6 – Security

Quellen:

- *„Das Sicherheitsloch – Buffer-Overflows und wie man sich davor schützt“, Kalnik, Schröter und Strobel, c‘t Artikel (23/01)*
- *„Buffer Overflows für Jedermann, Felix Opatz (2005)*
- *„Tutorials – Buffer Overflow Tutorial Teil1: Grundlagen“, http://www.online-tutorials.net*

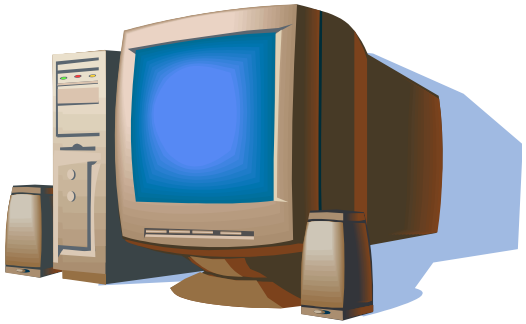
- *Wikipedia*
- *„Cross-Site-Scripting: Datenklau über Bande“, D. Bachfeld, 2003*
- *http://virtualforge.de/vmovie/xss_lesson_1/xss_selling_platform_v1.0.html*

- *„Die Passwortknacker“, c‘t Artikel (3/2013)*

Security
Überblick

- **Einleitung**
- **Buffer Overflows**
- **Cross Site Scripting**
- **Password Cracking**

Warum spielt Security eine wichtige Rolle?



- Rasant wachsende Geräteanzahl
- Großflächige Vernetzung von Geräten
- IT-Systeme übernehmen immer mehr Aufgaben in verschiedenen Anwendungsbereichen
- Zunahme von Speicherung und Austausch sensibler Daten
- Neue Technologien und Einsatzgebiete
  - > z.B. Radio Frequency Identification(RFID) und Near Field Communication (NFC)
    -> Smartpone als Geldbörse / Schlüsselbund

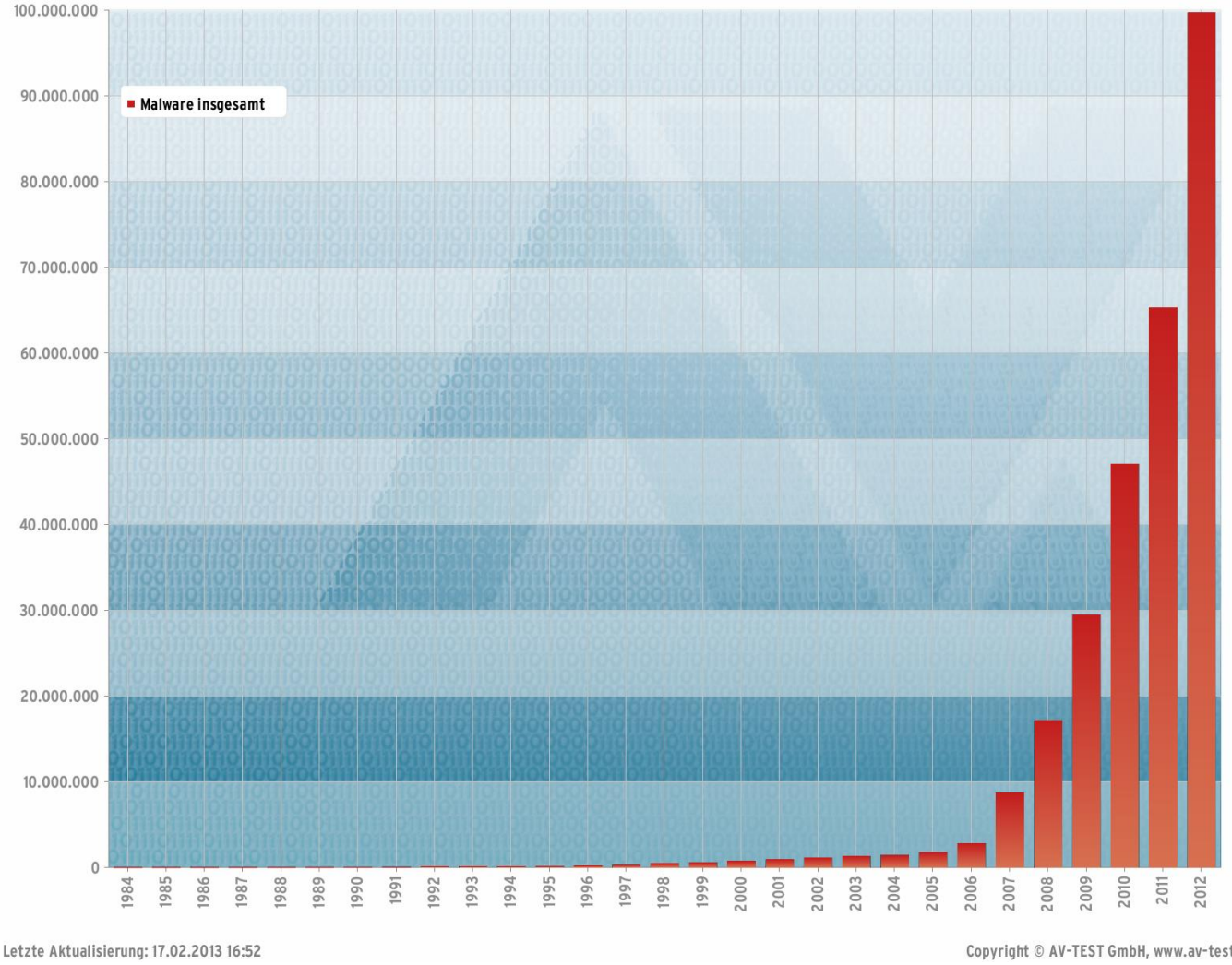
Warum spielt Security eine wichtige Rolle?

- Softwareprodukte werden immer komplexer
- Potenziellen Angreifern stehen zahlreiche, leicht bedienbare Tools zur Verfügung
  - > Angreifer brauchen häufig keine detaillierte Kenntnisse über das System
  - > Stark steigende Anzahl von Angriffen und Security-Vorfällen

Warum spielt Security eine wichtige Rolle?

- Schadsoftware und Angriffe verursachen enorme Kosten

- Angreifer können sehr unterschiedliche Ziele verfolgen
  - > Schaden anrichten
  - > Daten stehlen
  - > Ressourcen nutzen

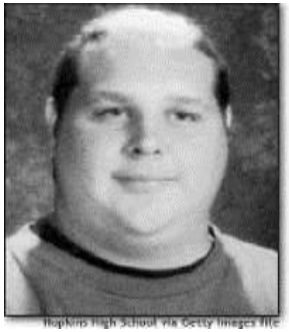

100.000.000
90.000.000
80.000.000
70.000.000
60.000.000
50.000.000
40.000.000
30.000.000
20.000.000
10.000.000
0
Malware insgesamt
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
Letzte Aktualisierung: 17.02.2013 16:52
Copyright © AV-TEST GmbH, www.av-test.org

Security
Wo liegt das Problem?

Profil eines Angreifers um 2000:

- Männlich
- Zwischen 14 und 34 Jahre
- Computer-affin
- Keine Freundin






No Commercial Interest !!!

Source: Raimund Genes

Working hours: approx. 2 minutes/day to manage Botnet
Monthly earnings: $6,800 on average
Daily Activities:

- Chatting with people while his bots make him money
- Recently paid $800 for an hour alone in a VIP room with several dancers

Job Description:

- Controls 13,000+ computers in more than 20 countries
- Infected Bot PCs download Adware then search for new victim PCs
- Adware displays ads and mines data on victim's online browsing habits.
- Bots collect password, e-mail address, SS#, credit and banking data
- Gets paid by companies like TopConverting.com, GammaCash.com, Loudcash, or 180Solutions.

Angriff:
Ein nicht autorisierter Zugriff bzw. Zugriffsversuch auf ein IT-System

Passiver Angriff:
Zugriff auf vertrauliche Informationen

- *Verlust der Vertraulichkeit*
- *Beispiele: Abhören von Leitungen, Lesen von geheimen Dateien*

Aktiver Angriff:
Modifikation von Datenobjekten oder Systemressourcen

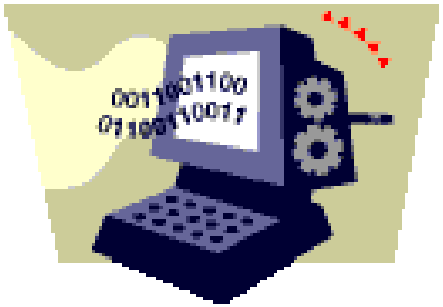
- *Verlust der Integrität / Verfügbarkeit)*
- *Beispiele: Verändern / Löschen von Dateien oder IP-Paketen, Überschwemmen mit TCP-Verbindungsanfragen („Denial-of-Service")*

Security
Funktionsweisen von Angriffen

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

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

Hintergrund



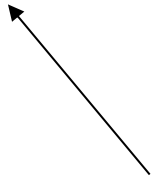
- Buffer Overflows (Pufferüberlauf) sind häufige Ursache für Verwundbarkeiten und Sicherheitslücken von Anwendungen
  - > Bekanntes Beispiel: Microsoft IIS Webserver
  - > Viele Software Produkte (Windows und Unix) betroffen

- Erste Angriffe dieser Art tauchten bereits 1998 auf
- Anwendungs- und Systemprogrammierer verantwortlich für das Auftreten solcher Schwachstellen
  - > Besonders häufig sind C Programme betroffen
  - > Verwendung von Funktionen wie *strcpy()* oder *gets()* kritisch, da diese nicht kontrollieren, ob der Speicherbereich an der Zieladresse groß genug ist

Was passiert bei einem Buffer Overflow?

- Ist der Puffer zu klein, werden die nachfolgenden Datenfelder im Speicher überschrieben
- In den meisten Fällen führt dies zu einem Programmabsturz, denn
  - > Variablen werden mit unsinnigen Werten überschrieben und das Programm kann deshalb nicht ordnungsgemäß fortgeführt werden
  - > Die Rücksprungadresse einer Funktion stimmt nicht mehr (Programm springt an eine zufällige Speicheradresse, was in der Regel zu einem „Speicherzugriffsfehler" führt)

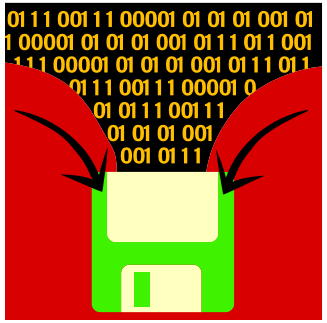
```
cmd = lies_aus_netz();
do_something(cmd);
.....
int do_something(char* InputString) {
        char buffer[4];
        strcpy (buffer, InputString);
        ...
        return 0;
}
```



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

Wie lässt sich ein Buffer Overflow ausnutzen?

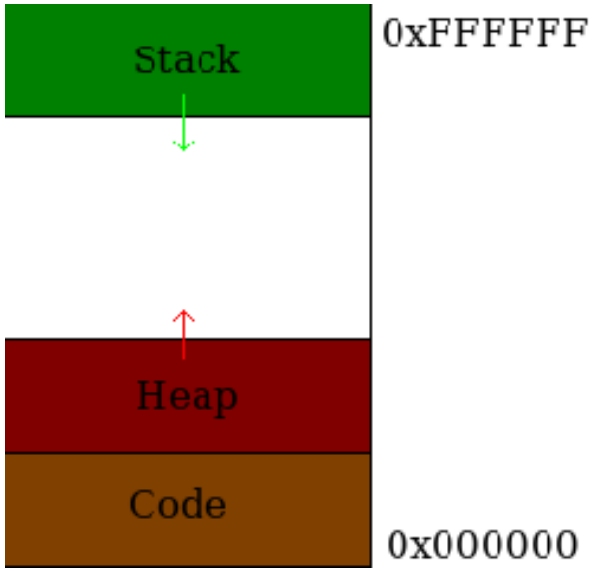
- Meist geht es um das Einschleusen von Code
  - > Nutzt die Tatsache, dass Code und Daten beim von Neumann-Rechner im selben Speicher liegen
  - > Hierzu kann die Rücksprungadresse einer Funktion durch bewusstes Überschreiben des Speichers geändert werden



- Dafür kann die Eingabe einer Zeichenkette genügen
- Das Ausnutzen von Sicherheitslücken eines Programms wird als „Exploit" bezeichnet
- Hierzu muss die Schwachstelle jedoch bekannt sein

Speicherverwaltung eines Programms

- einem Programm wird Speicher zugewiesen, welcher in der Regel aus drei Segmenten besteht
  - > Code-Segment: enthält die eigentlichen Maschinenbefehle
  - > Heap-Segment: wird für dynamische Speicherallokation verwendet (z.B. *malloc()*)
  - > Stacksegment: Zwischenspeicher für lokale Variablen und Prozessorregister



Speicherverwaltung eines Programms

- C Programme legen Übergabeparameter einer Funktion, die Rücksprung-adresse und lokale Variablen auf dem Stack ab
- Verwaltung der Stackpointer über Register ESP, EBP, EIP
  - > ESP – Stack Pointer: zeigt auf das aktuelle Ende des Stacks
  - > EBP – Base Pointer: zeigt auf den Anfang des Stacks zum Zeitpunkt des Betretens einer Funktion
  - > EIP – Instruction Pointer: enthält Speicheradresse der nächsten auszuführenden Instruktion
- Datenbereich einer Funktion wird als „Stackframe" bezeichnet
- Auf dem Stack können sich mehrere Stackframes befinden (z.B: bei verschachtelten Funktionen)

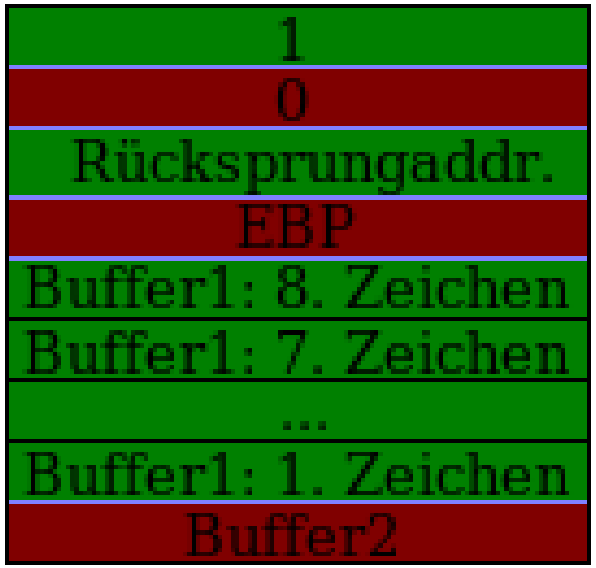
Speicherverwaltung eines Programms

- C Beispielprogramm:

```
int TestFunktion(int a, int b)
{
        char Buffer1[8];
        char Buffer2[16];

        return;
}

int main()
{
        TestFunktion(0, 1);
         return 0;
}
```



Manipulation des Stacksegments

- Wird eine Benutzereingabe auf einer lokalen Variablen innerhalb einer Funktion abgelegt, ohne dass die Länge der Eingabe geprüft wird, kann hierdurch der EIP manipuliert werden
- Dadurch lässt sich der Programmablauf beeinflussen
- Die Position des EIP im Stack und damit die benötigte Länge der Eingabe lässt sich durch Analyse des Programms ermitteln (z.B. mittels disassemble im Debugger)
- Benutze ein Exploit-Programm zum automatischen Erzeugen der Eingabe (verwende Ein-/Ausgabeumlenkung)

DEMO 2: Manipulation des EIP

Manipulation des Stacksegments

- Gewöhnlich soll nicht an eine andere Programmstelle gesprungen werden, sondern eigener Code ausgeführt werden: „Payload“
- Dieser kann im Stackframe der Funktion abgelegt und über die Eingabe eingeschleust werden
- Probleme:

1. Wie lässt sich die Rücksprungadresse (Anfang des eigenen Codes) bestimmen

2. Sonderzeichen müssen versteckt werden, z.B. „\0“ bei *gets()*

3. Aufruf von Funktionen und Adressierung von Eingabedaten

Manipulation des Stacksegments

- Mögliche Lösungen:

1. Wie lässt sich die Rücksprungadresse (Anfang des eigenen Codes) bestimmen
   > Liegt in der Nähe des Stackanfangs, welcher sich ermitteln lässt:

