Pentest-Report Nitrokey Storage Hardware 08.2015

Cure53, Dr.-Ing. Dmitry Nedospasov, Dr.-Ing. Mario Heiderich

Index

Introduction

“Nitrokey is an USB key to enable highly secure encryption and signing of emails and data, as well as login to the Web, networks and computers. Other than ordinary software solutions, the secret keys are always stored securely inside the Nitrokey. Their extraction is impossible which makes Nitrokey immune to computer viruses and Trojan horses. The user-chosen PIN and the tamper-proof smart card protect in case of loss and theft. Hardware and software are both available as Open Source to allow verifying the security and integration with other applications.”

From https://www.nitrokey.com/introduction

This penetration test against the Nitrokey Storage hardware was performed by one engineer over the period of 10 days. The test is one of the components within a larger scheme of security assessments, thus complimenting penetration tests of the software employed in the Nitrokey setup. The main scope of this assessment was to evaluate the security of the Nitrokey’s hardware against various offline hardware-based attacks.

The hardware design of the Nitrokey Storage is openly available on GitHub[1]. The EDA tool used for its design is the DesignSpark PCB suite[2], while the micro controller is Atmel AT32UC3A3256S[3]. The platform also includes an OpenGPG-based secure smart card and MicroSD (uSD) card for storage, though both of the cards were out of scope for this

[1] https://github.com/Nitrokey/nitrokey-storage-hardware
[2] https://en.wikipedia.org/wiki/DesignSpark_PCB
[3] http://www.atmel.com/devices/AT32UC3A3256S.aspx

assignment. The Printed Circuit Board (PCB) is a four layer design. The assessment included a review of hardware component placement, as well as a set of physical tests of the targeted devices, primarily focusing on the hardware interface.

The test was carried out in reference to a threat model delivered by the Nitrokey Team. In this threat model the attack scenarios considered vital in the realm of the overall hardware design are specified. These are as follows:

- Malware trying to steal/export private keys while the device is being used on the compromised computer.
- An attacker stealing or finding the (switched off) device, then trying to tamper with the device in order to get access to private keys or encrypted mass-data. This includes sophisticated physical attacks with laboratory equipment and (digital) brute force attacks ("offline").
- Nitrokey protects against Malware attempting to program malicious firmware onto the stick.

In addition, two security assumptions have been added to complete the threat model:

- The Admin PIN and Firmware Password are only used on secure computers for administrative purposes.
- The User PIN can be used on potentially compromised computers.

Note that these specifications are subject to change, so they are relevant first and foremost for this specific assessment. In the future, they may be augmented, changed or extended, for instance with the upcoming hardware revisions and within future firmware versions. In this document we report on the test findings, which have amounted to five hardware-related vulnerabilities and four general weaknesses. In addition, we propose some overall design recommendations.

None of the identified issues were classified to be of critical severity. At the same time, however, three issues were denoted with “High” severity markers since the underlying hardware-related vulnerabilities would allow for a compromise of the Nitrokey threat model.

Scope

The actual Nitrokey Storage Hardware implementation was handed out for testing.

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues identified during the assessment. Note that findings are listed in a chronological order rather than by their degree of severity and impact, which is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *NK-02-00X*) for the purpose of facilitating any future follow-up correspondence.

NK-02-001 JTAG Header accessible on the Nitrokey PCB *(**Critical**)*

The current revision of the PCB layout includes a header exposing the programming interface of the device. An attacker with access to the device's JTAG[4] can perform a "Chip Erase" of the micro controller and subsequently flash a malicious firmware. In combination with the weak security fuse settings, an attacker would be able to flash the device. This would occur without a prior erasing of any sensitive data stored in the non-volatile memory. The "Chip Erase" feature belongs to the Atmel AT32UC3A3256S micro controller family and cannot be disabled.

