Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Review-Report Polychain Laptop Security 09.-10.2020

Cure53, Dr.-Ing. M. Heiderich, M. Wege, L. Merino, MSc. C. Reitter

Index

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Introduction

"The emergence of bitcoin and subsequent blockchain technologies has generated a new digital asset class in which scarcity is based on mathematical properties. Through cryptographic verification and game-theoretic equilibrium, blockchain-based digital assets can be created, issued, and transmitted using software. Polychain is an investment firm committed to exceptional returns for investors through actively managed portfolios of these blockchain assets."

From https://polychain.capital/

This report describes a rather unusual assignment tackling security of the specialized device developed by Polychain LP. Carried out by Cure53 in September and October 2020, this assignment focused on the very specialized secure laptop setup.

To give some context, the project had a very clear focus in that Polychain LP sought to find out whether the sources, configurations and surrounding aspects of their laptop device and its setup are secure enough in the context of being used for security-demanding operations. In other words, the laptop devices would be deployed for high-risk processes, such as transaction signing and similar. Before the assessments began, Cure53 was thoroughly briefed about expectations, use-cases planned for the laptops, the running software, as well as possible threats and threat actors' capabilities. A long Q&A session was held prior to the quote being issued, so as to make sure that all involved parties are clear on the objectives and procedures for this assessment.

In terms of resources, the Cure53 team assembled for this task consisted of several hand-picked consultants who spent a total of twenty days conducting this work. It should be clarified that the team was given a budget to order special hardware needed for this project, namely the "Librem 15 v4" laptops from Purism, SPC. All hands-on testing was executed on those laptops, while standard code auditing took place on the consultants' setup as usual. Besides hardware, the team could leverage access to several code repositories hosting secure boot-loader code. Auditing said code and relevant dependencies was one of the main tasks in this project.

For better structuring, the project was then split into three separate work packages (WPs), namely:

- **WP1**: Security Tests & Code Audits against "*Heads*" & "*Airgap*" Sources
- **WP2**: Security Tests & Code Audits against relevant project dependencies
- **WP3**: Security Assessments against build- and release-system as well as system- & hardware-specific controls present on reference laptop "Librem 15 v4".

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Given the extensive preparatory phase, the project could start on time and mostly progressed well. Communications during this test took place via the Matrix messenger protocol of the Element application. A dedicated channel was created and populated with all involved personnel from Polychain and Cure53. The discussions and exchanges were quite intense and extensive. They were very helpful because, at the beginning of the project, a lot of time had to be spent to successfully complete installations of the software in scope on the tested hardware. The task necessitated substantial patching until a satisfying state was reached and the actual testing became possible.

It needs to be emphasized that the Polychain team was essential for facilitating good progress of this Cure53 examination. They were especially forthcoming in the early phase of the project. Once the initial hurdles were successfully tackled together, additional Cure53 testers joined the project and started working on the scope, from then on moving forward without noteworthy hold-ups. To reiterate, the long and thorough preparation phase and the mutual work on accomplishing a working setup cannot be disregarded in relation to findings.

As for the results, the project yielded eighteen discoveries. The vast majority of them - namely fifteen - should be considered as general weaknesses and miscellaneous issues that largely deal with hardening recommendations and have low or negligible threat capacities. There is one exception, however, as exposure of the attack surface within a kernel driver received a *High*-severity score. All three spotted security vulnerabilities were rated as *Medium.* Ahead of conclusions, it can be said that this represents a rather positive outcome with regard to the security picture of the tested and audited software and hardware combo.

