User Controlled Hardware Security Anchors: Evaluation and Designs

Dr David Oswald, Prof Mark Ryan, Prof Flavio Garcia

The University of Birmingham

Industry partners: HP Labs, Yubico
User Controlled Hardware Security Anchors: Evaluation and Designs

- **WP1**: Evaluate the security of available security anchors and **Trusted Execution Environments** (more later)

- **WP2**: Establishing secure channels between TEE and the user through ...
  - Auxiliary devices
  - Platform features for secure I/O
User Controlled Hardware Security Anchors: Evaluation and Designs

- **WP3**: Enhancing user authentication
  - Basis: FIDO(2) and U2F
  - Addressing enrollment and revocation
  - Authentication policies (e.g. location, ...)
  - Formal modelling and verification

- **WP4**: Demonstrators
  - TEE implementation
  - Smartphone app
  - Authentication token
WP1: Evaluating the state of TEE security
An overview
Hardware Security Anchors in a nutshell

- **Main technologies at present:**
  - **Trusted Platform Module** (separate chip or firmware)
  - **Intel Software Guard eXtensions** (microcode w/ HW)
  - **AMD Platform Security Processor** (separate core)
  - **ARM TrustZone** (software w/ HW support)
  - **Apple Secure Enclave Processor** (separate core, same die)
  - **Google Titan** (very recent, separate chip)

- All provide some form of running code or crypto operations in isolation

- Most require cooperation with the silicon/device manufacturer (to different extent)
Intel SGX

- Highest flexibility for the user, can run arbitrary code in “enclaves” – interesting for SW TPM
- Currently “dead” from a security perspective
  - Cache-timing side channels
  - Spectre and Meltdown variants (Foreshadow, see previous talk 😊)
  - *Software-driven faults (akin to RowHammer)?*
ARM TrustZone

```
  DCB 0, 0, 0
aStartOfRawMeta DCB "Start of Raw Metallica OTP Collected Data",0xA,0
                  ; DATA XREF: sub_30CE6+C↑o
  DCB 0
aBoot0Data     DCB "Boot 0 Data",0             ; DATA XREF: sub_30CE6+20↑o
aBoot1Data     DCB "Boot 1 Data",0             ; DATA XREF: sub_30CE6+32↑o
aSectorData    DCB "Sector Data",0             ; DATA XREF: sub_30CE6+48↑o
aEndOfRawMeta1 DCB "End of Raw Metallica OTP Collected Data",0xA,0
                  ; DATA XREF: sub_30CE6+58↑o
```
Previous attacks on Samsung TZ

- Up to Galaxy S7, attacker can roll back to old (vulnerable) versions of trustlets
- Beniamini discovered buffer overflow in OTP trustlet, allowing code execution in the context of this trustlet
- Lapid & Wool showed that KeyMaster Key Encryption Key can be extracted via OTP vuln or cache-timing side channel
Example: fingerprint matching trustlet
Example: fingerprint matching trustlet
Apple SEP

__const:0004BB60 ; Segment type: Pure data
__const:0004BB60
__const:0004BB60
__const:0004BB60
__const:0004BB60 aDerivedKey
__const:0004BB60C aSepDerivedKey
__const:0004BB7C aSepWhat
__const:0004BB85
__const:0004BB88

AREA __const, DATA, ALIGN=4
  .ORG 0x4BB60

DCB "derived_key",0
DCB "SEP derived_key",0
DCB "SE what?",0

ALIGN 4
DCD sub_14716+1
Understanding Apple SEP

- OS and firmware format documented at BH’16 in detail, but no attacks published.
- Firmware encrypted, but decryption keys for iPhone 5S published in 2017.
- Firmware image (IMG4) can be parsed and loaded into IDA using open tools.

- open file "sepdump07_sbio"

<table>
<thead>
<tr>
<th>Offset</th>
<th>Num</th>
<th>Description</th>
<th>Bits.Endian.Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0007b1f0</td>
<td>874</td>
<td>SHA256 Hash constant words K (0x428a2f98)</td>
<td>[32.le.256]</td>
</tr>
<tr>
<td>000bc5cc</td>
<td>536</td>
<td>CRC-16-IBM maxim/usb</td>
<td>[crc16.0xa001 lenorev 1.512]</td>
</tr>
<tr>
<td>000bc5cc</td>
<td>529</td>
<td>CRC-16-IBM maxim/usb</td>
<td>[crc16.0x8005 le rev int_min.512]</td>
</tr>
<tr>
<td>000bc7cc</td>
<td>648</td>
<td>CRC-32-IEEE 802.3</td>
<td>[crc32.0xedb88320 lenorev 1.1024]</td>
</tr>
<tr>
<td>000bc7cc</td>
<td>641</td>
<td>CRC-32-IEEE 802.3</td>
<td>[crc32.0x04c11db7 le rev int_min.1024]</td>
</tr>
<tr>
<td>000bd20c</td>
<td>897</td>
<td>Rijndael Te0 (0xc66363a5U)</td>
<td>[32.be.1024]</td>
</tr>
<tr>
<td>000bd60c</td>
<td>906</td>
<td>Rijndael Td0 (0x51f4a750U)</td>
<td>[32.be.1024]</td>
</tr>
<tr>
<td>000bda0c</td>
<td>894</td>
<td>AES Rijndael S / ARIA S1</td>
<td>[..256]</td>
</tr>
<tr>
<td>000bdb0c</td>
<td>895</td>
<td>AES Rijndael Si / ARIA X1</td>
<td>[..256]</td>
</tr>
<tr>
<td>000bdc30</td>
<td>878</td>
<td>Hash constant words K for SHA-384 and SHA-512</td>
<td>[64.le.640]</td>
</tr>
<tr>
<td>000bdeb0</td>
<td>1036</td>
<td>SHA1 / SHA0 / RIPEMD-160 initialization</td>
<td>[32.le.20&amp;]</td>
</tr>
<tr>
<td>000bdeb0</td>
<td>2402</td>
<td>Lucifer (outerbridge) DFLTKY</td>
<td>[..16]</td>
</tr>
<tr>
<td>000bdebc</td>
<td>2053</td>
<td>RIPEMD-128 InitState</td>
<td>[32.le.16&amp;]</td>
</tr>
<tr>
<td>000bdee4</td>
<td>1030</td>
<td>SHA256</td>
<td>[32.le.288&amp;]</td>
</tr>
<tr>
<td>000bdee4</td>
<td>876</td>
<td>SHA256 Initial hash value H (0x6a09e667UL)</td>
<td>[32.le.32&amp;]</td>
</tr>
<tr>
<td>000bdee8</td>
<td>2364</td>
<td>Crypton kp</td>
<td>[32.le.16]</td>
</tr>
</tbody>
</table>
Thanks for your attention!

Questions?

d.f.oswald@bham.ac.uk