S'amuser en Fuzzant
Tutoriel sur l'utilisation et la personnalisation d'un Fuzzer

D'après les tutoriaux de http://dyngnosis.com

Adaptation française : Jérôme ATHIAS
Dernière mise à jour : 28/06/2005

Nous allons commencer par une petite définition du fuzzing puis passer directement à l'aspect technique.

--
Le fuzzing est une méthode de test de logiciels. L'idée principale est de lier les entrées d'un programme à une source de données aléatoire.
Si le programme échoue (par exemple, en plantant), alors il est défectueux.

Le grand avantage du test par fuzzing est que le principe du test est extrêmement simple, et libre de toute opinion préconçue par rapport au comportement du système.
--

Qu'est-ce que cela signifie?
Basiquement, vous utilisez un script pour automatiser l'envoi de mauvaises commandes à un serveur et voyez si il plante.
Dans mon cas, je vais fuzzer un serveur FTP. J'ai choisi Bullet Proof FTP au hasard, car c'est simplement un exercice pour améliorer mon niveau de compétences.

Le script, ou "fuzzer" que je vais utiliser dans ce cas est "fuzzer-1.1" de Shadown [at] gmail.com. Vous pouvez le trouver sur packetstormsecurity.com (http://packetstormsecurity.nl/UNIX/misc/fuzzer-1.1.tar.gz).

J'ai choisi ce fuzzer pour plusieurs raisons; mes amis l'utilisent, il est écrit en python, et j'apprécie son interface "simple".

Mon environnement de test comprend une machine sous Linux avec python, ethereal,... installés et une machine Windows faisant tourner mon programme cible.
Après quelques configurations, je suis en mesure de me connecter au FTP (Windows) depuis ma machine Linux.
Je lance ensuite la commande suivante :

./fuzzer.py -h 192.168.1.199 -p 21 -t ftp
(Ajoutez un >NOMDEFICHIER à la fin pour écrire les résultats dans un fichier)

La commande ci-dessus lance le fuzzer contre ma machine cible sur le port 21 avec le protocole 'ftp' sélectionné.
Le fuzzer supporte peu de protocoles comme FTP (SMTP et POP3).

Il connait certaines des commandes qu'un serveur FTP utilise et tente d'ajouter des données à celles-ci... par exemple, pour s'authentifié à un serveur FTP, les commandes USER et PASS sont employées.
La tentative de connexion débute, la commande USER est envoyée, suivie d'une flopée de "AAA...AAA".
Puis le nombre de "AAA" est doublé, puis triplé, et ainsi de suite... jusqu'à ce que le "maximum" soit atteint. Le maximum est défini dans le script -- nous aborderons ceci plus tard en le modifiant pour améliorer les performances.
Après avoir tenté diverses manipulations sur la commande USER, il repasse en mode "par défaut" (également défini dans le script) et passe à la commande PASS.
Il va boucler avec toutes les commandes définis dans le script.

Après une bonne bière fraiche et un petit-encas, je vois que le fuzzer à terminé et que mon service ftp tourne toujours... il est temps de se pencher sur le code mes amis...

Je décide d'entreprendre quelques recherches sur BulletProof FTP et trouve leur manuel en ligne http://www.bpftpserver.com/help/bpftpserver.com/
Nous le parcourons jusqu'à la section "Advanced" et observons la section 4.1 "Commandes FTP"

· ABOR utilisée lorsque l'utilisateur abandonne sa commande actuelle
· ALLO param représente le nombre d'octets qu'un client souhaite envoyer
. ALLO vérifie qu'assez d'espace disque est disponible
· APPE param est le nom du fichier dont le transfert montant doit être repris
· CDUP / XCUP / PWD / XPWD / CWD navigation dans les répertoires
· HELP affiche les commandes supportées
· LIST / NLST liste les fichiers
· MDTM yyyymmddhhmmss[+-xxx] définie la nouvelle date/heure
· MKD / XMKD créée un répertoire
· NOOP demande au serveur de ne pas clore la session de l'utilisateur...
· PASV active le mode passif
· PORT définie l'adresse et le port de connexion
· PWD renvoie le répertoire courant
· QUIT termine la connexion
· REIN réinitialise l'utilisateur en cours sans être déconnecté...
· REST définie l'offset de départ pour l'envoi de fichier
· RETR / STOR / DELE / APPE RETRieve / STOre / DELEte / APPEnd fichiers...
· RMD efface le répertoire
· RNFR / RNTO ReName FRom, ReNameTO utilisées pour renommer un répertoire, fichier
· SITE CHAT envoie un message à un autre utilisateur connecté
· SITE PSWD permet à un utilisateur de changer son mot de passe
· SITE WHO affiche la liste des personnes connectées
· SITE ZONE affiche la zone horaire du serveur relative à UTC
· SIZE renvoie la taille d'un objet spécifié
· STAT renvoie des informations sur le serveur et l'utilisateur en cours
· STOR param représente le fichier qui doit être envoyé au serveur
· STOU semblable à stor mais génère un unique ...
· SYST renvoie le type du système
· TYPE défini le mode de transfert ASCII ou Binaire
· USER / PASS nom d'utilisateur et mot de passe pour l'authentification au serveur

C'est une mine d'or! Tout ce que nous avons alors à faire c'est d'étendre les commandes que le fuzzer vérifie. Il y'a une tonne de bonne choses ici qui accepte des paramètres utilisateur!

Nous allons utiliser la commande SITE CHAT comme exemple. Nous allons l'ajouter au fuzzer pour qu'il la test.

La première chose à faire est de modifier le fichier protocols.py - ouvrez le avec un bel éditeur de texte et observez son contenu.
Le protocole FTP est le premier dans la liste. Vous devez tout de suite porter votre attention sur la section "commandes".

est utilisé pour commenter une ligne en Python. Nous pouvons mettre en commentaires les commandes que nous avons déjà fuzzées - cela nous économisera du temps et du café.

Vous pouvez copier/coller une ligne de commande et la modifier comme ceci:

'command ' : 'SITE CHAT " : 'string' , 'default' : None , 'recv' : 1 , 'endw' : 'rn' ,'mustuse' : 0 },

Et faites de même avec les autres nouvelles commandes trouvées dans le Manuel.

Vous pouvez également regarder du coté des RFC...