The report will now shed some light on scope and test setup. Findings will be ordered first by groups of vulnerabilities and weaknesses, and then chronologically within the categories. Each finding will be accompanied by a technical description, a PoC where possible, as well as mitigation or fix advice. After that, the report will close with a conclusion, in which the Cure53 team will elaborate on the general impressions gained over the course of this test and audit. In this section, a dedicated attempt will be made at identifying where bug patterns worth looking into are located. Tailored hardening advice is also incorporated to the final section in order to facilitate further amelioration of the security standing of the Polychain laptop devices and the surrounding compound.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Security Reviews & Assessments against Polychain Laptop Setup**
  - **WP1**: Security Tests & Code Audits against *“Heads”* & *“Airgap”* Sources
    - *Access to the auditable sources granted; inspection and thorough audit of the source for issues potentially leading to vulnerabilities.*
    - *Core focus will be directed to issues leading to system compromise via user-controlled data, rogue hardware, faulty driver software or any other attack vector that manages to delegate potentially harmful user-input to any form of risky sink or internal device storage.*
  - **WP2**: Security Tests & Code Audits against relevant project dependencies
    - *The list of dependencies inspected; security audits against all parts of the dependencies that might contribute to the weakening or exploitation of the software or the hardware in scope for this audit (items mentioned in WP1 as well as the reference hardware)*
  - **WP3**: Security Assessments against build- and release-system as well as system- & hardware-specific controls present on reference laptop “Librem 15 v4”
    - *A review of the build pipeline, code version system, deployment, roll-out and other aspects of the utilized build and release systems in scope.*
    - *Further, an inspection of the reference hardware will be performed, checking said Librem 15 laptops for possible weaknesses relating to exposed hardware ports and connectors, pre-installed drivers, possible physical weaknesses and other means for an attacker to gain illegitimate control over the reliability and security controls offered by the solution in scope.*

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *POL-01-001*) for the purpose of facilitating any future follow-up correspondence.

POL-01-006 WP1: *git checkout <hash>* does not ensure integrity *(Medium)*

It was found that several scripts (such as *scripts/fetch* and *config/container/Dockerfile)* use *git* repositories, combining *git clone* with *git checkout <commit-id>* to set the repository contents to a well-known state. It has been observed that *git checkout* semantics are ambiguous and allow either checking out a given commit ID or a branch name, giving priority to verifying a branch when its name matches an already present commit ID.

This enables a source code supply-chain attack, where a malicious repository administrator might create an arbitrary number of branches in their repositories. Those could be named after well-known commit IDs to deliver modified source files with malicious behavior. A user cloning a repository and checking out a commit ID afterwards would receive the malicious copy of the source files instead of the well-known and expected ones. Additionally, the attack might succeed unnoticed in the absence of additional code integrity checks.

It has been observed that GitHub rejects "*branch or tag names consisting of 40 hex characters*" and all the repositories used in the audited *Dockerfile* are hosted on GitHub, making the attack not possible at the moment. Nevertheless, it is recommended not to rely on the GitHub mitigation and deploy additional checks.

Commands to reproduce:

```
$ cat setup.sh
nice-setup-command
$ git rev-parse HEAD
2ef5bba9853fc675d4ca0835c93adc49852a120c
$ echo 'super-evil-command' > setup.sh
$ git add setup.sh
$ git commit -m'nobody will notice this'
[master b40cd22] nobody will notice this
 1 file changed, 1 insertion(+), 1 deletion(-)
$ git branch 2ef5bba9853fc675d4ca0835c93adc49852a120c
$ git checkout 2ef5bba9853fc675d4ca0835c93adc49852a120c
warning: refname '2ef5bba9853fc675d4ca0835c93adc49852a120c' is ambiguous.
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
Git normally never creates a ref that ends with 40 hex characters
because it will be ignored when you just specify 40-hex. These refs
may be created by mistake. For example,

  git switch -c $br $(git rev-parse ...)

where "$br" is somehow empty and a 40-hex ref is created. Please
examine these refs and maybe delete them. Turn this message off by
running "git config advice.objectNameWarning false"
Switched to branch '2ef5bba9853fc675d4ca0835c93adc49852a120c'
$ echo $?
0
$ git rev-parse HEAD
b40cd22be259f46f57ff6e6369518b83f9ef3884
$ cat setup.sh
super-evil-command
```

As observed in the log, a branch was named after a known commit ID, and the *git checkout <commit-ID>* operation has switched to the malicious branch instead of the actual commit.

After checking out a particular commit by commit ID, a mitigation should take the form of a check that the right commit was landed. This should be done by performing a *git rev-parse HEAD* and matching the returned commit ID with the expected one.

