2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Sichere Software

Vermeidung von Angriffspunkten bei der Software-Entwicklung

Andreas Vombach

Einleitung

- Mein Hintergrund
  - Von der Hardware- zur Softwareentwicklung
- Software im Banking Bereich
  - Erlebnisse mit Security Officern (Ich will doch nur Admin sein)
- Warum der Vortrag?
  - Aufzeigen möglicher Angriffspunkte
  - Welche Software ist am meisten gefährdet?

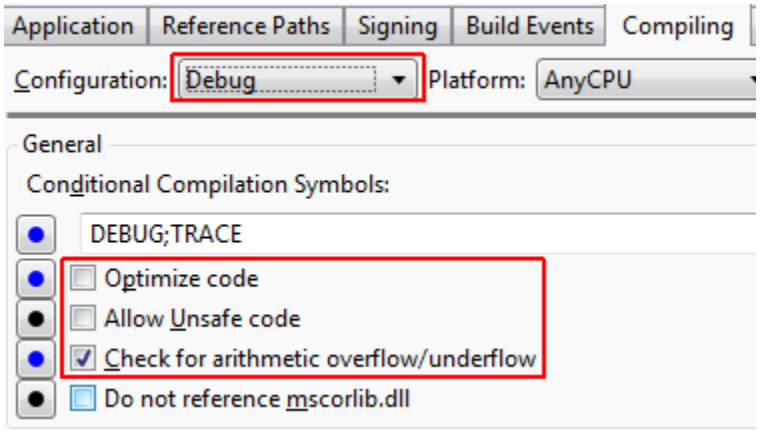
Überblick

- Welche Sicherheitsaspekte gibt es für Software?
- (Wie) erkennen wir potentielle Gefährdung?
- Am besten vermeiden wir Risiken (schon) bei der Entwicklung durch Security by Design!

- Frage:
- Wer entwickelt noch Standalone Applikationen?
- Wer kümmert sich um Sicherheit bei verteilten Applikationen?
- In welcher Sprache werden Web Applikationen entwickelt? (ASP, PHP ...)

Mögliche Angriffspunkte

- Standalone Applikationen, *statisch betrachtet*:
  - Compiler Einstellungen
    - Exe im Debug mode ausgeliefert



Mögliche Angriffspunkte

- Schlüssel befinden sich im code
  - Private Keys und symmetrische Verschlüsselung
- Obfuscation
  - Nicht einmal der Versuch der Verschleierung wurde unternommen

CSS-Hack [Bearbeiten]

Da die in CSS verwendete Kryptographie mit lediglich 40 Bit langen Schlüsseln, deren Komplexität auf 2^{25} verringert werden kann, mit heute verfügbaren normalen PCs in vertretbarer Zeit per Brute-Force-Attacke geknackt werden kann, ist auch der Aufwand zur Wiederherstellung der Inhalte beherrschbar. Dies musste vom DVD Forum bei der Standardisierung von CSS im Jahre 1996 wissentlich in Kauf genommen werden, da die damaligen Exportbeschränkungen der USA aus Sicherheitsgründen keinen Export von starker Kryptographie ins Ausland zuließen. Der Brute-Force-Ansatz erwies sich sehr bald sogar als unnötig, da Kryptographen und Hacker herausfanden, dass CSS fundamentale Designfehler enthält, die ein Knacken des Abspielschutzes innerhalb von Sekunden erlauben.

Bei allen Bemühungen der Industrie, die genaue Funktionsweise von CSS geheim zu halten, musste die Technologie doch in jedem einzelnen von Millionen von Geräten und Programmen (Software-DVD-Player) implementiert werden. Vermutlich gelangte die Funktionsweise der Technologie durch Reverse Engineering der Software-DVD-Player an die Öffentlichkeit. Schließlich verbreitete sich im Oktober 1999 das Programm DeCSS im Internet, mit dem sich CSS

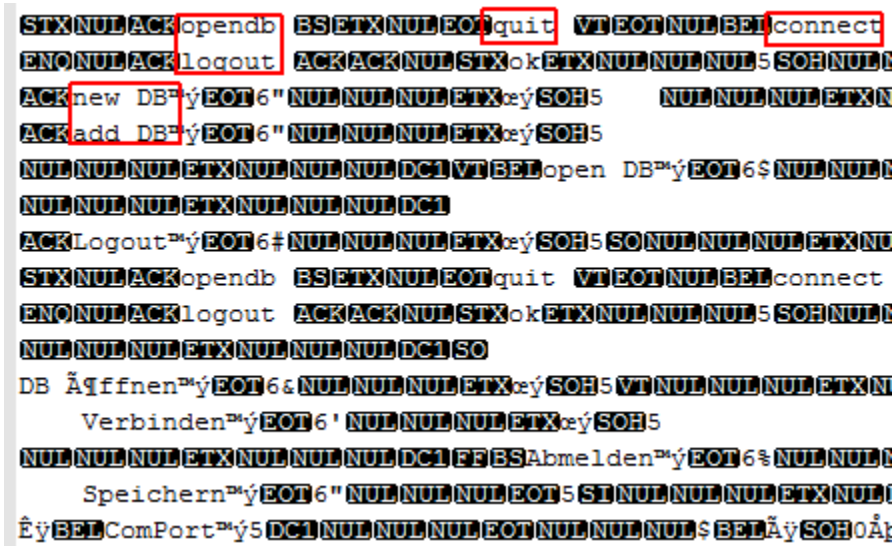
Wie sichern wir die Software?

