Corentin Roberge-Mentec
Mathieu Tabary
Paul-Émile Watelot

Université de Lorraine
Faculté des Sciences et Technologies
M1 Informatique

M1 - Projet réseau 2020

Étude de websnarf

1) Étudier et expliquer les fonctions, le rôle, l'utilité et le principe de fonctionnement de websnarf.

Websnarf est un script perl permettant de simuler un serveur web mais ne réalisant rien à par log toutes les tentatives de connexions et accès sur son port.
Il servait à comprendre les attaques menées par les Web Worms sur Internet qui visent à attaquer des serveurs/services web.
Pour fonctionner Websnarf écoute le port 80 sur toutes les interfaces du système où il est exécuté.
Websnarf a été utile pour comprendre quelles requêtes ont été effectuées pour mettre à mal l'Internet. Son principe de fonctionnement étant basique et peu complexe est la raison pour laquelle il a été copié et implanté dans la majorité des systèmes de serveurs web.

2) Analyser, expliquer et commenter le code Perl ainsi que les paramètres de la ligne de commande

le paramètre --port=## sert à définir le port d'écoute manuellement sinon c'est 80 par défaut.

le paramètre --timeout=## sert à définir le nombre de secondes après lequel websnarf ferme la connexion.

le paramètre --log=FILE permet d'enregistrer sur le disque (dans le fichier FILE) les logs.

le paramètre --max=## permet de définir la longueur max de la requête capturée.

le paramètre --save=DIR sauvegarde avec un fichier par requête dans le dossier DIR.

le paramètre --apache met les logs dans le format Apache.

le paramètre --version affiche la version.

Lignes :
1 - 25 : Explication du fonctionnement du script.
27 - 54 : Explication des options des options.
56 - 75 : Déclaration des variables.
77 - 135 : Analyse des options choisies.
143 - 153 : Création du fichier logfile quand l'option --log=logfile est choisie.
160 - 175 : Création du socket avec les paramètres définis ou non avant.
176 - 313 : Boucle d'écoute du programme dans lequel le programme attend une requête entrante et affiche dans la console les dites requêtes.
322 - 331 : met en forme le timestamp pour le format apache pour les logs
333 - 343 : met en forme le timestamp pour les log.

3) Expliquer, détailler et résoudre les « Open Issues » et « Security Issues »

- No IP-to-name lookup

Explication : On a accès à l'adresse IP du client, mais pas au nom du domaine de celui-ci.
Détail : On peut faire gagner du temps à l'utilisateur en faisant dans le programme la requête DNS pour l'adresse du client. Et ainsi comprendre plus rapidement de quelles machines viennent les requêtes.
Résolution : Résolu. On récupère le nom du domaine grâce à la fonction gethostbyaddr() de la librairie netdb.h.

- Questionable timeout processing on reads

Explication : La fin de connection peut poser des problèmes si le client se déconnecte sans envoyer de signal de fin.
Détail : Si la connection TCP n'est pas bien terminée, le script Perl pouvait rester bloquer sur une instruction de lecture de socket. Il est aussi possible d'avoir ce problème en C.
Résolution : Résolu. Pour éviter de rester bloquer sur une lecture de socket quand le client ne parle pas où coupe la connection sans la fermer, on ajoute à la socket les options SO_RCVTIMEO et SO_SNDTIMEO, qui permet d'interrompre l'instruction de lecture de socket au bout d'un certain temps.

- No real packet capture

Explication : Le script Perl ne capture pas le packet TCP directement (comme pourrait le faire Wireshark), mais les données remontées en couche application.
Détail : Récupérer tout le packet TCP augmenterais le nombre d'information à propos du WebWorm.
Résolution : Non résolu.

- Serial connection management

Explication : Le script Perl traite les connections les unes après les autres.
Détail : Lorsque plusieurs clients tentent de se connecter au serveur en même temps, le serveur doit les traiter les clients l'un après l'autre.
Résolution : Résolu. Nous avons réussis à faire un traitement des connections en parallèle avec la création de processus fils pour chaque connection TCP.

- It's not any kind of daemon

Explication : Le programme doit être lancé manuellement et le terminal qui a lancé le programme doit rester ouvert.
Détail : En executant ce programme au lancement de l'ordinateur, il pourrait écouter et accepter bien plus de connexions, et ainsi avoir plus d'opportunitées de voir des messages de web worms.
Le fait de tourner en tant que daemon est aussi une forme de confort, en effet, plus besoin de laisser tourner un terminal en avant-plan, le service est détaché du terminal et tourne en arrière-plan.

Résolution : Résolu. Pour faire tourner le programme en tant que daemon, nous avons ajouter l'option --daemon qui fera executer la commande daemon(0, 1) de la librairie unistd.h. Le premier paramètre est là car la sortie standard ne nous interesse plus, et si l'utilisateur veux garder une trace des requêtes effectuées ilpeut utiliser l'option --log. Le second paramètre indique que le repertoire d'exeuction ne doit pas être changé (pour maintenir l'écriture dans le fichier de log).

- We don't distinguish between the various kinds of Code Red worms out there

Explication : Les vers Code Red ont des comportements différents, pouvoir les différenciers avec cet outil pourrait être utile à distinguer quel vers infecte quels types de machines.
Détail :
Résolution : Non résolu. Nous n'avons pas trouvé les messages envoyés par les différentes versions de Code Red, et donc nous n'avons pas pu distinguer les différences entre les différent vers.

- We probably should offer an IIS logging format

Explication : Le format IIS est le format des serveurs victimes d'attaques de Code Red. C'est un serveur Web (HTTP) des différents systèmes d'exploitation Windows NT
Détail : Le fait d'avoir un fichier de log dans un format différent permet de faciliter le traitement des logs.
Résolution : Résolu. Nous avons créé un logfile de type IIS.

4) Autres idées de fonctionnalitées

1. Websnarf multiport

Explication : Jusqu'a maintenant, websnarf ne pouvait être lancé que pour écouter sur un seul port. Dorénavant, il sera possible de faire une écoute sur plusieurs ports en utilisant l'option --multiport=port_1,...port_n.
Détail : Quand cette option est selectionnée un processus est lancé pour chaque port indiqué (en donnant l'option --port=port_i à la place de --multiport), son nom dans le systeme est websnarf_port et les options sont transmises a chaque fils. Cette option est donc compatible avec toutes les autres options. Attention, en lançant plusieurs port en daemon, il faudra tuer chaque processus mannuelement car leur processus père ne sera plus là pour les arrêter.

2. Log pour chaque port

Explication : Quand l'option --log=LOGFILE est utilisé, il est possible d'utiliser l'option --logbyport pour que les logs soient enregistré dans le fichier LOGFILE_port.
Détail : Cette option est très utile si on souhaite séparer les logs des différents ports de l'option --multiport. Il est possible que le log se fasse plus lentement que l'affichage, il est donc recommandé de ne pas tuer un processus avant d'être sûr qu'il ai enregistré tous ses logs.