
    reload      (Reload ExaBGP's configuration > reload in progress)
    restart     (Reloads ExaBGP from scracth > restart in progress)
    version     (Show exabgp's version)
    show neighbor [optional neighbor ip] summary
    show neighbor [optional neighbor ip] extensive
    show neighbor [optional neighbor ip] configuration
    show ajd-rib in  [extensive] (will be integrated in neighbour)
    show ajd-rib out [extensive] (will be integrated in neighbour)
    announce watchdog
    announce route (will be deprecated when announce <afi> <safi> is introduced)
    announce eor
    announce flow
    announce operational
    announce vpls
    announce route-refresh
    teardown
```

8) On suppose que vous avez identifié l'IP de l'attaquant. Effectuez un *source-based blackhole routing* à l'aide d'ExaBGP sur le *route-server* en utilisant l'API RESTful :

```
curl --form "command=announce route 10.1.0.2/32 next-hop 192.0.2.1" \
    http://localhost:5000/
```

Vérifiez que la cible n'est plus joignable.
On suppose qu l'attaque est terminée. Réinstallez la route :

```
curl --form "command=withdraw route 10.1.0.2/32 next-hop 192.0.2.1" \
    http://localhost:5000/
```

9) On suppose que l'on souhaite bloquer l'attaquant uniquement sur le routeur 1.
Placez le routeur 1 dans la community 100 et le routeur 2 dans la community 200 (cf. point 2)).
Effectuez un *community-based blackhole routing* à l'aide d'ExaBGP sur le *route-server* en utilisant l'API RESTful en invoquant la communauté à laquelle appartient le routeur 1 :

```
curl --form "command=announce route 10.1.0.2/32 next-hop 192.0.2.1 \
    community 100" http://localhost:5000/
```

Vérifiez que l'attaquant ne peut plus attaquer par le routeur 1 mais le peut par le routeur 2.

On suppose que l'attaque est terminée. Réinstallez la route :

```
curl --form "command=withdraw route 10.1.0.2/32 next-hop 192.0.2.1 \
    community 100" http://localhost:5000/
```

10) Installez ERCO sur le *route-server* pour gérer les commandes BGP via une interface Web, plutôt que de taper les commandes à la main comme ci-dessus.

ERCO est récupérable ici : https://erco.xyz/
Lisez bien la procédure d'installation ainsi que la documentation.
Dans le fichier `exabgp.conf`, décommentez le bloc `process socket.`