FlexOS: Towards Flexible OS Isolation

Hugo Lefeuvre
The University of Manchester
Manchester, UK

Vlad-Andrei Bădoiu
University Politehnica of Bucharest
Bucharest, Romania

Alexander Jung
Lancaster University / *Unikraft.io*
Lancaster, UK

Stefan Lucian Teodorescu
University Politehnica of Bucharest
Bucharest, Romania

Sebastian Rauch
Karlsruhe Institute of Technology
Karlsruhe, Germany

Felipe Huici
NEC Labs Europe / *Unikraft.io*
Heidelberg, Germany

Costin Raiciu
University Politehnica
of Bucharest / *Correct Networks*
Bucharest, Romania

Pierre Olivier
The University of Manchester
Manchester, UK

1 Motivation

Modern OS architectures are heavily interlinked with the protection mechanisms they rely upon. OSes rigidly commit at design time on various high-level safety decisions, such as the use of software verification, hardware isolation, or runtime checking. Changing these post-design time is costly and thus rare. For instance, removing the traditional user/kernel separation [19] requires a lot of engineering effort, and so does breaking down a process into multiple address spaces for isolation purposes [11].

This rigid use of safety primitives in modern OSes poses a number of problems. First, it precludes per-application OS specialization [7, 10, 20, 21] at a time when modern applications exhibit a wide range of safety and performance requirements. Second, new isolation mechanisms [1, 2, 5, 6, 25, 26] are regularly being proposed by CPU manufacturers, and retrofitting support for a new mechanism typically requires significant redesign. This is an even greater problem when multiple mechanisms can be used for the same task; choosing the primitive that provides the best performance depends on many factors, and should ideally be postponed to deployment time. Finally, when the protection offered by a hardware primitive breaks down (e.g. Meltdown [17]), it is difficult to quickly decide how it should be replaced, and generally costly to do so.

The research question that we explore in this work is: how can we design an OS that empowers users to *easily and safely* switch between different isolation and protection primitives at *deployment time* instead of design time?

2 Limitations of the Traditional OS Landscape

The traditional OS design landscape broadly consists of microkernels [9, 12], which favor hardware protection and verification over performance, monolithic kernels [3], which choose privilege separation and address spaces to isolate applications, but assume all kernel code is trusted, and single-address-space OSes, which attempt to bring isolation within the address space [4, 8, 16], or ditch all protection for maximum performance [13, 18, 23]. Each of these approaches is a single point in the OS safety/performance design space. For each of these points, switching the isolation granularity or mechanism is a *tedious and costly* task. Recent approaches going into the direction of more flexibility in isolation granularity [22, 24] are still limited to a handful of points in the design space, due to their focus on a single isolation technology (e.g. MPK): porting them to other mechanisms (e.g. EPT, CHERI [26]) would required a tedious redesign.

3 FlexOS: Towards Flexible OS Isolation

In this work (published at HotOS'21 [15] and ASPLOS'22 [14]), we present *FlexOS*, a modular OS that decouples isolation strategy from design: its compartmentalization and protection profile can easily and safely be tailored towards a specific application or use-case at build time, as opposed to design time. To that aim, we extend the libOS model and augment its capacity to be specialized towards a given use case, historically done towards performance, towards the *safety* dimension.

FlexOS provides isolation at a flexible granularity, with a variety of hardware and software mechanisms, while maintaining high performance. With FlexOS, the user can decide, at build time, which of the fine-grained OS components should be compartmentalized (e.g. the scheduler, TCP/IP stack, etc.), how to instantiate isolation and protection primitives for each compartment (in particular what isolation mechanism to use), what data sharing strategies to use for communication between compartments, as well as what software hardening mechanisms should be applied on which compartments.

Due to the vast size of the design space unlocked by FlexOS' flexibility, it may be hard for a user to select a suitable configuration for a given use case. In that context, we also propose a semi-automated design-space exploration technique helping to identify relevant configurations for a given application with given safety/performance constraints. This technique uses partially ordered sets to classify configurations by probabilistic degrees of safety, and returns the subset of the safest ones satisfying a given performance constraint for an application.

FlexOS is available online under an open-source license: https://project-flexos.github.io/.

References

[1] ARM Ltd. 2009. Building a Secure System using TrustZone Technology. https://developer.arm.com/documentation/genc009492/c. Online; accessed Jan 24, 2021.

[2] ARM Ltd. 2019. ARM Morello Program. https://developer.arm.com/architectures/cpu-architecture/a-profile/morello. Online; accessed June 25, 2020.

[3] Daniel P Bovet and Marco Cesati. 2005. *Understanding the Linux Kernel: from I/O ports to process management.* O'Reilly Media, Inc.

[4] Jeffrey S. Chase, Henry M. Levy, Michael J. Feeley, and Edward D. Lazowska. 1994. Sharing and Protection in a Single-Address-Space Operating System. *ACM Trans. Comput. Syst.* 12, 4 (1994). https://doi.org/10.1145/195792.195795

[5] Jonathan Corbet. 2015. Memory protection keys. *Linux Weekly News* (2015). https://lwn.net/Articles/643797/.