- JAVA:
  - Code obfuscator einsetzen
  - Entschlüsselnden ClassLoader verwenden

```
public Class findClass(String name) throws ClassNotFoundException {
// Dieses Array nimmt den Bytecode auf.
byte[] byteCode = null;
// Zusammensetzen des Dateinamens
String fileName = name.replace('.', File.separatorChar)+"."+CRYPT_EXT;
try {
  // Entschlüsseln der Datei und Einlesen des Bytecodes
  byteCode = decryptFile(fileName);
}
catch(IOException e) {
  throw new ClassNotFoundException("Encrytepd class "
                                   +name+" not found.", e);
}
// Klasse in der VM erzeugen und Class-Objekt zurückliefern
Class c = defineClass(name, byteCode, 0, byteCode.length);
return c;
}
```

Wie sichern wir die Software?

- C / C#:
  - Im Release Mode compilieren!



Wie sichern wir die Software?

- C / C#:
  - Keinen unmanaged code zulassen
  - DEP und ASLR Flags aktivieren

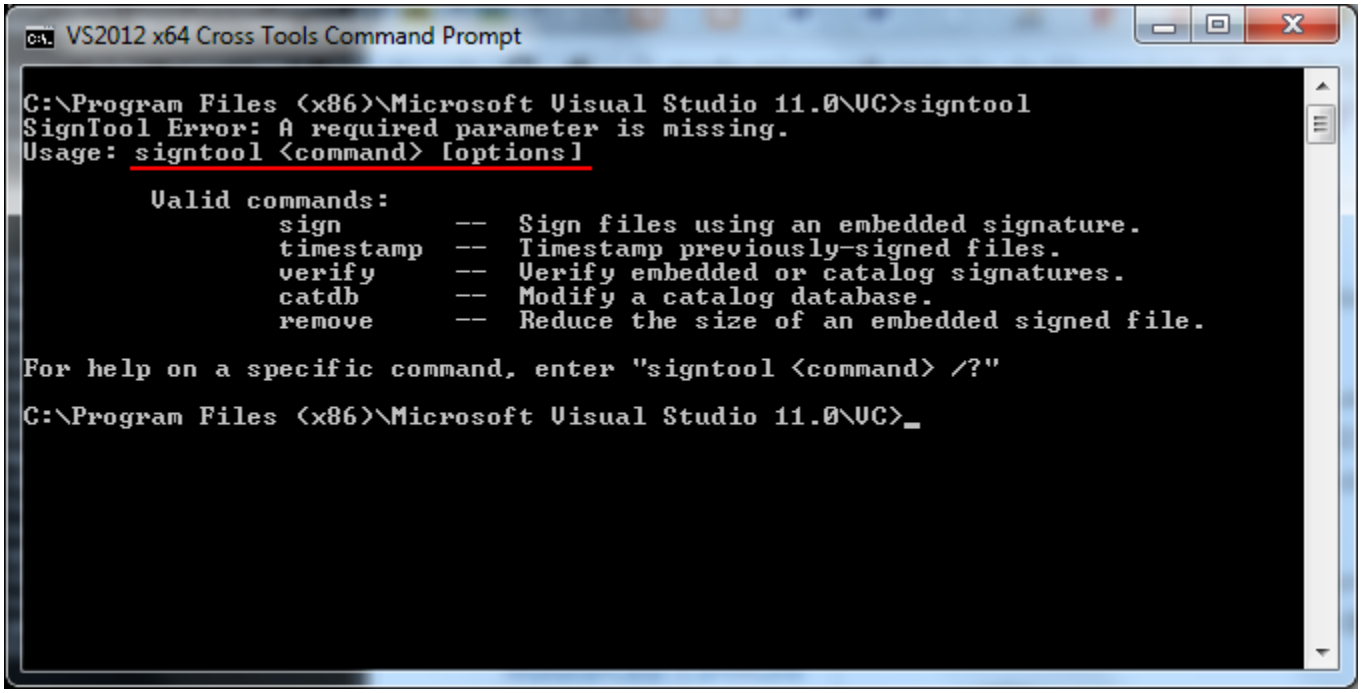
Code Red (Computerwurm)

Code Red ist eine Familie von Computerwürmern, die sich ab dem 12. Juli 2001 im Internet verbreitete. Die ersten befallenen Rechner wurden am 13. Juli an eEye Digital Security gemeldet, wo Marc Maiffret und Ryan Permeh die erste Analyse durchführten. Den Namen erhielt der Wurm in Referenz zur Defacement-Meldung und nach dem Getränk *Mountain Dew Code Red*, das die beiden Analysten während der Untersuchung tranken.[1]

