


Informatik Hochschule für angewandte
Wissenschaften Hamburg

Verteilte Systeme

Sicherheit





„It is easy to run a secure computer system. You merely have to disconnect all dial-up connections and permit only direct-wired terminals, put the machine and its terminals in a shielded room, and post a guard at the door.“

1

Informatik Hochschule für angewandte
Wissenschaften Hamburg

Problem Sicherheit

- Das Senden von Daten von einem zu einem anderen Computer ist **immer ein Risiko**.
- **Gefahren:**
 - **Mithören** (*Schnüffeln Sniffing/Lauschen Eavesdropping*): Versuch, ohne die entsprechende Berechtigung Nachrichten mitzuhören 
 - Vorgabe **falscher Identitäten** (*Parodieren Spoofing/Maskieren Masquerading*): Senden und Empfangen von Nachrichten unter einer anderen Identität (ohne die Erlaubnis dieser Identität)
 - **Unterbrechen**: Ein Teil des Systems, d.h. des gesamten Informationskanals, wird zerstört oder unbrauchbar. 

2

Informatik Hochschule für angewandte
Wissenschaften Hamburg

Problem Sicherheit

- **Änderung** von Nachrichten (*Verfälschen Tampering*): Abfangen von Nachrichten und Veränderung ihres Inhalts, bevor sie an den eigentlichen Empfänger weitergegeben werden (schwierig in Broadcast-Netzen, leicht bei Store-and-Forward)
- **Wiederholung** von Nachrichten (*Wiederholung Replay*): abgefangene Nachrichten werden abgespeichert und zu einem späteren Zeitpunkt erneut gesendet
- **Verweigerung** von Diensten (*Ablehnung von Diensten Denial of Service*): Eingeschleuste Komponenten verweigern die Dienstleistung, oder durch Überfluten eine Dienstverweigerung bewirken

3

Ziele von Sicherungsmaßnahmen

- **Vertraulichkeit** (Confidentiality)
Schutz der Informationen vor unautorisierter Einsichtnahme (Geheimhaltung!)
- **Integrität**
Schutz der Daten vor unautorisierter Veränderung (Verhindern von Modifikation oder Löschung!)
- **Authentizität**
Die Daten wurden wirklich von der Person gesendet, die behauptet, der Sender zu sein.
- **Verantwortlichkeit**
Jede sicherheitsrelevante Aktion im System kann eindeutig einem Urheber zugeordnet werden.
- **Verfügbarkeit**
Schutz des Systems vor (beabsichtigter) Störung (Verhindern von Abstürzen oder Performanceverlusten!), d.h. berechnete Zugriffe auf das System können nicht von Dritten verhindert werden.
- **Gefährdet durch** Konzeptionsfehler, Programmierfehler, Konfigurationsfehler

Angriffe



Angriffe

- **Angriff:**
Ein nicht autorisierter Zugriff bzw. Zugriffsversuch auf ein IT-System
- **Passiver Angriff:**
Zugriff auf vertrauliche Informationen (→ Verlust der Vertraulichkeit)
Beispiele: Abhören von Leitungen, Lesen von geheimen Dateien
- **Aktiver Angriff:**
Modifikation von Datenobjekten oder Systemressourcen (→ Verlust der Integrität / Verfügbarkeit)
Beispiele: Verändern / Löschen von Dateien oder IP-Paketen, Überschwemmen mit TCP-Verbindungsanfragen („Denial-of-Service“)

Funktionsweise von Angriffen

- ◆ Für einen Angriff, muß ein **Zugang zu dem System** bestehen.
- ◆ **Meist** über die **Kommunikationskanäle** des verteilten Systems.
- ◆ In den meisten Fällen werden **Angriffe von rechtmäßigen Benutzern** gestartet, die ihre Autorität mißbrauchen.
- ◆ **Nicht-zugangsberechtigte Angreifer** müssen Methoden wie das Raten oder Knacken von Passwörtern einsetzen.
- ◆ Außer diesen direkten Formen des Angriffs werden Programme eingesetzt, die das System von außen **infiltrieren**. (Passwort knacken, Virus, Wurm, ...)

7

Beispiel: Angriffstaktik eines Cracker-Angriffs

- ◆ Angriffsziel festlegen und **Informationen sammeln**
z.B. Erzeugen eines Pufferüberlaufs, Maskierung, ...
- ◆ Erstzugriff durch Ausnutzen von Schwachstellen
z.B. Knacken von Passwortdateien, Ausnutzen von Vertrauensbeziehungen
- ◆ Spuren verwischen
z.B. Manipulation von Protokolldateien, Verstecken von Dateien
- ◆ Hintertür offen lassen
z.B. Manipulation der Startdateien

8

Beispiel: Buffer Overflow

- ◆ **Problem:**
 - Nachlässige Programmierung
 - Unsichere Programmiersprache (meist C)
 - → Unzureichende Längenprüfung / Absicherung von Eingabedaten
- ◆ **Angriffstechnik:**
 - Durch Eingabedaten mit Überlänge (→ lokale Variablen, Parameter) werden Teile des Stacks überschrieben
 - Überschreiben der echten Rücksprungadresse
 - Platzieren von eigenem Assemblercode auf dem Stack oder einer gefälschten „Rücksprungadresse“ mit Aufruf einer Bibliotheksprozedur (LoadLibrary, Shell, ..)!
- ◆ **Fehlerquelle:**
 - Programmierer, der die Sprache „einfach so“ nutzt oder/und
 - Manager, der die Sprache C vorgibt, ohne entsprechende Hinweise, den Nachteil der Sprache durch eigene Implementierung (Fehlerbehandlung) abzufangen.

