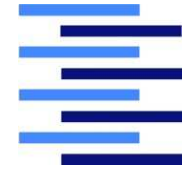


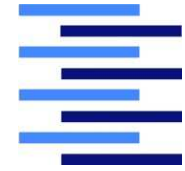
# Kapitel 2

## Bedrohungen und Angriffe



1. **Rechtslage**
2. Angreifer
3. Typische Angriffstechniken
4. Malware
5. Schwachstellen-Sammlungen

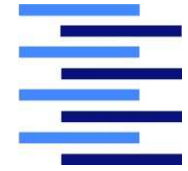
# Computer-Kriminalität



- **Viele Angriffe sind Verbrechen, aber nicht alle**
    - Kriminalität wird durch nationale Gesetze definiert – dem entsprechend gibt es viele Unterschiede und ungleiche Behandlung
    - Akzeptables Verhalten wird außerdem durch Sozialisierung und organisatorische Vorgaben definiert – und ggf. auch sanktioniert
- Ereignisse (Events) passieren
- Ereignisse werden dann klassifiziert
- und manche betrachten wir als Angriffe!

# Rechtslage in Deutschland

## (entspricht EU-Richtlinien)



Computerkriminalität im StGB (Strafgesetzbuch):

- StGB §202a,b,c: Ausspähen und Abfangen von Daten, Vorbereitung des Ausspähens/Abfangens von Daten
- StGB §263a: Computerbetrug
- StGB §268: Fälschung technischer Aufzeichnungen
- StGB §269: Fälschung beweiserheblicher Daten
- StGB §270: Täuschung im Rechtsverkehr bei Datenverarbeitung
- StGB §303a,b: Datenveränderung / Computersabotage
- StGB §316b: Störung öffentlicher Betriebe
- StGB §317: Störung von Telekommunikationsanlagen

In den meisten Fällen ist auch der Versuch strafbar.  
Teilweise ist ein Strafmaß von bis zu 10 Jahren  
Freiheitsstrafe vorgesehen.

# Strafgesetzbuch §202a - b



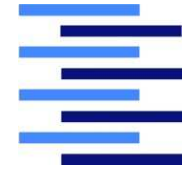
- **Ausspähen von Daten (§ 202a)**

- (1) Wer unbefugt sich oder einem anderen **Zugang zu Daten**, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, **unter Überwindung der Zugangssicherung verschafft**, wird mit **Freiheitsstrafe bis zu drei Jahren** oder mit Geldstrafe bestraft.
- (2) Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden.

- **Abfangen von Daten (§202b)**

Wer unbefugt sich oder einem anderen unter Anwendung von technischen Mitteln nicht für ihn bestimmte Daten (§ 202a Abs. 2) **aus einer nichtöffentlichen Datenübermittlung oder aus der elektromagnetischen Abstrahlung einer Datenverarbeitungsanlage verschafft**, wird mit **Freiheitsstrafe bis zu zwei Jahren** oder mit Geldstrafe bestraft, wenn die Tat nicht in anderen Vorschriften mit schwererer Strafe bedroht ist.

# Strafgesetzbuch §202c

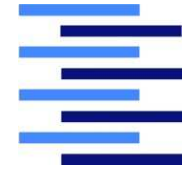


- **Vorbereiten des Ausspähens und Abfangens von Daten (§ 202c)**

- (1) Wer eine **Straftat** nach § 202a oder § 202b **vorbereitet**, indem er
1. Passwörter oder sonstige Sicherungscodes, die den Zugang zu Daten (§ 202a Abs. 2) ermöglichen, oder
  2. Computerprogramme, deren Zweck die Begehung einer solchen Tat ist, herstellt, sich oder einem anderen verschafft, verkauft, einem anderen überlässt, verbreitet oder sonst zugänglich macht, wird mit **Freiheitsstrafe bis zu einem Jahr** oder mit Geldstrafe bestraft.
- (2) § 149 Abs. 2 und 3 gilt entsprechend. *[„Tätige Reue“]*

Der Vorsatz ist rechtlich entscheidend!

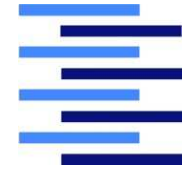
# Strafgesetzbuch §303a



- **Datenveränderung (§ 303a)**

- (1) Wer rechtswidrig Daten (§ 202a Abs. 2) löscht, unterdrückt, unbrauchbar macht oder verändert, wird mit **Freiheitsstrafe bis zu zwei Jahren** oder mit Geldstrafe bestraft.
- (2) Der Versuch ist strafbar.
- (3) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

# Strafgesetzbuch §303b



- **Computersabotage (§303b)**

- (1) Wer eine Datenverarbeitung, die für einen anderen von wesentlicher Bedeutung ist, dadurch erheblich stört, dass er
  1. eine Tat nach §303a Abs. 1 begeht
  2. Daten (§ 202a Abs. 2) in der Absicht, einem anderen Nachteil zuzufügen, eingibt oder übermittelt oder
  3. eine Datenverarbeitungsanlage oder einen Datenträger zerstört, beschädigt, unbrauchbar macht, beseitigt oder verändert,wird mit **Freiheitsstrafe bis zu drei Jahren** oder mit Geldstrafe bestraft.
- (2) Handelt es sich um eine Datenverarbeitung, die für einen fremden Betrieb, ein fremdes Unternehmen oder eine Behörde von wesentlicher Bedeutung ist, ist die Strafe **Freiheitsstrafe bis zu fünf Jahren** oder Geldstrafe.
- (3) Der Versuch ist strafbar.

# Strafgesetzbuch §303b (Forts.)

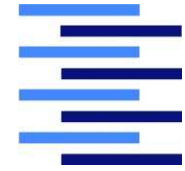


- **Computersabotage (§303b)**

- (4) In besonders schweren Fällen des Absatzes 2 ist die Strafe **Freiheitsstrafe von sechs Monaten bis zu zehn Jahren**. Ein besonders schwerer Fall liegt in der Regel vor, wenn der Täter
  1. einen Vermögensverlust großen Ausmaßes herbeiführt,
  2. gewerbsmäßig oder als Mitglied einer Bande handelt, die sich zur fortgesetzten Begehung von Computersabotage verbunden hat,
  3. durch die Tat die Versorgung der Bevölkerung mit lebenswichtigen Gütern oder Dienstleistungen oder die Sicherheit der Bundesrepublik Deutschland beeinträchtigt.
- (5) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.



