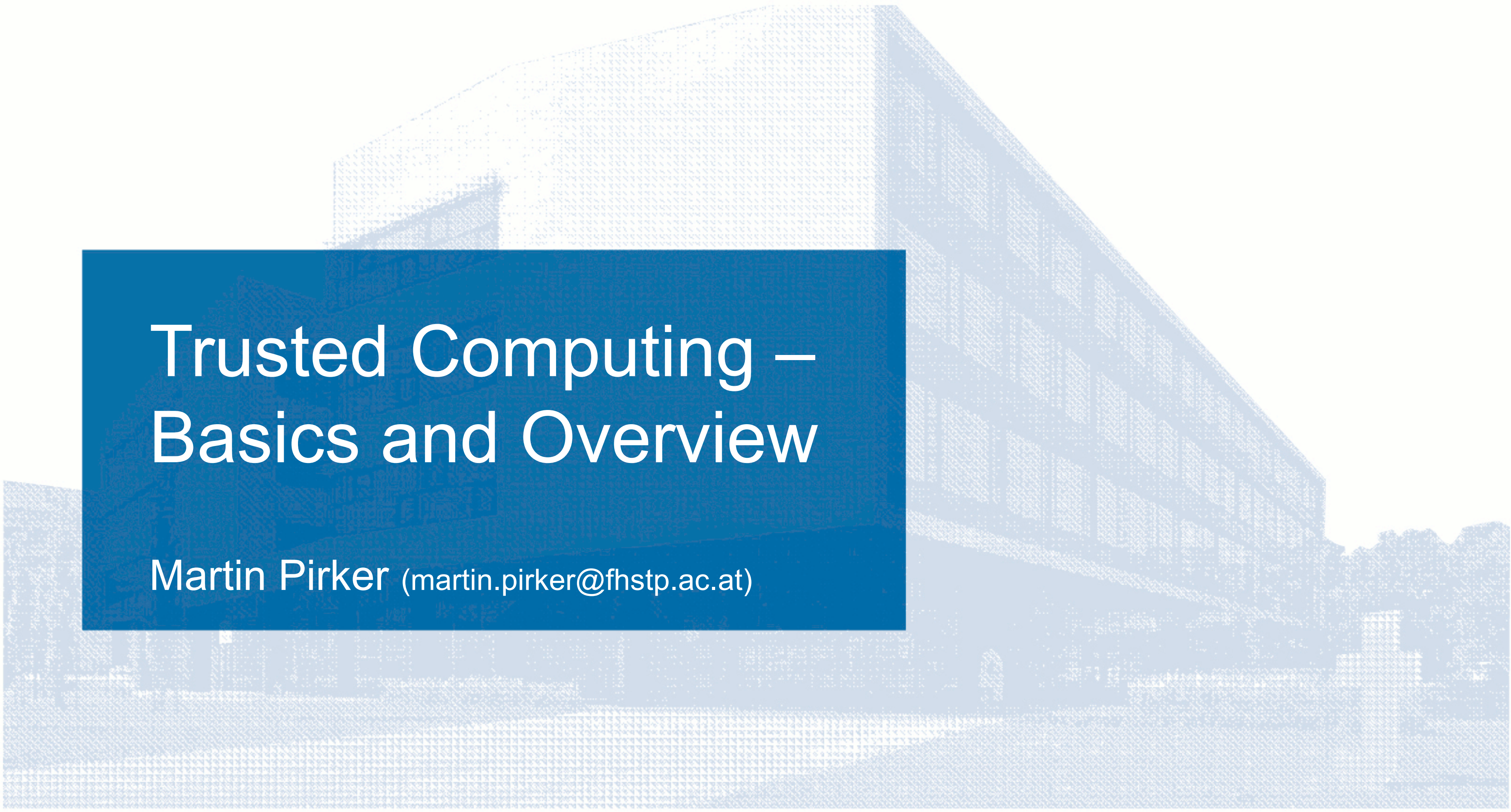


Trusted Computing – Basics and Overview

Martin Pirker (martin.pirker@fhstp.ac.at)

Motivation

- Do you trust your PC/Notebook to be unmodified?
  - Why?

Motivation

- Do you trust your PC/Notebook to be unmodified?
  - Why?
- If you don't know for sure that your operating system is really in the state it claims to be – why trust any software running on top of it?

Agenda

- Trust == ?

- Trusted Platform Module (TPM)
- TCG Software Stack (TSS)
- Trusted Execution Technology (TXT)
- Trusted Execution Environment (TEE)
- Software Guard Extensions (SGX)

- Incomplete system state(s) information
  → assessment?

Trust == ?

- Trust is expectation of behaviour

- A *trusted* system is one that *behaves in the expected manner for a particular purpose* [TCG]

Note: Trust is not necessarily good behaviour

Trust is...

- ...when something can be unambiguously identified
- ...when it operates unhindered
- ...when the user has first hand experience of consistent, good behaviour
- ...or when the user trusts someone who has provided references for consistent, good behaviour
- ...ultimately derived from people (hence organisations)

Platform (Security) State

- Example virus scanner
  - search platform for modifications
  - “all clear” → “opinion” of virus scanner

- Attacker and defence operate on the same level

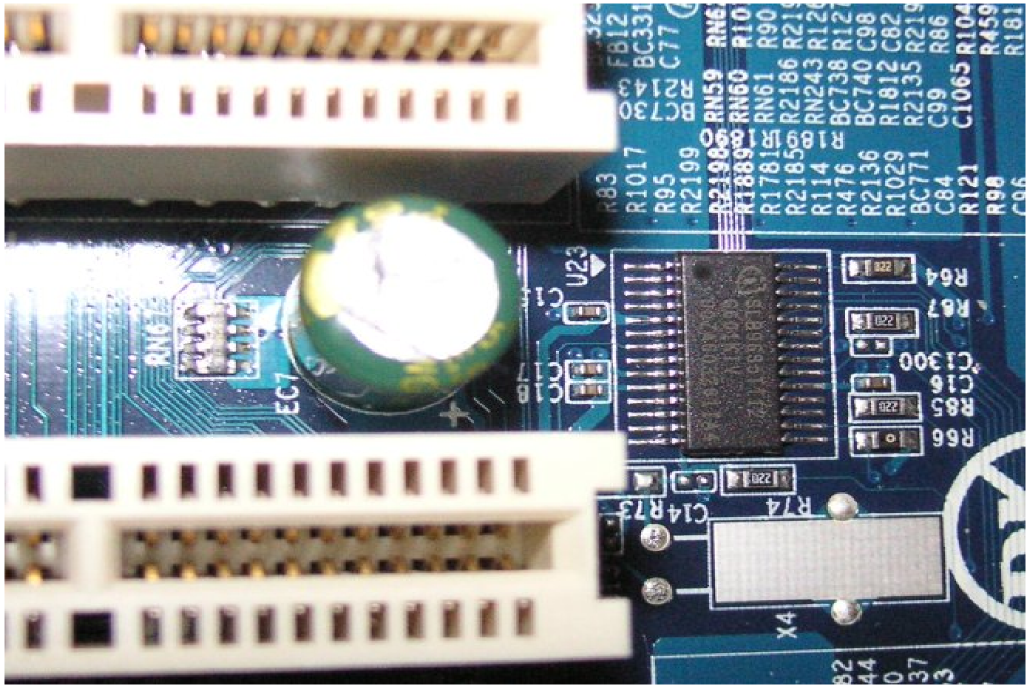
- System software compromised once – system security very likely broken forever

Trusted Platforms

- Dedicated hardware support for system state evaluation and/or reporting and/or protection of certain programs

- Available in mass-market platforms
  - Trusted Platform Module (TPM)
  - Intel Trusted Execution Technology (TXT)
  - Intel Software Guard Extensions (SGX)
  - ARM TrustZone

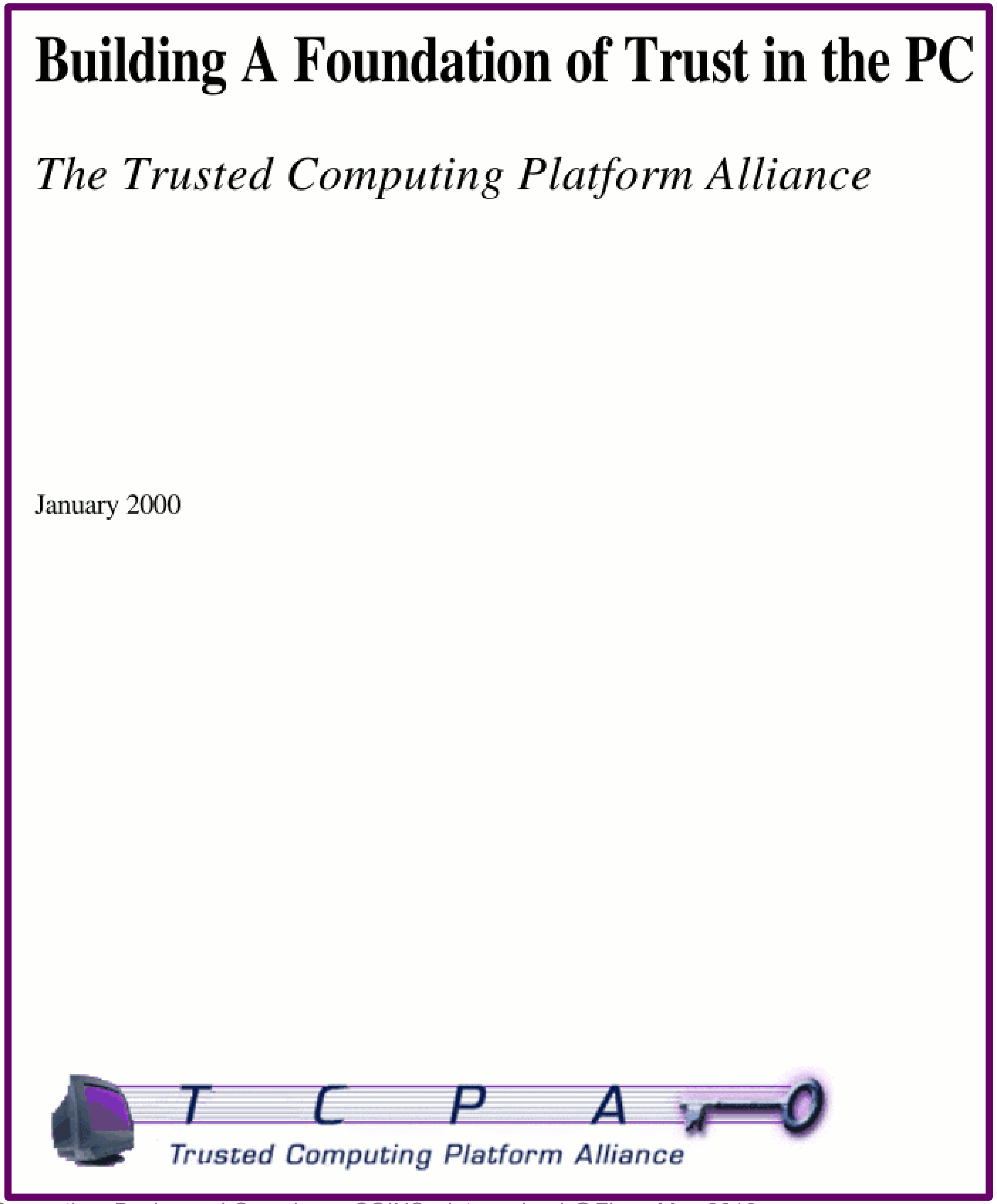
Trusted Platform Module



TPM Quick Overview

- Provide support functions for measurement and recording of software running on the platform
- Protect cryptographic keys
- Protect pieces of data
- Produce signed reports of the platform state
- ...

- ....TPM is a hardware *Root of Trust*



Building A Foundation of Trust in the PC

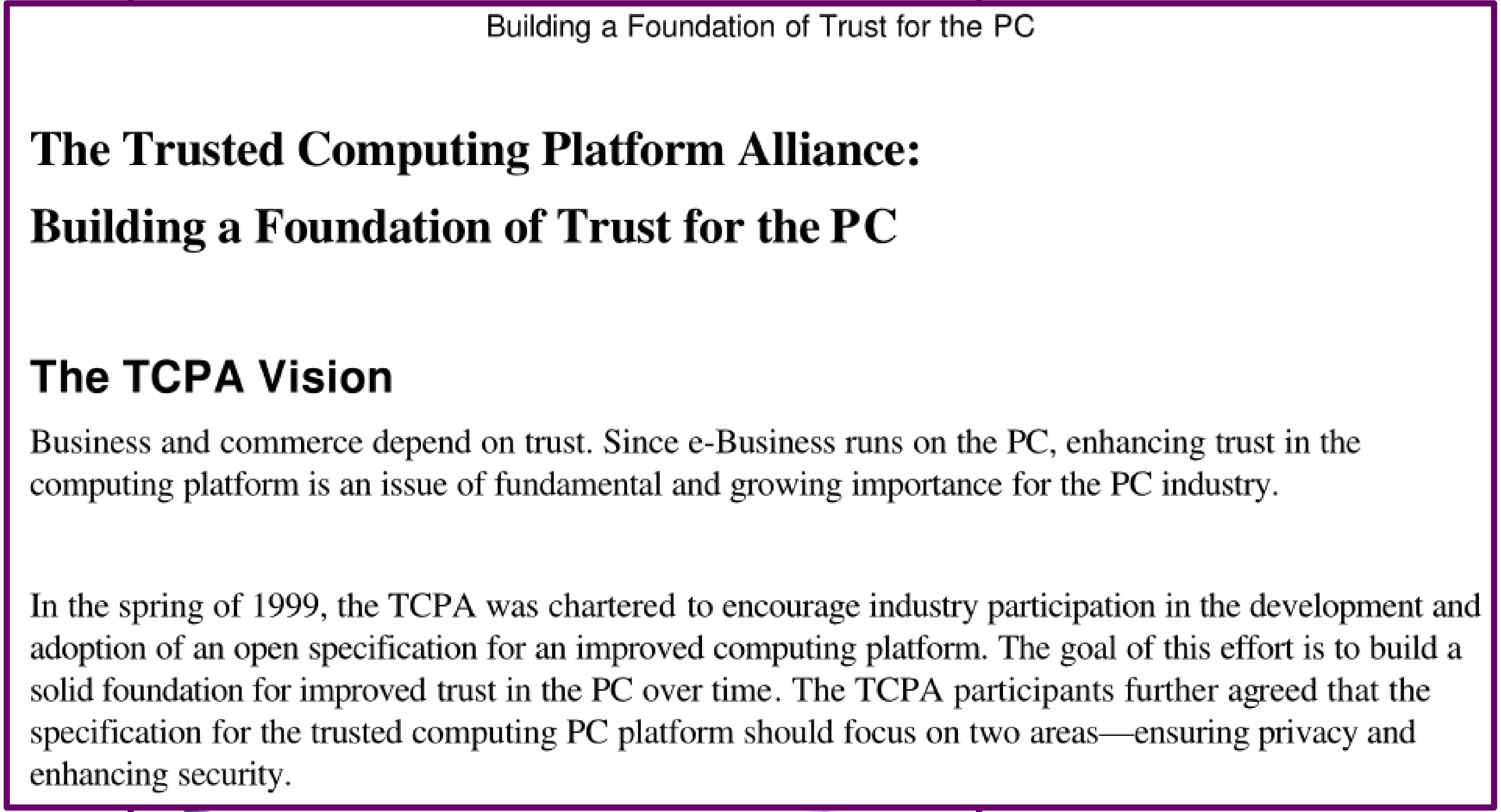
The Trusted Computing Platform Alliance

January 2000

TCPA

Trusted Computing Platform Alliance

Building A Foundation of Trust in the PC



Building a Foundation of Trust for the PC

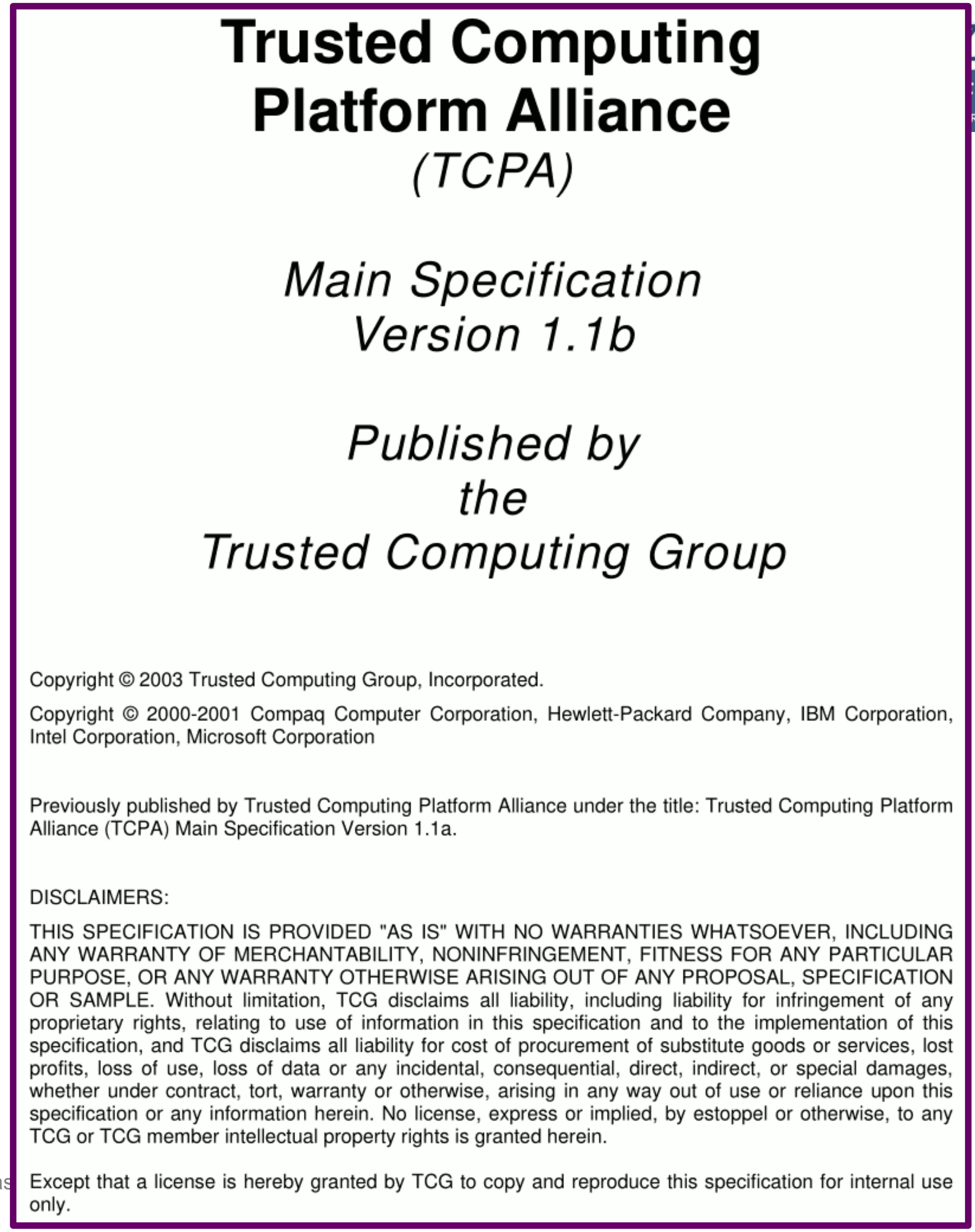
The Trusted Computing Platform Alliance:

Building a Foundation of Trust for the PC

The TCPA Vision

Business and commerce depend on trust. Since e-Business runs on the PC, enhancing trust in the computing platform is an issue of fundamental and growing importance for the PC industry.

In the spring of 1999, the TCPA was chartered to encourage industry participation in the development and adoption of an open specification for an improved computing platform. The goal of this effort is to build a solid foundation for improved trust in the PC over time. The TCPA participants further agreed that the specification for the trusted computing PC platform should focus on two areas—ensuring privacy and enhancing security.



Trusted Computing Platform Alliance
(TCPA)

Main Specification
Version 1.1b

Published by
the
Trusted Computing Group

Copyright © 2003 Trusted Computing Group, Incorporated.

Copyright © 2000-2001 Compaq Computer Corporation, Hewlett-Packard Company, IBM Corporation, Intel Corporation, Microsoft Corporation

Previously published by Trusted Computing Platform Alliance under the title: Trusted Computing Platform Alliance (TCPA) Main Specification Version 1.1a.

DISCLAIMERS:

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein. No license, express or implied, by estoppel or otherwise, to any TCG or TCG member intellectual property rights is granted herein.

Except that a license is hereby granted by TCG to copy and reproduce this specification for internal use only.

History

~2000	Trusted Computing Platform Alliance (TCPA)
Feb'02	TPM v1.1b specification
~2003	Trusted Computing Group (TCG)
Oct'03	TPM v1.2 rev. 62
Mar'11	TPM v1.2 rev.116
Oct'12	TPM v2.0 rev. 93
Jul'16	Microsoft requires all new Win 10 PC to be TPM 2.0 enabled

History

2005-2009:

EU Project Open Trusted Computing

23 partners, from all over Europe investigate opportunities of trusted computing technologies

TPM v1.2 Building Blocks

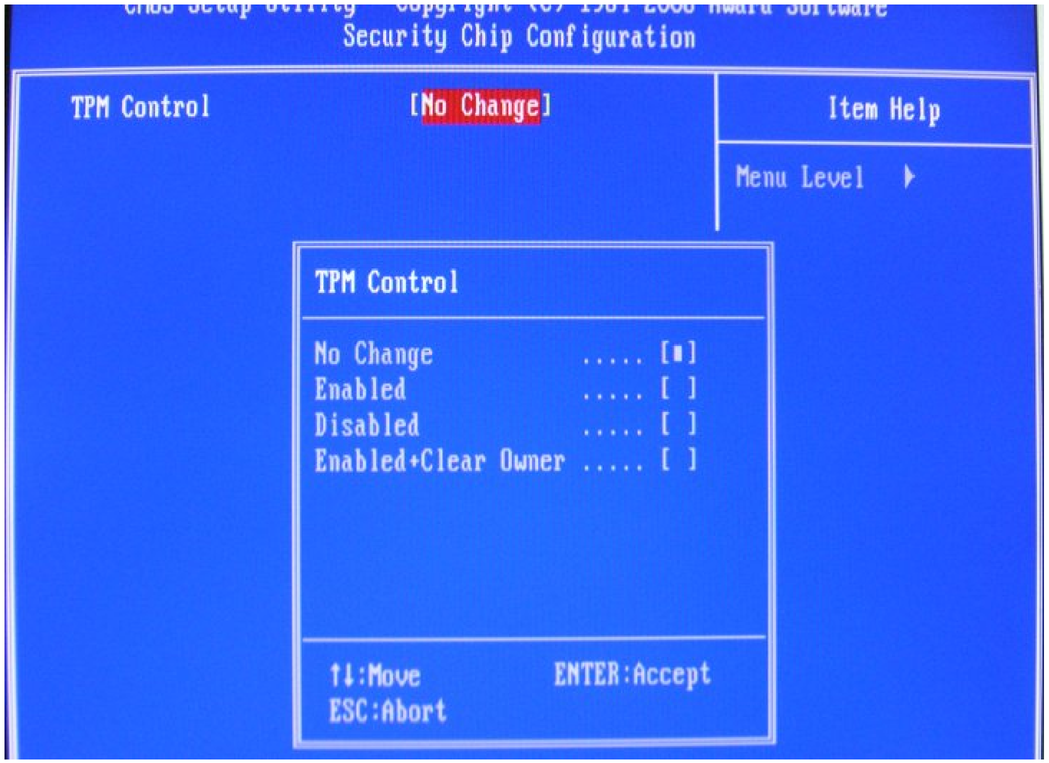
- Hardware Random Number Generator (RNG)
- RSA engine
- SHA-1 engine
- HMAC engine
- Volatile memory (working memory)
- Non-volatile memory (very limited)
- I/O connection to LPC bus
- CPU and firmware
- ….