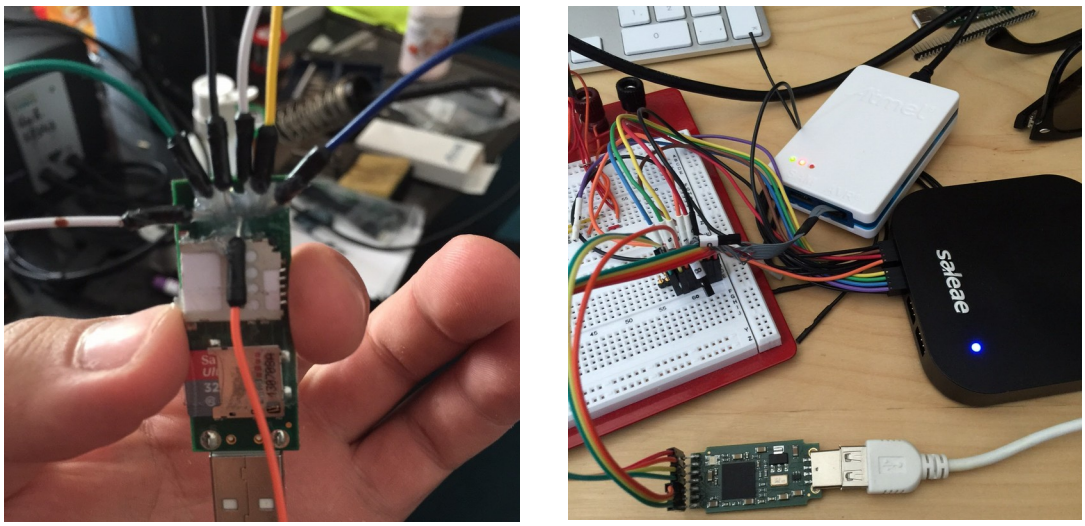

Fig.: JTAG Connector exposed using jumper wires. / JTAG ICE3 Programmer and Saleae Pro 16 Logic Analyzer connected to Nitrokey JTAG Interface via solderless breadboard.

[4] http://www.xjtag.com/support-jtag/what-is-jtag.php

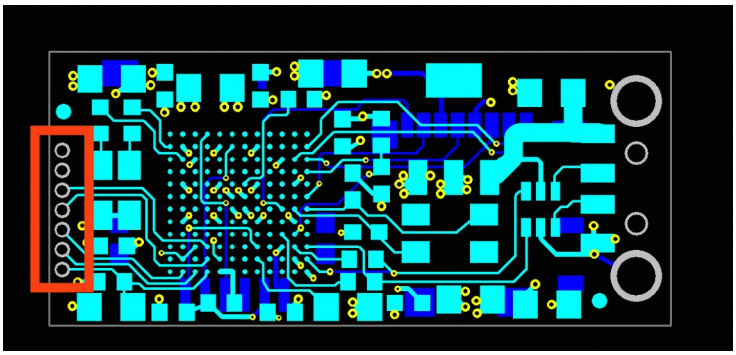

Fig.: Location of the JTAG Header within the design.

The resolution of this issue must take into account the above mentioned fact that the chip erase feature cannot be disabled. Therefore, it is vital to remove the programming interface entirely. The micro controllers used within the design can be flashed either by the chip manufacturer or a distributor. Alternatively a programming jig can be used for flashing the micro controllers on site during manufacturing. Another valid option is granted by physical removal of the programming header during the manufacturing phase. There are two pathways to achieve this. First method is to engage in physical milling away of the specific part of the PCB which contains the header. Second idea relies on placing the header on a portion of the PCB that is separated with a V-groove, meaning it is being snapped or broken off during manufacturing.

Note: This issue was fixed by the Nitrokey maintainers, the fix was verified by Cure53. The fix also affects the issue described in NK-02-003.

NK-02-002 Weak Security Fuse Configuration *(**Critical**)*

The security fuse was not set on the device. Setting the security fuse bit disables the following commands:[5]

- Write General-Purpose Fuse Bit (WGPB) to BOOTPROT or EPFL fuses
- Erase General-Purpose Fuse Bit (EGPB) to BOOTPROT or EPFL fuses
- Program General-Purpose Fuse Byte (PGPFB) of fuse byte 2
- Erase All General-Purpose Fuses (EAGPF)

Evidently the security fuse bit enforces the overall security configuration of the device. If it is not set, a malicious attacker can arbitrarily change the security configuration of the device. Subsequently, the attacker would be able to flash malicious firmware to the device. If the security bit is not set, then the device's fuse configuration can be changed arbitrarily by the attacker. This includes disabling the Brown Out[6] Detection (BOD) circuit and removing flash region lock bits. Eventually an attacker would be able to flash a

[5] Atmel "AT32UC3A3/A4 Series Complete" Page 137
[6] https://en.wikipedia.org/wiki/Brownout_%28electricity%29

malicious firmware without performing a chip erase, i.e. affecting all of the contents of the device.

