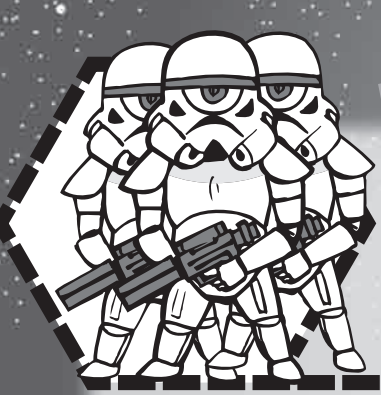

スクリプトを使って

より深くNmapを使いこなそう!

Nmap Scripting Engine

NmapにはNmap Scirpting Engineという自動処理エンジンが搭載されている。これを使いこなすと上級者の仲間入りできるかも!?

文=勇士Q

Nmap Scripting Engine とは?

Nmap Scirpting Engine（以下 NSE）とは多種多様なネットワーク処理を自動化するスクリプトを簡単に書くことができる、Nmap の強力で柔軟な機能の 1 つだ。この機能を使えば脆弱性検知やより詳しいサービスの判別を行うことができる。

また、NSE では Lua というスクリプト言語が使われており、これを使うことでより Nmap を使いこなすことが可能となる。他の手続き型言語（例えば C 言語）を知っている方ならば比較的覚えやすい言語だろう。これらのスクリプトとライブラリは Nmap をインストールする時に自動的にインストールされる。Linux の場合、標準ではスクリプトが /usr/local/share/nmap/scripts、ライブラリが /usr/local/share/nmap/nselib にそれぞれ入ることとなる。

NSE のオプション

ここからは NSE で使うことになるオプションについて説明していくことにする。

-sC オプション

最も標準的なスキャンを行う。--script default と同じ意味で、default カテゴリのスクリプトをすべて実行する。

--script オプション

カテゴリ、スクリプトを含んだディレクトリ、スクリプトファイルを 1 つまたは複数指定してスクリプトを実行する。複数指定したい場合はそれぞれをカンマで区切る。all を指定した場合はデータベースに登録されているすべてのスクリプトを実行する。

--script-arg オプション

スクリプトに引数を渡したい場合に使用する。

--script-trace オプション

スクリプトの実行をトレースする。

--script-updatedb オプション

データベースをアップデートする。スクリプトを追加・更新する際に実行する。

NSE のカテゴリ

NSE スクリプトは使用目的によってカテゴリに分かれており、1 つのスクリプトは複数のカテゴリに属すことができる。また自分で新しくカテゴリを定義することもできるため、目的別にカテゴリを作ることも可能だ。カテゴリごとにスクリプトを実行することにより、より柔軟にスキャニングできるようになる。

デフォルトのカテゴリ

```
auth/default/discovery/dos/exploit/
external/fuzzer/intrusive/malware/
safe/version/vuln
```

NSE を実際に使ってみよう

何はともあれ、実行してみなければ NSE がどういうものかわからない。まずはサンプルとして以下のコマンドを実行してみよう。example.com はダミーなので、各自の環境に置き換えてくれたまえ。

```
$ nmap -sC example.com
```

ここからは、この実行例（ページ下部参照）について解説していく。

①の部分が NSE スクリプトによって追加された部分だ。このように NSE を使うと通常の Nmap の出力に加えてより詳細な分析が行える。

default カテゴリ以外のスクリプトを実行したい場合は、--script オプションを使って --script vuln のように指定すればよい。複数のカテゴリを指定する場合はカンマで区切って --script vuln,malware のようにする。

また、スクリプトによっては引数をとることが可能だ。例えば

```
---script-args user="foo",pass=
"var",whois={whodb="nofollow"}
userdb="C:¥¥test"
```

のように、名前＝値のペアで指定する。個々の引数について知りたい場合はスクリプトに書いてある説明を読むか、Nmap のサイトから調べるといいだろう。

では最後に

```
nmap --script all example.com
```

を実行してみよう。もちろん example.com は各自の環境に置き換える必要がある。

実行が終了するまで少し時間がかかるかもしれないが、結果はどうなっただろう？　先ほどよりさらに多くの情報が出ているのではないだろうか。

筆者の環境でもデフォルトのままサービスをいろいろ立ち上げてスキャンしたところ、たくさんの情報が流れ出た。このように NSE を使うことで、手軽に多くの情報を取得できることがわかっていただけたかと思う。