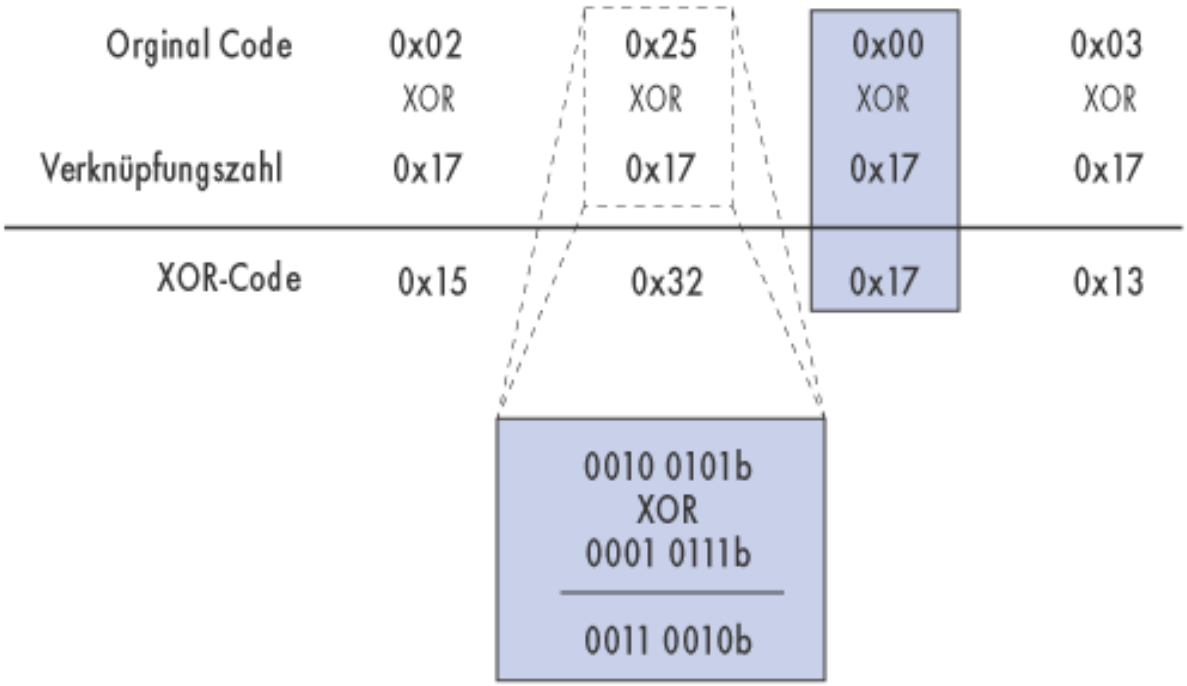
   ```
   __asm__ ("movl %esp, %eax")
   ```

   > NOPs als „Landing-Pad" einfügen

2. Sonderzeichen müssen versteckt werden, z.B. „\0" bei *gets()*
   > Kodiere die Kommandos über die XOR-Verküpfung mit einer Zahl (die selbst nicht im Code vorkommen darf)
   > Dekodierung kann über wenige Assembler-Befehle im eingeschleusten Code realisiert werden

Manipulation des Stacksegments

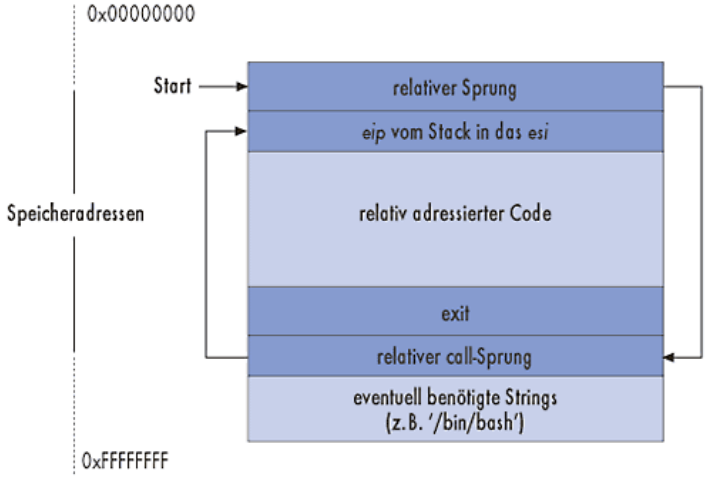
- Mögliche Lösungen:

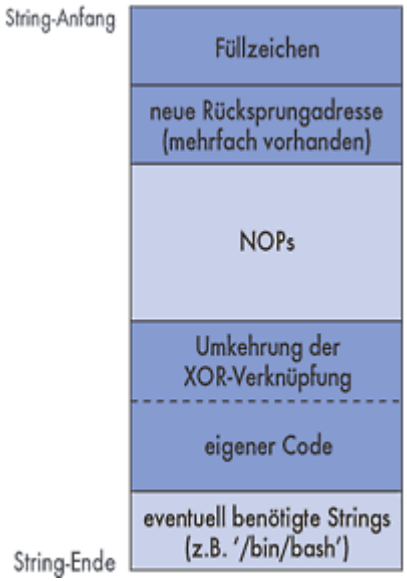


Manipulation des Stacksegments

- Mögliche Lösungen:

3. Adressierung von Eingabedaten
   > Um auf eigene Daten zugreifen zu können, kann eine relative Adressierung verwendet werden
   > Hierzu benötigte Adresse des Stringanfangs kann durch einen relativen Sprung und einen simulierten Funktionsaufruf ermittelt werden





Manipulation des Stacksegments

- Mögliche Lösungen:

  3. Aufruf von Funktionen

     > Wichtige Funktionen, wie z.b. „/bin/sh“ zum Erzeugen einer Shell, können über das Auslösen von Software-Interrupts („0x80“) und die Funktion „execve“ (zum Starten externer Programme) aufgerufen werden

```
…
mov     $1, %eax
int     $0x80
…
```

- Ein passendes Exploitprogramm kann also z.B. einen Buffer Overflow nutzen, um eine Shell zu öffnen

Gegenmaßnahmen

- Sicherheitsbewusste Programmierung
- Tools, die das Überschreiben der Rücksprungadresse verhindern, z.B. unter Linux:
  - > StackGuard
  - > StackShield
  - > libsafe

Quelle: bodhost.com

Hintergrund

- XSS bezeichnet Ausnutzen einer Sicherheitslücke in Webanwendungen
- Ziel sind in der Regel sensible Benutzerdaten oder die Manipulation von Webinhalten
- „Cross Site“, da der Angriff zwischen verschiedenen Aufrufen einer Webseite stattfindet (in der Regel der gleichen Seite)
- Meist werden für die Angriffsart Script-Sprachen wie JavaScript genutzt
- In der Regel nicht zielgerichtet auf ausgewählte Personen
- Verschleierung und Tarnung schadhafter URLs
- Durch script-fähige Mailprogramme auch per Email möglich

Hintergrund

- Weitere zum Teil ähnliche Angriffsmöglichkeiten für Webanwendungen sind
  - > Cross Frame Scripting (XFS)
  - > Cross Site Request Forgery (CSRF)
  - > Cross Site Tracing
  - > Cross Site Cooking

- Insbesondere Kombination aus Schwachstellen im Server und im Client sind gefährlich
  - > Ermöglichen Angreifern Übernahme von Benutzer-Sessions bis hin zur Kontrolle über den Client Rechner

Funktionsweise

- XSS ist eine Art HTML-Injection
- Tritt auf, wenn eine Webanwendung Benutzereingaben annimmt und ohne Prüfung an den Browser weitersendet

- Klassisches Beispiel ist die Übergabe von Parametern an ein serverseitiges Skript, das eine dynamische Webseite erzeugt, z.B.:
  - > Eingabeformular einer Webseite
- Häufig werden hierbei manipulierte Hyperlinks und Cookies genutzt

Funktionsweise

- Dem Angreifer gelingt es, schadhaften Code (z.B. JavaScript) an den Browser des Opfers zu senden
- Für den Browser stammt der Code scheinbar von einer vertrauenswürdigen Seite
- Hierdurch lassen sich z.B. die Cookies des Opfers und seine Sessiondaten auslesen und missbrauchen
  - > Einschränkung: der Angriff muss im gleichen Zeitraum erfolgen, da die Sessiondaten sonst ggf. nicht mehr gültig sind

Video: XSS Explanation

Hintergrund

- Speicherung von Passwörtern ist an vielen Stellen notwendig für die Benutzerauthentifizierung
- In der Regel werden Passwörter aus Sicherheitsgründen nicht im Klartext, sondern kodiert abgelegt
- Hierzu werden meist sogenannten Hash-Funktionen verwendet
- Werden Passwörter gestohlen, müssen diese erst noch entschlüsselt werden
  - > Wegen der speziellen Eigenschaften der Hash-Funktionen muss dies meist durch „ausprobieren“ erfolgen, da es keine Dekodierungsvorschrift gibt

Hintergrund

- Eine Hashfunktion f(x) hat folgende Eigenschaften
  - f(x) bildet eine beliebige aber wohldefinierte Eingabe auf einen Wert fester Länge (= Hashwert) ab
  - f(x) sollte injektiv sein, ist es aber meist nicht
  - f(x) ist nicht umkehrbar (für kryptologische Hashfunktionen)
- Beispiel: MD5 (*md5()* in PHP)
  - `MATSE` → *`0241739d4c2596e4f12a2fd531de6163`*

Hintergrund

- Dem Knacken von Passwörtern müssen nicht zwangsläufig böse Absichten zu Grunde liegen
  - > Kann auch zum bewussten Aufspüren schwacher Passwörter genutzt werden und der Erhöhung der Passwortsicherheit dienen
  - > Wird häufig als eine Art „Sport" betrieben bei dem regelmäßig Wettbewerbe ausgetragen werden („Crack WM")
  - > Kann genutzt werden, um vergessene Passwörter ermitteln

- Tools und Programme zum Knacken von Passwörtern nicht zwangsläufig illegal
  - > Passwortknacker „John the Ripper" war sogar auf Sicherheits-CDs des BSI (Bundesamt für Sicherheit in der Informationstechnik) enthalten

Hintergrund

- Enorme Rechenpower, die beim Passwortknacken zum Einsatz kommt, spielt eine wichtige Rolle
  - > Macht einen Teil der Faszination aus
  - > Hauptursache ist die Möglichkeit der Nutzung massiv paralleler Rechnerarchitekturen wie GPUs
- Graphikkarten eignen sich architekturbedingt sehr gut für die schnelle Ausführung von Programmen und Algorithmen für das Knacken von Passwörtern
  - > Sich ständig wiederholende Operationen mit ganzen Zahlen spielen in der Kryptographie eine große rolle und könne von GPUs extrem schnell ausgeführt werden
  - > Leistungen im TFlop/s Bereich stehen nun auch dem Heimanwender zur Verfügung

Vorgehensweisen

- Eingesetzte Software hat sich in den letzten Jahren deutlich verbessert
  - > Selbst Passwörter, die vor ein paar Jahren noch als sicher galten, können heutzutage mit wenig Aufwand geknackt werden
- Fast alle Angriffe beruhen mittlerweile auf Wörterbüchern
  - > Basis für ein Cracker-Wörterbuch ist in der Regel ein Duden
  - > Für zahlreich Sprachen frei im Netz verfügbar

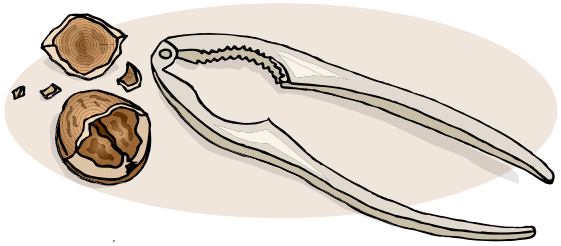
Vorgehensweisen

- Kontinuierliche Ergänzung von Einträgen
  - > Besonders wertvoll sind echte Passörter, die bereits entschlüsselt wurden
  - > Können mehrere 100 Mio. Einträge enthalten

- Klassische Wörterbuchattacken (durchgehen aller Einträge eines Wörterbuchs) werden meist nur noch zur Initialisierung verwendet
  - > Anschließend systematisches Modifizieren der Einträge, die gängige Passwortverbesserungen nachbilden

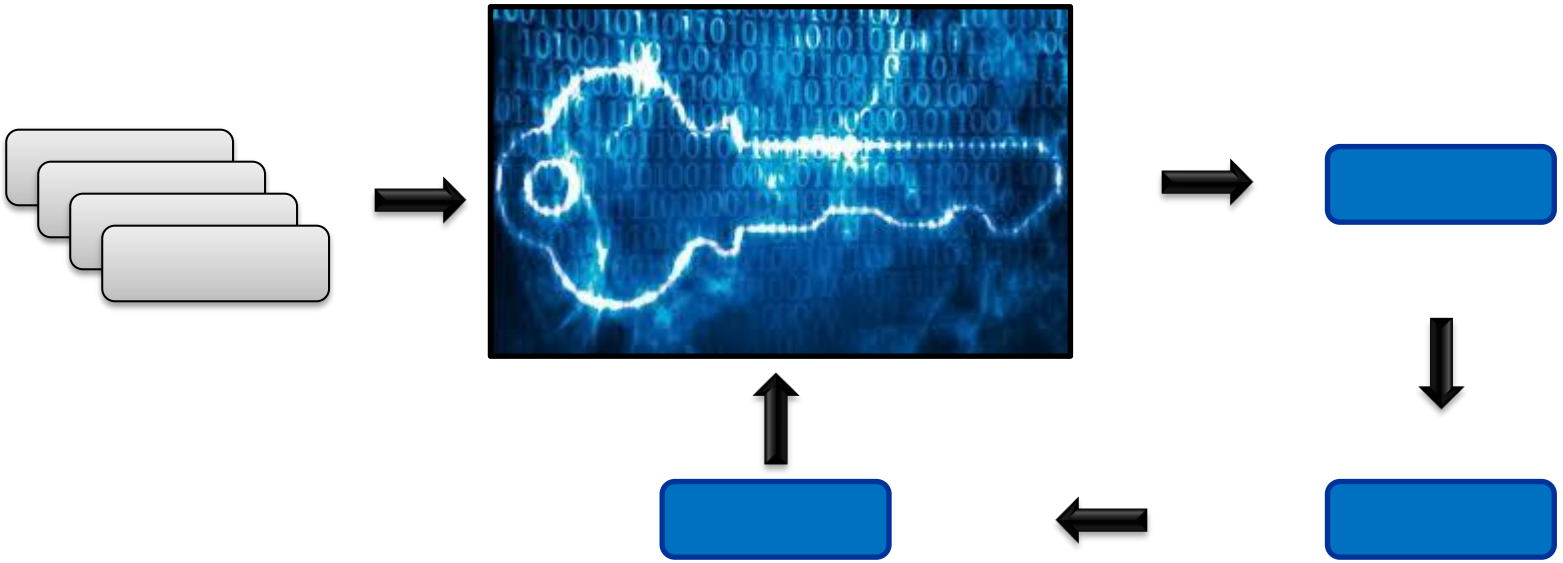
Vorgehensweisen

- Vorgegebene Regeln für Passwort-Variationen werden abgearbeitet
  - > Groß- und Kleinschreibung ändern, Zahlen anhängen
  - > Ersetzung bestimmter Buchstaben durch Zahlen („1337-Speak"): z.B: „l1nk3d1n" statt „LinkedIn"
  - > Spezielle Zeichenkombinationen, z.B. :-)
  - > Einbeziehen der Plattform, für die ein Passwort benutzt wird (z.B. Ebay)



Vorgehensweisen

- Geknackte Passwörter werden dynamisch mit eingebunden
  - > Lassen Rückschlüsse auf weitere Passwörter zu
  - > Bekannte Passwörter (oder Teile davon) werden miteinander kombiniert
  - > Analyse von Häufigkeiten und Verteilungen der Buchstaben



Vorgehensweisen

- Verwendung von Mustern bei der Passwortauswahl wird ausgenutzt
  - > Unter anderem auch Muster auf der Tastatur implementiert
  - > Verwendung von Markovketten oder z.B. Levenshtein Distanz

- Levenshtein Distanz
  - > Hilft bei der Erkennung von Mustern / Variationen
  - > z.B.: „Web“, „Web2“ -> erzeuge daraus neue Variationsregel