TPM Properties

- Slave device
- Does not initiate operations or communications with other devices
- Consequently, the TPM cannot alter the execution flow of the system (e.g. booting or execution of applications)
- TPM is a *low-cost* mass-market device
  - limited resistance against sophisticated hardware attacks
  - better than pure software solutions

TPM BIOS control

- ForceClear a TPM
- change TPM status (disabled/enabled)



```
                    Security Chip Configuration

   TPM Control              [No Change]          Item Help

                                                 Menu Level  ▸

                  TPM Control

                  No Change            ..... [■]
                  Enabled              ..... [ ]
                  Disabled             ..... [ ]
                  Enabled+Clear Owner  ..... [ ]

                   ↑↓:Move          ENTER:Accept
                   ESC:Abort
```

Chain of Trust

- How to measure/record what software is/was running?
  - Requires monitoring of boot process
  - Needs anchor to start the measurements from, a Root of Trust
  - Nobody should be able to modify or forge the measurement
  - A shielded location for storage of the measurement
- How to report that platform is in a defined state?
  - Why should someone believe a claimed system state?
  - Needs mechanism to securely report the state/measurements to a third party

TPM PCRs

- Platform Configuration Registers (PCRs)
  - 160 bit storage location per PCR
  - 24 PCRs in every TPM V1.2
  - can always be read from
  - can never be directly modified, but only *extended* with

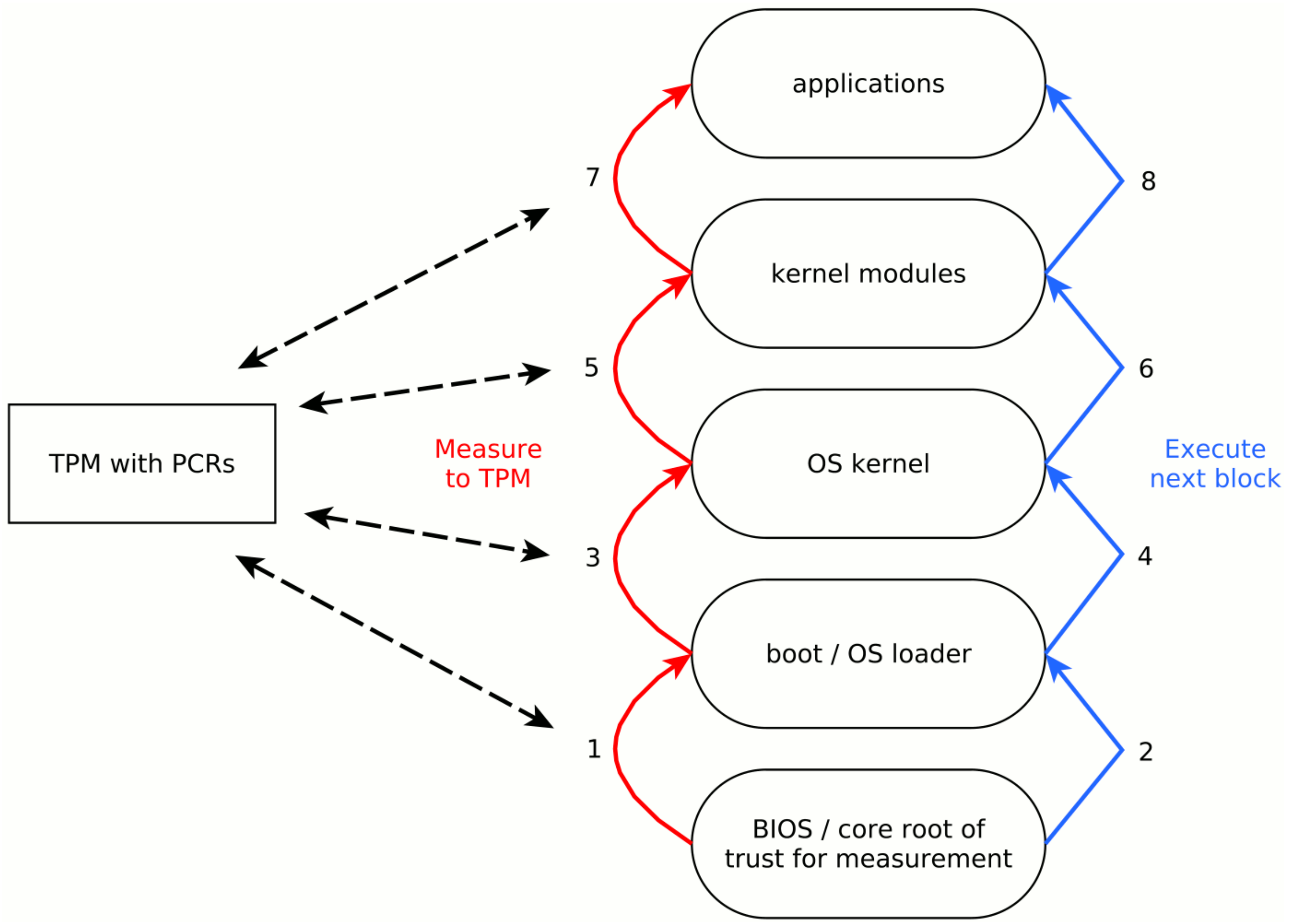
    $PCR_{t+1}[i] = SHA\text{-}1(\ PCR_t[i]\ ||\ newValue\ )$

  - static PCRs:
    - reset only at boot time
  - dynamic PCRs:
    - reset can only be triggered by special mechanism (e.g. TXT SINIT)
  - Note: PCR extends are not commutative (i.e. measuring A then B does not result in the same PCR value as measuring B then A)

PCRs After Reboot

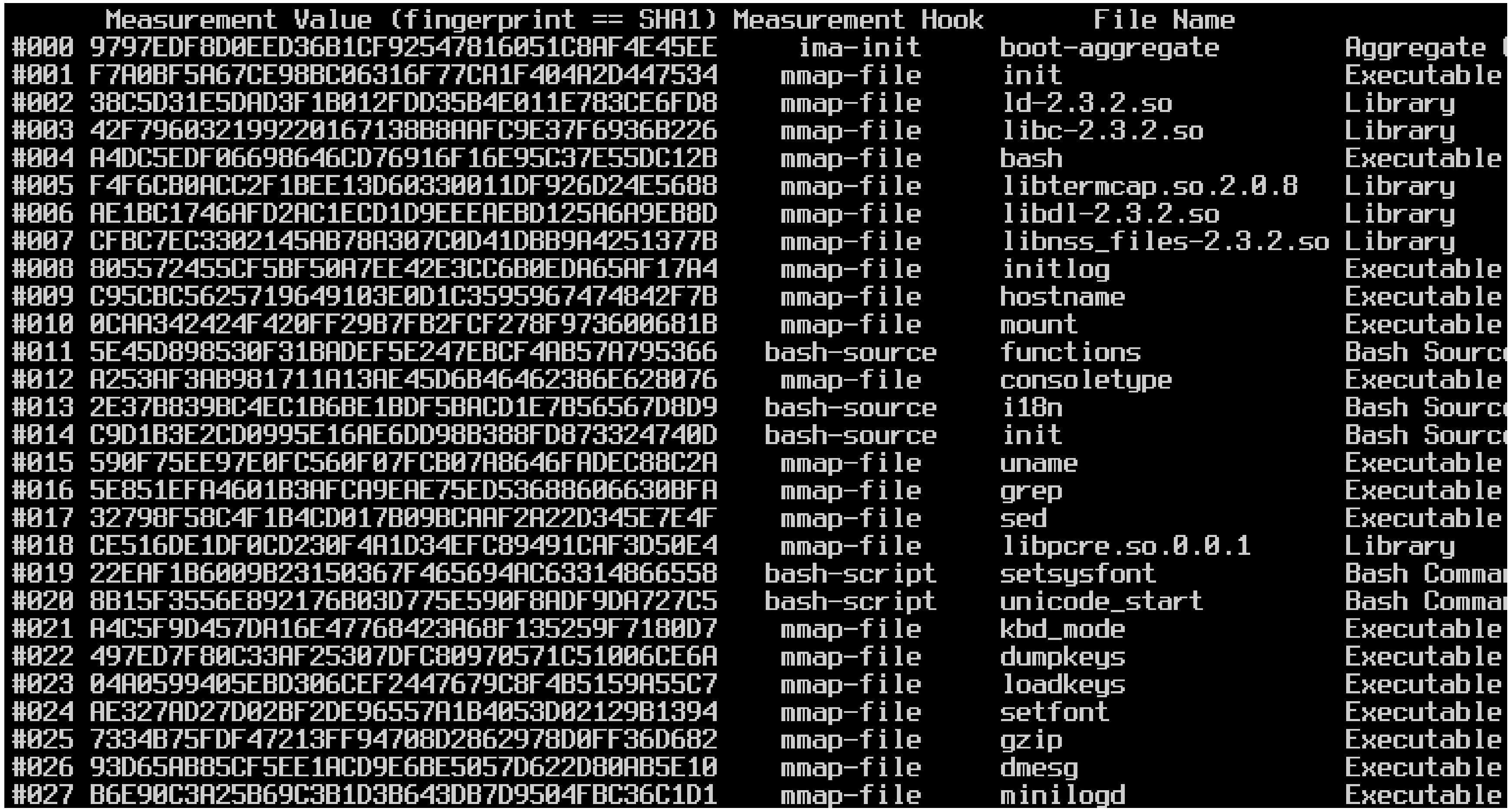
```
00:  e5 58 e7 ec 7f 82 99 c7 4a 24 63 6e 24 82 56 de f1 25 19 de
01:  a7 46 df 87 49 64 75 fe 5c b2 14 47 f3 8e 66 46 04 57 67 5b
02:  05 a8 c0 53 ed 4b 3d 48 ea 43 f4 64 f4 9a 96 b6 0b 05 0a 72
03:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
04:  4c 1c c1 30 4e 70 9b 16 fc f1 06 aa 44 9b 8b bb 90 bc 34 ac
05:  c3 1a 3d e8 6d 3f 46 bf 1a 01 03 39 c9 d6 3c 55 fa 42 c1 23
06:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
07:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
08:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
09:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
11:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
12:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
13:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
14:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
15:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
17:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
18:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
19:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
20:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
21:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
22:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
23:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Chain of Trust



Tail of Chain of Trust

- Ex.: IBM's Integrity Measurement Architecture (IMA)



```
      Measurement Value (fingerprint == SHA1) Measurement Hook      File Name
#000 9797EDF8D0EED36B1CF92547816051C8AF4E45EE     ima-init     boot-aggregate        Aggregate (
#001 F7A0BF5A67CE98BC06316F77CA1F404A2D447534    mmap-file     init                  Executable
#002 38C5D31E5DAD3F1B012FDD35B4E011E783CE6FD8    mmap-file     ld-2.3.2.so           Library
#003 42F7960321992201671388B8AAFC9E37F6936B226   mmap-file     libc-2.3.2.so         Library
#004 A4DC5EDF06698646CD76916F16E95C37E55DC12B    mmap-file     bash                  Executable
#005 F4F6CB0ACC2F1BEE13D60330011DF926D24E5688    mmap-file     libtermcap.so.2.0.8   Library
#006 AE1BC1746AFD2AC1ECD1D9EEEAEBD125A6A9EB8D    mmap-file     libdl-2.3.2.so        Library
#007 CFBC7EC3302145AB78A307C0D41DBB9A4251377B    mmap-file     libnss_files-2.3.2.so Library
#008 805572455CF5BF50A7EE42E3CC6B0EDA65AF17A4    mmap-file     initlog               Executable
#009 C95CBC5625719649103E0D1C35959674748 42F7B   mmap-file     hostname              Executable
#010 0CAA342424F420FF29B7FB2FCF278F973600681B    mmap-file     mount                 Executable
#011 5E45D898530F31BADEF5E247EBCF4AB57A795366   bash-source    functions             Bash Sourc
#012 A253AF3AB981711A13AE45D6B46462386E628076    mmap-file     consoletype           Executable
#013 2E37B839BC4EC1B6BE1BDF5BACD1E7B56567D8D9   bash-source    i18n                  Bash Sourc
#014 C9D1B3E2CD0995E16AE6DD98B388FD873324740D   bash-source    init                  Bash Sourc
#015 590F75EE97E0FC560F07FCB07A8646FADEC88C2A    mmap-file     uname                 Executable
#016 5E851EFA4601B3AFCA9EAE75ED53688606630BFA    mmap-file     grep                  Executable
#017 32798F58C4F1B4CD017B09BCAAF2A22D345E7E4F    mmap-file     sed                   Executable
#018 CE516DE1DF0CD230F4A1D34EFC89491CAF3D50E4    mmap-file     libpcre.so.0.0.1      Library
#019 22EAF1B6009B23150367F465694AC63314866558   bash-script    setsysfont            Bash Comma
#020 8B15F3556E892176B03D775E590F8ADF9DA727C5   bash-script    unicode_start         Bash Comma
#021 A4C5F9D457DA16E47768423A68F135259F7180D7    mmap-file     kbd_mode              Executable
#022 497ED7F80C33AF25307DFC80970571C51006CE6A    mmap-file     dumpkeys              Executable
#023 04A0599405EBD306CEF2447679C8F4B5159A55C7    mmap-file     loadkeys              Executable
#024 AE327AD27D02BF2DE96557A1B4053D02129B1394    mmap-file     setfont               Executable
#025 7334B75FDF47213FF94708D2862978D0FF36D682    mmap-file     gzip                  Executable
#026 93D65AB85CF5EE1ACD9E6BE5057D622D80AB5E10    mmap-file     dmesg                 Executable
#027 B6E90C3A25B69C3B1D3B643DB7D9504FBC36C1D1    mmap-file     minilogd              Executable
```

Security Assessment

- Measure
  - initial state
  - event(s) transition: current state $\rightarrow$ new state
- Compare
  - measurements $\leftrightarrow$ references (authenticity problem)
- Enforcement / Monitoring
  - actual $\neq$ expected?

Practical Complexity

- Defined order of measurements (hashchain)
- Log of individual measurements (replay)
- Database of good/known measurements (huge!)
- One weak link breaks the chain...

….so only for limited scenarios? :-(

TPM v1.2 Keys

- Endorsement key (EK)
  - unique platform identity
  - injected by manufacturer OR created by platform owner
- 2048 bit RSA key contained inside the TPM, private part never leaves the TPM, is unique for every TPM and therefore uniquely identifies a TPM
- The EK should be backed by an EK certificate
  - typically an EK certificate is issued by the TPM manufacturer
- EK certificate is the only proof that one is communicating with a genuine hardware TPM with an unique EK

TPM v1.2 Keys

- Storage Root Key (SRK)
  - is root element of TPM key hierarchy
  - typically with known password
- Storage Keys
  - wrap (encrypt) other elements in the TPM key hierarchy
- Signature Keys
  - signing operations

TPM v1.2 Keys

- Binding Keys
  - key used for binding operations (TPM_Bind, TPM_Unbind)

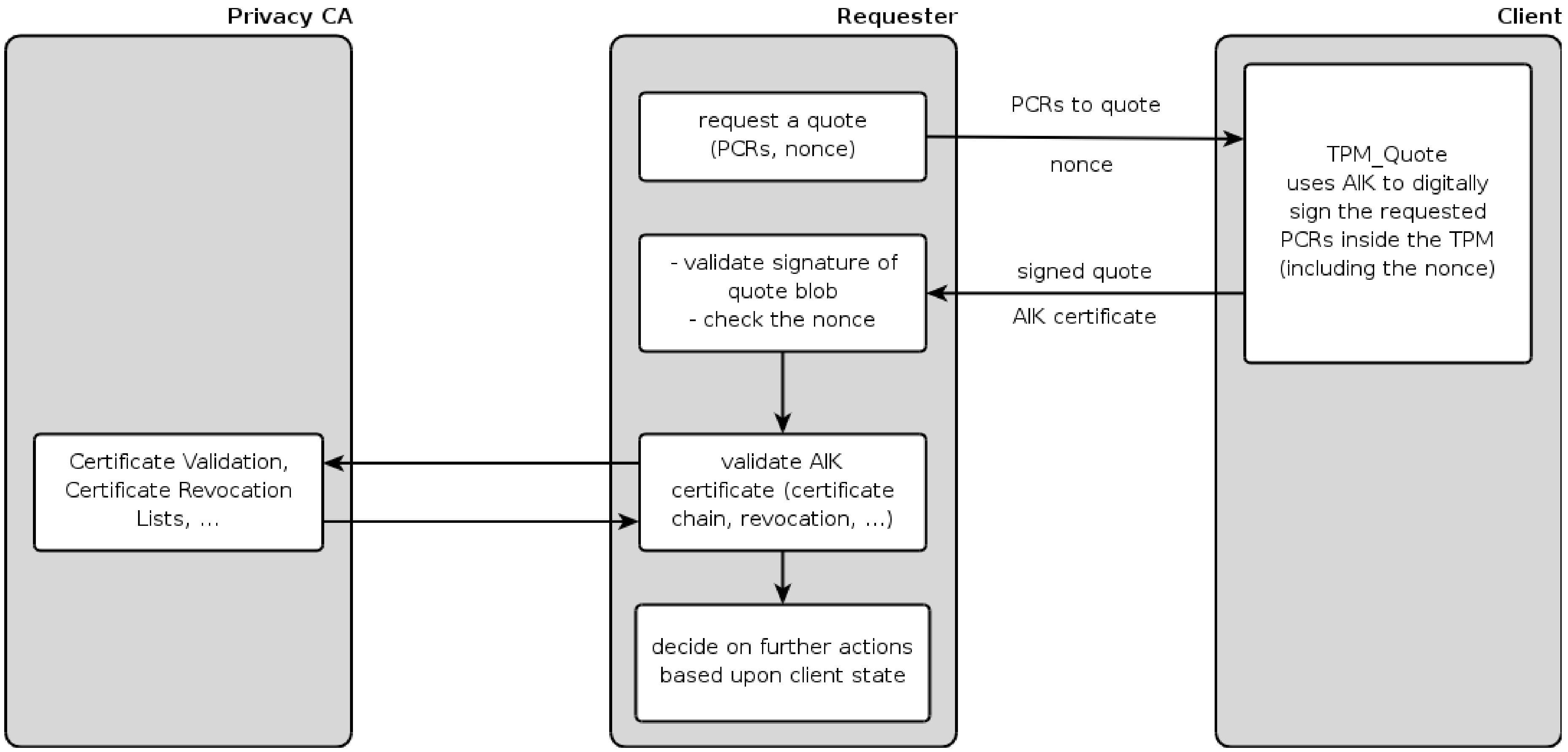
- ….

- Key Secrets
  - usage secret
    for all operations that make use of a TPM key
  - migration secret
    when migrating a key between different platforms

JRZ TARGET JOSEF RESSEL CENTER FOR UNIFIED THREAT INTELLIGENCE ON TARGETED ATTACKS

/fh/// st.pölten

Attestation


Privacy CA
Requester
Client
request a quote (PCRs, nonce)
PCRs to quote
nonce
TPM_Quote uses AIK to digitally sign the requested PCRs inside the TPM (including the nonce)
signed quote
AIK certificate
- validate signature of quote blob
- check the nonce
validate AIK certificate (certificate chain, revocation, ...)
Certificate Validation, Certificate Revocation Lists, ...
decide on further actions based upon client state

Attestation Identity Keys

- Uniqueness of the EK would be a privacy problem if the EK were used directly in digital signature operations
- EK operations very restricted, EK cannot sign

- Attestation Identity Keys (AIKs) have been introduced as alias keys for the EK. The AIKs are designed to provide privacy to users.

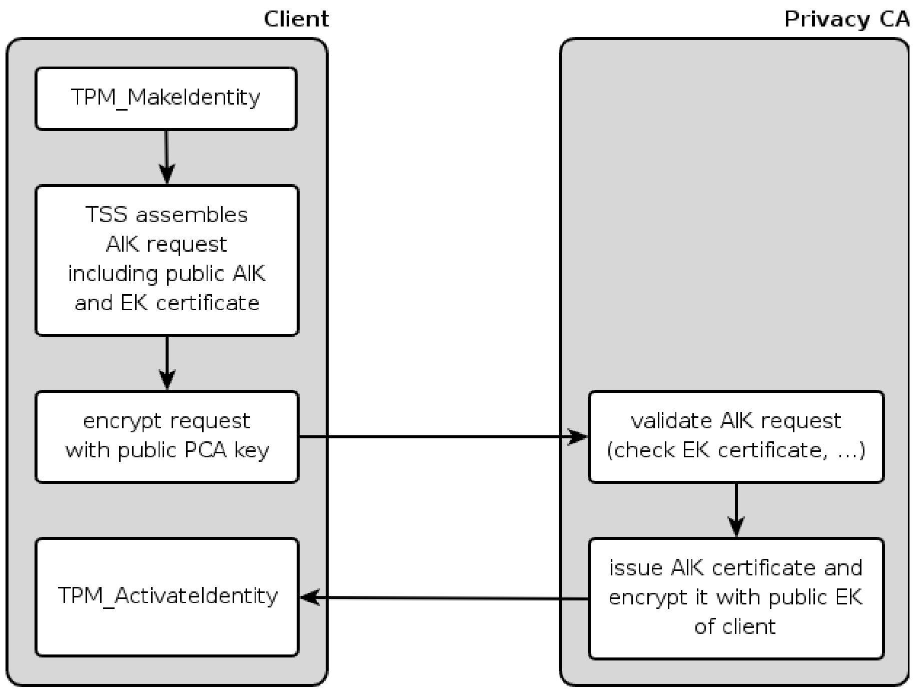
- Attestation Identity Keys (AIKs)
  - provide a mechanism to ensure that one is communicating with a TPM, but not which TPM specifically
  - a non-migratable signature key that signs only information generated inside the TPM

Obtaining AIKs

- when signing data with an AIK, the verifier wants assurance that the key is a TPM protected key
- AIKs are backed by a certificate that vouches for the fact that the AIK is such a TPM protected key
- Privacy CA:
  trusted third party that issues certificates for AIKs
- basic AIK cycle:
  - create AIK inside the TPM
  - AIK request data plus platform certificates are sent to PrivacyCA
  - PrivacyCA examines the supplied data
  - if PrivacyCA is convinced that the AIK is a TPM key it issues a certificate stating that fact

Basic AIK cycle

- client creates a new AIK key pair inside the TPM
- TSS assembles AIK request
  - TPM EK certificate
  - public AIK
  - …
- AIK request is encrypted with public key of PCA
- PrivacyCA validates request; checks EK certificate etc.; if PCA is convinced that the AIK is a proper TPM key, it issues an AIK certificate. PCA response is encrypted with public EK → only the requ. TPM can decrypt it
- client activates AIK and stores AIK certificate


Client
Privacy CA
TPM_MakeIdentity
TSS assembles AIK request including public AIK and EK certificate
encrypt request with public PCA key
TPM_ActivateIdentity
validate AIK request (check EK certificate, …)
issue AIK certificate and encrypt it with public EK of client

PCRs summary

- PCR usage scenarios
  - attestation of platform state (TPM_Quote)
  - protecting data (TPM_Seal/TPM_Unseal)
  - specify set of PCRs upon key creation:
    key is only usable if these PCRs are present
- Collection of measurements is done outside of the TPM
- Challenges
  - chain of trust must not be broken
  - what to measure (binaries, configuration files, scripts, ...)
  - how to handle system updates?
  - pool of measurement values can become very large

Advanced TPM concepts

- Key Migration and Certified Migratable Keys
- Monotonic Counters and Timestamping
- Delegation
- Manufacturer functions
- Localities
- Direct Anonymous Attestation
  - Anonymous groups signature based replacement for PrivacyCA
- ….

TPM Security

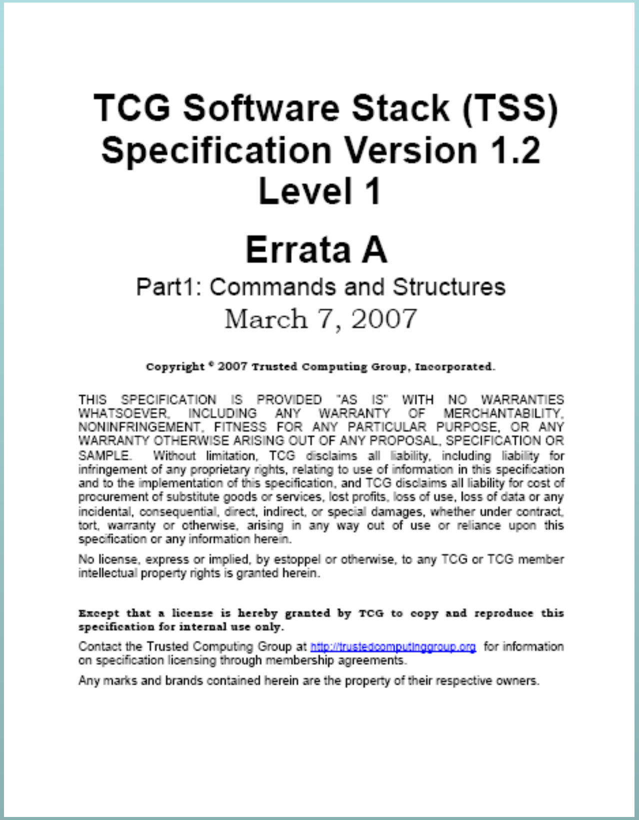
- TPMs are cheap – mass-market chips
- Deconstructing a Secure Processor

  Christopher Tanovsky @Blackhat DC (Feb 2010)

  - ~200000$ (focused-ion-beam-microscope...)
  - months of analysis
  - break-in for one series of chips
  - physical extraction of TPM keys in a certain chip series in a few hours possible

  - http://www.blackhat.com/html/bh-dc-10/bh-dc-10-briefings.html#Tarnovsky

TSS



TCG Software Stack (TSS) Specification Version 1.2 Level 1

Errata A

Part1: Commands and Structures

March 7, 2007

Copyright © 2007 Trusted Computing Group, Incorporated.

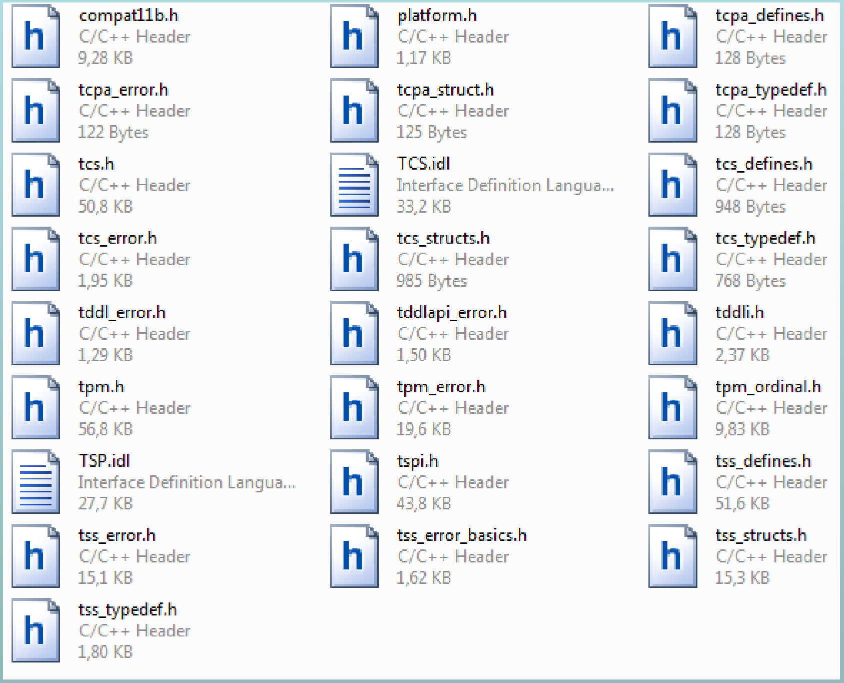
THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

No license, express or implied, by estoppel or otherwise, to any TCG or TCG member intellectual property rights is granted herein.

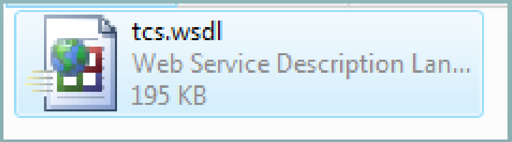
Except that a license is hereby granted by TCG to copy and reproduce this specification for internal use only.

Contact the Trusted Computing Group at http://trustedcomputinggroup.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.



compat11b.h	C/C++ Header	9,28 KB
platform.h	C/C++ Header	1,17 KB
tcpa_defines.h	C/C++ Header	128 Bytes
tcpa_error.h	C/C++ Header	122 Bytes
tcpa_struct.h	C/C++ Header	125 Bytes
tcpa_typedef.h	C/C++ Header	128 Bytes
tcs.h	C/C++ Header	50,8 KB
TCS.idl	Interface Definition Langua...	33,2 KB
tcs_defines.h	C/C++ Header	948 Bytes
tcs_error.h	C/C++ Header	1,95 KB
tcs_structs.h	C/C++ Header	985 Bytes
tcs_typedef.h	C/C++ Header	768 Bytes
tddl_error.h	C/C++ Header	1,29 KB
tddlapi_error.h	C/C++ Header	1,50 KB
tddli.h	C/C++ Header	2,37 KB
tpm.h	C/C++ Header	56,8 KB
tpm_error.h	C/C++ Header	19,6 KB
tpm_ordinal.h	C/C++ Header	9,83 KB
TSP.idl	Interface Definition Langua...	27,7 KB
tspi.h	C/C++ Header	43,8 KB
tss_defines.h	C/C++ Header	51,6 KB
tss_error.h	C/C++ Header	15,1 KB
tss_error_basics.h	C/C++ Header	1,62 KB
tss_structs.h	C/C++ Header	15,3 KB
tss_typedef.h	C/C++ Header	1,80 KB



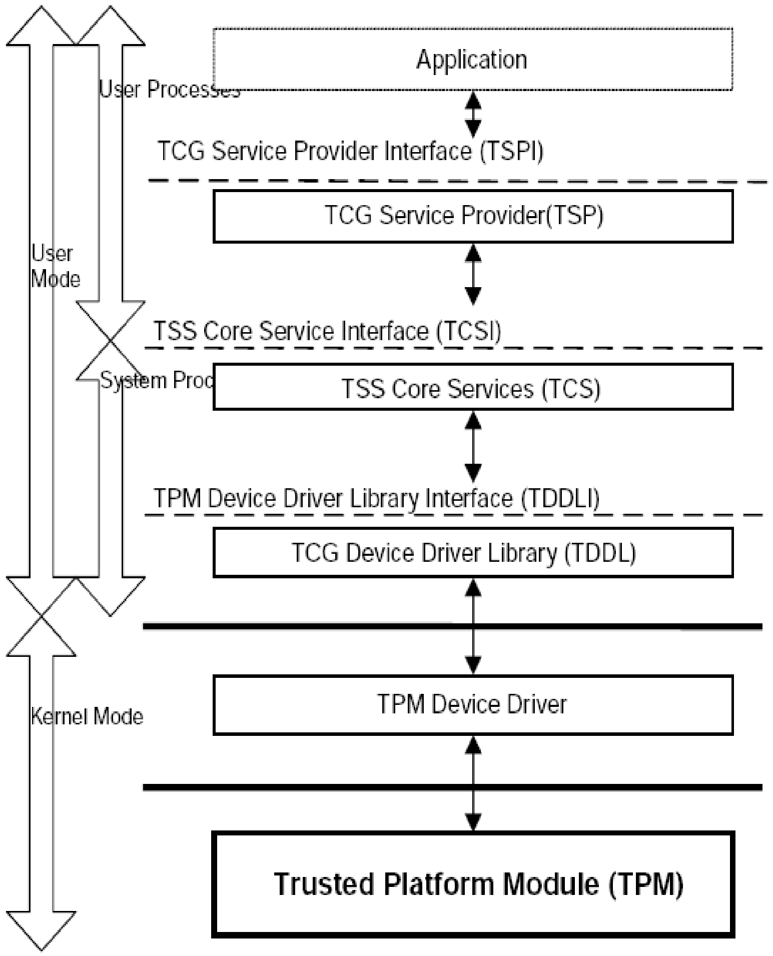
tcs.wsdl
Web Service Description Lan...
195 KB

TSS v1.2 Layers

- TSS is a stack of layers with defined interfaces

- TSS Service Provider (TSP)
  - standard API for applications
  - shared library
- TSS Core Services (TCS)
  - system singleton service daemon

- TSS Device Driver Library (TDDL)
  - Standard HW interface
- TPM device driver
  - kernel low-level portion


Application
User Processes
TCG Service Provider Interface (TSPI)
TCG Service Provider(TSP)
User Mode
TSS Core Service Interface (TCSI)
System Proc
TSS Core Services (TCS)
TPM Device Driver Library Interface (TDDLI)
TCG Device Driver Library (TDDL)
TPM Device Driver
Kernel Mode
Trusted Platform Module (TPM)

images © by TCG (TSS specification)

Linux TPM device

- In terminal / text console

- Check for initialisation message of TPM kernel driver