NSEの実行例

```
$ nmap -sC example.com

Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-23 21:32 JST
Interesting ports on www.example.com (192.168.1.2):
Not shown: 996 filtered ports
PORT     STATE  SERVICE
53/tcp   closed domain
80/tcp   open   http
|  robots.txt: has 1 disallowed entry
|_ /
|_ html-title: Example
1863/tcp open   msnp
5190/tcp open   aol

Nmap done: 1 IP address (1 host up) scanned in 2.85 seconds
```

①

Lua 言語入門

さてここからは NSE のスクリプトについて解説しようと思う。この NSE のスクリプトは Lua という言語で書かれているので、まずは Lua 言語について少し学んでおこう。

識別子

まず、Lua の識別子は他の言語と同じようにアルファベットか下線で始まり、1 文字目が数字でないものだ。大文字と小文字は区別するので注意しよう。エスケープシーケンスの表記は「¥n」のようにおなじみの形式である。

文字列

文字列は「'」か「"」で囲む。文字列を [[]] で囲んだ場合は、複数行に渡って書くことができるが、エスケープシーケンスは解釈されずそのまま使われる。

演算子と制御構造

演算子と制御構造は各言語とほぼ同じような動作なので列挙に留める。

代入演算子	=
四則演算	+ - * /
マイナス演算子	-
剰余	%
べき乗	^
関係演算子	< > <= >= == ~=
論理演算子	and or not
連結演算子	..

※ ~= は != に対応する。
※連結演算子は文字列を連結する

制御構造

制御構造に関しては、以下のような形式が使われる。

```
if exp1 then block {elseif exp2
then block} [else block] end
while exp do block end
repeat block until end
for var=exp1,exp2,exp3 do block end
```

if と while と for は C 言語と同じである。repeat は do while と同じように block 内を最低 1 回は実行する。break 文は現在のループの実行を終了し、次のループに制御を移す。なお、continue 文は存在しない。

関数の定義

関数は
function f (args) block end または、f = function (args) block end
のような形式で定義する。戻り値を使いたい場合は return value として使う。

なお、Lua 言語についてのより詳細な解説は公式サイト（英語）から確認してほしい。
http://www.lua.org/

NSE スクリプトを読んでみる

では Lua 言語を簡単に学んだので、実際にスクリプトがどのような処理を行っているか次ページのサンプルスクリプトを追ってみることにしよう。サンプルスクリプトに書かれた記号はそれぞれの解説に対応している。

NSE スクリプトは 5 つのフィールドと 1 つのルールセクションと 1 つのアクションから成り立っている。

●フィールド

それぞれのフィールドにはスクリプトの情報が

NSEサンプルスクリプト

```
description = [[
Checks if an FTP server allows anonymous logins.                    (a)
]]

---
-- @output
-- |_ ftp-anon: Anonymous FTP login allowed

author = "Eddie Bell <ejlbell@gmail.com>"                           (b)
license = "Same as Nmap--See http://nmap.org/book/man-legal.html"   (c)
categories = {"default", "auth", "safe"}                            (d)

require "shortport"                                                 ①

portrule = shortport.port_or_service(21,"ftp")                      ②

--- Connects to the FTP server and checks if the server allows anonymous logins.
action = function(host, port)
        local socket = nmap.new_socket()                            ③
        local result
        local status = true
        local isAnon = false

        local err_catch = function()
                socket:close()
        end

        local try = nmap.new_try(err_catch)                         ④

        socket:set_timeout(5000)                                    ⑤
        try(socket:connect(host.ip, port.number, port.protocol))    ⑥
        try(socket:send("USER anonymous¥r¥n"))                      ⑦
        try(socket:send("PASS IEUser@¥r¥n"))

        while status do                                             ⑧
                status, result = socket:receive_lines(1);
                if string.match(result, "^230") then
                        isAnon = true
                        break
                end
        end

        socket:close()                                              ⑨

        if(isAnon) then                                             ⑩
                return "Anonymous FTP login allowed"
        end
end
```

