Como funciona o *nmap*

Miguel Frade

Departamento de Engenharia Informática
Instituto Politécnico de Leiria

May 15, 2013

O que é o nmap?

- Ferramenta *open source* para exploração de redes de auditorias de segurança
- Usa pacotes IP *raw* para determinar:
  - que computadores estão disponíveis
  - que serviços (nomes e versões) estão disponíveis nesses computadores
  - que sistemas operativos e versão estão a correr
  - que tipo de *firewall* está em uso
  - ...
- Também é usada para tarefas comuns, como:
  - inventário de redes
  - gestão de calendários de atualizações de serviços
  - monitorização do tempo de serviço dos computadores e serviços (*uptime*)

Para que serve o nmap?

O `nmap` faz lista dos alvos verificados com alguma informação adicional (dependendo das opções usadas):

- Lista de serviços com o número do porto com os seguintes estados:
  - `open` - um serviço está à escuta e a receber ligações
  - `filtered` - algo está a bloquear o porto e o `nmap` não consegue dizer se o porto está aberto ou fechado
  - `closed` - não está nenhum serviço à escuta no porto em causa, mas pode vir a estar a qualquer momento
  - `unfiltered` - quando o serviço responde aos pedidos do `nmap`, mas mesmo assim não consegue determinar se está fechado ou aberto
  - também podem existir combinações de estados: `open|filtered` e `closed|filtered`
- Lista de protocolos suportados, com a opção `-sO` (em vez de lista de portos)
  - `TCP`, `UDP`, `ICMP`, `IGMP`, `...`

Como se usa o nmap?

Muitos usam o `nmap`, mas poucos conhecem o seu potencial

```
sudo nmap www.qualquercoisa.com
```

Neste caso o `nmap`

- decide qual o tipo de *scan* a fazer
- o formato da saída
- os portos de destino
- os portos e endereços IP de origem
- a temporização
- ...

Fases do *scanning*

Exemplo

```
sudo nmap 192.168.226.0/24
```

Existem duas fases:

1. `host descovery` - obter a lista de endereços IP dos computadores disponíveis
2. `port scan` - obter lista de portos disponíveis nos computadores encontrados
   - por omissão o `nmap` não tenta descobrir portos nos IPs onde não obteve resposta, por isso a 1ª fase é muito importante;
   - este comportamento pode ser alterado com `-PN` (*no ping*), mas é preciso usar com cuidado, porque vai demorar muito mais tempo;

Descobrir computadores ativos - *host descovery*

O `nmap` permite descobrir computadores ativos das seguintes formas:

- `TCP SYN`, opção `-PS<lista_de_portos>`
- `TCP ACK`, opção `-PA<lista_de_portos>`
- `UDP`, opção `-PU<lista_de_portos>`
- `ICMP Request/ping`, opção `-PE`
- `ICMP Timestamp Request`, opção `-PP`
- `ICMP Netmask Request`, opção `-PM`
- `ARP`, opção `-PR`
- `IP protocol`, opção `-PO<lista_de_protocolos>`

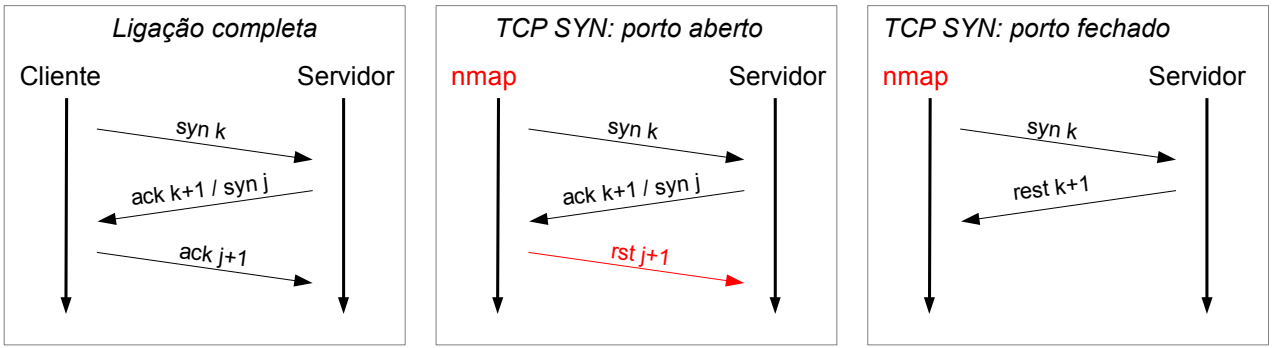
Para perceber o funcionamento temos de conhecer os protocolos

