CIP Security Hardening

Introduction

This document is intended to document CIP Security hardening guidelines and best practices. It will also help to meet IEC-62443-4-1 SG-3 process requirements. Different CIP workgroup members investigated security hardening practices which will help to harden CIP based systems and make it moer challenging to compromise CIP-based products. As security is always affected by several factors, it's important for CIP users to keep in mind these guidelines and apply them based on specific use cases.

Assumptions

It is assumed that the reader is familiar with the architecture and components of the CIP core/CIP kernel. See CIP User Manual for reference.

Table of contents

Revision History

Revision No	Date	Change description	Author	Reviewed by
001	2022-11-22	Template of Security hardening in CIP	Dinesh Kumar	
002	2022-11-22	Details added regarding guidelines, technical implementation	Stefan Schroeder	

---

1. Scope

This manual makes no claim of completeness for any specific use-case, product or project. The applicability of every guideline provided here must be evaluated by the user. Adherence to all these rules does not guarantee a secure product,

but serves as an additional puzzle-piece to reduce risk by reducing the attack surface.

Nothing in this document should be taken to contradict standards and guidelines made mandatory by national legislation. It has been created for adopters of the CIP operating system.

2. Terms

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

When this document “requires” a procedure this means that the procedure must be implemented to fulfill IEC-62443 requirements.

‘**(System) hardening**’: ‘**Hardening**’ or ‘**System hardening**’ is an umbrella term to denote a variety of activities to improve the overall security of an information system by reducing the attack surface.

Several methods are available to this end, depending on the phase of the development lifecycle:

During the **design** phase : A system architect can choose patterns, designs, or features based on their security merit.

- Example 1: A system may be designed to use the Telnet-protocol for remote access. A security expert would recommend to harden the system by replacing Telnet with a more secure protocol.
- Example 2: A system architect may be installing an operating system using a convenient default install template, thus including hundreds of applications that are not required for the intended use. A security expert would recommend to build the system from the ground from a minimal base, adding only what is necessary to perform the task the system is designed for.

During the **implementation** phase: Developers can implement software using secure coding best-practices and guidelines.

- Example: A developer may be offering to enter a password via the keyboard, echoing the letters as they are typed. A security expert would recommend to harden the implementation by replacing the letters with place-holders to prevent shoulder-surfing or accidental sharing during a video call.

During **builds**, **test-runs** and **deployment**: Testers can assess the integrity of build-artifacts by either Cryptographic integrity protection or provisioning of integrity information via separate channels to prevent tampering.

- Example: A tester may receive a test-object via mail or file share with no option to verify its integrity. A security expert would recommend to

sign artifacts digitally, encrypt them, or to provide file hashes via a second secure channel.

During **commissioning** and **maintenance**: Maintainers can improve the security posture of a product or device using the available configuration options to reduce the attack surface.

- Example: Maintainers may leave a deployed system in their original state as they were deployed. A security expert would recommend to review the configuration for potential security improvements, including but not limited to password updates, patching, and removal of unneeded functions.

Above all detailed hardening tips, there exists a list of established security principles that users **violate** at their peril:

- **The need-to-know principle:** Information should only be made available to users that require the information.
- **The principle-of-least-privilege:** Access to functionality should only be made available to users that need it to fulfill the system's intended purpose.
- **Avoid any single-point-of-failure:** Any single device may fail. Do not allow the failure of individual objects to tear down your entire system. Be resilient.
- **Separation-of-privilege:** Users of a system shall not have administrative privileges.
- **Defence-in-depth:** Failure of a single security feature should NOT allow an attacker to easily move laterally and escalate their attack.

'Hardening tools': Typically a system in a default state is not as secure as desired. There exists a variety of tools to harden a system based on predefined plans, templates and recommendations, e.g. Lynis or openSCAP. Some of the tools provide not only a hardening-function, but also auditing/compliance check features.

3. Hardening Guidelines

This section provides a list of well-established hardening guidelines.

- #REQ-CIP-HARD-001: You must deactivate/disable/remove default users and passwords. If a default password must be used, provide a function to change the password at the earliest opportunity. If passwords or other keys are used, they shall be stored in key management tools. If SSH is used, the password-less login with keys shall be used. If passwords must be used, you shall define a password policy that includes password update strategies, deputy policies, expiration, revival of locked accounts.
- #REQ-CIP-HARD-002: Every human user must have an individual account. There should be no account-sharing. User accounts shall be disabled

during off-boarding and it shall be cyclically checked if access is still required (e.g. by expiration). Separate administration accounts and user accounts.

