# Skanowanie sieci

Daniel Adamski – danadam@mars.iti.pk.edu.pl

## Spis Treści



## 1. Wstęp teoretyczny

Skanowanie komputera wykonuje się aby zebrać jak najwięcej informacji o usługach przez niego oferowanych. Skanowanie sieci ujawni nam jej strukturę. Wszystkie te informacje ułatwiają ewentualny atak tym złym hakerom i administrację tym dobrym administratorom :)

Skanowanie polega na wyszukaniu otwartych portów, można też ustalić jakie oprogramowanie nasłuchuje na tych portach. Posiadając tą wiedzę atakujący może wykorzystać błędy danego oprogramowania.

Jest wiele sposobów skanowania, poniżej omówię kilka z nich. W kilku przypadkach wspomnę o wpływie firewalla na wyniki skanowania. W dalszej części opiszę kilka metod ukrycia faktu skanowania. W przypadku skanowania ważny jest nie tylko fakt uzyskania informacji. Do niczego się

ona nie przyda jeśli administratorzy skanowanej sieci natychmiast dowiedzą się o tym, że ktoś próbuje się włamać. Dlatego należy pamiętać o istnieniu systemów IDS, które na szczęście (zależy dla kogo) mają wady, umożliwiające przeprowadzenie skanowania bez zwracania na siebie uwagi.

## 1.1. Metody skanowania

### 1.1.1. TCP Connect()

Najbardziej podstawowa metoda skanowania. Wykorzystuje się systemową funkcję connect(), aby połączyć się z interesującym nas portem. Jeśli port nasłuchuje, to wywołanie funkcji powiedzie się. Największa zaleta tego sposobu to taka, że nie trzeba posiadać żadnych specjalnych uprawnień. Każdy użytkownik może wykorzystać tą funkcję. Niestety metoda ma też dużą wadę – jest łatwo wykrywalna. W logach skanowanego hosta pojawi się wiele wpisów o nawiązaniu połączenia i jego natychmiastowym zerwaniu.

### 1.1.2. TCP SYN

Ta metoda często jest nazywana półotwarciem (ang. half-open). Nazwa wzięła się stąd, że przy otwieraniu połączenia nie wykonujemy wszystkich trzech kroków handshakingu. Wysyłamy segment z ustawionym bitem SYN i oczekujemy na odpowiedź. Jeśli otrzymamy segment z ustawionymi bitami SYN i ACK to port jest otwarty. Segment z ustawionym bitem RST wskazuje na port zamknięty. W normalnym przypadku odesłalibyśmy segment z ustawionymi bitami SYN i ACK (oczywiście jeśli port okazał się otwarty), jednak w tej metodzie zrywamy połączenie odsyłając segment z ustawionym bitem RST.

Zaletą tej metody jest mniejsza wykrywalność niż sposobu TCP connect(). Można przyjąć, że większość hostów loguje tylko w pełni otwarte połączenia, więc takie półotwarte powinno przejść niezauważone. Niestety, aby skorzystać z tej metody należy posiadać uprawnienia roota.

### 1.1.3. FIN, Xmas Tree, Null

Czasami nawet skanowanie metodą TCP SYN nie wystarcza. Niektóre zapory i filtry pakietów śledzą pakiety SYN, a niektóre programy potrafią wykryć takie skany. W takiej sytuacji można skorzystać ze skanowania metodą FIN, Xmas Tree lub Null.

Do skanowania używa się:

- segmentów z ustawionym bitem FIN
- segmentów z ustawionymi bitami FIN, URG i PSH
- segmentów z wyzerowanymi bitami

Zgodnie ze standardem (RFC 793) zamknięte porty powinny odpowiedzieć na takie dane segmentem z ustawionym bitem RST, natomiast otwarte porty powinny zignorować takie dane. Trzeba jednak wziąć pod uwagę, że nie wszystkie systemy zachowują się zgodnie ze standardem (m.in. Windows, Cisco, HP/UX). Odsyłają one segment z ustawionym bitem RST niezależnie od stanu portu.

