- TCPA und Linux -
Ein Chip sie zu knechten?

Stephan Uhlmann
<su@su2.info>

22.02.2004

Inhalt

1. Einleitung

2. Technischer Hintergrund

   (a) Architektur des TPM
   (b) Funktionsweise

3. Demo

   (a) IBM Toolkit
   (b) Enforcer Linux Security Module

4. Diskussion

Einleitung

Trusted Computing

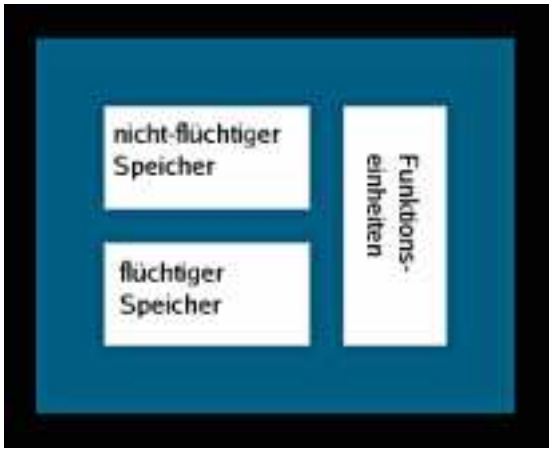
- Trust = Vertrauen
- Vertrauen Dritter gegenüber meinem Computer
- Misstrauen gegenüber dem Benutzer des Computers

Einleitung

- Trusted Computing Group (TCG)
  - ehem. Trusted Computing Platform Alliance (TCPA)
  - Microsoft, Intel, IBM, HP, ...
  - definiert Standards für ein “kryptographisches Subsystem”
- Trusted Platform Module (TPM)
  - a.k.a. “Fritz-Chip”
  - Chip zur Umsetzung des Standards auf Hardwareseite
  - ähnlich einer fest verdrahteten Smart Card

TPM Architektur

- Funktionseinheiten
- nicht-flüchtiger Speicher
- flüchtiger Speicher



TPM Funktionseinheiten

- Zufallszahlengenerator (RNG)
- Hash-Einheit (SHA-1)
- HMAC-Einheit
- RSA-Schlüsselerzeugung (bis 2048 Bit)
- RSA Ver- und Entschlüsselung, Signierung

TPM nicht-flüchtiger Speicher

- Endorsement Key, EK (2048 Bit RSA)
  - wird bei der Chip-Herstellung zufällig erzeugt
  - identifiziert das TPM (und damit den Computer)
  - kann nicht bzw. nur mit hohem Aufwand entfernt oder ausgetauscht werden
- Storage Root Key, SRK (2048 Bit RSA)
  - wird beim TPM_TakeOwnership Kommando erzeugt
  - privater Schlüssel bleibt im TPM
  - öffentlicher Schlüssel kann exportiert werden
  - verschlüsselt eigene Schlüssel beim Import ins TPM

TPM flüchtiger Speicher

- RSA Schlüssel Speicher, “Slots” (0 bis 9)
  - 2048 Bit
- Platform Configuration Registers, PCR (0 bis 15)
  - 160 Bit Hashes
  - zur Überprüfung der Software-Integrität (BIOS, MBR, Kernel, ...)

TPM Funktionsweise

- initiale Besitzübernahme (TakeOwnership)
  - bedarf physikalischer Anwesenheit des Benutzers (“physical presence”)
  - Erzeugung des SRK (verschlüsselt mit EK)
- Erstellung eigener Schlüssel
- Import eigener Schlüssel ins TPM
  - Verschlüsselung mit SRK (“wrapping”)
  - weiterer Zugriff nur mittels SRK

TPM Funktionsweise

- Versiegeln/Entsiegeln (sealing/unsealing) von Daten
  - Ver- bzw. Entschlüsselung mit dem SRK (eigene Schlüssel gehen nicht)
  - Bindung an das System (PCR-Register) -> Daten können nur auf dem selben System im selben Zustand entschlüsselt werden
  - max. 256 Bytes (reicht für Passphrases)
- Bildung digitaler Signaturen von Dateien
- Belegung und Abfrage der PCR-Register
- “(remote) attestation”: Beglaubigung der Systemintegrität