It is recommended to set the security bit during manufacturing. Additionally, the flash memory regions should be locked in order to be prevented from being overwritten by the firmware.

Note: This issue was fixed by the Nitrokey maintainers, the fix was verified by Cure53.

NK-02-003 Security Relevant Signals Routed On Surface Layers *(**Critical**)*

The current Nitrokey hardware design exposes security-relevant signals on the surface layers. An attacker with sufficient equipment would henceforth be allowed to interface to these signals. The following signals were identified on the top layer of the PCB:

- TMS - Test Mode Select, part of the JTAG programming interface
- TDO - Test Data Out, part of the JTAG programming interface
- TDI - Test Data In, part of the JTAG programming interface
- TCK - Test Clock, part of the JTAG programming interface
- NRST - Reset signal for the system, also part of the JTAG programming interface

Exposed signals on either the top or the bottom signal layer are easily accessible. An attacker with access to soldering tools would be able to contact the signals on the surface PCB layers by simply removing the solder mask.

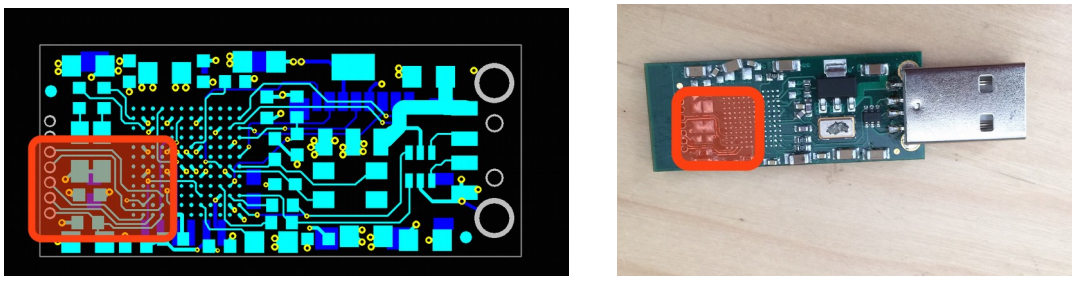

Fig.: Security-Relevant Signals in the top layer of the design / Security-Relevant Signals exposed using a scalpel. Note that several components were desoldered.

Let us reiterate that a consequence would be that an attacker would become able to interface to the specified signals. In case of the JTAG signals (TMS, TDO, TDI, TCK, NRST), an attacker would be able to program the device. Further, with access to the NRST signal, an attacker would gain a capacity to dynamically and reliably reset the device at arbitrary points in time. Though it was left out of scope for this hardware assessment, the reset signal is essential for mitigating certain types of countermeasures implemented in software and thus should not be easily accessible either.

It is recommended to route all security-relevant signals on internal signal layers. Since the AT32UC3A3256S utilizes a BGA package, security relevant signals should utilize blind vias to interface to the micro controller, ensuring that they are not exposed on the opposite side of the PCB. Any other vias should be of a buried type and ensure the signals remain on internal layers.

Note: This issue was addressed in full by the Nitrokey maintainers. The mitigation for this problem is related to the fixes described for NK-02-001.

NK-02-004 Microcontroller Contains DFU bootloader *(Low)*

The AT32UC3A3256S micro controllers come preprogrammed with a Device Firmware Update (DFU) bootloader capable of flashing the device. The bootloader itself was not evaluated within the scope of this assessment. Moreover, the *ISP_FORCE* and *ISP_IO_COND_EN* fuse bits were set to 0, ensuring that the DFU was disabled. However, although the DFU is protected by fuse bits, it may be possible to gain control over the execution during power up. This could potentially be done by utilizing techniques like fault injection. Following this scenario the attacker would be able to program the device next.

It is recommended to ensure that the bootloader is overwritten in the program memory of the micro controller during the programming of the device.

Note: This issue has been discussed with the team. It's been agreed to that this problem is not going to be addressed in the near future and that the recommended changes are seen as an optional enhancement for later releases. The mentioned security bit was set, mitigating this issue in parts.

NK-02-005 Brown Out Detection Not Enabled *(High)*

The BOD fuse bits are not set on the Nitrokey. The AT32UC3A3256S contains multiple Brown Out Detectors (BOD).[7] These can be used to detect abnormalities in the supply voltage. Tampering with the supply voltage is commonly performed to inject faults during execution. Enabling the BOD ensures that the system resets upon detecting a brown out condition. This makes the analysis more difficult for the attacker.

