Éventail des différents outils de Fuzzing

par Pierre Therrode (Espace de Pierre Therrode)

Date de publication : 2009-01-18

Dernière mise à jour : 2009-01-31

Introduction au fuzzing tiré du magazine de Septembre 2008 de http://www.hakin9.org

I - Qu'est-ce le Fuzzing ?

Le fuzzing (dérivé du mot anglais fuzzy qui signifie flou) est une méthode permettant de traquer facilement des vulnérabilités dans des systèmes ou programmes. La technique est très simple : tout d'abord, il faut générer des données aléatoires à peu près conformes au protocole. Par exemple, ces données peuvent être de longues entrées comme 3425 x z, ou des indications de format tel que %a%a%a%, ou encore des valeurs d'entiers : 0,1,2, etc.., ou tout simplement des caractères Unicode (de U+0000 à U+0FFF).

Cependant, ces données ne sont pas totalement aléatoires, car elles ressemblent aux spécifications du protocole. En effet, elles doivent obéir à la logique de programmation du système ciblé, notamment sur les limites dans les opérations de parsing.

Si le fuzzer n'envoie que des données totalement aléatoires au système ciblé, celui-ci les rejettera dès qu'elles ne correspondront plus à ce qui est attendu.

Il faudra donc envoyer des données dites semi-valides pour que le parser ne les refuse pas tout de suite, mais qui pourront cependant causer quelques problèmes.

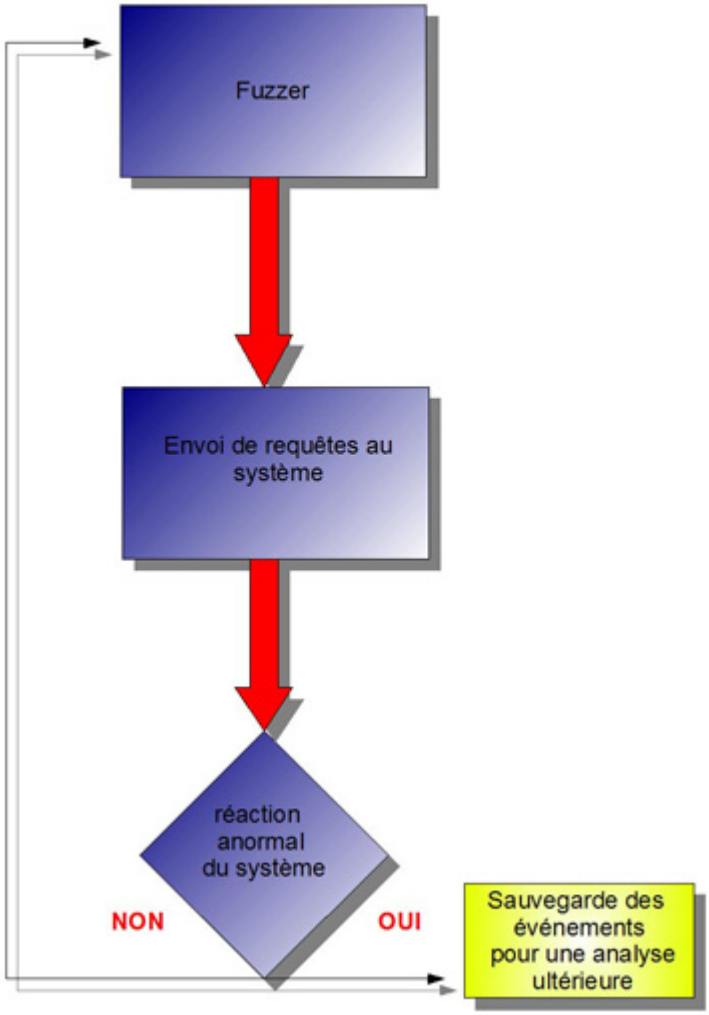
Il n'est pas rare que le système de parsing soit le maillon faible d'un système ciblé (ex: navigateur qui plante suite à une page web contenant des données aléatoires et complexes).

Dans certains cas, il se peut que le système ait soudainement un besoin de ressource en mémoire, ou que la charge du processeur tende violemment vers les 100 %

Ces réactions atypiques montrent que l'envoi de données semi-valides permet d'influencer le fonctionnement.