- `TCP`, `UDP`, `ICMP`, `IP` e `ARP`

Como funciona o TCP SYN scan

Opção `-PS<lista_de_portos>`

- Este método simula o estabelecimento de uma ligação `TCP`



- Se o porto estiver aberto recebemos o `SYN/ACK`
- Se o porto estiver fechado recebemos um `RST`, depois o kernel onde corre o `nmap` envia um `RST` (não é enviado pelo `nmap`)
- Se recebermos um `RST` ou um `SYN/ACK` significa que o computador está ligado

Como funciona o TCP SYN scan

- É necessário especificar portos `TCP`, usar os mais comuns
  - `22` (ssh), `25` (smtp), `80` (http), ...

Exemplo

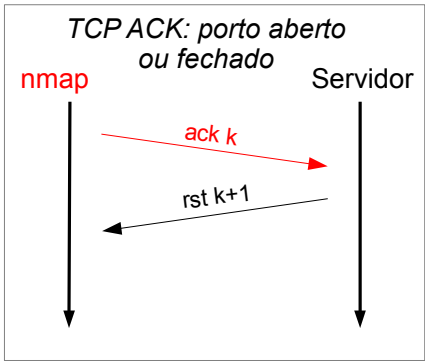
```
# -sP sem port scanning
sudo nmap -PS22,25,80 -sP 192.168.226.0/24

Nmap done: 256 IP addresses (12 hosts up) scanned in 8.14
    seconds
```

Como funciona o TCP ACK scan

Opção `-PA<lista_de_portos>`

- É semelhante ao `TCP SYN`



- Envia um `ACK` para uma ligação que não existe
- Neste caso o servidor deve enviar um `RST`, revelando a sua existencia
- O objetivo deste tipo de *scan* é maximizar a probabilidade de passar *firewalls*
  - *firewalls* mal configuradas impedem apenas a entrada de pacotes `SYN`, mas deixam passar outros
  - Este método não funciona em *firewalls stateful*

Como funciona o TCP ACK scan

- Também é preciso especificar portos, usar os mais comuns
- Para aumentar o sucesso deve-se usar os dois métodos, `SYN` e `ACK`, em simultâneo

Exemplo

```
# -sP sem port scanning
sudo nmap -PS22,25,80 -PA21,23,80,3389 -sP 192.168.226.0/24

Nmap done: 256 IP addresses (13 hosts up) scanned in 5.90
    seconds
```

- Neste caso permitiu descobrir mais um computador ativo do que só com o `TCP SYN`

Como funciona o UDP scan

Opção `-PU<lista_de_portos>`

- Envia pacotes `UDP` para os portos especificados
- Neste caso o comportamento é diferente:
  - se o pacote chegar a porto fechado, deve ser gerado um `ICMP port unreachable`
  - se receber um pacote `icmp host/network unreachable` ou `TTL exceeded` indica que provavelmente o computador está inacessível ou desligado
  - se o porto estiver aberto a maioria das vezes o pacote é ignorado e não há resposta

Como funciona o UDP scan

- neste caso devem ser usados portos invulgares, por omissão é usado o `40125`
- o porto `53` também pode ser interessante para este tipo de scan
- A vantagem deste tipo de *scan* é que permite passar *firewalls* mal configuradas que monitorizam apenas as ligações `TCP`

Exemplo

```
# -sP sem port scanning
sudo nmap -PU53,40125 -sP 192.168.226.0/24

Nmap done: 256 IP addresses (9 hosts up) scanned in 3.05
    seconds
```