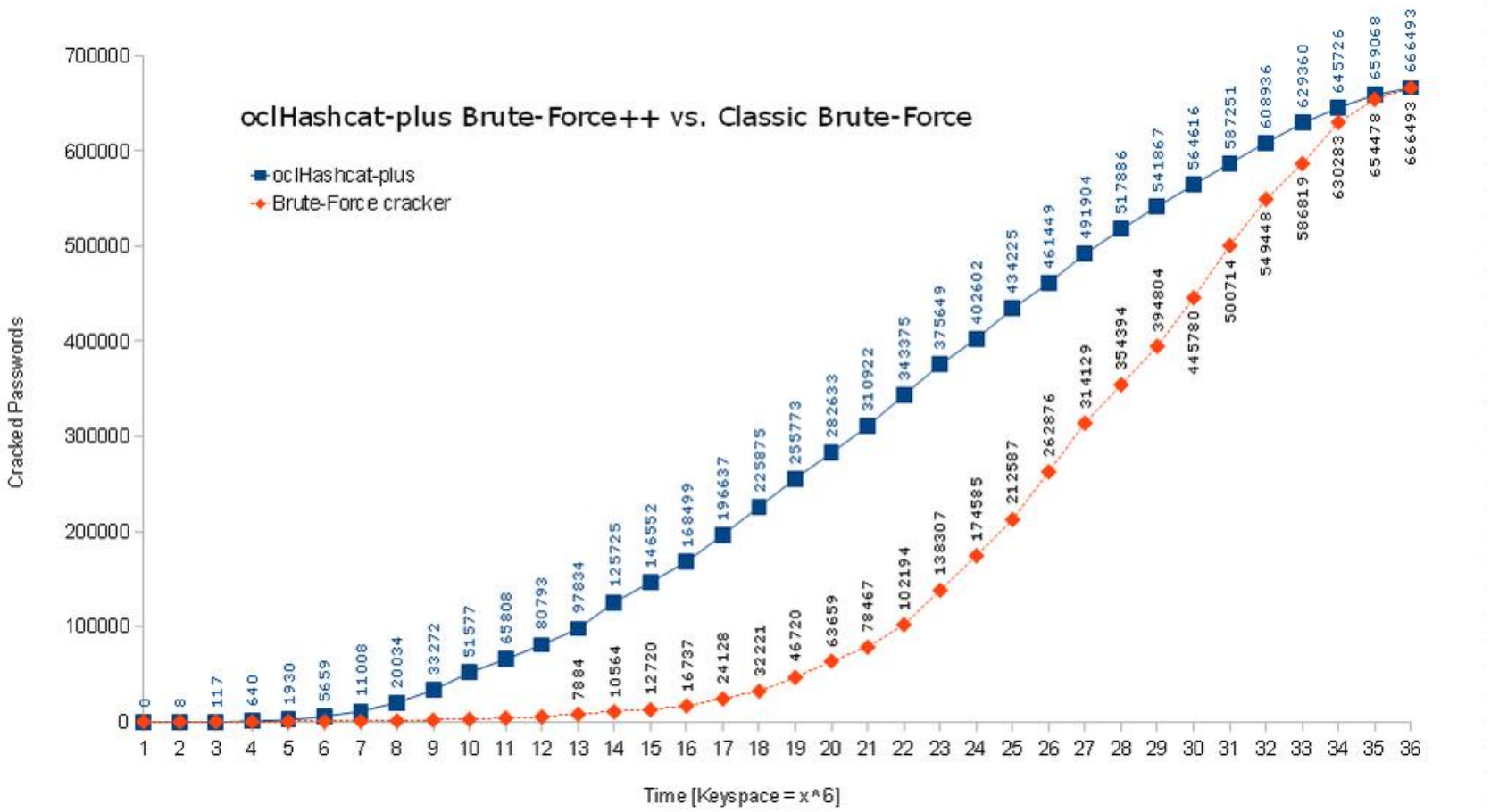
Vorgehensweisen

- Markov-Ketten/-Modell
  - > Arbeitet mit bedingten Wahrscheinlichkeiten
  - > Versuche, das Durchprobieren von Buchstabenkombinationen möglichst „intelligent“ zu machen
  - > Bsp.: „u“ folgt häufig auf „q“, Großbuchstaben erscheinen häufig am Wortanfang

$$P\left(X_{t+1} = s_{jt+1} \mid X_t = s_{jt}, \ldots, X_{t-n+1} = s_{jt-n+1}\right)$$

mit $S = \{s_1 \ldots s_m\}$, $t = 0,1,2,\ldots$

Vorgehensweisen



Vorgehensweisen

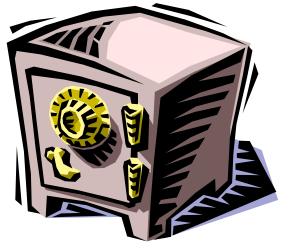
- Weitere Technik: Verwendung von „Rainbow-Tables“
  - > Hierbei werden riesige Tabellen mit potentiellen Passwörtern und deren Hashes im voraus erzeugt
  - > Tabellen ebenfalls im Netz zu finden
  - > Wird nicht mehr oft benutzt (da für kleinere (< 8 Zeichen) Passwörter kaum noch Vorteile gegenüber Wörterbuch-Ansatz und für längere (> 8 Zeichen) Passwörter kaum existierende Tabellen)

Vorgehensweisen

- Cracking findet in der Regel offline statt
  - > Online-Angriffe müssen mehr Hürden überwinden, z.B. penalties oder Sperrung des Accounts bei ungültiger Passworteingabe
  - > Zugang zu „Rohdaten" z.B. durch Sicherheitslücken in Webservern möglich
  - > Mit passender Hardware können für bestimmte Algorithmen wie MD5 oder SHA1 mehrere Milliarden Hashes pro Sekunde erzeugt werden

- Crack-Programme beherrschen häufig viele verschiedene Hash-Algorithmen
  - > Oft sogar automatische Erkennung, welcher Algorithmus zu Grunde liegt

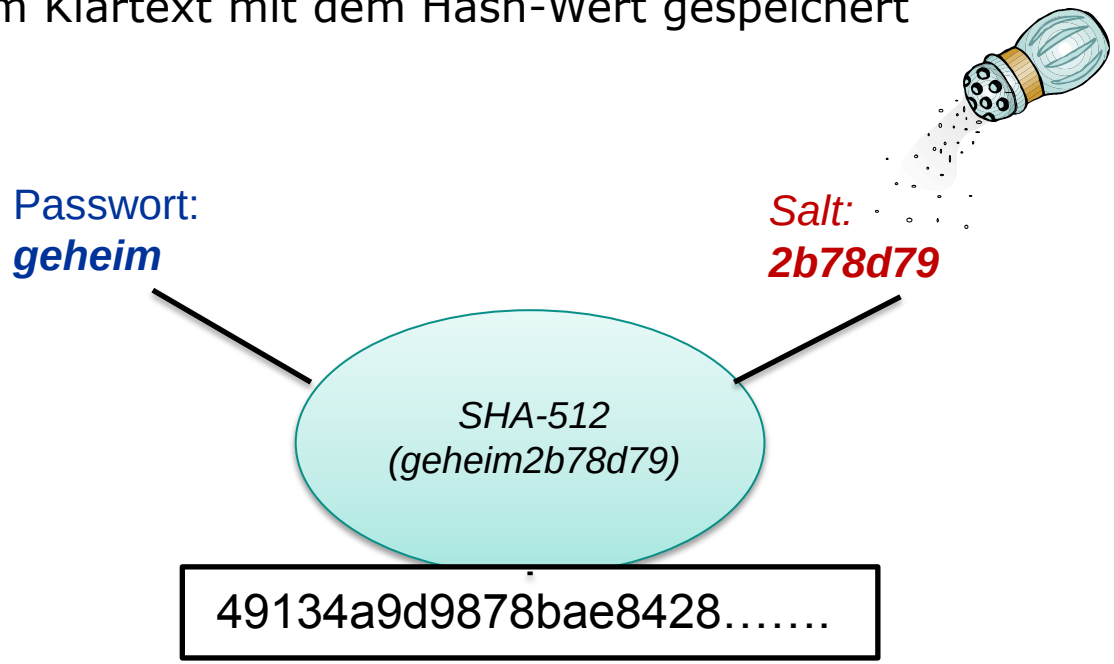
Verbesserung der Passwortsicherheit



- Neben bekannten Regeln für die Passwortwahl helfen
  - > Tools zur Passwort-Erzeugung
  - > Passwort-Safes zur Verwaltung von Passwörtern
  - > Nutzung von Wegwerfpasswörtern
  - > Wiederholte Anwendung von Hash-Funktionen
  - > Künstliche Verlängerung von Passwörtern durch das System / den Administrator („Salt“)

Verbesserung der Passwortsicherheit

- vor dem Hash-Vorgang wird ein Salt an das Passwort angehängt
- Salt wird im Klartext mit dem Hash-Wert gespeichert



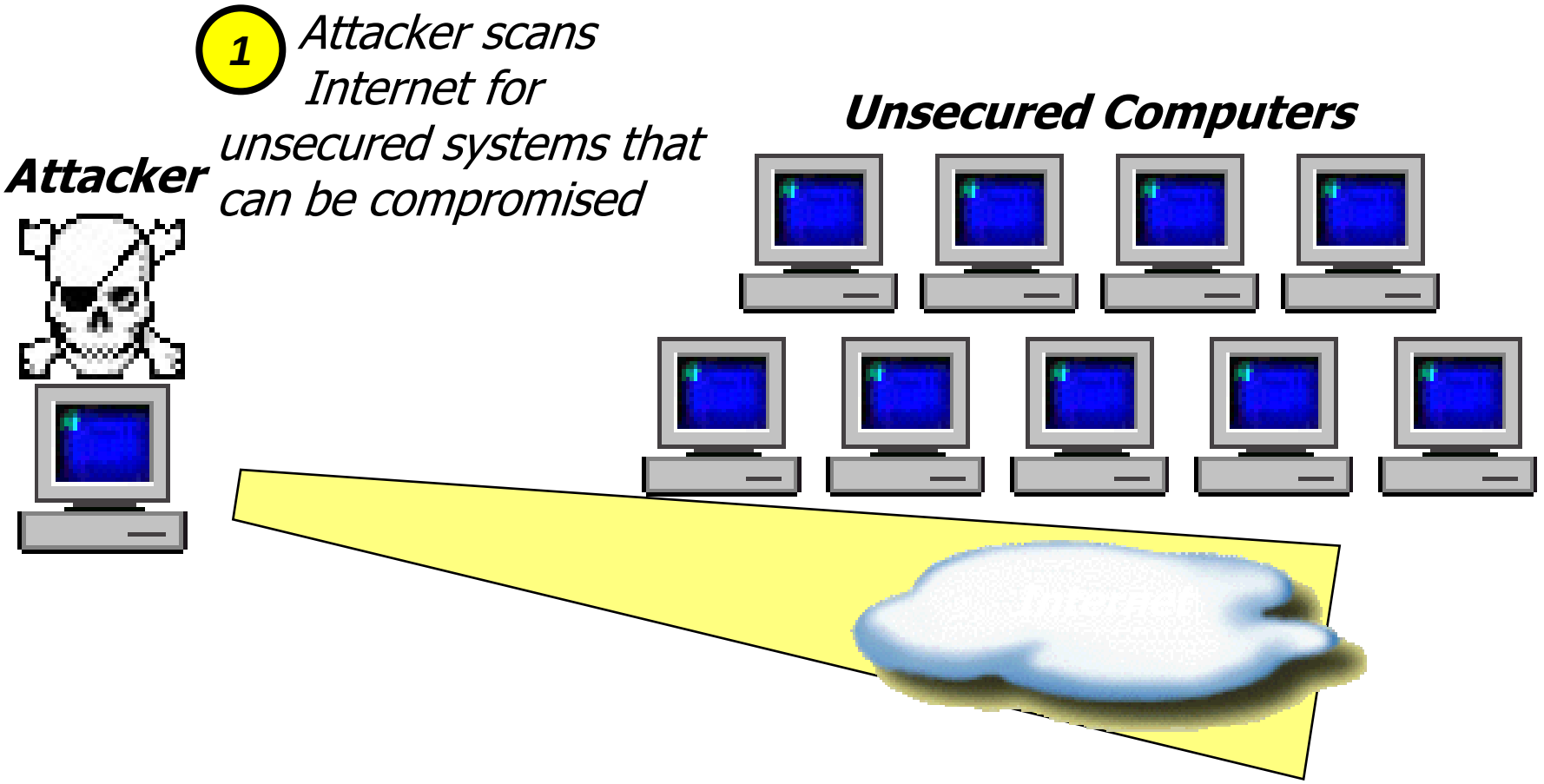
Security
Zombies und Botnets

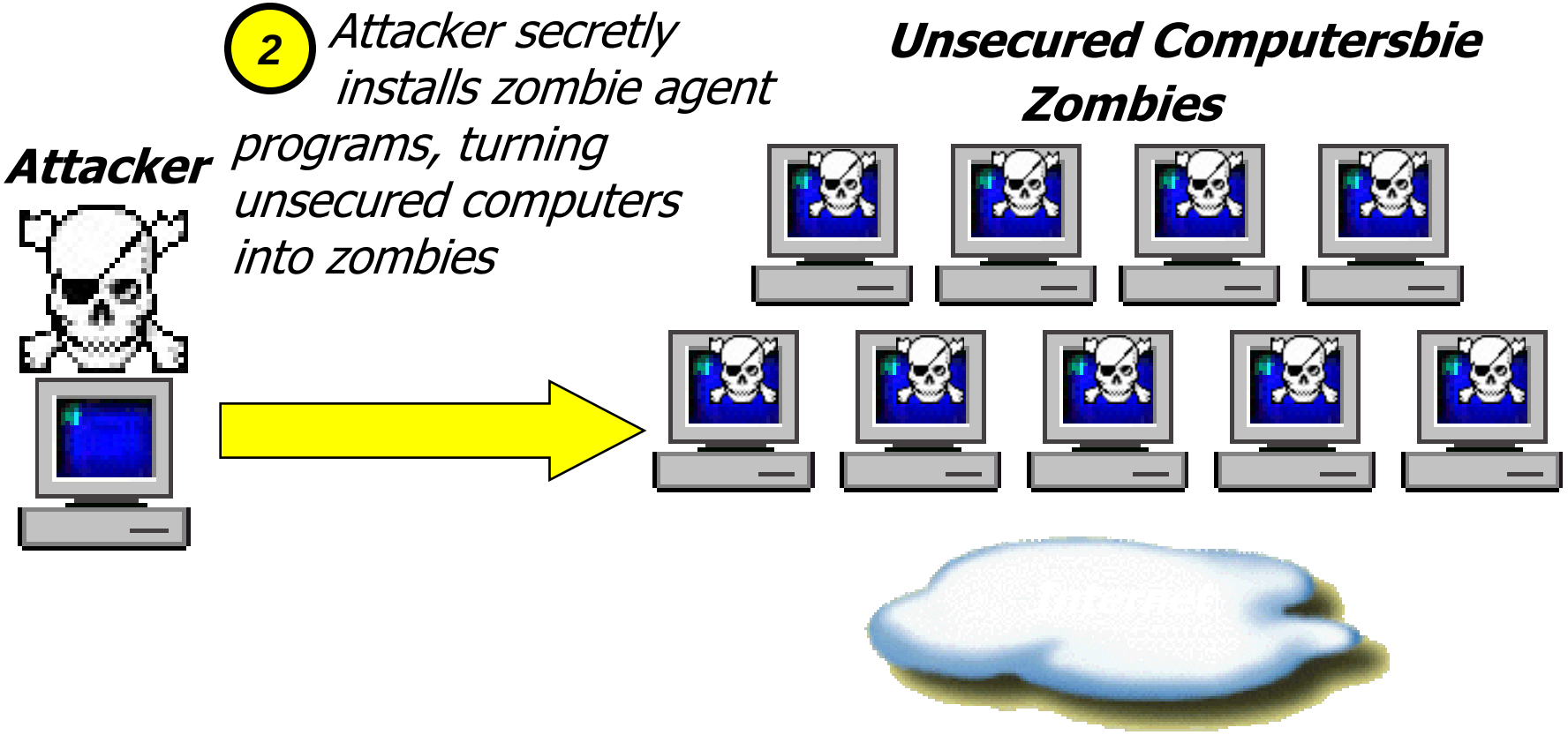
- Übernimmt eine Schadsoftware unbemerkt ungewollte Dienste auf einem vernetzten Rechner so ist dies schon ein Problem
- Integriert diese den infizierten Rechner in ein Zombie- oder Bot-Net, dann kann noch schlimmeres passieren
- Definition eines Zombi-Nets
  > a collection of compromised machines running programs, usually referred to as worms, Trojan horses, or backdoors, under a common command and control infrastructure.

