First we break your tag, then we break your System Attacks to RFID Systems

Lukas Grunwald
Phreak.de
www.phreak.de
Agenda

- What is RFID?
- How to exploit and attack RFID systems
- Attacks against the middleware
- Reader-emulation, soft-tags
- Unexpected risk middleware
- New ways to exploit the system
- Encrypted RFID Tags (14443, MRTD)
What is RFID?

- Radio Frequency Identification (RFID)
  - Wireless transmission of information between transponder and reader without visibility
  - Bidirectional transfer (read and write)
  - Transponder (tag) can be attached, embedded or implanted
  - Automatic correlation between object and saved data
RFID is often used as generic term for complete infrastructures.

- A transponder (aka RFID-chip, -tag, -label, wireless label or simple chip)
- A reader (in fact most of them can write to the tag too)
- Some middleware, which connects the reader to a server
- Some communication infrastructure
- Integration with server farms, data warehouses, services and supporting systems
Variants

Different types of RFID transponders

<table>
<thead>
<tr>
<th>Short range</th>
<th>Mid range</th>
<th>Long range</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 15 centimeter</td>
<td>&lt;= 5 meter</td>
<td>Up to 500 meter</td>
</tr>
<tr>
<td>ISO 14443 A+B</td>
<td>ISO 15693</td>
<td>ISO 18000-xx</td>
</tr>
<tr>
<td>13.56 MHz, 125-134.2kHz</td>
<td>13.56 MHz, 125-135kHz</td>
<td>860-956 MHz (UHF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.4 GHz (Microwave)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.8 GHz (Microwave)</td>
</tr>
<tr>
<td>E-field, magnetic</td>
<td>EM-field</td>
<td>EM-field</td>
</tr>
<tr>
<td>field</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transponders

There are different kinds of transponders:

- Only transmitting a unique ID (serial-number)
  - Only passive
  - Identification
  - Tracking (Fast-track)
  - Only clear text communication
There are different types of transponders:

- **Storage of Data / Metadata R/W WORM**
  - Most passive, some active
  - EPC
  - Smart Labels
  - Most use clear text communication, some are with encrypted communication
Transponders

There are different types of transponders:

- Act as Smart Card Interface
  - Most active, some passive
  - Biometric Passport (ICAO - MRTD)
  - Access Control System (Mifare DESFire)
  - Encryption, authentication, encrypted communication
Generic Attacks

- Sniffing of the communication between transponder and reader
  - Counterfeiting of the communication
  - Obtain UID, user data and meta data
  - Basic attack on structures and tags
  - Replay attack to fool the access control systems
Generic Attacks

- Counterfeiting the identity of the reader and unauthorized writing to the tag
  - Change of UID via manipulation of the administrative block
  - Declare false identity
  - UID must be readable in clear text
  - Manipulation of product groups and prices
Generic Attacks

- Manipulation of data stored on the transponder
  - Manipulation of data
  - Manipulation of metadata
  - Swap of objects
  - Logical duplication of objects
Generic Attacks

- Deactivation of the transponder
  - Disable the traceability of objects
  - Disable the visibility of objects
Generic Attacks

- Attack the structures in the middleware and backends, manipulation of data structures.
  - Injection of malware into the backend and middleware systems
  - E.g. database worms
  - Manipulation of backend systems
  - Denial of Service attack against the infrastructure
Generic Attacks

- Jamming of the RFID frequencies
  - Use of “out-of-the-box” police jammer (broadband jamming transmitter)
  - Attack against anti-collision (RSA attack)
  - Prevent reading of the tag
  - Simple denial of service attack against the RFID System
  - Shut down production, sales or access
Encrypted RFID

- MIFARE are the most used RFID transponders featuring encryption
  - Technology is owned by Philips Austria GmbH
  - Technology is based on
    - ISO 14443
    - 13.56 MHz Frequency
MIFARE Tags

- MIFARE Standard
  - Proprietary high-level protocol
  - Philips proprietary security protocol for authentication and ciphering
  - MIFARE UltraLight: same tags without encryption
MIFARE Tags

- MIFARE Pro, ProX, and SmartMX
  - Fully comply to ISO 14443-4 standard
  - The different types of tags offer memory protected by two different keys (A and B)
  - Each sector could be protected with one of these keys.
Brute Force the Tag

- $2^{68}$ bit for the keyspace
- 25 ms per try with a brute force perl script using Linux and a self written driver
- Using one RFID reader

\[ \frac{6^{28} \cdot 0.025s}{3600s} \approx 81445305 \text{ Days} \approx 22623 \text{ Years} \]
Brute Force the Tag

- $2^{68}$ bit for the keyspace
- 25 ms per try with a brute force perl script using Linux and a self written driver
- Using 1,000 RFID readers

\[
\frac{6^{28} \times 0.025s}{3600s \times 1000} \approx 81445 \text{ Days} \approx 226 \text{ Years}
\]
MIFARE Sector Keys

- Philips puts all information under NDA
- We are not interested to sign an NDA
- Extract information from RFID software via "UNIX strings"
- Google helps a lot, Google desktop search is very popular at smartcard developers’ PCs ;)
- Look at the results
Mifare smart card
Command for loadkey function is Dx4C : Where Key A = a0a1a2a3a4a5 Key B = b0b1b2b3b4b5 : Than may be the key set 0, key set 1, and key set 2, was wrong.