POL-01-007 WP1: Wireless USB device attack *(Medium)*

It was observed that the usage of an external USB WiFi dongle potentially opens the door to remote attacks. Depending on other USB characteristics, an attached device may allow bidirectional RF communication between the laptop and the outside world to defeat the *Airgap* properties. The current *AirgapOS buildroot* kernel has a full set of device drivers including USB wireless devices, so it would likely be possible to establish actual IP network communications with the laptop.

It is recommended that, in order to eliminate the risk of malicious and rogue USB devices being plugged into and used by the *Airgap* system, the USB driver support in the Linux kernel should be completely removed. Additionally, the maintainers are encouraged to remove networking drivers from the *Airgap* build.

The client has acknowledged this finding and the underlying issue of exfiltrating data via USB devices regardless of the physical transmission or storage properties. It has been indicated that this particular aspect will be resolved soon. Their main countermeasure is dual custody for those devices, as well as treating them as somewhat compromised in the threat model.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

POL-01-008 WP1: Emulated keyboard USB device attack *(Medium)*

It was found that, similar to POL-01-007, any attached USB device can be enumerated as an additional Human Interface Device (HID). This opens the door to a range of well-known emulated keyboard attacks that work by injecting keystrokes in the Airgap laptop. One example could involve inserting additional command line parameters or commands when interacting with a *shell.*

Notably, legitimate devices like the YubiKey series 5 devices use a similar HID enumeration functionality to automatically type certain keys upon request in their default configuration.

> *The HID Keyboard interface passes output from the YubiKey to the host system as keystrokes from a virtual keyboard, and can use the HID Keyboard channel to communicate back to the YubiKey.*

It is recommended to take measures to severely restrict or completely remove HID support in the host system for USB devices which are inserted during the key ceremony. According to a discussion with the client, the built-in laptop keyboard of the target Librem hardware is not connected via USB HID and no legitimate HID devices are required after boot. Therefore, one solution could be to remove or block the relevant HID Linux kernel driver to prevent usage. Alternatively, legitimate HID devices can be accepted individually after insertion.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

POL-01-001 WP3: Attack surface in compromised USB devices *(Medium)*

It was observed that the usage of external USB devices such as the Nitrokey (for trusted boot verification) and USB-based GPG smartcard devices (for key ceremony operations) represent a significant attack vector towards the trusted boot system, operating system and userland software.

Compromised USB devices, which are introduced to the key ceremony by either a malicious insider or a supply chain attack, provide an opportunity to trigger software problems in a wide range of subsystems unless extensive measures are taken to prevent this. While no serious problems in the analyzed software (such as the nitrokey-hotp-verification tool) have been found, it is recommended to treat the physical USB access as one of the main attack vectors.

It is recommended to undertake additional efforts to:

- Reduce the array of software reachable via USB
- Harden the essential software which is functionally required for USB interactions
- Require manual approval before new USB devices are enumerated post-boot
- Detect physical changes of essential USB devices via tamper-evidence

It is further recommended to look into the question of making the Nitrokey more trustworthy and tamper-evident. Previous vulnerabilities affecting similar Nitrokey hardware variants suggest that partial cloning of an original device after destructive physical access should be taken into account when deciding on the verification steps. Depending on the attack method, such a tamper check would need to detect the presence of additional circuitry or modified firmware, which is unfortunately somewhat difficult to achieve during a key ceremony. See POL-01-005 for additional recommendations.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

POL-01-002 WP3: Insufficient *Docker seccomp* sandboxing *(Low)*

It was found that the Docker containers for the reproducible build verification steps are started with `--security-opt seccomp=unconfined` which effectively removes seccomp sandboxing from the host kernel.

Complete removal of *seccomp* limits is not advised. It is recommended to isolate the required permissions and instead use a custom *seccomp* profile, if the default one is not found permissive enough for the required build operations.

POL-01-003 WP3: Redundant CVE information in automatic audit report *(Info)*

During the analysis of the audit step within the build environment, it was observed that over half of the listed CVEs in *container_package_cves.txt* were identical duplicates. This behavior is caused by the *debsecan* report generation program.

