Wasefire

Julien Cretin – Google
Jean-Michel Picod – Google

BACKGROUND

Embedded systems are ubiquitous, found in everything from vehicles to medical devices and industrial control systems. These systems often control critical infrastructure and processes, making them attractive targets for cyberattacks.

> **Compromised embedded systems** have **severe consequences**, such as data breaches, financial loss, and even physical damage or loss of life.

For example, security vulnerabilities in medical devices can be exploited to compromise patient safety or disrupt critical care. One of the key challenges in embedded systems security is the limited resources available on these devices, such as constrained processing power, memory, and storage capacity. These limitations make implementing robust security measures challenging, and put security at a tradeoff with performance and functionality.

VISION

We introduce a project aimed at simplifying building safer embedded systems.

> Wasefire is an open-source project with a permissive license that offers a **simple-to-use** and **secure-by-default** ecosystem for firmware development.

Wasefire provides a streamlined and secure development environment for firmware that simplifies the development process and incorporates security best practices by default. This enables developers to create secure firmware without requiring extensive security expertise and only focusing on the business logic they want to implement.

DESIGN

The framework provides a secure platform, written in Rust for its performance and built-in memory safety, on which sandboxed applets can run. Both the platform and applets are upgradable by default. Applets can be written in any language compiling to WebAssembly (a safe, portable, and compact low-level code format). The platform is hardware independent and can run on Linux. Applets can be tested without hardware by running them on this Linux platform.

COMPARISON

Existing solutions are either, not secure by default, target embedded developers or security experts, or only provide source-level portability. Wasefire in comparison is secure by design, targets software engineers (without embedded or security expertise), and provides binary-level portability.

LIMITATIONS

Wasefire currently suffers some temporary limitations:

- WebAssembly interpretation is 10x to 100x slower than native code. This will improve by supporting CHERI (hardware fine-grained isolation), Pulley (a WebAssembly target byte code optimized for interpretation), or native applets with proper isolation (native applets are currently supported but without isolation).
- Only a few boards are supported. Some important features are not yet implemented. The applet API is not as rich as it should be. All those points will improve over time, possibly through contributions thanks to the project being open-source.
- The project is still experimental and the APIs are not stable. This will improve as more users rely on the framework and the project grows in maturity.

ADOPTION

Wasefire is currently used internally on two projects.

> We're looking to **onboard an external industrial use case** to further develop the platform.

Use-cases that would benefit the most from using Wasefire satisfy some or all of the following criteria:

- The engineering team doesn't have embedded and/or security expertise
- The firmware is mostly light interactive logic (no heavy batch computations)
- The device is resource constrained (such that Linux is not an option)

We can provide assistance to support a new language, support a new board, extend the applet API, and implement security features.