Portscanning und Portscan-Detection

Thomas Walpuski
Thomas.Walpuski@gmx.net

2000-12-20

Portscanning - Was ist das?

Als Portscanning bezeichnet man die Erkennung offener TCP- und UDP-Ports mittels verschiedener Techniken, die später näher erklärt werden. Alle diese Techniken basieren auf einem Testen der Reaktion des zu scannenden Rechners auf bestimmte TCP-Pakete bzw. UDP-Datagramme, die an die zu scannenden Ports adressiert sind.

Wozu nützt das?

Die Erkennung offener Ports bedeutet auch die Erkennung laufender Services, die ggf. deaktiviert werden müssen um:

- Sicherheitsproblemen vorzubeugen oder
- Resourcen zu sparen.

Weiterhin kann man mit Portscannern Firewall-Konfigurationen testen, Betriebssysteme erkennen …

Arten des Portscannings

- TCP-connect()-Scan
- TCP-SYN-Scan
- FIN-Scan
- Xmas-Scan
- Null-Scan
- UDP-ICMP-Scan
- UDP-write()/recvfrom()-Scan

TCP-connect()-Scan

Per `connect()` wird versucht, eine Verbindung aufzubauen. Liefert `connect()` 0 zurück, ist der Port offen, andernfalls (`connect()` liefert -1) ist der Port geschlossen.

Der TCP-connect()-Scan ist der einzige TCP-Scan, der *keine* root-Rechte benötigt. Zudem ist er sehr schnell. Er kann jedoch auch mit einfachen Methoden erkannt werden, da ein kompletter Verbindungsaufbau per 3-way-handshake durchgeführt wird.

Folgendes wurde während eines `nmap -P0 -sT -p 79, 80 localhost` mit `tethereal -lo0` gesnifft:

```
localhost -> localhost TCP 3721 > finger [SYN] Seq=1713108672 Ack=0 Win=16384 Len=0
localhost -> localhost TCP finger > 3721 [RST, ACK] Seq=0 Ack=1713108673 Win=0 Len=0
localhost -> localhost TCP 3722 > http [SYN] Seq=1713231337 Ack=0 Win=16384 Len=0
localhost -> localhost TCP http > 3722 [SYN, ACK] Seq=1713350423 Ack=1713231338 Win=57344 Len=0
localhost -> localhost TCP 3722 > http [ACK] Seq=1713231338 Ack=1713350424 Win=57344 Len=0
localhost -> localhost TCP 3722 > http [RST, ACK] Seq=1713231338 Ack=1713350424 Win=57344 Len=0
```

TCP-SYN-Scan

Der TCP-SYN-Scan sendet ein Paket, bei dem nur das SYN-Flag gesetzt ist. Anwortet der zu scannende Rechner mit einem SYN|ACK, so ist der Port offen, andernfalls (der Rechner anwortet mit RST|ACK) ist der Port geschlossen. Sofort nach der Antwort des Ziel-Rechners wird ein RST gesendet, um den Verbindungsaufbau abzubrechen. Daher spricht man auch von halboffenem Scanning.

Folgendes wurde während eines `nmap -P0 -sS -p 79, 80 localhost` mit `tethereal -lo0` gesnifft:

```
localhost -> localhost TCP 50529 > finger [SYN] Seq=971906275 Ack=0 Win=2048 Len=0
localhost -> localhost TCP finger > 50529 [RST, ACK] Seq=0 Ack=971906276 Win=0 Len=0
localhost -> localhost TCP 50529 > http [SYN] Seq=971906275 Ack=0 Win=2048 Len=0
localhost -> localhost TCP http > 50529 [SYN, ACK] Seq=1797564532 Ack=971906276 Win=49152 Len=0
localhost -> localhost TCP 50529 > http [RST] Seq=971906276 Ack=0 Win=0 Len=0
```

FIN-Scan

Der FIN-Scan basiert darauf, dass „schwachsinnige“ TCP-Pakete mit einem RST|ACK beantwortet werden.

Es wird ein Paket gesendet, bei dem nur das FIN-Flag gesetzt ist. Antwortet der zuscannende Host mit einem RST, ist der Port geschlossen.

Folgendes wurde während eines `nmap -P0 -sF -p 79, 80 localhost` mit `tethereal -lo0` gesnifft:

```
localhost -> localhost TCP 49452 > finger [FIN] Seq=0 Ack=0 Win=3072 Len=0
localhost -> localhost TCP finger > 49452 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
localhost -> localhost TCP 49452 > http [FIN] Seq=0 Ack=0 Win=3072 Len=0
localhost -> localhost TCP 49453 > http [FIN] Seq=0 Ack=0 Win=3072 Len=0
```

Xmas-Scan

Der Xmas-Scan ist eine Abwandlung des FIN-Scan. Zusätzlich zum FIN sind die Flags PSH und URG gesetzt.

Folgendes wurde während eines `nmap -P0 -sX -p 79, 80 localhost` mit `tethereal -lo0` gesnifft:

```
localhost -> localhost TCP 53023 > finger [FIN, PSH, URG] Seq=0 Ack=0 Win=1024 Urg=0 Len=0
localhost -> localhost TCP finger > 53023 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
localhost -> localhost TCP 50978 > http [FIN, PSH, URG] Seq=0 Ack=0 Win=1024 Urg=0 Len=0
localhost -> localhost TCP 50979 > http [FIN, PSH, URG] Seq=0 Ack=0 Win=1024 Urg=0 Len=0
```

Null-Scan

Auch der Null-Scan ist eine Variante des FIN-Scan. Diesmal sind keine Flags gesetzt.