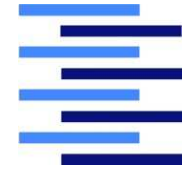
# IT-Sicherheitsgesetz vom 24. Juli 2015



- Pflicht zur Erfüllung von **Mindestanforderungen** an IT-Sicherheit für **Betreiber kritischer Infrastrukturen** und **Telekommunikationsanbieter**
- Pflicht zur **Meldung** erheblicher **IT-Sicherheitsvorfälle** für **Betreiber kritischer Infrastrukturen** und **Telekommunikationsanbieter**
- Verpflichtung der **Telekommunikationsanbieter** zur **Information der Nutzer** über Schadprogramme und zur Bereitstellung technischer Hilfsmittel für ihre Erkennung und Beseitigung
- Jährliche **Berichtspflicht** des **BSI**

# Kapitel 2

## Bedrohungen und Angriffe



1. Rechtslage

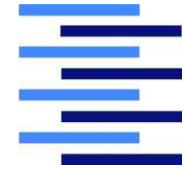
2. Angreifer

3. Typische Angriffstechniken

4. Malware

5. Schwachstellen-Sammlungen

# Angreifer-Typen („Täter“) Teil I



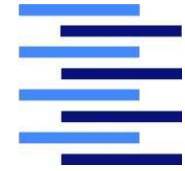
Bezeichnung	Charakterisierung	Ziele	Motive
<b>Hacker</b> („White Hats“)	Sicherheitsfachleute	<ul style="list-style-type: none"><li>• auf Schwachstellen („Exploits“) aufmerksam machen</li></ul>	<ul style="list-style-type: none"><li>• Wissenserwerb</li></ul>
<b>Cracker</b> („Black Hats“)	Technisch versierte Kriminelle (ggf. Organisationen)	<ul style="list-style-type: none"><li>• Diebstahl von Informationen (<i>Kreditkartennummern, Identitäten, Passwörter, ...</i>)</li><li>• Verkauf von Dienstleistungen über Bot-Netze (<i>Spamversand</i>)</li><li>• Erpressung (<i>Systemverfügbarkeit</i>)</li></ul>	<ul style="list-style-type: none"><li>• Bereicherung</li></ul>
<b>Skriptkids</b>	jugendlich, technisch unbedarft, nutzt im Internet veröffentlichte Schwachstellen und Tools	<ul style="list-style-type: none"><li>• Ruhm in der Szene</li><li>• Spiellust</li><li>• Faszination</li></ul>	<ul style="list-style-type: none"><li>• Eitelkeit</li><li>• Neugier</li></ul>

# Angreifer-Typen („Täter“) Teil II



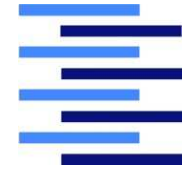
Bezeichnung	Charakterisierung	Ziele	Motive
<b>Geheimdienste</b>	Technisch versierte GeheimdienstmitarbeiterInnen	<ul style="list-style-type: none"><li>• Wirtschaftsspionage</li><li>• Militärische Spionage</li><li>• Terrorbekämpfung</li></ul>	<ul style="list-style-type: none"><li>• Wirtschaftliche Vorteile für lokale Firmen</li><li>• „Nationale Sicherheit“</li></ul>
<b>Interne MitarbeiterInnen</b>	Personen mit internen Kenntnissen und Zugriffsrechten	<ul style="list-style-type: none"><li>• Sabotage</li><li>• Sammeln interner Informationen</li><li>• Wirtschaftsspionage für Konkurrenzfirmen</li></ul>	<ul style="list-style-type: none"><li>• Frust und Wut</li><li>• Neugier</li><li>• Bereicherung</li></ul>

# Angriffstaktik eines Cracker-Angriffs über das Internet



1. Angriffsziel festlegen und Informationen sammeln
2. Erstzugriff durch Ausnutzen von Schwachstellen
  - z.B. Erzeugen eines Buffer Overflows, Social Hacking (Mail, Telefon), ...
3. Ausbau der Zugriffsberechtigungen
  - z.B. Knacken von Passwortdateien, Ausnutzen von Vertrauensbeziehungen
4. Spuren verwischen
  - z.B. Manipulation von Protokolldateien, Verstecken von Dateien (NTFS-Streams!)
5. Hintertür offen lassen („Backdoor“)
  - z.B. Manipulation der Startdateien

**z.T. durch „Trojaner“ automatisiert!  
→ „Rootkits“**



# Informationen sammeln

## 1. Footprinting

- Sammeln aller öffentlich zugänglichen IT-Sicherheitsinformationen eines Unternehmens
- **Beispiel-Ergebnisse:** DNS-Namen, IP-Adressen, Namen von Mitarbeitern, Remote Access-Telefonnummern

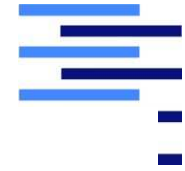
## 2. Scanning

- Feststellen, welche Systeme wie betrieben werden und aus dem Internet erreichbar sind
- **Beispiel-Ergebnisse:** Informationen über offene TCP/UDP-Ports, eingesetzte Betriebssysteme und Firewalls

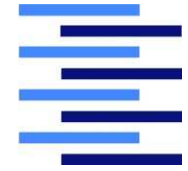
## 3. Auswertung

- Ermitteln von Benutzerkonten, Ressourcennamen und Anwendungen
- Auswertungstechniken sind i.d.R. vom Betriebssystem abhängig!

# Scanning: TCP-/UDP-Portscans



- **Ziele:**
  - Systematische Ermittlung, auf welchen Ports ein Verbindungsaufbau möglich ist
  - Welche Dienste werden von welchem Server angeboten?
  - Welche Betriebssysteme werden verwendet?
- **Scan-Typen (Auswahl):**
  - **TCP-Connect-Scan:** Versuch, eine vollständige TCP-Verbindung aufzubauen (→ 3-Wege-Handshake SYN, SYN/ACK, ACK). Wenn der Server mit RST/ACK antwortet, ist der Port inaktiv.
  - **TCP-SYN-Scan:** Lässt TCP-Verbindungen „halboffen“, da der Client kein ACK auf ein SYN/ACK des Servers mehr sendet
  - **UDP-Scan:** UDP-Paket zum Zielport senden. Wenn der Server mit „ICMP Port unreachable“ antwortet, ist der Port inaktiv.
- **Betriebssystem-Erkennung:**
  - Test auf spezifische Implementierungsdetails (Abweichungen von RFCs, Default-Parameterwerte, ...)