```
$ dmesg | grep tpm
[...]tpm_inf_pnp 00:0a: Found TPM with ID IFX0102
[...]tpm_inf_pnp 00:0a: TPM found: config base 0x4e, data base
0xcb0, chip version 0x000b, vendor id 0x15d1 (Infineon),
product id 0x000b (SLB 9635 TT 1.2)
```

- Check for existence of TPM device

```
$ ls -la /dev/tpm*
crw------- 1 tss tss 10, 224 Aug 26 14:31 /dev/tpm0
```

TSS Stacks

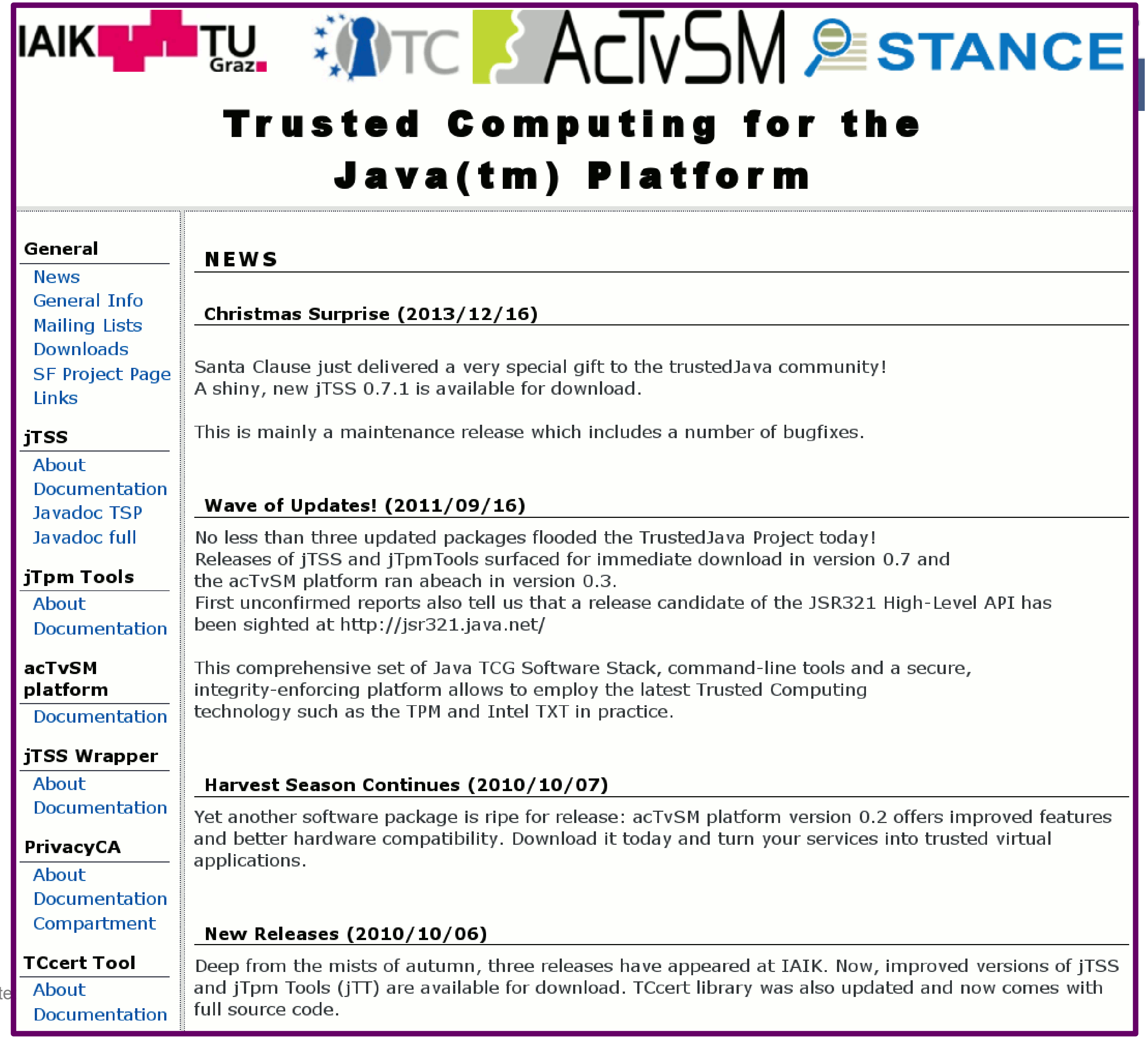
- 2 full open-source implementations of the TCG v1.2 Software Stack specifications
- TrouSerS / C
  - http://trousers.sourceforge.net/
- jTSS / Java
  - http://trustedjava.sourceforge.net/

C / TrouSerS

```
▪     UINT32                  pulRespDataLength;
      BYTE                    *pNumPCRs;
      UINT32                  subCap, subCapLength, numPcrs;
      TSS_HCONTEXT            hContext;
      TSS_HTPM                hTPM;
      TSS_RESULT              result;

▪     // Create Context
      result = Tspi_Context_Create( &hContext );
      if ( result != TSS_SUCCESS ) { ... }

▪     // Connect Context
      result = Tspi_Context_Connect( hContext, get_server( ... ) );
      if ( result != TSS_SUCCESS ) { ... }

▪     // Retrieve TPM object of context
      result = Tspi_Context_GetTpmObject( hContext, &hTPM );
      if ( result != TSS_SUCCESS ) { ... }

▪     subCap = TSS_TPMCAP_PROP_PCR;
      subCapLength = sizeof(UINT32);

▪     // Get number of PCRs
      result = Tspi_TPM_GetCapability( hTPM, TSS_TPMCAP_PROPERTY,
         subCapLength, (BYTE *)&subCap, &pulRespDataLength, &pNumPCRs );
      if ( result != TSS_SUCCESS ) { ... }

▪     // Print number of PCRs
      fprintf(stderr, "\tThere are %u PCRs supported by this TPM\n",
           *(UINT32 *)pNumPCRs );

▪     Tspi_Context_FreeMemory( hContext, NULL );
      Tspi_Context_Close( hContext );
```

Java / jTSS

```
▪   try {
▪     // Create Context
      TcIContext context = new TcTssContextFactory().newContextObject();
▪     // Connect Context
      context.connect();
▪     // Retrieve TPM object of context
      TcITpm tpm = context.getTpmObject();
▪     // Get number of PCRs
      long numberOfPCRS = tpm.getCapabilityUINT32(
          TcTssConstants.TSS_TPMCAP_PROPERTY,
          TcBlobData.newUNIT32(TcTssConstants.TSS_TPMCAP_PROP_PCR));
▪     // Print number of PCRs
      System.out.println("This TPM features " + numberOfPCRs + "PCRs.");

▪     // Free all open resources and close connection
      context.closeContext();

▪   } catch (TcTSSException e) {
     ...
    }
```



Trusted Computing for the Java(tm) Platform

General
- News
- General Info
- Mailing Lists
- Downloads
- SF Project Page
- Links

jTSS
- About
- Documentation
- Javadoc TSP
- Javadoc full

jTpm Tools
- About
- Documentation

acTvSM platform
- Documentation

jTSS Wrapper
- About
- Documentation

PrivacyCA
- About
- Documentation
- Compartment

TCcert Tool
- About
- Documentation

NEWS

Christmas Surprise (2013/12/16)

Santa Clause just delivered a very special gift to the trustedJava community!
A shiny, new jTSS 0.7.1 is available for download.

This is mainly a maintenance release which includes a number of bugfixes.

Wave of Updates! (2011/09/16)

No less than three updated packages flooded the TrustedJava Project today!
Releases of jTSS and jTpmTools surfaced for immediate download in version 0.7 and the acTvSM platform ran abeach in version 0.3.
First unconfirmed reports also tell us that a release candidate of the JSR321 High-Level API has been sighted at http://jsr321.java.net/

This comprehensive set of Java TCG Software Stack, command-line tools and a secure, integrity-enforcing platform allows to employ the latest Trusted Computing technology such as the TPM and Intel TXT in practice.

Harvest Season Continues (2010/10/07)

Yet another software package is ripe for release: acTvSM platform version 0.2 offers improved features and better hardware compatibility. Download it today and turn your services into trusted virtual applications.

New Releases (2010/10/06)

Deep from the mists of autumn, three releases have appeared at IAIK. Now, improved versions of jTSS and jTpm Tools (jTT) are available for download. TCcert library was also updated and now comes with full source code.

[http://trustedjava.sourceforge.net/]

TXT

Intel Trusted Execution Technology (TXT)

- 3 components on platform in (hard-wired) co-operation

- CPU
  - Special opcode to allow *late launch* into measured system state
  - Transparent virtualization of unmodified OS

- Chipset
  - Advanced I/O control, assign specific device to specific virtualization compartment (restrict DMA etc.)
  - Scrub memory with all zeros on reboot

- TPM
  - Storage for measurement chain state
  - Storage for launch control policy (LCP) of allowed binary

PCRs After System Boot

```
00:  e5 58 e7 ec 7f 82 99 c7 4a 24 63 6e 24 82 56 de f1 25 19 de
01:  a7 46 df 87 49 64 75 fe 5c b2 14 47 f3 8e 66 46 04 57 67 5b
02:  05 a8 c0 53 ed 4b 3d 48 ea 43 f4 64 f4 9a 96 b6 0b 05 0a 72
03:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
04:  4c 1c c1 30 4e 70 9b 16 fc f1 06 aa 44 9b 8b bb 90 bc 34 ac
05:  c3 1a 3d e8 6d 3f 46 bf 1a 01 03 39 c9 d6 3c 55 fa 42 c1 23
06:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
07:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
08:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
09:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
11:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
12:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
13:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
14:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
15:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
17:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
18:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
19:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
20:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
21:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
22:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
23:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

PCRs After TXT Late Launch

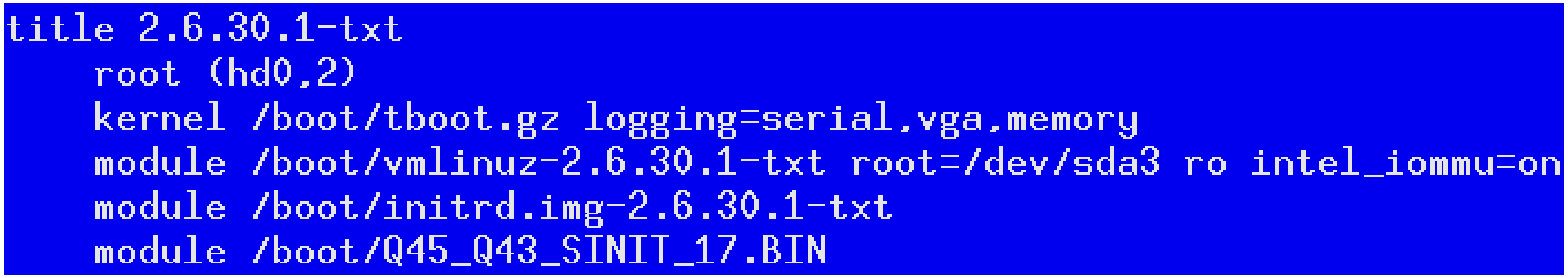
```
~~00:  e5 58 e7 ec 7f 82 99 c7 4a 24 63 6e 24 82 56 de f1 25 19 de~~
~~01:  a7 46 df 87 49 64 75 fe 5c b2 14 47 f3 8e 66 46 04 57 67 5b~~
~~02:  05 a8 c0 53 ed 4b 3d 48 ea 43 f4 64 f4 9a 96 b6 0b 05 0a 72~~
~~03:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75~~
~~04:  4c 1c c1 30 4e 70 9b 16 fc f1 06 aa 44 9b 8b bb 90 bc 34 ac~~
~~05:  c3 1a 3d e8 6d 3f 46 bf 1a 01 03 39 c9 d6 3c 55 fa 42 c1 23~~
~~06:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75~~
~~07:  3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75~~
08:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
09:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
11:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
12:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
13:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
14:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
15:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

17:  13 cc db 74 65 10 b6 30 5c 1c c1 1d 10 95 ec a0 1f 06 36 d7
18:  e8 63 a1 f4 69 74 67 72 95 18 5f 6b ab b9 59 7e 90 30 6e 9b

19:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
20:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
21:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
22:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
23:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

TBoot

- “An open source, pre-kernel/VMM module that uses Intel TXT to perform a measured and verified launch of an OS kernel / VMM“.

  http://sourceforge.net/projects/tboot/

- Boot order:
  GRUB → TBoot → SINIT → TBoot → Linux kernel



