Eric Sesterhenn <eric.sesterhenn@x41-dsec.de>
2018
X41 D-SEC GmbH
• Eric Sesterhenn
• Pentesting/Code Auditing at X41
• CCCMZ (CCCWI)

Hacktrain to 19c3
More Information

- 19C3: Smartcards mit SOSSE sind lecker
- Camp 2003: hacking smart cards
- 23C3: A not so smart card - How bad security decisions can ruin a debit card design
- 24C3: Smartcard protocol sniffing
- Camp 2011: Reviving smart card analysis
- 29C3: Milking the Digital Cash Cow - Extracting Secret Keys of Contactless Smartcards

You can find these at https://media.ccc.de
• The issues presented here have been reported and fixed!
• These are open source projects - help them!
• I am (usually) not interested in testing / debugging proprietary stuff in my spare time.
Trust the Smartcard

- Smartcards control authentication!
- Authentication runs as root!
- Driver developers and users trust the smartcard!
- Let’s abuse that

WHAT IF WE TRUST THE CARD

...BUT IT IS EVIL?
Smartcard Stack Summary

- Application (pam)
- PKCS11
- PC/SC
- APDU
- Physical Card
What is a Smartcard?

- Physical, tamper-proof device
- Designed to keep information secret
- Contains memory and a processor

https://en.wikipedia.org/wiki/Smart_card#/media/File:SmartCardPinout.svg
Application Protocol Data Unit

- APDUs form the protocol to talk to smartcards
- ISO/IEC 7816-4 Identification cards
  - Integrated circuit cards
- \( T=0 \) is character oriented / \( T=1 \) is block-oriented
- Verify: 00 20 00 01 04 31323334

<table>
<thead>
<tr>
<th>CLA</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>LC</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0-3</td>
<td>NC</td>
</tr>
</tbody>
</table>

35C3
61XX  Response bytes still available
63C0  Verify fail, no try left.
63C3  Verify fail, 3 tries left.
6982  Security condition not satisfied.
6A00  No information given (Bytes P1 and/or P2 are incorrect).
9000  Command successfully executed (OK).
9004  PIN not successfully verified, 3 or more PIN tries left.
PC/SC API

- PC/SC API can be used on win and *nix
- Other libraries have a similar interface

```c
LONG WINAPI SCardTransmit(
    SCARDHANDLE hCard,
    LPCSCARD_IO_REQUEST pioSendPci,
    LPCBYTE pbSendBuffer,
    DWORD cbSendLength,
    PSCARD_IO_REQUEST pioRecvPci,
    LPBYTE pbRecvBuffer,
    LPDWORD pcbRecvLength
);
```
PKCS11

• PKCS11 is a platform independent API for cryptographic token
• Supported by OpenSSL, browsers,... (eg. via libp11)
• Windows uses smartcard Minidriver now
• Driver for each card, uses ATR to match

```
 CK_RV C_FindObjectsInit(
    CK_SESSION_HANDLE hSession,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount
);
```

Smartcard for Sign-On

- PAM
  - Get Certificates
  - Certificate
  - Validate Certificate and User
  - Revocation Check
  - Generate Nonce
  - Sign Request for Nonce
  - Signature
  - Check Signature Against Certificate

- Smartcard

- CRL Server
  - CRL
Documento Nacional de Identidad electrónico is the Spanish eID. It is in line with the EU directive on electronic ID, and it is a “smart” identity card with a chip containing certificates for authentication and digital signature.
"file" : {
    "fortify_source" : "no",
    "fortify-able" : "4",
    "pie" : "dso",
    "rpath" : "no",
    "relro" : "partial",
    "fortified" : "0",
    "nx" : "yes",
    "canary" : "no",
    "filename" : "libpkcs11-dnie.so",
}

DNI Hardening
Third Party Code

- CryptoPP - 5.2MB text size
- ASN1C - 1.4MB text size
- No copyright notice with package