Die meisten Infektionen verursachte die zweite Version von *Code Red*, die über 359.000 Rechner am ersten Tag infizierte.[2] Gefährlicher war der neue Wurm *Code Red II*, der ab Anfang August kursierte, da er eine Backdoor installierte.[3] Alle Varianten zusammen haben schätzungsweise 760.000 Rechner infiziert.[4]

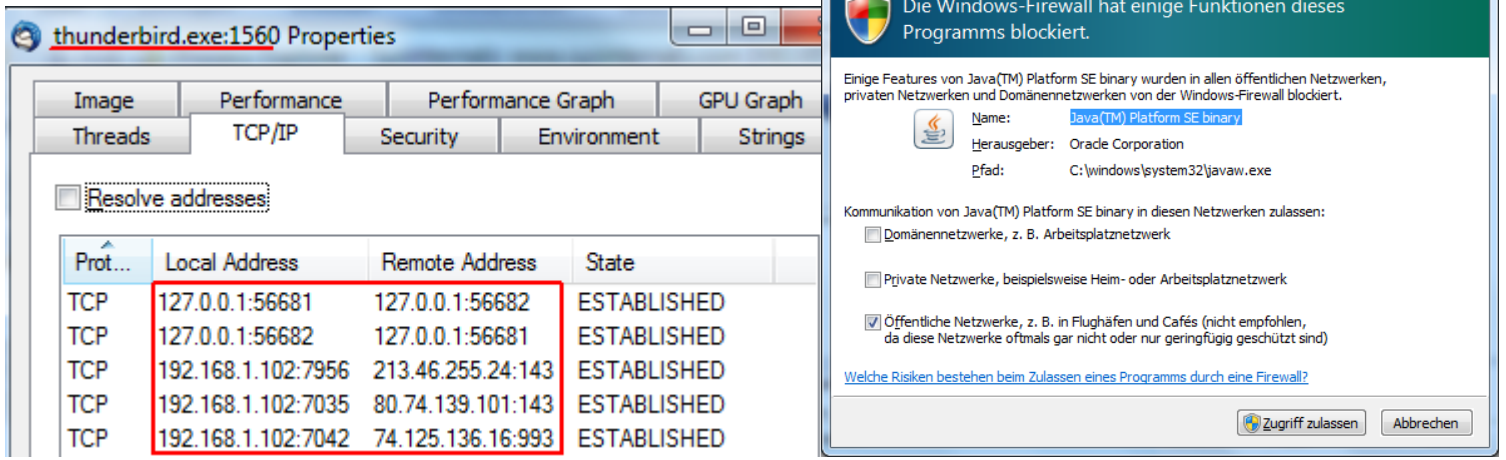
Wie sichern wir die Software?

- Windows executables
  - Code signieren



Wie sichern wir die Software?

- Testen ob das Programm durch den Benutzer zum Absturz gebracht werden kann (*dynamisches Verhalten*)
  - Arithmetik checks beim Compiler aktivieren
- Netzwerkverbindungen des Programm kontrollieren
  - Listening Ports (Process Monitor)



Mögliche Angriffspunkte im Netz

- Ungesicherte WLAN Netze
- Email und Skype (Sicherheitsrisiko Mensch bei Download)
- Andere Messaging Apps (Übertragungsprotokoll im Klartext)
- VPNs, Mobile Devices und Smartmeter

Intelligente Stromzähler in Puerto Rico häufig für dumm verkauft

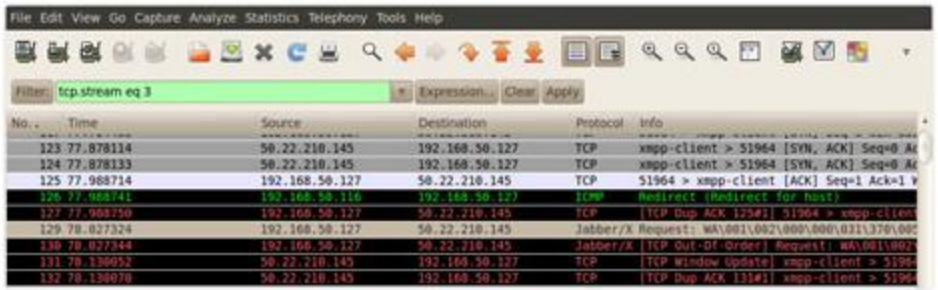
vorlesen / MP3-Download

In Puerto Rico sind offenbar nicht nur Smart Meter selbst bereits selbstverständlich, sondern auch die Manipulation der "intelligenten Stromzähler", wie der US-Journalist Brian Krebs berichtet. Er beruft sich dabei auf einen auf das Jahr 2010 datierten FBI-Bericht. Demnach haben Stichproben eines nicht näher genannten Stromversorgers ergeben, dass etwa jedes zehnte Smart Meter manipuliert ist. Den daraus entstehenden Schaden schätzt der Versorger auf bis zu 400 Millionen US-Dollar jährlich – entsprechend derzeit 300 Millionen Euro.