Como funciona o ICMP scan

Opção `-PE`

- A forma mais simples de descobrir computadores é através do `ping`
- O `nmap` também suporta o mesmo tipo de pacotes: `icmp echo-request` e `icmp echo-reply`
- Mas atualmente a maioria dos computadores e firewalls bloqueiam estes pacotes, em vez de responderem como diz a RFC 1122
- Por isso, este tipo de scan não é confiável
- No entanto é útil para os administradores fazerem monitorização de LANs internas

Exemplo

```
# -sP sem port scanning
sudo nmap -PE -sP 192.168.226.0/24
```

Como funciona o ICMP scan

Opções `-PP` e `-PM`

- Quando o `ping` está bloqueado, temos como alternativas:
  - `-PP` usa pacotes `timestamp-request` e `timestamp-reply`
  - `-PM` usa pacotes `address-mask-request` e `address-mask-reply`
- Se for recebido um pacote `timestamp-reply` ou `address-mask-reply` mostra que o computador está ativo

Exemplo

```
# -sP sem port scanning
sudo nmap -PP -PM -sP 192.168.226.0/24
```

Como funciona o ARP scan

Opção `-PR`

- Este tipo de *scan* é usado para cerificar uma LAN *ethernet*
- Se fizermos um scan `icmp` o SO tem determinar primeiro o endereço MAC correspondente ao IP (pedido de `ARP`)
- Numa LAN *ethernet* este passo não é necessário, a resposta ao `ARP` já dá a informação que precisamos
- Este método é muito mais rápido e eficaz que o `-PE`, `-PP` e `-PM`
- Se o endereço do computador com o `nmap` é o mesmo da rede a inspecionar é usado o `-PR`, mesmo que seja especificado outro modo

Exemplo

```
# --send-eth envia pacotes raw em vez de usar o SO
sudo nmap -PR --send-eth --spoof_mac Apple -sP
    192.168.226.0/24
```

Como funciona o PROTOCOL scan

Opção `-PO<lista_de_protocolos>`

- Envia pacotes IP com o protocolo especificado na lista
- Por omissão usa os protocolos `icmp` (protocolo 1), `igmp` (protocolo 2) e `IP-in-IP` (protocolo 4)
- O computador alvo está ativo se:
  - Se receber um pacote com o protocolo igual ao enviado,
  - ou um pacote `icmp protocol-unreachable`

Exemplo

```
# -sP sem port scanning
sudo nmap -PO1,2,4,6 -sP 192.168.226.0/24
```

Scan intensivo

Exemplo scan combinando várias técnicas

```
sudo nmap -sP -PE -PP -PS21,22,23,25,80,113,21339
    -PA80,113,443,10042 --source_port 53 -n -T4
    192.168.226.0/24
```

- neste exemplo são enviados `13` pacotes sonda para cada IP — demora mais tempo
- `-sP` não determinar os portos abertos, apenas para descobrir computadores ativos
- `-n` não fazer resolução de nomes, permite aumentar o desempenho do `nmap`
- `-T4` o tempo entre envio de pacotes sonda é de 10 ms, envia pacotes em paralelo
- `--source_port 53` usar o porto de origem especificado, normalmente permitido pelas *firewalls*

Scan intensivo

Outras opções de temporização:

- `-T0` intervalo de 5 minutos, `-T1` 15 segundos
  - têm como objetivo evitar alarmes em IDS
  - não consumir a largura de banda e evitar crashes dos alvos
  - mas é mais eficaz não fazer a deteção da versão do que usar `-T0` ou `-T1`
- `-T2` 0.4 segundos de intervalo, `-T3` é o valor pré-definido
- `-T4` é o valor recomendado pelo autor do `nmap`, usa paralelização
- `-T5` 5 ms de intervalo, usa paralelização

Descoberta de portos – *port scan*

O `nmap` suporta os seguintes modos:

- `TCP SYN`, opção `-sS`
- `TCP connect`, opção `-sT`
- `UDP`, opção `-sU`
- `TCP NULL`, `FIN` e `Xmas`, opções `-sN`, `-sF` e `-sX`
- `TCP ACK`, opção `-sA`
- `TCP Window`, opção `-sW`
- `TCP Maimon`, opção `-sM`
- `TCP Idle`, opção `-sI`
- `IP protocol`, opção `-sO`

