# Sicherheit in Software

Fabian Cordt und Friedrich Eder

3. Juni 2011

## Allgemeines

- Begriffserklärung
- Woher

## Die 19 Todsünden

- 1 - Teil
- 2 - Teil
- 3 - Teil

## Was kann passieren

- Probleme beim Porgramm
- Durch Lücken die entstehen

# Sicherheitslücken

- Fehler in der Software
- Schadprogramme können in den Rechner gelangen
- viele verschiedene Angriffsmöglichkeiten
- viel Schadcode im Umlauf

# Wodurch kommt es zu Sicherheitlücken

- Lücken entstehen durch Fehler im Programm
- 1000 Programmzeilen/1 Fehler
  - Hervorgerufen durch Zeitdruck
  - Programmiersprachen (C, Assembler,..)
  - Kenntnisse fehlen
- Große Gemeinde die Spass hat Fehler auszunutzen

# Die 19 Todsünden 1 - Teil

- Buffer Overruns
- Format-String-Problems
- Integer Overflows
- SQL Injection
- Command-Injection
- Failing to Handle Errors

# Die 19 Todsünden 2 - Teil

- Cross-Site Scripting
- Failing to Protect Network Traffic
- Use of Magic URLs and Hidden Form Fields
- Improper Use of SSL and TLS
- Use of Weak Passwords-Based Systems
- Failing to Store and Protect Data Securely

# Die 19 Todsünden 3 - Teil

- Information Leakage
- Improper File Access
- Trusting Network Name Resolution
- Race Conditions
- Unauthenticated Key-Exchange
- Cryptographically Strong Random Numbers
- Poor Usability

# Was kann passieren

- Das Programm ...
  - gibt falsches Ergebnis
  - beschädigt Daten
  - stürzt ab
  - bring Betriebsystem zum abstürzt

# Was kann passieren

- Durch Lücken kommt es zu ...
  - abstürzen von Programmen
  - Datenbeschädigung
  - Ressourcenraub
  - Datenraub

# Allgemeines

- Mehrere Arten
- Eingabedaten werden über die Grenzen der Variablen geschrieben
- Angreifer kann Speicherbereiche verändern, auf die er keinen Zugriff haben sollte
- Programmiersprachenabhängig
- Kann vom Benutzer ohne Quellcode nicht gesehen werden

# Gefährdete Sprachen

- Sprachen, die “Pointerarithmetik” erlauben
- c
- c++

# Beispiel

```
void doSomething(const char *string)
{
        char buffer[128];
        strcpy(buffer, string);
        //do something
}

int main(int argc, const char* argv[])
{
        char *input = charEinlesen();
        doSomething(input);
        return 0;
}
```

# Stack-based Buffer Overflow

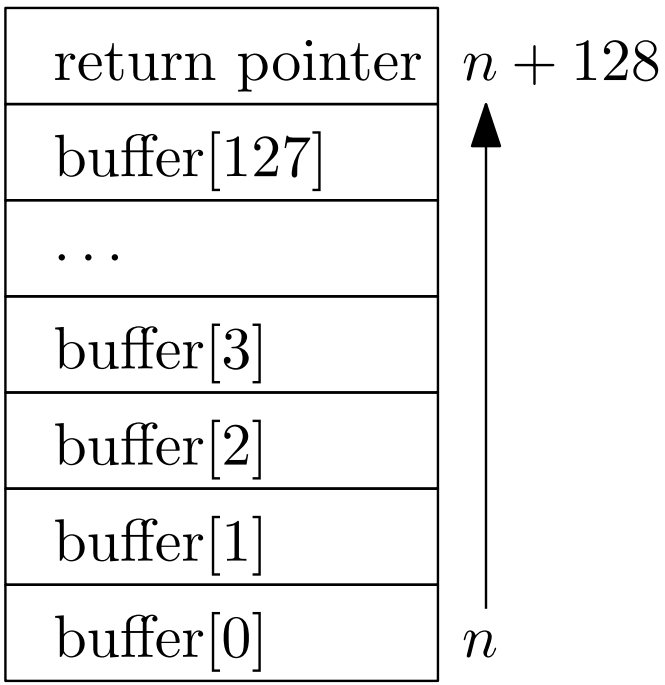


Abbildung: Der Stack

- Funktionsinterne Variablen werden auf dem Stack abgelegt
- Return Pointer liegt ganz "oben" auf dem Stack

# Angriffsmöglichkeit und Probleme

- Interne Variablen können überschrieben werden
- Return Pointer kann verändert werden
- eigene Eingaben können als Programmcode ausgeführt werden
- Angreifer kann sich Berechtigungen des Programmes beschaffen

# Was kann man als Programmierer dagegen tun

- Längenüberprüfung im Programm einfügen
- Warnungen des Compilers beachten
- geeignete Funktionen zum Einlesen von Daten verwenden
- eine aktuelle Version des Compilers verwenden

# Gegenmanahmen des Compilers

- Stack Protection
- “Canary” wird nach Return Pointer auf den Stack gelegt
- Bietet keinen 100% Schutz
- Angreifer kann Canary erraten
- gcc: ProPolice

# Allgemeines

- Durch das Programm selbst
  - sehr aktuell
  - große Schäden
  - Veränderung von Daten
  - Diebstahl von Daten
  - Beschädigung von Daten
  - Systemmissbrauch (Ressourcennutzung,...)
  - Datenbankveränderung

# Angriffe

- Benutzereingaben kommen in den SQL-Interpreter
- es gibt so genannte „Cheat Sheets“
- viele Foren mit genauer Erklärungen
- auch groe Firmen sind immer wieder betroffen

# Zugriff auf falsche Daten

- SELECT * FROM TABLE WHERE USERNAME = 'input' mit Eingabe "something' OR '1'='1"
- SELECT * FROM TABLE WHERE USERNAME = 'something' OR '1'='1'

# Was kann man dagegen tun

- Daten die Beteutung für den SQL-Interpreter nehmen
- Maskieren $\rightarrow$ Escapen
- Daten vom SQL-Interpreter fernhalten
- Privilegien genau abmessen
- Web Application Firewalls

# Quelle

19 Deadly Sins of Software Security
by Michael Howard,David LeBlanc and John Viega
ISBN:0072260858
citeseer1
citeseer2
galileocomputing

# Ende

- Sie erreichen uns unter
  - ederfri[at]stud.sbg.ac.at
  - cordtfa[at]stud.sbg.ac.at
- Wir danken für die Aufmerksamkeit