Die Stromdiebe des karibischen Inselstaats, der ein assoziierter Freistaat der USA ist, manipulieren die Stromzähler laut dem FBI über die Infrarot-Wartungsschnittstelle mit optischen Ausleseköpfen (Optical Probes), die sie für rund 400 US-Dollar im Internet bestellen. Die passende Software ist nur einen Download entfernt. Dabei bleibt die Hardware des Zählers unversehrt. Unter Umständen genügt es laut dem FBI jedoch auch schon, einen sehr starken Magneten an dem Smart Meter zu platzieren, um an Gratis-Strom zu kommen.

Beispiel WhatsApp

Und MAC-Adressen sind nicht etwa sonderlich versteckt, sondern lassen sich sowohl in den Info-Einstellungen des iPhones als auch in jedem W-Lan Netz mit minimalem Aufwand auslesen.



Ist die Mac-Adresse und die Telefonnummer eines “WhatsApp”-Nutzers bekannt, kann sein Account “lebenslang” übernommen werden. Nachrichten lassen sich unter seinem Namen verschicken, andere Nutzer im “WhatsApp”-Netz ausfindig machen und mehr. Web-Entwickler können etwa diese PHP-Klasse nutzen um sich einen eigenen “WhatsApp”-Client im Web-Browser zusammen zu bauen und Nachrichten anschließend unter fremden Namen zu versenden.

Wie sichern wir die Software?

- Es geht jetzt (auch) um *Laufzeit Verhalten*!
- Security Audit:
  - Sind die Verbindungen gesichert?
  - Beispiel: 3 Tier App mit Verbindung zu einer Datenbank

Access Control List

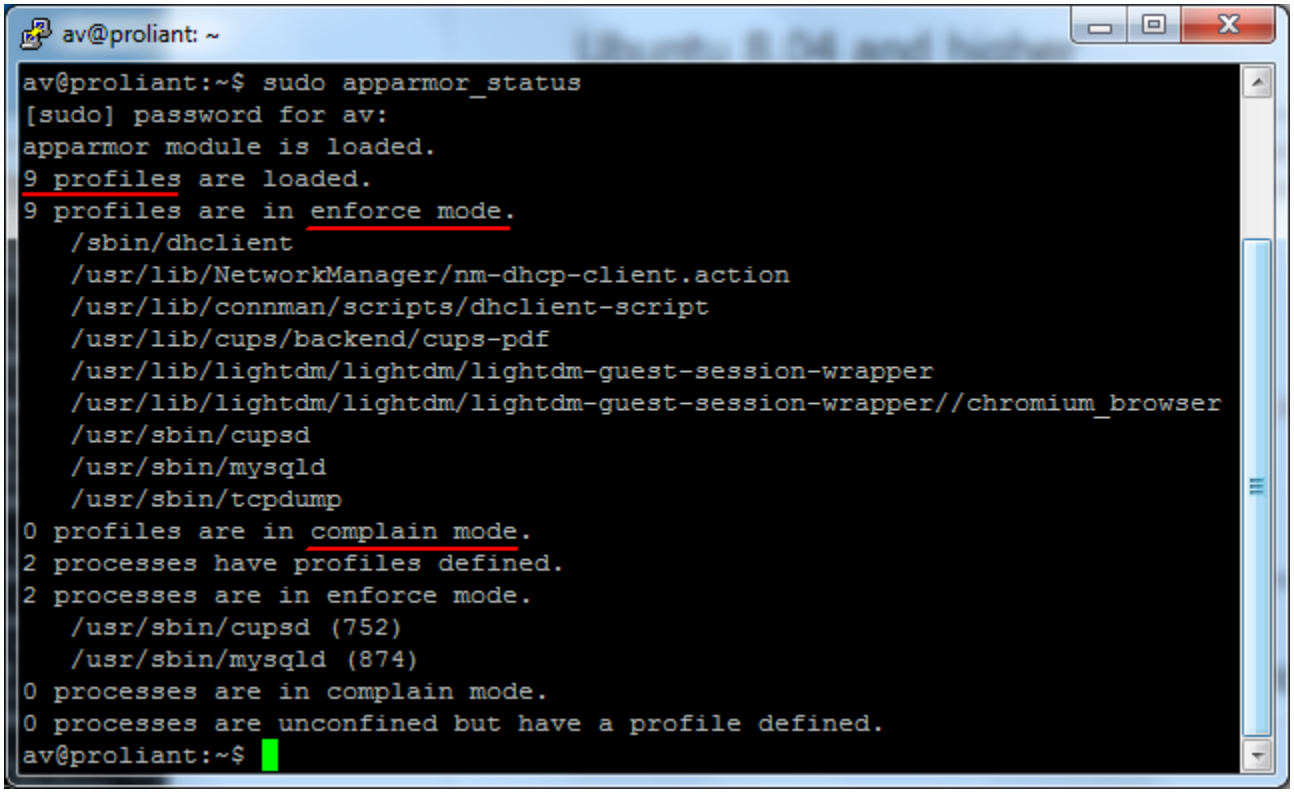
Eine **Access Control List** (**ACL**), deutsch **Zugriffssteuerungsliste**, ist eine Software-Technik, mit der Betriebssysteme und Anwendungsprogramme Zugriffe auf Daten und Funktionen eingrenzen können. Eine ACL legt fest, in welchem Umfang einzelne Benutzer und Systemprozesse Zugriff auf bestimmte Objekte (wie Dienste, Dateien, Registryeinträge etc.) haben.