[6] Victor Costan and S. Devadas. 2016. Intel SGX Explained. *IACR Cryptol. ePrint Arch.* 2016, 86 (2016). https://eprint.iacr.org/2016/086.pdf

[7] D. R. Engler, M. F. Kaashoek, and J. O'Toole. 1995. Exokernel: An Operating System Architecture for Application-Level Resource Management. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP'95)*. Association for Computing Machinery. https://doi.org/10.1145/224056.224076

[8] Gernot Heiser, Kevin Elphinstone, Jerry Vochteloo, Stephen Russell, and Jochen Liedtke. 1999. The Mungi Single-Address-Space Operating System. *Software: Practice and Experience* 28, 9 (1999). https://doi.org/10.1002/(SICI)1097-024X(19980725)28:9%3C901::AID-SPE181%3E3.0.CO;2-7

[9] Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, and Andrew S. Tanenbaum. 2006. MINIX 3: A Highly Reliable, Self-Repairing Operating System. *SIGOPS Oper. Syst. Rev.* 40, 3 (2006). https://doi.org/10.1145/1151374.1151391

[10] M. Frans Kaashoek, Dawson R Engler, Gregory R Ganger, Héctor M Briceno, Russell Hunt, David Mazieres, Thomas Pinckney, Robert Grimm, John Jannotti, and Kenneth Mackenzie. 1997. Application performance and flexibility on exokernel systems. In *Proceedings of the 16th ACM symposium on Operating systems principles*. https://dl.acm.org/doi/10.1145/268998.266644

[11] Douglas Kilpatrick. 2003. Privman: A Library for Partitioning Applications. In *USENIX Annual Technical Conference, FREENIX Track (ATC'03)*. https://www.usenix.org/legacy/events/usenix03/tech/freenix03/kilpatrick.html

[12] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. 2009. SeL4: Formal Verification of an OS Kernel. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP'09)*. Association for Computing Machinery. https://doi.org/10.1145/1629575.1629596

[13] Simon Kuenzer, Vlad-Andrei Bădoiu, Hugo Lefeuvre, Sharan Santhanam, Alexander Jung, Gaulthier Gain, Cyril Soldani, Costin Lupu, Ştefan Teodorescu, Costi Răducanu, Cristian Banu, Laurent Mathy, Răzvan Deaconescu, Costin Raiciu, and Felipe Huici. 2021. Unikraft: Fast, Specialized Unikernels the Easy Way. In *Proceedings of the 16th European Conference on Computer Systems (EuroSys'21)*. Association for Computing Machinery. https://doi.org/10.1145/3447786.3456248

[14] Hugo Lefeuvre, Vlad-Andrei Bădoiu, Alexander Jung, Stefan Teodorescu, Sebastian Rauch, Felipe Huici, Costin Raiciu, and Pierre Olivier. 2022. FlexOS: Towards Flexible OS Isolation. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'22)*. https://doi.org/10.1145/3503222.3507759

[15] Hugo Lefeuvre, Vlad-Andrei Bădoiu, Stefan Teodorescu, Pierre Olivier, Tiberiu Mosnoi, Răzvan Deaconescu, Felipe Huici, and Costin Raiciu. 2021. FlexOS: Making OS Isolation Flexible. In *Proceedings of the 18th Workshop on Hot Topics in Operating Systems (HotOS'21)*. https://sigops.org/s/conferences/hotos/2021/

[16] I. M. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns, and E. Hyden. 1996. The design and implementation of an operating system to support distributed multimedia applications. *IEEE Journal on Selected Areas in Communications* 14, 7 (1996). https://doi.org/10.1109/49.536480

[17] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, et al. 2018. Meltdown: Reading kernel memory from user space. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security'18)*. https://www.usenix.org/conference/usenixsecurity18/presentation/lipp

[18] Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. 2013. Unikernels: Library Operating Systems for the Cloud. In *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'13)*. Association for Computing Machinery. https://dl.acm.org/doi/10.1145/2451116.2451167

[19] Toshiyuki Maeda and Akinori Yonezawa. 2003. Kernel Mode Linux: Toward an operating system protected by a type theory. In *Annual Asian Computing Science Conference*. Springer. https://rd.springer.com/chapter/10.1007/978-3-540-40965-6_2

[20] Filipe Manco, Costin Lupu, Florian Schmidt, Jose Mendes, Simon Kuenzer, Sumit Sati, Kenichi Yasukata, Costin Raiciu, and Felipe Huici. 2017. My VM is Lighter (and Safer) than your Container. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP'17)*. Association for Computing Machinery. https://dl.acm.org/doi/abs/10.1145/3132747.3132763

[21] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. 2014. ClickOS and the Art of Network Function Virtualization. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI'14)*. USENIX Association. https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/martins

[22] Ruslan Nikolaev, Mincheol Sung, and Binoy Ravindran. 2020. LibrettOS: A Dynamically Adaptable Multiserver-Library OS. In *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'20)*. Association for Computing Machinery. https://doi.org/10.1145/3381052.3381316

[23] Pierre Olivier, Daniel Chiba, Stefan Lankes, Changwoo Min, and Binoy Ravindran. 2019. A Binary-Compatible Unikernel. In *Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2019)*. Association for Computing Machinery. https://doi.org/10.1145/3313808.3313817

[24] Vasily A. Sartakov, Lluis Vilanova, and Peter Pietzuch. 2021. CubicleOS: A Library OS with Software Componentisation for Practical Isolation. In *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'21)*. Association for Computing Machinery. https://dl.acm.org/doi/10.1145/3445814.3446731

[25] David Schrammel, Samuel Weiser, Stefan Steinegger, Martin Schwarzl, Michael Schwarz, Stefan Mangard, and Daniel Gruss. 2020. Donky: Domain Keys – Efficient In-Process Isolation for RISC-V and x86. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security'20)*. USENIX Association. https://www.usenix.org/conference/usenixsecurity20/presentation/schrammel

[26] Robert NM Watson, Jonathan Woodruff, Peter G Neumann, Simon W Moore, Jonathan Anderson, David Chisnall, Nirav Dave, Brooks Davis, Khilan Gudka, Ben Laurie, et al. 2015. CHERI: A hybrid capability-system architecture for scalable software compartmentalization. In *2015 IEEE Symposium on Security and Privacy*. IEEE. https://doi.org/10.1109/SP.2015.9