<table>
<thead>
<tr>
<th>#</th>
<th>CVE ID</th>
<th>CWE ID</th>
<th># of Exploits</th>
<th>Vulnerability Type(s)</th>
<th>Publish Date</th>
<th>Update Date</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CVE-2017-9424</td>
<td>125</td>
<td></td>
<td></td>
<td>2017-06-05</td>
<td>2017-06-13</td>
<td>5.0</td>
</tr>
<tr>
<td>2</td>
<td>CVE-2016-9339</td>
<td>20</td>
<td></td>
<td></td>
<td>2017-01-30</td>
<td>2017-02-07</td>
<td>5.0</td>
</tr>
<tr>
<td>3</td>
<td>CVE-2016-7544</td>
<td>399</td>
<td></td>
<td></td>
<td>2017-01-30</td>
<td>2017-02-07</td>
<td>5.0</td>
</tr>
<tr>
<td>4</td>
<td>CVE-2016-7420</td>
<td>200</td>
<td>+Info</td>
<td></td>
<td>2016-09-16</td>
<td>2016-11-28</td>
<td>4.3</td>
</tr>
<tr>
<td>5</td>
<td>CVE-2016-3996</td>
<td>200</td>
<td>+Info</td>
<td></td>
<td>2017-02-13</td>
<td>2017-03-03</td>
<td>5.0</td>
</tr>
</tbody>
</table>
Toying around - DoS

> 00 c0 00 00 00  
< 61 00  
I have another 0 bytes

> 00 c0 00 00 00  
< 61 00  
I have another 0 bytes

> 00 c0 00 00 00  
< 61 00  
I have another 0 bytes

> 00 c0 00 00 00  
< 61 00  
I have another 0 bytes
Toying around - Crash

> 00 c0 00 00 00 Get Response
< ........ 61 FF I have another 255 bytes

> 00 c0 00 00 00 Get Response
< ........ 61 FF I have another 255 bytes

> 00 c0 00 00 00 Get Response
< ........ 61 FF I have another 255 bytes

#8 0xb6e697ff in operator new (sz=2097152000) at
→ ../.../.../.../src/libstdc++-v3/libsupc++/new_op.cc:54
Toying around - Crash

> 00 b0 92 00 04
< 90 00

Read Binary,
Everything is fine

0xb796c94a in CCommunicator::readEF_sequence(unsigned short, byteBuffer&,
unsigned short) () from /usr/lib/libpkcs11-dnie.so
Toying around

0xb7f9bc70 in CUtil::GetBit(BIT_STRING_s*, unsigned long) () from
← /usr/lib/libpkcs11-dnie.so

OCTET_STRING_decode_ber: Assertion `ctx->left >= 0' failed.

Use of uninitialised value of size 4

0xb7f7bf5b in CP15TokenInfo::LoadTokenInfo(CK_TOKEN_INFO*) () from
← /usr/lib/libpkcs11-dnie.so
# Bugs

<table>
<thead>
<tr>
<th>Project</th>
<th># Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>libykneomgr</td>
<td>1</td>
</tr>
<tr>
<td>OpenSC</td>
<td>Over 9000 ;-)</td>
</tr>
<tr>
<td>pam_pkcs11</td>
<td>1</td>
</tr>
<tr>
<td>smartcardservices</td>
<td>2</td>
</tr>
<tr>
<td>Yubico-Piv</td>
<td>2</td>
</tr>
</tbody>
</table>

No, I did not fuzz the &$#?$! out of it... but guess which one I fuzzed the most ;-) 
Thanks to Frank Morgner for fixing!
do {
    cacreturn = cacToken.exchangeAPDU(command, sizeof(command), result, resultLength);
    if ((cacreturn & 0xFF00) != 0x6300)
        CACError::check(cacreturn);

    memcpy(certificate + certificateLength, result, resultLength - 2);
    certificateLength += resultLength - 2;
    // Number of bytes to fetch next time around is in the last byte returned.
    command[4] = cacreturn & 0xFF;
} while ((cacreturn & 0xFF00) == 0x6300);
u8 buf[2048], *p = buf;
size_t bufsize, keysize;

