HERRAMIENTA ModTester

Danel Lopez Agoües

1.Índice

2. Introducción

Modtester en una recopilación/adaptación de diferentes herramientas de pentesting para el protocolo industrial Modbus.

Actualmente existen diferentes framework y software para hacer pentesting del protocolo modbus. La finalidad de esta herramienta es hacer un toolkit unificado, juntando los software en un mismo metasploit, para que sean más fáciles las labores de pentesting de dicho protocolo.

La herramienta se encuentra en desarrollo, actualmente incluye funcionalidades de escaneo y de DoS.

3. Instalación

El metaexploit se encuentra disponible en https://github.com/Dangoro94/ModTester.

No hace falta instalar la aplicación, está escrita en python, por lo que solo debemos descargar el repositorio y ejecutar el archivo de python como root.
El exploit está escrito en python, por lo que antes de nada, deberemos disponer de la librería de python correspondiente. El metasploit también tiene diferentes dependencias, ya que los diferentes módulos funcionan con librerías diferentes. Las dependencias de todas las librerías se reúnen en la siguiente tabla.

3.1 Tabla de dependencias

	Funcionalidades	Dependencias	Instalación
ModTester	SMOD(ejecutable)	python 2.7.17	https://tecadmin.net/install-python-2-7-on-ubuntu-and-linuxmint/
		scapy 2.4.0	https://scapy.readthedocs.io/en/latest/installation.html

	Hping3 (necesaria instalación)	libpcap
		
		Hping3
			`sudo apt-get update`
			`sudo apt-get install libpcap-dev`
			
			sudo apt-get install hping3

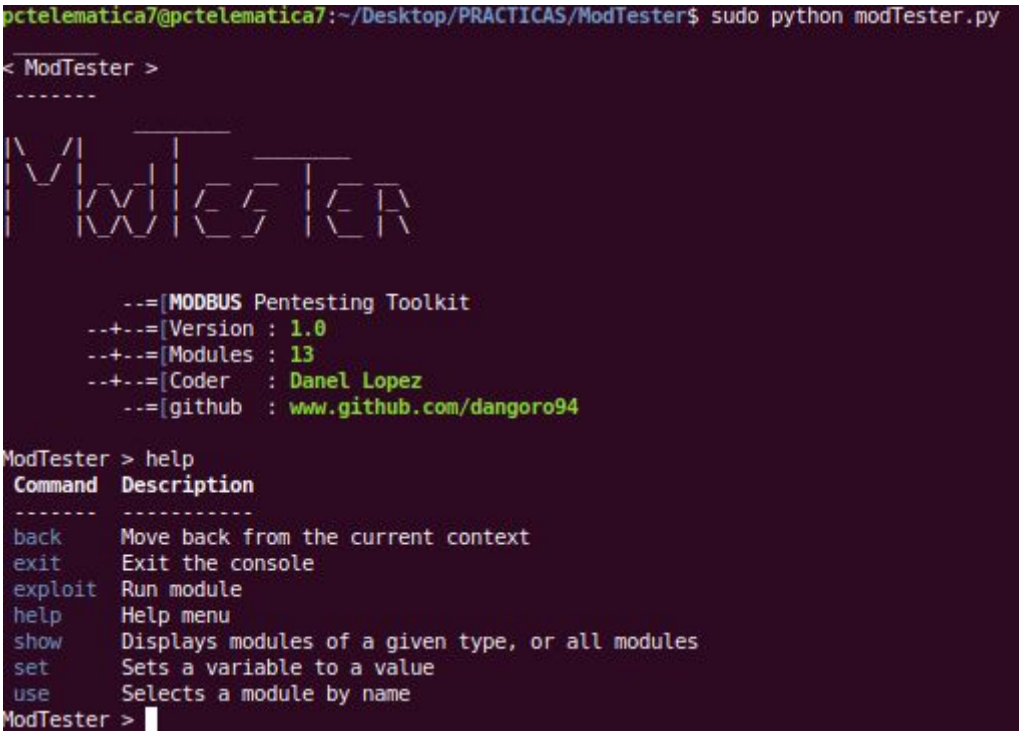
4. Usabilidad

La herramienta está basada en la interfaz del metasploit Smod, y sigue teniendo el mismo sistema para navegar por los diferentes módulos. El funcionamiento es el siguiente:

Corremos la aplicación desde una terminal.

sudo python modTester.py

Al iniciar nos sale la pantalla inicial y el prompt de ModTester. Con “help” obtendremos todos los comandos disponibles para navegar/usar el metasploit. (¡¡Importante tiene autocompletar!!)



