Formal Verification now-*ish* in European Cyber Resiliency standard for OS

June Andronick

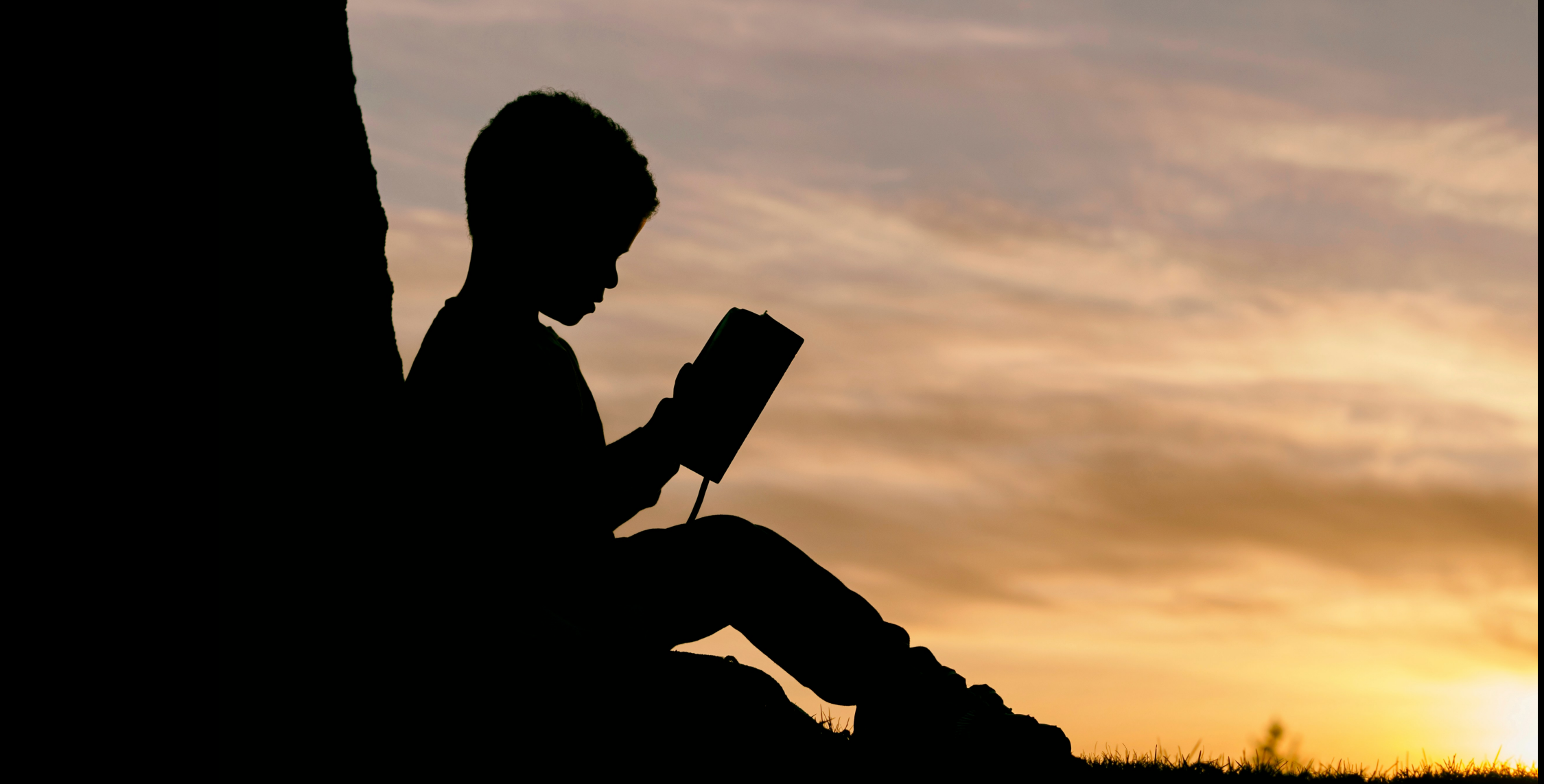


Photo by Aaron Burden on Unsplash

The story in 1 slide

Before

no mention of formal verification

Operating Systems standard

("Cybersecurity requirements for operating systems")

commissioned by the European Union

still being finalised

May'26 — *last week*

Now-*ish*

formal verification added as a *mitigation for:

- ~~correctness~~ access control
- ~~integrity~~ access control
- ~~confidentiality~~ access control

* *one option to address*

(still hope to change it back)

European Commission

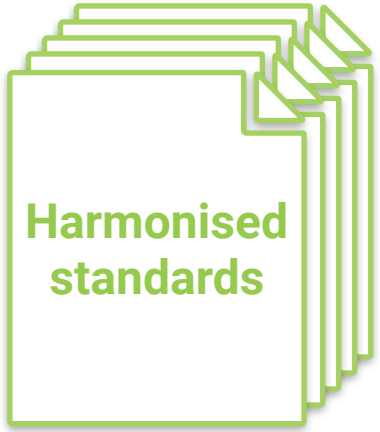
CRA (Cyber Resiliency Act)

an EU law that sets cybersecurity requirements for products with digital elements

2024

European Standards Organisations

CEN | **CENELEC** | **ETSI**



propose a way to comply to the CRA;

provides ***"presumption of conformity"***

(still being finalised)

European Commission

European Standards Organisations

CEN | CENELEC | ETSI

JuneA

Where do I fit in all this?

CRA (Cyber Resiliency Act)

*an EU law that sets cybersecurity requirements for products with digital elements**

Horizontal standards

applies across types of products

Vertical standards

applies to a specific product category

e.g.

Operating Systems

Browsers

Smartcards

...

Sep'25 Nov'25 Feb'26 May'26 Aug'26

** in non regulated sectors (i.e. excl. Automotive, Aviation, Marine, Medical, Defence)*

Harmonised vertical standard EN 304 626

Cybersecurity requirements for operating systems

(up to May'26)

- **Technical Requirements**
  - Mitigations
- **Use cases**
  - Required Mitigations

- **TR-SSDD: Secure design and development**
  - MI-SSCA: Static source code analysis for memory errors
  - MI-IMSL: Implement in a memory-safe language
  - . . .
  - MI-FVFC: Formal verification of functional correctness
- **TR-IDST: Integrity of data stored**
  - MI-DCST: Detect corruption of data stored
  - . . .
  - MI-FIST: Formal proof of enforcement of integrity of data stored
- **TR-CDST: Confidentiality of data stored on the product**
  - . . .
  - MI-FCST: Formal proof of enforcement of confidentiality of data stored

(up to May'26)

- **TR-SSDD: Secure design and development**
  - **MI-SSCA: Static source code analysis for memory errors**
  - **MI-IMSL: Implement in a memory-safe language**
  - . . .
  - **MI-FVFC: Formal verification of functional correctness**

5.2.3.2 MI-SSCA: Static source code analysis for mem

All security-relevant parts of the product shall be **checke memory errors using a source code analysis tool** that d that may produce common memory errors, such as:

- buffer overflow
- out-of-bounds
- use after free
- double free
- use of uninitialized variables
- dereference of invalid pointer

The sufficiency of the source code analysis tool and the manner of running it shall be documented.

All warnings, annotations, or other method of suppressi from the analysis tool shall be documented with a ratior it does not constitute an unacceptable risk.

5.2.3.7 MI-FVFC: Formal verification of functional correctness

All security-relevant parts of the product shall be **formally proved correct** with respect to a formal specification using a **sound formal verification tool** that produces a **formal proof of conformity** between the source code and the formal specification. Such a proof may imply the absence of common memory errors, such as:

- buffer overflow
- [. . .]

The sufficiency of the formal verification tool and the formal specification shall be documented. The process to run the formal verification tool to produce a formal proof shall be documented.

An explicit list of the assumptions required for the mitigation to apply to the running product shall be documented, in enough detail to know what should be checked in practice on the product to ensure that the assumptions are indeed being met.