書かれる。author フィールドと license フィールドは省略可能だが、できるだけ省略しない方がいいだろう。

（a）description フィールド
スクリプトの説明が書かれる。

（b）author フィールド
スクリプトの制作者名や連絡先が書かれる。

（c）license フィールド
スクリプトのライセンス形態が書かれる。基本的に上記と同じものになる。

（d）categories フィールド
スクリプトが属するカテゴリが書かれる。case-insensitive で複数のカテゴリを指定する場合はカンマで区切って書かれることになる。

（※）dependencies フィールド
前ページのスクリプトでは省略されているのだが、もう 1 つ dependencies フィールドがある。
Version 5.10BETA2 までにあった runlevel フィールドに置き換わる物であり、このスクリプトより先に実行しておきたいスクリプトを指定するために使う。

●ルールセクション

ルールセクションは特定のサービスやホストに対してスクリプトの動作をスキップか実行するかを決める関数である。
スクリプトは host ルールと port ルールのどちらか 1 つのみを含み、それぞれメソッドに与えられる host の情報と port の情報に基づき戻り値を true か false で返し、true の時のみアクション関数が実行されることとなる。

port ルール（portrule）

port テーブルの情報（ポート番号やプロトコル）を受け取り、スクリプトを実行すべきか決定する。

host ルール（hostrule）

host テーブルの情報（ホスト名や IP アドレス）を受け取り、スクリプトを実行する。

●アクション関数

NSE スクリプトのメインとなる部分。この関数の戻り値は Nmap の出力として表示される。

① require は外部のライブラリを読み込む場合に使う。このスクリプトでは shortport という Nmap に付いてくるライブラリを読み込んでいる。

② shortport.port_or_service は引数に渡されるポートかサービス名に一致すると true を返す。ここではサービス名が ftp かポート 21 番のとき true を返す。true を返したときに以下の action ③ が実行される。

③ ここでソケットを作成している。

④ 例外が発生したときにソケットを閉じる例外ハンドラを作成している。以下にある try() で囲まれた関数は例外発生時ソケットを閉じるようになっている。

⑤ タイムアウトの時間をミリ秒で設定している。

⑥ ホストの IP アドレスとポート番号とプロトコルを指定してコネクションを築く。

⑦ FTP の USER コマンドと PASS コマンド使って anonymous アカウントでログインを試みる。

⑧ レスポンスコードをチェックし、230（ログイン状態）であるか確認する。ログインが成功していたなら、isAnon を true にして break を使いループから抜ける。

⑨ ソケットを使い終えたので閉じる。

⑩ anonymous アカウントでログインできたならここに書かれた文字列「Anonymous FTP login allowed」 を返す。ここで返された文字列が Nmap の結果に表示されることになる。

スクリプトの紹介

ここからはNmapにインストールされているスクリプトの一部を紹介していくことにする。スクリプトはNmap公式サイト※からすべて入手できるので、コードに興味のある人はダウンロードして読んでみるのもいいだろう。

banner.nse

TCPポートに接続してバナー情報を取得する。

```
$ nmap --script banner.nse example.com
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-23 20:28 JST
Interesting ports on example.com (192.168.1.2):
Not shown: 991 closed ports
PORT     STATE SERVICE22/tcp   open  ssh
|_ banner: SSH-2.0-OpenSSH_5.1
3306/tcp open  mysql
|_ banner: 4¥x00¥x00¥x00¥x0A5.1.38¥x00¥x18¥x00¥x00¥x00z;TunLmb¥x00¥xFF¥...

Nmap done: 1 IP address (1 host up) scanned in 15.83 seconds
```

ftp-anon.nse

FTPサーバーがanonymousアカウントでのログインを許可しているかを表示する。先ほど見たサンプルスクリプトはこれだ。

```
$ nmap --script ftp-anon.nse example.com
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-23 20:43 JST
Interesting ports on examplecom (192.168.1.2):
Not shown: 984 filtered ports
PORT      STATE  SERVICE
21/tcp    open   ftp
|_ ftp-anon: Anonymous FTP login allowed

Nmap done: 1 IP address (1 host up) scanned in 4.40 seconds
```

http-enum.nse

有名なWebアプリやサービスによって使われたディレクトリを列挙する。

```
$ nmap --script http-enum.nse example.com
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-23 21:18 JST
Interesting ports on example.com (192.168.1.2):
Not shown: 991 closed ports
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
|  http-enum:
|  /icons/ Icons and images

Nmap done: 1 IP address (1 host up) scanned in 5.32 seconds
```

