

Introdução ao Nmap Scripting Engine NSE

[13 de abril de 2015](#) / [maximoz](#)

Introdução

Todos nós conhecemos ou já ouvimos falar da ferramenta Nmap, que teve como propósito inicial a varredura de portas e mapeamento de redes, mas que com o tempo, suas utilidades foram se diversificando nos dando uma quantidade maior de informações sobre o alvo por ser uma ferramenta de código aberto. Assim, com a ajuda da comunidade, a ferramenta evoluiu. Com isso, o Nmap hoje pode ser usado também como ferramenta de teste de invasão cujo o auxiliar é o Nmap Scripting Engine, ou NSE. Essa engine, integrada ao Nmap, usa scripts escritos na linguagem Lua, desenvolvidos para variadas finalidades dentro de um teste de invasão, como detecção de vulnerabilidades, a extensão dos scripts são .nse e se encontram no diretório `/usr/share/nmap/scripts`. Sendo assim, você pode escrever um script seu e usá-lo junto ao Nmap.

Obs.: Lembrando que o NSE não vai saber se o script que está sendo executado é malicioso ou prejudicial ao seu sistema, então cuidado ao usar scripts feitos por outras pessoas.

Os scripts

Todos os scripts, atualmente quase 500, são divididos em categorias para facilitar e automatizar tarefas na hora do scan (um script pode estar em mais de uma categoria). Por exemplo, você pode fazer o scan usando todos os scripts da categoria **auth** (mecanismos de autenticação), como pode fazer usando todos os scripts de todas as categorias ou pode usar um script específico. As categorias são: **auth, broadcast, brute, default, discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, vuln**. Você pode ter acesso a todos os scripts com suas descrições e em suas respectivas categorias no site do Nmap: <http://nmap.org/nsedoc/>

Obs.: A categoria **intrusive** são os scripts considerados inseguros pelos desenvolvedores, ao contrário da categoria **safe**, que são os seguros.

Uso padrão do NSE:

```
$ -sC
$ --script=default
$ -A
```

Essa flag tem como objetivo usar somente a categoria padrão de scripts para um scan mais rápido usando menos scripts mas que também retorne informações satisfatórias para uma das etapas do teste de invasão. Vamos, então, à prática.

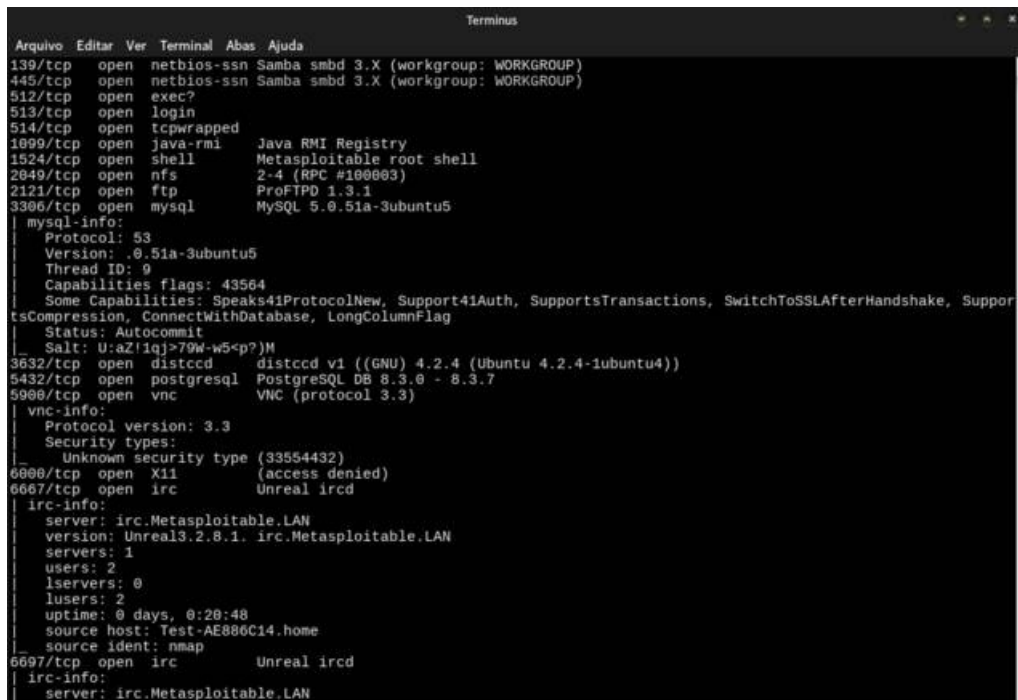
Fiz uma varredura padrão usando `-sS` (*SYN Scan*) e chamei o Scripting Engine (`-sC`). Vemos que, entre outros, tomando como exemplo, foi utilizado dois scripts, o [ssh-hostkey](#), que mostra as chaves do

