IN SOVIET RUSSIA
SMARTCARD HACKS YOU

@35C3 - 2018

Eric Sesterhenn <eric.sesterhenn@x41-dsec.de>

2018

X41 D-SEC GmbH

whoami

- Eric Sesterhenn
- Pentesting/Code Auditing at X41
- CCCMZ (CCCWI)

Hacktrain to 19c3

More Information

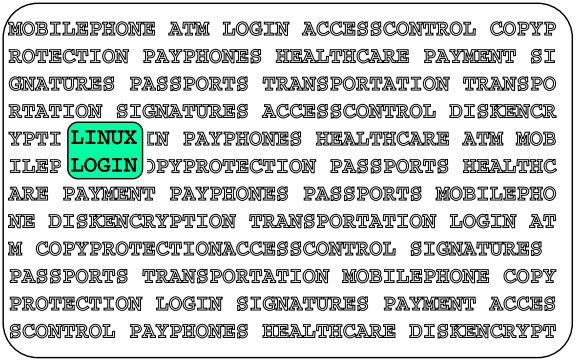
- 19C3: Smartcards mit SOSSE sind lecker
- Camp 2003: hacking smart cards
- 23C3: A not so smart card - How bad security decisions can ruin a debit card design
- 24C3: Smartcard protocol sniffing
- Camp 2011: Reviving smart card analysis
- 29C3: Milking the Digital Cash Cow - Extracting Secret Keys of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

You can find these at `https://media.ccc.de`

Disclaimer

- 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.

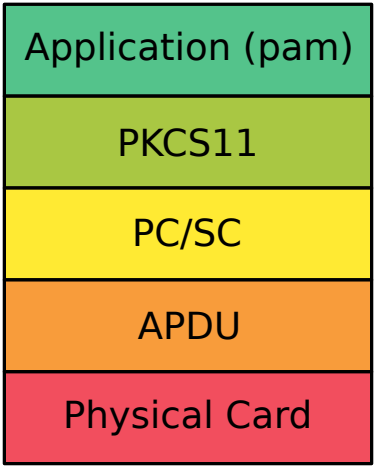
Targets


MOBILEPHONE ATM LOGIN ACCESSCONTROL COPYP
ROTECTION PAYPHONES HEALTHCARE PAYMENT SI
GNATURES PASSPORTS TRANSPORTATION TRANSPO
RTATION SIGNATURES ACCESSCONTROL DISKENCR
YPTI LINUX N PAYPHONES HEALTHCARE ATM MOB
ILEP LOGIN PYPROTECTION PASSPORTS HEALTHC
ARE PAYMENT PAYPHONES PASSPORTS MOBILEPHO
NE DISKENCRYPTION TRANSPORTATION LOGIN AT
M COPYPROTECTIONACCESSCONTROL SIGNATURES
PASSPORTS TRANSPORTATION MOBILEPHONE COPY
PROTECTION LOGIN SIGNATURES PAYMENT ACCES
SCONTROL PAYPHONES HEALTHCARE DISKENCRYPT

Trust the Smartcard

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

Smartcard Stack Summary



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

CLA	INS	P1	P2	L_C	Data
1	1	1	1	0-3	N_C

APDU Responses

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 succesfully 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

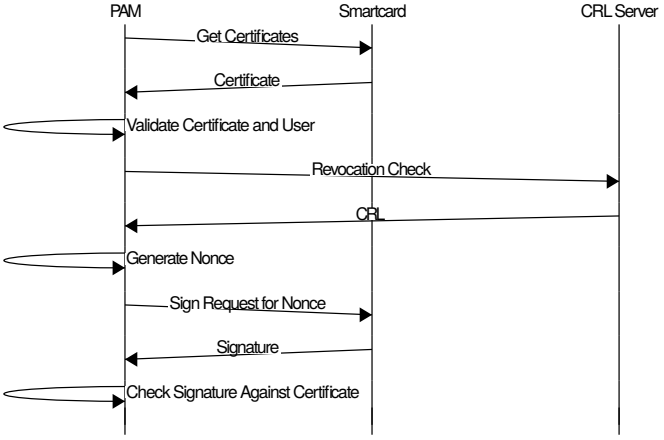
```
LONG WINAPI SCardTransmit(
  SCARDHANDLE         hCard,
  LPCSCARD_IO_REQUEST pioSendPci,
  LPCBYTE             pbSendBuffer,
  DWORD               cbSendLength,
  PSCARD_IO_REQUEST   pioRecvPci,
  LPBYTE              pbRecvBuffer,
  LPDWORD             pcbRecvLength
);
```

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



DNIE

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

DNIE Hardening

```
"file" : {
   "fortify_source" : "no",
   "fortify-able" : "4",
   "pie" : "dso",
   "rpath" : "no",
   "relro" : "partial",
   "fortified" : "0",
   "nx" : "yes",
   "canary" : "no",
   "filename" : "libpkcs11-dnie.so",
}
```