Para movernos por la herramienta, primero visualizamos todos los módulos (“show modules”). Una vez elijamos uno, navegaremos hasta él para usarlo (“Use modbus/dos/xxx”). Una vez dentro del módulo seleccionado, cambiará el prompt y el metasploit se quedara en ese modulo a la espera de variables. Visualizamos (“show options”) las variables de ese módulo, y les daremos valor (“set RHOSTS dirección-ip”)

```
ModTester > show modules
 Modules                                  Description
 -------                                  -----------
 modbus/dos/floodingAttack                Flooding Attack with Hping script
 modbus/dos/writeSingleCoils              DOS Write Single Coil Function
 modbus/dos/writeSingleCoilsID            DOS Write Single Coil Function
 modbus/dos/writeSingleRegister           DOS Write Single Register
 modbus/dos/writeSingleRegisterID         DOS Write Single Register
 modbus/scanner/coilDiscover               Modbus Coilstatus discover
 modbus/scanner/discover                  Check Modbus Protocols
 modbus/scanner/discreteInputDiscover      Modbus Discrete Input Scanner
 modbus/scanner/getfunc                   Enumeration Function on Modbus
 modbus/scanner/holdingRegisterDiscover    Modbus Holding Register Scanner
 modbus/scanner/inputRegisterDiscover      Modbus Input Register Scanner
 modbus/scanner/portDiscover               Port scan with Hping
 modbus/scanner/uid                       Brute Force UID
ModTester > use modbus/scanner/uid
SMOD modbus(uid) > show options
 Name      Current Setting  Required  Description
 ----      ---------------  --------  -----------
 Function  1                False     Function code, Defualt:Read Coils.
 Output    True             False     The stdout save in output directory
 RHOSTS                     True      The target address range or CIDR identifier
 RPORT     502              False     The port number for modbus protocol
 Threads   1                False     The number of concurrent threads
SMOD modbus(uid) > set RHOSTS 214.67.123.45
SMOD modbus(uid) >
```

Una vez añadamos las variables, ejecutaremos el módulo con “exploit”. Con “back” volveremos al menú inicial para volver a navegar a otra funcionalidad.

```
SMOD modbus(uid) > exploit
[+] Module Brute Force UID Start
[+] Start Brute Force UID on : 214.67.123.45
Connection unsuccessful due to the following error :

SMOD modbus(uid) > back
ModTester >
```

5. Módulos disponibles

5.1 Dos

writeSingleRegister: Pasamos como parámetro una IP, un PUERTO, un VALOR(16 bit) y la DIRECCION de un registro de memoria Modbus. El módulo crea paquetes para modificar el valor de esa dirección al valor que le hemos dado, y hace un ataque de inyección continuo intentando cambiar el Register. Al estar inyectando nuevos valores de manera continua, si un operario intenta cambiar los valores del HMI, este será reescrito por el ataque de inyección, ya que se procesan muchas más peticiones, con lo que denegamos el servicio al HMI, no pudiendo controlar los Registros donde estamos inyectando datos.

writeSingleCoils: Funciona de la misma manera que *writeSingleRegister* pero en vez de direccionar a un registro, lo hace a un coil, el valor es solo de un bit, 1 o 0, a diferencia del valor de los registros.

floodingAttack: Esta funcionalidad utiliza el comando de linux hping3 para mandar paquetes en modo flood, intentando saturar la máquina objetivo y provocar un DoS. Tenemos diferentes parámetros que harán diferentes instancias de ataques, cuyo impacto dependerá de cómo usamos las variables. Como parámetros podremos elegir, el FLAG que usar en nuestros paquetes, una dirección IP, el PUERTO y la IP origen.

FLAG disponibles: -S, -F, -R, -P, -A
sIP (ip origen): “--rand-source” para ip aleatorias, o cualquier IP válida.

Con esta funcionalidad, obtenemos diferentes DoS dependiendo de los FLAG o la manera en la que intercambiamos paquetes con FLAG y sIP específicos para cerrar conexiones. Podemos generar diferentes DoS con el mismo módulo, dependiendo de las opciones.

5.2 Scanner

portDiscover: Le daremos como parámetro una IP y el número de puertos a descubrir (1000 por defecto). Nos devolverá los puertos que tienes abiertos y su protocolo. (Muchas veces no especifica si esel puerto no es por defecto el del protocolo, por ejemplo, el puerto 502 es modbus, pero si cambiamos a 520 nos sale que lo tiene abierto, pero no que esté ejecutando un servidor de modbus, para ello tendremos el siguiente ataque de descubrimiento).

Discover: ataque de reconocimiento. Como parámetros le daremos un PUERTO y una IP o rango de IPs. Si sabemos la IP de la máquina objetivo podemos comprobar

directamente si tiene modbus corriendo o no. Si pasamos como variable un rango de direcciones (192.123.123.1/24) el programa conectara con todas las direcciones y devolverá en cuales corre modbus.

UID: Con esta funcionalidad, obtenemos cuantos slaves puede tener esa maquina.

getFunc: Le pasaremos una IP. Nos devolverá cuales son las funciones de modbus que soporta el protocolo corriendo en la máquina objetivo.

Los siguientes 4 módulos hacen prácticamente lo mismo pero para cada rango de memoria modbus. Analizan qué direcciones devuelven excepciones o no, y muestra en pantalla las direcciones válidas y su valor.

coilDiscover: Devuelve todos los coils activos y su valor (0 o 1).

discreteInputDiscover: Devuelve todos los discreteInput activos y su valor(0 o 1).

holdingRegisterDiscover: Devuelve todos los holding register activos y su valor (2 palabras de 8 bit).

inputRegisterDiscover: Devuelve todos los input register activos y su valor. (La traducción a hexadecimal no se hace del todo bien, devuelve 2 números de 8 bit, para sacar el número real hay que sumarlos antes de traducirlo a hexadecimal.)