sc_format_path("I1012", &path);
r = sc_select_file(card, &path, &file);
if (r)
    return 2;
bufsize = file->size;
sc_file_free(file);
r = sc_read_binary(card, 0, buf, bufsize, 0);
Popping calcs...

```
snakebyte@smartcard:~$ cryptoflex-tool
Usage: cryptoflex-tool [OPTIONS]
Options:
  -l, --list-keys          Lists all keys in a public key file
  -c, --create-key-files <arg> Creates new RSA key files for <arg> keys
  -P, --create-pin-file <arg> Creates a new CHV<arg> file
  -g, --generate-key      Generates a new RSA key pair
  -R, --read-key          Reads a public key from the card
  -V, --verify-pin        Verifies CHV1 before issuing commands
  -k, --key-num <arg>     Selects which key number to operate on [1]
  -a, --app-df <arg>      Selects the DF to operate in
  -p, --prkey-file <arg>  Private key file
  -u, --pubkey-file <arg> Public key file
  -e, --exponent <arg>    The RSA exponent to use in key generation [3]
  -m, --modulus-length <arg> Modulus length to use in key generation [1024]
  -r, --reader <arg>      Uses reader <arg>
  -W, --wait              Wait for card insertion
  -V, --verbose           Verbose operation. Use several times to enable debug output

snakebyte@smartcard:~$ cryptoflex-tool -R
Using reader with a card: libfuzzy
Using card driver: Schlumberger Multiflex/Cryptoflex
Unable to read public key file: Card command failed
bc 1.06.95
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
3+4
7
```
if(*out_len + recv_len - 2 > max_out) {
    fprintf(stderr,
        "Output buffer to small, wanted to write %lu, max was %lu."
        , *out_len + recv_len - 2, max_out);
}
if(out_data) {
    memcpy(out_data, data, recv_len - 2);
    out_data += recv_len - 2;
    *out_len += recv_len - 2;
}
Logging in...

Debian GNU/Linux 9 smartcard tty0

Hint: Num Lock on

smartcard login: _
Basic Smartcard Exploitation in 2018

- Basiccard gives you nice control,... yes BASIC!
- Allows to specify custom ATR
- Controls full communication
- http://basiccard.com/
Declare Command &HC0 &HA4 MySelectFile(S$)
Declare Command &HC0 &HB0 MyReadBinary(Lc=0, S$)

Declare ATR = Chr$(&H3B, &H95, &H15, &H40, &H20, &H68, &H01, &H02, &H00, &] ← H00)
Implementation

Command &HC0 &HA4 MySelectFile(S$)
...

If Lc = 2 Then
  S$ = BinToHex$(S$)
  If S$ = "3F00" Then
    S$ = SelectFile1$
  Else If S$ = "1012" Then
    S$ = SelectFile2$
  End If
End If
SW1SW2  = swCommandOK
End Command
Exploit Released

• Example exploit code available now!
• Just for flextool, kinda silly but shows how it works
• https://x41-dsec.de/Ke<inline>

Kevin.zip
Challenges in fuzzing a protocol

- Most modern fuzzers are file-oriented
- Radamsa: Generates a corpus of files
- Hongfuzz: passes a file (filename different each run)
- libfuzzer: passes a buffer and length
- AFL: passes a file
Challenges in fuzzing a protocol

- `SCardTransmit()` tells us how much data it expects
- Read this from a file on each call and error out if EOF
- No complicated poll handling like for network sockets required

```c
LONG WINAPI SCardTransmit(
    SCARDHANDLE hCard,
    LPCSCARD_IO_REQUEST pioSendPci,
    LPCBYTE pbSendBuffer,
    DWORD cbSendLength,
    PSCARD_IO_REQUEST pioRecvPci,
    LPBYTE pbRecvBuffer,
    LPDWORD pcbRecvLength
);
```
How to fuzz - OpenSC

- reader-fuzzy.c
- Implements a (virtual) smartcard reader interface
- Responds with malicious data read from file (OPENSCEFUZZ_FILE)
- Have fun with AFL
How to fuzz - Winscard and PC/SC

• Winscard(.dll) on Linux and Unix
• For proprietary code
• Preload the library
• Have fun with non-feedback fuzzers (e.g. radamsa) or AFL in qemu mode
• Tavis loadlibrary
• Extended to support Winscard drivers
• Fuzz the windows drivers on linux without all the overhead

Surprise, I ported Windows Defender to Linux. 😎

taviso/loadlibrary
Porting Windows Dynamic Link Libraries to Linux. Contribute to loadlibrary development by creating an account on GitHub.
github.com

14:45 - 23. Mai 2017
Smartcard fuzzing

- Released at DEF CON 2018
- https://github.com/x41sec/x41-smartcard-fuzzing
# AFL fuzzing