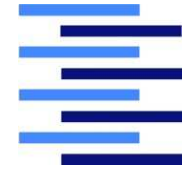
# Penetrationstests

- **Test-Angriffe**, um Schwachstellen zu ermitteln
- Vorgehensweise wie bei „echten“ Angreifern
  - Interne Sicht: Whitebox-Ansatz
  - Externe Sicht: Blackbox-Ansatz
- Problematisch, aber nötig: **Tool-Einsatz (→ §202c StGB!)**
- OpenSource-Framework: **Metasploit**
  - Kann verschiedene Tools integrieren, z.B.
    - Nessus (BS-Schwachstellen-Scanner)
    - NeXpose (BS-Schwachstellen-Scanner)
    - Nikto (Webserver-Schwachstellen-Scanner)
    - Nmap (Portscanner)
- Integrierte Tool-Sammlung: Kali-Linux ([www.kali.org](http://www.kali.org))



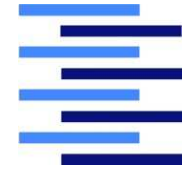
# Kapitel 2

## Bedrohungen und Angriffe



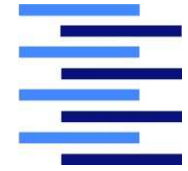
1. Rechtslage
2. Angreifer
3. Typische Angriffstechniken
4. Malware
5. Schwachstellen-Sammlungen

# Angriffstechniken – Übersicht (1)

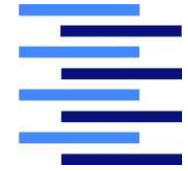


- **Einbruch in IT-Systeme durch**
  - Ausnutzung von programmtechnischen Schwachstellen, z.B.
    - C-Programme: Buffer Overflow / Format String / Linearisierung / ...
    - Web-Anwendungen: Injection / Cross-Site Scripting / ...
  - Maskierung / Manipulation der Kommunikation
    - Fälschen der Identität („Spoofing“)
    - „Man-in-the-middle“- Attacke
  - Erschleichen von Zugangsberechtigungen
    - Geschicktes Befragen („Social Hacking“)
    - Raten / Ausprobieren von Passwörtern („Cracking“)
    - Abhören des Netzverkehrs („Sniffer“)
    - Wiedereinspielen von Zugangsberechtigungen ("Replay")

# Angriffstechniken – Übersicht (2)



- IT-Systeme gezielt ausschalten („Denial-of-Service“)
  - Überschwemmung mit Verbindungsanfragen oder anderen Dienstanforderungen
- Schadsoftware („Malware“) zur Ausführung bringen
  - Viren
  - Würmer
  - Trojanische Pferde



# Buffer Overflow - Beispiel

```
cmd = lies_aus_netz();
```

```
do_something(cmd);
```

```
.....
```

```
int do_something(char* InputString) {
```

```
    char buffer[4];
```

```
    strcpy (buffer, InputString);
```

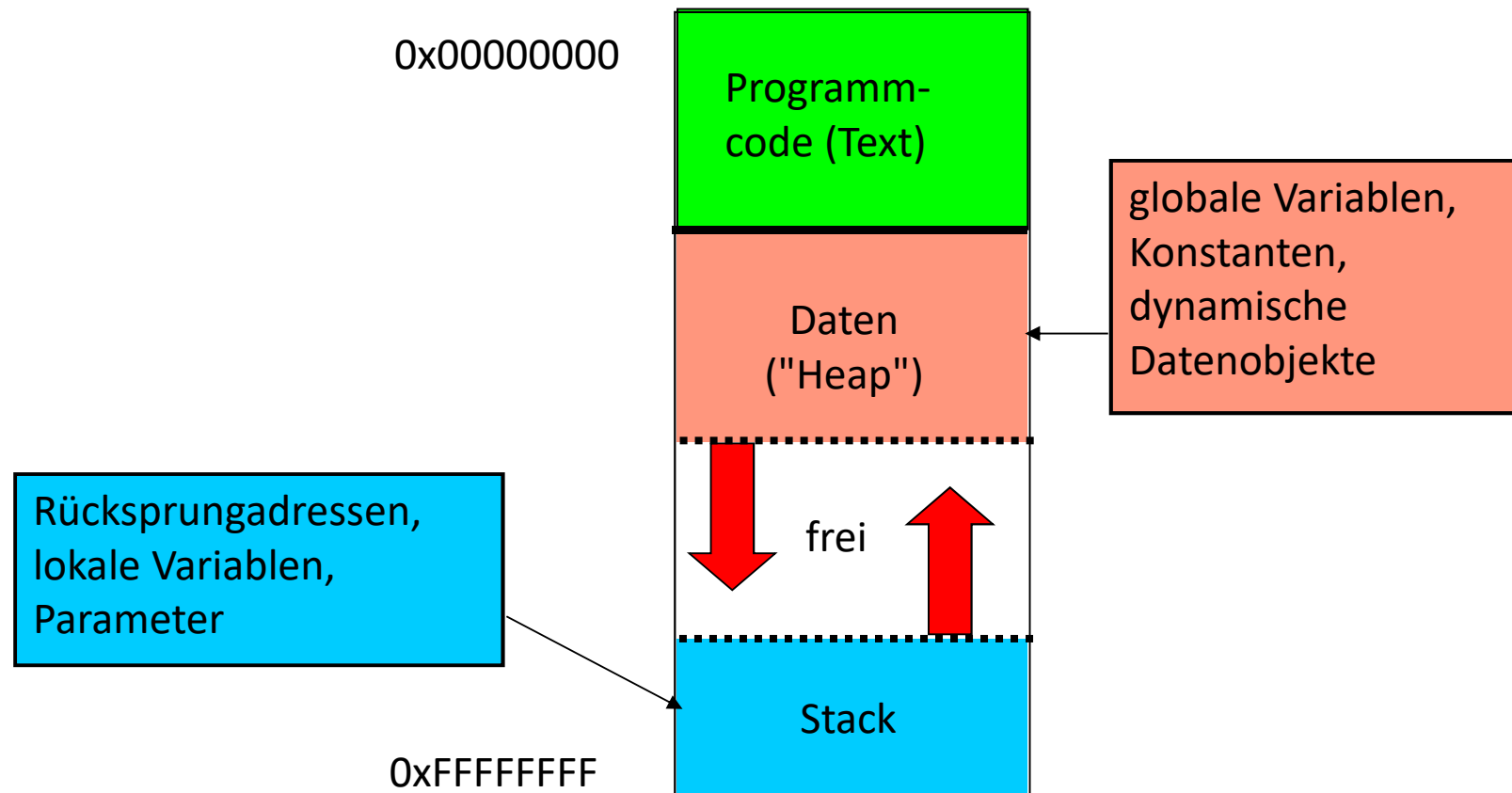
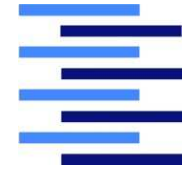
```
    ...
```

```
    return 0;
```

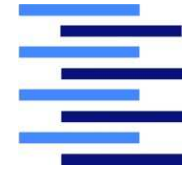
```
}
```

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

# Typisches Prozesslayout im Hauptspeicher



# Buffer Overflow - Angriff



- **Problem:**

- Nachlässige Programmierung, unsichere Programmiersprache (meist C)
- ➔ Unzureichende Längenprüfung / Absicherung von Eingabedaten

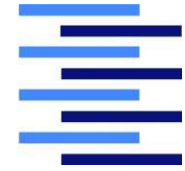
- **Angriffstechnik:**

- Durch Eingabedaten mit Überlänge (→ lokale Variablen, Parameter) werden Teile des Stacks überschrieben
- Platzieren von fremdem Assemblercode auf dem Stack, Überschreiben der echten Rücksprungadresse