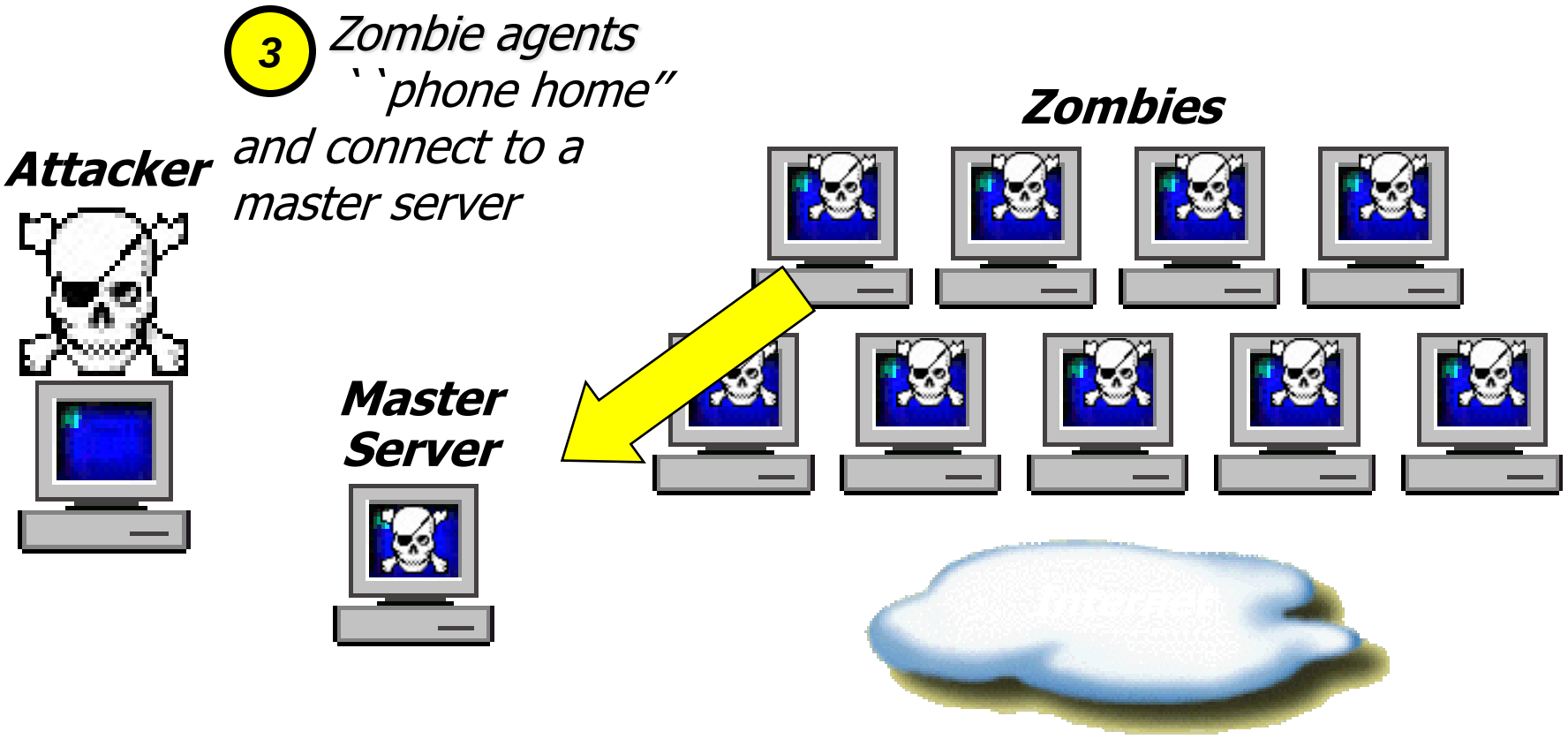
Wird häufig für spezifische Attacken genutzt

- E.g., DDoS, phishing, spamming, cracking

Beispiel: Die 18 Mio. geklauten eMail-Daten

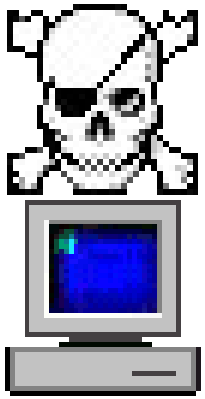

1
Attacker scans Internet for unsecured systems that can be compromised
Attacker
Unsecured Computers

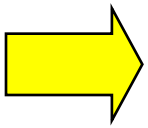

2
Attacker secretly installs zombie agent programs, turning unsecured computers into zombies
Attacker
Unsecured Computersbie Zombies


3
Zombie agents ``phone home'' and connect to a master server
Attacker
Master Server
Zombies

4 *Attacker sends commands to Master Server to launch a DDoS attack against a targeted system*

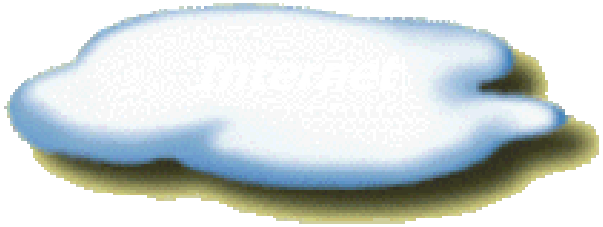
Attacker

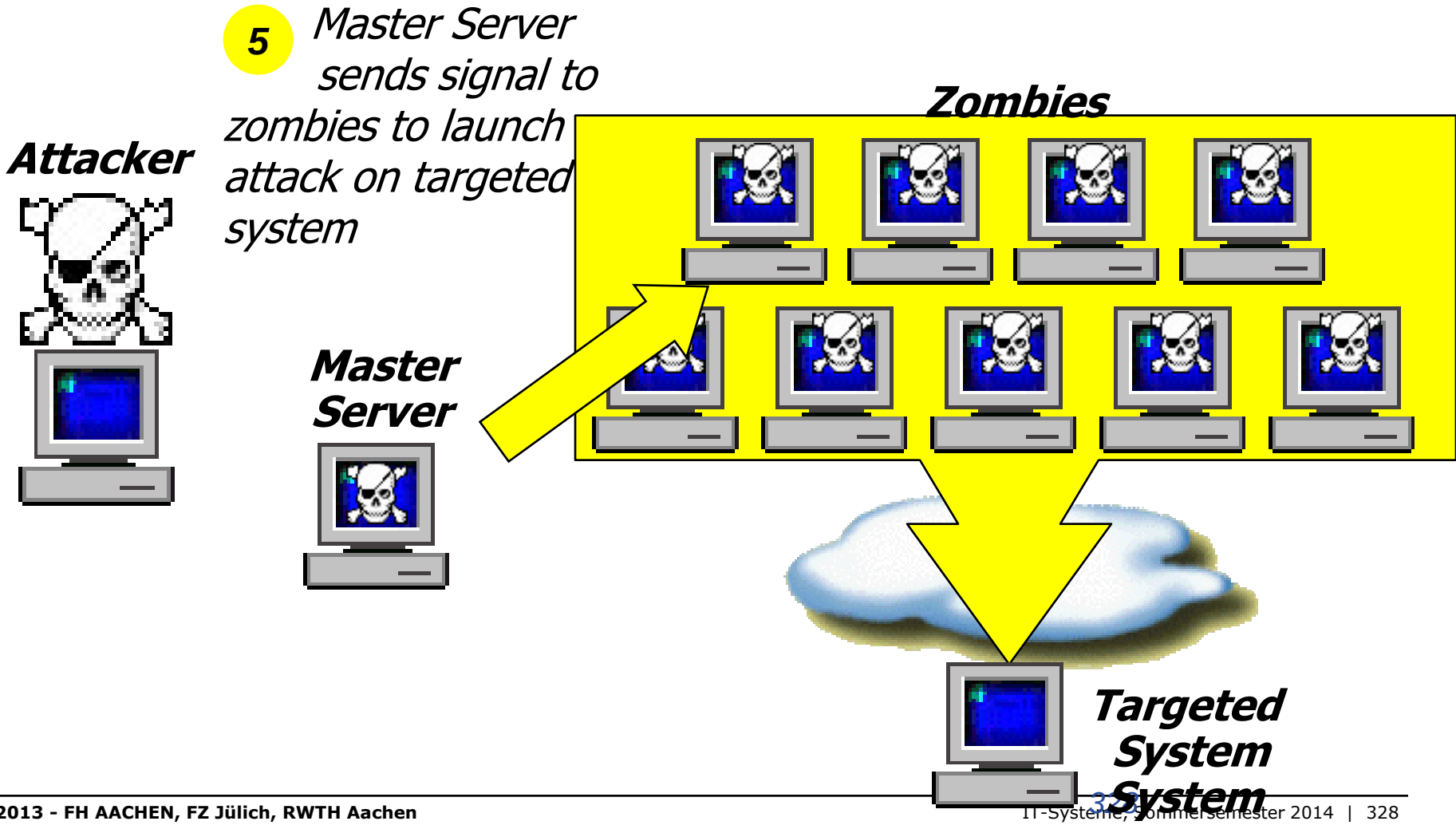


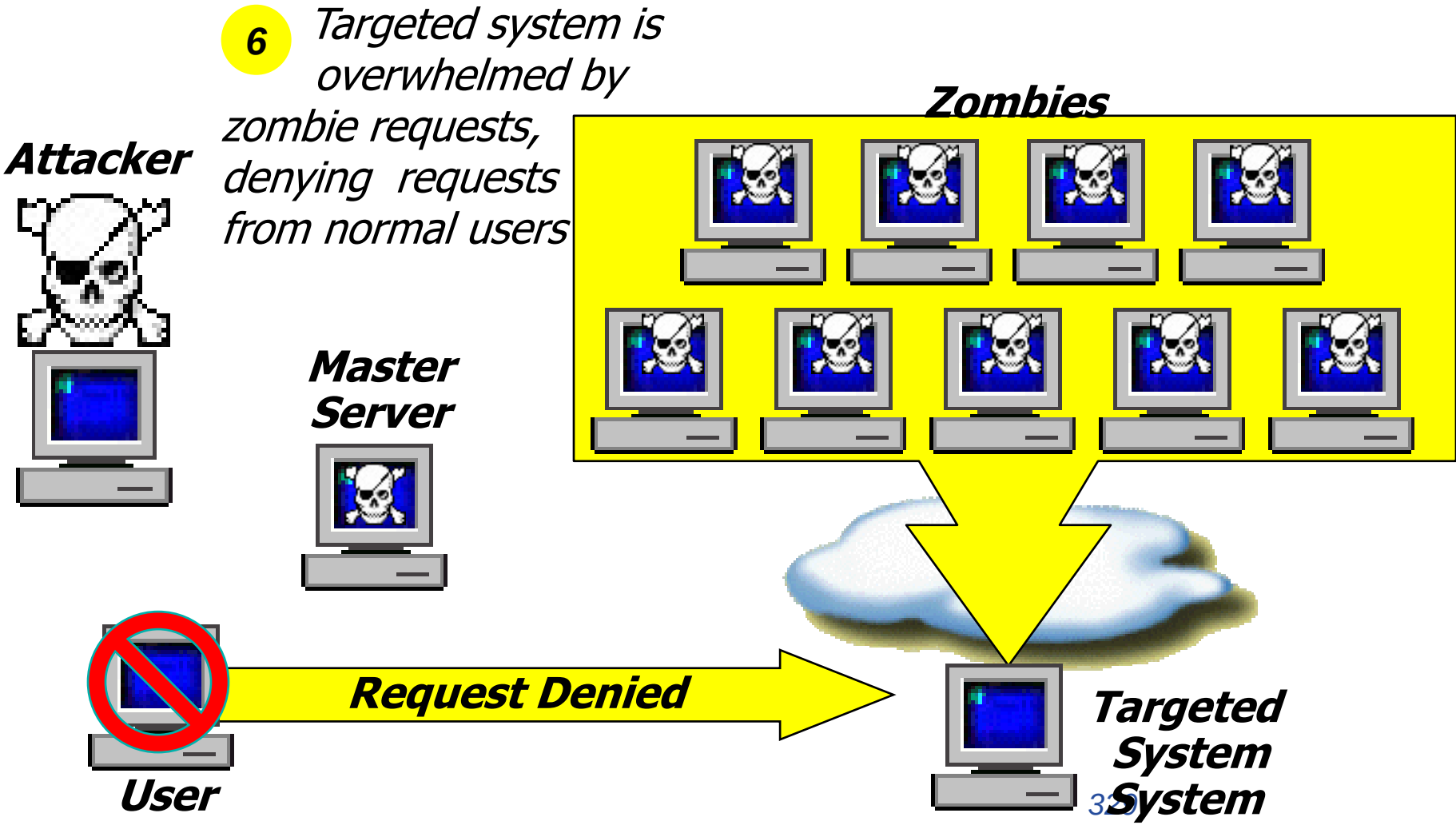


Master Server

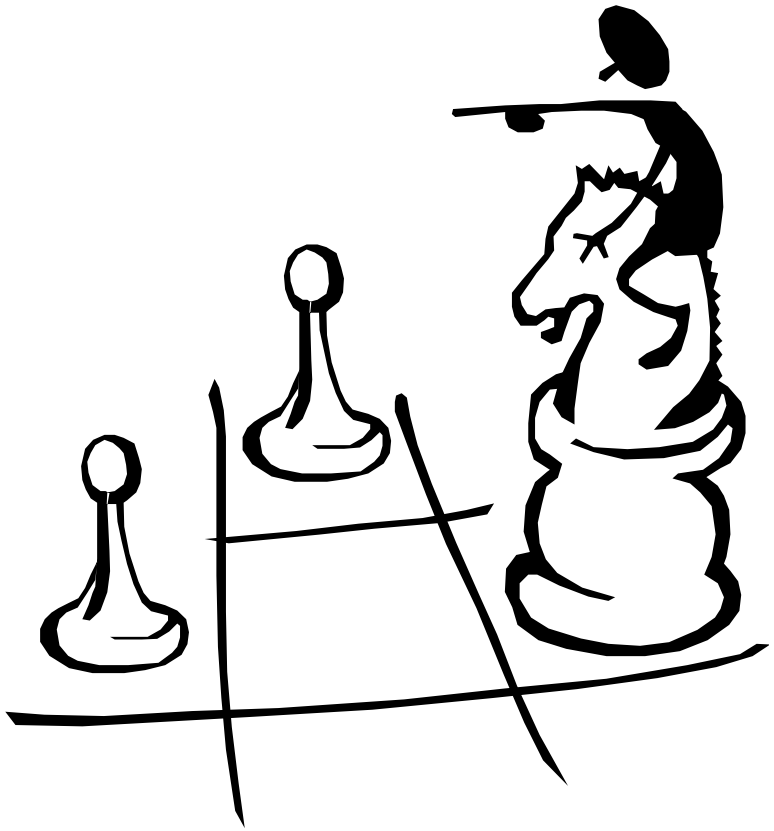
Zombies




5
Master Server sends signal to zombies to launch attack on targeted system
Attacker
Zombies
Master Server
Targeted System
System


6
Targeted system is overwhelmed by zombie requests, denying requests from normal users
Zombies
Attacker
Master Server
Request Denied
User
Targeted System
System

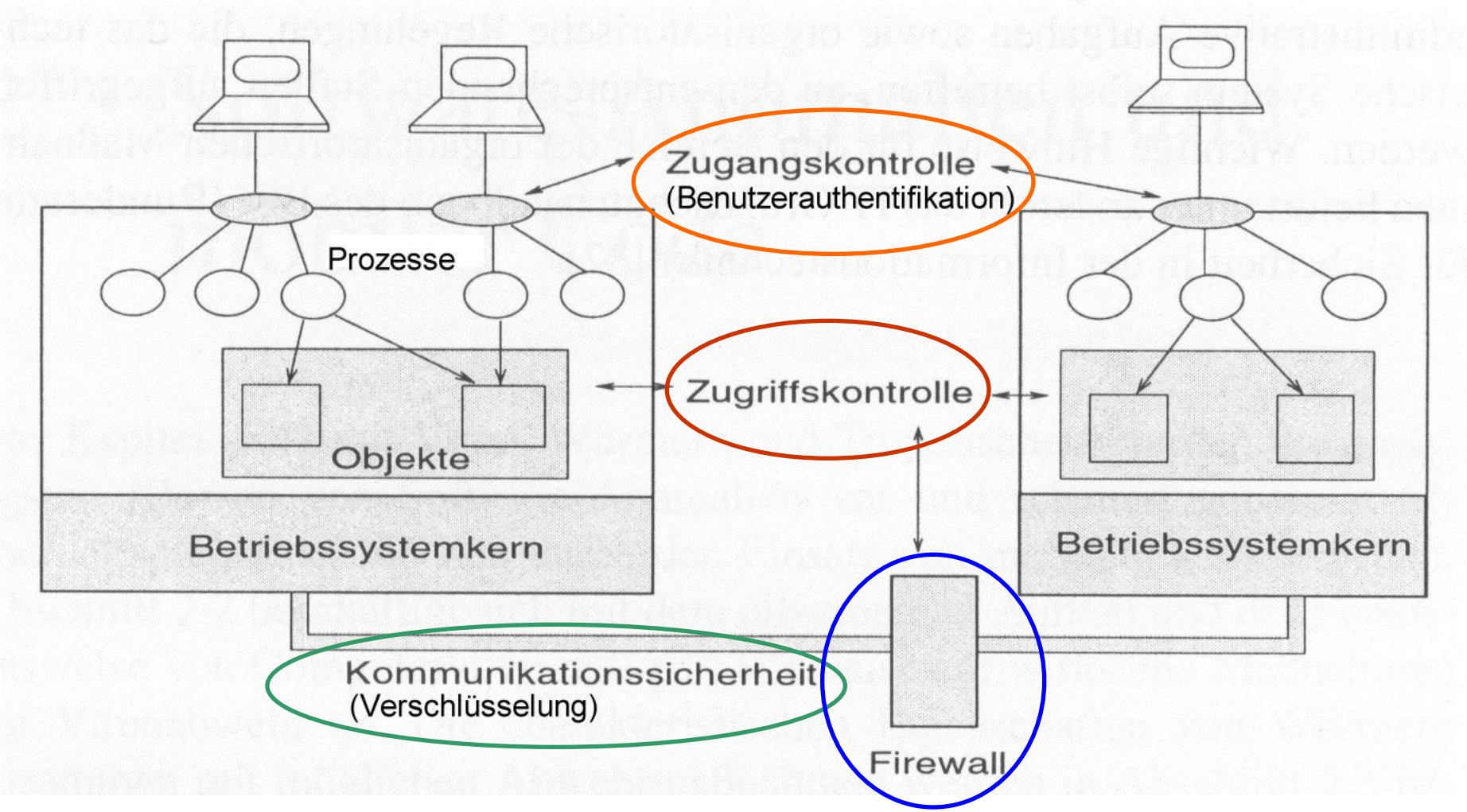
- Botnets arbeiten ähnlich, nur nutzen sie eine peer-to-peer-Struktur und keine zentral Steuerung
- Auch nutzen Sie Encrypting/authenticating



„Die Sicherheitsstrategie eines IT-Systems oder einer organisatorischen Einheit definiert alle technischen und organisatorischen Regeln, Verhaltensrichtlinien, Verantwortlichkeiten und Rollen sowie alle Maßnahmen, um die angestrebten Schutzziele zu erreichen."

