Demo: On-The-Fly Generation of Unikernels for Software-Defined Security in Cloud Infrastructures

Maxime Compastié*†, Rémi Badonnel*, Olivier Festor*, and Ruan He†
*Madynes Research Team - LORIA/INRIA, France. {remi.badonnel,olivier.festor}@loria.fr
†Orange Labs, France.{maxime.compastie,ruan.he}@orange.com

***Abstract*—The programmability of security mechanisms through software-defined security permits the outsourcing of security management to a dedicated plan. Unikernels offer new perspectives for supporting this programmability, and addressing the challenges with respect to the heterogeneity and the dynamics of cloud resources. In this demo, we demonstrate how unikernel properties may enable an adequate security enforcement at the resource level. We present a framework for integrating security mechanisms into unikernel virtual machines, and align them to a given security policy, through the on-the-fly unikernel VM generation. We showcase an implementation prototype and confront it to cloud exploitation scenarios.**

I. Introduction

Cloud computing [1] has become a widespread architectural model for service supply based on the abstraction of infrastructures and depending hardware. The management of this cloud infrastructure is challenged by the inherent dynamism of its resources allocation and configuration due to the service provision constrains (e.g. on-demand provisioning, scalability). From a security perspective, the management issue also affects the security policy specification and its enforcement in a dedicated security context. This issue is stressed by the resource distribution across both infrastructures (multi-cloud property) and stakeholders (multi-tenancy property), providing heterogeneous security context to cloud resource.

The Software-defined Security (SDSec) [2] addresses this issue by enabling the programmability of security mechanisms to enforce a security policy in coherence with a specific security context. While the security decision-making process is dispatched in the security control plane, the resource needing protection and their dedicated security mechanism constitute the data plane. However, the enforcement exhaustiveness is subjected to the coverage of security mechanisms over resource requiring protection, and their technical adequacies. Moreover, resource lifecycles have to be properly synchronized with security mechanisms to ensure the coherence of the security enforcement with the policy.

Tackling this issue, we explore a novel approach for the cloud resource enforcement based on the unikernel system architecture properties. We propose a framework for the on-the-fly generation of secured unikernel images conforming with security constraints from a security policy, and complying with its modifications. The enforcement is committed by programmable security mechanisms that are embeds in unikernel images. This demo exposes an implementation prototype, and confronts it to cloud exploitation scenarios.

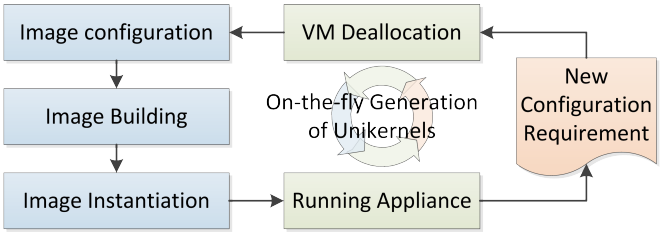

Figure 1. On-the-fly generation of secured unikernels

II. Virtualization and Unikernels

Performance improvement efforts have led to several initiatives in the area of system virtualization, in order to reduce the footprint of appliance execution environments. The thriving adoption of containerization has successfully contributed to this objective, as well as refining the granularity of cloud scalability, improving their portability and promoting new appliance lifecycles (e.g. DevOps [3]). However, containerized applications require OS kernel modifications, banning several applications that have still to comply with complex executions driven by legacy support constraints.

Unikernels tackle this issue by proposing a simplified system architecture complying with the library OS paradigm and addressing legacy support constraints [4]. It enables the design of VMs packed with a single application, its minimum runtime dependencies and optimized virtual hardware management, contributing to application efficiency and a minimized attack surface. From a technical perspective, all routines in unikernel VMs are executed in privileged mode, and share the same address space. This design choice impacts the VM lifecycle through an ex-situ configuration management and a shorter lifespan, as illustrated on Figure 1.

III. Unikernel Generation Framework

We have designed in [5] a framework for generating secured unikernel images and instantiating them as VMs. The main building blocks are the unikernel source code repository, the unikernel building toolchain in a dedicated environment, the VM exploitation environment, and the management plane able to cope with security requirements.