- **Gegenmaßnahmen:**

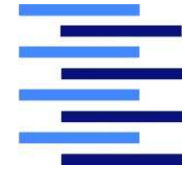
- Im C-Programmcode nur Funktionen mit Angabe einer Maximallänge verwenden (`strncpy` statt `strcpy`)
- Compiler-Optionen aktivieren („Canary“-Prüfungen)
- Verboten, dass Code, der auf dem Stack liegt, ausgeführt wird („Data Execution Prevention“)
- ASLR: Address Space Layout Randomization

# Format String - Angriff



- **Problem:**
  - Schlampige Verwendung von `printf()` (in C/C++):  
`printf (var)` statt z.B. `printf („%s“, var)`
  - `printf`-Argumente werden vom Stack gelesen
  - Wenn zu wenig Argumente vorhanden  
→ Weiterlesen der vorhandenen Stack-Daten
  - `printf`-Option `%n`: Schreiben der Anzahl bisher ausgegebener Zeichen an eine als Argument übergebene Speicheradresse (!)
- **Angriffstechnik (analog Buffer Overflow-Angriff):**
  - Eingabestring mit vielen eigenen Format-Anweisungen (`%x`, `%n`) und Adressdaten versehen
  - Manipulation des Stacks möglich: Mit `%x%x%x...` Code auf dem Stack ablegen und mit `%n` Rücksprungadresse überschreiben
- **Gegenmaßnahme:**
  - Bei `printf` immer als erstes Argument einen Formatstring angeben!

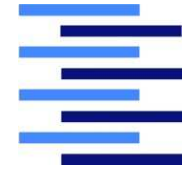
# Linearisierung – Beispielprogramm: Prüfung einer Seriennummer



```
int main (int argc, const char *argv[]) {
    int i;
    char* serial="S123N456";

    if (strlen(argv[1]) < 8) {
        printf("Error");
        exit(0);
    }
    for (i=0; i < 8; ++i) {
        if(argv[1][i] != serial[i])
            break;
    }
    if (i == 8) {
        printf("Serial number is correct!");
    }
}
```





# Linearisierungs - Angriff

- **Problem:**

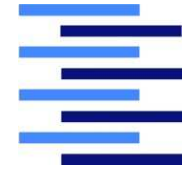
- Eingaben, die geheime Schlüssel (z.B. ein Passwort) darstellen, werden von einem Programm in einzelnen Schritten (z.B. byteweise) geprüft.
- Das Verhalten des Programms lässt Rückschlüsse über die Korrektheit der einzelnen Eingabeteile zu (z.B. anhand der Antwortzeiten).

- **Angriffstechnik:**

- Bei n Eingabeteilen:
- For i = 1 to n do
  - Ermitteln des korrekten i-ten Eingabeteils durch systematisches Ausprobieren

- **Gegenmaßnahmen:**

- Möglichst alle Informationskanäle eines Programms absichern
- Möglichst gleichartiges Programmverhalten garantieren, unabhängig vom Ergebnis



# Beispiel: SQL-Injection

- Eine JAVA-Anwendung generiert dynamisch SQL-Code für eine Datenbankabfrage aufgrund einer Benutzereingabe (html-Form):

```
String sql = new String("SELECT * FROM  
kunde WHERE card= ' " +  
request.getParameter("cardname") + " ' ")
```

- Beispiel für eine korrekte Eingabe:

➤ **SELECT \* FROM kunde WHERE card = 'visa'**

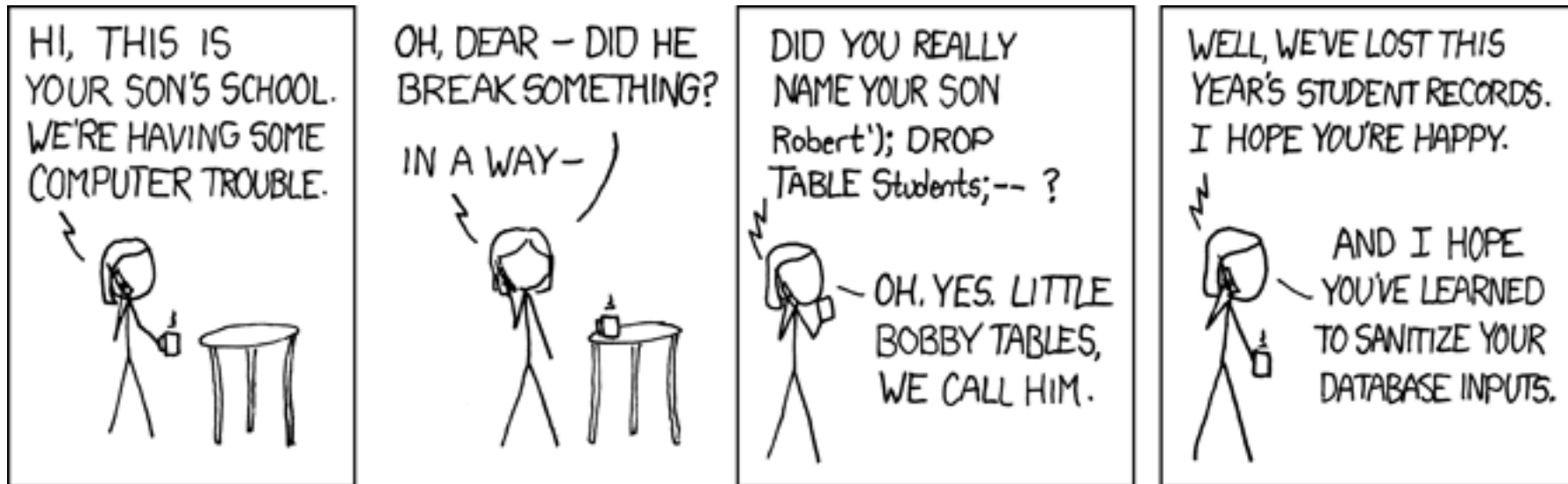
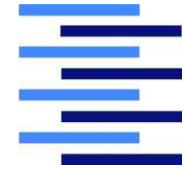
- Bösertige Eingabe:

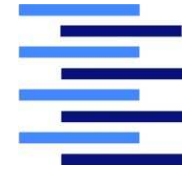
➤ **' ;DROP TABLE KUNDE--**

→ Was passiert?

# Zusatz-Beispiel:

## Die Schule meldet sich bei der Mutter ...





# Beispiel: Cross-Site-Scripting (XSS)

- index.php – Beispiel:

```
<?php
    $name = $_GET['name'];
    echo "Welcome $name<br>";
?>
```

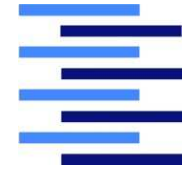
- Ein Angreifer sendet folgende URL an den Webserver:

```
index.php?name=guest<script>alert('attacked')</script>
```

- Was passiert?