Im Unterschied zu einfachen Zugriffsrechten sind ACLs feiner einstellbar. So können etwa bei Linux für eine Datei für mehrere Benutzer und Gruppen unterschiedliche Rechte vergeben werden, während reguläre Zugriffsrechte nur die Rechtevergabe für einen Benutzer, eine Gruppe und den „Rest der Welt“ zulassen.

L)

Wie sichern wir die Software?

- AppArmor (Rechteverwaltung für Applikationen, MAC)



Wie sichern wir die Software?

- MAC (Mandatory Access Control)

Biba [Bearbeiten]

→ *Hauptartikel: Biba-Modell*

Das Biba-Modell stellt eine *Umkehrung* des Bell-LaPadula-Modells dar: Hier werden Informationen nicht vor dem Lesen, sondern vor Manipulation durch Unbefugte geschützt. Das Biba-Modell legt einen Schwerpunkt auf die Integrität der Daten. Es wird einerseits in der Informationstechnik verwendet, z. B. als Gegenmaßnahme bei Angriffen auf sicherheitsrelevante Systeme wie Firewalls, andererseits auch bei militärischen Systemen, wo es grundlegend wichtig ist, dass ein Befehl in der Kommandokette nicht modifiziert werden kann und somit eine falsche Anweisung weitergegeben wird.

Bell LaPadula [Bearbeiten]

→ *Hauptartikel: Bell-LaPadula-Sicherheitsmodell*

Das Bell-LaPadula-Modell befasst sich mit der Vertraulichkeit der Daten. Es soll nicht möglich sein, Informationen einer höheren Schutzstufe zu lesen oder Informationen einer höheren Schutzstufe in eine tiefere Schutzstufe zu überführen. Systeme, die auf dem *Bell-LaPadula*-Prinzip basieren, wurden vor allem dann verwendet, wenn Daten einer gewissen Geheimhaltung unterliegen. Die klassischen *Bell-LaPadula*-Systeme wurden durch *Lattice*- oder *Compartment*-basierende Systeme abgelöst.

Mögliche Angriffspunkte bei Web Apps

- SQL injection
- ASP vulnerabilities

	Erwarteter Aufruf
Aufruf	http://webserver/search.aspx?keyword=*sql*
Erzeugtes SQL	SELECT url, title FROM myindex WHERE keyword LIKE '%*sql*%'
	SQL-Injection
Aufruf	http://webserver/search.aspx?keyword=*sql'+;GO+EXEC+cmdshell('shutdown+/s')*+--
Erzeugtes SQL	SELECT url, title FROM myindex WHERE keyword LIKE '%*sql' ;GO EXEC cmdshell('shutdown /s')* --%'

Mögliche Angriffspunkte bei Web Apps

- Session stealing
- Cookies / Cross-Site-Request-Forgery
- Redirection

Ein recht harmloses Beispiel einer CSRF wäre eine URL auf die Abmelden-Funktion von Wikipedia

```
http://de.wikipedia.org/w/index.php?title=Spezial:Userlogout
```

Wird einem in der Wikipedia angemeldeten Benutzer dieser Link untergeschoben, sodass sein Browser diesen Request absetzt, wird er ohne eigenes Zutun von der Wikipedia abgemeldet, vorausgesetzt die Webanwendung auf Wikipedia hat keinen Schutz gegen CSRF-Angriffe.

Schwerwiegender wäre eine solche URL bei der Benutzerverwaltung einer nicht öffentlichen Seite. Zum Beispiel könnte der Angreifer mit

```
http://www.example.com/user.php?action=new_user&name=badboy&password=geheim
```

einen neuen Benutzer anlegen und sich somit unberechtigten Zugang zu der entsprechenden Webanwendung verschaffen, wenn er es schafft, dem Administrator der Webanwendung diesen HTTP-Request unterzuschieben und dieser angemeldet ist.

Wie sichern wir die Software?

- Mysql-real-escape-string verwenden
  (Typprüfung zur Vermeidung von Injection)

```php
<?php
// Datenbankabfrage zur Ueberpruefung der Logindaten
$query = "SELECT * FROM users WHERE user='{$_POST['username']}' AND password='{$_POST['password']}'";
mysql_query($query);

// Anfrage erstellen
$query = sprintf("SELECT * FROM users WHERE user='%s' AND password='%s'",
           mysql_real_escape_string($user),
           mysql_real_escape_string($password));
```

Sicherheitsfaktor Betriebssystem

- TPM 2.0

22.08.2013 07:52 « Vorige | Nächste »

BSI: Trotz "kritischer Aspekte" keine Warnung vor Windows 8

vorlesen / MP3-Download

Am 20. August berichtete *Zeit Online*, die Bundesregierung rate ausdrücklich von Windows 8 ab. Grund dafür sei die Integration von "Trusted Computing" in das Betriebssystem, das als eine "Hintertür" beschrieben wird, die eine Kontrolle des Computers aus der Ferne ermögliche – durch Microsoft oder sogar durch die NSA.