```
title 2.6.30.1-txt
    root (hd0,2)
    kernel /boot/tboot.gz logging=serial,vga,memory
    module /boot/vmlinuz-2.6.30.1-txt root=/dev/sda3 ro intel_iommu=on
    module /boot/initrd.img-2.6.30.1-txt
    module /boot/Q45_Q43_SINIT_17.BIN
```

TXT Security

- System Management Mode (SMM)
  - Has Ring -2 priviledges – more priviledged than hypervisor Ring -1 code
  - → After DRTM sequence hypervisor can be modified by malicious SMM code

    http://invisiblethingslab.com/resources/bh09dc/Attacking%20Intel%20TXT%20-%20paper.pdf

- SINIT code is not Open-Source
- Reverse engineering prohibited by license
  - Exploit found (Dec 2009)
    http://invisiblethingslab.com/resources/misc09/Another%20TXT%20Attack.pdf
  - Memory verification, 32bit vs. 64bit error

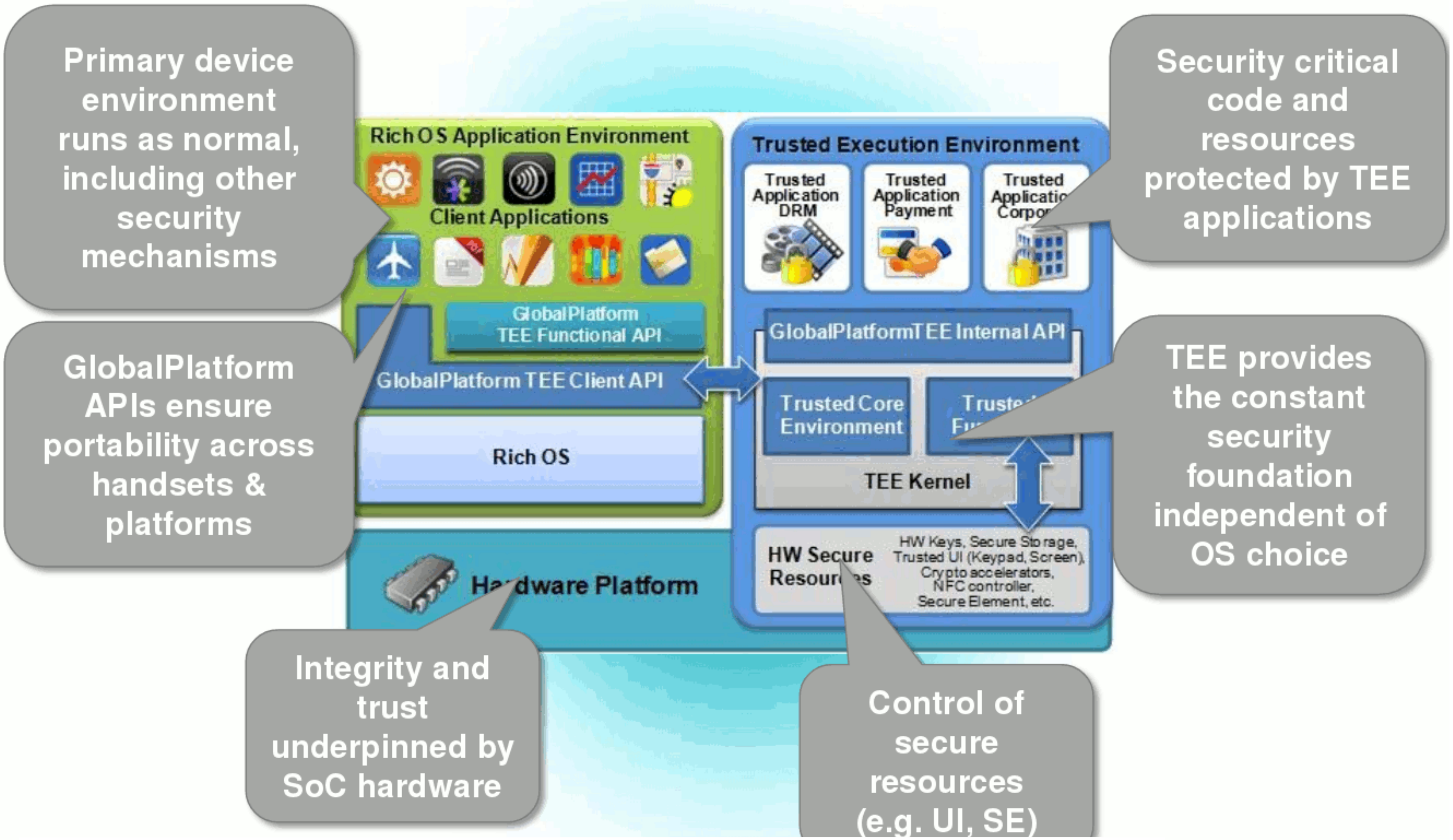
Trusted Execution Envivonment (TEE)
ARM TrustZone (TZ)

Mobile Trusted Module

- Idea of a MTM
  - similar to TPM on PC platform
  - provides trusted resources
  - subset of “full” TPM commands
  - additional MTM specific features not found with TPM

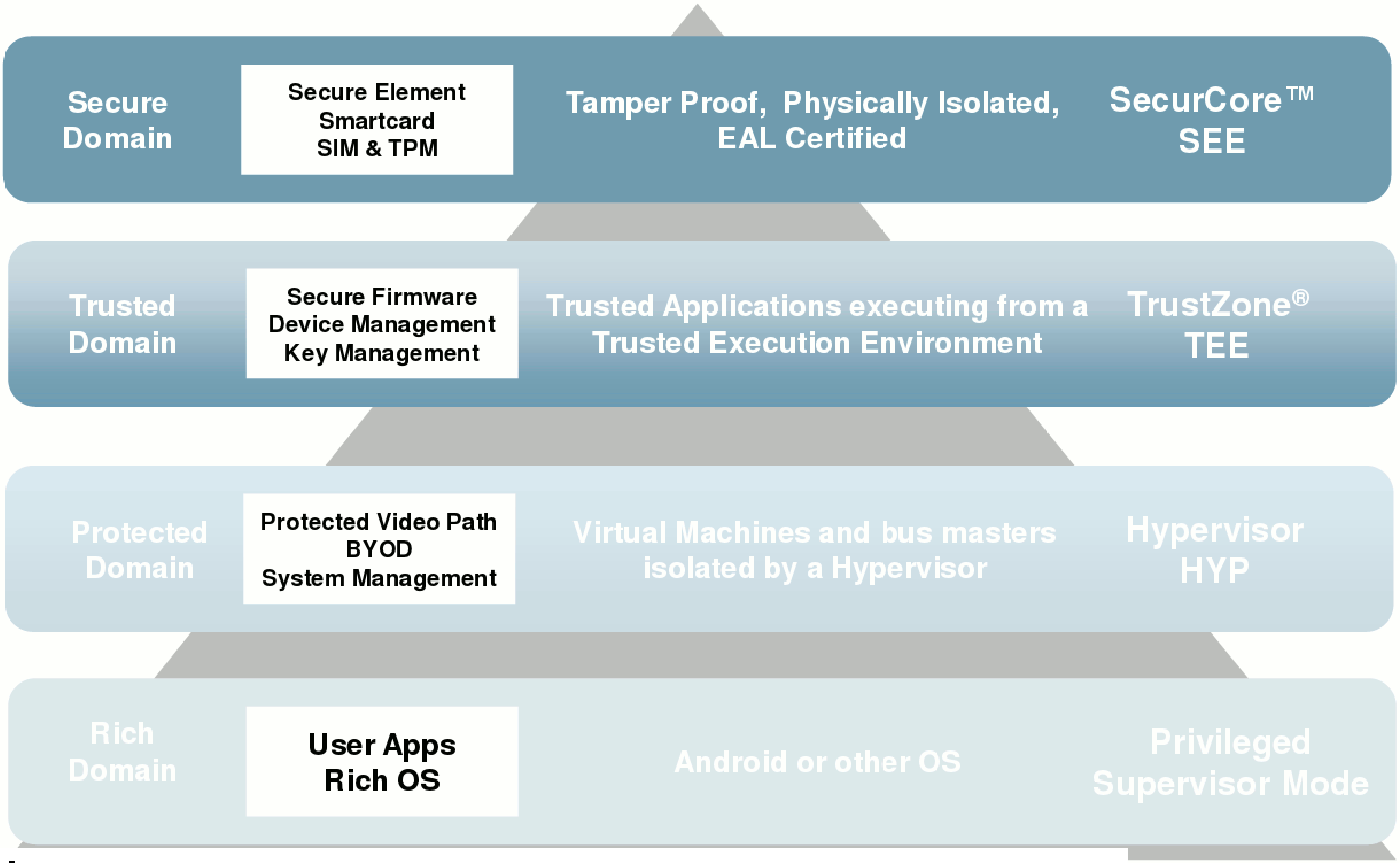
  - ...additional hardware :-(
  - ...or software implementation? :-)

GlobalPlatform TEE


Primary device environment runs as normal, including other security mechanisms
GlobalPlatform APIs ensure portability across handsets & platforms
Integrity and trust underpinned by SoC hardware
Control of secure resources (e.g. UI, SE)
TEE provides the constant security foundation independent of OS choice
Security critical code and resources protected by TEE applications
Rich OS Application Environment
Client Applications
GlobalPlatform TEE Functional API
GlobalPlatform TEE Client API
Rich OS
Hardware Platform
Trusted Execution Environment
Trusted Application DRM
Trusted Application Payment
Trusted Application Corpor
GlobalPlatform TEE Internal API
Trusted Core Environment
Truste Fu
TEE Kernel
HW Secure Resources
HW Keys, Secure Storage, Trusted UI (Keypad, Screen), Crypto accelerators, NFC controller, Secure Element, etc.

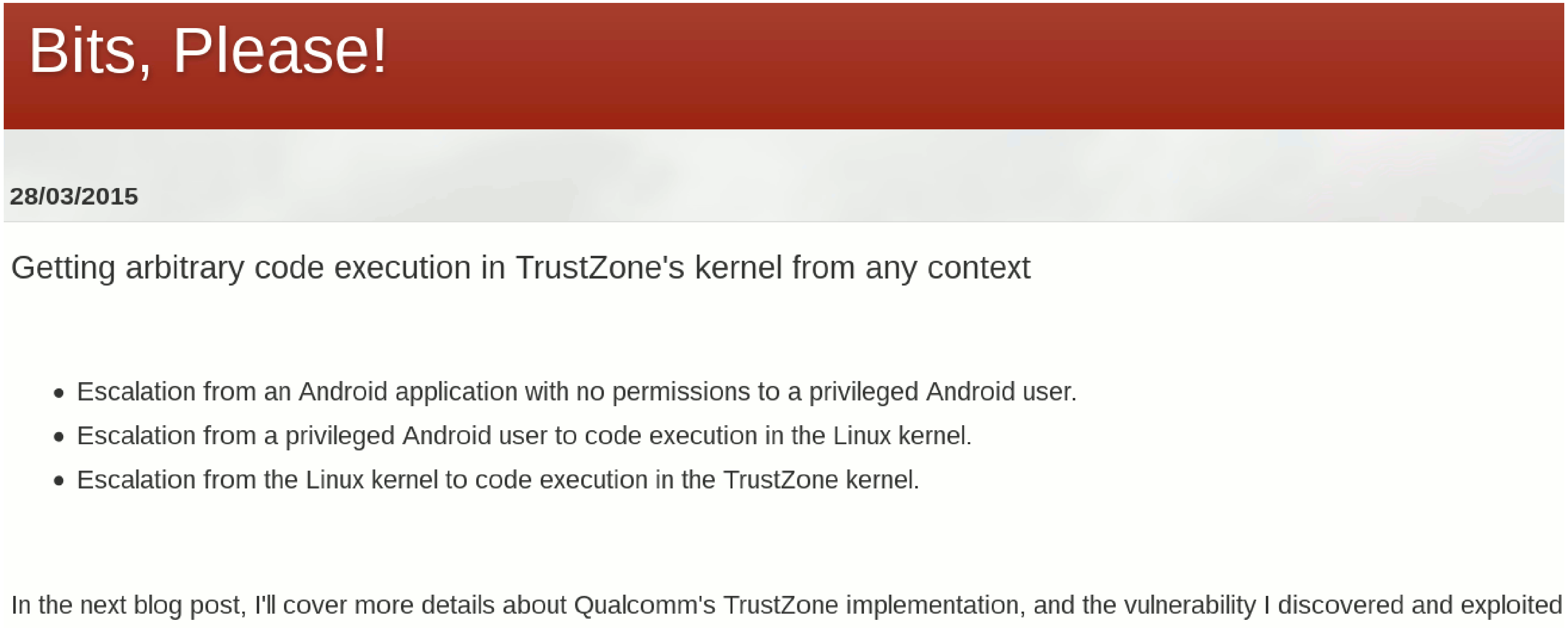
[R. Coombs, S. Moore; GlobalPlatform TEE & ARM Trustzone technology: Building security into your platform, 2013]

4 Compartment Model – Hierarchy of Trust



Secure Domain	Secure Element Smartcard SIM & TPM	Tamper Proof, Physically Isolated, EAL Certified	SecurCore™ SEE
Trusted Domain	Secure Firmware Device Management Key Management	Trusted Applications executing from a Trusted Execution Environment	TrustZone® TEE
Protected Domain	Protected Video Path BYOD System Management	Virtual Machines and bus masters isolated by a Hypervisor	Hypervisor HYP
Rich Domain	User Apps Rich OS	Android or other OS	Privileged Supervisor Mode

[R. Coombs, S. Moore; GlobalPlatform TEE & ARM Trustzone technology: Building security into your platform, 2013]


Bits, Please!

28/03/2015

Getting arbitrary code execution in TrustZone's kernel from any context

- Escalation from an Android application with no permissions to a privileged Android user.
- Escalation from a privileged Android user to code execution in the Linux kernel.
- Escalation from the Linux kernel to code execution in the TrustZone kernel.

In the next blog post, I'll cover more details about Qualcomm's TrustZone implementation, and the vulnerability I discovered and exploited

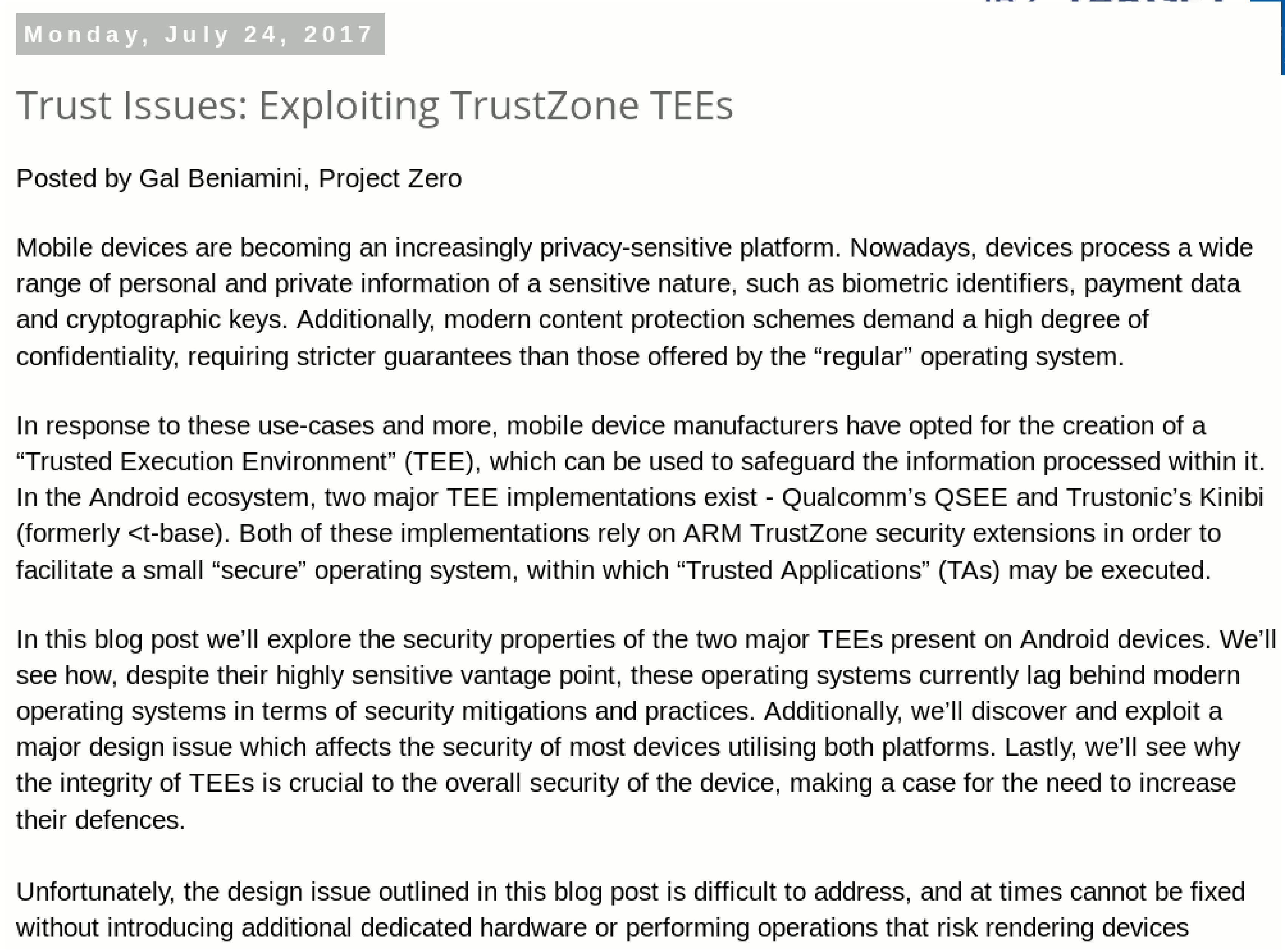
[https://bits-please.blogspot.com/2015/03/getting-arbitrary-code-execution-in.html]

KNOXout: Samsung Knox vulnerabilities give hackers 'full control' of devices

Attack allows hackers to execute code on the Galaxy S6 and Note 5

SECURITY RESEARCHERS have uncovered three vulnerabilities in Samsung's Knox system that could allow hackers to gain "full control" of Galaxy S6 and Note 5 smartphones.

[https://www.theinquirer.net/inquirer/news/2473037/knoxout-samsung-knox-vulnerabilities-give-hackers-full-control-of-devices]


Monday, July 24, 2017

Trust Issues: Exploiting TrustZone TEEs

Posted by Gal Beniamini, Project Zero

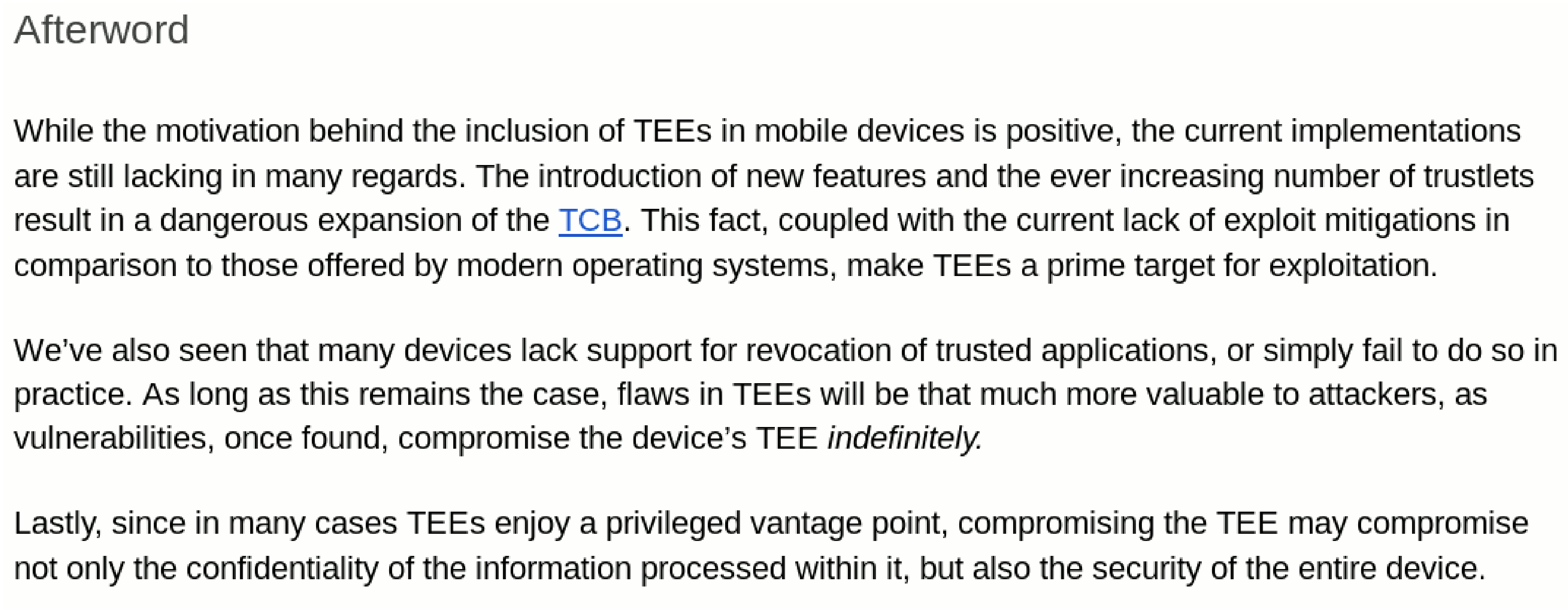
Mobile devices are becoming an increasingly privacy-sensitive platform. Nowadays, devices process a wide range of personal and private information of a sensitive nature, such as biometric identifiers, payment data and cryptographic keys. Additionally, modern content protection schemes demand a high degree of confidentiality, requiring stricter guarantees than those offered by the "regular" operating system.

In response to these use-cases and more, mobile device manufacturers have opted for the creation of a "Trusted Execution Environment" (TEE), which can be used to safeguard the information processed within it. In the Android ecosystem, two major TEE implementations exist - Qualcomm's QSEE and Trustonic's Kinibi (formerly <t-base). Both of these implementations rely on ARM TrustZone security extensions in order to facilitate a small "secure" operating system, within which "Trusted Applications" (TAs) may be executed.

In this blog post we'll explore the security properties of the two major TEEs present on Android devices. We'll see how, despite their highly sensitive vantage point, these operating systems currently lag behind modern operating systems in terms of security mitigations and practices. Additionally, we'll discover and exploit a major design issue which affects the security of most devices utilising both platforms. Lastly, we'll see why the integrity of TEEs is crucial to the overall security of the device, making a case for the need to increase their defences.

Unfortunately, the design issue outlined in this blog post is difficult to address, and at times cannot be fixed without introducing additional dedicated hardware or performing operations that risk rendering devices

[https://googleprojectzero.blogspot.com/2017/07/trust-issues-exploiting-trustzone-tees.html]



Afterword

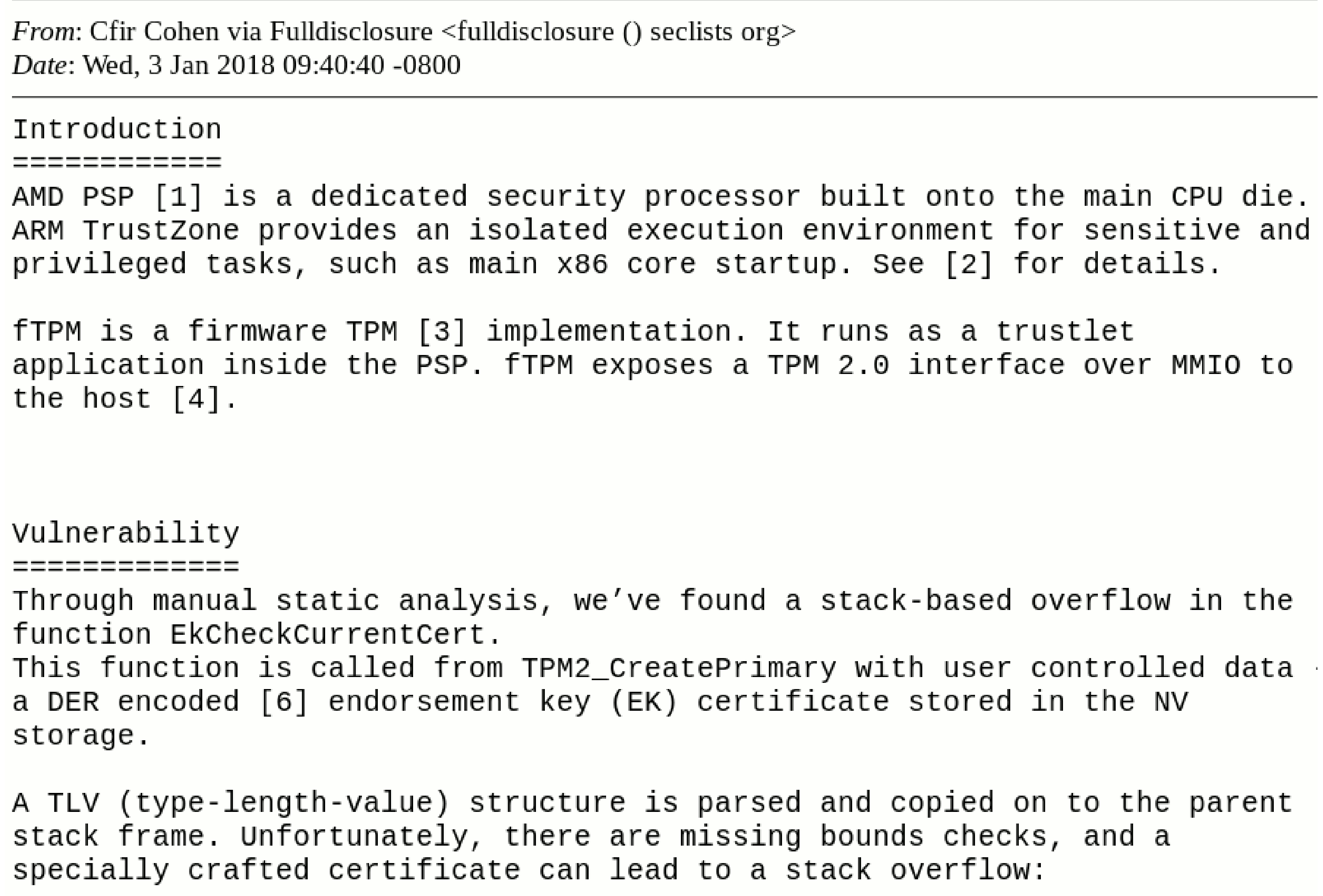
While the motivation behind the inclusion of TEEs in mobile devices is positive, the current implementations are still lacking in many regards. The introduction of new features and the ever increasing number of trustlets result in a dangerous expansion of the TCB. This fact, coupled with the current lack of exploit mitigations in comparison to those offered by modern operating systems, make TEEs a prime target for exploitation.

We've also seen that many devices lack support for revocation of trusted applications, or simply fail to do so in practice. As long as this remains the case, flaws in TEEs will be that much more valuable to attackers, as vulnerabilities, once found, compromise the device's TEE *indefinitely.*

Lastly, since in many cases TEEs enjoy a privileged vantage point, compromising the TEE may compromise not only the confidentiality of the information processed within it, but also the security of the entire device.

[https://googleprojectzero.blogspot.com/2017/07/trust-issues-exploiting-trustzone-tees.html]

AMD-PSP: fTPM Remote Code Execution via crafted EK certificate

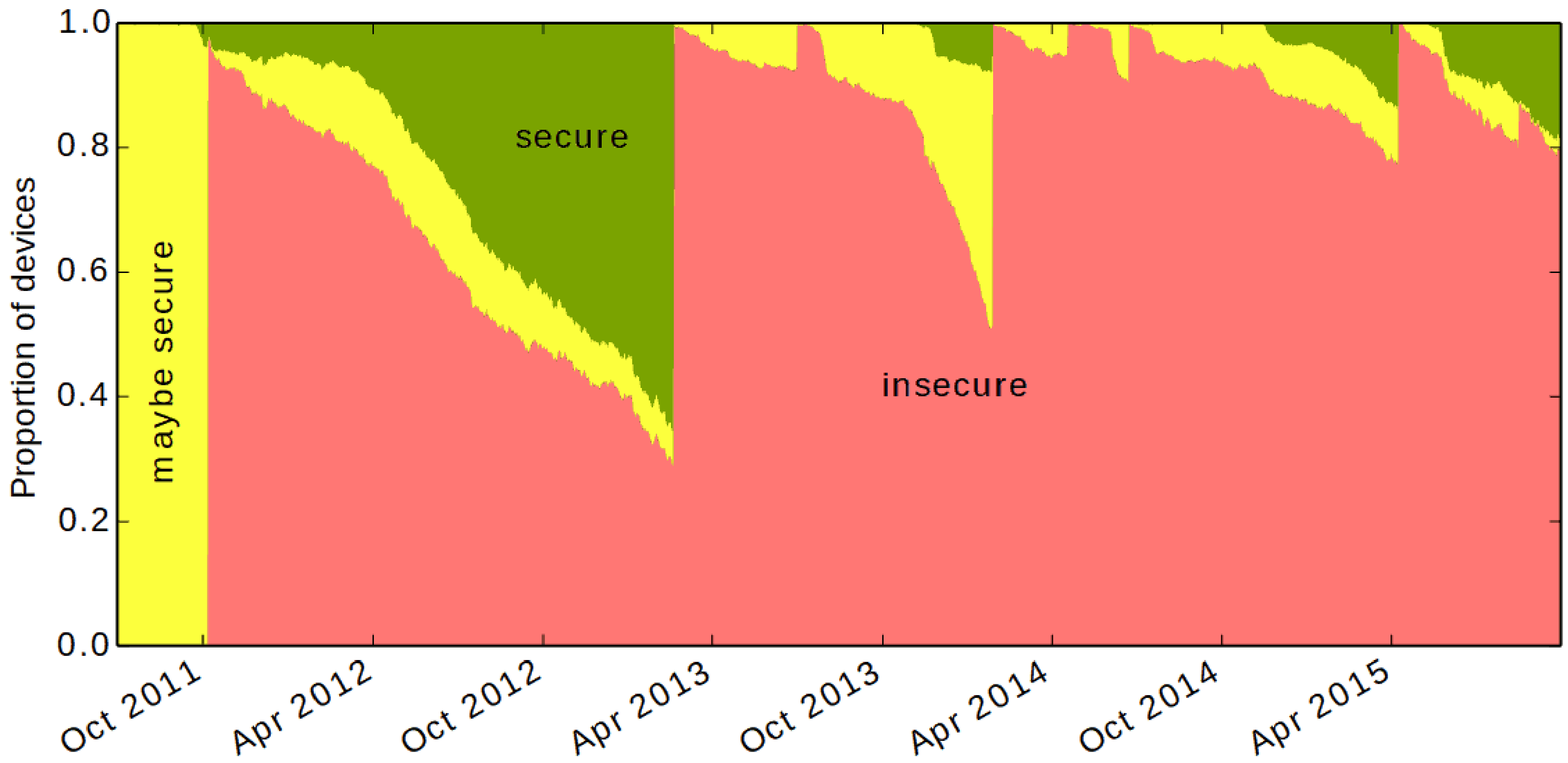


From: Cfir Cohen via Fulldisclosure <fulldisclosure () seclists org>
Date: Wed, 3 Jan 2018 09:40:40 -0800