### 1.1.4. Ping, ACK:

Czasami chcemy się dowiedzieć, które hosty są uruchomione. Można to wykonać za pomocą pingowania. Niestety może się zdażyć, że pakiety ICMP echo request i ICMP echo reply będą blokowane przez firewall. W takim wypadku istnieje druga możliwość, a mianowicie wysłanie segmentu TCP z ustawionym bitem ACK. Jeśli w odpowiedzi otrzymamy segment z ustawionym bitem RST, będzie to oznaczało, że skanowany host jest uruchomiony.

Skanowanie ACK pozwoli nam również określić z jakim typem zapory ogniowej mamy do czynienia (ze śledzeniem stanu czy bez). Przypuśćmy, że na skanowanie TCP SYN nie dostaliśmy żadnej odpowiedzi. Oznacza to, że docelowy komputer jest wyłączony lub firewall odfiltrował nasz segment SYN. Następnie wykonujemy skanowanie TCP ACK. Załóżmy, że tym razem otrzymamy w odpowiedzi segment TCP RST. Na tej podstawie możemy stwierdzić, że firewall, który stoi na naszej drodze nie śledzi stanów połączeń. W przeciwnym razie nie przepuściłby segmentu TCP ACK, które nie należy do żadnego połączenia. Otrzymujemy dwie informacje: o rodzaju firewalla i o tym, że docelowy komputer jest uruchomiony. Jeśli firewall nie śledzi stanów połączeń to możemy użyć skanowania FIN, Xmas lub Null, aby wyszukać otwarte porty.

### 1.1.5. UDP:

Niektórzy uważają, że skanowanie portów UDP jest bezsensowne. Zapominają oni, że wiele usług podatnych na ataki używa właśnie portów UDP. Niektóre z nich to: snmp, tftp, NFS, itd.

Skanowanie polega na wysłanie datagramu UDP bez żadnych danych. Jeśli w odpowiedzi otrzymamy informację ICMP port unreachable, wtedy port jest zamknięty. W przeciwnym wypadku jest otwarty lub odfiltrowany przez firewall. Niestety skanowanie portów UDP może być wolne. Większość systemów stosuje się do sugestii zawartej w RFC 1812 dotyczącej ograniczania ilości wysyłanych wiadomości ICMP. Na przykład jądro Linuksa nie wygeneruje więcej wiadomości ICMP port unreachable niż 80 na 4 sekundy. Dodatkowo nastąpi przerwa trwająca 1/4 sekundy jeśli ta wartość zostanie przekroczona. System Solaris jest jeszcze bardziej surowy. Jednym z systemów, który nie stosuje się do tej sugestii to Windows, więc można przeskanować wszystkie porty w krótkim czasie.

### 1.1.6. Idlescan, FTP bounce:

Powyższe metody mają tą wadę, że w wysyłanych pakietach znajduje się nasz adres IP. Pozwala to łatwo wyśledzić osobę przeprowadzającą skanowanie. Poniżej przedstawię dwa sposoby skanowania, które bardzo utrudniają wykrycie osoby przeprowadzającej skanowanie, nawet jeśli sam fakt skanowania zostanie zauważony.

Metoda Idlescan wykorzystuje pewną niedoróbkę w stosach tcp. Teoretycznie numery ID pakietów IP powinny być generowane losowo. W większości przypadków tak nie jest, a numer ID jest zwiększany o 1 dla każdego kolejnego pakietu IP. Procedura skanowania wygląda następująco:

- znajdujemy komputer, który nie generuje ruchu
- co kilka minut wysyłamy do niego pakiet ICMP echo request i sprawdzamy numer ID pakietu ICMP echo reply (dzięki temu dowiemy się, czy stos tcp na tym hoście nadaje się do pośredniczenia w ataku). Załóżmy, że kolejne pakiety ICMP echo reply mają numer ID zwiększany o 1.
- wysyłamy segment TCP SYN do komputera, który chcemy skanować z adresem źródłowym ustawionym na host pośredniczący
- jeśli port na skanowanym hoście jest otwarty, to odeśle on (do hosta pośredniczącego) segment TCP SYN,ACK. Host pośredniczący nie otwierał żadnego połączenia, więc prześle do skanowanego hosta segment TCP RST.
- jeśli port na skanowanym hoście jest zamknięty, to odeśle on (do hosta pośredniczącego) segment TCP RST. Host pośredniczący zignoruje go
- teraz my wysłamy jeszcze raz pakiet ICMP echo request do host pośredniczącego i sprawdzamy numer ID pakietu otrzymanego w odpowiedzi. Jeśli zwiększył się o 1 od ostatniego razu, to wnioskujemy, że w między czasie nic nie wysyłał. Oznacza to, że port w skanowanym hoście jest zamknięty. Jeśli numer ID zwiększył się o 2, to znaczy że host pośredniczący przesłał w między

