

Scripting con NMAP



Objetivos

- Presentar las funcionalidades de NSE
- Analizar y adecuar los Scripts a nuestra necesidad
- Desarrollar un script base.
- Análisis de resultados en pruebas de seguridad



Y que es Lua?

Lua : (pronunciado LOO-ah) = Luna



Lua es un Lenguaje de Scripting creado en 1993 por estudiantes de la Universidad de Rio de Janeiro – Brazil. Pensado inicialmente para su uso en computación gráfica.

Es un lenguaje “portable” ligero, muy amigable y de fácil integración con otras tecnologías.

Ej: Adobe's Photoshop Lightroom, El proyecto Ginga (middleware para televisión digital), World of Warcraft.

Lua : Sintaxis Basica

Lenguaje “case-sensitive”

Lua puede llamar (y manejar) funciones escritas en Lua y funciones escritas en C

Tipos de dato :

nil, boolean, number, string, function, userdata, thread y table.

Nil es el tipo del valor nil, cuya principal propiedad es ser diferente de cualquier otro valor.

Userdata : guarda en memoria datos en C

Tables : implementa arrays que pueden ser indexados con cualquier valor (excepto nil)

Referencia: <http://www.lua.org/manual/5.1/>

Lua : Sintaxis Basica

Lua puede convertir automáticamente entre valores string y valores numéricos en tiempo de ejecución. Cualquier operación aritmética aplicada a un string intenta convertir el mismo en un número, siguiendo las reglas normales de conversión. Y viceversa, cuando un número se usa donde se espera un string el número se convierte a string, con un formato razonable.

Referencia: <http://www.lua.org/manual/5.1/>

Introducción Lua (cont) :

Definición de Variables

```
-- esto es un comentario
```

```
n1=3
```

```
n2=3.0
```

```
n3=3.1416
```

```
n4=314.16e-2
```

```
n5= 0.31416E1
```

```
n6=0xff
```

```
print (n1, n2, n3, n4, n5, n6)
```

**Definiendo
Variables**

**Imprime el valor
que contienen
las variables**

Introducción Lua (cont) :

Definición de Variables

```
text1='Hola\n"Mundo"'
text2="Hola\n"Mundo\""
text2= text2 .. "-ekoparty 2012"
text3='\072\111\108\097'
text4=[[ \072\111\108\097]]
text5=[[=[Hola
      Mundo]=]]

print (text1, text2, text3, text4, text5)
```

**Definiendo
Variables**

ascii table : <http://www.theasciicode.com.ar/>

Introducción Lua (cont)

```
a ,b = 8, 3
```

```
c = -a
```

```
print (c)
```

```
print (a - b)
```

```
print (a +b)
```

```
print (a * b)
```

```
print (a /b)
```

```
print (a^b)
```

**Operadores
Aritméticos**

Introducción Lua (cont) :

```
a="Hola"; b="hola"; c="hola"
```

```
print (a == b)
```

```
print (a ~= b)
```

```
print (a < b)
```

```
print (a > b)
```

```
print (a >= c)
```

```
print (b <= c)
```

Comparaciones

Introducción Lua (cont) :

“and” “or” y “not”

```
a = true; c = false; c = true;
```

```
print ((a or b) and (c or b))
```

```
print ((a and b) or (c and b))
```

```
print ( not (a or b))
```

Lógicas

Introducción Lua (cont)

Tipos de Variables :

```
A = 1          -- variable global A con valor 1
do
  local A = A   -- nueva variable 'A', de tipo local valor 1
  print(A)      --> imprime variable local A
  A = A+1
  print(A)      --> valor 2 luego de sumarle 1
do
  -- otro bloque
  local A = A+1  -- otra variable local 'A'
  print(A)       --> 3
end
print(A)        --> 2
end
print(A)        --> 1 (el valor de la variable global)
```

