

UNIVERSIDAD DE MURCIA

GRADO EN INGENIERÍA INFORMÁTICA

4º CURSO

GRUPO 6

CURSO 2016/2017 - JUNIO

---

# Seguridad

---

## Práctica final

---

Alumnos:

Cristian Roche Borja

DNI: 76581531H

Alicia Ruiz Tovar

DNI: 48693813F

Docentes:

Gabriel López Millán

Gregorio Martínez Pérez





# Índice

<b>1. NMAP y Metasploit</b>	<b>4</b>
1.1. Víctima . . . . .	4
1.2. Atacante . . . . .	4
1.2.1. NMAP . . . . .	4
1.2.2. NMAP con Metasploit . . . . .	6
1.2.3. Wireshak: trazas . . . . .	6
1.3. Scripts NMAP . . . . .	8
1.3.1. Scripts /usr/share/nmap/scripts . . . . .	8
1.3.2. Realización de script básico . . . . .	8

# 1. NMAP y Metasploit

## 1.1. Víctima

Utilizaremos una máquina virtual de prueba. Esta máquina ha sido creada con vulnerabilidades para la práctica de ataques. La URL de descarga es la siguiente: [wiki.inf.um.es/metasploitable2/metasploitable-linux-2.0.0.zip](http://wiki.inf.um.es/metasploitable2/metasploitable-linux-2.0.0.zip).

La IP de esta máquina es la 192.168.62.189.

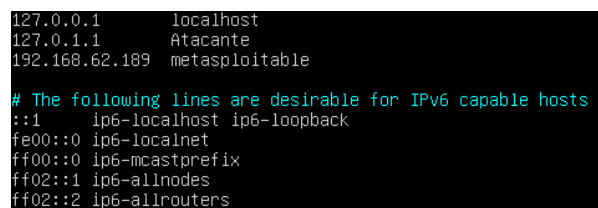
## 1.2. Atacante

### 1.2.1. NMAP

El equipo que actuará como atacante hace uso de la herramienta NMAP. Para instalarla ejecutamos el siguiente comando:

```
$ sudo apt-get install nmap
```

Establecemos en el archivo `/etc/hosts`, equivalente al DNS local, la IP de la víctima (192.168.62.189) y la denominamos `metasploitable`, como muestra la figura 1.



```
127.0.0.1    localhost
127.0.1.1    Atacante
192.168.62.189 metasploitable

# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

Figura 1: Atacante\_dns\_victima.

De esta forma, tenemos dos opciones para hacer referencia a la víctima. En la figura 2 se observa el resultado de este escaneo simple fruto de cualquiera de estas dos opciones.

```
$ nmap 192.168.62.189
$ nmap metasploitable
```

De forma un poco más elaborada, se puede ejecutar el escaneo de puertos haciendo uso de otras técnicas:

- Mediante listado de equipos: `$ nmap 192.168.62.1 192.168.62.10 192.168.62.189`
- Mediante subred: `$ nmap 192.168.62.0/24`
- Mediante un fichero que almacene las IPs (o las expresiones de las mismas) a analizar: `$ nmap -iL hosts.txt`, como muestra la figura 3.

```
Starting Nmap 7.01 ( https://nmap.org ) at 2017-04-23 13:06 CEST
Nmap scan report for 192.168.62.189
Host is up (0.0010s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds
alumno@Atacante:~$
```

Figura 2: Atacante\_nmap\_simplescan.