It is recommended to evaluate the built-in filter parameters of *debsecan,* such as `--only-fixed` along with some post-processing on the output listing. This will help to reduce manual work required to analyze the CVE reports.

POL-01-004 WP1: Build container allows expired package release signing *(Low)*

It was found that the Airgap build container, as configured in *config/container/Dockerfile*, globally disables the *Acquire::Check-Valid-Until APT* flag, thus permitting replay attacks against the otherwise expired APT archive signatures. In combination with the fact that plain HTTP connections are used to connect to the repository mirrors, this enables a whole range of downgrade and APT attacks via mirror administration and network MitM.

Affected Files:
apt.conf

```
Acquire::Check-Valid-Until "false";
```

sources.list

```
deb http://deb.debian.org/debian buster main
deb http://snapshot.debian.org/archive/debian/20200910T000000Z buster main
deb http://security.debian.org/debian-security buster/updates main
deb http://snapshot.debian.org/archive/debian-security/20200910T000000Z
buster/updates main
deb http://deb.debian.org/debian buster-updates main
deb http://snapshot.debian.org/archive/debian/20200910T000000Z buster-updates
main
```

It is assumed the aforementioned flag has been disabled to keep snapshot mirrors working under extended circumstances. Thus, it is recommended to apply the *Check-Valid-Until* option to any *sources.list* entries requiring it. This should be done together

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

with introducing an additional mechanism which preserves integrity of the downloaded packages (e.g., signed hashes per package or similar means). It is additionally highly recommended to upgrade the mirror connections to HTTPS.

The client acknowledged that the current deployment is intentional and aimed at keeping apt archive snapshots functional. It is, however, advised to implement at least some sort of release key validity restrictions. One approach would be to only allow keys that were actually still valid on the snapshot date. Another approach would be to add a custom verification step via direct hash comparisons, though this depends heavily on implementation details. See *airgap/scripts/update-packages* for dynamic apt source configuration.

POL-01-005 WP3: One-sided single-device root of trust verification *(Low)*

It was observed that the HOTP verification mechanism allows a verification of the laptop-provided boot state via the Nitrokey USB key, which is a powerful protection against attacks on the BIOS Root-of-Trust. To our knowledge, however, there is currently no designated key ceremony mechanism in place to cryptographically verify the authenticity of the Nitrokey or allow the redundant usage of multiple Nitrokey devices for extended Root-of-Trust verification.

A malicious insider could, therefore, replace the Nitrokey device with a functional clone that accepts a different set of HOTP codes as “verified”. In an extreme form, this clone may simply accept all HOTP codes as “verified”, regardless of the actual laptop boot security state. Combined with other attacks, this may eventually signify full circumvention of the hardware-assisted root of trust verification.

It is recommended to apply as many of the following countermeasures as possible:

- Make the paired Nitrokey device harder to replace covertly during the ceremony by attaching it to a larger physical object.
- Allow visual detection of a replaced device by adding custom seals or markers that are difficult to clone. Inspect them early during the ceremony.
- Use additional cryptographic protocols of the Nitrokey, such as OpenGPG support, to verify authenticity of the device.
- Use the *Heads’* TPMTOTP verification mechanism to establish a second verification channel on another trusted verification device. Special care should be taken that this second device conforms to the existing *Airgap* requirements.

The proposed countermeasures have already been discussed extensively with the client.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

POL-01-009 WP3: Mandatory Access Control mechanism missing *(Low)*

It was found that the analyzed version of the *Airgap buildroot* system is not utilizing any sort of Mandatory Access Control (MAC) mechanism, such as SELinux[1] or AppArmor[2]. These are normally used to restrict capabilities and permissions of programs running inside the *Airgap* system and, thus, achieve a strict isolation and reduction of attack surface One of the benefits of using AppArmor over SELinux is that AppArmor controls access to programs rather than to users via Linux kernel-loaded profiles.

Besides applying either SELinux or AppArmor, an additional layer of security could be achieved by using *seccomp*[3] filtering, effectively limiting the number of system calls an application is allowed to invoke. It is recommended to utilize SELinux / AppArmor as well as *seccomp* filtering in order to reduce the potential attack surface and properly sandbox applications running inside the *Airgap* system.