[PDF] ap_dev_data_sheet
The cards do not contain access control data, but are programmed with. Philips default keys (A0A1A2A3A4A5 & B0B1B2B3B4B5) in all sector, trailers.

[PDF] standardisation_group_observin.jpg
released for public reading using the default key A: a0a1a2a3a4a5 hex. ... key A: a0a1a2a3a4a5 hex. Access conditions should allow reading with key A|B and ...

[PDF] CardMan_5x21-CL_Reader_Developer_222s_Guide
Key A: A0A1A2A3A4A5, Key B: B0B1B2B3B4B5. The Mifare cards supplied with the ... The public key for MAD is "A0A1A2A3A4A5". For complete understanding of MAD ...

Found the following default keys:

- Key A A0 A1 A2 A3 A4 A5
- Key A FF FF FF FF FF FF FF
- Key B B0 B1 B2 B3 B4 B5
- Key B FF FF FF FF FF FF FF
- About 60 keys from example applications
- No protection 00 00 00 00 00 00 00
Additional found the **Mifare Application Directory**, a PDF that shows how MIFARE are specifying the type of use of one of the transponders, each applications should have an entry to show the Type of Service
Example Layouts

- In the datasheets and „googled“ documentation are a lot of examples.
- These examples include different keys and tag / memory layout and data structure for:
  - Ticketing
  - Access Control
  - Online Payment
Software developers are lazy

- Checking a couple of cards shows that more than 75% use one of these default keys!
- It compiles let's ship it!
- The programmers use not only the example layouts, they also use the example keys!
Attack the Tag

- Directory attacks are possible with found default and example keys
  - Variations of the directory are always possible
- „Smart“ brute-force attack to the tag are possible
  - never seen a lockout or false login counter
  - a delay for a false key does not exist
Attacks to the Backend

- The memory of a ISO 15693 tag acts like a normal storage
- RFDump (Black Hat 2004) could help to manipulate data like with a hex-editor
- SQL-Injection and other attacks are possible
Preventing security functions

- If the tag is "read only" read it with RFDump and write the manipulated data to an empty one.
- Checksum, some implementations use the UID (Unique ID) as mirror block in the UD, both must be changed.
- If the block is encrypted, the Sector Key must be broken.
The RFID Supply chain

- Data Warehouse
- Production
- Distribution
- Customer Care
- Point of Sales
- Customer
- Lifecycle Management
<table>
<thead>
<tr>
<th>Addr</th>
<th>0x0</th>
<th>0x1</th>
<th>0x2</th>
<th>0x3</th>
<th>0x4</th>
<th>0x5</th>
<th>0x6</th>
<th>0x7</th>
<th>0x8</th>
<th>0x9</th>
<th>0xa</th>
<th>0xb</th>
<th>0xc</th>
<th>0xd</th>
<th>0xe</th>
<th>0xf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xb</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Problem Memory Size

Page 0x76

Byte 6
Representation to the Backend

- Looks like unlimited space on the tag
  - E.g. RFDump uses a tag database to avoid reading over the boundary
- Normally reading is event-driven
  - Reading up to the EOF
  - Input is unchecked in all implementations we have seen
Tag DoS with C-Strings

### End of String

<table>
<thead>
<tr>
<th>Addr</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1</td>
<td>68547369 69202073 6e616520 6178706d 656c6f20 206620616d696e75706100</td>
</tr>
<tr>
<td>0x2</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0x3</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0x4</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0x5</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0x6</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0x7</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0x8</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0x9</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0xa</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0xb</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0xc</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0xd</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0xe</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
<tr>
<td>0xf</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFFF</td>
</tr>
</tbody>
</table>
**Tag DoS with XML**

### Mass reading

<table>
<thead>
<tr>
<th>Addr</th>
<th>Memory in ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1</td>
<td><code>&lt;rfuid:ID&gt;urn:epc:1:4.16.36&lt;/rfuid:ID&gt;</code></td>
</tr>
<tr>
<td>0x2</td>
<td><code>&lt;rfidcore:Observation&gt;</code> <code>&lt;rfidcore:DateTime&gt;</code></td>
</tr>
<tr>
<td>0x3</td>
<td><code>&lt;rfidcore:DateTime&gt;2002-11-06T13:04:34-06:00&lt;/rfidcore:DateTime&gt;</code></td>
</tr>
<tr>
<td>0x4</td>
<td><code>&lt;pmrcore:DateTime&gt;</code></td>
</tr>
<tr>
<td>0x5</td>
<td></td>
</tr>
<tr>
<td>0x6</td>
<td></td>
</tr>
</tbody>
</table>

### Inf. Items in one Tag

<table>
<thead>
<tr>
<th>Addr</th>
<th>Memory in ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1</td>
<td><code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code></td>
</tr>
<tr>
<td>0x2</td>
<td><code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code></td>
</tr>
<tr>
<td>0x3</td>
<td><code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code></td>
</tr>
<tr>
<td>0x4</td>
<td><code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code></td>
</tr>
<tr>
<td>0x5</td>
<td><code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code></td>
</tr>
<tr>
<td>0x6</td>
<td><code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code> <code>&lt;rfuid:ID&gt;</code></td>
</tr>
</tbody>
</table>
Soft-Tags

- Emulation of RFID-Tag and/or reader
- Serial-Emulation of any ISO 15693 tag
- Useful for testing backend and middleware
- Reads „backup“ from real tags
- Manipulation of any UID, User Data or administrative block.
Thank You

Questions ?