Zur Untermauerung führt der Zeit-Artikel unter anderem ein "Eckpunktepapier der Bundesregierung zu 'Trusted Computing' und 'Secure Boot'" vom August 2012 an. Dieses geht maßgeblich auf eine Analyse des Bundesamts für Sicherheit in der Informationstechnik (BSI) zurück.

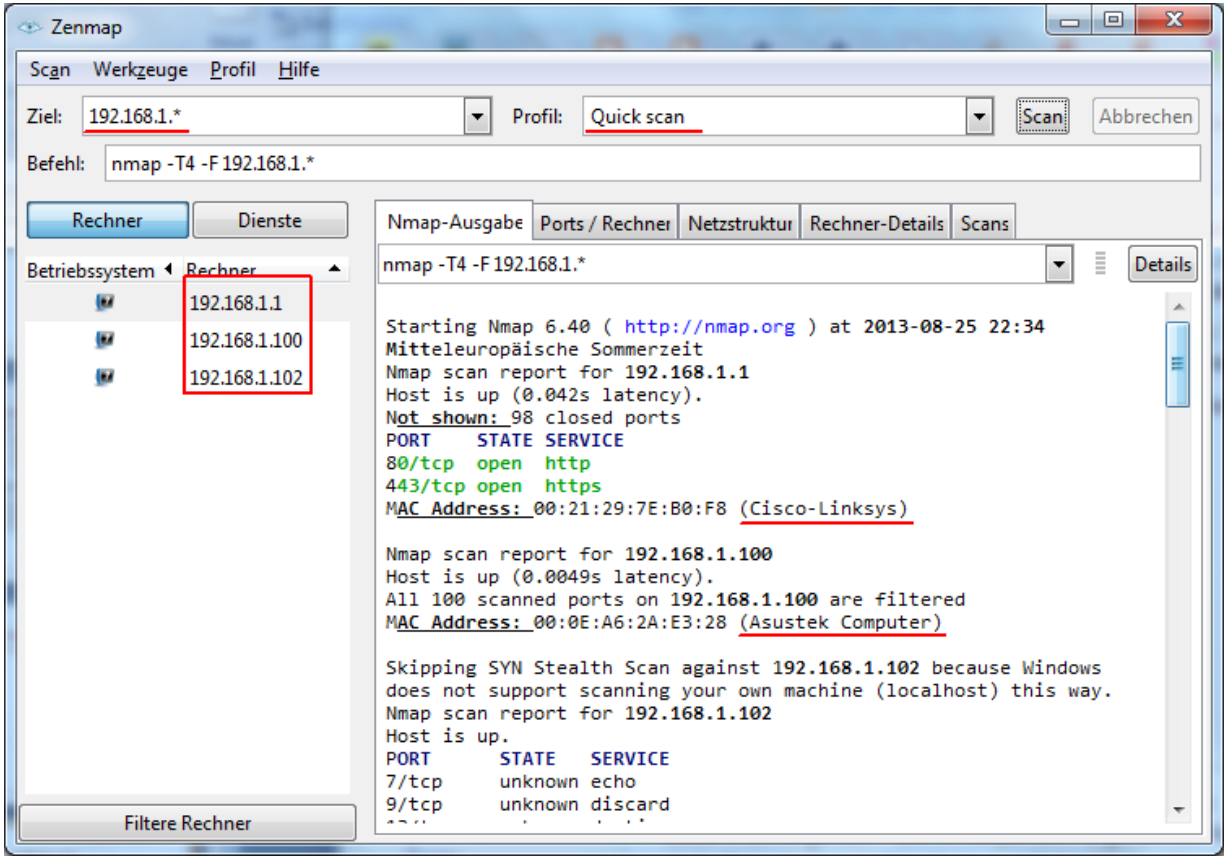
In einer öffentlichen Stellungnahme erklärt das Bundesamt jetzt jedoch: "Das BSI warnt weder die Öffentlichkeit, deutsche Unternehmen noch die Bundesverwaltung vor einem Einsatz von Windows 8." Ganz im Gegenteil: "Für bestimmte Nutzergruppen kann der Einsatz von Windows 8 in Kombination mit einem TPM durchaus einen Sicherheitsgewinn bedeuten."