POL-01-010 WP3: User-privilege separation in build environment missing *(Low)*

It was observed that the audited version of the *Airgap* build system contains the following Docker configuration in *config/container/Dockerfile:*

```
## Create build user with sudo privs
RUN useradd -G plugdev,sudo -ms /bin/bash build \
      && chown -R build:build /home/build \
      && echo '%sudo ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers
```

This essentially grants the default *build* user-account full root permissions within the container. Malicious code running at build time due to some compromised build dependency could leverage the missing privilege separation towards a container escape.

It is, therefore, recommended to implement stronger *sudo* restrictions or completely remove *sudo* permissions from the *build* user, if possible. The unconstrained *sudo* functionality described above was confirmed by the client and flagged for removal in the latest revision of *Airgap.*

[1] https://wiki.centos.org/HowTos/SELinux
[2] https://wiki.ubuntu.com/AppArmor
[3] https://man7.org/linux/man-pages/man2/seccomp.2.html

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

POL-01-011 WP3: Reducing attack surface of *Airgap* OS kernel driver *(High)*

It was found that the analyzed version of the *Airgap buildroot* system contains a generic Linux build configuration with a wide variety of Linux device drivers, e.g. networking drivers, among others. Consequently, the system has a large attack surface with regard to drivers that may be loaded, depending on external device enumeration or made available to malicious Linux userland software. The following screenshot provides a list of loaded kernel drivers as modules.

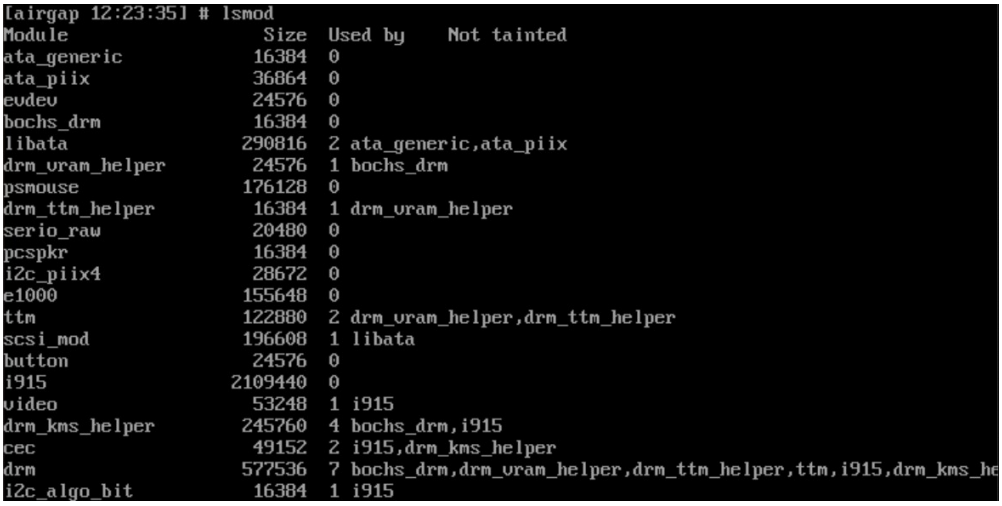

Fig.: lsmod output on Airgap OS

It has to be noted that many additional drivers are located inside the *Airgap* build and could potentially be loaded by the *Airgap* system. The above list contains all loaded kernel modules, including the ones which are dynamically loaded, meaning not only the drivers which are built statically into the kernel image.

It is recommended to evaluate stronger restrictions for the kernel build profile, for example using items based on the *Heads* configuration for Librem devices. Besides security improvements, this should also largely accelerate the build process for the main system.

