Investigating and Implementing Intra-Kernel Security in Unikraft

Tom Cassar

April 2024

Unikernels are lightweight single address space machines (1) which are created by compiling an application with the libraries on which it depends as well as a minimal operating system layer. This creates a lightweight kernel executable which is then generally run on a virtual machine on top of a hypervisor. Unikernels boast both security and performance benefits: fast system calls (which stem from their single address space) (2), millisecond boot times (3), a minimal attack surface, and stronger isolation than alternative technologies (such as containers) on systems where multiple unikernels are run (4).

Unikernels are compiled missing many security features that come standard on many other operating systems. This partly because unikernels are designed to only run one application, and partly due to a lack of maturity. Michaels and Dilelo reported that "features like ASLR, `W^X`, stack canaries, heap integrity checks and more are completely absent or seriously flawed" (5). 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).

Unikraft (6) is a collection of tools intended to be used to for creating POSIX-compliant operating systems. It is open source, actively under development, and as of today the most mature unikernel (7). 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 (e.g. ASLR, `W^X`, enhanced intrusion detection mechanisms, etc). 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) (8)

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 (9), and recently was mentioned as part of a document released by the NSA (10) 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] Olivier P, Barbalace A, Ravindran B. The case for Intra-Unikernel isolation; 2020. Available from: https://www.research.ed.ac.uk/en/publications/the-case-for-intra-unikernel-isolation.

[2] Olivier P, Lefeuvre H, Chiba D, Lankes S, Min C, Ravindran B. A Syscall-Level Binary-Compatible Unikernel. IEEE Transactions on Computers. 2022;71(9):2116-27.

[3] Unikraft;. Available from: https://github.com/unikraft/.

[4] Talbot J, Pikula P, Sweetmore C, Rowe S, Hindy H, Tachtatzis C, et al. A Security Perspective on Unikernels. In: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security); 2020. p. 1-7.

[5] Michaels S, Dileo J. Assessing unikernel security. Technical report, NCC group, Tech Rep. 2019.

[6] Kuenzer S, Bădoiu VA, Lefeuvre H, Santhanam S, Jung A, Gain G, et al. Unikraft: fast, specialized unikernels the easy way. In: Proceedings of the Sixteenth European Conference on Computer Systems. EuroSys '21. New York, NY, USA: Association for Computing Machinery; 2021. p. 376–394. Available from: https://doi.org/10.1145/3447786.3456248.

[7] Lefeuvre H, Gain G, Dinca D, Jung A, Kuenzer S, Badoiu VA, et al. Unikraft and the coming of age of unikernels. login; The Usenix Magazine. 2021.

[8] Lefeuvre H, Bădoiu VA, Jung A, Teodorescu SL, Rauch S, Huici F, et al. FlexOS: towards flexible OS isolation. In: Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS '22. New York, NY, USA: Association for Computing Machinery; 2022. p. 467–482. Available from: https://doi.org/10.1145/3503222.3507759.

[9] De Simone S. Using Rust to write safe and correct Linux kernel drivers. 2021 4. Available from: https://www.infoq.com/news/2021/04/rust-linux-kernel-development/.

[10] National Security Agency. CSI Software Memory Safety. National Security Agency; 2024. Accessed: 2023-04-17. Available from: https://www.nsa.gov/Press-Room/News-Highlights/Article/Article/3215760/nsa-releases-guidance-on-how-to-protect-against-software-memory-safety-issues/.