Variable
Global

Variable
Local

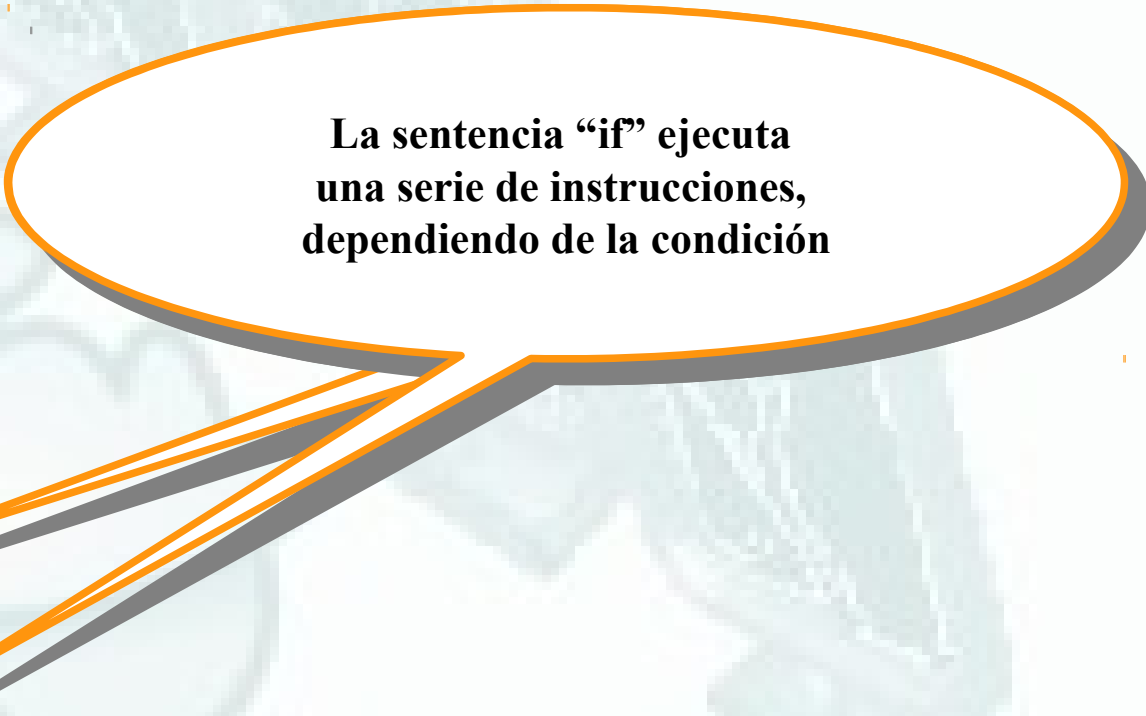
Variable
Local

Lua : Estructuras condicionales

if:else:end
if:else:elseif:end

```
local answer
repeat
    io.write("Ingrese un Puerto: ")
    io.flush()
    answer=io.read()
until answer ~= nil
```

```
if answer == "80" then
    print ("Servidor Web")
elseif answer=="25" then
    print ("Servidor SMTP")
else
    print ("Puerto no reconocido como HTTP o SMTP")
end
```



La sentencia “if” ejecuta
una serie de instrucciones,
dependiendo de la condición

Lua : Bucles

```
local rdm = math.random
for i = 1, 10, 2 do
  for j= 1,10,2 do
    j=j*i;
    local x = rdm(i,j); print (x)
  end
end
```

```
while i <= 11 do
  print (i)
end
```

```
i = 5
repeat
  print (i)
  i = i + 0.5
until i > 11
```

Bucles con
for – do -end
while do - end
repeat - until

math.sin | math.random (Librería matemática – seno, coseno, random, etc)

Lua : manejo de archivos

```
=====
mivariable = io.open("miarchivo.txt","r")
while true do
    line = mivariable.read(mivariable)
    if not line then break end
    print (line)
end
=====
```

- *"r" = lee del archivo sin modificarlo*
- *"w" = crea o sobrescribe el archivo*
- *"a" = agrega el texto al final del archivo.*

Manejo de archivos
con libreria :

io.open(file,modo)