This liberal configuration of the available modules for the current setup has been acknowledged by the client. Apparently, a much more restricted version was already available with previous releases of *Airgap* but was temporarily relaxed for functional reasons. A maximally restricted version is being worked on by the client at present.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

POL-01-012 WP3: Re-evaluate lax IOMMU configuration for *Heads* *(Low)*

It was observed that the audited version of the *Airgap* system configures *Heads* to boot with a reduced IO memory management unit (IOMMU) safety state via *intel_iommu=igfx_off*.

```
build/coreboot-4.12/librem15v4/.config
CONFIG_LINUX_COMMAND_LINE="intel_iommu=igfx_off quiet loglevel=3"
```

This allows the graphics' subsystem to perform insecure device memory accesses during boot. It is recommended to evaluate if a relaxed memory access configuration is still required, even though Cure53 is unaware of any direct security-related consequences. It has to be noted that the main Linux starts without the *igfx_off* flag.

The described relaxation of memory access restrictions was apparently necessary to work around some functional graphics issues in the past and is already being re-evaluated by the client.

POL-01-013 WP3: Mitigating physical power analysis *(Info)*

It was found that power analysis attacks via high-frequency observations of the laptop power consumption represent a risk for the confidentiality of the cryptographic operations on the *Airgap* laptop. Special electronic circuitry in the power supply path or with connection to essential voltage rails may covertly sample, process, record or transmit such trace data. In the worst-case scenario, this may allow the eventual reconstruction of cryptographic keys or similar cryptographic secrets by an adversarial third-party.

A particularly interesting place to hide electronic observation equipment during the key ceremony could be the standard AC-DC external laptop power supply. It is difficult to inspect non-destructively, inconspicuously and interchangeably. It also provides extensive options in terms of power budget and physical volume for the power analysis equipment when compared to other implant locations within cables, laptop subsystems or USB devices.

Besides power analysis, the power supply could also contain other surveillance equipment. It is therefore not recommended to leave the power supply plugged in or in close proximity to the Airgap laptop during essential cryptographic operations. Additionally, electrical connection to the AC power grid may allow some emissions to be traceable from a wider distance, even with a power supply that has not been tampered with. Due to concerns about the trustworthiness of the regular internal laptop battery in high-assurance setups (see POL-01-014), it is recommended to consider a custom

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

battery-based low-voltage DC supply, which is able to power the laptop via the external DC jack for the duration of the key ceremony.

The battery design could be based on:

- Replaceable lithium battery cells that can be safely investigated individually
- A low-efficiency, low-noise voltage regulator without high frequency operations
- The omission of programmable or otherwise complex electronic components
- Extensive RC/LC power supply filtering stages

Ideally, this battery should exhibit at least acceptable runtime characteristics and overall reliability while making any modifications difficult to hide.

POL-01-014 WP3: Removal of internal lithium battery *(Info)*

It was observed that modern laptop batteries include complex management and monitoring functionality via an embedded microcontroller, which is typically connected via SMBus. The required firmware and hardware are usually not well-documented or in any way verifiable. Other laptop designs with strong transparency requirements, such as Novena, have opted for a custom battery controller solution to avoid this limitation. To our knowledge, the targeted Librem 15v4 laptop uses a battery controller that is manufactured by a third-party and not fully controlled or open:

```
Device: /org/freedesktop/UPower/devices/battery_BAT
 native-path:          BAT
 vendor:               TPS
 model:                S10
 power supply:         yes
[...]
 has history:          yes
 has statistics:       yes
```

This internal subsystem runs custom code, has access to (limited) power measurements and may present a hidden persistent storage device towards the main system via SMBus interaction. The *Airgap* laptop security design mandates the removal of all internally persistent storage devices to prevent code or data that persists across reboots.

It is advised to avoid this risk by physically removing the internal battery from the laptop for high-assurance use-cases and supplying power via another power source from the external DC jack that has no data connection. Note that POL-01-013 should be taken into account when selecting a suitable external power supply.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

POL-01-015 WP1: Missing security hardening flags in *buildroot* config. *(Low)*

The analyzed version of the *Airgap buildroot* system's default configuration has been reviewed according to security best practices. Cure53 identified that the *buildroot* configuration lacks various security hardening flags / features.