Beispiele:

- *Systemverhalten (Speicherbereinigung, Beweissicherung, Rechteprüfung)*
- *Zugriffsrechte*
- *Administrationsrollen*
- *Benutzerverhalten (Passwortrichtlinien)*
- *Einsatz von spezieller Software (Virenscanner, Firewalls, Verschlüsselung)*
- *Absicherung von Räumen*
- *Datensicherungsmaßnahmen*


Zugangskontrolle
(Benutzerauthentifikation)
Prozesse
Zugriffskontrolle
Objekte
Betriebssystemkern
Betriebssystemkern
Kommunikationssicherheit
(Verschlüsselung)
Firewall

Zugangskontrolle

- *Authentifikation von Subjekten*
- *„Ist X wirklich derjenige, für den er sich ausgibt?"*

Zugriffskontrolle

- *Autorisation von Subjekten für den Zugriff auf bestimmte Objekte*
- *„Darf X auf Objekt Y zugreifen?"*
- *z.B. Standard: S. Godik, T. Moses: e**X**tensible **A**ccess **C**ontrol **M**arkup **L**anguage; OASIS Commitee Spezification, 2003*

Verschlüsselung

- *Anwendung kryptografischer Methoden*
- *„Wie kann X Informationen in der Form speichern, dass nur Y (autorisiert) darauf zugreifen kann?"*

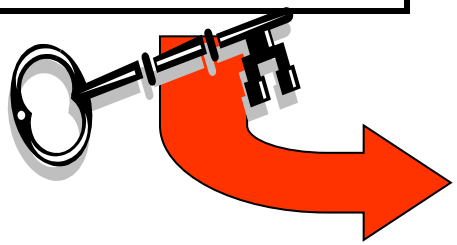
Firewall-Einsatz

- *Filtern von Netzzugriffen*
- *„Welche Anfragen / Verbindungswünsche lasse ich in das interne Netz hinein bzw. daraus heraus?"*

Security
Schutz durch Verschlüsselung

Klartext M

Dies ist ein unverschlüsselter Text



Verschlüsselung mit Schlüssel K_E*:*

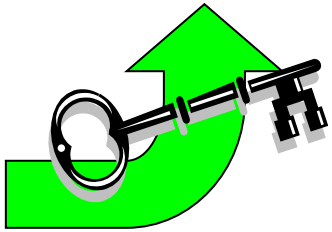
$E(M, K_E) = C$

Chiffretext C

pivlqfaymohrmq rjvpyüzmimgiif eom

Klartext M

Dies ist ein unverschlüsselter Text



Entschlüsselung mit Schlüssel K_D*:*

$D(C, K_D) = M$

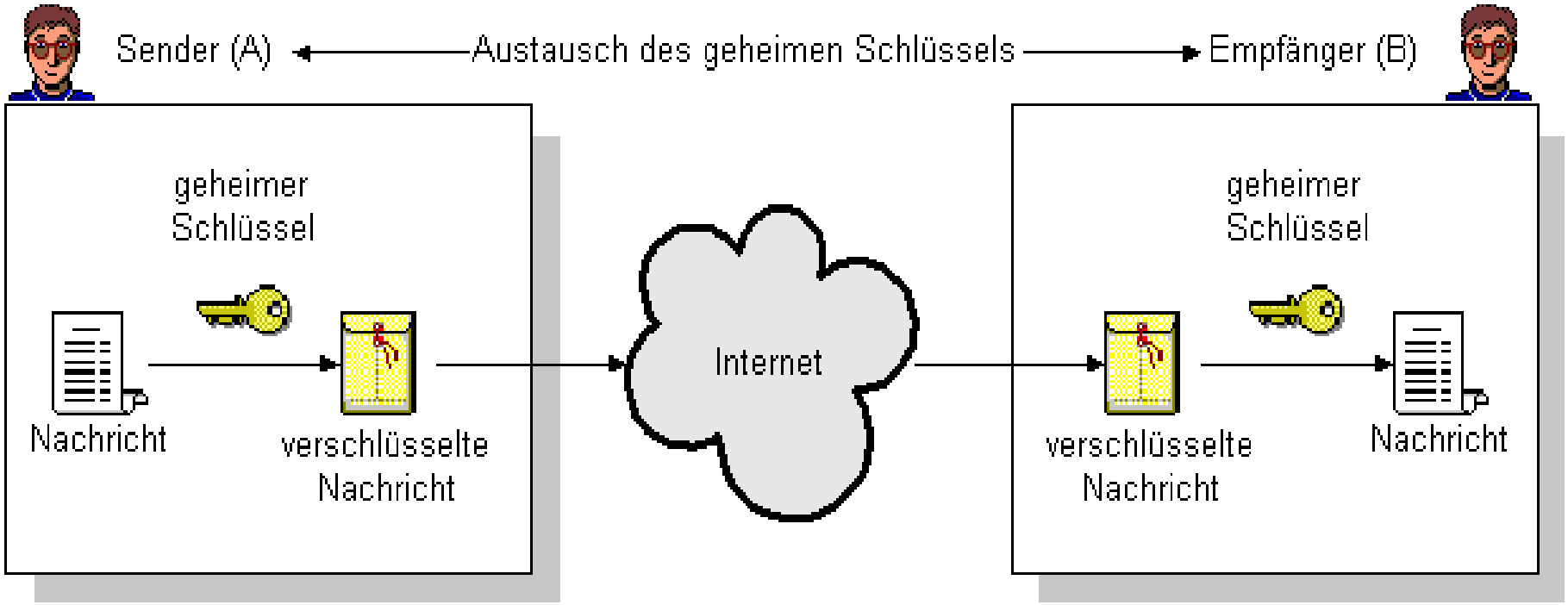
Symmetrische Verfahren („Secret-Key-Verfahren")

- *Beide Schlüssel (für Ver- und Entschlüsselung) sind **gleich**:* $K_E = K_D$
- *Beide Kommunikationspartner müssen den gemeinsamen Schlüssel kennen (vorher geheimer Austausch nötig!)*
- *„Schnelle" Verfahren*

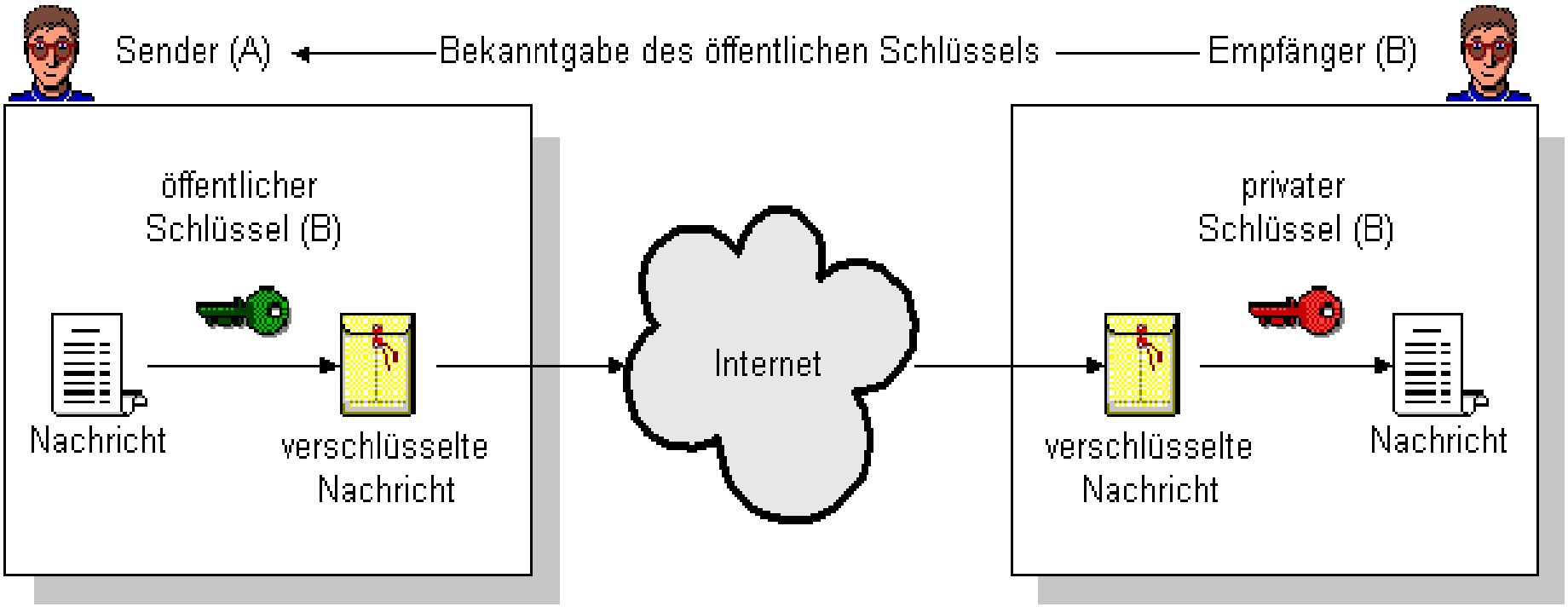
Asymmetrische Verfahren („Public-Key-Verfahren")

- *Erzeugung von Schlüsselpaaren* (K_E, K_D)
  - > *Schlüssel* K_E *zur Verschlüsselung sind **öffentlich** bekannt*
  - > *Schlüssel* K_D *zur Entschlüsselung sind **geheim** (privat)*
- *Kein geheimer Austausch nötig*
- *„Langsame" Verfahren*

Security
Symmetrische Verschlüsselung



Security
Asymmetrische Verschlüsselung

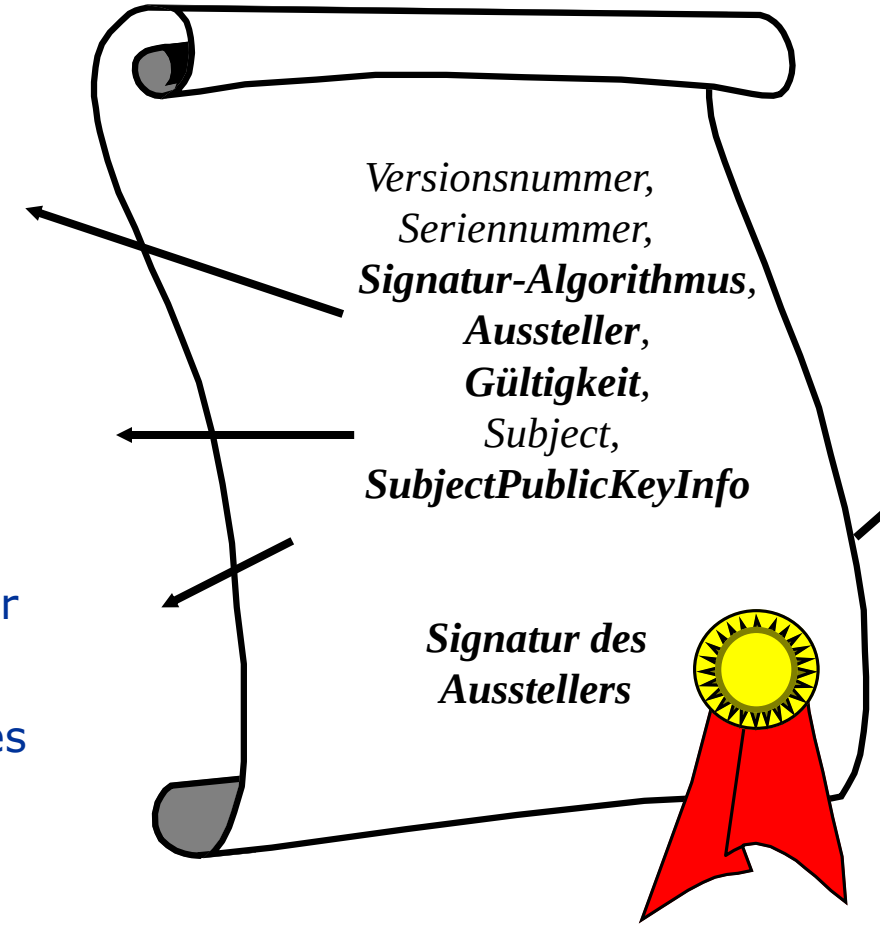


…und umgekehrt: Die mit dem privaten Schlüssel von B verschlüsselten Nachrichten können nur von B sein.

Aussteller der die Identität validiert hat

Die zu prüfend e Identität

Der Aussteller bestätigt die zuordnung des Public Key



{MD5 hash} Mit dem private key des Ausstellers Verschlüsse lter Hash

Bekannte und akzeptierte CA's:

> *VeriSign*
> *Thawte*

Zertifikate lassen sich hierarchisch strukturieren

- *Firma kann eigene Zertifikate ausstellen*
- *Die Zertifizierungsstelle wurde von einer allgemein bekannten CA zertifiziert*
- *Authentisierung beim End-Nutzer durchläuft die Hierarchie bis zur Wurzel*

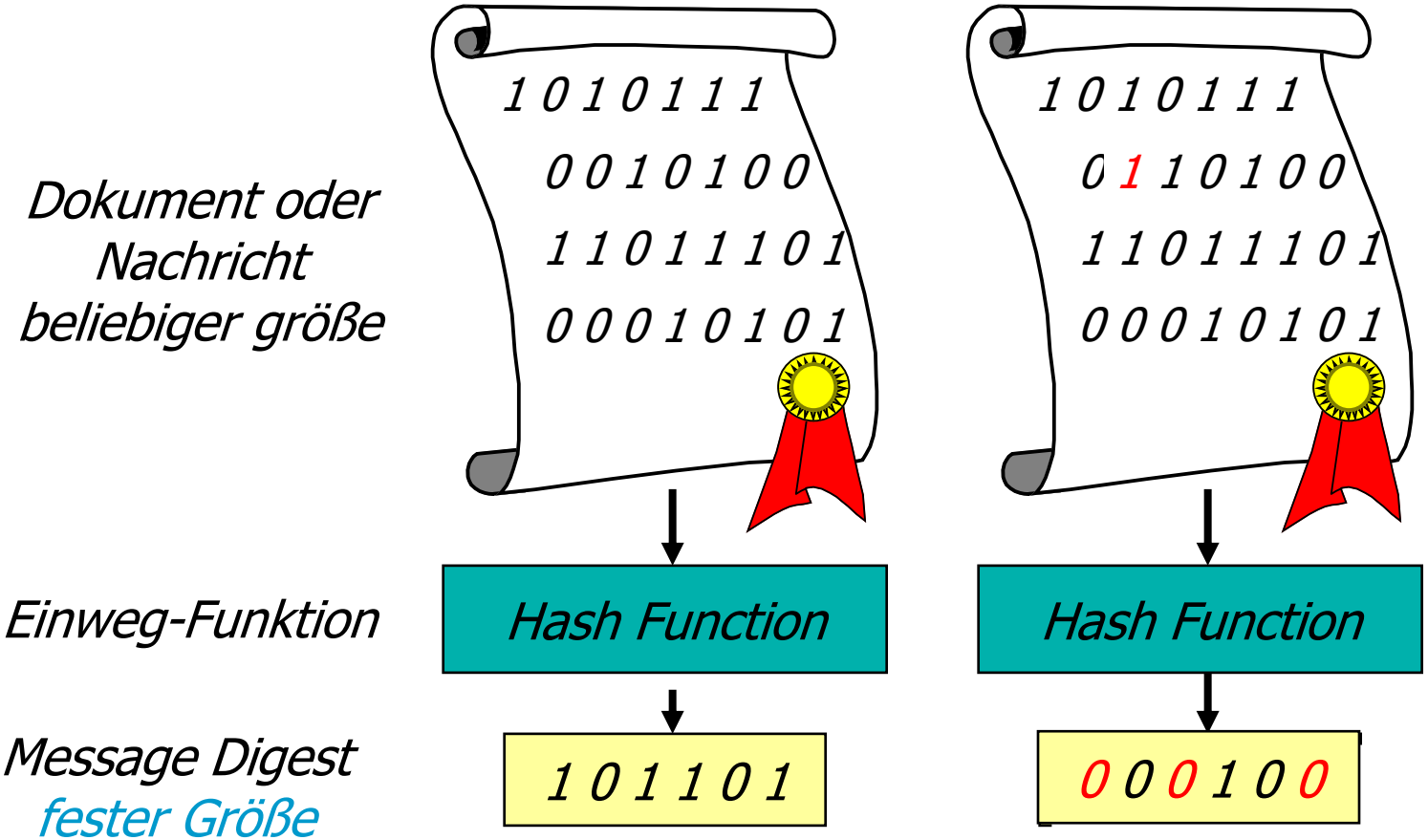
Self-Signed-Zertifikat

- *Zertifikat welches von keiner anderen Instanz zertifiziert wurde*
- *Wird meist in großen Zeitungen veröffentlicht*