[7] Atmel "AT32UC3A3/A4 Series Complete" Page 147

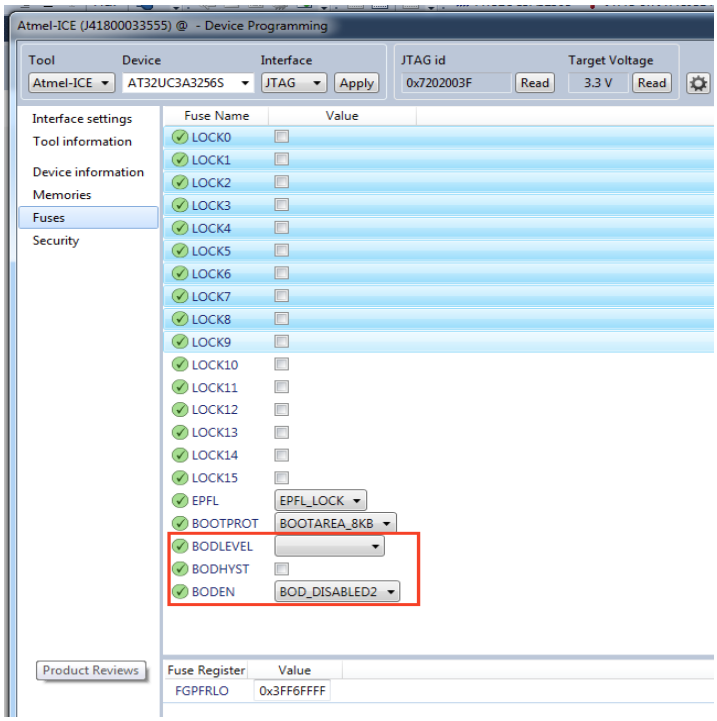

Fig.: Atmel Device Programming GUI. Note that the BOD is disabled.

It is recommended for the BOD to be enabled in hardware by setting the BODEN fuses to 0x1. The result of this operation will be "BOD enabled, BOD reset enabled". Because the Nitrokey is powered over USB, the BODLEVEL can be set to a narrow operating band. Moreover, the BOD33 should be configured in software for the purpose of monitoring supply voltage and initiating device resets.

Note: This issue was fixed by the Nitrokey maintainers, the fix was verified by Cure53.

NK-02-006 Micro SD and Smartcard Slots lack ejection switch *(High)*

Malicious attacks may utilize removal of either the SD or Smartcard for further analysis. Similarly, they can take advantage of a malicious card that may be inserted. The current design of the Nitrokey is incapable of detecting that a card has been removed. It is recommended for the next design revision to utilize card slots that include ejection switches. The current hardware design has sufficient GPIO pins remaining on the AT32UC3A3256S, and thus could be used for connecting additional ejection switches. An ejection of a card should result in a security reset on the Nitrokey.

Note: It was agreed with the maintainers of the project that this issue isn't covered by the currently communicated threat model and will therefore be closed as 'wontfix' for now.

NK-02-007 Current Design Lacks Tamper Switches *(Medium)*

The current revision of the hardware lacks tamper switches. As a result the device is incapable of detecting that that the enclosure has been tampered with or removed. It is recommended that the enclosure includes tamper switches. The current hardware design has sufficient GPIO pins remaining on the AT32UC3A3256S. Any additional tamper switches can be connected to the remaining GPIO. Upon detecting a tamper even, the device should perform a security reset.

Note: This issue has been discussed with the team. It's been agreed to that this problem is not going to be addressed in the near future and that the recommended changes are seen as an optional enhancement for later releases.

NK-02-008 No Offline Tampering Detection *(Medium)*

The current revision of the hardware lacks any offline tampering detection. One idea is that tamper switches could be integrated into the plastic case of the Nitrokey. In its current state Nitrokey is powered off when unplugged. However, a battery or a “supercap” could be utilized to ensure that the device can enter a lower power state. While in a low power state, the device would still be capable of detecting I/O interrupts from the tamper switches. Moreover, sensitive data could be stored in volatile memory. A security reset can be used to erase the contents of the volatile memory, rendering the device inoperable until restored.

