

Artigo



Metasploit - parte III

Como exportar dados de escaneamento em formatos legíveis por geradores de relatórios é o tema deste mês.

Por Alexandre Borges

Terminamos a coluna anterior comentando a respeito da possibilidade de armazenar os resultados do nosso escaneamento na base de dados PostgreSQL do próprio Metasploit e, como o leitor pôde acompanhar, aprendemos que é muito simples apagar e recriar novas *workspaces* dentro do framework. Entretanto, não é difícil nos depararmos com a necessidade de armazenar os resultados de nosso escaneamento através do NMAP em outros locais e aplicativos que não seja o próprio Metasploit. Caso o leitor enfrente uma situação semelhante, a alternativa é salvar a saída do NMAP em XML.

Como ambiente de testes continuo usado a máquina virtual *Metasploitable2* e, por isto, podemos seguir executando:

```
# nmap -sS 10.10.3.131 -D 200.132.131.19, 192.158.1.231 -A -oX xml_report
```

Será gerado o relatório “xml_report” em XML contendo o resultado do escaneamento (usando “decoy” para confundir o alvo e fazer parecer com que os pacotes originem-se de endereços IP diferentes do real) o qual pode ser importado em qualquer ferramenta de geração de relatórios.

Acredito que seria muito frustrante acreditar que um framework excepcional como o Metasploit pudesse somente nos permitir escanear portas de outras máquinas através do NMAP (embora, na minha opinião, seja de longe o melhor scanner de todos). Para nossa sorte existem outras opções muito úteis não somente para verificação de portas assim como de aplicativos. Observe:

```
# msfconsole
```

```
msf > search portscan
```

Matching Modules

=====

Name	Disclosure Date	Rank	Description
auxiliary/scanner/http/wordpress_pingback_access		normal	Wordpress Pingback Locator
auxiliary/scanner/natpmp/natpmp_portscan		normal	NAT-PMP External Port Scanner
auxiliary/scanner/portscan/ack		normal	TCP ACK Firewall Scanner
auxiliary/scanner/portscan/ftpbounce		normal	FTP Bounce Port Scanner
auxiliary/scanner/portscan/syn		normal	TCP SYN Port Scanner
auxiliary/scanner/portscan/tcp		normal	TCP Port Scanner
auxiliary/scanner/portscan/xmas		normal	TCP "XMas" Port Scanner

A saída nos mostra que existem diversos tipos de scanners de porta que podemos usar além do tradicional NMAP. Vamos a um exemplo:

```
msf > use auxiliary/scanner/portscan/tcp
```

```
msf auxiliary(tcp) > show options
```

Module options (*auxiliary/scanner/portscan/tcp*):

Name	Current Setting	Required	Description
CONCURRENCY	10	yes	The number of concurrent ports to check per host
PORTS	1-10000	yes	Ports to scan (e.g. 22-25,80,110-900)
RHOSTS		yes	The target address range or CIDR identifier
THREADS	1	yes	The number of concurrent threads
TIMEOUT	1000	yes	The socket connect timeout in milliseconds

O que nós fizemo ? Usamos o comando `use` para escolher o módulo de escaneamento mais conveniente para nossas necessidades (neste caso a técnica escolhida foi a mais simples disponível que é justamente o “*TCP scan*” que realiza todo o processo de *handshake*) e, em seguida, avaliamos quais opções são necessárias configurar sendo que as principais são aquelas que não possui valor padrão. Assim, o único parâmetro que não possui um valor padrão é justamente o IP da máquina que desejamos escanear, então executamos:

```
msf auxiliary(tcp) > set RHOSTS 10.10.3.131
RHOSTS => 10.10.3.131
msf auxiliary(tcp) > set PORTS 1-500
PORTS => 1-500
```

É claro que poderíamos configurar o número máximo de portas escaneadas simultaneamente (CONCURRENCY – deixamos o valor padrão), o intervalo de portas (PORTS – alteramos para o intervalo 1-500) e o número de threads realizando o escaneamento (THREADS – mantivemos este valor intocado) que nos permitiria verificar diversas máquinas em simultâneo. Assim:

```
msf auxiliary(tcp) > run

[*] 10.10.3.131:25 - TCP OPEN
[*] 10.10.3.131:23 - TCP OPEN
[*] 10.10.3.131:22 - TCP OPEN
[*] 10.10.3.131:21 - TCP OPEN
[*] 10.10.3.131:53 - TCP OPEN
[*] 10.10.3.131:80 - TCP OPEN
[*] 10.10.3.131:111 - TCP OPEN
[*] 10.10.3.131:139 - TCP OPEN
[*] 10.10.3.131:445 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) >
```

observe que existem muitas portas abertas embora tenhamos feito uso de um intervalo bem restrito de portas assim como também usamos o método mais simples de escaneamento (TCP). Aconselho que o leitor teste os outros métodos de escaneamento assim como um intervalo mais amplo.

Vou continuar este assunto em breve.

Até mais.