Third Party Code

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

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score
1	CVE-2017-9434	125			2017-06-05	2017-06-13	5.0
Crypto++ (aka cryptopp) through 5.6.5 contains an out-of-bounds read vulnerability in zinflate.cpp in the							
2	CVE-2016-9939	20			2017-01-30	2017-02-07	5.0
Crypto++ (aka cryptopp and libcrypto++) 5.6.4 contained a bug in its ASN.1 BER decoding routine. The li there is not enough content octets in the ASN.1 object, then the function will fail and the memory block w large allocation.							
3	CVE-2016-7544	399			2017-01-30	2017-02-07	5.0
Crypto++ 5.6.4 incorrectly uses Microsoft's stack-based _malloca and _freea functions. The library will re then the wrong pointer could be freed.							
4	CVE-2016-7420	200		+Info	2016-09-16	2016-11-28	4.3
Crypto++ (aka cryptopp) through 5.6.4 does not document the requirement for a compile-time NDEBUG might allow context-dependent attackers to obtain sensitive information by leveraging access to process							
5	CVE-2016-3995	200		+Info	2017-02-13	2017-03-03	5.0
The timing attack protection in Rijndael::Enc::ProcessAndXorBlock and Rijndael::Dec::ProcessAndXorBlo which allows attackers to conduct timing attacks.							

Toying around - DoS

```
> 00 c0 00 00 00            Get Response
< 61 00                     I have another 0 bytes

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

> 00 c0 00 00 00            Get Response
< 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            Read Binary,
< 90 00                     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

Project	# Bugs
libykneomgr	1
OpenSC	Over 9000 ;-)
pam_pkcs11	1
smartcardservices	2
Yubico-Piv	2

No, I did not fuzz the &$#?@! out of it...
but guess which one I fuzzed the most ;-)
Thanks to Frank Morgner for fixing!

Apple Smartcardservices

```
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);
```

OpenSC - CryptoFlex

```
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
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
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 tty3
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/`

Card Definitions

```
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/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

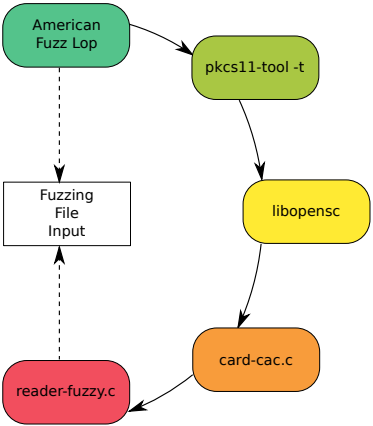
Challenges in fuzzing a protocol

- 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

```
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 (OPENSC_FUZZ_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

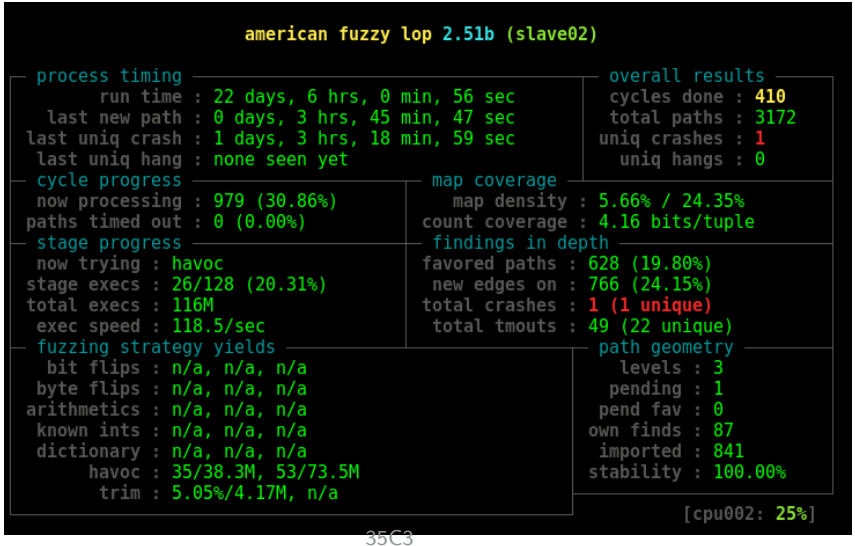
How to fuzz - Winscard 2

- Tavis loadlibrary
- Extended to support Winscard drivers
- Fuzz the windows drivers on linux without all the overhead

Smartcard fuzzing

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

AFL fuzzing


american fuzzy lop 2.51b (slave02)
process timing
run time : 22 days, 6 hrs, 0 min, 56 sec
last new path : 0 days, 3 hrs, 45 min, 47 sec
last uniq crash : 1 days, 3 hrs, 18 min, 59 sec
last uniq hang : none seen yet
overall results
cycles done : 410
total paths : 3172
uniq crashes : 1
uniq hangs : 0
cycle progress
now processing : 979 (30.86%)
paths timed out : 0 (0.00%)
map coverage
map density : 5.66% / 24.35%
count coverage : 4.16 bits/tuple
stage progress
now trying : havoc
stage execs : 26/128 (20.31%)
total execs : 116M
exec speed : 118.5/sec
findings in depth
favored paths : 628 (19.80%)
new edges on : 766 (24.15%)
total crashes : 1 (1 unique)
total tmouts : 49 (22 unique)
fuzzing strategy yields
bit flips : n/a, n/a, n/a
byte flips : n/a, n/a, n/a
arithmetics : n/a, n/a, n/a
known ints : n/a, n/a, n/a
dictionary : n/a, n/a, n/a
havoc : 35/38.3M, 53/73.5M
trim : 5.05%/4.17M, n/a
path geometry
levels : 3
pending : 1
pend fav : 0
own finds : 87
imported : 841
stability : 100.00%
[cpu002: 25%]