servidor SSH, e o [http-title](#), que tem como finalidade retornar o título da página.

Vejamos outro exemplo. Aqui usei uma máquina virtual *Metasploitable*, que é uma máquina proveniente do *Metasploit Framework* que podemos usar como nosso laboratório para testes como esse pois já vem configurada com vulnerabilidades para podermos explorá-las.

```
$ nmap -sS -sV -O -v -sC -p- 192.168.25.47
```

Utilizei novamente a flag de scripts padrões do NSE (*-sC*) sobre todas as portas (*-p-*), no modo verbose (*-v*), lembrando que as flags *-sV* e *-O* foram desnecessárias nesse comando pois quando utilizamos o NSE, dentro dos scripts padrões, já está incluso a detecção de versão e de sistema operacional. Voltando ao que interessa, retornou o seguinte:



```
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
139/tcp   open    netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp   open    netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
512/tcp   open    exec?
513/tcp   open    login
514/tcp   open    tcpwrapped
1099/tcp  open    java-rmi    Java RMI Registry
1524/tcp  open    shell      Metasploitable root shell
2049/tcp  open    nfs        2.4 (RPC #100003)
2121/tcp  open    ftp        ProFTPD 1.3.1
3306/tcp  open    mysql      MySQL 5.0.51a-3ubuntu5
mysql-info:
  Protocol: 53
  Version: 0.51a-3ubuntu5
  Thread ID: 0
  Capabilities flags: 43564
  Some Capabilities: Speaks41ProtocolNew, Support41Auth, SupportsTransactions, SwitchToSSLAfterHandshake, Support
  Compression, ConnectWithDatabase, LongColumnFlag
  Status: Autocommit
  Salt: U:az!1qj>79W-w5<p?)M
3632/tcp  open    distccd    distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open    postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open    vnc        VNC (protocol 3.3)
vnc-info:
  Protocol version: 3.3
  Security types:
    Unknown security type (33554432)
6000/tcp  open    X11        (access denied)
6667/tcp  open    irc        Unreal ircd
irc-info:
  server: irc.Metasploitable.LAN
  version: Unreal3.2.8.1. irc.Metasploitable.LAN
  servers: 1
  users: 2
  lservers: 0
  lusers: 2
  uptime: 0 days, 0:20:48
  source host: Test-AE886C14.home
  source ident: nmap
6697/tcp  open    irc        Unreal ircd
irc-info:
  server: irc.Metasploitable.LAN
```

Dessa vez mais informações nos foi dada pois na *Metasploitable* há mais portas abertas e, consequentemente, mais scripts foram utilizados. Foram esses da imagem exceto vários outros que não couberam no print. Percebemos que o NSE nos ajudou a obter dados com mais detalhes do que quando fazemos um scan normal com o Nmap.