Lua : manejo de archivos

```
local mivariable = io.open("miarchivo.txt", "r")
while true do
    line = mivariable:read("*l")
    print (line)
    break
end
print ("Saliendo del while")
print ("Imprimiendo la siguiente linea con *l ")
line = mivariable:read("*l")
print (line)
print ("Imprimiendo el resto del texto con *a ")
line = mivariable:read("*a")
print (line)
```

Leyendo archivos:
io.open(file,modo)

Lua : manejo de archivos

Escribiendo archivos:

`io.open(file,modo)`

```
local mivariable = io.open("miarchivo2.txt","a")
mivariable:write("Añadiendo texto a la ultima linea \n\r")
mivariable:flush() -- (Grabando)
mivariable:close() --(Cerrando el archivo y grabándolo)
print("Texto agregado correctamente")
```

Introducción Lua (Funciones)

```
function multiplicar(n,m)
```

```
    return n*m
```

```
end
```

```
x = function(a,b) return a+b, a-b end
```

```
print (multiplicar(2,3))
```

```
print (x(5,6))
```

Creando las funciones
multiplicar() y x()

Invocando la funciones
multiplicar() y x()

Referencias

<http://www.lua.org/manual/5.1/>

<http://lua-users.org/wiki/>

<http://lua.gts-stolberg.de/en/>

<http://seguridadyredes.wordpress.com>

Link Oficial



NMAP SCRIPTING ENGINE



Que es NSE?

NSE (Nmap Scripting Engine), es un motor de scripting que viene integrado con NMAP y nos permite la ejecución de scripts (Desarrollados en el Lenguaje LUA) de manera rápida y sencilla, permitiéndonos obtener mayor información con nuestros escaneos.

Nmap en sus últimas versiones ya incorpora varios scripts, algunos de los cuales han resultado muy útiles tanto para la realización de un pentesting como para la gestión de una red.

Usos de NSE?

Dentro de las tareas que se pueden realizar con los scripts en NSE tenemos:

- Descubrimiento de red.
- Detección de versiones/servicios mejorada
- Detección de vulnerabilidades.
- Detección de gusanos y backdoors.
- Explotación de vulnerabilidades, etc.

NSEDoc

[Index](#)

[NSE Documentation](#)

Categories

[auth](#)

[broadcast](#)

[brute](#)

[default](#)

[discovery](#)

[dos](#)

[exploit](#)

[external](#)

[fuzzer](#)

[intrusive](#)

[malware](#)

[safe](#)

[version](#)

[vuln](#)

[Scripts \(show 419\)](#)

[Libraries \(show 103\)](#)

Categorías de un NSE

brute: Estos scripts utilizan ataques de fuerza bruta para obtener las credenciales de autenticación de un servidor remoto.

auth: Estos scripts tratan de autenticar con las credenciales de autenticación que les pasemos como parametro o haciendo algún tipo de bypassing

vuln : Estos scripts verifican vulnerabilidades puntuales, en base a vulnerabilidades conocidas, y se obtiene un output se comprueba que el objetivo es vulnerable.

Categorías de un NSE

default: Son los scripts predeterminados y pueden ser usados mediante los parametros `-sC` o `-A`.

discovery: Estos scripts intentan obtener información haciendo consultas a diversos servicios accesibles desde la red, como dispositivos con SNMP habilitado, servicios de directorio, etc.

broadcast: Los Scripts de esta categoría suelen servir para descubrir hosts que no son vistos mediante la línea de comandos, por Broadcasting, dentro de la red local.

Categorías de un NSE

exploit : Estos scripts tienen como objetivo explotar alguna vulnerabilidad de manera activa.

external: Los Scripts en esta categoría pueden enviar o consultar datos sobre DB de terceros o contra otro recurso de red.

dos : Los Scripts en esta categoria pueden causar denegación de servicio, esto ocurre ya que generan la caída de un servicio como efecto secundario al verificar una vulnerabilidad.