**Code wird vom Browser ausgeführt, wenn die Metazeichen  
'<' und '>' nicht vom Web-Server ersetzt worden sind  
(PHP: htmlspecialchars() – Funktion)**

# Webanwendungen: Injection - / XSS-Angriff



- **Problem:**

- Eingaben eines Benutzers oder eine anderen Anwendung werden von der Webanwendung ungeprüft übernommen.
- Diese Eingaben enthalten Code-Fragmente, die dann unbeabsichtigt ausgeführt werden.

- **Angriffstechnik:**

- Analyse des Script-Codes einer Webseite
- Eingabe von böartigem Code anstelle der erwarteten Daten
- Manipulation der Webseite / Datenbank oder Ausspähen von Informationen (z.B. Session-IDs)

- **Gegenmaßnahme:**

- Alle Eingaben prüfen bzw. absichern!

# Beispiel: Webserver-Spoofing

File Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe


MH Links × Willkommen bei Facebook... × +

file:///C:/Users/HBN/Downloads/www.facebook.com.html Suchen

Wir verwenden Cookies, um Inhalte zu personalisieren, Werbeanzeigen maßzuschneidern und zu messen sowie die Sicherheit unserer Nutzer zu erhöhen. Wenn du auf unsere Webseite klickst oder hier navigierst, stimmst du der Erfassung von Informationen durch Cookies auf und außerhalb von Facebook zu. Weitere Informationen zu unseren Cookies und dazu, wie du die Kontrolle darüber behältst, findest du hier: [Cookie-Richtlinie](#).

E-Mail-Adresse oder Handynummer Passwort   Anmelden  
Konto vergessen?

## Auf Facebook bleibst du mit Menschen in Verbindung und teilst Fotos, Videos und vieles mehr mit ihnen.



## Registrieren

Facebook ist nicht kostenlos, sondern benutzt deine Daten für obscure Zwecke!

Vorname  Nachname

Handynummer oder E-Mail-Adresse

Handynummer/E-Mail erneut eingeben

Neues Passwort

Geburtsdag

Tag  Monat  Jahr  Warum muss ich meinen Geburtsdag angeben?

☐ Weiblich ☐ Männlich

Indem du auf „Registrieren“ klickst, erklärst du dich mit unseren [Nutzungsbedingungen](#) einverstanden und bestätigst, dass du unsere [Datenrichtlinie](#) einschließlich unserer [Cookie-Richtlinie](#) gelesen hast. Eventuell erhältst du SMS-Benachrichtigungen von Facebook, die du jederzeit abbestellen kannst.

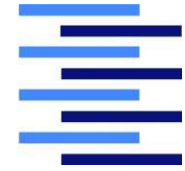
**Registrieren**

Erstelle eine Seite für einen Star, eine Band oder ein Unternehmen.

Deutsch English (US) Türkçe Polski Italiano Română Français (France) Русский العربية Español Português (Brasil)

Registrieren	Anmelden	Messenger	Facebook Lite	Handy	Freunde finden	Nutzer	Seiten	Orte	Spiele	Standorte
Stars	Marktplatz	Gruppen	Moments	Instagram	Über uns	Werbeanzeige erstellen	Seite erstellen	Entwickler	Karriere	Datenschutz
Cookies	Datenschutzinfo	Impressum/Nutzungsbedingungen	Hilfe							

# Maskierung: Fälschen der Identität („Spoofing“)



- **Problem:**
  - Fälschen von Absenderadressen (ohne zusätzliche Authentifizierungsmaßnahmen) ist oft möglich
  - Einsatz meist in Kombination mit verschiedenen Angriffstechniken
- **IP-Spoofing:**
  - Fälschen der **Senderadresse** in IP-Paketen
- **ARP-Spoofing**
  - Beantworten einer ARP-Broadcast-Anfrage anstelle von Host A  
→ Eintrag in ARP-Tabelle: **eigene MAC-Adresse, IP-Adresse von Host A**
  - Datenverkehr an A wird umgeleitet (A erhält nichts mehr)
  - Funktioniert zuverlässig meist nur innerhalb eines Netzsegments
- **DNS-Spoofing**
  - Modifizieren des Caches eines DNS-Servers durch falsche Zusatz-Angaben  
**Name von Host A, eigene IP-Adresse (→ „Cache Poisoning“)**
- **Mail-Spoofing**
  - Angabe einer **falschen Mailadresse** als Absender
- **Webserver-Spoofing**
  - Umlenken auf **falschen Web-Server** (Mail, Link, DNS-“Cache Poisoning“) und erschleichen von Anmeldedaten („Phishing“)

# Maskierung: „Man-in-the-Middle“ - Attacke



- **Problem:**

- Schwache Authentifikation oder Erschleichung von Zugangsdaten in einem gemeinsamen Netzwerk

- **Angriffstechnik:**

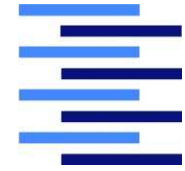
- Rechner X gibt sich gegenüber A als B und gegenüber B als A aus
- A und B glauben, direkt miteinander zu kommunizieren und tauschen evtl. geheime Nachrichten aus, die aber von X empfangen und dann lediglich weitergeleitet werden (ggf. modifiziert).

- **Beispiel**

- Zwei Bluetooth-Geräte A und B haben einen Sitzungsschlüssel vereinbart (PIN) und jeweils für weitere gemeinsame Sessions gespeichert
- Wenn X den Sitzungsschlüssel kompromittiert (knackt), kann X jeweils mit A und B eine Verbindung wie oben beschrieben aufbauen



# Erschleichen von Zugangsberechtigungen: „Social Hacking“ / „Social Engineering“



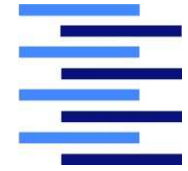
- **Definition:**
  - „Social Hacking“ / „Social Engineering“ umfasst alle Vorgehensweisen, bei denen Menschen getäuscht, gelenkt oder manipuliert werden, so dass Sie vertrauliche Informationen preisgeben.
- **Problem:**
  - Viele Benutzer und Supportmitarbeiter sind äußerst leichtgläubig und scheuen im zwischenmenschlichen Umgang oft formale Absicherungen, um Arbeit und evtl. Konflikte zu vermeiden („Misstrauen Sie mir etwa?“)
- **Angriffstechniken (Beispiele):**
  - **Anruf beim Support:** Neuer Mitarbeiter, auswärts tätig, benötigt neue Kennung, Passwort und Einwahlinformationen ...
  - **Anruf bei Benutzer:** Supportmitarbeiter, benötigt Benutzerpasswort, um administrative Arbeiten ausführen zu können ...
  - ....