Folgendes wurde während eines `nmap -P0 -sN -p 79, 80 localhost` mit `tethereal -lo0` gesnifft:

```
localhost -> localhost TCP 36614 > finger [] Seq=0 Ack=0 Win=2048 Len=0
localhost -> localhost TCP finger > 36614 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
localhost -> localhost TCP 36614 > http [] Seq=0 Ack=0 Win=2048 Len=0
localhost -> localhost TCP 36615 > http [] Seq=0 Ack=0 Win=2048 Len=0
```

UDP-ICMP-Scan

Der UDP-ICMP-Scan basiert darauf, dass auf ein UDP-Datagram an einen geschlossenen Port der Ziel-Rechner ein ICMP port unreachable sendet.

Folgendes wurde während eines `nmap -P0 -sU -p 513, 514 localhost` mit `tethereal -lo0` gesnifft:

```
localhost -> localhost WHO Source port:  33108 Destination port:  who
localhost -> localhost ICMP Destination unreachable
localhost -> localhost UDP Source port:  33600 Destination port:  syslog
localhost -> localhost UDP Source port:  33601 Destination port:  syslog
```

UDP-write()/recvfrom()-Scan

Der UDP-write()/recvfrom()-Scan ist ein UDP-ICMP-Scan, der vollkommen ohne root-Rechte (also ohne Zugriff auf RAW-Sockets, libpcap ..) durchgeführt werden kann. Die Schwierigkeit besteht darin zu erkennen, dass ICMP port unreachable gesendet wurde.

Der UDP-write()-Scan basiert darauf, dass der Empfang des ICMP port unreachable durch das Fehlschlagen eines zweiten `write()`s (`write()` liefert -1 als Rückgabewert) angezeigt wird.

Die Grundlage des UDP-recvfrom()-Scan ist, dass das Erhalten des ICMP port unreachable bei einem `recvfrom()` auf einen NONBLOCKING-Socket durch das Setzen von ERRNO auf 111 (ECONNREFUSED) angezeigt wird.

Welche Portscanner gibt es? Und was können sie?

- **Nmap** - (fast) alles (`http://www.insecure.org/nmap/`)
- **Strobe** - sehr schneller TCP-connect()-Scanner, nutzt NONBLOCKING-Sockets
- **UDP SCAN** - UDP-ICMP-Scanner (`http://shadowpenguin.backsection.net/toolbox/`)
- etc. pp. (`http://packetstorm.securify.com/UNIX/scanner/`)

Woran erkenne ich Portscanns?

Es gibt viele Umstände, bei denen von einem Portscan ausgangen werden kann:

- In einer bestimmten Zeit (z. B. 30s) wird eine bestimme Anzahl (z. B. 5) von Ports mit Paketen „beliefert“.
- Das zuerst an einen Port gesendete Paket ist charakteristisch für Stealth-Scans (siehe S. **??** ff.).
- Ungewöhnlich hoher Anteil von ICMP port unreachable am ICMP-Traffic.
- etc. pp.

Portscan-Detection in der Praxis

Grundsätzlich gibt es zwei verschiedene Arten, wie Portscan-Detection realisiert wird:

1. Der Portscan-Detector nistet sich auf bestimmten Ports mit `listen()` ein und registriert versuchte Verbindungsaufbauten.
   Vorteile:
   - man benötigt keine root-Rechte
   - es ist Ressourcen schonend

   Nachteile:
   - mit Stealth-Scans einfach zu umgehen
   - man stößt schnell an die Grenze der maximal offenen Dateien

   Diese Technik der Portscan-Erkennung verwendet z. B. Portsentry.
2. Der Portscan-Detector snifft den gesamten Netzwerkverkehr auf einem Device mit und registriert verdächtige Pakete.
   Vorteile:

- schwer zu umgehen

Nachteile:

- Ressourcen fressend

Diese Technik der Portscan-Erkennung verwendet z. B. Scanlogd.

Welche Probleme gibt es bei der Portscan-Detection?

- Umgehung des Erkennungsmechanismus durch stark **fragmentierte Pakete** (selbst die Header sind fragmentiert)
- Umgehung des Loggingmechanismus durch **Log-Floodging** - der Loggingmechanismus wird so lange mit gespooften Scans „gefüttert“, bis er keine weiteren Scans loggen kann, danach erfolgt der eigentliche Scan
- **DoS-Attacken** - durch Rekonfiguration der Firewall in Folge gespoofter Scans

Welche Programme zur Portscan-Detection gibt es?

- **Scanlogd** (`http://www.openwall.com/scanlogd/`)
- **Snort-Portscan-Preprocessor** (`http://www.snort.org/`)
- **Portsentry** (`http://www.psionic.com/abacus/portsentry/`)
- etc. pp. (`http://packetstorm.securify.com/UNIX/scan_detect/`)

Literatur

Literatur

[1] TCP port Stealth Scanning (http://phrack.infonexus.com/search.phtml?view&article=p49-15)

[2] The Art of Scanning (http://phrack.infonexus.com/search.phtml?view&article=p5111)

[3] Port Scan Detection Tools (http://phrack.infonexus.com/search.phtml?view&article=p53-13)