czasie jeden pakiet. Oznacza to, że port w skanowanym hoście jest otwarty.
Zaletą tej metody jest to, że trudno ustalić kto tak naprawdę wykonuje skanowanie. Poza tym host pośredniczący może być bardziej zaufany dla skanowanego hosta lub firewalla po drodze, niż my.

Metoda FTP bounce wykorzystuje pewną "cechę" protokołu FTP, która umożliwia przekazywanie wyników do innej stacji niż ta, która zainicjowała sesję. Kod jaki zwraca taka operacja mówi nam czy port na tej innej stacji jest otwarty czy zamknięty. Aby włączyć przekierowanie, po zalogowaniiu się na serwer FTP, wykonujemy komendę:

```
PORT A,B,C,D,X,Y
```

gdzie A.B.C.D to adres IP skanowanego hosta, a X*256+Y to port, który chcemy sprawdzić. Jeśli serwer FTP umożliwia przekierowanie to zwróci kod 200, w przeciwnym razie 500. Teraz wykonujemy jakąkolwiek komendę przesyłającą dane. Trafi ona do hosta o adresie A.B.C.D na port X*256+Y. Jeśli port jest otwarty to zwrócony zostanie kod 150 i 226, w przeciwnym razie 426.

## 1.2. Ukrywanie skanowania

Istnieje kilka sposobów ukrycia faktu skanowania. Wykorzystuje się przy tym niedoskonałości firewalli i systemów IDS. Systemy bezpieczeństwa nie mogą dokładnie sprawdzać każdego pakietu, gdyż wymagałoby to bardzo dużej mocy obliczeniowej lub wprowadzało zbyt duże opóźnienia w ruchu.

### 1.2.1. Wabik

Podczas skanowania możemy generować wiele pakietów z podmienionym adresem źródłowym. Adresy które wstawiamy powinny należeć do aktywnych hostów. Utrudni to wyśledzenie inicjatora skanowania. Poza tym systemy IDS mogą śledzić tylko ograniczoną ilość połączeń, więc nasze może przejść niezauważone.

### 1.2.2. Fragmentacja, powolne skanowanie

Również z powodów wydajnościowych wiele systemów nie śledzi pofragmentowanych pakietów. Możemy to wykorzystać i specjalnie je pofragmentować.
Systemy bezpieczeństwa podnoszą alarm, gdy liczba podejrzanych pakietów przekroczy ustaloną graniczną wartość. Jeśli będziemy wysyłać pakiety skanujące w długich odstępach czasu, to nie wzbudzimy niczyich podejrzeń.

### 1.2.3. Numery segmentów TCP

Ze względów wydajnościowych systemy często nie sprawdzają poprawności numerów segmentów TCP. Zwracają uwagę tylko na ustawione bity. Jeśli wyślemy segment z bitami FIN lub RST i nieprawidłowym numerem segmentu, to system IDS będzie uważał, że połączenie zostało zakończone i przestanie je śledzić. Jednak w rzeczywistości skanowany host odrzuci taki segment (z powodu jego złego numeru) i połączenie będzie nadal otwarte.

### 1.2.4. Suma kontrolna

Jeśli zdarzy się, że mamy do czynienia z dobrym systemem IDS, który sprawdza poprawność

numerów segmentów TCP, to możemy spróbować zabaw z sumą kontrolną. Możliwe, że nie jest ona sprawdzana. W środku połączenia wysyłamy segmenty z poprawnym numerem, z ustawionymi bitami FIN lub RST i ze złą sumą kontrolną. Podobnie jak wyżej, dla systemu IDS będzie to zakończenie połączenia, podczas gdy skanowany host odrzuci ten segment i będzie kontunuował połączenie.