Hersteller

- IBM Thinkpad’s seit Oktober 2002 (T30, T40, A31p, ...)
- TPM Chips: Atmel AT97SC3201 und Infineon SLD 9630TT1.1
  - EEPROM für nicht-flüchtigen Speicher
  - angeschlossen am LPC Bus
  - RSA Hardwarebeschleunigung

Demo IBM Toolkit

- libtcpa
- Kernel-Modul tpm.o
- Gerätedatei /dev/tpm (character device, major 10, minor 224)
- Standardfunktionen:
  - take_own
  - sealfile, unsealfile
  - createkey, loadkey, evictkey
  - signfile, verifyfile

Demo Enforcer Linux Security Module

- SHA-1 Datenbank von wichtigen Dateien
- Enforcer überprüft beim Laden einer Datei ob Hash-Wert mit dem in einer Datenbank übereinstimmt
- ähnlich Tripwire
- Datenbank kann signiert und versiegelt werden (TPM gibt Datenbank nur im “sicheren” Zustand frei)
- TCPA enabled LILO
  - schreibt Hashes von MBR und Kernel in PCR-Register

Diskussion

Gutes TCPA

- TPM: sichere Schlüsselaufbewahrung
- Systembindung bestimmter Daten (Bsp. Passwörter)
- TCG Mitglieder: keine unsichere Software (Viren, Trojaner, Bufferoverflows, ...) mehr

Diskussion

Böses TCPA

- DRM - Digital Restrictions Management!
- Hersteller hat Zugriff auf EK
- EK wahrscheinlich aus Kostengründen ausserhalb des TPM erzeugt
- Backdoors? nicht überprüfbar (Microsoft: “...never voluntarily...”).
- EEPROM’s haben begrenzte Schreib-/Lösch-Zyklen (ca. 500 -> SRK nicht beliebig oft erzeugbar)

Diskussion

Auswirkungen für Open Source

- Bindung von Dokumenten an Applikationen (-> kein Word.doc in OpenOffice mehr)
- nur zertifizierte Applikationen miteinander “kompatibel” (-> Samba!?)
- hoher Aufwand für Zertifizierung
  - für Open Source Projekte kaum machbar

Diskussion

- CCC: 4 Forderungen
  - vollständige Kontrolle über alle Schlüssel
  - keine verborgenen Kanäle
  - Übertragung von Schlüsseln auf andere PCs
  - Transparenz der Zertifizierungs-Mechanismen
- EFF: owner override

Diskussion

Änderungen der TCG-Spezifikation (v1.2)

- Direct Anonymous Attestation
  - verschiedene Attestation Identity Keys (AIK) möglich
- Löschen des Endorsement Keys
  - Aufwand für Neuerstellung?

Referenzen

1. TPM Spezifikation v1.2, TCG
   https://www.trustedcomputinggroup.org

2. TCPA Device Driver for Linux, IBM Global Security Analysis Lab
   http://www.research.ibm.com/gsal/tcpa/

3. Enforcer Linux Security Module
   http://enforcer.sourceforge.net/

4. “Trusted or Treacherous?”, Votrag von Rüdiger Weis und Andreas Bogk auf dem Chaos Communication Congress 2003
   http://www.ccc.de/congress/2003/fahrplan/event/542.de.html

5. “Take Control of TCPA”, Safford/Kravitz/Doorn
   http://www.linuxjournal.com/article.php?sid=6633

6. “Trusted Computing: Promise and Risk”, Seth Schoen, Electronic Frontier Foundation
   http://www.eff.org/Infra/trusted_computing/20031001_tc.php

7. CCC Forderungen zu TCPA
   http://www.ccc.de/digital-rights/forderungen