# Erschleichen von Zugangsberechtigungen: Cracking von Passwortdateien



- **Problem:** „Schwache“ Passwörter können leicht erraten werden
  - Zu kurz
  - Bekannte Wörter (Namen, Begriffe, ...)
  - Trivial (Firmenname, Passwort = Benutzername), ...
- **Angriffstechnik:** Tools verfügbar zum
  - Auslesen einer Passwortdatei
  - automatischen Ausprobieren von Passwörtern mit Hilfe eines Wörterbuchs und/oder Erzeugung aller Kombinationsmöglichkeiten (z.B. „*John the Ripper*“)

# Erschleichen von Zugangsberechtigungen: Abhören des Netzverkehrs („Sniffer“)



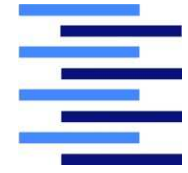
- **Problem:**

- In Broadcast-Netzen (z.B. WLAN) kann **jeder** angeschlossene Rechner **jedes** Datenpaket „hören“
- Pakete, die an einen anderen Rechner adressiert sind, sollten eigentlich ignoriert werden ...

- **Angriffstechnik:**

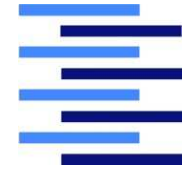
- Betrieb der Netzkarte im „Promiscuous“-Modus
- Analyse sämtlicher Pakete (inkl. IP-/TCP-Header + Nutzdaten) durch „Sniffer“-Tool (z.B. „Wireshark“)
- Herausfiltern von vertraulichen Informationen

# Wiedereinspielen von Zugangsberechtigungen („Replay“-Angriff)



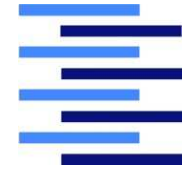
- **Problem:**
  - Zugangsberechtigungen (Schlüssel, Ausweise) können ggf. abgehört werden
- **Angriffstechnik:**
  - Der Angreifer X gelangt durch Abhören des Netzverkehrs in den Besitz des (verschlüsselten) Passworts von Benutzer M auf Server S
  - Dieses wird von X benutzt, um sich später selbst als M auf dem Server S anzumelden

# IT-Systeme gezielt ausschalten: „Denial-of-Service“ (DoS)



- **Problem:**

- Viele IT-Systeme bieten (in einem Netzwerk) Dienste an, ohne beeinflussen zu können, wer wieviele Ressourcen **anfordert**
- Bei einer plötzlichen massenhaften Dienstanforderung – **ggf. mit ungültigen Parametern** – werden oft so viele Ressourcen (Netzwerkbandbreite, Pufferspeicher, CPU) in Anspruch genommen, dass das IT-System (bei schlechter Implementierung) abstürzt

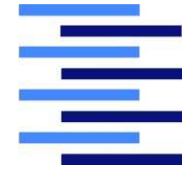


# „Denial-of-Service“ – Angriffstechnik (Beispiele)

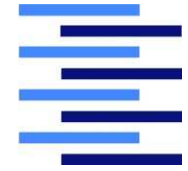
- **SYN-flooding-Angriff:**
  - Massenhaft TCP-SYN-Pakete (Verbindungsanfrage) mit gefälschter (ungültiger) Absenderadresse an Opfer versenden
    - ➔ Opfer reserviert für jede Anfrage TCP-Pufferspeicher, versendet massenhaft SYN/ACK-Antworten und wartet jeweils auf Timeout
    - ➔ **Pufferplatzmangel!**
- **Smurf-Attacke (→ Distributed DoS-Angriff):**
  - Massenhaft ICMP Echo-Anfragen (**ping**) an viele Rechner senden (Broadcast-Anfrage an Netzwerkadresse!)
  - Senderadresse (gefälscht): Adresse des Opfers
  - Alle Rechner werden die **ping**-Antwort an das Opfer zurücksenden ➔ **Überlastung der Netzkomponenten!**

# Kapitel 2

## Bedrohungen und Angriffe



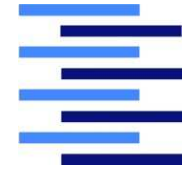
1. Rechtslage
2. Angreifer
3. Typische Angriffstechniken
4. **Malware**
5. Schwachstellen-Sammlungen



# Was ist ein Virus?

- Ein Stück ausführbarer **Code**
- Kein eigenständiges Programm → benötigt „**Wirt**“
- Kann sich selbst in andere Programme ein- oder anhängen („**Infektion**“)  
→ Fähigkeit zur eigenen Reproduktion
- Verursacht **Schaden** (Verletzung von → Schutzzielen)
- Kann sich beim Kopieren in andere Programme ggf. selbst verändern („**mutieren**“)

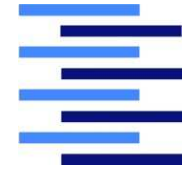




# Was ist ein Wurm?

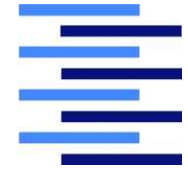
Unscharfe Abgrenzung zum Virus! Versuch nach [CE]:

- Ein **eigenständiges** Programm (evtl. Skript)
- Kann selbstständig in andere IT-Systeme eindringen („**Infektion**“) → Fähigkeit zur eigenen Reproduktion
- Verursacht **Schaden** (Verletzung von → Schutzzielen)
- Kann sich beim Reproduzieren ggf. selbst verändern („**mutieren**“)
- **Verbreitung** über
  - gezielte Angriffe (z.B. „Stuxnet“ 2010: Windows-Schwachstellen, siehe <https://de.wikipedia.org/wiki/Stuxnet> ) oder
  - E-Mail – Anhänge (z.B. „ILOVEYOU“ 2000: VB-Skript)



# Wurm-Beispiel: ILOVEYOU

- 4. Mai 2000: **Verbreitung** von Südostasien über Europa in die USA
- **E-Mail** mit
  - dem Betreff „I LOVE YOU“
  - dem Body „kindly check the attached LOVELETTER coming from me.“
  - dem Anhang „LOVE-LETTER-FOR-YOU.TXT.VBS“
- Öffnen des Anhangs → Ausführen des **VB-Skripts**
- **Wirkungen:**
  - Versendet sich selbst an alle Adressen aus dem Outlook-Adressbuch
  - Löscht Dateien
  - Versucht, weiteren Code von einer Website zu laden
  - Modifiziert die Registry
  - ...
- Bisher in **82 Varianten** aufgetreten!



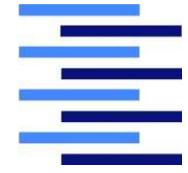
# Was ist ein Trojanisches Pferd („Trojaner“)?