#### 1.2.5. TTL

Również modyfikacje pola TTL mogą nam się przydać. Szczególnie w przypadku, gdy system IDS jest postawiony na routerze lub przed nim. Po otwarciu połączenia wysyłamy segment TCP z ustawionymi bitami FIN lub RST i tak dobraną wartością TTL, żeby dotarł do systemu IDS ale do skanowanego hosta już nie. W wyniku dostaniemy sytuację analogiczną jak poprzednich dwóch przypadkach.

# 2. Labolatorium

## 2.1. Zadanie nr 1 – nmap ping

W wyniku skanowania metodą ping otrzymałem:

```
[root@stanowisko5 root]# nmap -sP 192.168.202.1-254

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.202.1) appears to be up.
Host  (192.168.202.2) appears to be up.
Host  (192.168.202.4) appears to be up.
Host stanowisko5 (192.168.202.5) appears to be up.
Host  (192.168.202.7) appears to be up.
Host  (192.168.202.11) appears to be up.
Host  (192.168.202.12) appears to be up.
Host  (192.168.202.13) appears to be up.
Host  (192.168.202.14) appears to be up.
Host  (192.168.202.15) appears to be up.
Host  (192.168.202.102) appears to be up.
Host  (192.168.202.107) appears to be up.
Host  (192.168.202.250) appears to be up.
Host  (192.168.202.254) appears to be up.
Nmap run completed -- 254 IP addresses (14 hosts up) scanned in 8 seconds
```

Nmap wysłał pakiety ICMP echo request pod wszystkie adresy IP podane jako argument. Te hosty, które odpowiedziały zostały wyświetlone.

## 2.2. Zadanie nr 2 – nmap skanowanie

Wykonałem skanowanie localhosta metodami: connect(), z podaniem zakresu portów, TCP SYN. Wyniki wyświetlone w każdym z tych wypadków były identyczne (z wyjątkiem informacji o liczbie przeskanowanych portów:

```
[root@stanowisko5 root]# nmap -sT 127.0.0.1
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
```

```
(The 1592 ports scanned but not shown below are in state: closed)

[root@stanowisko5 root]# nmap -p 1-32000 -sT 127.0.0.1
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 31991 ports scanned but not shown below are in state: closed)

[root@stanowisko5 root]# nmap -sS 127.0.0.1
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1592 ports scanned but not shown below are in state: closed)

Port       State       Service
22/tcp     open        ssh
23/tcp     open        telnet
25/tcp     open        smtp
80/tcp     open        http
111/tcp    open        sunrpc
1024/tcp   open        kdm
1025/tcp   open        NFS-or-IIS
3306/tcp   open        mysql
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds
```

Z listy nasłuchujących portów można wywnioskować, że mamy do czynienia z systemem Linuks (X11 nasłuchujący na porcie 6000 i kdm nasłuchujący na porcie 1024 wskazują na to dość jednoznacznie)

Następnie wykonałem skanowanie portów UDP:

```
[root@stanowisko5 root]# nmap -sU 127.0.0.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1461 ports scanned but not shown below are in state: closed)
Port       State       Service
68/udp     open        dhcpclient
111/udp    open        sunrpc
123/udp    open        ntp
137/udp    open        netbios-ns
138/udp    open        netbios-dgm
747/udp    open        fujitsu-dev
1024/udp   open        unknown
```

Możemy wywnioskować, że na localhoście jest uruchomiony m.in. serwer czasu (NTP – Network Time Protocol) i Samba.

Aby dowiedzieć się więcej o systemie uruchomionym na localhoście skorzystałem z parametru -O:

```
[root@stanowisko5 root]# nmap -O 127.0.0.1
```

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1592 ports scanned but not shown below are in state: closed)
Port       State       Service
[...]
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.010 days (since Thu Mar 31 07:39:06 2005)
```

W podobny sposób wykonałem skanowanie komputera z systemem Windows:

```
[root@stanowisko5 root]# nmap -sS 192.168.202.6
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.202.6):
(The 1598 ports scanned but not shown below are in state: closed)
Port       State       Service
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
1025/tcp   open        NFS-or-IIS

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds

[root@stanowisko5 root]# nmap -sU 192.168.202.6
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.202.6):
(The 1465 ports scanned but not shown below are in state: closed)
Port       State       Service
135/udp    open        loc-srv
137/udp    open        netbios-ns
138/udp    open        netbios-dgm

Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds
```

Na porcie 135 działa usługa Microsoft RPC Endpoint Mapper. W 2003 r w Microsoft RPC znaleziono poważne luki, które zostały wykorzystane przez wirusy Blaster oraz Nachi/Welchia. Port 135/UDP jest też często wykorzystywany do wysyłania komunikatów typu winpopup.