Dans chaque cas, les résultats sont sauvegardés pour une analyse ultérieure, dans d'autres cas, la même démarche peut être effectuée à nouveau, mais de façons différentes si le système n'a pas réagi.

Généralement, grâce à l'utilisation de cette technique, des erreurs dans la gestion de la mémoire ou dans d'autres parties du système ciblé sont découvertes.



Principe du fuzzing

Ces vulnérabilités peuvent alors être utilisées pour obtenir le total contrôle du système ainsi testé, si biensur cela est utilisé lors de testes de sécurité.

Évidemment, les faiblesses découvertes par le fuzzer ne sont pas toutes des vulnérabilités critiques, loin s'en faut, mais elles peuvent toutefois être utilisées par un individu malveillant.

Dans la suite de l'article, nous aborderons les différents types de fuzzers.

II - Les différents fuzzers

Il existe deux grandes catégories de Fuzzers.

Le premier type de fuzzer est composé de Framework qui reposent sur un API dans le but de créer des entrées et implanter des protocoles spécifiques.

Cette rubrique est composée de différents fuzzers.

II-A - Les Fuzzers de Frameworks

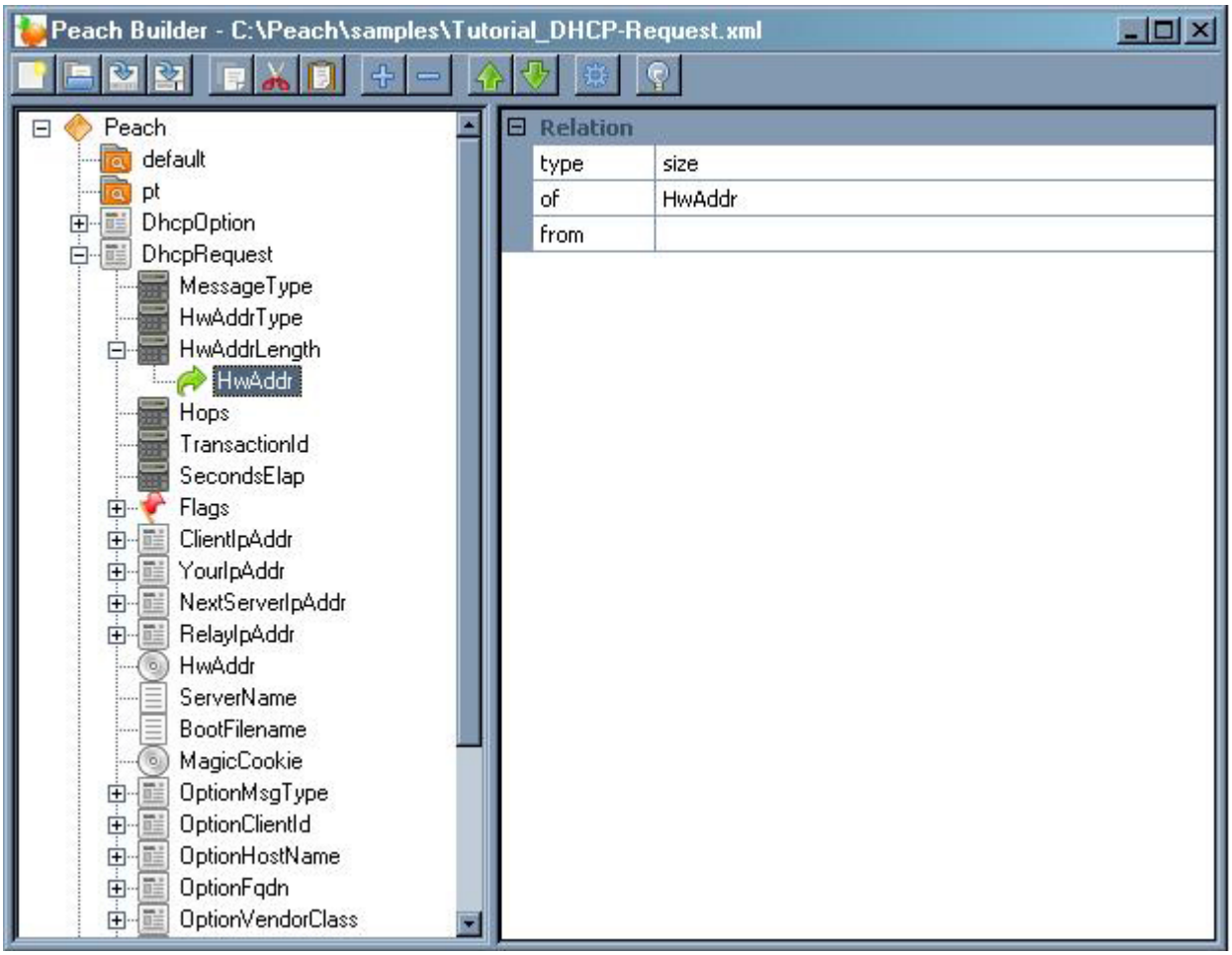
II-A-1 - ANTIPARSER

ANTIPARSER est un fuzzer à toute épreuve, il permet l'envoi de données mal formées au niveau de l' API. Le but d'Antiparser est de fournir une API utilisable sur des modèles de protocoles de réseaux et sur différents formats de fichier. Une fois qu'un modèle a été créé, Antiparser utilise diverses méthodes pour créer des données aléatoires de façon à déclencher des bugs dans le logiciel ou le système ciblé. Le fuzzer inclut dans son fichier de téléchargement un tutoriel et un manuel d'utilisation très détaillé. Enfin, Antiparser fonctionne seulement avec la version supérieure de python 2.3.

Site officiel : **http://antiparser.sourceforge.net**

II-A-2 - PEACH