Como funciona a descoberta de portos

- São enviados pacotes sonda a vários portos (`1000` por omissão)
- Com a opção `-p<lista_potos>` podemos especificar os portos desejados
  - `-p22`, `-p1-1024`, `-p U:53,111,137,T:21-25,80,139,8080`
  - `-p-` serve para percorrer todos os portos [1; 65535]
- o tipo de scan pré-definido é `TCP SYN` (`-sS`)
- não se pode fazer mais do que um tipo de scan para o mesmo protocolo
  - `-sS -sA` não é permitido, são dois scans `TCP`
  - `-sS -sU` é permitido, um é para `TCP` e o outro para `UDP`
- `-sP` procura apenas computadores ativos, não tenta detetar os portos abertos/fechados
- `-sO` não verifica portos, mas protolos IP, pode usar a opção `-p` para especificar os protocolos

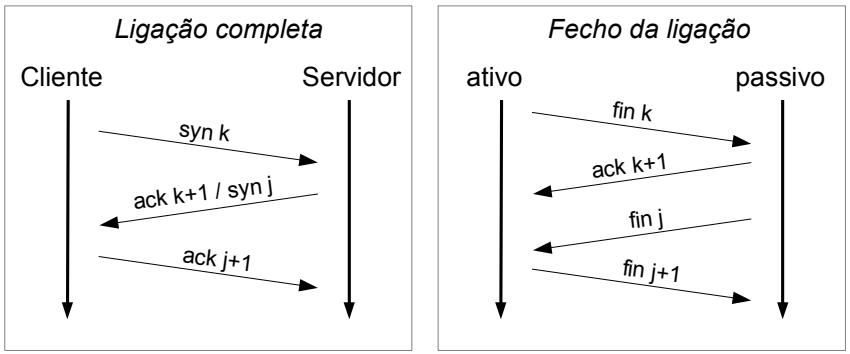
Mesmo princípio de funcionamento

- Modos de scan com o mesmo princípio de funcionamento à descoberta de computadores
- A diferença é que se pode percorrer uma gama alargada de portos, em vez de apenas determinar se está activo ou não
  - `TCP SYN` $\longrightarrow$ `-sS` $\approx$ `-PS`
  - `TCP ACK` $\longrightarrow$ `-sA` $\approx$ `-PA`
  - `UDP` $\longrightarrow$ `-sU` $\approx$ `-PU`
  - `IP protocol` $\longrightarrow$ `-sO` $\approx$ `-PO`
- `TCP SYN` é o modo pré-definido, pode verificar milhares de portas por segundo

Como funciona o *TCP connect*

O `TCP connect (-sT)`

- faz uma ligação completa e depois tem de a terminar



- depende do SO e é lento
- é fácilmente detetável
- é a única opção para utilizadores sem previlégios de administração

Como funciona o *TCP NULL, FIN e Xmas*

Os modos `TCP NULL` (`-sN`), `FIN` (`-sF`) e `Xmas` (`-sX`)

- têm todos o mesmo comportamento
  - se receber um pacote `RST` → porto fechado
  - se não houver resposta → porto aberto ou filtrado
  - se receber um `ICMP unreachable` → porto filtrado
  - problema: nem todos os SO implementam à risca a `RFC 793`
- mas mudam as *flags* `TCP` usadas
  - `-sN` → não ativa nenhuma *flag*
  - `-sF` → ativa apenas a *flag* `FIN`
  - `-sX` → ativas as *flags* `FIN`, `PSH` e `URG` como se fosse uma árvore de Natal
- este tipo de scan consegue passar por algumas *firewalls stateless*

Como funciona o *TCP Window*

O `TCP window` (`-sW`)

- é semelhante ao `ACK` scan, mas explora um pormenor de implementação de alguns sistemas
- inspeciona o campo *window* do cabeçalho TCP
  - `RST` com $window = 0 \rightarrow$ porto fechado
  - `RST` com $window > 0 \rightarrow$ porto aberto
- depois deve-se comparar com os resultados anteriores