The following list of configuration options are missing or not configured properly:

- `BR2_RELRO_FULL`
- `BR2_FORTIFY_SOURCE_2` (*Fortify Source* needs a *glibc* toolchain and optimization)
- `BR2_SSP_ALL` (*Stack Smashing Protection* needs a toolchain with SSP)

It is recommended to consider hardening the buildroot's default configuration as an important step in elevating the general security posture of *Airgap*. *Elinux.org*[4] is an excellent online reference, describing possible hardening options. The client has already been made aware of these flags ahead of time and requested to document them here as a reference for future additions to the *Airgap* system.

POL-01-016 WP1: Adding support for RAM encryption *(Info)*

Modern Intel chipsets come with hardware extensions, named Total Memory Encryption (TME)[5], providing the capability to encrypt the entirety of the physical memory of a system with a single encryption key generated by the CPU on every boot of the system. Multi-Key Total Memory Encryption (MKTME) builds on top of TME and adds support for multiple encryption keys. While TME provides robust mitigations against single-read physical attacks, such as physically removing a DIMM and inspecting its contents, MKTME offers further mitigations, as described within the proposed patch for adding MKTME to the Linux kernel[6].

Enabling TME / MKTME requires CPU support as well as changes within *coreboot/ Heads*. In particular, the compile time option named "INTEL_TME"[7] must be set. Adding support for MKTME to the Linux kernel is still an ongoing process and several patches have already been proposed. It is, nevertheless, recommended to consider these, even though no reference implementations of TME / MKTME within one of the well-established Linux distributions existed at the time of writing. Still, it would be beneficial to eventually add RAM encryption as an additional feature in *Airgap*, maybe optionally allowing *Airgap* users to selectively enable this additional layer of security.

[4] https://elinux.org/Buildroot:SecurityHardening
[5] https://software.intel.com/sites/default/files/managed/a5/16/Multi-Key-...pec.pdf
[6] https://lwn.net/ml/linux-kernel/20190731150813.26289-1-kirill.shutemov@linux.intel.com/
[7] https://coreboot.org/status/kconfig-options.html

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

The subject of RAM encryption has already been discussed with the client and it was requested for this item to be added to the report for future research. This is connected to the pending release of *Airgap* OS to the public, as there may be other users and applications with different constraints and scenarios that potentially require such advanced memory encryption features.

POL-01-017 WP1: *Airgap* OS kernel lacks hardening flags *(Medium)*

The current *Airgap* OS Linux Kernel version is 5.7.19[8]. While reviewing the kernel configuration of the *Airgap* OS Linux kernel, it was noticed that the configuration lacks various security-related configuration options. An excellent utility assisting determination of the hardening state of a running kernel configuration is *kconfig-hardened-check*[9]. It is recommended to use this utility to check the kernel configuration file *build/buildroot/output/build/linux-5.7.19/.config* inside the *Airgap* build.

Besides the recommended kernel hardening options raised by *kconfig-hardened-check*, kernel Control-Flow-Integrity (CFI) protection is another useful mitigation which makes the exploitation of vulnerabilities with the Linux kernel more difficult. It has to be noted though that enabling CFI for the Linux kernel requires compilation of the kernel using *clang*[10] instead of *GCC*.

It was furthermore noticed that the running kernel has various debugging options enabled. It is recommended to disable *debug* kernel configuration options for released builds:

- `CONFIG_DEBUG_KERNEL=n`
- `CONFIG_STACKTRACE=n`
- `CONFIG_DEBUG_BUGVERBOSE=n`
- `CONFIG_DEBUG_KERNEL=n`

Revising the kernel configuration according to best practices is advised. It would be beneficial to apply hardening options concurrently to disabling *debug* options for all release builds, so as to enhance rigidity of the *Airgap* OS Linux kernel.

[8] For *Airgap* version 1.0.0rc13
[9] https://github.com/a13xp0p0v/kconfig-hardened-check
[10] https://outflux.net/blog/archives/2019/11/20/experimenting-with-clang-cfi-on-upstream-linux/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

POL-01-018 WP1: Updating *Airgap* OS Linux kernel to 5.8.y *(Low)*

It was observed that the Linux Kernel version 5.7.19, used in and compiled for the current version of Airgap OS, has already reached End-of-Life and will no longer receive any updates or potentially critical security fixes.