PEACH est un fuzzing écrit lui aussi en python. Le fuzzer a une facilité d'emploi remarquable et permet de gagner du temps de développement. Peach fuzze à peu près tout comme .NET, COM/ActiveX, le SQL, des dll, des applications de réseaux, ainsi que certaines structures web. À noter aussi que le site officiel de peach inclut un tutoriel d'utilisation bien détaillé.



Site officiel : **http://peachfuzz.sourceforge.net**

II-A-3 - SPIKE

SPIKE permet d'écrire facilement des API de protocoles et son utilisation est très simple. Le fuzzer étudie par ingénierie inverse les protocoles de réseaux. Il dispose de plusieurs exemples permettant de prendre facilement en main son utilisation. Il inclut une technique de brute force pour le serveur Web NTLM.

Site officiel : **http://www.immunitysec.com/resources-freesoftware.shtml**

II-A-4 - SCRATCH

SCRATCH est un fuzzer de protocole écrit en python, qui permet dans les trois quarts du temps de trouver une grande variété de vulnérabilités à partir d'un simple paquet. Scratch fait une analyse heuristique des syntaxes de fichiers binaires pour déterminer les bugs exploitables. Scratch a une structure qui lui permet de fuzzer les protocoles binaires comme SSL et SMB.

Lien : **http://packetstormsecurity.org/UNIX/misc/scratch.rar**

II-B - Les Fuzzers pour le Web

Après les quatre principaux fuzzers d'API et de Frameworks, voyons maintenant la deuxième catégorie de fuzzers, celle des logiciels et des protocoles.

II-B-1 - WAPITI

WAPITI est un fuzzer pour les applications Web écrites en python. Il utilise une technique qui lui est propre car il scanne toute l'application ciblée et il identifie toutes les saisies d'un utilisateur. Après le scan, il exécute alors une série d'essais sur chaque variable, comme l'injection de caractères spéciaux (et de ponctuations) cherchant ainsi des réactions atypiques de l'application. Wapiti s'utilise pour automatiser la découverte de faille de type SQL, des attaques d'injection de code, ainsi que les XSS.

