R^3: Random Robust Rust

August 2017

supervisors	Martin Steffen
group	PMA, HiB
type	60 ECTS
study program	computer science
planned date of completion	May 2018

Short description

The task is to design and implement a radmomization concept for Rust compilation

Background and motivation

Embedded system in the rapidly growing IoT ("internet of things") and "smart" consumer appliances have come under increasing attacks. An established technique in more convential settings to harden the system is *randomization* for instance of the memory layout. Having exact knowledge of such layout can help to exploit weaknesses in a devastating manner like taking control of the system itself (as opposed to just crashing it). Randomiziaton helps insofar that a possible adversary has figure out the memory layout for every individual systems anew. The individual system still may be vulnerable, but it's no longer easy to do certain attacks for the whole /class/ of systems, thereby making the overall /infrastructure/ more robust an less tempting for an attack altogether.

Rust [1] is a quite recent programming language, with a first stable release 2015, which is gaining traction.

The stated goal of the language is to enable fast, efficient, and memory safe *systems programming*, stressing safe code even for low-level, hardware-close applications and in the presence of *concurrency*. It combines a number of advanced and/or novel features, for example sophisticated memory management, but *not* based on carbage collection but based on ownership. Being targeted also for "embedded systems", one feature of the memory management is that it runs without the support of a surrounding operating system or run-time system (on the "naked hardware"). That makes it an attractive language for IoT and related applications. On the downside, the memory management cannot rely on the operating system to assist in radmonization

Problem setting

The task consists in desiging and implementing a concept for randomization of memory layout for Rust. Collaboration with the *Security Lab* is possible, and the implementation may target platforms like *Rasberry Pi* as representing IoT platforms.

Keywords: security, IoT, compiler, code generation, hardening, randomization, compiler plugin, Rust

References

[1] Rust programming language. `https://www.rust-lang.org/`, Dec. 2016.