As communicated by Greg Kroah-Hartman[11]:

> *Note, this is the LAST 5.7.y kernel to be released (5.7.19). This release series is now end-of-life, please move to 5.8.y at this point in time.*
> *[…]*
> *All users of the 5.7 kernel series must upgrade.*

It is recommended to update the *Airgap* OS Linux kernel to a maintained version[12] in order to keep receiving important updates and security fixes.

[11] http://lkml.iu.edu/hypermail/linux/kernel/2008.3/05171.html
[12] https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

As noted in the *Introduction,* the overall outcomes of this assessment point to an already good security standing of the examined Polychain laptop devices, the investigated software as well as periphery. After an extensive examination in autumn 2020, Cure53 can confirm that most security risks have been accounted for in compliance with the threat model adopted by Polychain. While the prevalence of weaknesses over vulnerabilities is praiseworthy, it is important to note that all eighteen findings should be carefully reviewed and, ideally, mitigated.

It is, further, crucial to comment on the testing process itself, as it can give valuable insights into planning future work. Specifically, early on in the extensive project preparation phase, it became apparent that the build system was not as portable between host systems as initially hoped for. It took a certain amount of effort by both the client and the Cure53 team to arrive at a point of testability. With combined efforts, such state was reached ahead of the actual auditing commencement and the individual system analysis could start and move forward without further delay.

In Cure53's expert opinion, the Polychain team has taken measures to isolate the build process from local system dependencies seriously. Having said that, it quickly had to be admitted that the reproducible build step was not yet production-ready. It took time well into the last phase of this assessment for the builds to be fully operational, i.e. successfully and reproducibly verifies across all employed platforms. In particular, depending on certain aspects of local tool dependencies proved to be cumbersome and tedious to debug.

Quite clearly, special effort was made with regard to building *Airgap* OS in a reproducible and auditable way, which helps building trust within the system. Albeit carefully thought through, the system complexity still exposes a wide attack surface which might pose a substantial risk. A reduced and hardened kernel will help in mitigating the risk, while additional physical measures might be needed against rogue USB devices and other malicious actors.

A strong focus on removing not needed devices from the computer helps reduce the chances of adversarial interactions, while concurrently eliminating the attack surface and reducing the risk of supply chain attacks and/or simplifying the threat model. Nevertheless, there are additional measures that could be taken in relation to risk mitigation, for instance the recommended removal of the laptop's internal battery to eliminate its microcontroller. This is because externally plugged-in devices still present a risk which may require physical barriers.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

It should be underlined that a malicious power supply would potentially be able to perform side-channel or fault injection attacks, presenting a clear opportunity for setup hardening. At the same time, USB devices expected to be attached may inadvertently be replaced by devices that have been tampered with during the ceremony, a risk which can be mitigated by additional physical measures.

The security of the *Airgap* OS system is already in a decent state and it is evident that the customer has invested time and resources into hardening, thus evading many potential attack vectors. Nevertheless, areas for further improvement exist and can be seen in regard to the USB interface, which potentially puts the *Airgap* system at risk, as well as the *Airgap* OS kernel that needs to be reduced and more hardened towards security.

At a meta-level, the *Airgap* OS should incorporate counter-measures against the following two crucial attack vectors. First, there is a need to introduce a protection against Peripheral Access, blocking all external media, in particular USB, from connecting to the *Airgap* system. Second, Polychain should reduce software dependencies and minimize external dependencies to a smallest possible set in order to reduce the potential risk of using outdated and vulnerable software as well as supply chain attacks.

The extensive online discussions with the client during this assessment have shown that there are still numerous areas where Airgap OS can benefit from amelioration. This does not change the fact that the general security posture of the complex is already quite good in light of this autumn 2020 assessment's findings. The Polychain team has to be commended for the extensively thought-through approach of tackling the given Root-of-Trust problem space and the extensive use of verifiable open source components to achieve ambitious security goals. While the current security premise might be described as unexpectedly good for such a large and complex approach, it has to be clearly stated that further reduction in complexity and attack surface would benefit the project.

Cure53 would like to thank Lance Vick and Rob Witoff from the Polychain team for their excellent project coordination, support and assistance, both before and during this assignment.