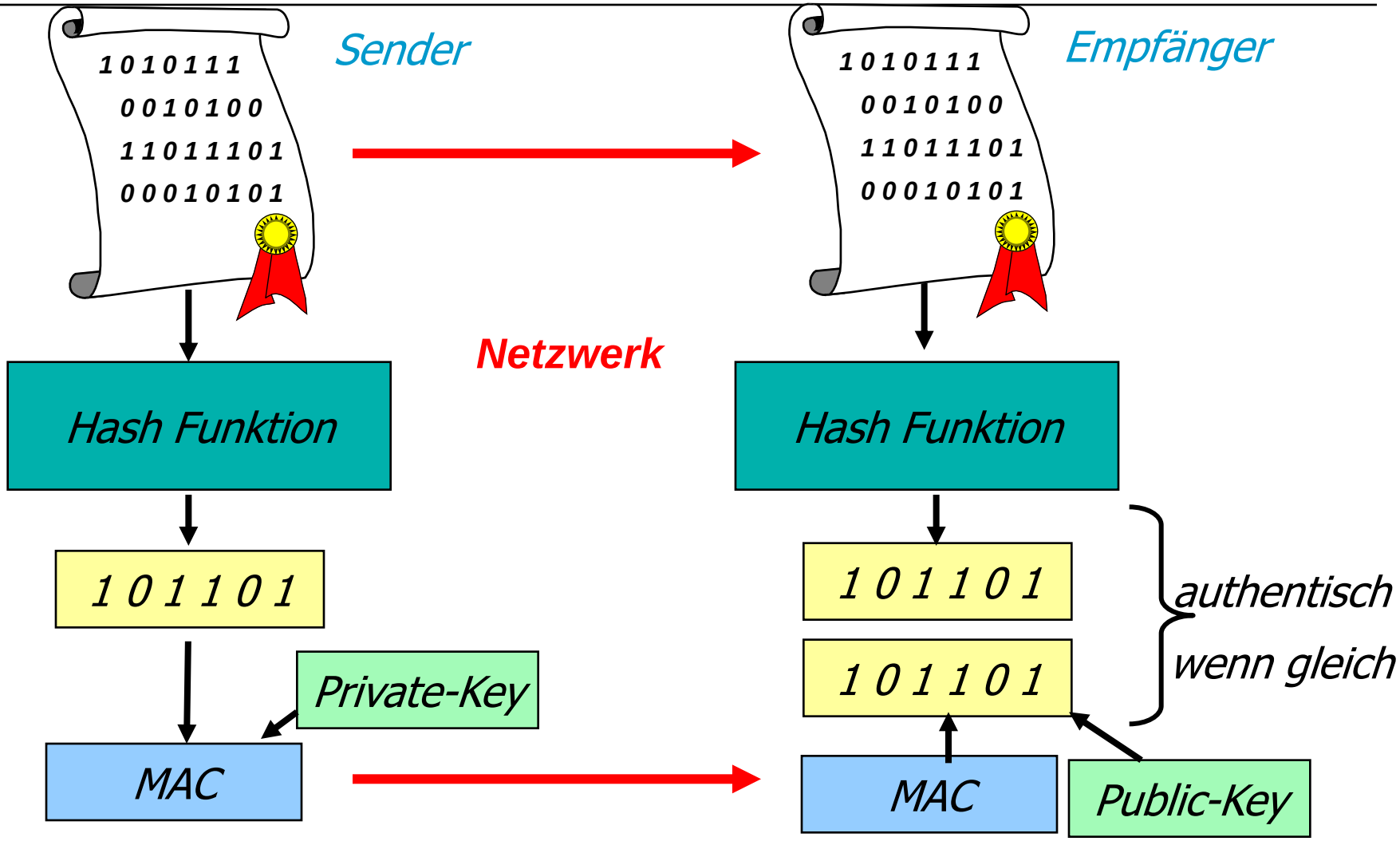
- Prinzip wie bei einer Hash-Tabelle
- Erzeugen aus einer beliebig großen Menge von Eingabedaten einen Hash-Wert (Message Digest) mit fester Länge

- Hash-Funktionen gehören zur Klasse der Einwegfunktionen
- Dauer einer Umkehrung:
  Mehr als > 1.000.000 Jahre

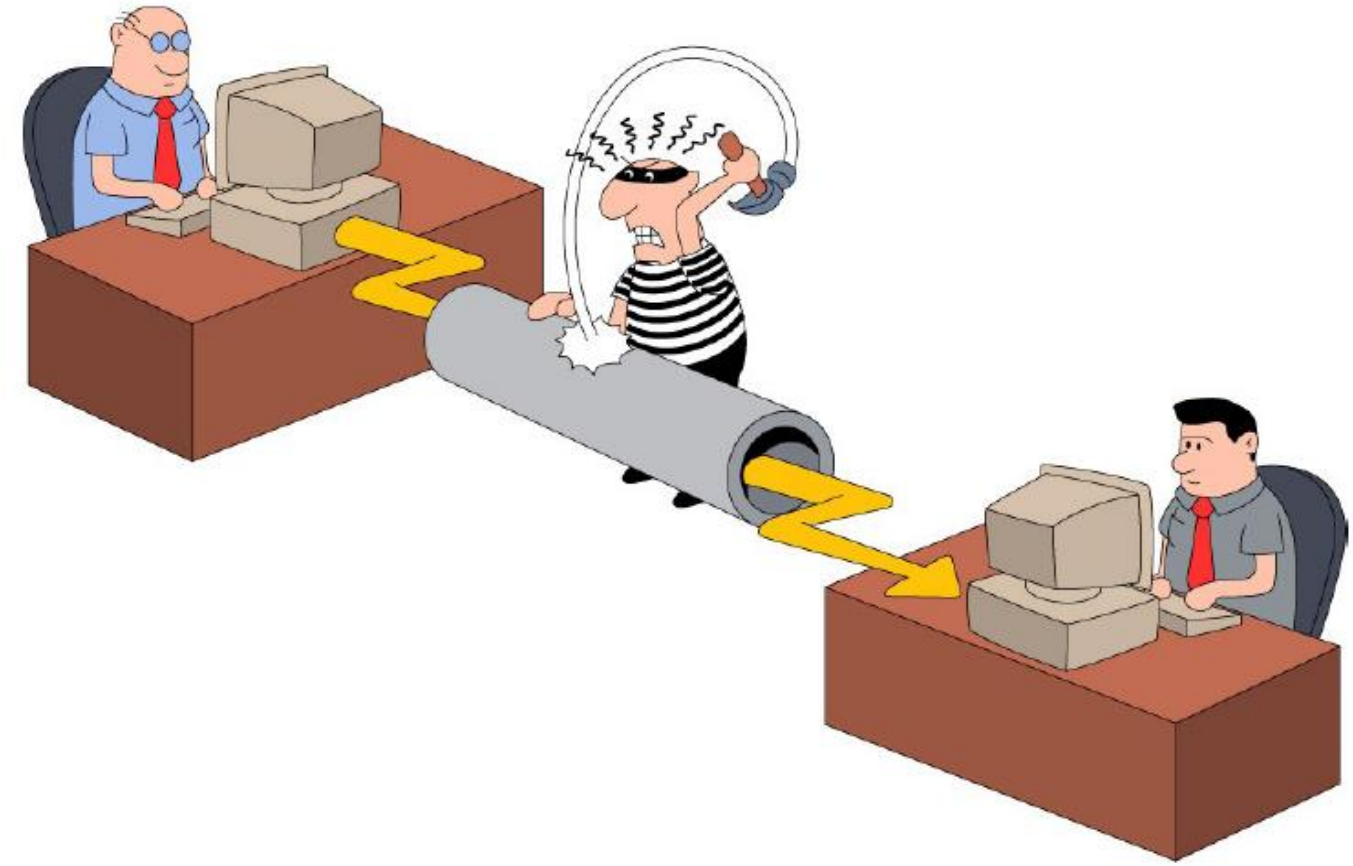
- Bekannte Algorithmen:
  - > MD5
  - > HMAC
  - > SHA-1 und -2



Die Änderung eines einzelnen Bits im Dokument sollte ungefähr 50% der Hash-bits verändern!


1 0 1 0 1 1 1
0 0 1 0 1 0 0
1 1 0 1 1 1 0 1
0 0 0 1 0 1 0 1
Sender
Empfänger
Netzwerk
Hash Funktion
Hash Funktion
1 0 1 1 0 1
1 0 1 1 0 1
authentisch
wenn gleich
Private-Key
MAC
MAC
Public-Key

Security
Absichern der Kommunikation



Security
Transport Layer Security (TLS/SSL)

- Eine effiziente Verschlüsselung der Kommunikation lässt sich eigentlich nur durch Verwendung eines symmetrischen Schlüssels verwenden
- Allerdings besteht die Problematik des Austauschs des Schlüssels
- Umgekehrt hat genau hier die asymmetrische Verschlüsselung Vorteile

- TLS/SSL greift dies aus und implementiert ein 2-Phasen Protokoll

1. Im Handshake-Protokoll wird mittels Zertifikaten und dem Austausch verschlüsselter Nachrichten die Information verteilt, aus der ein sogenannter Session-Key ableitet werden kann. Zusätzlich können sich beide Partner mittels Zertifikaten und dem Nachweis der Kenntnis der Private-Keys auch authentifizieren.

2. Im Record-Protokoll wird für eine begrenzte Zeit der symmetrische Schlüssel verwendet

- https bedeutet: http mit SSL verschlüsselt
- Der Handshake lässt sich auch abkürzen
  - Hintergrund ist, dass der klassische Handshake durchaus aufwendig ist
  - http hat kein eigentliches Verbindungskonzept
  - Jede (viele) Anfrage(n) müssten so durch den Handshake, was Lastprobleme bei den großen Web-Servern verursacht hätte
  - Daher kann man im Handshake auf einen bereits ausgehandelten Key in sicherer Art und Weise hinweisen und diesen

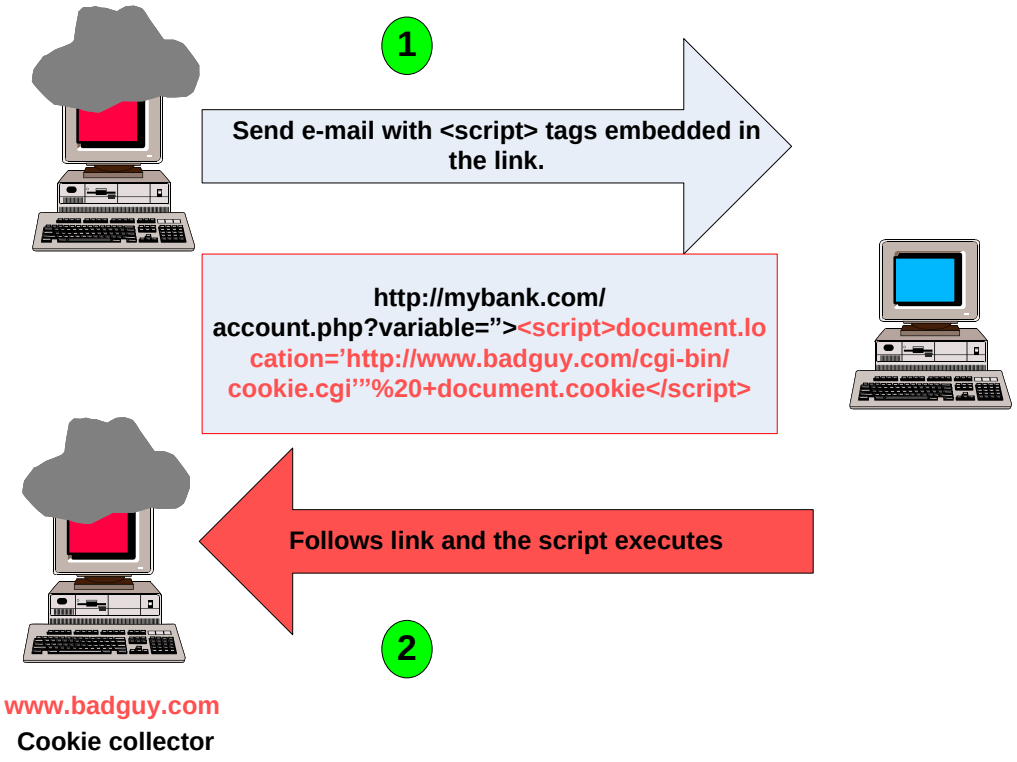
Warum ist die Lebenszeit des symmetrischen Schlüssels begrenzt?

Einige zusätzliche Informationen

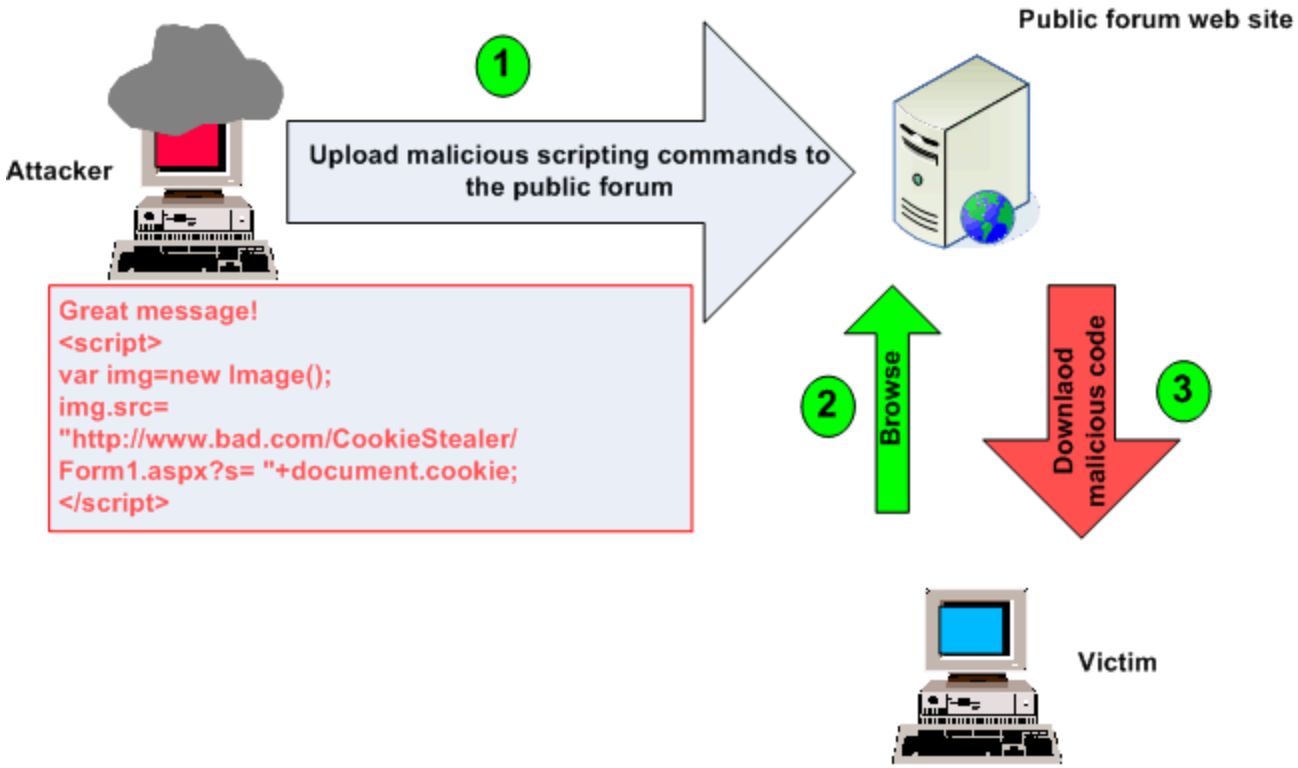
Drei bekannte Arten

- ***Reflected (Non-Persistent)***
  - *Link in externer Website oder email*
- ***Stored (Persistent)***
  - *Forum, Blog, Bulletin board, feedback form*
- ***Local***
  - *PDF Adobe Reader , FLASH player*

Security
XSS- Reflected Not Persistent

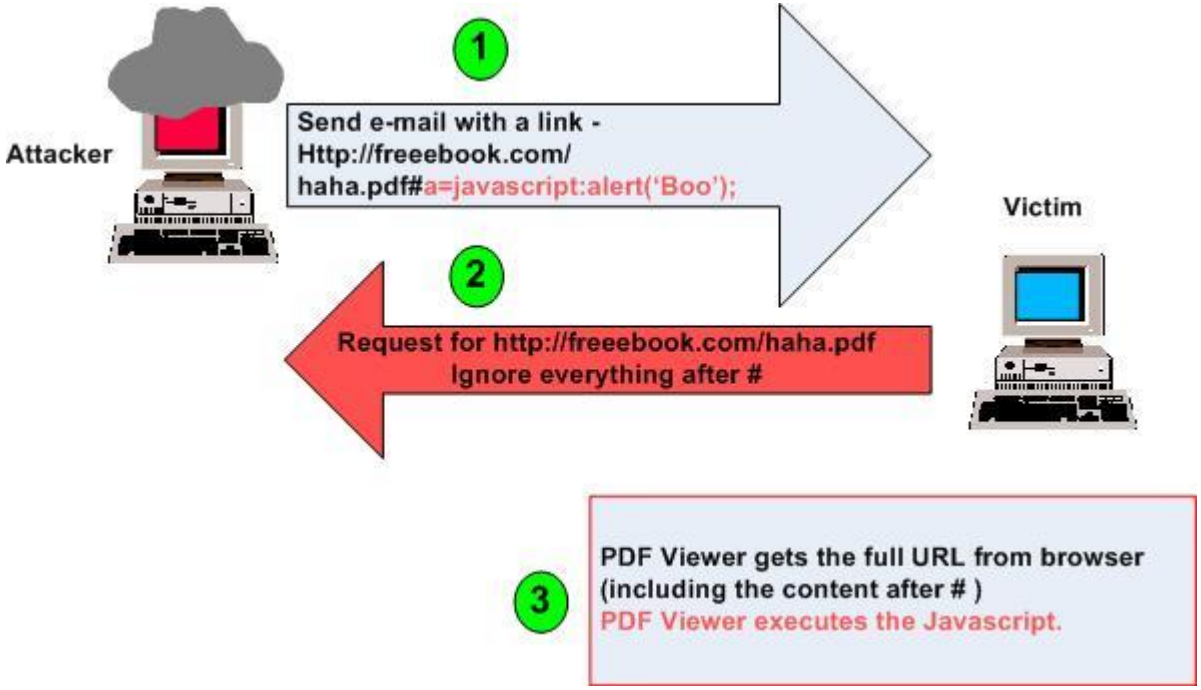


Security
XSS - Stored Persistent



- *Der Server speichert den schadhaften Inhalt (z.B. durch Eingabe von Script-Anweisungen in einem Blog)*
- *Der Server überträgt den schadthaften Inhalt in der Originalform*

Security
XSS – Lokal



- *Das injizierte Skript wird nicht an den Server übertragen, vielmehr dem Opfer "untergejubelt"*

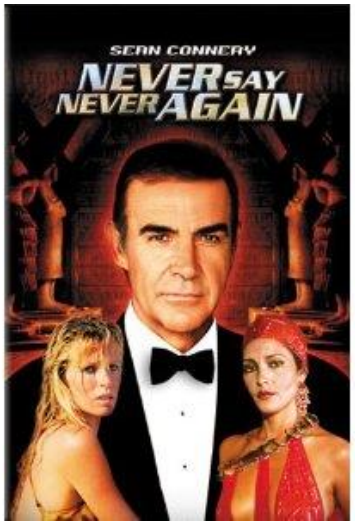
http://www.steve.org.uk/Security/XSS/Tutorial/

https://www.ghostery.com/de/faq

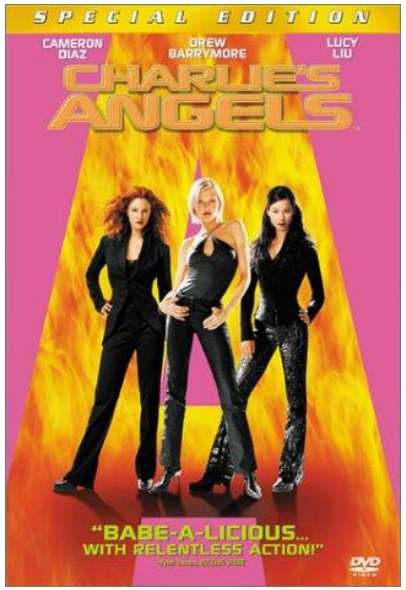
© Scott Adams, Inc./Dist. by UFS, Inc.