- #REQ-CIP-HARD-003: You must remove all software packages and services that are not required for the use case, especially build-tools and any network programs.
- #REQ-CIP-HARD-004: You should keep track of all the installed software packages and cyclically compare them to the desired state. You should keep monitor the open network ports and cyclically compare them to the desired state.
- #REQ-CIP-HARD-005: You must employ firewall technology. The firewall must be configured to the most restrictive permissible settings.
- #REQ-CIP-HARD-006: You should remove any hardware ports and devices that are not required for the use case. All installed applications and systems shall be configured to have the minimum permissible permission set.
- #REQ-CIP-HARD-007: You must keep the CIP based system (including firmware) up-to-date with respect to security-patches. Ensure that you install the latest security patches in a timely manner. CIP developers regularly share latest CVE information and CIP users are advised to use CVE information and update the system regularly to receive the latest CVEs fixes.
- #REQ-CIP-HARD-008: CIP users SHALL apply CIP kernel updates on timely manner. Kernel updates provide security fixes, stability improvements, updated drivers, new functionality and improved performance. You should subscribe to the cip-dev mailing list to get the latest information.
- #REQ-CIP-HARD-009: You must use recommended encryption algorithms/cipher suites (e.g. by NIST SP 800-52) and provide the option to upgrade encryption ciphers in the final product. You must not use encryption algorithms/cipher suites which are known to be insecure.
- #REQ-CIP-HARD-010: You should not invent your own cryptography.
- #REQ-CIP-HARD-011: If you are using antivirus-software, keep the virus-database up-to-date.
- #REQ-CIP-HARD-012: Document and maintain secure configuration options through the entire life cycle of a product.
- #REQ-CIP-HARD-013: You must separate administrative and user accounts. Regular users should not be required to perform administrative duties.
- #REQ-CIP-HARD-014: Data at rest shall be encrypted if confidentiality, integrity and availabilty are at risk.

- #REQ-CIP-HARD-015: Measures shall be implemented to secure the operational environment of the CIP devices. (Trusted Execution Environment TEE)
- #REQ-CIP-HARD-016: CIP users MUST apply basic security using the update board support package (BSP). OPENPOINT: What's this?
- #REQ-CIP-HARD-017: You should separate static and changing data. The static (immutable) data shall be checked for integrity. See also 004. The changing data should be backed up. Rollback should be tested.

Note: As the CIP based system is based on the Debian operating system, you are advised to follow the latest guidelines published by the Debian project.

4. Technical Implementation details

The CIP security working group prepared a list of security enhancing packages (and dependencies) for CIP core as, but is not limited to (*alphabetical order*):

OPENPOINT: Will there be a separate "CIP core image security"?

#	Package	Debian Buster	Debian Bullseye
1.	`acl`	see package	see package
2.	`aide`	see package	see package
3.	`audispd-plugins`	see package	see package
4.	`auditd`	see package	see package
5.	`chrony`	see package	see package
6.	`fail2ban`	see package	see package
7.	`libtss2-esys-3.0.2-0`	-	see package
	`libtss2-esys0`	see package	-
8.	`libpam-cracklib`	see package	see package
9.	`libpam-pkcs11`	see package	see package
10.	`nftables`	see package	see package
11.	`openssh-client`	see package	see package
12.	`openssh-server`	see package	see package
13.	`openssh-sftp-server`	see package	see package
14.	`openssl`	see package	see package
15.	`sudo`	see package	see package
16.	`syslog-ng-core`	see package	see package
17.	`syslog-ng-mod-journal`	see package	see package
18.	`tpm2-abrmd`	see package	see package
19.	`tpm2-tools`	see package	see package
20.	`uuid-runtime`	see package	see package

Pre-installed Security Packages are defined in ISAR CIP Core Images Security.

Further, see Best Practices section.

- #REQ-CIP-HARD-101: You should include, configure and maintain the security packages from this list in your product.

Technical hints

- To check the list of installed packages, run

```
sudo dpkg-query -l | tee list_of_installed_packages.txt
```

- Integrity protection of the filesystem in Debian is provided by either aide, tripwire or samhain.
- Root-kit detection packages include chkrootkit, rkhunter and debcheckroot (not in included in Debian-repository 11-2022).
- To investigate open network ports on a system use nmap.

5. Normative references

There exists no agreed international 'Hardening Standard'. A variety of organisations and institutions publish hardening guidelines to improve the security stance of information systems. The following list may serve as an introduction to the topic:

- NIST Special Publication 800-123: Guide to General Server Security
- The Charter of Trust, specifically Principle 3
- CIS Operating System Hardening Benchmarks (members only)
- STIGS by the US-DOD
- SANS Security Policy Templates
- Germany: BSI ICS-Security Kompendium v 1.23

CIP users are also advised to see/refer to the community and/or third-party applications guidelines.

Further reading:

- BASICS OF THE CIS HARDENING GUIDELINES
- Securing and Hardening Embedded Linux Devices: Theory and Practice.
- M2: Insecure Data Storage from OWASP
- What is a trusted execution environment (TEE) and how can it improve the safety of your data?
- Vulnerabilities, Exploits, and Threats: Defining three key terms in cybersecurity
- Hardening Network Devices recommendations from NSA.
- Securing Debian Manual
- Hardening Debian

All major operating systems provide hardening guidelines, e.g. Red Hat, Suse, Arch, Debian and Microsoft. The ISO/IEC 62443 4-1 does not require a specific

set of hardening rules to be applied, but mandates the existence of a Hardening Guide for a product.

6. History

Initial: Nov, 2022 by Dinesh Kumar and Stefan Schroeder.

Revision History

Revision No	Date	Change description	Author	Reviewed by
001	2022-11	Initial	Dinesh Kumar and Stefan Schroeder	
002	2023-03-13	Integrated https://gitlab.com/cip-project/cip-documents/-/blob/master/security/security_hardening_guidelines.md?plain=1 by Djuned Fernando Djusdek		