## american fuzzy lop 2.51b (slave02)

<table>
<thead>
<tr>
<th>process timing</th>
<th>overall results</th>
</tr>
</thead>
<tbody>
<tr>
<td>run time: 22 days, 6 hrs, 0 min, 56 sec</td>
<td>cycles done: <strong>410</strong></td>
</tr>
<tr>
<td>last new path: 0 days, 3 hrs, 45 min, 47 sec</td>
<td>total paths: <strong>3172</strong></td>
</tr>
<tr>
<td>last uniq crash: 1 days, 3 hrs, 18 min, 59 sec</td>
<td>uniq crashes: <strong>1</strong></td>
</tr>
<tr>
<td>last uniq hang: none seen yet</td>
<td>uniq hangs: <strong>0</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cycle progress</th>
<th>map coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>now processing: 979 (30.86%)</td>
<td>map density: 5.66% / 24.35%</td>
</tr>
<tr>
<td>paths timed out: 0 (0.00%)</td>
<td>count coverage: 4.16 bits/tuple</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stage progress</th>
<th>findings in depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>now trying: havoc</td>
<td>favored paths: 628 (19.80%)</td>
</tr>
<tr>
<td>stage execs: 26/128 (20.31%)</td>
<td>new edges on: 766 (24.15%)</td>
</tr>
<tr>
<td>total execs: 116M</td>
<td>total crashes: 1 (1 unique)</td>
</tr>
<tr>
<td>exec speed: 118.5/sec</td>
<td>total tmouts: 49 (22 unique)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fuzzing strategy yields</th>
<th>path geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit flips: n/a, n/a, n/a</td>
<td>levels: 3</td>
</tr>
<tr>
<td>byte flips: n/a, n/a, n/a</td>
<td>pending: 1</td>
</tr>
<tr>
<td>arithmetics: n/a, n/a, n/a</td>
<td>pend fav: 0</td>
</tr>
<tr>
<td>known ints: n/a, n/a, n/a</td>
<td>own finds: 87</td>
</tr>
<tr>
<td>dictionary: n/a, n/a, n/a</td>
<td>imported: 841</td>
</tr>
<tr>
<td>havoc: 35/38.3M, 53/73.5M</td>
<td>stability: 100.00%</td>
</tr>
<tr>
<td>trim: 5.05%/4.17M, n/a</td>
<td></td>
</tr>
</tbody>
</table>

[cpu002: 25%]
### LCOV - code coverage report

**Current view:** top level - libopensc  
**Test:** trace.lcov.info_final  
**Date:** 2018-08-07 17:54:04

<table>
<thead>
<tr>
<th>Filename</th>
<th>Line Coverage</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>pkcs15-skey.c</td>
<td>0.0 % 0/89</td>
<td>0.0 % 0/2</td>
</tr>
<tr>
<td>sc-ossi-compat.h</td>
<td>0.0 % 0/37</td>
<td>0.0 % 0/3</td>
</tr>
<tr>
<td>base64.c</td>
<td>0.0 % 0/75</td>
<td>0.0 % 0/3</td>
</tr>
<tr>
<td>ctbc.c</td>
<td>0.0 % 0/123</td>
<td>0.0 % 0/3</td>
</tr>
<tr>
<td>pkcs15-cache.c</td>
<td>0.0 % 0/80</td>
<td>0.0 % 0/3</td>
</tr>
<tr>
<td>pkcs15-sec.c</td>
<td>0.0 % 0/201</td>
<td>0.0 % 0/6</td>
</tr>
<tr>
<td>padding.c</td>
<td>0.0 % 0/120</td>
<td>0.0 % 0/7</td>
</tr>
<tr>
<td>p15card_helper.c</td>
<td>0.0 % 0/176</td>
<td>0.0 % 0/10</td>
</tr>
<tr>
<td>reader-tr03119.c</td>
<td>0.0 % 0/393</td>
<td>0.0 % 0/12</td>
</tr>
<tr>
<td>card-mioois.c</td>
<td>0.0 % 0/220</td>
<td>0.0 % 0/14</td>
</tr>
<tr>
<td>pkcs15-algo.c</td>
<td>0.0 % 0/198</td>
<td>0.0 % 0/16</td>
</tr>
<tr>
<td>jasecc-sm.c</td>
<td>0.0 % 0/333</td>
<td>0.0 % 0/17</td>
</tr>
<tr>
<td>card-jcop.c</td>
<td>0.0 % 0/476</td>
<td>0.0 % 0/21</td>
</tr>
<tr>
<td>jasecc-sdo.c</td>
<td>0.0 % 0/713</td>
<td>0.0 % 0/25</td>
</tr>
<tr>
<td>cwa14890.c</td>
<td>3.9 % 29/745</td>
<td>6.5 % 2/31</td>
</tr>
</tbody>
</table>