Clone a biometric without victim's knowledge or assistance

"my voice is my password"



cloned retina



Fingerprints from beer bottles
Eye laser scan

Bad news: it works!

[Matsumoto]

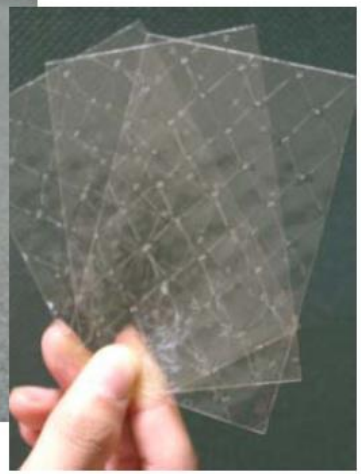
Making an Artificial Finger from a Residual Fingerprint

Materials

A photosensitive coated Printed Circuit Board (PCB)
"10K" by Sanhayato Co., Ltd .

320JPY/sheet

Solid gelatin sheet
"GELATINE LEAF "
by MARUHA CORP

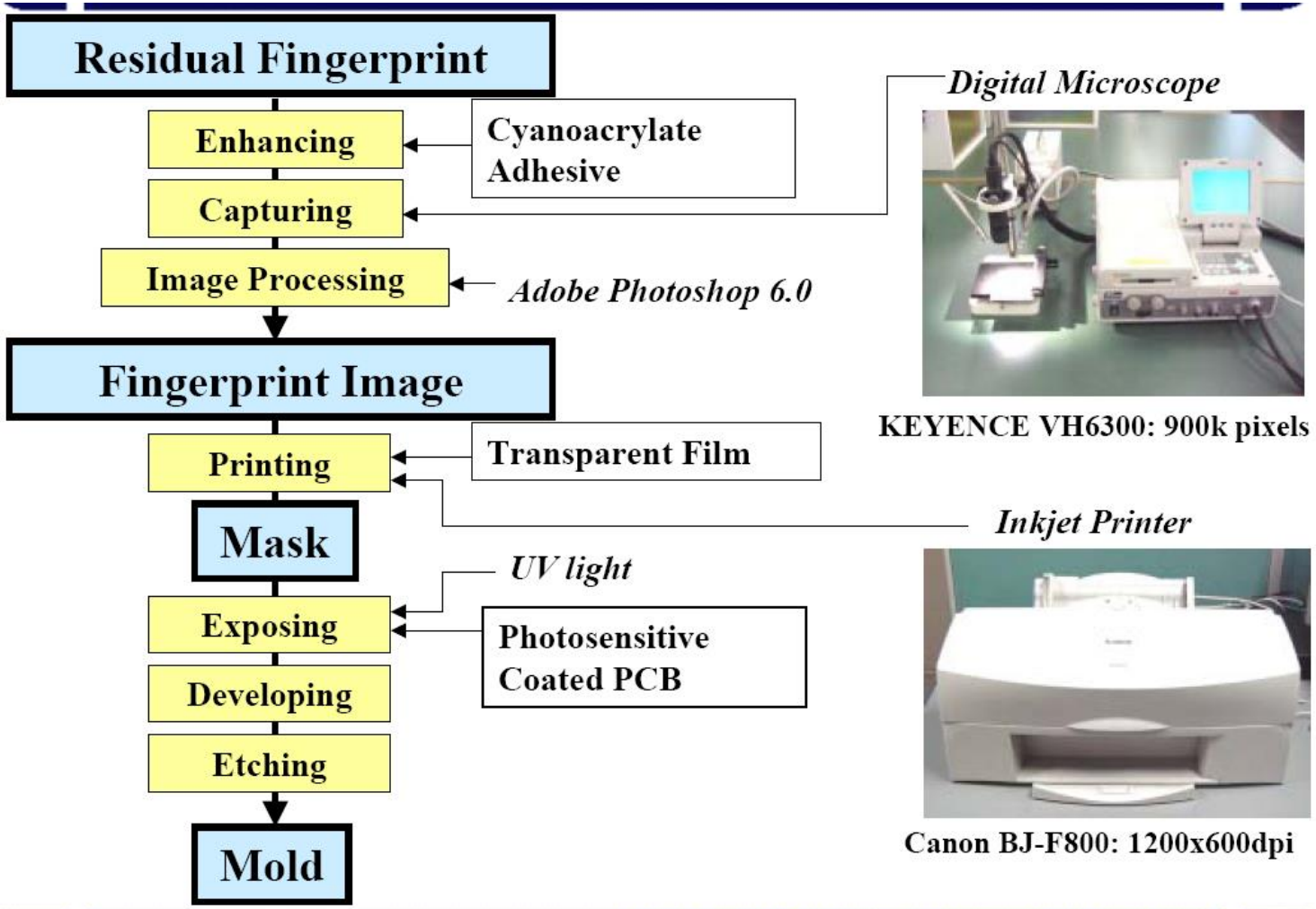


200JPY/30grams

Yokohama Nat. Univ. Matsumoto Laboratory

Security
Biometrie – Cloning eines Fingerabdruckes

[Matsumoto]



Security
Knetgummifinger

[Schuckers]

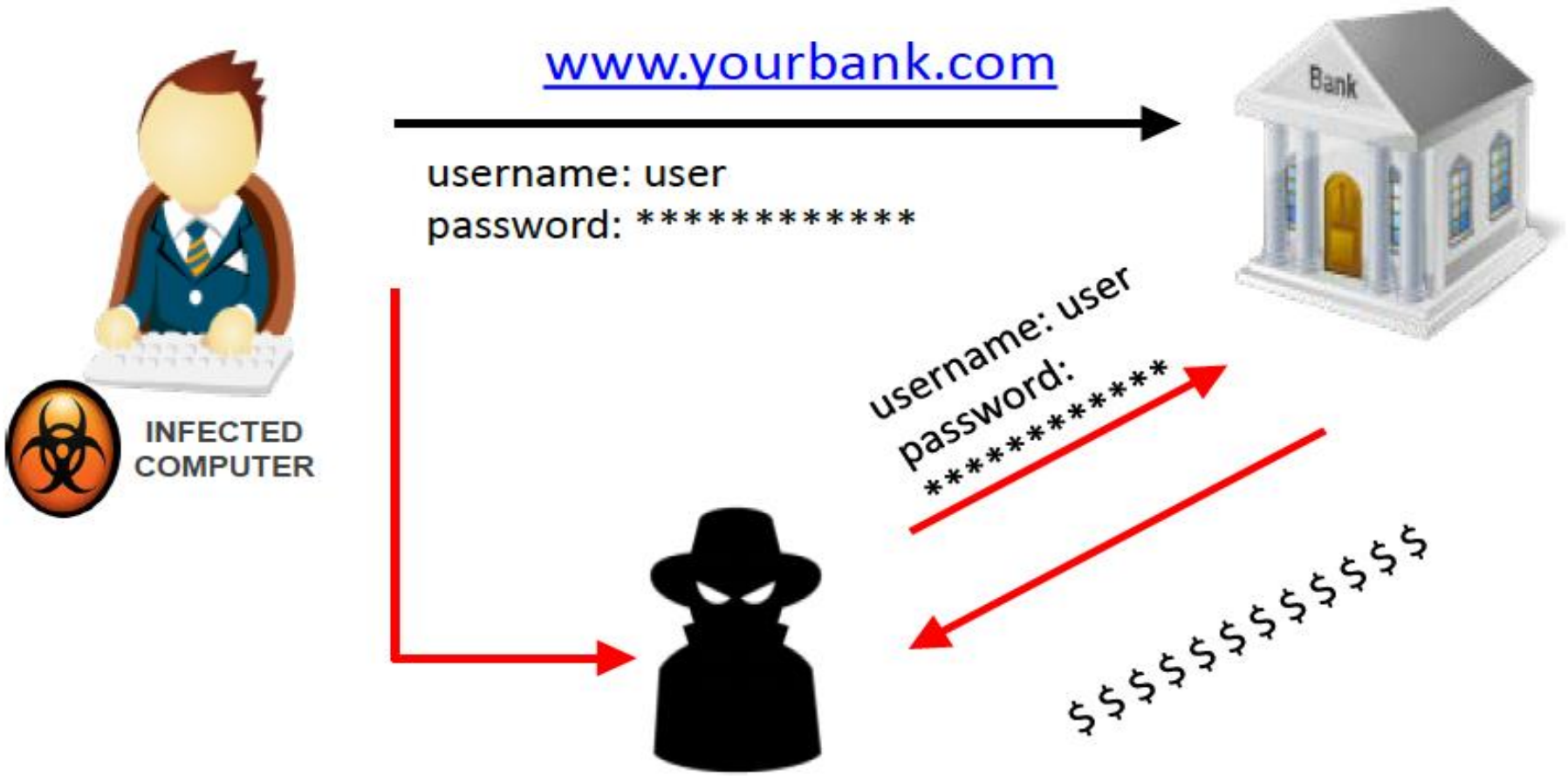
Alternative to gelatin
Play-Doh fingers fool 90% of fingerprint scanners

- *Clarkson University study*

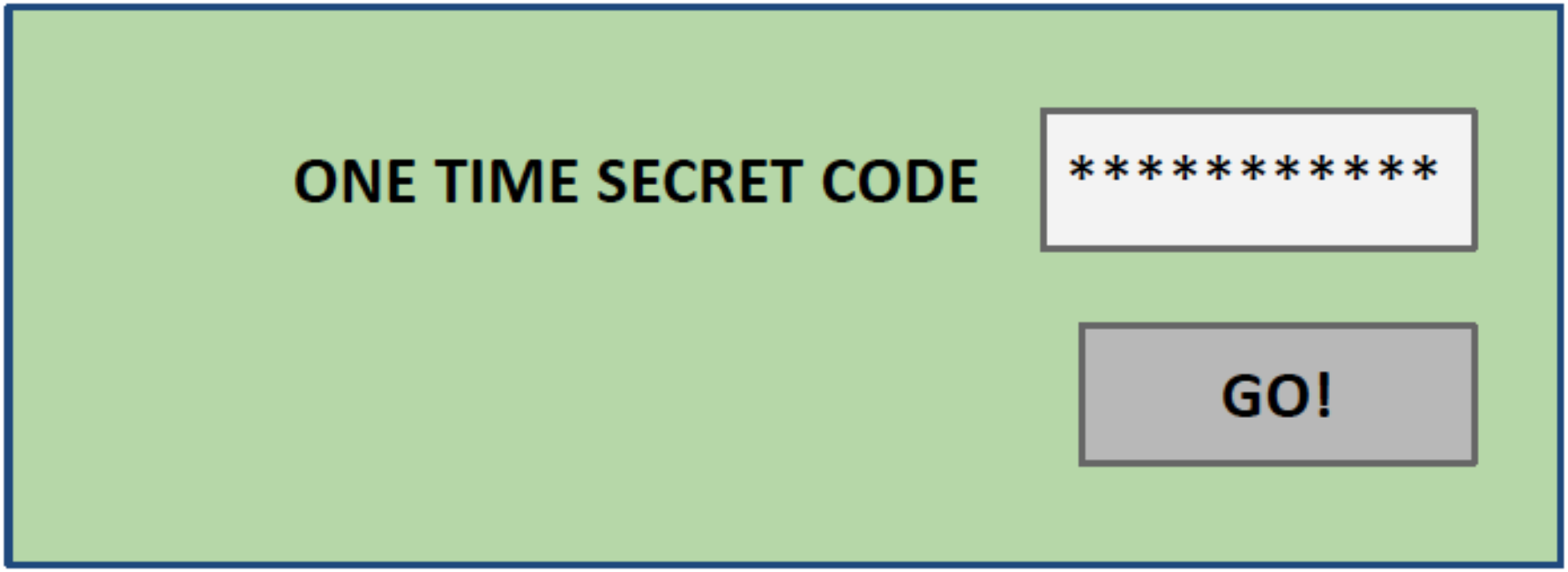
Suggested perspiration measurement to test "liveness" of the finger

"Eingesetzt haben sie dabei z. B. synthetische Finger aus „Play-Doh", einem künstlichen Ton, der in Amerika als Kinderspielzeug sehr beliebt ist; die „Druckvorlagen" hatten sie zuvor mit gewöhnlichem Dentalgips erstellt und dann ausgegossen."

Security
Online-Banking funktioniert anders



Security
2-Faktor-Authentifikation (3-Faktor?)



Security
Das mTAN-Verfahren (nach BSI)

Das mTAN-Verfahren – TAN-Versand per SMS

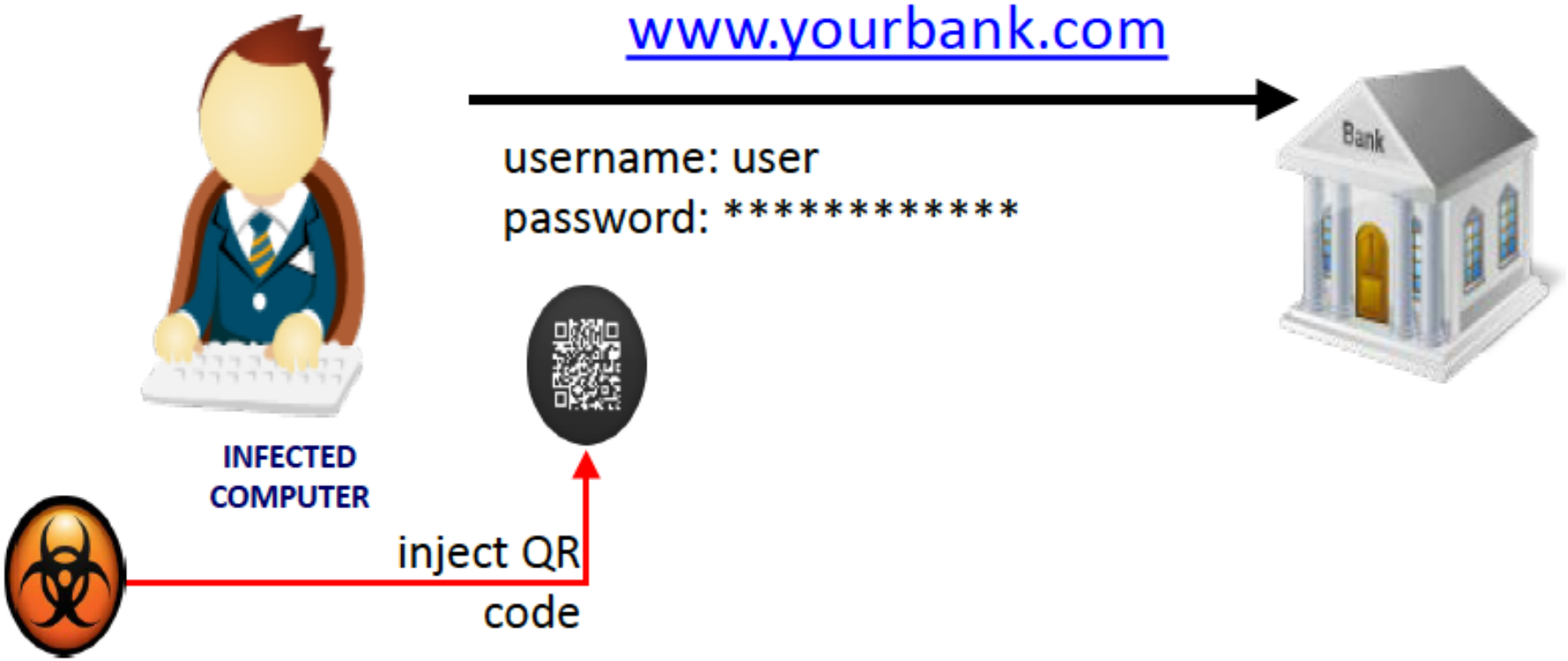
Das mTAN-Verfahren (auch "mobileTAN" oder "smsTAN" genannt) ist eine Alternative zu klassischen TAN-Verfahren für alle Anwender, die ein Mobiltelefon besitzen. Nutzer dieses Verfahrens bekommen keine TAN-Liste auf Papier zugeschickt. Stattdessen verschickt die Bank nach Aufforderung durch den Anwender bei jeder Überweisung eine "mobile TAN" per SMS auf das vorher registrierte Mobilgerät des Kunden.