```
Introduction
============
AMD PSP [1] is a dedicated security processor built onto the main CPU die.
ARM TrustZone provides an isolated execution environment for sensitive and
privileged tasks, such as main x86 core startup. See [2] for details.

fTPM is a firmware TPM [3] implementation. It runs as a trustlet
application inside the PSP. fTPM exposes a TPM 2.0 interface over MMIO to
the host [4].



Vulnerability
=============
Through manual static analysis, we’ve found a stack-based overflow in the
function EkCheckCurrentCert.
This function is called from TPM2_CreatePrimary with user controlled data
a DER encoded [6] endorsement key (EK) certificate stored in the NV
storage.

A TLV (type-length-value) structure is parsed and copied on to the parent
stack frame. Unfortunately, there are missing bounds checks, and a
specially crafted certificate can lead to a stack overflow:
```

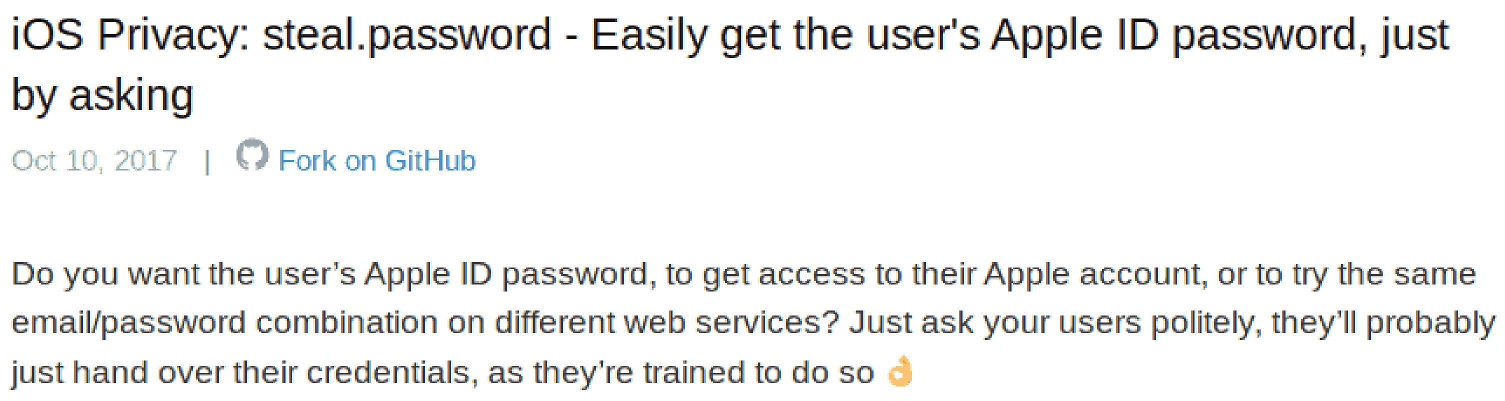
[http://seclists.org/fulldisclosure/2018/Jan/12]

Proportion of devices running vulnerable versions of Android



[https://androidvulnerabilities.org/]

Who Wants My Password?



iOS Privacy: steal.password - Easily get the user's Apple ID password, just by asking

Oct 10, 2017 | Fork on GitHub

Do you want the user's Apple ID password, to get access to their Apple account, or to try the same email/password combination on different web services? Just ask your users politely, they'll probably just hand over their credentials, as they're trained to do so 👌

[https://krausefx.com/blog/ios-privacy-stealpassword-easily-get-the-users-apple-id-password-just-by-asking]

Who Really Wants My Password?

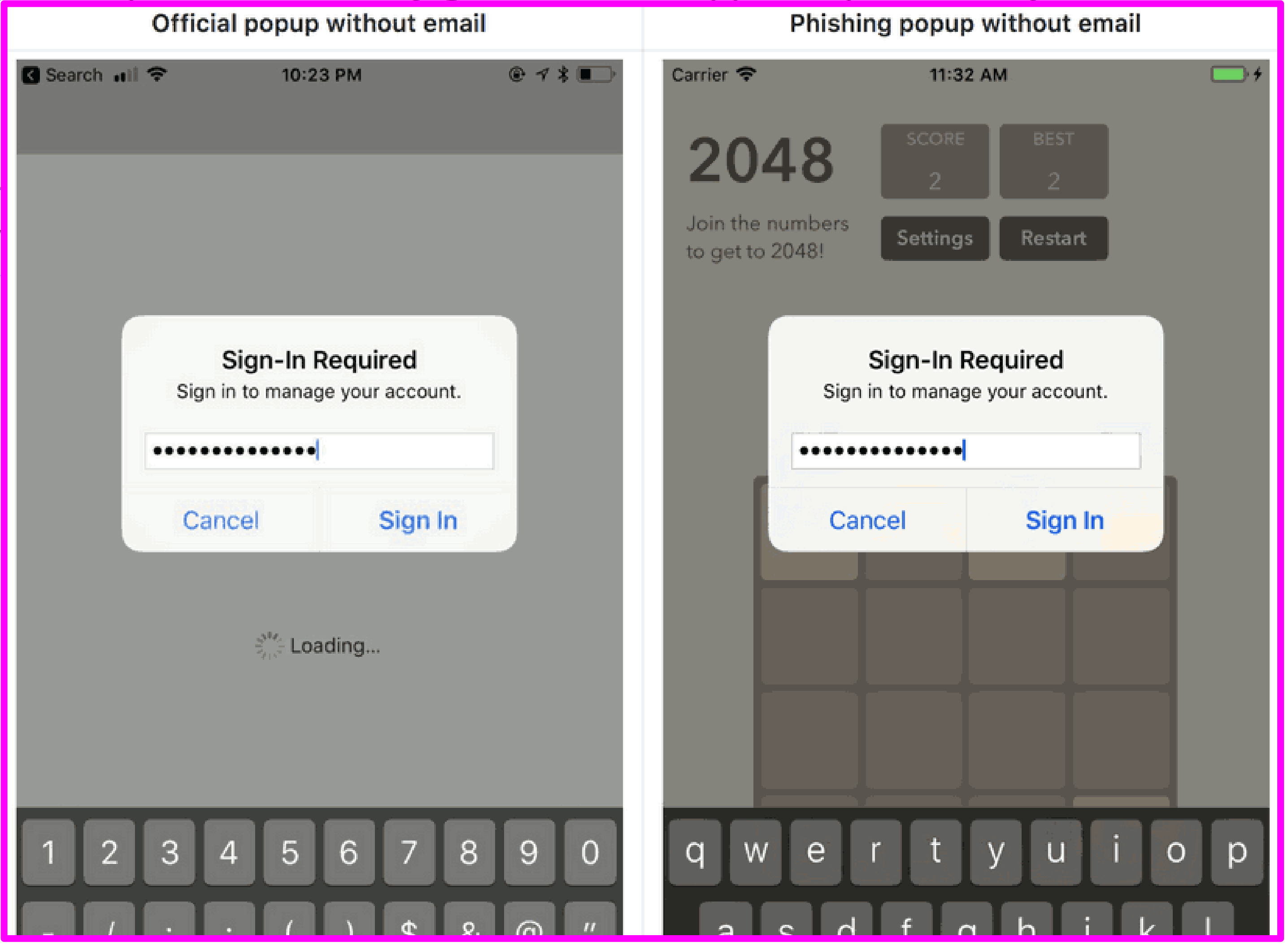
JRZ TARGET
JOSEF RESSEL CENTER
FOR UNIFIED THREAT INTELLIGENCE ON TARGETED ATTACKS

/fh/// st.pölten

iOS Privacy: steal.password - Easily get the user's Apple ID password, just by asking

Oct 10, 2017 |

Do you want t
email/passwor
just hand over


Official popup without email
Phishing popup without email
Sign-In Required
Sign in to manage your account.
Cancel
Sign In
Loading...
2048
SCORE 2
BEST 2
Join the numbers to get to 2048!
Settings
Restart

[https://krausefx.com/blog/ios-privacy-stealpassword-easily-get-the-users-apple-id-password-just-by-asking]

SGX

Software Guard Extensions (SGX)

- A malicious app exploits OS and gains privileges
- Now
  - applications not protected from privileged code (apps always have to trust kernel code!)
- Goal
  - app gains ability to defend its own secrets
  - small attack surface
  - malware that subverts “everything” cannot steal app secrets

Software Guard Extensions (SGX)

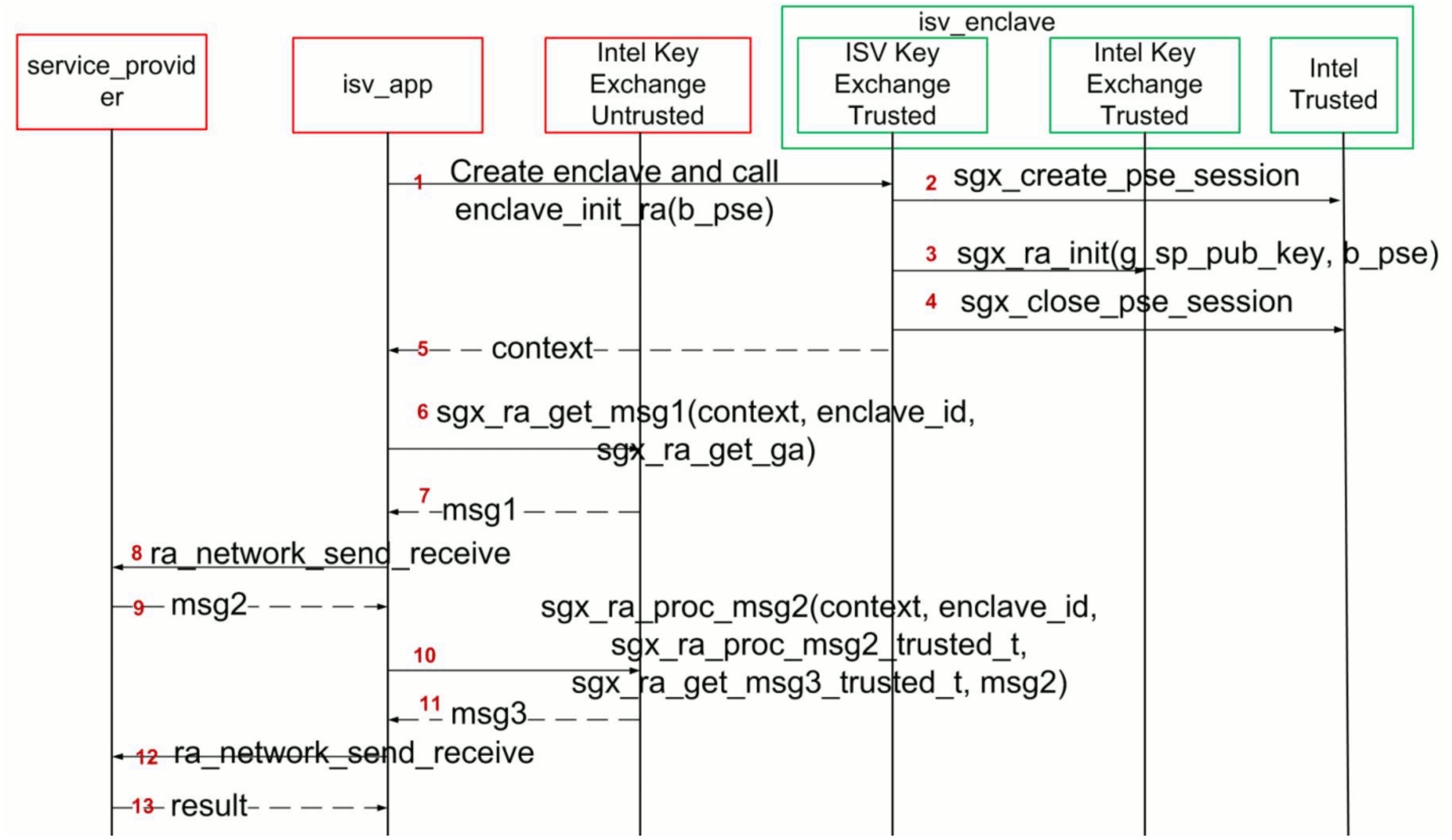
- Intel Skylake processors (late 2015) and newer
- Enclaves
  - encrypted code+data
  - integrity
  - confidentiality
  - controlled entry/exit points

Software Guard Extensions (SGX)

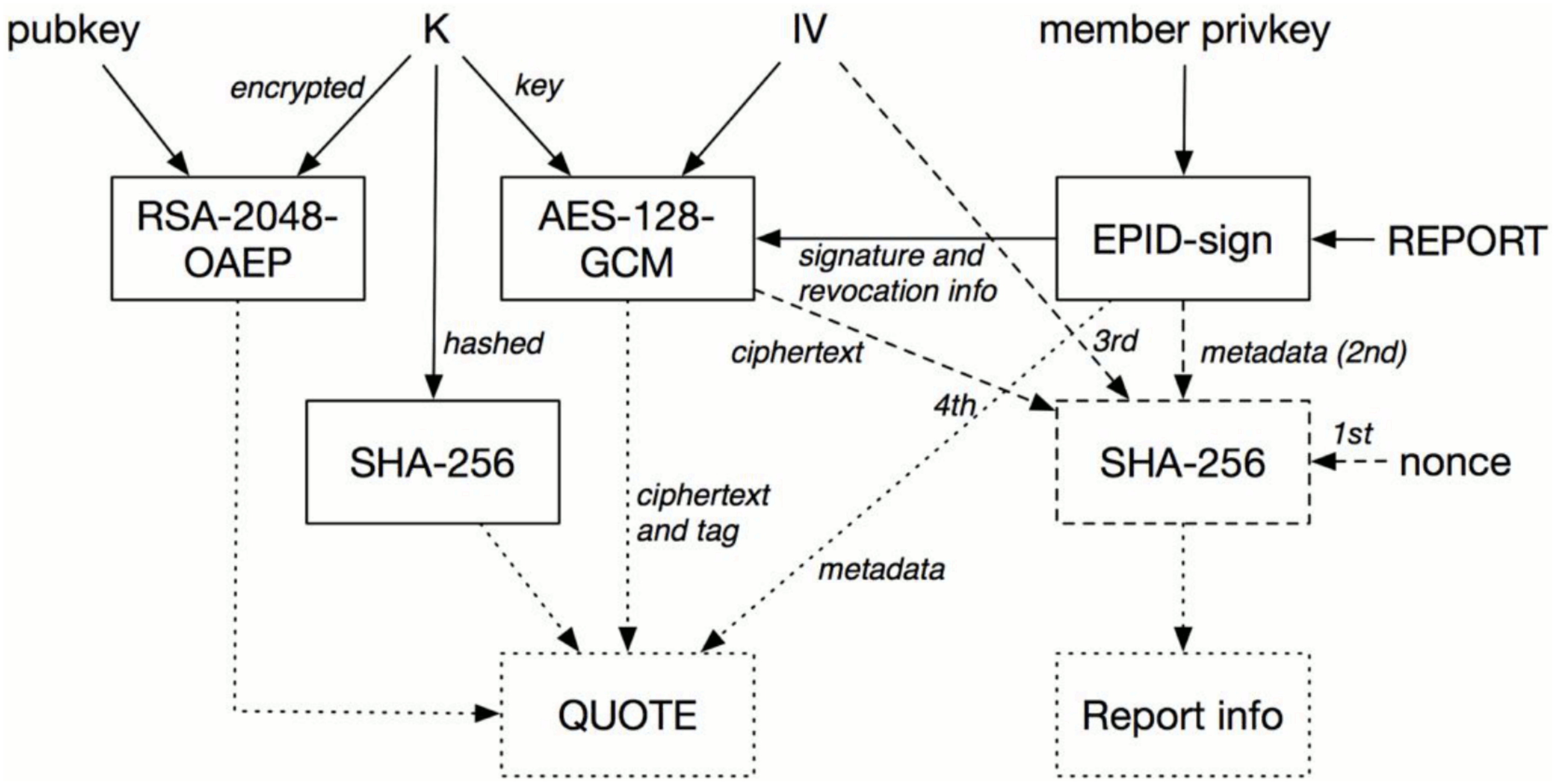
- Problem: Attestation of Enclave

- Attestation Enclave ↔ Enclave on same platform
- Attestation Enclave ↔ Outside

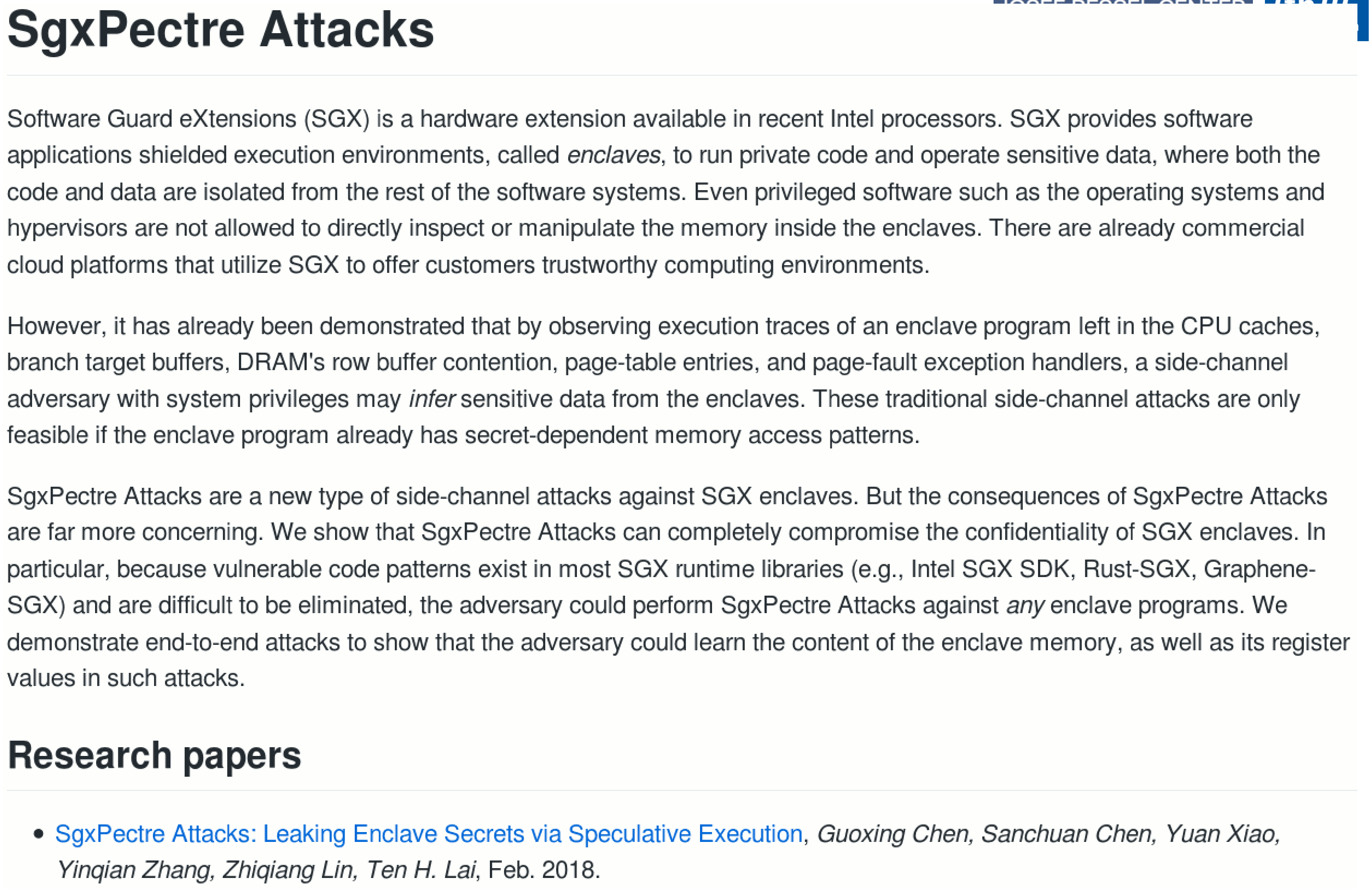
- For a detailed tutorial on SGX, see e.g.
  https://software.intel.com/sites/default/files/332680-002.pdf


isv_enclave
service_provider
isv_app
Intel Key Exchange Untrusted
ISV Key Exchange Trusted
Intel Key Exchange Trusted
Intel Trusted
1 Create enclave and call enclave_init_ra(b_pse)
2 sgx_create_pse_session
3 sgx_ra_init(g_sp_pub_key, b_pse)
4 sgx_close_pse_session
5 context
6 sgx_ra_get_msg1(context, enclave_id, sgx_ra_get_ga)
7 msg1
8 ra_network_send_receive
9 msg2
10 sgx_ra_proc_msg2(context, enclave_id, sgx_ra_proc_msg2_trusted_t, sgx_ra_get_msg3_trusted_t, msg2)
11 msg3
12 ra_network_send_receive
13 result

[originally Intel; in presentation
JP Aumasson, L. Merino; SGX Secure Enclaves in Practice: Security and Crypto Review; Black Hat 2016]



[JP Aumasson, L. Merino; SGX Secure Enclaves in Practice: Security and Crypto Review; Black Hat 2016]



SgxPectre Attacks

Software Guard eXtensions (SGX) is a hardware extension available in recent Intel processors. SGX provides software applications shielded execution environments, called *enclaves*, to run private code and operate sensitive data, where both the code and data are isolated from the rest of the software systems. Even privileged software such as the operating systems and hypervisors are not allowed to directly inspect or manipulate the memory inside the enclaves. There are already commercial cloud platforms that utilize SGX to offer customers trustworthy computing environments.

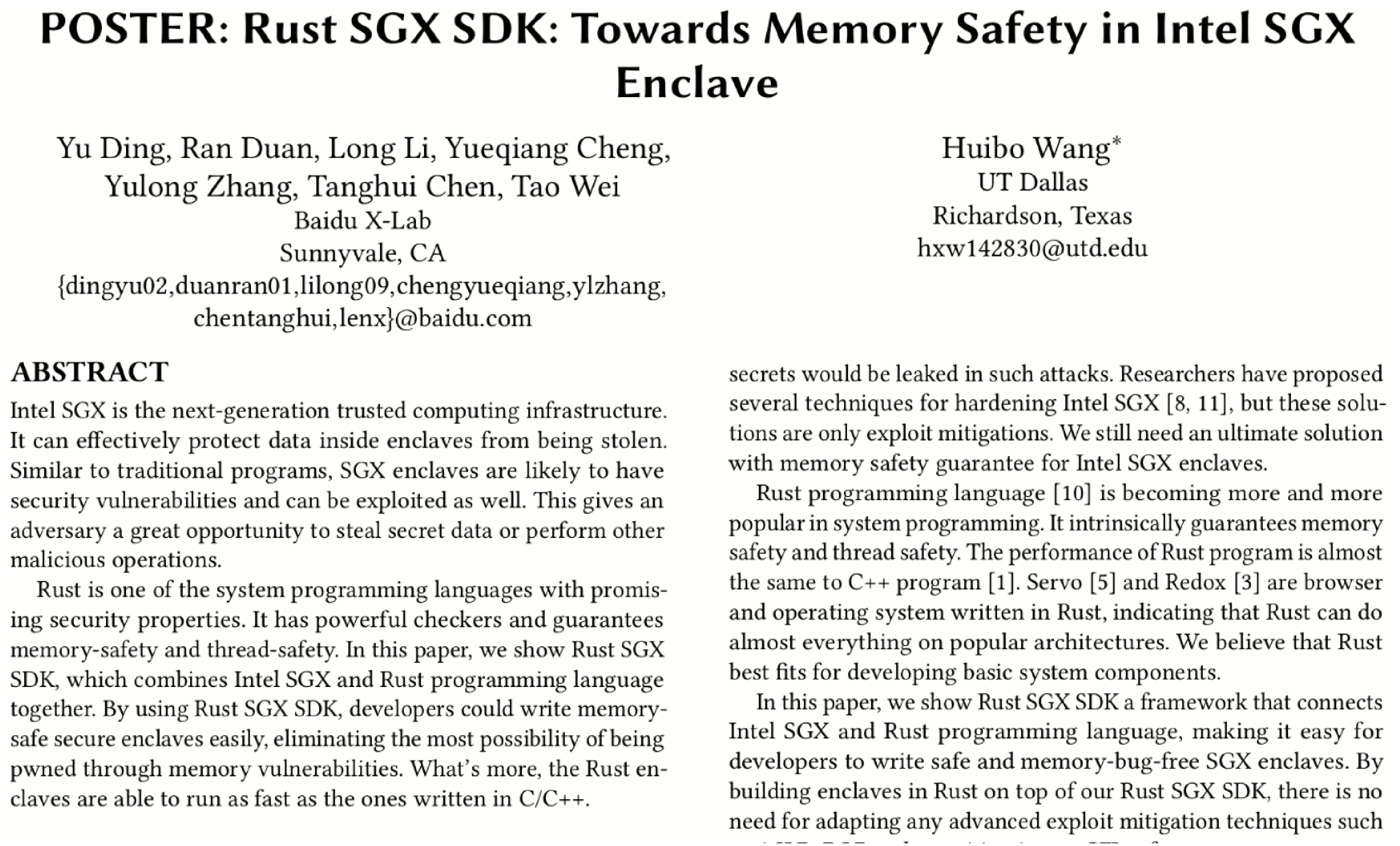
However, it has already been demonstrated that by observing execution traces of an enclave program left in the CPU caches, branch target buffers, DRAM's row buffer contention, page-table entries, and page-fault exception handlers, a side-channel adversary with system privileges may *infer* sensitive data from the enclaves. These traditional side-channel attacks are only feasible if the enclave program already has secret-dependent memory access patterns.