**Lines:** 13213 41611 31.8 %  
**Functions:** 796 1913 41.6 %
<table>
<thead>
<tr>
<th>File</th>
<th>Coverage</th>
<th>Lines</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>sc.c</td>
<td>73.4%</td>
<td>351/478</td>
<td></td>
<td>100%</td>
<td>38/49</td>
</tr>
<tr>
<td>card-belpic.c</td>
<td>69.3%</td>
<td>95/137</td>
<td></td>
<td>100%</td>
<td>7/9</td>
</tr>
<tr>
<td>card-cac.c</td>
<td>70.7%</td>
<td>573/811</td>
<td></td>
<td>100%</td>
<td>33/42</td>
</tr>
<tr>
<td>pkcs15-openpgp.c</td>
<td>24.7%</td>
<td>39/158</td>
<td></td>
<td>100%</td>
<td>4/5</td>
</tr>
<tr>
<td>muscle-filesystem.c</td>
<td>85.0%</td>
<td>102/120</td>
<td></td>
<td>100%</td>
<td>9/11</td>
</tr>
<tr>
<td>card-jpki.c</td>
<td>68.8%</td>
<td>139/202</td>
<td></td>
<td>100%</td>
<td>10/12</td>
</tr>
<tr>
<td>reader-fuzzy.c</td>
<td>87.7%</td>
<td>93/106</td>
<td></td>
<td>100%</td>
<td>10/11</td>
</tr>
<tr>
<td>errors.c</td>
<td>91.2%</td>
<td>31/34</td>
<td></td>
<td>100%</td>
<td>1/1</td>
</tr>
<tr>
<td>pkcs15-jpki.c</td>
<td>94.0%</td>
<td>79/84</td>
<td></td>
<td>100%</td>
<td>2/2</td>
</tr>
<tr>
<td>simpletv.c</td>
<td>100.0%</td>
<td>33/33</td>
<td></td>
<td>100%</td>
<td>2/2</td>
</tr>
<tr>
<td>ef-gdo.c</td>
<td>89.4%</td>
<td>42/47</td>
<td></td>
<td>100%</td>
<td>2/2</td>
</tr>
<tr>
<td>pkcs15-esinit.c</td>
<td>89.3%</td>
<td>25/28</td>
<td></td>
<td>100%</td>
<td>3/3</td>
</tr>
<tr>
<td>pkcs15-westcos.c</td>
<td>69.6%</td>
<td>87/125</td>
<td></td>
<td>100%</td>
<td>3/3</td>
</tr>
<tr>
<td>ef-atr.c</td>
<td>93.9%</td>
<td>77/82</td>
<td></td>
<td>100%</td>
<td>3/3</td>
</tr>
<tr>
<td>gp.c</td>
<td>100.0%</td>
<td>18/18</td>
<td></td>
<td>100%</td>
<td>3/3</td>
</tr>
<tr>
<td>pkcs15-atrust-acos.c</td>
<td>86.0%</td>
<td>92/107</td>
<td></td>
<td>100%</td>
<td>4/4</td>
</tr>
<tr>
<td>sm.c</td>
<td>89.0%</td>
<td>65/73</td>
<td></td>
<td>100%</td>
<td>4/4</td>
</tr>
<tr>
<td>pkcs15-gemsafeV1.c</td>
<td>94.3%</td>
<td>198/210</td>
<td></td>
<td>100%</td>
<td>8/8</td>
</tr>
<tr>
<td>pkcs15-tcos.c</td>
<td>82.5%</td>
<td>217/263</td>
<td></td>
<td>100%</td>
<td>9/9</td>
</tr>
<tr>
<td>apdu.c</td>
<td>70.0%</td>
<td>219/313</td>
<td></td>
<td>100%</td>
<td>12/12</td>
</tr>
</tbody>
</table>
pam_pkcs11: Replay an Authentication
pam_pkcs11: Replay an Authentication

- User logs into attacker controlled computer
- Attacker asks for Nonce and for Signature
- Attacker creates malicious card and can replay the authentication

This is even worse if the key is also used to sign other data!
Roadblocks

- Channel back to card is quite limited
- Might need to use revocation list check for information leaks
- Interaction during exploitation not possible with basiccard, get SIMtrace for that
- But: A single bitflip from false to true during login can be enough :)}
• Think about trust models!
• Some security measures increase your attack surface big time!
• Fuzz Everything!
• Limit attack surface by disabling certain drivers.
• Do not write drivers in C ;-)}
Thanks

- Q & A
- https://github.com/x41sec/x41-smartcard-fuzzing
- eric.sesterhenn@x41-dsec.de
- Sorry no Twitter... stalk me on LinkedIn if you must ;-)

https://www.x41-dsec.de/