Como funciona o *TCP Maimon*

`TCP Maimon`, opção `-sM`

- técnica semelhante ao `TCP NULL`, `FIN` e `Xmas`
- a diferença está nos pacotes sonda que têm as flags `FIN` e `ACK` ativas
- de acordo com a `RFC 793` deve ser gerada uma resposta `RST` quer o porto esteja aberto, ou fechado
- no entanto alguns sistemas `BSD` não respodem quando o porto está aberto

Como funciona o *TCP Idle*

O `TCP Idle`, opção `-sI zombie host[:probeport]`

- método de scan mais complexo e furtivo
- nos pacotes sonda o IP de origem é o de um zombie
- do ponto de vista dum IDS o scan é realizado pelo zombie
- método de funcionamento
  - baseia-se na previsão do número de sequencia da fragmentação IP no zombie
  - obter # ID do zombie → enviar pacote sonda → ler # ID do zombie
  - se # ID incrementou 2 unidades → porto aberto
  - se # ID incrementou 1 unidade → porto fechado ou filtrado

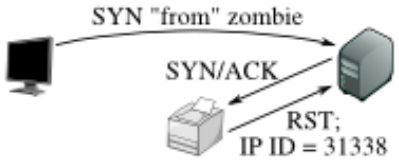
Como funciona o *TCP Idle*

Porto aberto

Step 1: Probe the zombie's IP ID.

The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID.

Step 2: Forge a SYN packet from the zombie.



The target sends a SYN/ACK in response to the SYN that appears to come from the zombie. The zombie, not expecting it, sends back a RST, incrementing its IP ID in the process.

Step 3: Probe the zombie's IP ID again.

The zombie's IP ID has increased by 2 since step 1, so the port is open!

Fonte: `http://nmap.org/book/idlescan.html`

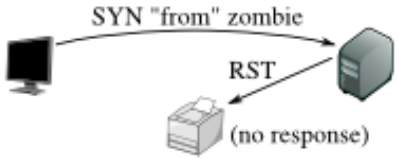
Como funciona o *TCP Idle*

Porto fechado

Step 1: Probe the zombie's IP ID.

The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID. This step is always the same.

Step 2: Forge a SYN packet from the zombie.



The target sends a RST (the port is closed) in response to the SYN that appears to come from the zombie. The zombie ignores the unsolicited RST, leaving its IP ID unchanged.

Step 3: Probe the zombie's IP ID again.

The zombie's IP ID has increased by only 1 since step 1, so the port is not open.

Fonte: `http://nmap.org/book/idlescan.html`

Como funciona o *TCP Idle*

Porto filtrado

Step 1: Probe the zombie's IP ID.

Just as in the other two cases, the attacker sends a SYN/ACK to the zombie. The zombie discloses its IP ID.

Step 2: Forge a SYN packet from the zombie.

The target, obstinately filtering its port, ignores the SYN that appears to come from the zombie. The zombie, unaware that anything has happened, does not increment its IP ID.

Step 3: Probe the zombie's IP ID again.

The zombie's IP ID has increased by only 1 since step 1, so the port is not open. From the attacker's point of view this filtered port is indistinguishable from a closed port.

Fonte: `http://nmap.org/book/idlescan.html`

Descoberta de um único serviço

Objetivo:

- por exemplo, descobrir servidores `HTTP`

Comando

```
sudo nmap -PO -p80 -oG gnmap.txt 192.168.226.0/24
```

- `-PO` faz o scan de protocolos porque para 1 só serviço é mais rápido
- `-p80` procura o porto do `HTTP`
- `-oG gnmap.txt` grava no ficheiro num formato pesquisável com o `grep`

Descoberta de um único serviço

Objetivo: melhorar o desempenho do comando anterior

Comando

```
sudo nmap -n -T4 --max_rtt_timeout 200
    --initial_rtt_timeout 150 -P0 -p80 -oG gnmap.txt
    192.168.226.0/24
```