Site officiel : **http://wapiti.sourceforge.net**

II-B-2 - WEBFUZZER

WEBFUZZER est une application écrite en langage C par le groupe gunzip, le fuzzer se spécialise pour scanner les sites web, il teste à distance différentes vulnérabilités comme l'injection sql, cross site scripting, les failles php includes, l'exécution de code à distance, etc...

```
//extrait du code-source de web fuzzer
{
# define FMTSTDOUT GREEN KYA "(" DEF "%s" GREEN KYA ")" \
DEF GREEN ": %s " KYA "(" DEF RED "http://%s%s%c%s" GREEN KYA ")" DEF "\n"
# define FMTTXT "(%s): %s (http://%s%s%c%s)\n"
# define FMTHTML "<p class=\"report\"> <span class=\"proto\"> (%s) </span>\n" \
# "<span class=\"vuln\"> %s </span>\n" \ "<span class=\"url\"> http://%s%s%c%s
# </span>\n</p>\n\n"
# define MESSAGE (proto == POST) ? "POST" : "GET", \ message, T->host, ((proto ==
# POST) || action ) ? action : "", \ qm, url
/**
** format is a string like
** <GET|POST> <possible vuln> <http://host/page?query=bla> [error]
**/
char qm = ((proto == POST) ? '?' : '/');
fprintf( stdout, FMTSTDOUT, MESSAGE );
if ( T->opt->logtype == HTML )
freport( T->opt->logfile, FMTHTML, MESSAGE );
else
freport( T->opt->logfile, FMTTXT, MESSAGE );
if ( error ) {
if ( T->opt->logtype == HTML )
freport( T->opt->logfile, "<div class=\"error\">\n%s\n</div>\n", error );
else
freport( T->opt->logfile, "--[ %s ]--\n", error );
fprintf( stdout, KYA RED "--[ %s ]--\n" DEF, error );
}
}
```

Site officiel : **http://gunzip.altervista.org**.

II-C - Les Fuzzers pour le navigateur

Intéressons-nous maintenant aux différents fuzzers pour le navigateur. Il existe beaucoup d'applications disponibles sur le net, mais je n'en citerai que deux : AXMAN et HAMACHI.

II-C-1 - AXMAN

AXMAN est un fuzzer développé par le chercheur américain HD Moore, spécialisé dans le fuzzing d'ActiveX. Il permet de déterminer tous les objets COM proposés par Internet Explorer. Depuis le lancement de AxMan, de nombreuses failles ont été découvertes (et rendues publiques), notamment au niveau des navigateurs web, comme Internet Explorer, Firefox, Safari, Konqueror et Opera.

L'auteur a notamment dédié un blog à ces recherches **http://browserfun.blogspot.com**.

A noter aussi que le site officiel propose une démonstration en ligne : **http://www.metasploit.com/users/hdm/tools/axman**.

II-C-2 - HAMACHI

HAMACHI est un projet issu d'une communauté de développeurs, avec à la tête H D Moore et Aviv Raff, et ayant pour but de vérifier l'intégrité des navigateurs.

Hamachi est compilé en javascript et permet la recherche des DHTML (Dynamic HyperText Markup Language).

Le fuzzer bombarde le navigateur ciblé avec un maximum de valeurs dans le but de provoquer une réaction non prévue par les développeurs, amenant à la découverte d'une ou plusieurs vulnérabilités.

Site officiel : **http://metasploit.com/users/hdm/tools/hamachi/hamachi.html**

II-D - Les Fuzzers de protocoles et de services

II-D-1 - PROTOS

PROTOS, le projet Protos recherche différentes approches de tests au niveau du protocole en utilisant diverses techniques de type black-box. L'objectif est de soutenir activement l'élimination des failles de sécurité. Il existe d'autres fuzzers issus de ce projet, comme Protos HTTP-reply, Protos WAP, Protos SIP, etc...

L'adresse du site officiel est **http://www.ee.oulu.fi/research/ouspg/protos/**

II-D-2 - SNMPFUZZER

SNMPFUZZER est un dérivé du développement du fuzzer Protos, il contient un nouveau moteur de générateur de bugs écrit en Perl. Il fournit des méthodes efficaces pour déterminer le test à l'origine d'une réaction inattendue du système ciblé, il offre aussi aux utilisateurs une interface plus conviviale.

Lien : **http://packetstormsecurity.org/UNIX/misc/snmp-fuzzer-0.1.1.tar.bz2**

II-D-3 - FTPFUZZ

FTPFUZZ, est un fuzzer d'interface très simple pour évaluer les mises en oeuvre de serveur FTPD. Il permet à l'utilisateur de spécifier des commandes FTP par le biais des paramètres du fuzzer et de définir le cheminement à suivre pour chaque type de cas auquel l'utilisateur est confronté.

Son utilisation a mis à jour de nombreuses vulnérabilités dans de nombreux services FTP, notamment des vulnérabilités permettant l'exploitation à distance.

Lien : **http://www.infigo.hr/files/ftpfuzz.zip**

II-D-4 - Bluetooth Stack Smasher

BSS, ou Bluetooth Stack Smasher, est un fuzzer écrit et développé en langage C par Pierre Betouin dans le but d'effectuer différents tests sur les périphériques Bluetooth. Les codes-source du programme sont accompagnés d'une

documentation réalisée par l'auteur lui-même. La documentation aide à mettre en pratique l'utilisation de BSS, sur différents réseaux bluetooth, comme ceux des téléphones portables...