SgxPectre Attacks are a new type of side-channel attacks against SGX enclaves. But the consequences of SgxPectre Attacks are far more concerning. We show that SgxPectre Attacks can completely compromise the confidentiality of SGX enclaves. In particular, because vulnerable code patterns exist in most SGX runtime libraries (e.g., Intel SGX SDK, Rust-SGX, Graphene-SGX) and are difficult to be eliminated, the adversary could perform SgxPectre Attacks against *any* enclave programs. We demonstrate end-to-end attacks to show that the adversary could learn the content of the enclave memory, as well as its register values in such attacks.

Research papers

- SgxPectre Attacks: Leaking Enclave Secrets via Speculative Execution, *Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, Ten H. Lai*, Feb. 2018.

Works in Progress



POSTER: Rust SGX SDK: Towards Memory Safety in Intel SGX Enclave

Yu Ding, Ran Duan, Long Li, Yueqiang Cheng,
Yulong Zhang, Tanghui Chen, Tao Wei
Baidu X-Lab
Sunnyvale, CA
{dingyu02,duanran01,lilong09,chengyueqiang,ylzhang,
chentanghui,lenx}@baidu.com

Huibo Wang*
UT Dallas
Richardson, Texas
hxw142830@utd.edu

ABSTRACT

Intel SGX is the next-generation trusted computing infrastructure. It can effectively protect data inside enclaves from being stolen. Similar to traditional programs, SGX enclaves are likely to have security vulnerabilities and can be exploited as well. This gives an adversary a great opportunity to steal secret data or perform other malicious operations.

Rust is one of the system programming languages with promising security properties. It has powerful checkers and guarantees memory-safety and thread-safety. In this paper, we show Rust SGX SDK, which combines Intel SGX and Rust programming language together. By using Rust SGX SDK, developers could write memory-safe secure enclaves easily, eliminating the most possibility of being pwned through memory vulnerabilities. What's more, the Rust enclaves are able to run as fast as the ones written in C/C++.

secrets would be leaked in such attacks. Researchers have proposed several techniques for hardening Intel SGX [8, 11], but these solutions are only exploit mitigations. We still need an ultimate solution with memory safety guarantee for Intel SGX enclaves.

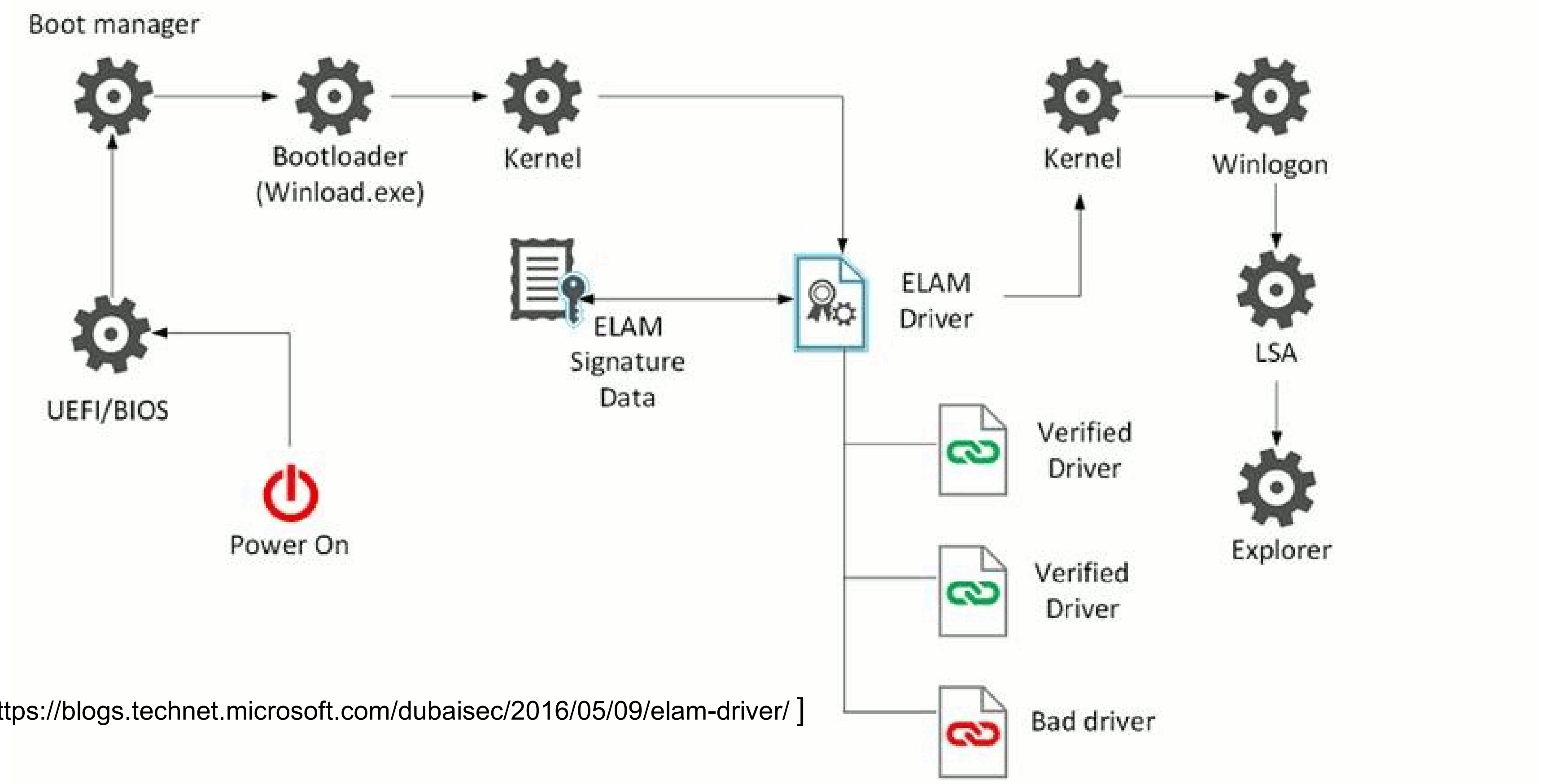
Rust programming language [10] is becoming more and more popular in system programming. It intrinsically guarantees memory safety and thread safety. The performance of Rust program is almost the same to C++ program [1]. Servo [5] and Redox [3] are browser and operating system written in Rust, indicating that Rust can do almost everything on popular architectures. We believe that Rust best fits for developing basic system components.

In this paper, we show Rust SGX SDK a framework that connects Intel SGX and Rust programming language, making it easy for developers to write safe and memory-bug-free SGX enclaves. By building enclaves in Rust on top of our Rust SGX SDK, there is no need for adapting any advanced exploit mitigation techniques such

[https://github.com/baidu/rust-sgx-sdk]

UEFI Secure Boot and Windows Chain of Trust

ELAM – **Early Launch Anti-Malware** is a piece of code that is loaded in the pre-boot environment and is responsible for verification of other drivers before they are loaded into memory. ELAM driver is being called by the Kernel and it reports the status of the driver images: if they are safe or unsafe to load. ELAM driver has been introduced in Windows 8 and is not dependent on UEFI nor BIOS implementation.

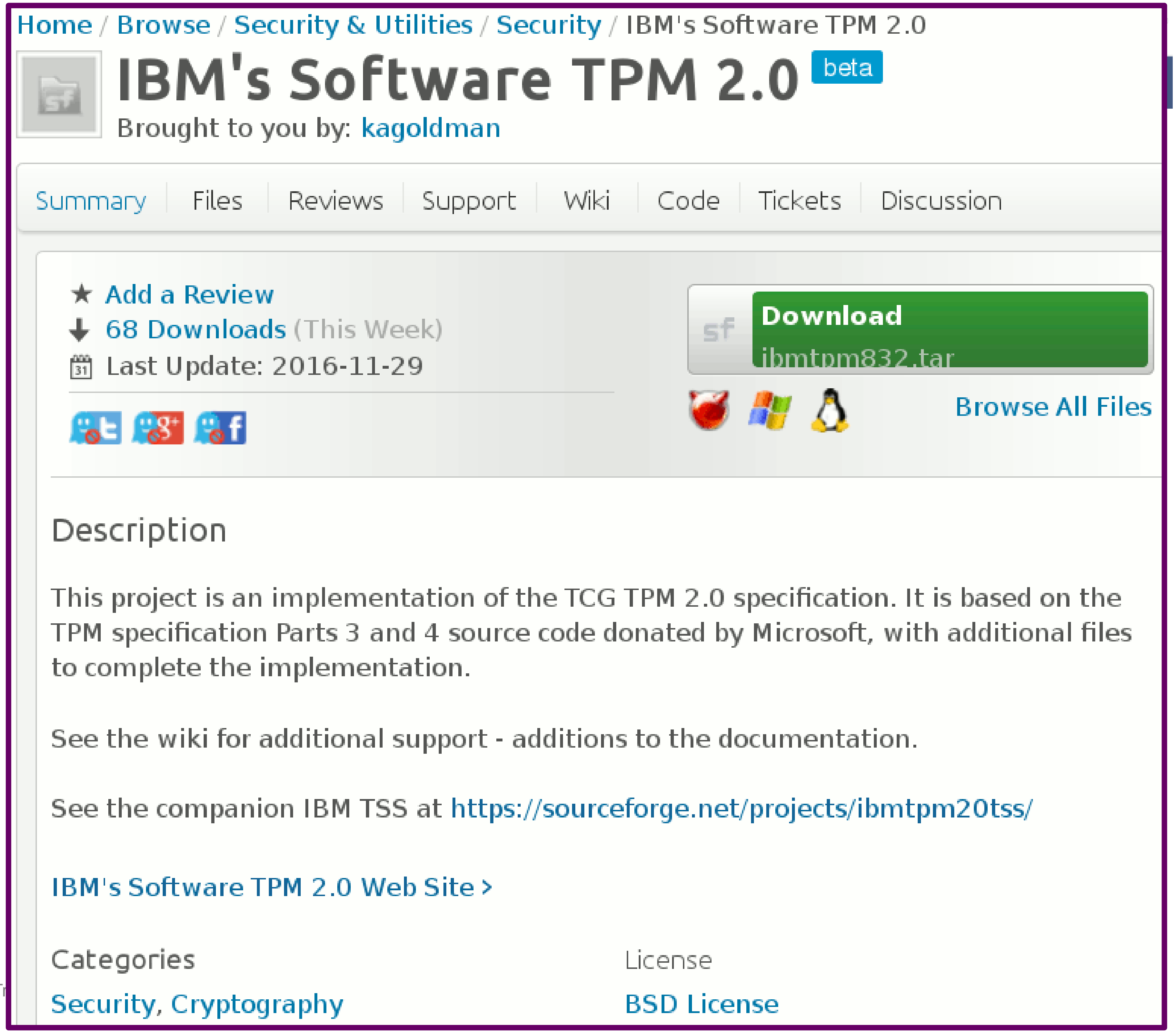


[https://blogs.technet.microsoft.com/dubaisec/2016/05/09/elam-driver/]

TPM v2 Specification Promise

“The information in this document is formatted so that it may be converted to standard computer-language formats by an automated process. The purpose of this automated process is to minimize the transcription errors that often occur during the conversion process […]”

[https://sourceforge.net/projects/ibmswtpm2/]


Home / Browse / Security & Utilities / Security / IBM's Software TPM 2.0

IBM's Software TPM 2.0 beta

Brought to you by: kagoldman

Summary | Files | Reviews | Support | Wiki | Code | Tickets | Discussion

★ Add a Review
68 Downloads (This Week)
Last Update: 2016-11-29

Download
ibmtpm832.tar

Browse All Files

Description

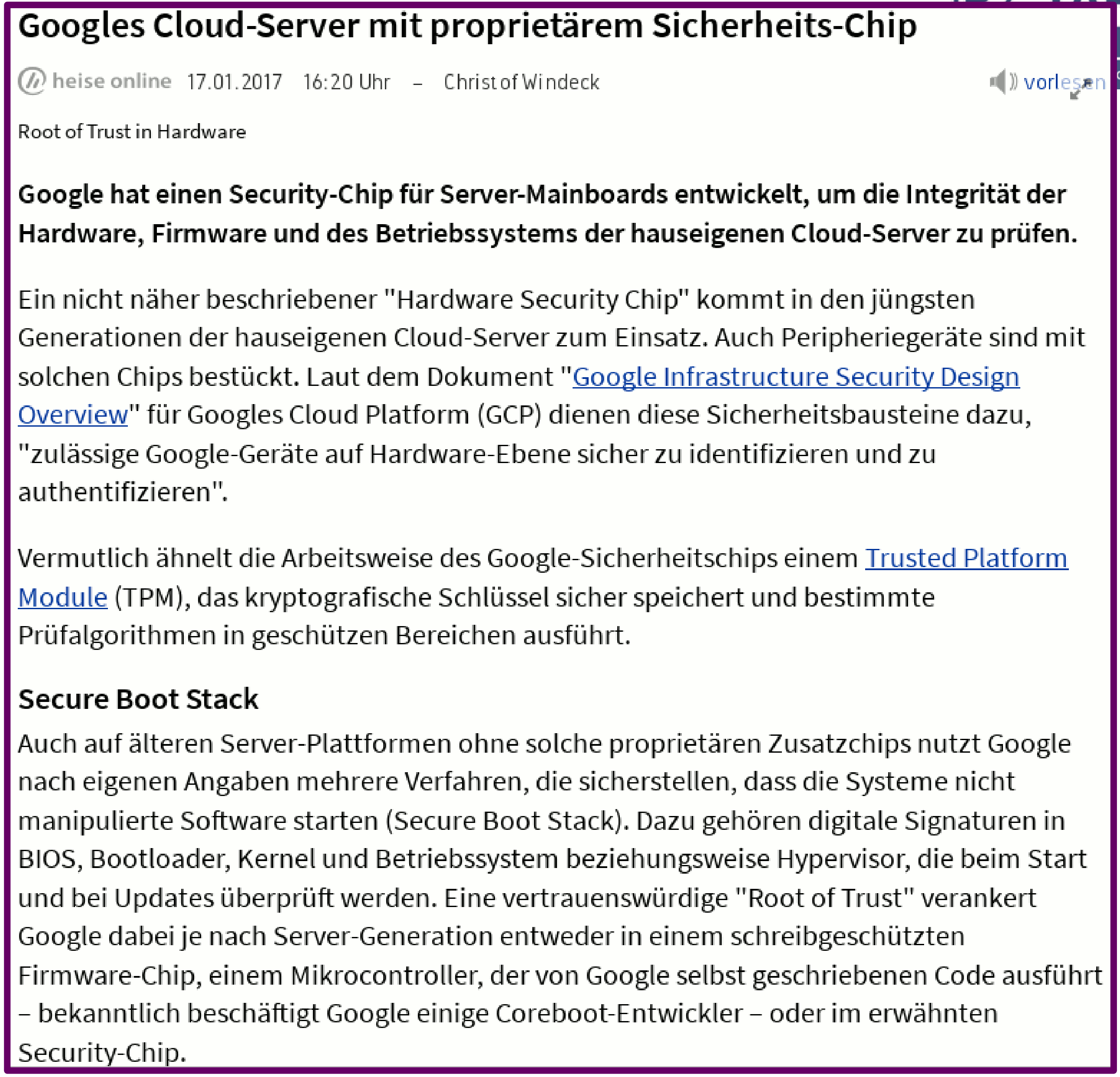
This project is an implementation of the TCG TPM 2.0 specification. It is based on the TPM specification Parts 3 and 4 source code donated by Microsoft, with additional files to complete the implementation.

See the wiki for additional support - additions to the documentation.

See the companion IBM TSS at https://sourceforge.net/projects/ibmtpm20tss/

IBM's Software TPM 2.0 Web Site >

Categories	License
Security, Cryptography	BSD License



Googles Cloud-Server mit proprietärem Sicherheits-Chip

heise online 17.01.2017 16:20 Uhr – Christof Windeck vorlesen

Root of Trust in Hardware

Google hat einen Security-Chip für Server-Mainboards entwickelt, um die Integrität der Hardware, Firmware und des Betriebssystems der hauseigenen Cloud-Server zu prüfen.

