

# Netzwerke

Bierfert, Feresst, Günther, Schuster

21. März 2006

## Scanning

TCP

Grundlagen

Offene Scans

Stealth Scans

Idle Scan

Scanner

# Transmission Control Protocol (RFC 793)

- ▶ Zwei Endpunkte, bezeichnet mit Server und Client
- ▶ Server und Client aus je einem geordneten Paar mit IP-Adresse und Port

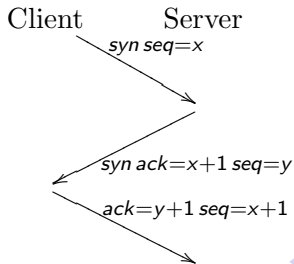
## TCP Header

← 32-bit →								
Source-Port						Destination-Port		
Sequence-Number								
Acknowledgment-Number								
Data Offset	Reserved	URG	ACK	PSH	RST	SYN	FIN	Window
Checksum							Urgent Pointer	
Options								

# TCP Verbindungsaufbau: Drei-Wege-Handshake

- ▶ Client sendet ein SYN Packet
- ▶ Server antwortet mit einem SYN/ACK Packet
- ▶ Client bestätigt den Erhalt des SYN/ACK Packets

## Three Way Handshake



# TCP Flags

- ▶ URG Daten auf die der Urgent Pointer zeigt werden sofort bearbeitet
- ▶ ACK Acknowledgment beim Verbinden (SYN) sowie beim Datentransfer
- ▶ PSH Daten die mit dem push-Flag gesetzt werden umgehen den Buffer
- ▶ RST Abbruch der Verbindung (auch als Antwort auf SYN)
- ▶ SYN Verbindungsaufbau
- ▶ FIN Verbindungsfreigabe

## Warum und wie?

- ▶ Sicherheitsprüfung eines Netzwerks/Servers
- ▶ Suchen von Schwachstellen in feindlichen Netzwerken um Angriffspunkte zu finden
- ▶ Verschiedene Arten von Scans und Möglichkeiten Scans zu erkennen und blockieren (IDS-Systeme und Firewalls)

# TCP Connect Scan

- ▶ s.g. 'Offener Scan'
- ▶ Kompletter Drei-Wege-Handshake auf gewünschten Ports
- ▶ Ist der Port offen ist der Versuch erfolgreich, andernfalls RST als Antwort
- ▶ Erkennung durch IDS-Systeme und Firewalls leicht möglich
- ▶ Normalerweise in Log-Dateien zu erkennen

## TCP SYN Scan

- ▶ s.g. 'Halboffener Scan'
- ▶ Statt des ACK-Paketes des Clients wird als Antwort ein RST-Paket geschickt
- ▶ Wird nicht von allen Rechnern mitgeloggt
- ▶ Trotzdem leicht zu verfolgen und blocken

# TCP Stealth Scans

- ▶ Ziel: Scans unerkannt durchführen
- ▶ Mittel: Ausnutzung von im Protokollablauf nach RFC 793 nicht vorgesehenen 'ungültigen' Paketen
- ▶ Nach RFC 793 gibt es zwei Möglichkeiten:
  - ▶ Port offen: Paket wird ignoriert oder Bestätigung durch ACK
  - ▶ Port geschlossen: RST als Antwort
- ▶ Nicht alle Implementierungen halten sich an diese Vorgabe (z.B. Windows sendet immer RST)
- ▶ Senden von Paketen der realen IP-Adresse ist trotzdem notwendig

## Verschiedene Stealth Scans

- ▶ FIN Scan: Das FIN Flag wird gesetzt und an den Port gesendet
- ▶ NULL Scan: Kein Flag des TCP Headers ist gesetzt
- ▶ XMAS Scan: FIN, URG, PUSH sind gesetzt
- ▶ ACK Scan: ACK ist gesetzt

# Idle Scan 1

- ▶ Komplette anonymer Scan durch Zuhilfenahme einer Zombie-Node
- ▶ Keine Pakete werden von der realen IP-Adresse zum Server geschickt
- ▶ IDS-Systeme und Firewalls verdächtigen die Zombie-Node und nicht den eigentlichen Client
- ▶ 'Aufwändigster' Scan

## Idle Scan 2

### Informationen des Zombie erfragen

Client                  Zombie

→ SYN/ACK  
IPID-Probe

← RST  
IPID= $x$

## Idle Scan 3

### Scan eines offenen Ports

Client                  Server

SYN/SRC-IP: Zombie  
→  
Session Start

Zombie                  Server

← SYN/ACK  
Session Acknowledgment

→ RST  
IPID=x+1

### Scan eines geschlossenen Ports

Client                  Server

SYN/SRC-IP: Zombie  
→  
Session Start

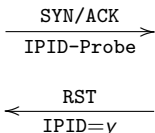
Zombie                  Server

← RST  
Port ist geschlossen

## Idle Scan 4

### Ergebnis abfragen

Client            Zombie



- ▶ Wenn für die IPID  $y = x + 1$  gilt, ist der Port geschlossen.
- ▶ Wenn für die IPID  $y = x + 2$  gilt, ist der Port offen.
- ▶ Funktioniert nur bei nicht zu frequentierten 'Zombies' daher Idle-Scan

# Scanner

Es gibt zwei Arten von Scannern:

- ▶ Portscanner
- ▶ Vulnerability Scanner

# Portscanner

- ▶ Verwendung zum Aufspüren von offenen Ports
- ▶ Möglichkeit verschiedener Scans
- ▶ Informationen über Betriebssystem
- ▶ Tests laufen automatisiert ab
- ▶ nmap als Quasi-Standard (<http://www.insecure.org/nmap/>) (GPL)

# Vulnerability Scanner

- ▶ Setzen auf Portscannern auf
- ▶ Erster Schritt ist ein Portscan
- ▶ Offene Ports/Services werden auf bekannte Probleme überprüft
- ▶ nessus als Beispiel (<http://http://www.nessus.org/>)
  - ▶ Version < 3 (GPL)
  - ▶ Nessus Attack Scripting Language zur Beschreibung von Sicherheitslücken
  - ▶ Sehr viele Scripte (s.g. Plugins) verfügbar
  - ▶ OpenSource Nachfolger OpenVAS (<http://www.openvas.org/>)