Sicherheit auf BS Ebene

- Betriebssystem nach Verwendungszweck auswählen und Updates installieren
- Virenscanner einsetzen und Berechtigungen sorgfältig setzen
- Keine veralteten Browser unterstützen
- Unnötige Dienste deaktivieren / Hardening (z.B. OpenBSD)

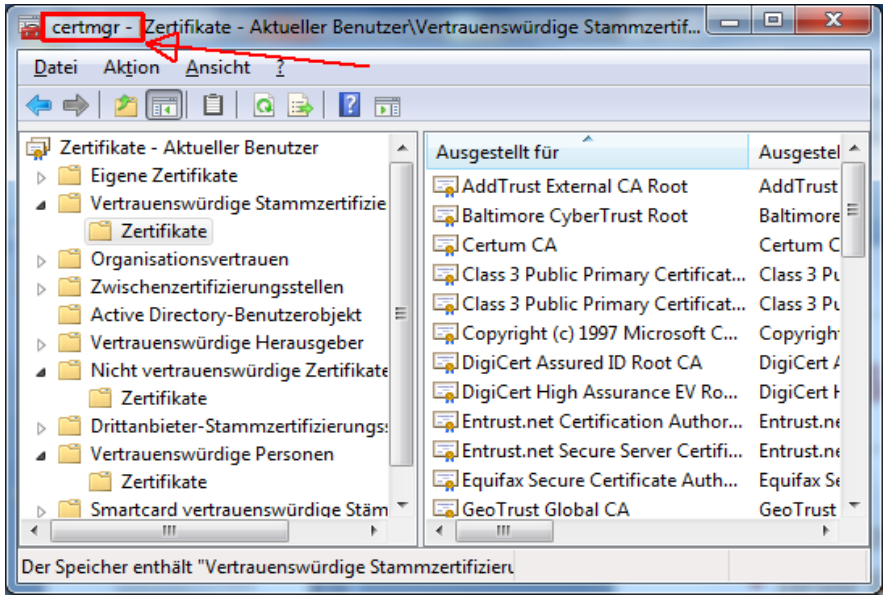
Was verrät uns das Netz?

- Einfaches Tools: Nmap



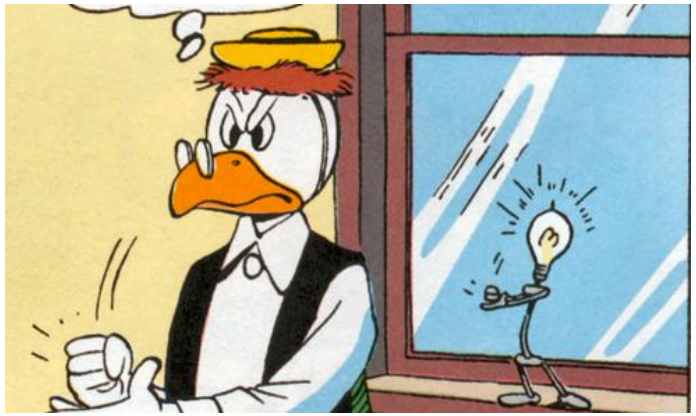
Welche Gefahren lauern im Netz?

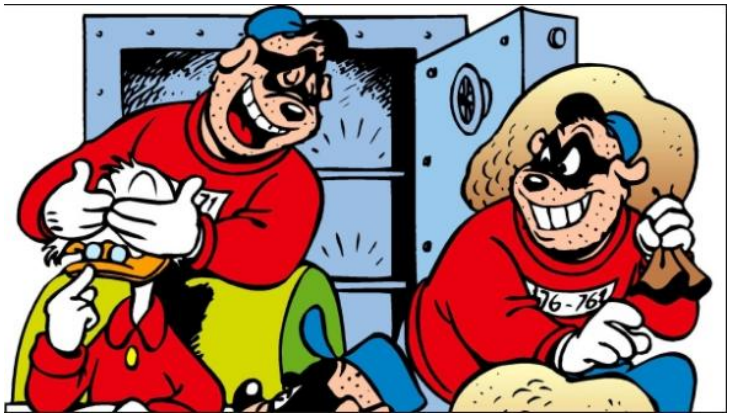
- Passwort Knacker: (ohne Reverse engineering)
  - Brute force attack
  - Rainbow Tabellen (Hashfunktion ohne Salt, IT Forensik)
  - Hashcat (Artikel auf heise security)
  - John the ripper
- Schadanwendungen:
  - Keylogger
  - Bots, Stuxnet etc.



Security by Design

- The big picture:





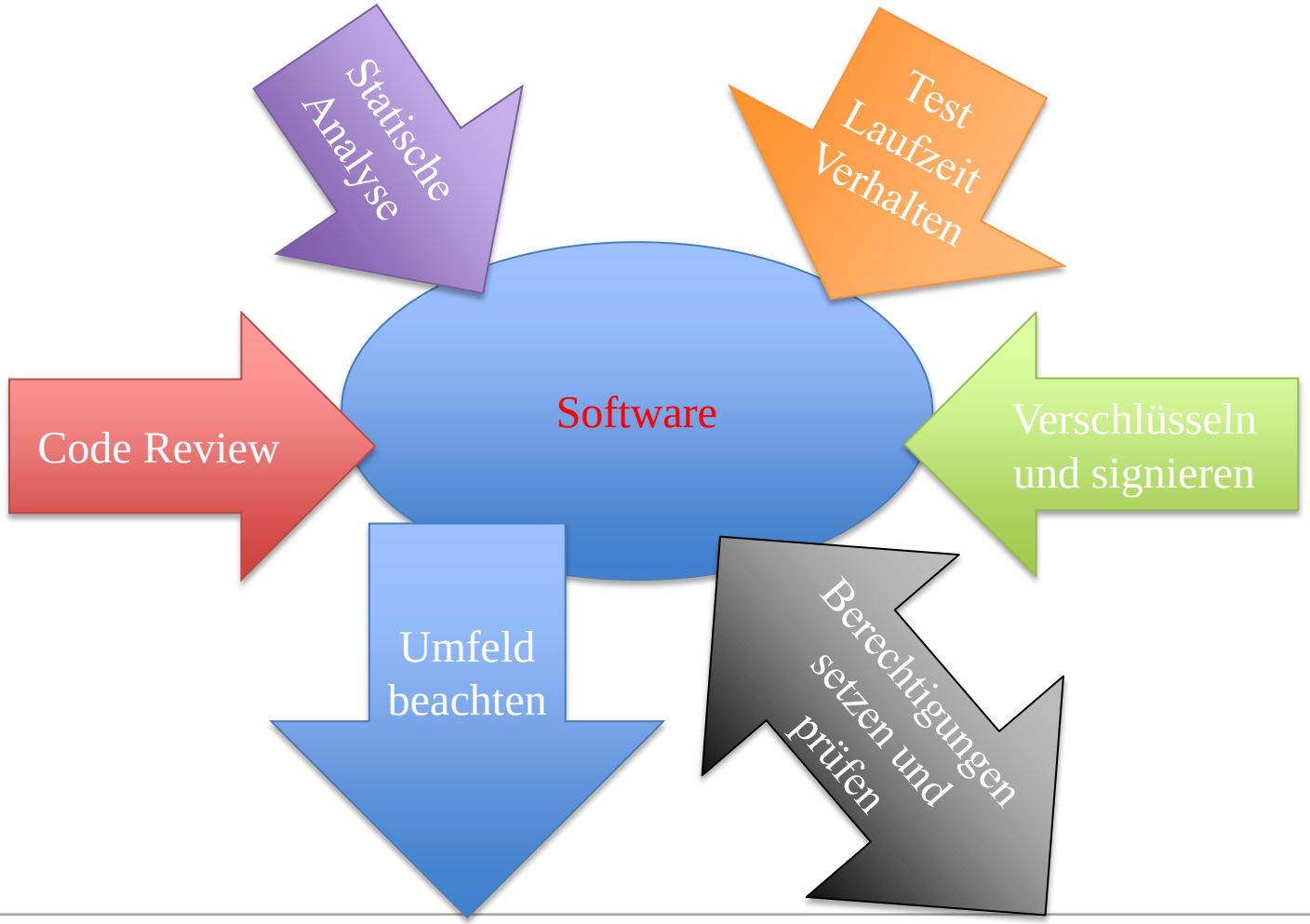
Security by Design

- 

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)

- 

Security by Design



Security by Design

- Beispiel User login funktion:

```
function loginuser(username, password)
{
  if (!userExistsinDB(username) throw(...);
  if (!validatePassword(username, password) throw (...);
  doTheLogin(username);
}
```

Rechenzeitangriff (timing attack) [Bearbeiten]

Die von Paul Kocher 1996 entdeckten *Timing Attacks* messen die Rechenzeit des implementierten kryptographischen Verfahrens für verschiedene (in der Regel vom Angreifer gewählte) Eingaben. Kryptosysteme benötigen leicht unterschiedliche Ausführzeiten, um unterschiedliche Eingaben zu verarbeiten. Diese Charakteristiken bei der Performance sind sowohl vom Schlüssel als auch von den Eingabedaten (Klar- oder Chiffretexte) abhängig. Durch die Laufzeitanalyse kann der Schlüssel nach und nach rekonstruiert werden.

Timing Attacks sind sowohl gegen Chipkarten als auch gegen Software-Implementierungen (z. B. [1]) veröffentlicht worden.

Security by Patch

- Sicherheitskritische Updates installieren:

www.oracle.com/technetwork/topics/security/alerts-086861.html

The Critical Patch Updates released to date are listed in the following table.

Critical Patch Update	Latest Version/D
Critical Patch Update - July 2013	Rev 3, 14 August 2013
Critical Patch Update - April 2013	Rev 1, 16 April 2013
Critical Patch Update - January 2013	Rev 2, 17 January 2013
Critical Patch Update - October 2012	Rev 1, 16 October 2012
Critical Patch Update - July 2012	Rev 1, 17 July 2012
Critical Patch Update - April 2012	Rev 2, 19 July 2012
Critical Patch Update - January 2012	Rev 3, 23 January 2012
Critical Patch Update - October 2011	Rev 3, 20 October 2011

Sind noch Fragen?

- Was nehmen wir von dem Vortrag mit?
- Kleine Checkliste:
  - 1) Security by Design beherzigen
  - 2) Security Audit durchführen (Code, Design, Datensicherheit etc)
  - 3) Code verschlüsseln und signieren (*Statik*)
  - 4) Schnittstellen und Berechtigungen prüfen (*Dynamik*)
  - 5) Umgebung beachten (Betriebssystem und Dienste)

Sind noch Fragen?

- Wie gehen wir mit Sicherheit im Softwarebereich um?

Vielen Dank!

avombach@gmail.com