Categorías de un NSE

fuzzer: Estos scripts están diseñados para enviar en cada paquete campos inesperados o seleccionados al azar contra el software del servidor. Si bien esta técnica puede ser útil para encontrar errores y descubrir las vulnerabilidades del software, a la vez es un proceso lento y con un intenso uso del ancho de banda.

intrusivo: *Scripts con los riesgos son demasiado altos, impactando mayor mente en la disponibilidad del equipo, generando un uso excesivo de recursos (CPU / Ancho de Banda) y activando alertas de algún tipo en el equipo evaluado (IPS /HIPS).*

Categorías de un NSE

malware: *Scripts que nos ayudan al descubrimiento de malware y similares*

safe : *modo seguro para la ejecución de scripts, suele ser inofensivo para el objetivo.*

version : *Los scripts de esta categoría son una extensión para la característica de detección de versiones, agregándole mayor precisión en la detección de versiones. (requiere -sV)*

Direcciones Importantes

Rutas Importantes :

/usr/local/share/nmap/scripts

/usr/local/share/nmap/nse-lib

Ultimo listado oficial de scripts

<http://nmap.org/svn/scripts/>

Repositorios de nmap-Experimental

<https://svn.nmap.org/nmap-exp>

Estructura del NSE

ENCABEZADO :

- **id** : Identificador, nombre corto del script, el cual será mostrado en el output de Nmap.
- **description**: Reseña o explicación breve acerca del script, notas del autor, etc. Esta información es de carácter informativo.
- **author**: Persona que desarrollo el script.
- **tags / categories** : categoría del script
- **license** : licencia (no es procesado)
- **dependencies / require** : scripts que necesita ejecutar antes, depende de ellos para funcionar.

Estructura del NSE

```
1  description = [[
2    Checks if a VNC server is vulnerable to the RealVNC authentication bypass
3    (CVE-2006-2369).
4  ]]
5  author = "Brandon Enright"
6  license = "Same as Nmap--See http://nmap.org/book/man-legal.html"
7  ---
8  -- @output
9  -- PORT      STATE SERVICE VERSION
10 -- 5900/tcp open  vnc      VNC (protocol 3.8)
11 -- |_realvnc-auth-bypass: Vulnerable
12
13 categories = {"auth", "default", "safe"}
14
15 require "shortport"
16
17 portrule = shortport.port_or_service(5900, "vnc")
18
19 action = function(host, port)
20   local socket = nmap.new_socket()
21   local result
22   local status = true
23
24   socket:connect(host, port)
25
26   status, result = socket:receive_lines(1)
27
28   if (not status) then
29
30
31
32   socket:send("RFB 003.008\n")
33   status, result = socket:receive_bytes(2)
```

Lua length: 1018 lines: 51 Ln: 51 Col: 4 Sel: 0 UNIX ANSI INS

Estructura del NSE

FUNCIONES:

- **portrule / hostrule** (reglas): Se espera unos datos para decidir si se cumple y ejecuta el script. Los datos pueden ser host y/o port.
- **action (acción)** : es la parte importante de script, es la acción que tomará el script (el script propiamente dicho).

Estructura del NSE

```
13 categories = {"auth", "default", "safe"}
14
15 require "shortport"
16
17 portrule = shortport.port_or_service(5900, "vnc")
18 action = function(host, port)
19     local socket = nmap.new_socket()
20     local result
21     local status = true
22
23     socket:connect(host, port)
24
25     status, result = socket:receive_lines(1)
26
27     if (not status) then
28
29
30
31
32     socket:send("RFB 003.008\n")
33     status, result = socket:receive_bytes(2)
34
35     if (not status or result ~= "\001\002") then
36
37
38
39
40     socket:send("\001")
41     status, result = socket:receive_bytes(4)
42
43     if (not status or result ~= "\000\000\000\000") then
44
45
46
47
48     socket:close()
49
50     return "Vulnerable"
51 end
```


Usos de NSE

```
--version-light: Limit to most likely probes (intensity 2)
--version-all: Try every single probe (intensity 9)
--version-trace: Show detailed version scan activity (for debugging)
```

SCRIPT SCAN:

```
-sC: equivalent to --script=default
--script=<Lua scripts>: <Lua scripts> is a comma separated list of
    directories, script-files or script-categories
--script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
--script-trace: Show all data sent and received
--script-updatedb: Update the script database.
```