```
[root@stanowisko5 root]# nmap -O 192.168.202.6
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.202.6):
(The 1598 ports scanned but not shown below are in state: closed)
[...]
Remote operating system guess: Windows Millennium Edition (Me), Win 2000,
or WinXP

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

## 2.3. Zadanie nr 3 – Nessus

Skanowanie za pomocą programu Nessus wykryło następujące zagrożenia:

- telnet (23/tcp) (Security warnings found)
- ssh (22/tcp) (Security hole found)
- http (80/tcp) (Security hole found)
- msg (1241/tcp) (Security warnings found)
- mysql (3306/tcp) (Security warnings found)
- x11 (6000/tcp) (Security warnings found)
- unknown (1024/udp) (Security hole found)

Większość ostrzeżeń dotyczy zbyt starych wersji oprogramowania, które są podatne na ataki. Najwięcej z nich dotyczy portu 22 (nieaktualna wersja OpenSSH) i 80 (nieaktualna wersja PHP i Apache)

## 2.4. Zadanie nr 4 – Nessus diff scanned

W tym zadaniu miałem skorzystać z opcji diff scan Nessusa. Jej użycie powoduje, że raport ze skanowania zawiera tylko zmiany jakie zostały wykryte od ostatniego skanowania. Nessus wykorzystuje do tego tzw. Knowledge Base, gdzie pamięta wyniki poprzednich skanowań.

Ustawiłem opcje tak jak podano w instrukcji. Przeprowadziłem skanowanie, a następnie uruchomiłem dodatkowe usługi: vsftpd, named, nfs. Kolejne skanowanie wykryło następujące zagrożenia:

- ftp (21/tcp) (Security warnings found)
- domain (53/tcp) (Security hole found)
- unknown (810/udp) (Security warnings found)
- unknown (1691/udp) (Security warnings found)
- nfs (2049/udp) (Security warnings found)
- nfs (2049/tcp) (Security warnings found)
- netbios-ns (137/udp) (Security warnings found)
- unknown (1692/tcp) (Security hole found)

Jak widać nie ma już na przykład informacji o zagrożeniach na portach 22 (ssh), 80 (http) czy 3306 (mysql)

## 2.5. Zadanie nr 5 – Nessus opcje

Opcje Nessusa pozwalają na przeprowadzanie różnych rodzajów skanowań. Możliwe są wszystkie opisane we wstępie, a także kilka innych. Można również skonfigurować metody oszukiwania systemów IDS (również opisane we wstępie). Poniżej przedstawiony jest rezultat skanowania metodą ping i Xmas tree:

- telnet (23/tcp) (Security warnings found)
- ssh (22/tcp) (Security hole found)
- ftp (21/tcp) (Security warnings found)
- domain (53/tcp) (Security hole found)
- http (80/tcp) (Security hole found)
- sunrpc (111/tcp) (Security notes found)
- msg (1241/tcp) (Security warnings found)

- mysql (3306/tcp) (Security warnings found)
- x11 (6000/tcp) (Security warnings found)
- unknown (1024/udp) (Security hole found)
- NFS-or-IIS (1025/tcp) (Security warnings found)
- unknown (810/udp) (Security warnings found)
- unknown (1691/udp) (Security warnings found)
- nfs (2049/udp) (Security warnings found)
- nfs (2049/tcp) (Security warnings found)
- netbios-ns (137/udp) (Security warnings found)
- unknown (1692/tcp) (Security hole found)

# 3. Wnioski

- Instrukcje do ćwiczeń są w pełni zrozumiałe
- Labolatorium można wykonać w przeznaczonym na to czasie
- Na początku pojawił się problem z uaktualnianiem pluginów (były ściągane bardzo wolno, a skrypt ściągający nie wyświetlał żadnych informacji nawet z opcją verbose)
- Zakłócenie przebiegu labolatorium z systemów IDS powinno być dodatkowym ćwiczeniem w instrukcji, a nie tylko sugestią prowadzącego :-)