```
alumno@Atacante:~$ cat hosts.txt
192.168.62.189
192.168.62.1
alumno@Atacante:~$ cat hosts2.txt
192.168.61.0/24
metasploitable
192.168.62.1
192.168.62.200-220
alumno@Atacante:~$
```

Figura 3: Atacante\_nmapscan\_filecomplex.



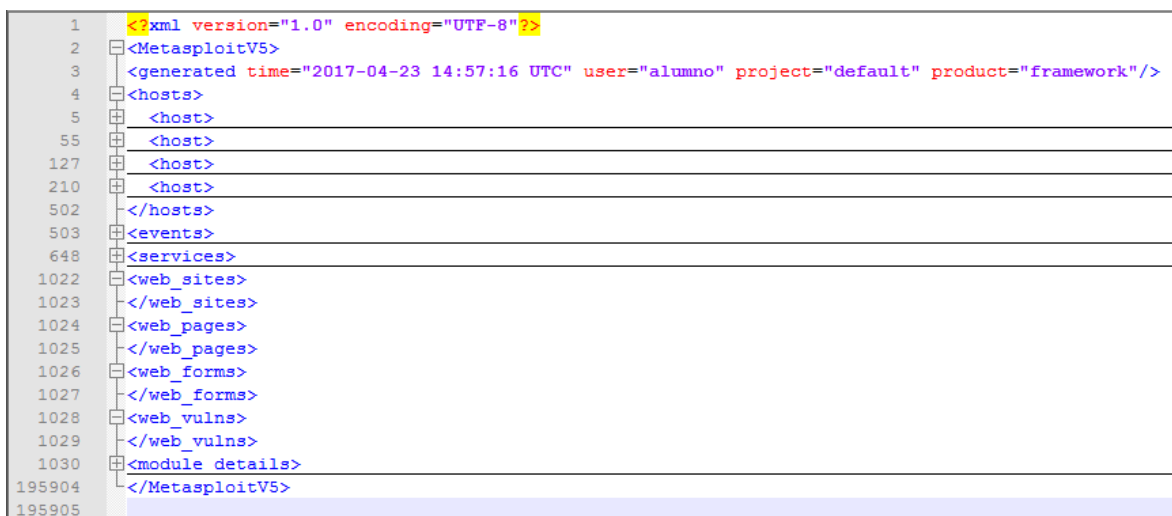


Figura 5: Atacante\_wireshar\_scaneo\_delay.

- \$sudo nmap -sS -mtu 24 -p 80 metasploitable 192.168.62.102. En el hots metasploitable y en la IP 192.168.62.102 se lanza un escaneo al puerto 80 con el bit SYN activado, como se muestra en la figura 6 Lo que se hace es enviar un paquete SYN, como si se fuera a abrir una conexión real y después se espera una respuesta. Si se recibe un paquete SYN/ACK esto indica que el puerto está abierto, mientras que si se recibe un RST (reset) indica que no hay nada escuchando en el puerto. Si no se recibe ninguna respuesta después de realizar algunas retransmisiones o se recibe un ICMP entonces el puerto se marca como filtrado.

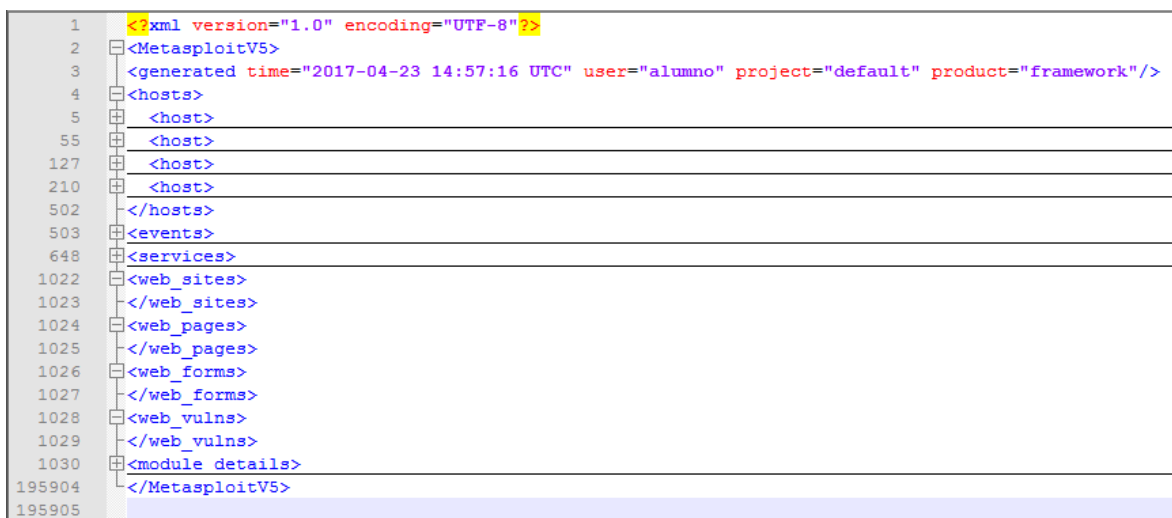


Figura 6: Atacante\_wireshar\_scaneo\_delay.

## 1.3. Scripts NMAP

### 1.3.1. Scripts `/usr/share/nmap/scripts`

En la instalación de NMAP se crea el directorio `/usr/share/nmap/scripts`, este directorio contiene una lista de scripts implementados por otros usuarios y que están diseñados para ser invocados desde el comando `nmap`. A continuación se describen algunos:

- `http-git.nse`: Realiza una conexión al puerto 80 de la víctima en busca de un servidor web activo, si el puerto está abierto, se intenta localizar un directorio `.git`. La existencia de este directorio implica que la víctima está realizando un control de versiones, por tanto, el siguiente paso que realiza el script es la búsqueda de coincidencias en *Github*, si se encuentran coincidencias, se muestran un mensaje al usuario con toda la información que se ha podido extraer de la víctima en *Github*.
- `smb-server-stats.nse`:
- `ssh2-enum-algos.nse`:
- `dhcp-discover.nse`:

### 1.3.2. Realización de script básico

Se pueden crear nuevos scripts adaptados a nuestras necesidades, que automaticen tareas habituales, o repetitivas. En la siguiente url <http://nmap.org/book/nse-tutorial.html> se describe la estructura que debe tener el script. Para poner en práctica este apartado, a continuación, incluye el contenido de un script realizado por nosotros, las acciones que realiza son las siguientes:

- Comprobar si el equipo objeto tiene el puerto 80 abierto (el número de puerto se puede cambiar a la hora de ejecutar el comando)
- En el caso de que se cumpla el paso anterior, se entiende que existe un servidor web en el equipo, por tanto, se solicita la página `index.html`, dicha página se crea por defecto en los navegadores web.
- La página web descargada se almacena en un fichero con el mismo nombre `index.html`

Para ejecutar el script, se debe escribir el siguiente comando:

```
$ nmap -p 80 <ip> --script=http-index
```

```
local http = require "http"
```



```
local io = require "io"
local shortport = require "shortport"
local stdnse = require "stdnse"

description = [[
Comprobamos si el host remoto tiene el puerto indicado activo, en ese
]]

---
-- @usage
-- nmap -p 80 <ip> --script=http-index
--
--80/tcp open  http
--|_http-index: /index.html Obtenido correctamente!
--
-- Version 0.1
-- Created 23/04/2017 - v0.1 - created by R&R_Asociados
--

author = "R&R Asociados"
license = "Open License"
categories = {"discovery"}

portrule=shortport.http

action = function( host, port )

local result
local output = stdnse.output_table()
local request_type
path = "/index.html"
    result = http.get(host, port, path)
    request_type = "GET"

if ( not(200 <= result.status and result.status < 210) ) then
    output.error = ("ERROR: Fallo al obtener la url %s"):format(port)
    return output,output.error
end

local fname = "index.html"
local f = io.open(fname,"w")
```

```
if ( not(f) ) then
    output.error = ("ERROR: Fallo al crear/abrir el fichero %s"):format(path)
    return output,output.error
end

    io.output(f)
    io.write(table.tostring( result ))
    f:close()

if ( 200 <= result.status and result.status < 210 ) then
    output.result = ("%s Obtenido correctamente!"):format(path)
    return output,output.result
end

    return
end

-- Transformacion de tipo table en string
function table.val_to_str ( v )
    if "string" == type( v ) then
        string.gsub( v, "\n", "\\n" )
        if string.match( string.gsub(v,"['\""]",""), '^"+$' ) then
            return "'" .. v .. "'"
        end
        return '"' .. string.gsub(v,'"', '\\"' ) .. '"'
    else
        return "table" == type( v ) and table.tostring( v ) or
            tostring( v )
    end
end

function table.key_to_str ( k )
    if "string" == type( k ) and string.match( k, "^[_%a][_%a%d]*$" ) then
        return k
    else
        return "[" .. table.val_to_str( k ) .. "]"
    end
end

function table.tostring( tbl )
    local result, done = {}, {}
    for k, v in ipairs( tbl ) do
```

```
        table.insert( result, table.val_to_str( v ) )
        done[ k ] = true
    end
    for k, v in pairs( tbl ) do
        if not done[ k ] then
            table.insert( result,
                table.key_to_str( k ) .. "=" .. table.val_to_str( v ) )
        end
    end
    return "{" .. table.concat( result, "," ) .. "}"
end
```

El script contiene un control de errores, por lo que se mostrará uno de los siguientes resultados:

- "ERROR: Fallo al obtener la url index.html"
- "ERROR: Fallo al crear/abrir el fichero index.html"
- "index.html Obtenido correctamente!"