Coverage

LCOV - code coverage report

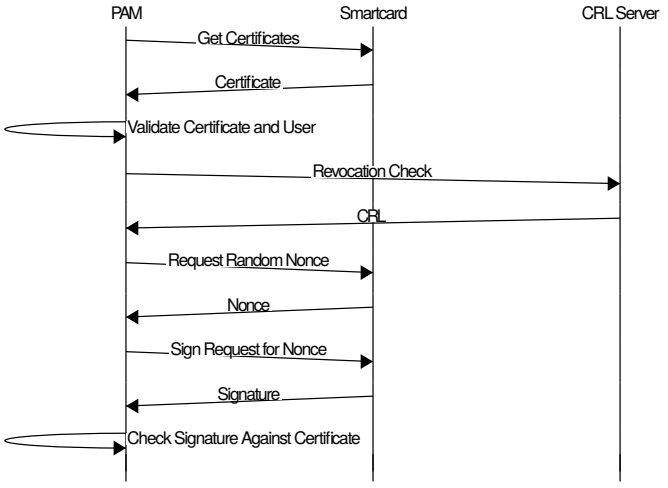
		Hit	Total	Coverage
Current view:	top level - libopensc			
Test:	trace.lcov_info_final	**Lines:**	13213	41611	31.8 %
Date:	2018-08-07 17:54:04	**Functions:**	796	1913	41.6 %

Filename	Line Coverage			Functions		
pkcs15-skey.c		0.0 %	0 / 89	0.0 %	0 / 2
sc-ossl-compat.h		0.0 %	0 / 37	0.0 %	0 / 3
base64.c		0.0 %	0 / 75	0.0 %	0 / 3
ctbcs.c		0.0 %	0 / 123	0.0 %	0 / 3
pkcs15-cache.c		0.0 %	0 / 80	0.0 %	0 / 3
pkcs15-sec.c		0.0 %	0 / 201	0.0 %	0 / 6
padding.c		0.0 %	0 / 120	0.0 %	0 / 7
p15card-helper.c		0.0 %	0 / 176	0.0 %	0 / 10
reader-tr03119.c		0.0 %	0 / 393	0.0 %	0 / 12
card-miocos.c		0.0 %	0 / 220	0.0 %	0 / 14
pkcs15-algo.c		0.0 %	0 / 198	0.0 %	0 / 16
iasecc-sm.c		0.0 %	0 / 333	0.0 %	0 / 17
card-jcop.c		0.0 %	0 / 476	0.0 %	0 / 21
iasecc-sdo.c		0.0 %	0 / 713	0.0 %	0 / 25
cwa14890.c		3.9 %	29 / 745	6.5 %	2 / 31

Coverage

sc.c		**73.4 %**	351 / 478	**77.6 %**	38 / 49
card-belpic.c		**69.3 %**	95 / 137	**77.8 %**	7 / 9
card-cac.c		**70.7 %**	573 / 811	**78.6 %**	33 / 42
pkcs15-openpgp.c		**24.7 %**	39 / 158	**80.0 %**	4 / 5
muscle-filesystem.c		**85.0 %**	102 / 120	**81.8 %**	9 / 11
card-jpki.c		**68.8 %**	139 / 202	**83.3 %**	10 / 12
reader-fuzzy.c		**87.7 %**	93 / 106	**90.9 %**	10 / 11
errors.c		**91.2 %**	31 / 34	**100.0 %**	1 / 1
pkcs15-jpki.c		**94.0 %**	79 / 84	**100.0 %**	2 / 2
simpletlv.c		**100.0 %**	33 / 33	**100.0 %**	2 / 2
ef-gdo.c		**89.4 %**	42 / 47	**100.0 %**	2 / 2
pkcs15-esinit.c		**89.3 %**	25 / 28	**100.0 %**	3 / 3
pkcs15-westcos.c		**69.6 %**	87 / 125	**100.0 %**	3 / 3
ef-atr.c		**93.9 %**	77 / 82	**100.0 %**	3 / 3
gp.c		**100.0 %**	18 / 18	**100.0 %**	3 / 3
pkcs15-atrust-acos.c		**86.0 %**	92 / 107	**100.0 %**	4 / 4
sm.c		**89.0 %**	65 / 73	**100.0 %**	4 / 4
pkcs15-gemsafeV1.c		**94.3 %**	198 / 210	**100.0 %**	8 / 8
pkcs15-tcos.c		**82.5 %**	217 / 263	**100.0 %**	9 / 9
apdu.c		**70.0 %**	219 / 313	**100.0 %**	12 / 12

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 :)

Takeaways / Conclusions

- 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 ;-)