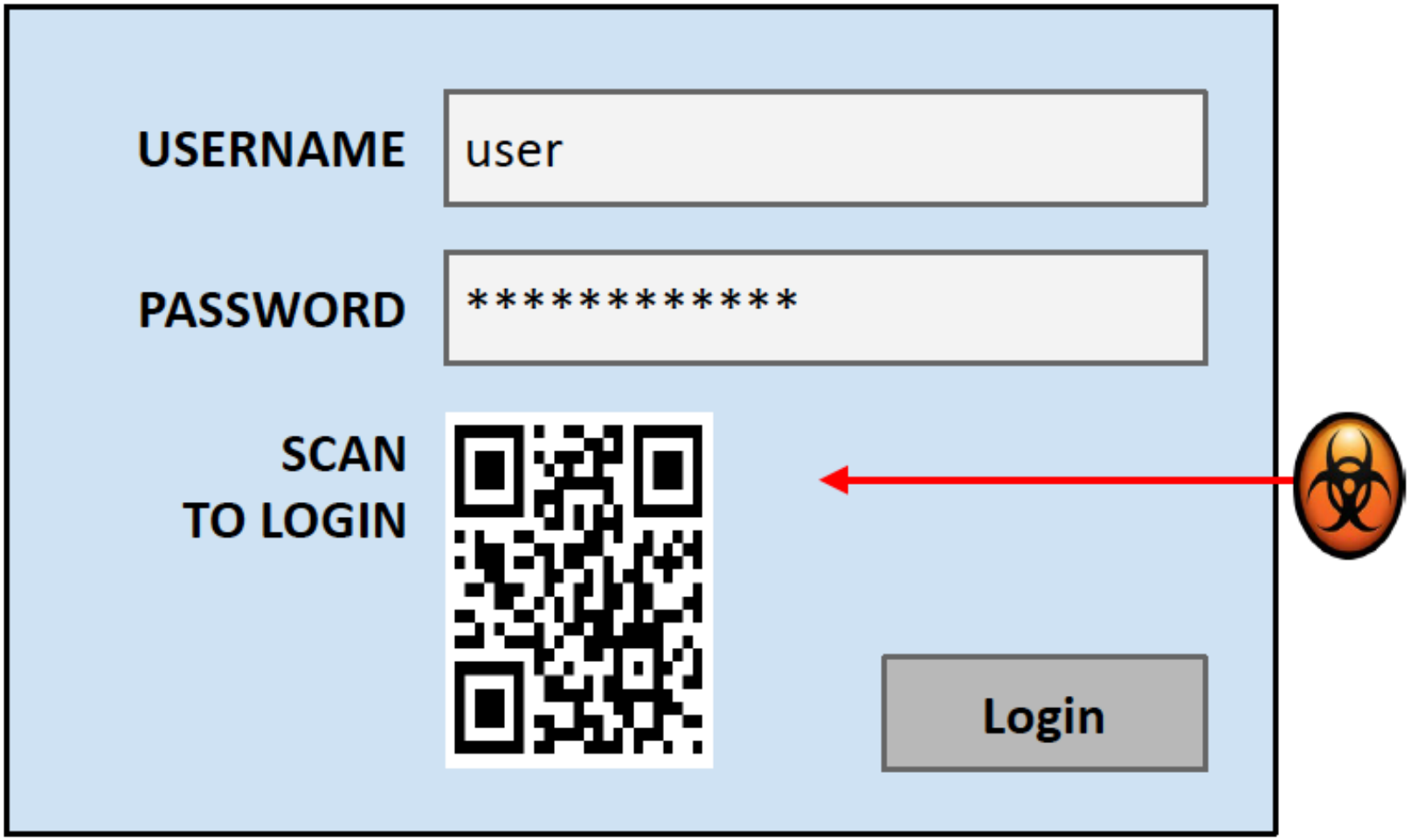
Dies trägt nicht zuletzt zur Benutzerfreundlichkeit des Verfahrens bei: Weil Anwender weder TAN-Liste noch TAN-Generator bei sich tragen müssen, können Kunden das Online-Banking mit dem mTAN-Verfahren (auch "mobileTAN" oder "smsTAN" genannt) auch unterwegs jederzeit nutzen.

...

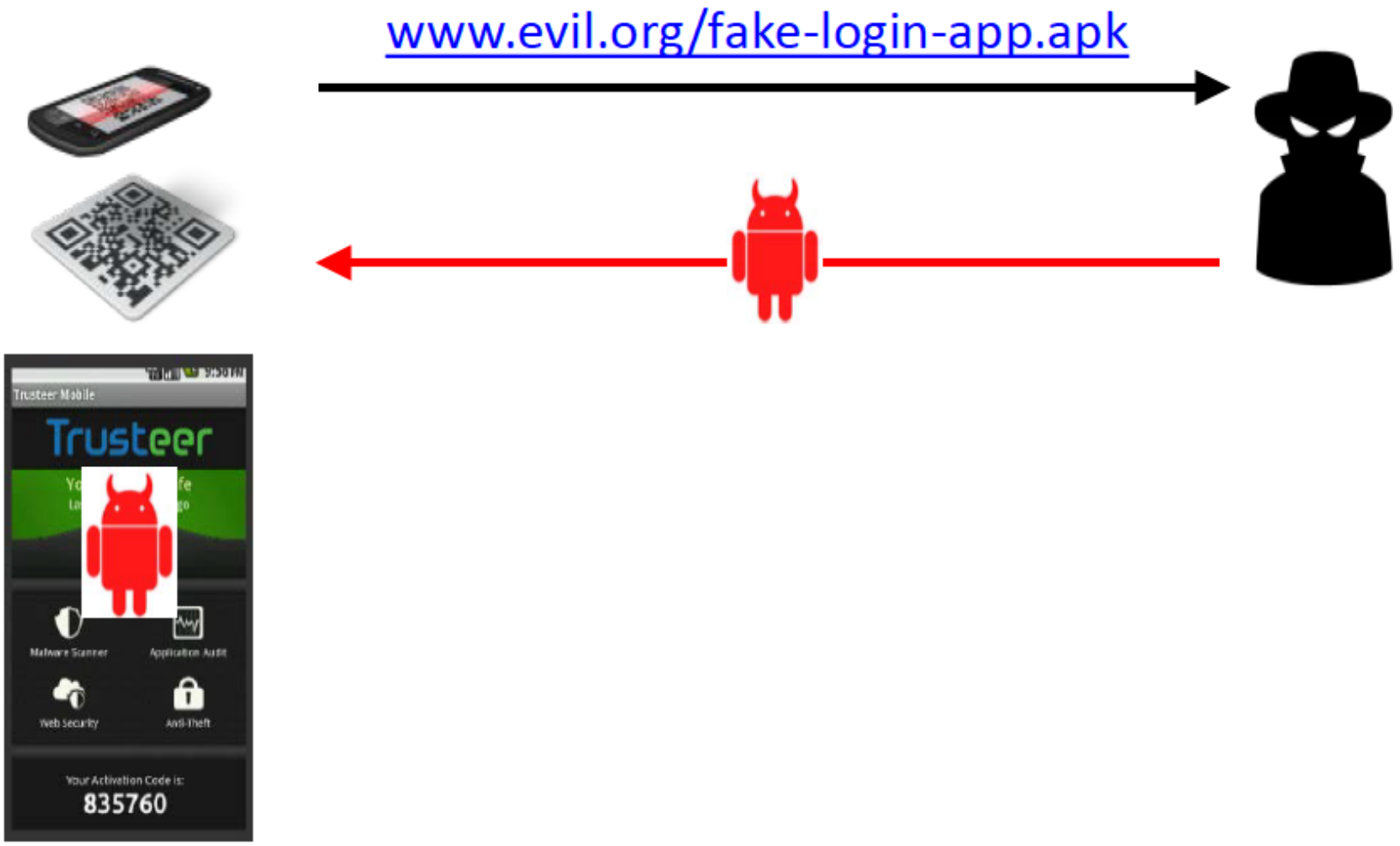
Beachten Sie aber, dass dieser Sicherheitsvorteil beim Online-Banking mit dem Smartphone nicht gegeben ist – hier ist das mTAN-Verfahren nicht zu empfehlen.

Security
mTANs und Handys

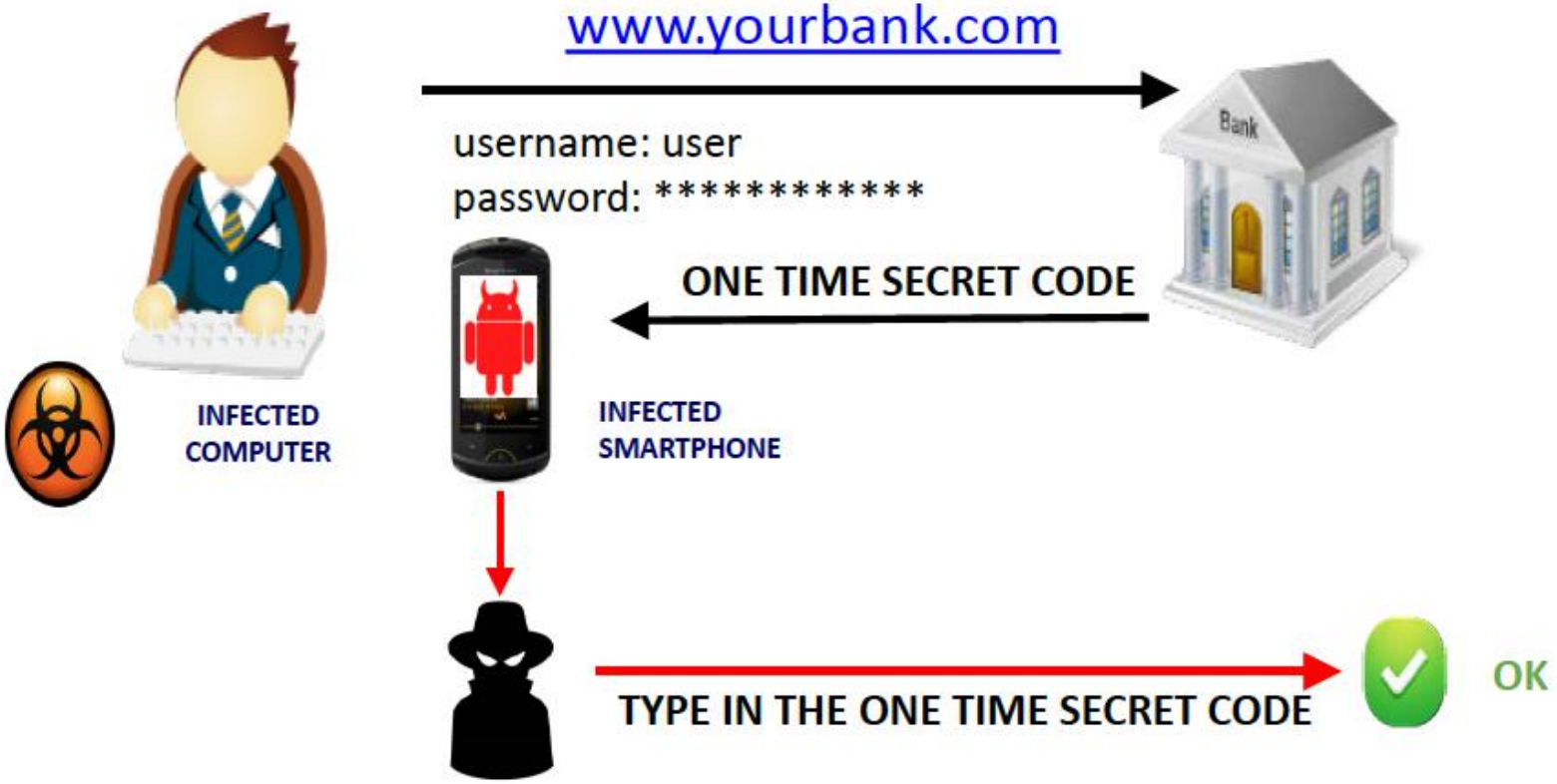

www.yourbank.com
username: user
password: **********
Bank
INFECTED
COMPUTER
inject QR
code


USERNAME
user
PASSWORD
****
SCAN
TO LOGIN
Login

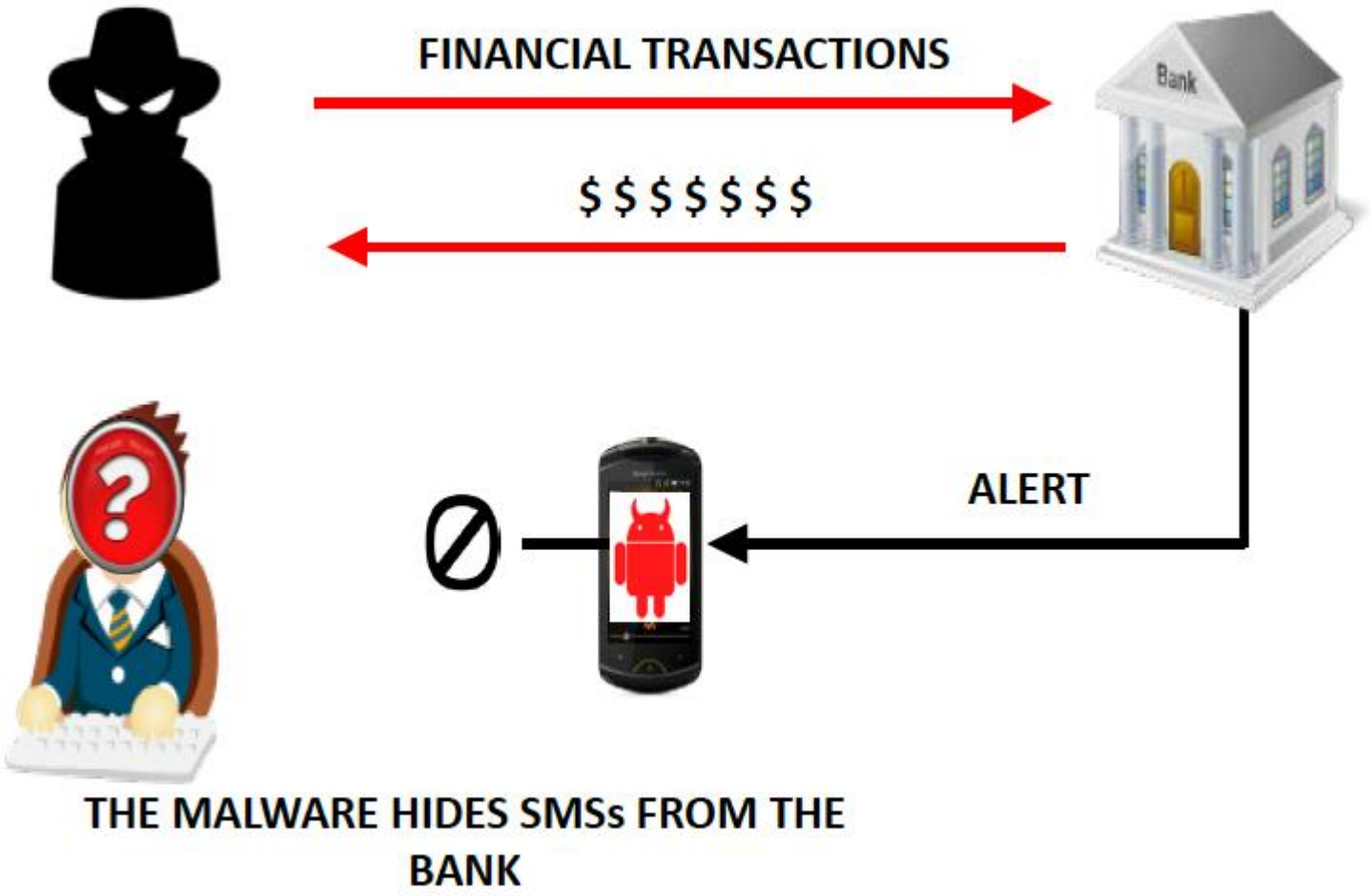
Security
mTANs und Handys



Security
mTANs und Handys



Security
mTANs und Handys



http://www.kaspersky.com/about/news/virus/2011/Teamwork_How_the_ZitMo_Trojan_Bypasses_Online_Banking_Security

http://www.spiegel.de/netzwelt/web/mtan-verfahren-neue-betrugsserie-beim-onlinebanking-nach-a-929671.html