Lien : **http://www.secuobs.com/bss-0.6.tar.gz**

II-E - Les Fuzzers pour les couches de transport et de réseau

II-E-1 - ISIC

ISIC est une suite d'outils de test de la stabilité d'une pile IP et de ses composantes (TCP, UDP, ICMP etc...). Il génère des piles de paquets pseudo-aléatoires sur le protocole ciblé. Dans la plupart des cas, les paquets ainsi créés sont acceptés. Ils sont ensuite envoyés à la machine ciblée pour contourner les règles de pare-feu ou trouver des bugs dans la pile IP. ISIC contient également un utilitaire servant à examiner le Hardware.

Site officiel : **http://www.packetfactory.net/Projects/ISIC**

II-E-2 - ip6sic

Ip6sic est un outil écrit en langage C qui permet de fuzzer la pile IPv6. Il fonctionne d'une manière très semblable à ISIC. Il a été élaboré pour fonctionner sur FreeBSD ainsi que sur OpenBSD et Linux.

Site officiel: **http://ip6sic.sourceforge.net**

III - À qui s'adresse ces outils ?

Quand un bug a été découvert, il est alors possible de réaliser d'autres tests pour savoir si la découverte est susceptible d'être exploitée. Si tel est le cas, un code d'exploitation est créé, le plus souvent à partir du framewok Metasploit. De ce fait, le fuzzing se destine principalement aux développeurs ainsi qu'aux chercheurs en sécurité afin de corriger la (ou les) vulnérabilité(s), comme ce fut le cas durant l'été 2006. En effet, au cours de cette période, l'expert en sécurité HD Moore à rendu "publique" l'utilisation du fuzzing.

Toutefois, les crackers utilisent ce même procédé afin de tirer profit de la faiblesse du système ou de l'application (contrôle à distance, contournement de la sécurité, etc...)

IV - Conclusion

Vous avez vu les principes fondamentaux du fuzzing, ainsi que les différents fuzzers. Ils permettent de découvrir plus rapidement des vulnérabilités parmi des milliers de lignes qui composent le code source d'un programme, et ils s'avèrent intérréssants pour traquer les failles de type cross-site scripting (XSS) DoS (déni de service), injection SQL, buffer overflow et format string. Il existe une grande variété de fuzzers issus notamment du monde Open Source, lequel développe de nombreux logiciels d'audits et de tests d'intrusions (voir la rubrique "liens"). Les outils de fuzzing sont de plus en plus présents dans le domaine de la sécurité, du fait d'une politique de sécurité applicative de plus en plus stricte. Néanmoins, le fuzzing n'est pas un procédé infaillible, car d'une part, certains critères d'un logiciel peuvent rendre son exploitation très complexe et d'autre part, un bug peut en cacher un autre.

V - Liens

http://packetstormsecurity.org/fuzzer - d'autres fuzzers.

http://safari.oreilly.com/9780321446114 - Ouvrage traitant des fuzzers, disponibles chez l'éditeur O'reillys

http://www.amazon.fr/Open-Source-Fuzzing-Tools-Rathaus/dp/1597491950 - Ouvrage traitant des fuzzers open source, disponible chez l'éditeur Syngress Media

Vidéo de la conférence de black-hat 2006 sur le fuzzing

http://www.journaldunet.com/solutions/securite/analyses/07/1105-logiciels-test-intrusion.shtml - Article traitant des tests d'intrusion grâce aux Fuzzers, réalisé par Christophe AUFFRAY pour JDN Solutions

VI - Remerciments

Je tiens à remercier jean philipe Dubbé (jpjp507) pour m'avoir fait rentré dans l'équipe de rédaction et pour m'avoir aidé à publier ce premier article. La nouvelle rédactrice en chef du magazine Hakin9, Margot Kompiel pour avoir autoriser la publication de cet article sur developpez.com. Je tennais aussi à remercier Mr sebastien Gioria,représentant Français de l'organisation OWASP, qui durant l'écriture de ce premier article pour Hakin9, s'est intérréssé à celui-ci tout en m'encourageant.

Enfin, je remercie la communauté de Developpez.com pour son accueil au sein de l'équipe.