※http://nmap.org/nsedoc/index.html

http-iis-webdav-vuln.nse

MS09-020の脆弱性がないか検証する。

```
$ nmap --script http-iis-webdav-vuln.nse  example.com
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-23 21:53 JST
Interesting ports on o242.office.exmedia.jp (192.168.1.2):
Not shown: 992 closed ports
PORT     STATE SERVICE
80/tcp open  http
|_ http-iis-webdav-vuln: WebDAV is ENABLED. Vulnerable folders discovered:
/secret, /webdav

Nmap done: 1 IP address (1 host up) scanned in 34.15 seconds
```

html-title.nse

htmlのタイトルを表示する。

```
$ nmap --script html-title.nse example.com -p 80
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-24 17:27 JST
Interesting ports on www.example.com (192.168.1.2):
Not shown: 991 closed ports
PORT   STATE SERVICE
80/tcp open  http
|_ html-title: Example Web Page

Nmap done: 1 IP address (1 host up) scanned in 2.96 seconds
```

robots.txt.nse

robots.txtで巡回をブロックされているフォルダを表示する。

```
$ nmap --script robots.txt.nse example.com -p 80
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-24 18:11 JST
Interesting ports on www.example.com (192.168.1.2):
Not shown: 992 closed ports
PORT   STATE SERVICE
80/tcp open  http
|  robots.txt: has 2 disallowed entry
|_ /images /test

Nmap done: 1 IP address (1 host up) scanned in 2.90 seconds
```

sniffer-detect.nse

ネットワークカードがプロミスキャスモードであるか調査する。

```
$ nmap --script smb-check-vulns.nse example.com
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-24 17:45 JST
Interesting ports on example.com (192.168.1.2):
Not shown: 997 closed ports

Host script results:
|_ sniffer-detect: Likely in promiscuous mode (tests: "11111111")

Nmap done: 1 IP address (1 host up) scanned in 2.54 seconds
```

smb-check-vulns.nse

MS08-067のexploit。サーバーが脆弱な場合落ちる可能性が非常に高いようなので、くれぐれも実行の際は気をつけてほしい。

```
$ nmap --script smb-check-vulns.nse example.com
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-24 16:41 JST
Interesting ports on www.example.com (192.168.1.2):
Not shown: 997 closed ports

Host script results:
|   smb-check-vulns:
|   MS08-067: FIXED
|   Conficker: Likely INFECTED
|   regsvc DoS: FIXED
|_  SMBv2 DoS (CVE-2009-3103): VULNERABLE

Nmap done: 1 IP address (1 host up) scanned in 2.54 seconds
```

ms-sql-info.nse

SQLの情報を取得する。

```
$ nmap --script mysql-info.nse example.com -p 3306
Starting Nmap 5.00 ( http://nmap.org ) at 2009-09-24 21:59 JST
Interesting ports on example.com (192.168.1.2):
PORT     STATE SERVICE
3306/tcp open  mysql
|   mysql-info: Protocol: 10
|   Version: 5.1.30
|   Thread ID: 15636192
|   Some Capabilities: Long Passwords, Connect with DB, Compress, ODBC, SSL,
Transactions, Secure Connection
|   Status: Autocommit
|_  Salt: ggO6^X{'J}[/v*fIv9V&

Nmap done: 1 IP address n(1 host up) scanned in 0.72 seconds
```

sql-injection.nse

SQLインジェクションができるか簡単に調査する。

http-passwd.nse

ディレクトリトラバーサルで/etc/passwdが表示されるか調査する。

p2p-conficker.nse

2009年4月頃に話題となったConficker.Cかそれより上のバージョンをP2P通信を元に検出する。

最後に

NSEを使えばNmapをより強力に使えることになる。しかし、日本語の情報もほとんどないのでまだまだ認知度が低いのではないだろうか。自分のパソコンから漏れている情報を詳細に調査したり、はたまたCTFのようなものにも役立つと思うので、趣味や仕事でぜひ使ってみてはいかがだろうか。

NSEについてより詳しく知りたい方は下記を参考にするといいだろう。

http://nmap.org/book/nse.html
http://nmap.org/nsedoc/index.html
Nmapのmanページ