Université de Bordeaux — année 2020-2021

Master CSI/Master Info. — Sécurité des réseaux

TD - TP noté (1h30)

— Les réponse doivent être saisies sur l'activité rendre le TP (https://moodle1.u-bordeaux.fr/mod/vpl/view.php?id=299125) du moodle de l'UE de sécurité des réseaux.

— Lancer le script de démarrage `/net/stockage/aguermou/SR/devoir/1/qemunet.sh` en lui fournissant la description de la topologie réseau à l'aide de l'option `-t` ainsi que l'archive contenant la configuration initiale des machines à l'aide de l'option `-a`. Ceci revient à lancer les commandes suivantes :

```
user@machine# cd /net/stockage/aguermou/SR/devoir/1/; ./qemunet.sh -d tmux -b \
                  -t topology -a archive_devoir.tgz;
user@machine# tmux a
```

Le contexte

Le *port-knocking* est une méthode permettant de modifier le comportement d'un firewall en temps réel en provoquant l'ouverture de ports permettant la communication, grâce au lancement préalable d'une suite de connexions sur des ports distincts dans le bon ordre, à l'instar d'un code frappé à une porte.

Le *port-knocking* consiste à vérifier le caractère sain du client avant toute action de sa part (le client étant une machine qui tente d'ouvrir une connexion vers notre serveur). La question est donc : Comment vérifier qu'on peut faire confiance au client ? Le *port-knocking* y répond en disant qu'il faut que le client entre "le bon mot de passe" ou "la bonne clé". Cette clé n'est censée être connue que par le serveur et les clients susceptibles de s'y connecter. Ainsi, si le client précise la bonne clé, le serveur acceptera de le recevoir (ce qu'on entend par "recevoir" c'est le laisser ouvrir la connexion). Dans le cas contraire, il ne réagira pas. C'est justement toute l'originalité de cette solution : tant qu'on ne lui communique pas la bonne clé, le serveur fait "le mort". Si cette technique est appliquée à tous les services tournant sur notre serveur, vu de l'extérieur, notre serveur ressemblera de plus en plus à un client tout ce qu'il y a de plus quelconque.

Prenons un petit exemple pour en expliquer le fonctionnement. Initialement tous les ports du serveur sont fermés par un firewall. Il est possible d'accéder) l'un d'entre eux (par exemple le port ssh) grâce à une clé, qui n'est autre qu'une série de ports. Pour ouvrir une connexion sur le port ssh, il est donc nécessaire d'envoyer des paquets particuliers (requête de connexion ou autre chose) aux ports constituant la clé (dans le bon ordre). Ces paquets ne sont pas traités, il servent juste à authentifier le client vis-à-vis du serveur. Si l'authentification réussit, le serveur reconfigure le firewall pour autoriser ce client en particulier ouvrir une connexion sur le port 22. Cette autorisation est souvent temporaire et expire au bout d'un certain temps. Il est donc nécessaire dans ce cas de refaire l'étape de *port-knocking*. En pratique, sur le serveur, un processus s'exécute en arrière-plan (daemon) et scrute le journal des connexions du pare-feu, où une analyse de paquets passant par un port déjà ouvert, permet de détecter une séquence particulière.

En résumé, le port knocking est donc une méthode simple pour autoriser un accès distant sur un port non constamment ouvert. Cela permet d'éviter les scans de ports et certaines attaques.

Le but du challenge est de prendre la main depuis atg sur la machine windows pour récupérer le contenu d'un fichier dont le nom est `C:\tresor.txt`. Pour ce faire, vous allez devoir :

— d'une part trouver les identifiants (login/mot de passe) qui vous permettront d'ouvrir une session sur immortal à partir d'atg. Pour vous "simplifier" la vie, grave se connecte périodiquement à immortal en utilisant le protocole `ssh` à l'aide des identifiants recherchés (le dialogue est donc chiffré).

— enfin, syl empêche tout trafic en provenance du réseau de atg/nile vers immortal. Pour autant, elle accepte d'autoriser la communication vers un port demandé à condition que la bonne séquence de *port-knocking* soit utilisée. La machine nile effectue cette opération périodiquement : Elle émet la séquence de port knocking avant de tenter d'ouvrir une connexion TCP sur le port 22 immortal. Vous trouverez des détails sur le port knocking en annexe.