Ein nicht näher beschriebener "Hardware Security Chip" kommt in den jüngsten Generationen der hauseigenen Cloud-Server zum Einsatz. Auch Peripheriegeräte sind mit solchen Chips bestückt. Laut dem Dokument "Google Infrastructure Security Design Overview" für Googles Cloud Platform (GCP) dienen diese Sicherheitsbausteine dazu, "zulässige Google-Geräte auf Hardware-Ebene sicher zu identifizieren und zu authentifizieren".

Vermutlich ähnelt die Arbeitsweise des Google-Sicherheitschips einem Trusted Platform Module (TPM), das kryptografische Schlüssel sicher speichert und bestimmte Prüfalgorithmen in geschützen Bereichen ausführt.

Secure Boot Stack

Auch auf älteren Server-Plattformen ohne solche proprietären Zusatzchips nutzt Google nach eigenen Angaben mehrere Verfahren, die sicherstellen, dass die Systeme nicht manipulierte Software starten (Secure Boot Stack). Dazu gehören digitale Signaturen in BIOS, Bootloader, Kernel und Betriebssystem beziehungsweise Hypervisor, die beim Start und bei Updates überprüft werden. Eine vertrauenswürdige "Root of Trust" verankert Google dabei je nach Server-Generation entweder in einem schreibgeschützten Firmware-Chip, einem Mikrocontroller, der von Google selbst geschriebenen Code ausführt – bekanntlich beschäftigt Google einige Coreboot-Entwickler – oder im erwähnten Security-Chip.

[https://www.heise.de/newsticker/meldung/Googles-Cloud-Server-mit-proprietaerem-Sicherheits-Chip-3599429.html]

Privacy and Trusted Platforms

"[…] But let's frame the question in its most compelling form. Imagine a system of digital surveillance in which the algorithm was known and verifiable: We knew, that is, exactly what was being searched for; we trusted that's all that was being searched for. That surveillance was broad and indiscriminate. But before anything could be done on the basis of the results from that surveillance, a court would have to act. So the machine would spit out bits of data implicating X in some targeted crime, and a court would decide whether that data sufficed either to justify an arrest or a more traditional search. And finally, to make the system as protective as we can, the only evidence that could be used from this surveillance would be evidence directed against the crimes being surveilled for. So for example, if you're looking for terrorists, you don't use the evidence to prosecute for tax evasion. I'm not saying what the targeted crimes are; all I'm saying is that we don't use the traditional rule that allows all evidence gathered legally to be usable for any legal end. […]"

[Lessig, http://codev2.cc/]

Would such a "trusted filter" system violate privacy?

Imperfect System State Evaluation via Events Comparison

Reality

- 100% exact measurements and (side) effects analysis in practice impossible

  → Only incomplete/fuzzy collection of "important" system events and state transitions

Research

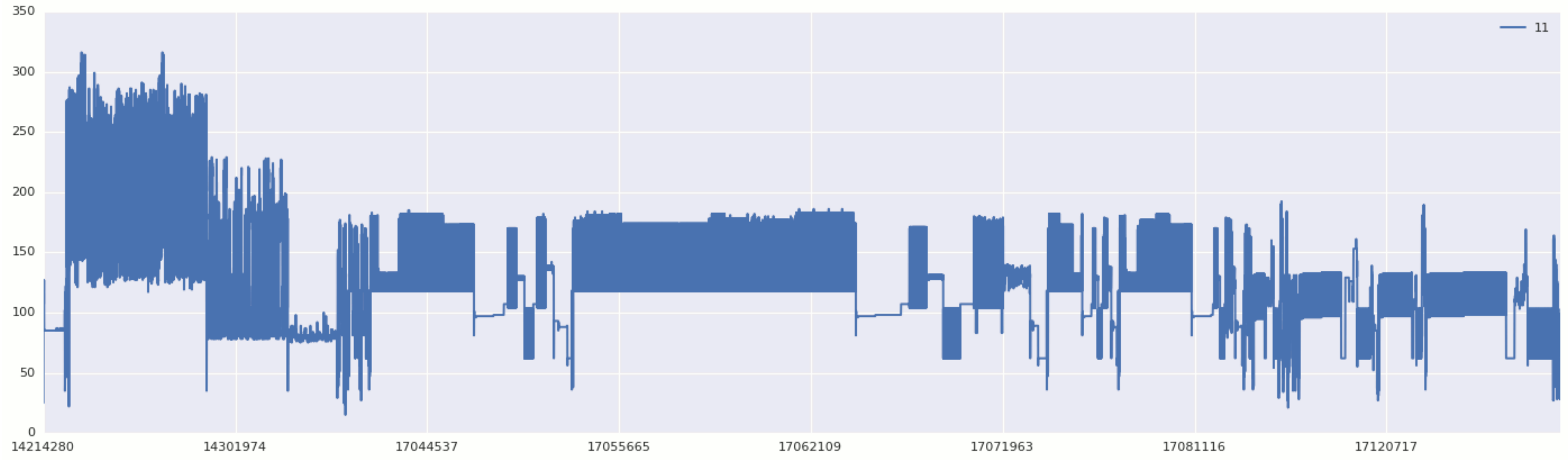
- Optimistic assumption:
  There are always *some* system events that an attacker must trigger while he transitions to root/administrator privileges.

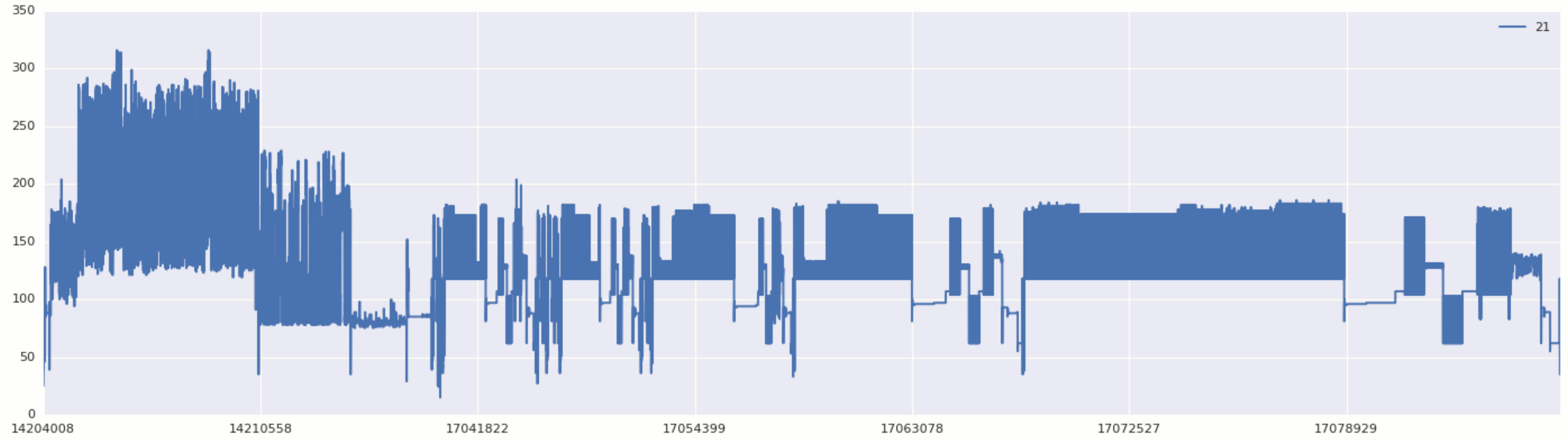
- “If something behaves differently there may be a new kind of attack going on. It should show up in a recording of system events.”

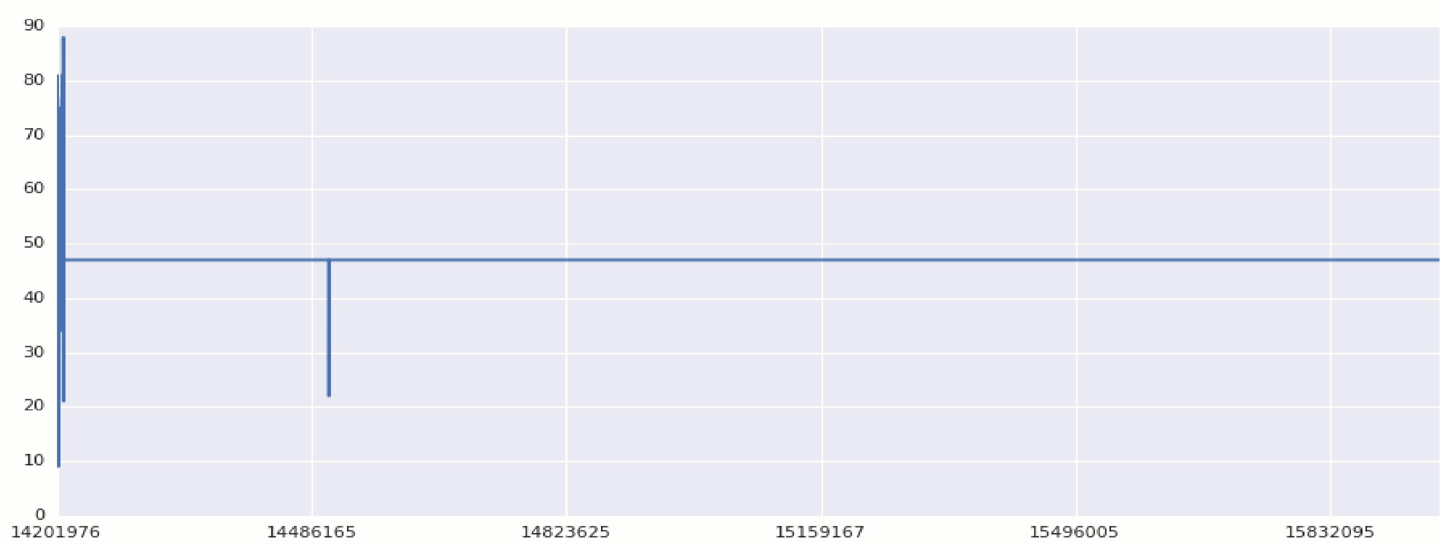
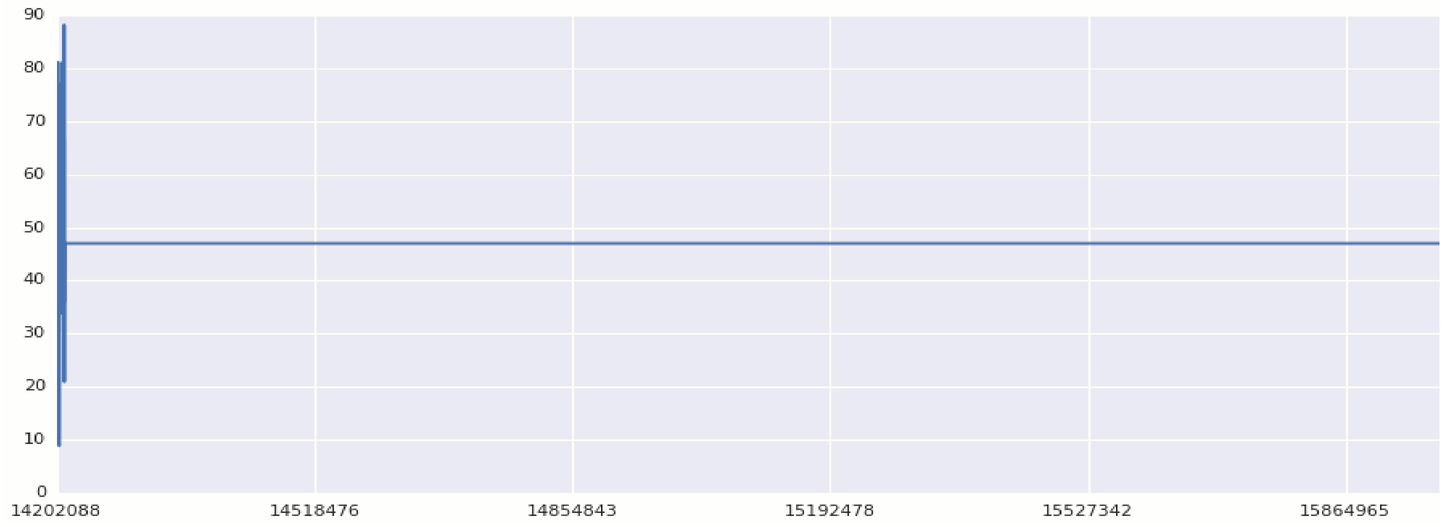
Research

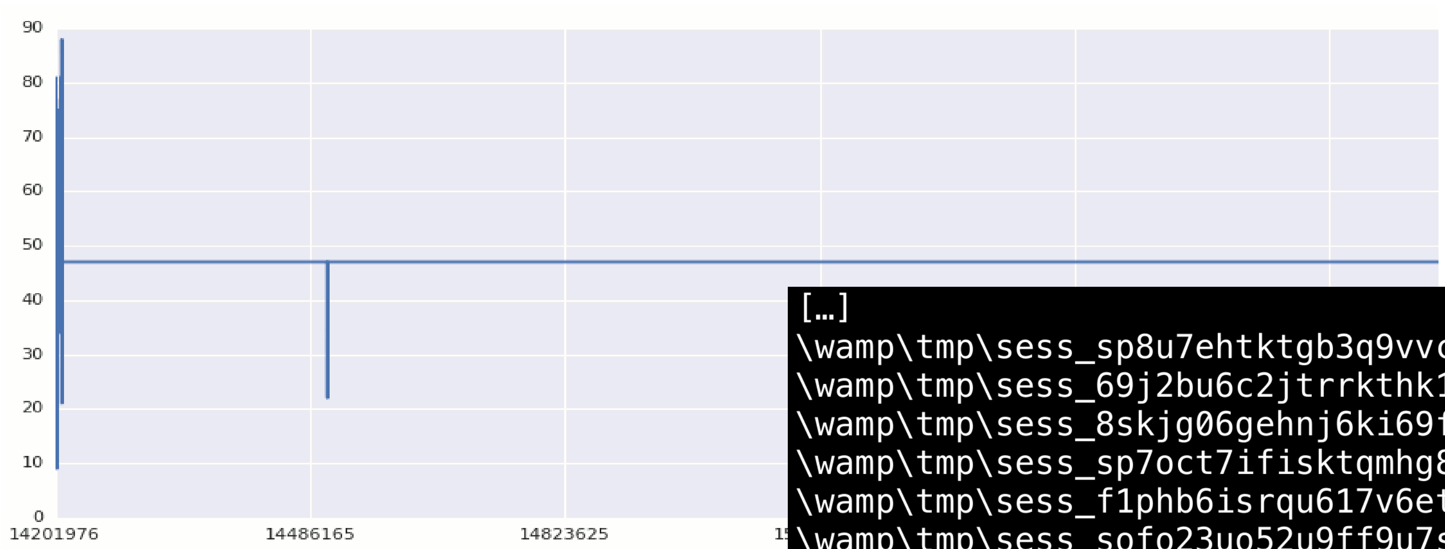
- Is it possible to match similar (good) events as they happen on multiple systems?

  → only suspicious actions remain?

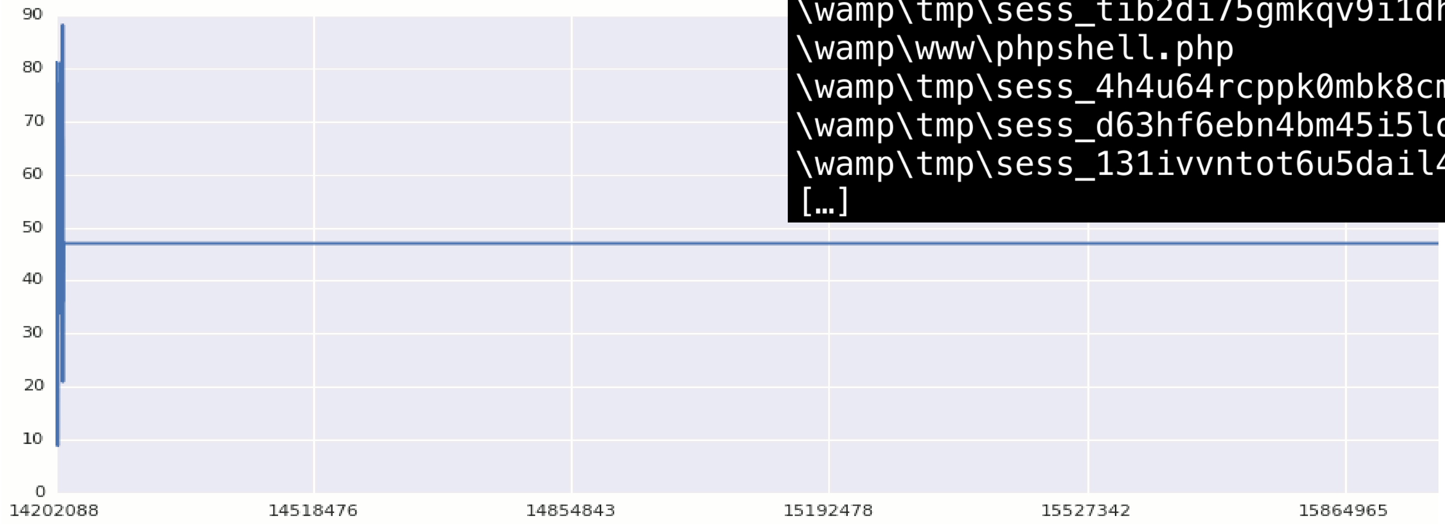

350
300
250
200
150
100
50
0
14214280
14301974
17044537
17055665
17062109
17071963
17081116
17120717
11


350
300
250
200
150
100
50
0
14204008
14210558
17041822
17054399
17063078
17072527
17078929
21


90
80
70
60
50
40
30
20
10
0
14201976
14486165
14823625
15159167
15496005
15832095

90
80
70
60
50
40
30
20
10
0
14202088
14518476
14854843
15192478
15527342
15864965

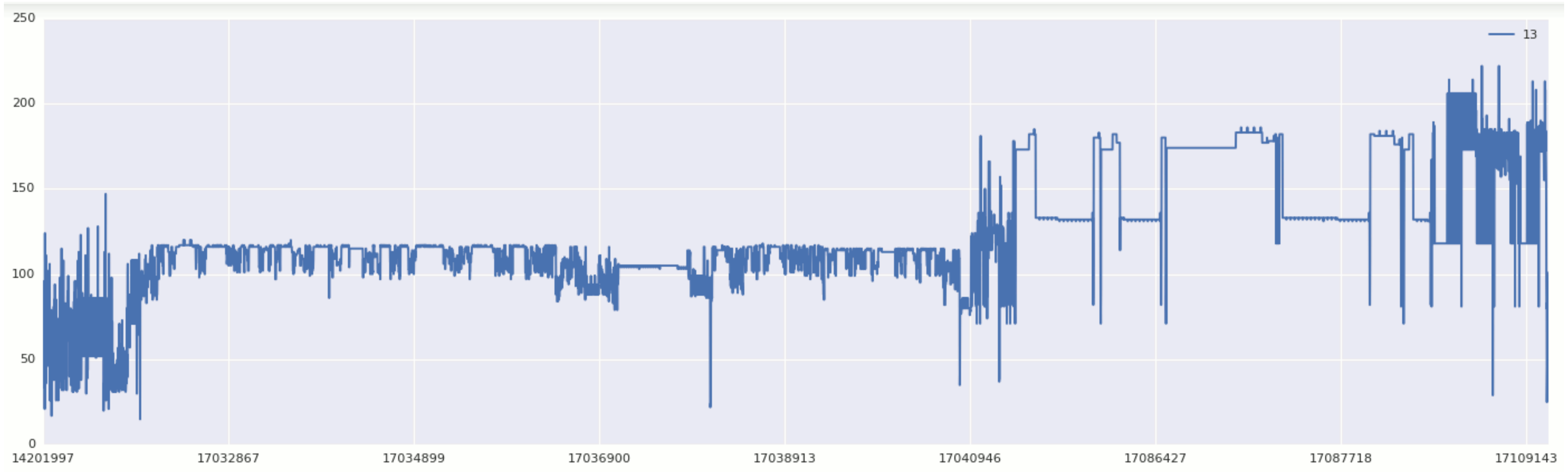


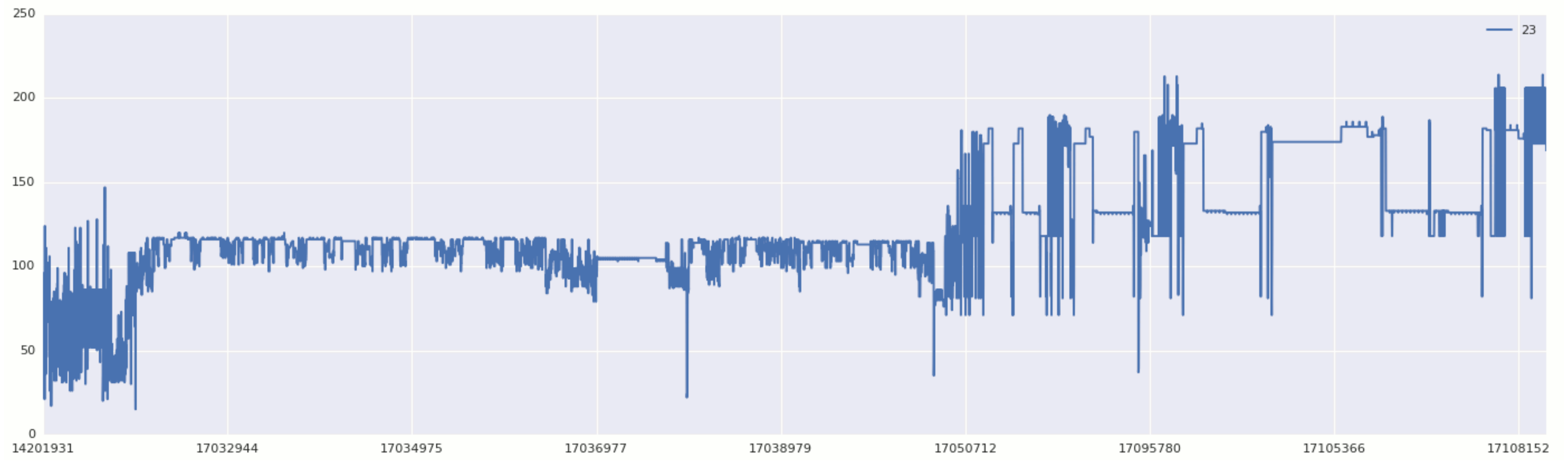
```
[…]
\wamp\tmp\sess_sp8u7ehtktgb3q9vvclhkll094gvgklf
\wamp\tmp\sess_69j2bu6c2jtrrkthk1e0i50967cjch73
\wamp\tmp\sess_8skjg06gehnj6ki69f130jg3j820bu60
\wamp\tmp\sess_sp7oct7ifisktqmhg8mf9jn3na3ca5ju
\wamp\tmp\sess_f1phb6isrqu617v6eti54n8aqc4c8od3
\wamp\tmp\sess_sofo23uo52u9ff9u7slb5var6eqkhmt7
\wamp\tmp\sess_tib2di75gmkqv9i1dh4bbor4jn1t9ku5
\wamp\www\phpshell.php
\wamp\tmp\sess_4h4u64rcppk0mbk8cmh9p8tc93u4vr75
\wamp\tmp\sess_d63hf6ebn4bm45i5lqam0kepj7llde18
\wamp\tmp\sess_131ivvntot6u5dail49l2420qs07esme
[…]
```



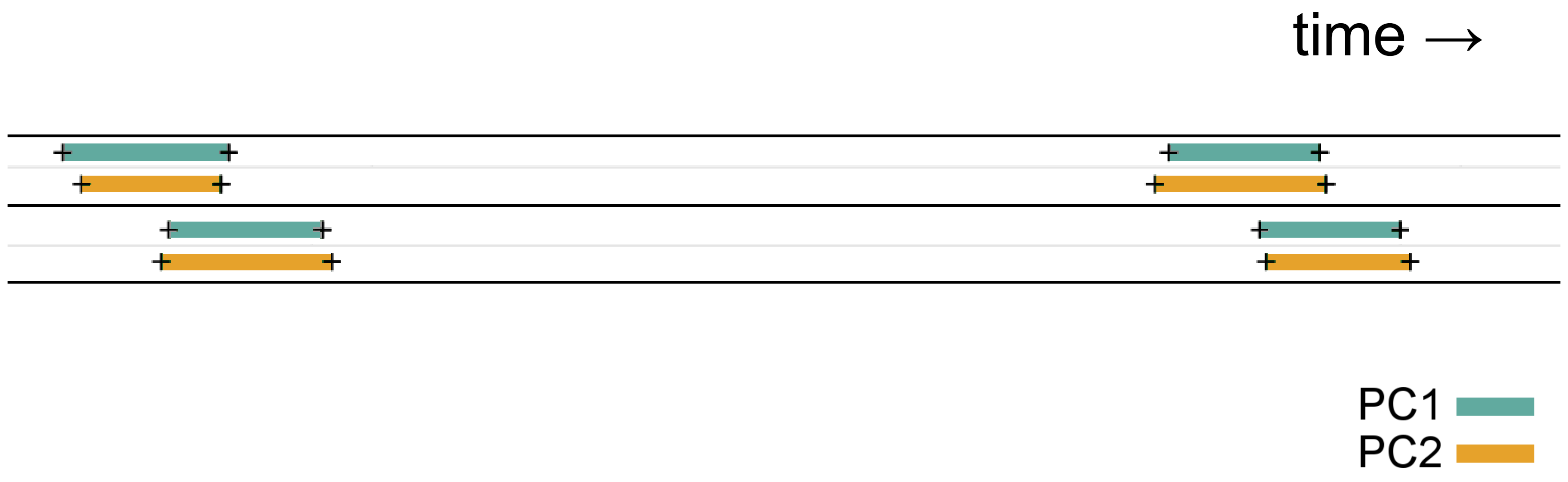
JRZ TARGET JOSEF RESSEL CENTER FOR UNIFIED THREAT INTELLIGENCE ON TARGETED ATTACKS

/fh/// st.pölten





Process Matching



Categories of Attacks

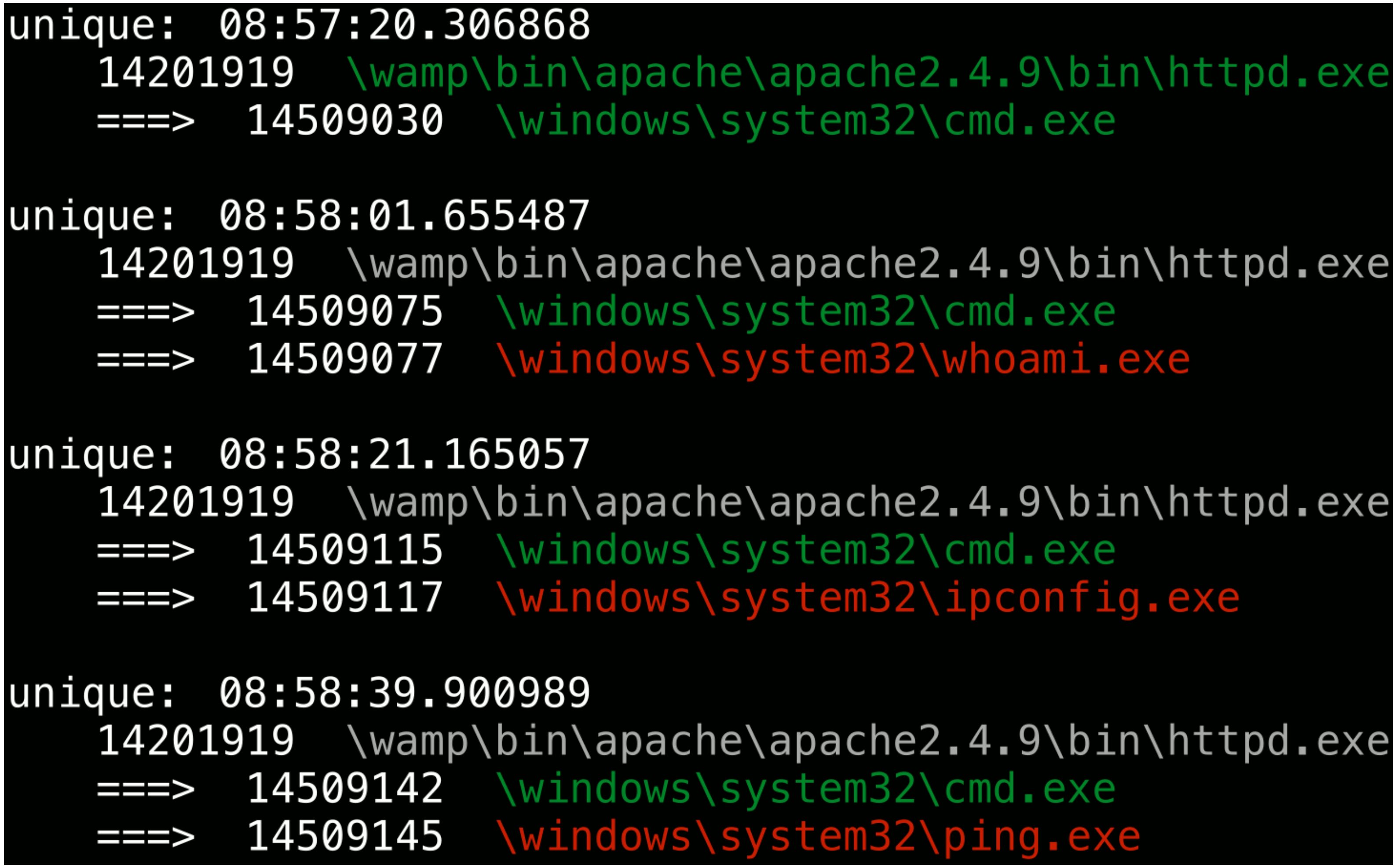
- Case 1:

  Attacker spawns new process(es)

- Case 2:
- Attacker takes control of a “good” process and nudges it to do “new” stuff

  Example: Webserver does not serve files from his html directory, but starts accessing whole filesystem

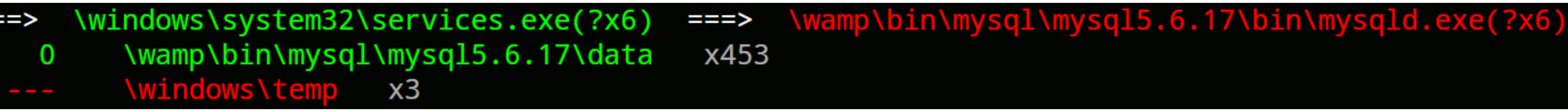
Case 1: Attacker spawns new processes