OS DETECTION:

```
-O: Enable OS detection
--osscan-limit: Limit OS detection to promising targets
--osscan-guess: Guess OS more aggressively
```

TIMING AND PERFORMANCE:

```
Options which take <time> are in seconds, or append 'ms' (milliseconds),
's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
-T<0-5>: Set timing template (higher is faster)
--min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
--min-parallelism/max-parallelism <numprobes>: Probe parallelization
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
```

--More--

Usos de NSE

Para utilizar estos Script solo debemos ejecutar la siguiente sitaxis : **--script** (categoría|directorio|nombre|all)

Tenemos también:

--script-args: para pasar argumentos al script

--script-trace: nos muestra información de comunicaciones internas realizadas por el script

--script-updatedb : para actualizar la base de datps de los script contenida en /scripts

Usos de NSE

```
Nmap done: 1 IP address (1 host up) scanned in 83.78 seconds
int31@labz:~$ nmap -PN --script="http* and not http-brut* and not http-*--bru*" [REDACTED]

Starting Nmap 5.51 ( http://nmap.org ) at 2011-11-05 05:18 UTC
Nmap scan report for [REDACTED]
Host is up (0.040s latency).
Not shown: 981 closed ports
PORT      STATE      SERVICE
21/tcp    open      ftp
22/tcp    filtered  ssh
25/tcp    open      smtp
80/tcp    open      http
|_ http-date: Sat, 05 Nov 2011 05:31:05 GMT; +12m10s from local time.
|_ http-domino-enum-passwords:
|_   ERROR: No valid credentials were found (see domino-enum-passwords.username and domino-enum-passwords.password)
|_ http-methods: OPTIONS TRACE GET HEAD POST
|_ Potentially risky methods: TRACE
|_ See http://nmap.org/nsedoc/scripts/http-methods.html
|_ http-title: Error
|_ http-iis-webdav-vuln: WebDAV is DISABLED. Server is not currently vulnerable.
|_ http-malware-host: Host appears to be clean
|_ http-headers:
|_   Content-Length: 218
|_   Content-Type: text/html
|_   Server: Microsoft-IIS/6.0
|_   MicrosoftOfficeWebServer: 5.0_Pub
|_   X-Powered-By: ASP.NET
|_   Date: Sat, 05 Nov 2011 05:31:05 GMT
|_   Connection: close
|_
|_ (Request type: GET)
|_ http-enum:
|_   /_vti_bin/fpcount.exe?Page=default.asp|Image=3: Frontpage folder
|_   /_vti_bin/shtml.dll: Frontpage folder
135/tcp    filtered  msrpc
139/tcp    filtered  netbios-ssn
443/tcp    open      https
```

Aqui un ejemplo:

```
description= [[Banner en ftp y ftp-data ]]
author ="ekoparty"
categories = {"safe","discovery"}
require "comm"
require "shortport"
portrule = shortport.port_or_service(21,"ftp",{"tcp", "udp"})
action=function(host, port)
    local status, result = comm.exchange(host, port, "data", {lines=1,
    proto=port.protocol})
    if status then
        return result
    end
end
end
```

Hagamos uno:

```
description= [[Banner en ftp y ftp-data ]]  
author ="ekoparty"  
categories = {"safe","discovery"}  
require "comm"  
require "shortport"
```