9

informatik Hochschule für angewandte
Wissenschaften Hamburg

Beispiel: Buffer Overflow

```

cmd = lies_aus_netz();
do_something(cmd);
.....
int do_something(char* InputString) {
    char buffer[4];
    strcpy (buffer, InputString);
    ...
    return 0;
}

```

strcpy kopiert ohne Prüfung
solange in den Speicher, bis
NULL gelesen wird!!!

10

informatik Hochschule für angewandte
Wissenschaften Hamburg

Beispiel: TCP SYN Flooding

- Typ: **Verweigerung** von Diensten
- Schwachstelle: **Implementierung des TCP**
- **Fehlerquelle:** konzeptueller Fehler beim Entwurf des TCP
- TCP-Verbindungsaufbau normalerweise im Handshake (->SYN; <-SYN,ACK(SYN); ->ACK(SYN))
- Bei **SYN-Flooding** werden viele SYN-Nachrichten als TCP-Verbindungsaufbau geschickt
- Die **ACK(SYN)**'s zum kompletten Verbindungsaufbau werden **unterdrückt**
- Es werden viele halboffene TCP Verbindungen beim Opfer erzeugt:
 - Jede **halboffene Verbindung frisst Ressourcen**
 - Es ist nur eine **begrenzte Anzahl an Puffern** vorhanden
 - wenn der Angreifer alle Puffer belegt, können keine Verbindungen mehr angenommen werden
 - die Attacke ist dann also geglückt (Denial of Service)

11

informatik Hochschule für angewandte
Wissenschaften Hamburg

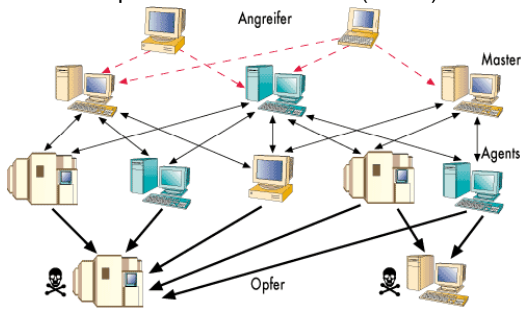
Beispiel: TCP SYN Flooding

12

Beispiel: Verteilte Attacken (DDoS)

- Typ: **Verteilte Verweigerung** von Diensten (DistributedDenialofService)
- Idee: **Überfluten** des Opfers durch **Pakete von sehr vielen** Rechnern!
- **Fehlerquelle**: Eigentlich, der Wunsch im WWW Dienste anzubieten oder/und in der Charakteristik der Anwender; wird zu konzeptuellem Fehler deklariert und durch eindeutige Identifizierbarkeit der Sender versucht zu beheben.
- Dadurch: **Erkennung**, woher Angriff kommt, **schwer** bis unmöglich (Unterscheidung Angreifer vs. echter Nutzer)
- **Neueste Generation** der Verweigerung von Diensten (DoS) Attacken: Seit Juli 1999 bekannt! Gefährlichste DoS-Attacke im Internet
- „**Tools**“ zur Attacke im Internet im Quelltext **vorhanden!**
 - Auch weniger Versierte können sie nutzen
 - Ständige Anpassung an Abwehrmechanismen

Beispiel: Verteilte Attacken (DDoS)



Typen von Viren

- **Programm-Viren**
 - Benutzen System- oder Anwendungsprogramme
- **Boot-Viren**
 - Kopieren sich an den Anfang des Bootsektors einer Festplatte oder Diskette
 - Laden sich bei jedem Bootvorgang selbst in den Hauptspeicher
- **Makro-Viren**
 - Hängen sich als „normale“ Makro-Programme an Office-Dokumente
 - Nutzen die „auto_open“-Funktionalität
- **Retro-Viren**
 - Manipulieren Konfigurationsdateien oder Datenbanken von Virensclannern
- **Hoax-Viren**
 - Veranlassen unbedarfte Benutzer, sich selbst wie ein Virus zu verhalten (→ „Social Engineering“!)

Würmer

- **Virus** = Programm, das andere Programme infiziert
- **Wurm** = Programm, das sich auf einem Rechner „einschleicht“, sich dort vervielfältigt und anschließend auf weitere Rechner verbreitet.
- 2. November 1988: Robert Morris Jr., Administrator an der Cornell University, initiierte sein Programm, später „Morris Wurm“ genannt
 - **Infektion** von **2.600 Systemen** (nach Clifford-Stoll)
 - Wirtschaftlicher Schaden: **97 Mio. USD** (nach J.McAfee)
 - Daraufhin: Entstehung des **CERT** (Comp. Emergency Response Team)
- **LOVELETTER.TXT.vbs** am 4. Mai 2000
 - Ursache: VB Scripting-Funktionalität in Microsoft E-Mail-Programmen
 - Bei Öffnen verschickt sich der Wurm an alle E-Mail-Adressen im Adressbuch
 - Da der Wurm im Quelltext vorliegt (VBScript ist ja eine interpretierte Sprache), können relativ leicht abgewandelte Würmer erzeugt werden (Mittlerweile über 82 Abwandlungen bekannt)

16

Trojanische Pferde

- **Trojanische Pferde:** Programme mit Vorgabe (und meist auch Erfüllung) einer Funktionalität, jedoch mit zusätzlich versteckter, unerwünschter Funktionalität; die z.B. externen Zugriff auf den Rechner ermöglicht
- **Verbreitung:** meist in kleinen Programmen als Anhang an e-Mails; seltener auch veränderte Originalsoftware auf nicht-vertrauenswürdigen Archiven (FTP o.ä.)
- **Illustre Namen:** GirlFriend, SubSeven, BackOrifice



17