- `-n` evita fazer a resolução de nomes porque é lento (ativo por omissão)
- `--max_rtt_timeout 200 --initial_rtt_timeout 150` opções avançadas de temporização
  - na Internet a maioria dos pacotes são filtrados
  - o `nmap` precisa de ajuda para melhorar a temporização
  - obter valores através do `ping` ou `hping`, usar os valores mais elevados

Descoberta de um único serviço

Objetivo: verificar as versões dos serviços

Comando

```
sudo nmap -n -T4 -P0 -p80 -sV -oG gnmap.txt 192.168.226.3
```

- serve para verificar se os serviços estão atualizados
  - o `nmap` tem mais de 1000 assinaturas de serviços
- permite detetar eventuais `backdoors`
  - excelente para identificar computadores infetados
  - por exemplo o `mydoom`
- suscetível de gerar alarmes IDS

Descobrir portos abertos – SYN scan

Começar com as opções habituais

```
sudo nmap -sS -T4 192.168.226.3

(The 998 ports scanned but not shown below are in state:
    filtered)
```

Se houver portos `filtered`, tentar outros tipos de scan:

- `-sF` envia pacotes `FIN` (não funciona numa *firewall stateful*)
  - se receber um `RST` o porto está fechado
  - se não receber nada o porto está aberto ou filtrado

```
sudo nmap -sF -T4 192.168.226.3
(The 997 ports scanned but not shown below are in state: closed)
PORT      STATE         SERVICE
9/tcp     open|filtered discard
11/tcp    open|filtered systat
13/tcp    open|filtered daytime
```

Descobrir portos abertos – FIN scan

- O estado `open|filtered` não é conclusivo
- Mas o resto dos portos estão fechados `closed`
- intersetar os resultados dos scan anteriores
  - `-sS` 2 portos abertos, 998 filtrados
  - `-sF` 3 portos abertos—filtrados, 997 fechados
  - conclusão: dos 998 filtrados, 997 estão efetivamente fechados
- Tentar outro tipo de scan: `-sA`
  - envia um pacote `ACK`
  - os portos abertos e fechados enviam um pacote `RST`, neste caso não está filtrado (`unfiltered`)
  - se não receber nada, ou receber um `icmp destination-unreachable`, está filtrado
  - este scan não determina se os portos estão abertos ou fechados
  - serve apenas para determinar o tipo de *firewall stateless* ou *stateful* e para verificar as regras configuradas

Descobrir portos abertos – ACK scan

```
sudo nmap -sA -T4 192.168.226.3
(The 998 ports scanned but not shown below are in state:
    UNfiltered)
PORT     STATE    SERVICE
135/tcp  filtered msrpc
1434/tcp filtered ms-sql-m
```

- Intersetar os resultados dos scan anteriores, se para um porto
  - um scan diz `filter` e outro diz `open|filter` → o porto está filtrado
  - um scan diz `unfiltered` e outro diz `open|filter` → o porto está aberto

Comparar resultados

Para comparar resultados usar o `ndiff`

- recebe 2 ficheiros `xml` com os resultados do `nmap`
- e mostra as diferenças de estado
  - dos computadores
  - dos portos
  - das versões dos serviços
  - comparações de identificação do SO

Como usar o *ndiff*

Fazer o scan e guardar o XML

```
sudo nmap -n -PO -sS -oX sS.xml 192.168.226.3
sudo nmap -n -PO -sF -oX sF.xml 192.168.226.3
```

Comparar

```
ndiff sS.xml sF.xml
-Nmap 5.21 at 2013-05-08 16:14
+Nmap 5.21 at 2013-05-08 16:15

192.168.226.3:
PORT     STATE          SERVICE      VERSION
-22/tcp   open           ssh
+22/tcp   open|filtered ssh
-53/tcp   open           domain
+53/tcp   open|filtered domain
(...)
```

Bibliografia

1. `man nmap`
2. `man nmap`
3. `man nmap`
4. `http://nmap.org/book/toc.html`

Demonstração do uso do `nmap` para verificar a configuração de uma *firewall iptables*: ▸ Youtube: Firewall IPTABLES (parte 4)