```
unique:  08:57:20.306868
    14201919  \wamp\bin\apache\apache2.4.9\bin\httpd.exe
    ===>  14509030  \windows\system32\cmd.exe

unique:  08:58:01.655487
    14201919  \wamp\bin\apache\apache2.4.9\bin\httpd.exe
    ===>  14509075  \windows\system32\cmd.exe
    ===>  14509077  \windows\system32\whoami.exe

unique:  08:58:21.165057
    14201919  \wamp\bin\apache\apache2.4.9\bin\httpd.exe
    ===>  14509115  \windows\system32\cmd.exe
    ===>  14509117  \windows\system32\ipconfig.exe

unique:  08:58:39.900989
    14201919  \wamp\bin\apache\apache2.4.9\bin\httpd.exe
    ===>  14509142  \windows\system32\cmd.exe
    ===>  14509145  \windows\system32\ping.exe
```

- Case 2:

  Attacker takes control of a well-known process and nudges it to do *new* stuff



```
==>  \windows\system32\services.exe(?x6)  ===>  \wamp\bin\mysql\mysql5.6.17\bin\mysqld.exe(?x6)
  0     \wamp\bin\mysql\mysql5.6.17\data   x453
 ---    \windows\temp   x3
```

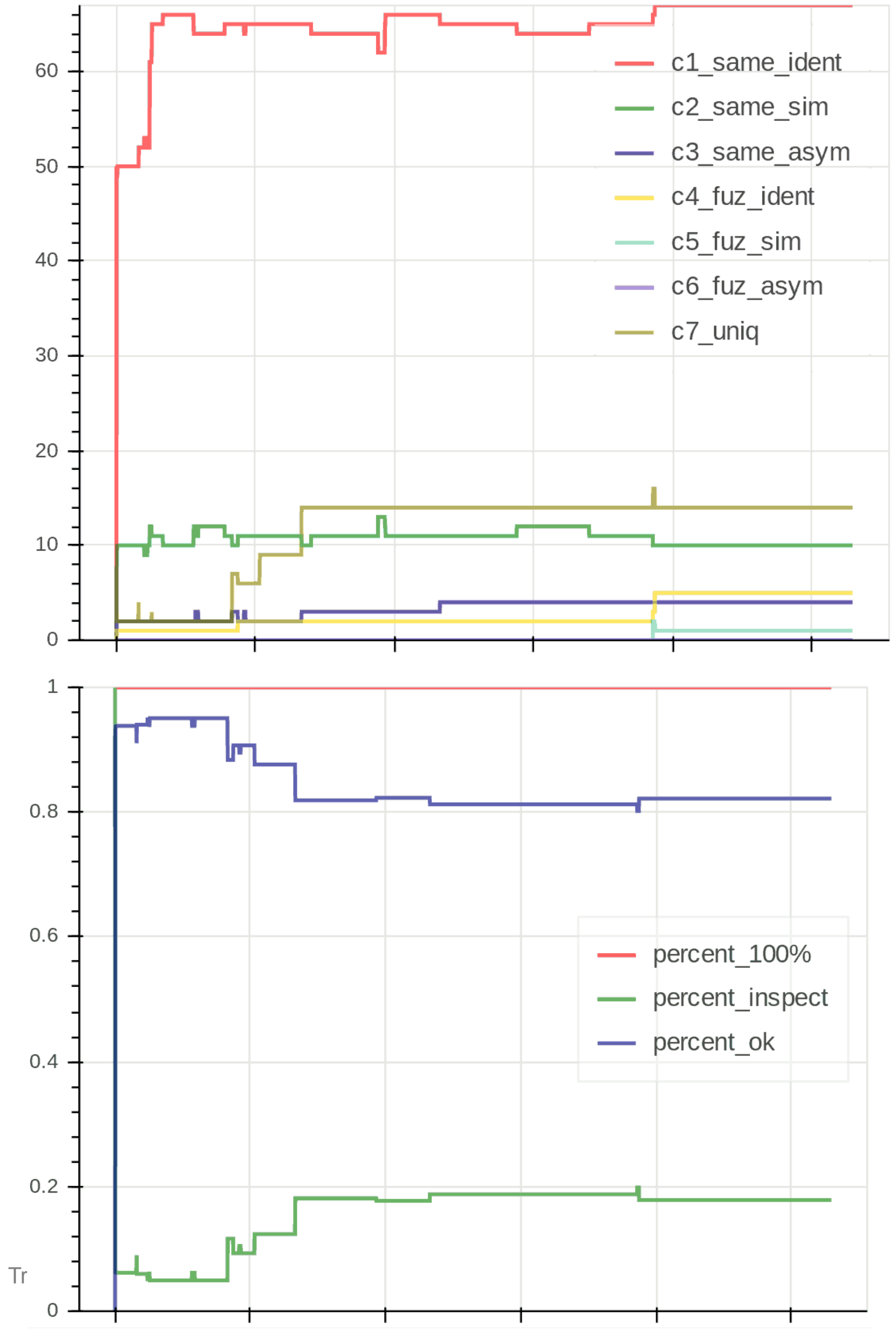
Sameness ↔ Uniqueness

- Identical process, identical events

- Identical process, quite similar
- Process looks similar, activities quite similar
- Process looks similar, because activities identical

- Identical process, some unique activity
- Process looks similar, related but some unique

- Unique/new process



[M. Pirker, P. Kochberger, S. Schwandter; Behavioural Comparison of Systems for Anomaly Detection; in: Int. Conf. On Availability, Reliability and Security (ARES), 2018]

Acknowledgements / Credits

- Some slides previously presented in IAIK's Trusted Computing lectures at Graz University of Technology.

  Thank you to all previous collaborators!

- Material cited from various sources, original referenced as [...]

References – Websites

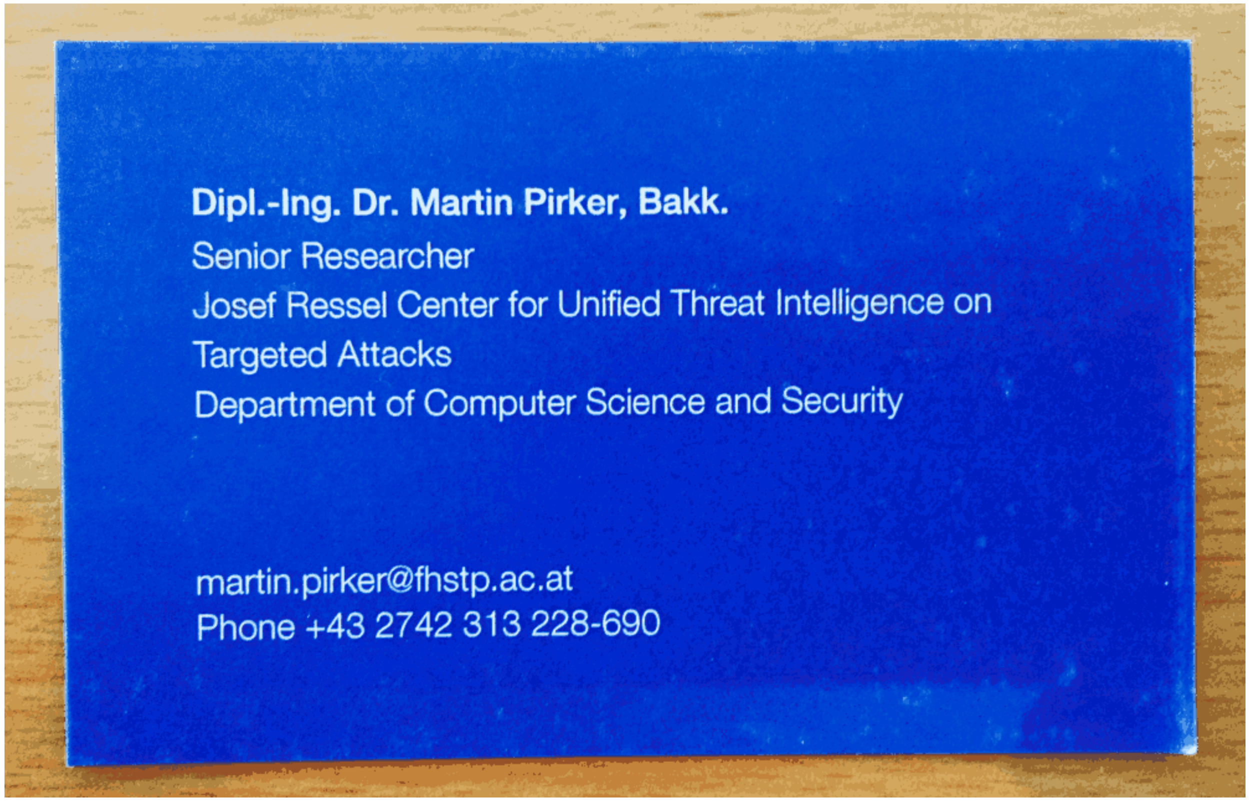
- [TCG] Trusted Computing Group, http://www.trustedcomputinggroup.org/
- Trusted Computing for the Java™ Platform http://trustedjava.sourceforge.net/
- TSS and tools in and for the C programming language http://trousers.sourceforge.net/
- TPM v1.2 software emulators https://tpm-emulator.berlios.de/ http://sourceforge.net/projects/ibmswtpm/

References – Books

- David Grawrock
  “Dynamics of a Trusted Platform”
  A building block approach
  Intel Press, 2nd edition, 2009

- Bryan Parno, Jonathan McCune, Adrian Perrig
  “Bootstrapping Trust in Modern Computers”
  Springer, 2011

Thank you! Questions?



Dipl.-Ing. Dr. Martin Pirker, Bakk.
Senior Researcher
Josef Ressel Center for Unified Threat Intelligence on Targeted Attacks
Department of Computer Science and Security

martin.pirker@fhstp.ac.at
Phone +43 2742 313 228-690