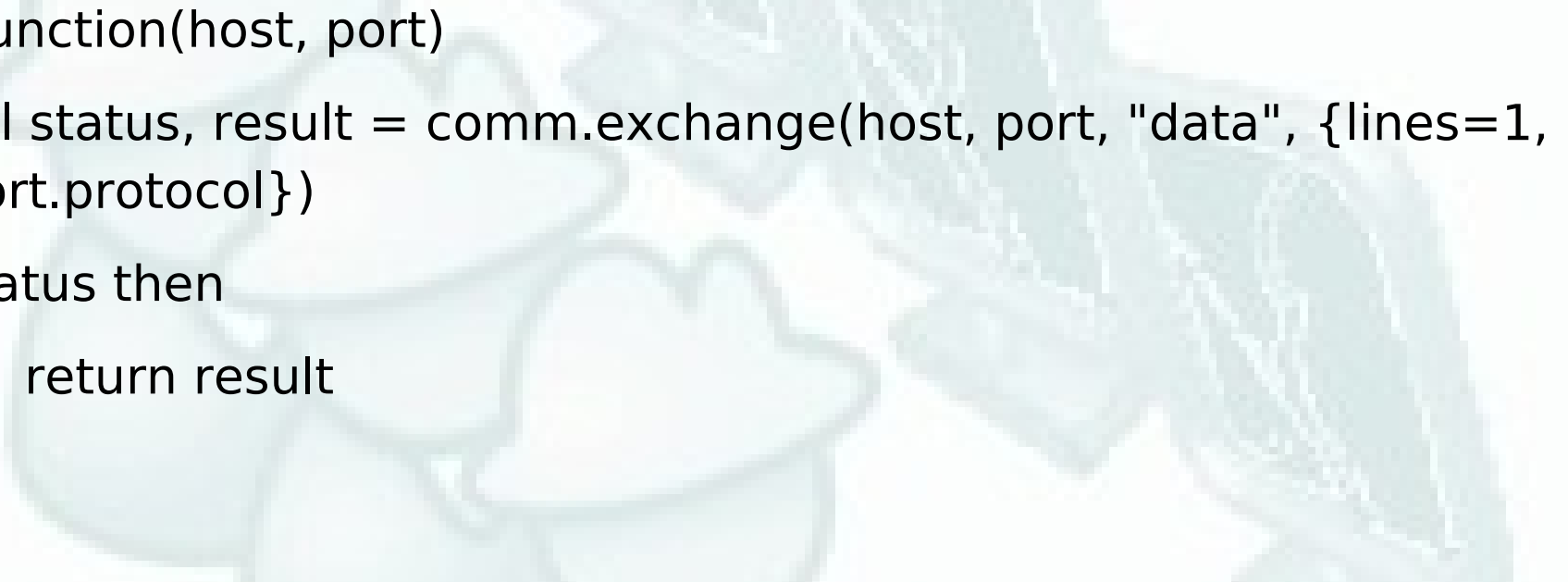
Hagamos uno:

```
--portrule = shortport.port_or_service(21,"ftp",{ "tcp", "udp"}
```

```
    portrule=function(host, port)
        if(
            port.number == 20 or
            port.number == 21 or
            port.service == "ftp-data" or
            port.service == "ftp"
            and port.protocol == "tcp" )
        then
            return true
        else
            return false
        end
    end
end
```

Hagamos uno:

```
action=function(host, port)
    local status, result = comm.exchange(host, port, "data", {lines=1,
proto=port.protocol})
    if status then
        return result
    end
end
end
```



Usos de NSE

```
root@ukupacha:/# nmap -p20-22 192.168.1.100 --script=ftp-banner.nse

Starting Nmap 6.01 ( http://nmap.org ) at 2012-09-13 00:04 PET
Nmap scan report for 192.168.1.100
Host is up (0.00076s latency).
PORT      STATE SERVICE
20/tcp    closed ftp-data
21/tcp    open  ftp
|_ftp-banner: 220 Bienvenidos al ftp.fbi.gov sus acciones estan siendo mon
22/tcp    closed ssh
MAC Address: A0:88:B4:55:23:A4 (Intel Corporate)

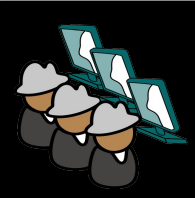
Nmap done: 1 IP address (1 host up) scanned in 0.35 seconds
root@ukupacha:/#
```




Reto time ! Reto 4

Ahora que ya ubicamos los FTP, podemos implementar en el script la validación de usuario anónimo?

Valor : 5 puntazos !



Reto time ! Reto 5

Nos encontramos en una red con múltiples servicios y requerimos obtener mediante scripts las versiones (mediante los banners) de todos los servicios web.

Valor : 10 puntazos !



Preguntas