Challenge

Dans ce qui suit, il est **interdit** de reconfigurer les interfaces réseaux des machines ou de toucher au routage.

1. Expliquez en quoi le principe du *port-knocking* présenté en annexe peut améliorer la sécurité. Est-il possible avec une technique de brute-force de découvrir la séquence de *port-knocking* permettant de débloquer un port.
2. Déterminez les adresses IP de toutes les interfaces réseau de toutes les machines (dans la mesure du possible :-)). L'adresse IP de la machine windows ne peut être trouvée que plus tard.

   Indications : Dans le cas où vous souhaiteriez écouter le trafic réseau en mode texte (`tmux`), vous pouvez utiliser la commande `tshark` (un équivalent à `wireshark` en mode texte) :
   `tshark -i <interface> -n` ou alors `tshark -i <interface> -n > fichier` (fichier contiendra alors le résultat de la commande)
3. Trouvez la séquence de *port-knocking* utilisée par nile pour demander l'accès du port ssh d'immortal auprès de syl (cette séquence est composée de 3 paquets TCP).

   Indications : Lorsque vous aurez trouvé la séquence de port-knocking, vous pourrez l'utiliser à l'aide de la commande `knock` :
   `knock @ip_syl port1:tcp port2:tcp port3:tcp ...` (ou proto peut valoir tcp ou udp)
4. Trouvez un moyen de récupérer les identifiants nécessaires à l'ouverture d'une connexion à distance en utilisant ssh depuis atg vers immortal. Il s'agirat de détourner le communication entre grave et immortal pour qu'elle passe par atg puis d'utiliser l'outil `ssh-mitm` pour récupérer les identifiants (voir ci-dessous)

   Indications :
   — Dans le cas ou sur une machine vous avez besoin de répondre à un trafic que vous recevez mais qui ne vous est pas destiné, pensez à utiliser l'action `REDIRECT` de la table nat d'iptables. Un exemple d'utilisation de ce type de règle est fourni ici : https://github.com/moxie0/sslsniff.
   — syl et darkane font office de serveur dns pour grave.
   — Un serveur ssh modifié permettant l'écoute active d'une connexion ssh est installé sur atg. Il doit être lancé **en tant que** l'utilisateur ssh-mitm de la manière suivante :
   ```
   su - ssh-mitm; /home/ssh-mitm/sbin/sshd -f \
            /home/ssh-mitm/etc/sshd_config
   ```
   Ce serveur, faisant office de proxy ssh, sera en écoute sur le port 2222 et écrira le contenu en clair du trafic intercepté dans un fichier dont le nom est de la forme `shell_session_*.txt` se trouvant dans le home de l'utilisateur `ssh-mitm`.
5. Il s'agit maintenant de récupérer depuis la machine windows le contenu du fichier dont le nom est `tresor.txt`. Il faut donc trouver un moyen d'ouvrir une session à distance sur la machine windows depuis atg pour lire le contenu du fichier. La machine immortal n'ayant pas de framework d'attaque installé, il vous faudra lancer votre offensive depuis atg en utilisant immortal comme pivot. Ceci peut être fait en établissant un tunnel `ssh` avec choix dynamique de port et d'utiliser un outil de tunneling avancé tel que `proxychains`.

Indications :

— L'adresse IP de la machine windows est 10.2.32.43.

— Si vous souhaitez faire un scan de port via le tunnel `ssh`, il est nécessaire d'utiliser le scan *TCP Connect* de `nmap` :
`nmap -sT ...`

— Dans le cas où il serait nécessaire de redémarrer la machine windows, il suffit de :

(a) Cliquer sur l'onglet correspondant à la machine. Vous avez alors accès au moniteur `qemu` de la machine.

(b) Lancer la commande `system_reset` depuis le moniteur.

— Pour éteindre la machine windows, il suffit de taper la commande `quit` dans le moniteur.

— Dans le cas d'une utilisation de `metasploit` via un tunnel, seuls les `payload` de type `bind` fonctionnent.