Para fazer um scan mais específico, você pode usar a flag *--script* cujo os argumentos são usados com operadores lógicos (*and*, *or*, *not*) para definir o que será usado, um script especificando seu caminho, uma categoria, todos os scripts (*all*), ou fazer o uso de expressões, que são o uso dos operadores lógicos entre aspas duplas. Vejamos:

```
$ --script script | categoria | diretório | expressão | all
```

Com uso de expressões:

```
$ --script "default or safe"
```

// Faz o scan usando as categorias *default* ou *safe*

```
$ --script "http-* and not (brute or dos or exploit)"
```

// Usa somente os scripts que começarem com *http-* exceto os das categorias *brute*, *dos* e *exploit*

Vejamos um exemplo usando o seguinte comando:

```
$ nmap -sS --script "http-* and not(brute or dos or exploit)" -p 80 192.168.25.47
```

```
| http-email-harvest:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=new-host-13.home
| SomeWikiName@somewhere.test
| you@yourdomain.com
| a@z.com
| secondary@home.com
| webmaster@your.comp
| Peter@Thoeny.com
| _ name@domain.com
```

```
80/tcp open http
```

```
| http-enum:  
| /tikiwiki/: Tikiwiki  
| /test/: Test page  
| /phpinfo.php: Possible information file  
| /phpMyAdmin/: phpMyAdmin  
| /doc/: Potentially interesting directory w/ listing on 'apache/2.2.8 (ubuntu) dav/2'  
| /icons/: Potentially interesting folder w/ directory listing  
|_ /index/: Potentially interesting folder
```

Como foram muitos scripts, destaquei somente alguns resultados mas, ainda assim, relevantes. Vemos aqui o [http-email-harvest](#) que, evidentemente, extrai todos os emails do servidor e o [http-enum](#) que nos enumera diretórios e arquivos do servidor que podem ser acessados sem autenticação como o arquivo phpinfo.php que há muitas informações que podem ajudar um atacante, como a versão do kernel do sistema.

```
| http-slowloris-check:  
| VULNERABLE:  
| Slowloris DOS attack  
| State: VULNERABLE  
| Description:  
| Slowloris tries to keep many connections to the target web server open and hold them open as long as possible.  
| It accomplishes this by opening connections to the target web server and sending a partial request. By doing so, it starves the http server's resources causing Denial Of Service.  
| Disclosure date: 2009-09-17  
| References:  
|_ http://ha.ckers.org/slowloris/
```

Descobrimos que o servidor é vulnerável a ataque DoS com Slowloris cujo já há um artigo falando sobre essa [ferramenta](#).

Passando parâmetros:

É possível passar parâmetros/argumentos para os scripts, possibilitando seu scan ficar mais personalizado e você ter resultados mais precisos e específicos. Você estará usando as seguintes flags:

```
$ ... -script-args argumento1=valor1, argumentoX=valorX
```

Por exemplo, para scripts que funcionam em cima de mecanismos de autenticação você usará:

```
$ ... -script-args user="foo",pass="bar"
```

Para scripts que eventualmente realizarão comandos internos, com o argumento a seguir setado, o script mostrará um arquivo sensível do sistema:

```
$ ... -script-args cmd='cat /etc/shadow'
```

ou

```
$ ... -script-args cmd='grep root /etc/shadow' // Mostrará informações como o hash de senha do usuário root
```

Se você tiver a ideia de combinações de argumentos mas que são grandes, você pode escrevê-los em um arquivo separado e depois usá-lo sem poluir muito o terminal usando a seguinte flag:

```
$ ... -script-args-file <arquivo>
```

Dica

Se você quiser salvar todo o seu scan e importar pra algum banco de dados, você pode usar a flag de output `-oX <file>`, por exemplo, para salvar um arquivo em XML e importá-lo para o metasploit para uma exploração de vulnerabilidades com outras ferramentas.

Conclusão

É isso aí, pessoal, só avisando que esse tutorial é baseado num Webinar da Clavis, tentei explicar o mais claramente possível, e saibam que essa é só a ponta do iceberg, tem muito mais assunto relacionado a esse que cabe a você e sua vontade de estudar, só queria que vocês tivessem uma ideia do poder que tem tal ferramenta. Deixarei alguns links que acho úteis relacionados a esse assunto. Espero que vocês tenham gostado, esse é o meu primeiro artigo que escrevo, tenham piedade.

Links úteis

[Webinar Clavis: Teste de invasão com Nmap Scripting Engine](#)

[Documentação do NSE](#)

[Máquina Laboratório Metasploitable](#)