- Ein **eigenständiges** Programm (evtl. Skript), das zusätzlich zur angegebenen Funktionalität weitere **beabsichtigte**, aber **verborgene Funktionen** ausführt (ggf. durch Zusatzcode in mp3/jpg – Dateien).
- Verursacht **Schaden** (Verletzung von → Schutzzielen) oder wird zur „**Fernsteuerung**“ von Rechnern benutzt (z.B. „Bot-Netze“ zum Versand von Spam-Mails)
- **Angriffsbeispiel:**
  - Download und Installation eines "Freeware"-Programms aus dem Internet
  - Ein Programm zum Mitprotokollieren von Tastatureingaben – insbesondere Passwörtern – wird unbemerkt zusätzlich installiert
  - Geheime Daten werden bei Aktivierung einer Internetverbindung versendet

# Malware – Schutzmaßnahmen

- **Vorsicht** bei der Ausführung von fremden Dateien (auf Vertrauenswürdigkeit der Quelle achten – z.B. Signierter Code)!
- Einsatz von **Virensclannern** mit aktuellen Virenmustern (zentral und dezentral)!
- Direkte Ausführung von **Mailanhängen** sperren
- **Makroausführung** nur nach Rückfrage
- **Zugriffsbeschränkungen** für Benutzer („need-to-know“-Prinzip)
- **Smartcards** statt Passwörter verwenden (kein Mitprotokollieren möglich)





# Techniken zur Entdeckung von „Malware“

## 1. Prüfung auf eine „Signatur“

- Signatur (hier): Bekannte Bit-Sequenz im Code

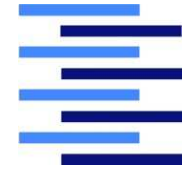
Problem: **Obfuscation** („Verdunkelung“)

- Code reordering
- Junk insertion
- Packing

ggf. in Kombination mit **Selbstmodifikation**  
(„Metamorphic Code“)

→ Code muss von Virensclannern „normalisiert“ werden

→ Forschungsthema!



# Techniken zur Entdeckung von „Malware“

## 2. Prüfung auf Veränderungen

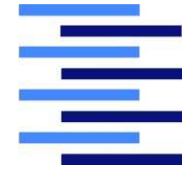
- Dateilängen
- Dateiinhalt (über berechneten Hashwert)

## 3. Prüfung auf Anomalien

- Ungewöhnliches Programmverhalten (über Logfiles)
- Zusätzliche Dateien / Benutzer
- Ungewöhnliche Konfiguration (über Parametereinstellungen)
- Angriffsversuche („Intrusion Detection“ – Systeme, z.B. *SNORT*)

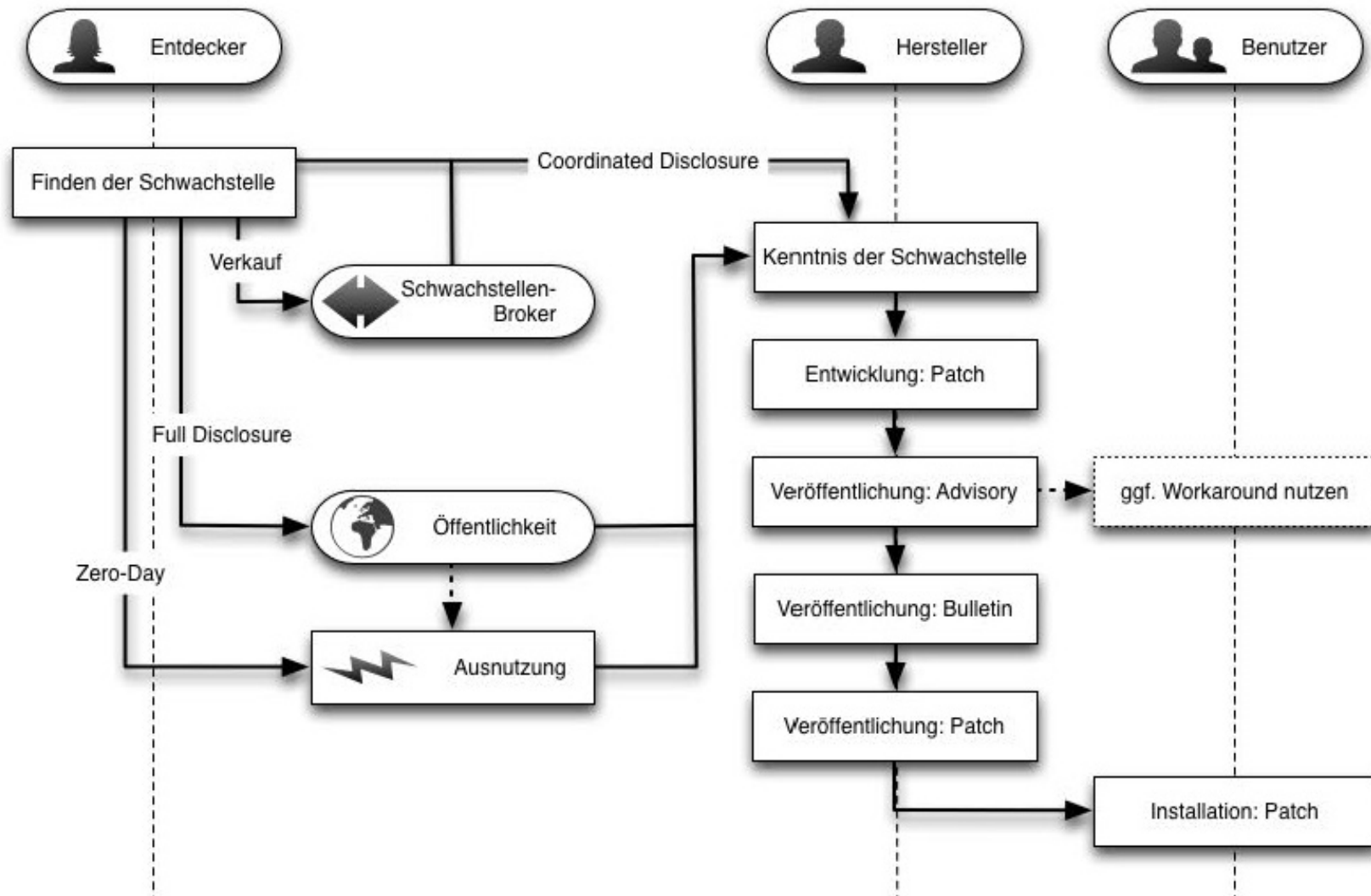
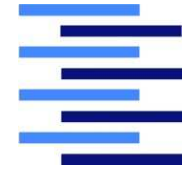
# Kapitel 2

## Bedrohungen und Angriffe



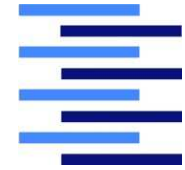
1. Rechtslage
2. Angreifer
3. Typische Angriffstechniken
4. Malware
5. **Schwachstellen-Sammlungen**

# Lebenszyklus einer Schwachstelle





# CVE ("Common Vulnerabilities and Exposures")



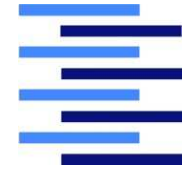
- Herstellerübergreifender Industriestandard zur einheitlichen **Benennung von öffentlich bekannten Schwachstellen** und zur Sammlung der darüber verfügbaren Informationen
- Liste (aktuell über 97.000 Einträge): <http://cve.mitre.org>
- *Beispiel-Suche nach "Buffer":*

## Search Results

There are **9802** CVE entries that match your search.

