Investigating and Implementing Intra-Kernel Security in Unikraft

Tom Cassar

Unikernels are lightweight single address space virtual machines[1] which are created by compiling an application together with the operating system libraries on which it depends. This creates a lightweight executable which can boot in milliseconds, has a minimal attack surface, and boasts stronger isolation than alternative technologies (such as containers) on systems where multiple unikernels are being run [2].

Since unikernels are designed to run only one application, they are compiled missing many security features that come standard on other operating systems. Michaels and Dilelo [3] found that the unikernels they investigated had minimal levels of security and were compared to embedded systems. They wrote that "features like "ASLR, `W^X`, stack canaries, heap integrity checks and more are completely absent or seriously flawed". Other papers have argued that the assumption that unikernels can be viewed as a single unit of trust are not suitable for today's cloud security requirements.[1-1]

Unikraft is a collection of tools intended to be used to for creating POSIX-compliant operating systems [2-1] is open source and actively under development. This makes Unikraft a good platform to use to research and implement intra-kernel security.

The project being proposed here would look to:

1. Investigate existing intra-kernel security features present in the Unikraft unikernel.
2. Implement and contribute one or more intra-kernel security features to Unikraft. These features should be chosen such that they solve a problem discovered during the investigation without significantly affecting unikernel performance (as was shown possible with FlexOS [4]).

Rust is a low-level memory safe language created in 2006. It is the only language other than C to be used in the Linux kernel,[5] and recently was mentioned as part of a document released by the NSA [6] discussing Software Memory Safety and "how a software program manages memory is core to preventing many vulnerabilities and ensuring a program is robust". Thus, it makes Rust a sensible choice for a development language. However, the exact task is still to be determined, and the nature of the task may influence the choice of language. Unikraft is mostly in C, but efforts porting to Rust have already started.

References

1. https://www.research.ed.ac.uk/en/publications/the-case-for-intra-unikernel-isolation ↩ ↩
2. https://ieeexplore.ieee.org/abstract/document/9138883 ↩ ↩
3. https://people.cs.pitt.edu/~babay/courses/cs3551/projects/SCADA_Unikernel/NCC_Group-Assessing_Unikernel_Security.pdf ↩
4. https://dl.acm.org/doi/abs/10.1145/3503222.3507759 ↩
5. https://www.infoq.com/news/2021/04/rust-linux-kernel-development/ ↩
6. https://media.defense.gov/2022/Nov/10/2003112742/-1/-1/0/CSI_SOFTWARE_MEMORY_SAFETY.PDF ↩