The current revision of the Nitrokey enclosure and the density of the PCB layout provide sufficient space for adding extra components. It is recommended that a low-power solution for powering the device becomes integrated to guarantee that the device is powered, even when it is disconnected from the PC. A tamper event should render the device inoperable by destroying contents within the device's volatile memory. A restore function can be optionally implemented with the goal of restoring the device to a nominal operational state in mind.

Note: This issue was not yet addressed in full and is being treated an a possible enhancement for future versions.

NK-02-009 Insufficient Design Density *(Low)*

The use of rather large components means that the overall design density is low. Smaller components make the design harder to handle and signify that it is harder to interface to the device. Furthermore, an increased density also eliminates the possibility of contacting buried signals within the device's internal signal layers.

It is recommended to employ the smallest possible SMD components. The use of arrays for passive components can also help to increase the overall device density. A voltage regulator with a smaller footprint should be selected. Passive components should be placed on both sides of the Nitrokey PCB.

Note: This issue has been discussed with the team and confirmed to be generally valid. However, given the complexity and system stability paired with the rather low severity, an implementation of the fix has not been scheduled yet.

Conclusion

The Nitrokey has a lot of potential. Several key design decisions taken along the way have ensured that the overall platform design is made to withstand a wide range of attacks. In particular, these include the use of a secure smart card family in combination with a general-purpose Atmel micro controller. The latter also notably provides sufficient fuse settings and an AES engine. Although the platform may still be susceptible to online attacks, such as Side Channel Analysis techniques, these were not part of the scope for this assessment. Most importantly, the hardware architecture itself makes an impression of being sufficiently resilient against all forms of *non-invasive* offline attacks.

Unfortunately, at the time of testing, the current hardware design was not resilient against invasive offline hardware attacks. This is primarily due to the fact that the JTAG interface is fully exposed. The weak fuse settings also contribute to this less than stellar evaluation. As a result, it is trivially easy for an attacker to flash a malicious firmware onto the device without first erasing the contents of the device's non-volatile memory. This means that an attacker would succeed in flashing a firmware capable of extracting any sensitive data stored on the device. Moreover, an attacker could flash a malicious firmware capable of stealing cryptographic keys or passwords from the user by mimicking normal behavior. Such a firmware could similarly implement a function to exfiltrate any secret user data, completely compromising the privacy and security of the user in effect.

The tested hardware design required a major revision. The biggest constraint of the design was the fact that the chosen micro controller family, the AT32UC3A3256S, retains a "Chip Erase" feature, which can be used to erase the device and render it programmable, independent of the security fuse bit configuration. Nevertheless, this family is capable of preventing sensitive data stored in the non-volatile memory from being read out. For this reason, it is critical that the security fuse bit is set on the AT32UC3A3256S. Next, it is essential that the JTAG interface remains not exposed on the final design, namely the one that enters the field. For development purposes a separate design may be used. Alternatively, the design can be flashed via a DFU bootloader during development, which can then be removed in the firmware of the devices that are shipped. Preferably, the JTAG interface should be omitted entirely and all of the security-relevant signals should be left unrouted. The AT32UC3A3256S should be pre-programmed prior to assembly. This can be performed by flashing the devices on-site, either by using a programming jig, or by employing a service that flashes the micro controllers before they are sent to manufacturing. This service is offered by most major component distributors. Alternatively, the design could utilize a JTAG header, yet that should be physically removed from the PCB after the device has been programmed and configured.

In all cases, it is essential that security-relevant signals, and the signals of the programming interface in particular, are consistently routed on the internal signal layers of the PCB. This will significantly increase the amount of effort required for the attacker to expose the signals in question. Ideally, however, all security-relevant signals should be left unrouted as they are currently only used by the programming interface. A final possibility is to use the DFU bootloader to flash the device and configure the fuses accordingly. The DFU bootloader can then be overwritten, effectively making sure that the device can no longer be programmed.

Note though, that at the time of writing this final report, all issues spotted in the actual Nitrokey Storage implementation have been reported to the project maintainers and those that were considered actionable have been fixed promptly and quickly. All fixes have been reviewed and verified as working by the Cure53 team.

Cure53 would like to thank Jan Suhr and the entire Nitrokey Team for this interesting project as well as their continuously good support and assistance during this assignment.

We would like to further express our gratitude to the Open Technology Fund in Washington D.C., USA, for generously funding this and other penetration test projects and enabling us to publish the results.