Name	Description
<a href="#">CVE-2018-7409</a>	In unixODBC before 2.3.5, there is a buffer overflow in the unicode_to_ansi_copy() function in DriverManager/__info.c.
<a href="#">CVE-2018-7284</a>	A Buffer Overflow issue was discovered in Asterisk through 13.19.1, 14.x through 14.7.5, and 15.x through 15.2.1, and Certified Asterisk through 13.18-cert2. When processing a SUBSCRIBE request, the res_pjsip_pubsub module stores the accepted formats present in the Accept headers of the request. This code did not limit the number of headers it processed, despite having a fixed limit of 32. If more than 32 Accept headers were present, the code would write outside of its memory and cause a crash.
<a href="#">CVE-2018-7254</a>	The ParseCaffHeaderConfig function of the cli/caff.c file of WavPack 5.1.0 allows a remote attacker to cause a denial-of-service (global buffer over-read), or possibly trigger a buffer overflow or incorrect memory allocation, via a maliciously crafted CAF file.
<a href="#">CVE-2018-7253</a>	The ParseDsdiffHeaderConfig function of the cli/dsdiff.c file of WavPack 5.1.0 allows a remote attacker to cause a denial-of-service (heap-based buffer over-read) or possibly overwrite the heap via a maliciously crafted DSDIFF file.
<a href="#">CVE-2018-7247</a>	An issue was discovered in pixHtmlViewer in prog/htmlviewer.c in Leptonica before 1.75.3. Unsanitized input (rootname) can overflow a buffer, leading potentially to arbitrary code execution or possibly unspecified other impact.

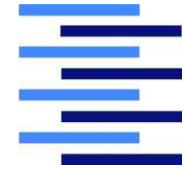
# Die 25 gefährlichsten Programmierfehler



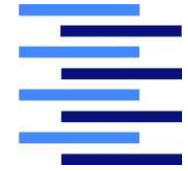
Quelle:

- **CWE (“Common Weakness Enumeration”)**
- Ein Projekt unterstützt von der National Cyber Security Division of the US Department of Homeland Security
- Ziel: Klassifikation von sicherheitsrelevanten Programmierfehlern („security bugs“)
- Beiträge erwünscht (Community-basiert)!
- <http://cwe.mitre.org>
- Aktueller Stand: 2011

# Die 25 gefährlichsten Programmierfehler



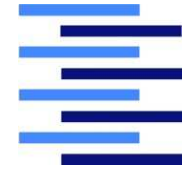
Rank	Score	ID	Name
[1]	93.8	<a href="#">CWE-89</a>	Improper Neutralization of Special Elements used in an SQL Command (' <b>SQL Injection</b> ')
[2]	83.3	<a href="#">CWE-78</a>	Improper Neutralization of Special Elements used in an OS Command (' <b>OS Command Injection</b> ')
[3]	79.0	<a href="#">CWE-120</a>	Buffer Copy without Checking Size of Input ('Classic <b>Buffer Overflow</b> ')
[4]	77.7	<a href="#">CWE-79</a>	Improper Neutralization of Input During Web Page Generation (' <b>Cross-site Scripting</b> ')
[5]	76.9	<a href="#">CWE-306</a>	Missing <b>Authentication</b> for Critical Function
[6]	76.8	<a href="#">CWE-862</a>	Missing <b>Authorization</b>
[7]	75.0	<a href="#">CWE-798</a>	Use of Hard-coded <b>Credentials</b>
[8]	75.0	<a href="#">CWE-311</a>	Missing <b>Encryption</b> of Sensitive Data
[9]	74.0	<a href="#">CWE-434</a>	Unrestricted <b>Upload of File</b> with Dangerous Type
[10]	73.8	<a href="#">CWE-807</a>	Reliance on <b>Untrusted Inputs</b> in a Security Decision
[11]	73.1	<a href="#">CWE-250</a>	Execution with <b>Unnecessary Privileges</b>
[12]	70.1	<a href="#">CWE-352</a>	<b>Cross-Site Request Forgery</b> (CSRF)



# Die 25 gefährlichsten Programmierfehler

Rank	Score	ID	Name
[13]	69.3	<a href="#">CWE-22</a>	Improper Limitation of a Pathname to a Restricted Directory (' <b>Path Traversal</b> ')
[14]	68.5	<a href="#">CWE-494</a>	Download of Code Without <b>Integrity Check</b>
[15]	67.8	<a href="#">CWE-863</a>	Incorrect <b>Authorization</b>
[16]	66.0	<a href="#">CWE-829</a>	Inclusion of Functionality from <b>Untrusted Control Sphere</b>
[17]	65.5	<a href="#">CWE-732</a>	<b>Incorrect Permission Assignment</b> for Critical Resource
[18]	64.6	<a href="#">CWE-676</a>	Use of <b>Potentially Dangerous Function</b>
[19]	64.1	<a href="#">CWE-327</a>	Use of a Broken or <b>Risky Cryptographic Algorithm</b>
[20]	62.4	<a href="#">CWE-131</a>	Incorrect Calculation of <b>Buffer Size</b>
[21]	61.5	<a href="#">CWE-307</a>	Improper <b>Restriction of Excessive Authentication Attempts</b>
[22]	61.1	<a href="#">CWE-601</a>	URL Redirection to Untrusted Site (' <b>Open Redirect</b> ')
[23]	61.0	<a href="#">CWE-134</a>	Uncontrolled <b>Format String</b>
[24]	60.3	<a href="#">CWE-190</a>	<b>Integer Overflow</b> or Wraparound
[25]	59.9	<a href="#">CWE-759</a>	Use of a One-Way Hash without a <b>Salt</b>

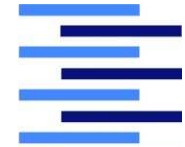
# Die „Top 10“ gefährlichsten Programmierfehler für Webanwendungen



## Quelle:

- **OWASP („Open Web Application Security Project“)**
- Eine Stiftung, gegründet von Firmen und Non-Profit-Organisationen
- Ziel: Unterstützung bei der Entwicklung von **sicheren** Webanwendungen
- Beiträge erwünscht (Community-basiert)!
- <https://www.owasp.org>
- Aktueller Stand: 2017

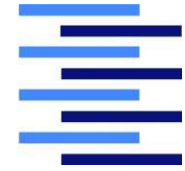
# Die OWASP-„Top 10“ des Jahres 2013/2017



OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

*Komplette Beschreibungen inkl. Gegenmaßnahmen → Pub*

# Ende des 2. Kapitels: Was haben wir geschafft?



## Bedrohungen und Angriffe

1. Angreifer-Typen und Vorgehensweisen
2. Typische Angriffstechniken
  - Einbruch in IT-Systeme durch
    - Ausnutzung von programmtechnischen Schwachstellen
    - Maskierung / Manipulation der Kommunikation
    - Erschleichen von Zugangsberechtigungen
  - IT-Systeme gezielt ausschalten („Denial-of-Service“)
    - Überschwemmung mit Verbindungsanfragen oder anderen Dienstanforderungen
4. Malware
5. Schwachstellen-Sammlungen