(up to May'26)

- **Technical Requirements**
  - Mitigations
- **Use cases**
  - Required Mitigations

- **TR-SSDD: Secure design and development**
  - MI-SSCA: Static source code analysis for memory errors
  - MI-IMSL: Implement in a memory-safe language
  - . . .
  - MI-FVFC: Formal verification of functional correctness
- **TR-IDST: Integrity of data stored**
  - MI-DCST: Detect corruption of data stored
  - . . .
  - MI-FIST: Formal proof of enforcement of integrity of data stored
- **TR-CDST: Confidentiality of data stored on the product**
  - . . .
  - MI-FCST: Formal proof of enforcement of confidentiality of data stored

- **UC-IoT-1: Non-internet-connected device such as a bluetooth speaker**
  - Required Mitigations: None

- **UC-LA-2: Enterprise laptop**
  - Required Mitigations:
    1-(SSCA or FVFC)
    2- (IMSL or …)
    3- …

(up to May'26)

- Technic
  - Mitig
- Use case
  - Requi

- TR-SSDD: Secure d… …ment
- …sis for memory errors
- …afe language
- …tional correctness
- …ored
- U… de…
- …ored
- …op
- …s:
- …(IMSL or…
- 3- …

12 Feb 2026:
accepted

15 May 2026:
even after the major rewrite,
no further discussion on the topic,
contributions still part of the standard

but…

(up to May'26)

- **Technical Requirements**
  - Mitigations
- **Use cases**
  - Required Mitigations

(as of Aug'26)

- **TR-SSDD: Secure design and development**
  - MI-SSCA: Static source code analysis for memory errors
  - MI-IMSL: Implement in a memory-safe language
  - . . .
  - MI-FVFC: Formal verification of functional correctness
- **TR-IDST: Integrity of data stored**
  - MI-DCST: Detect corruption of data stored
  - . . .
  - ~~MI-FIST: Formal proof of enforcement of integrity of data stored~~
- **TR-CDST: Confidentiality of data stored on the product**
  - . . .
  - ~~MI-FCST: Formal proof of enforcement of confidentiality of data stored~~
- **TR-AUTH: Authentication and access control**
  - [. . .]
  - MI-ACSD: Access control for stored data

6.9 MI-ACSD: Access control for stored data

The product shall protect stored data from **unauthorised access or modification** by enforcing access controls and privilege separation at the operating system level.

Assessment preparation

1) **List the types of data stored on the product that are protected from unauthorised access or modification**, and for each **the mechanism** the manufacturer documents, whether access controls and privilege separation at the operating system level, another documented mechanism, **or a documented formal proof**, and the methods of accessing or modifying that data available to an attacker based on the risk assessment.

[. . .]

NOTE: Guidance: [. . .]

A formal proof, in a formal verification tool, that the product protects stored data from unauthorised access or modification is **an acceptable alternative** means of demonstrating conformance with this mitigation, provided the tool selection and configuration criteria, the **formal definition of confidentiality and integrity**, and the formal argument applying it to a formalisation of the product's **source code** are documented.

- **MI-ACSD: Access control for stored data**

(up to May'26)

- **Technical Requirements**
  - Mitigations
- **Use cases**
  - Required Mitigations

- **TR-SSDD: Secure design and development**
  - MI-SSCA: Static source code analysis for memory errors
  - MI-IMSL: Implement in a memory-safe language
  - . . .
  - MI-FVFC: Formal verification of functional correctness
- **TR-IDST: Integrity of data stored**
  - MI-DCST: Detect corruption of data stored
  - . . .
  - ~~MI-FIST: Formal proof of enforcement of integrity of data stored~~
- **TR-CDST: Confidentiality of data stored on the product**
  - . . .
  - ~~MI-FCST: Formal proof of enforcement of confidentiality of data stored~~
- **TR-AUTH: Authentication and access control**
  - [. . .]
  - MI-ACSD: Access control for stored data

- **UC-REST: Restricted operating system**
- **UC-GEN: General-purpose operating system**

Takeaway

Hopefully at least awareness has been raised to consider formal verification

… but there's still some resistance…

despite making it an optional path that they don't have to use

We need at least push for formal verification to **not be penalised**!

… until we succeed in **mandating it** where it matters