Figure 2 describes the unikernel image building and instantiation process, and the related components. Before any exploitation (0), we assume the existences of an unikernel application source code, of its dependencies stored in an

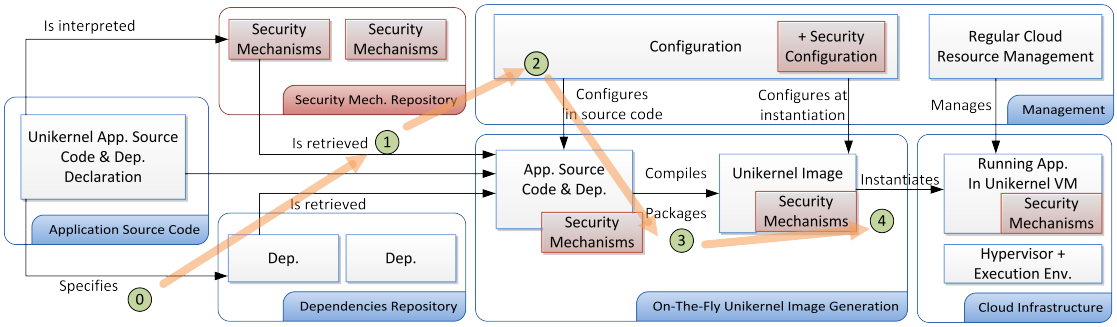

Figure 2. SDSec framework for cloud infrastructures with on-the-fly generation of unikernel images

external repository, and programmable security mechanisms stored on a dedicated repository, ready to be tied to unikernel source code. When the building process is triggered (1), all those three elements are fetched to a common building environment. Then (2), the management plan configures the unikernel source code, including the security mechanisms in accordance with the security policy. The configured source code is then compiled with the unikernel toolchain (3), and produces an image ready to be instantiated. Finally (4), at instance time, the image is pulled to the exploitation environment, and is instantiated as a VM. The management plane can configure the VM at boot time through boot arguments.

The software-defined security approach is taken into account through the programmability of security mechanisms to be included in the unikernel image, through their source code-based configuration, and through boot argument supply. Putting on an equal footing regular software components and security mechanisms goes in favor of security and performances: security mechanisms are closely integrated with other resources in unikernels, enabling it to interact with each of them. In addition, the library OS paradigm permits the mechanisms to also address the hardware management, promoting a more inclusive enforcement at the resource level. Finally, as security mechanisms and regular unikernel source codes undergo the same compilation process, they are optimized to coexist, and minimize the overhead in the unikernel exploitation.

The major short-coming of this approach is the alignment with the security policy: as the security mechanisms are either statically set up in the source code or at boot time, a police change may induce a VM reboot, or a source code modification, inducing a whole recompilation. Although this approach is not relevant for regular VMs, the lightness of unikernel footprint, their fast boot times and their limited source bases promote them as candidates for fast VM recompilations. Thus, the framework takes benefits of these unikernel properties to enable a on-the-fly image generation and VM reinstantiation, following a security policy change.

IV. The demo

We propose a demo to showcase an implementation prototype of our framework, and analyze its behavior through different scenarios. We plan to demonstrate the feasibility of the unikernel dynamic generation, and illustrate how such unikernels may contribute to cloud infrastructure security. A visual medium illustrating the key concepts of our framework will also be provided.

Our prototype is implemented based on the MirageOS unikernel platform, which provides building tools and unikernel resources. Therefore, software dependencies are OCaml libraries that are obtained from the OPAM package manager. The exploitation environment is emulated through KVM hypervisor usage and ukvm monitor. Specifically for this demonstration, the management plane relies on the Jenkins continuous integration software product, to get a didactic way to launch scenarios and explore delay time metrics. As a unikernel application, we have implemented a web server based on the CoHTTP processing library. The security mechanism is an insertable authorization and authentication agent capable of inspecting CoHTTP requests. Its configuration represents the security policy we want enforce.

During the demo, we propose to explore several scenarios illustrating the performances regarding the unikernel image generation and exploitation, with and without a security mechanism. Later, we analyze the insertion and the removal of credentials in the security policy, as well as the insertion of new cloud resources covered by the security policy.

References

[1] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.

[2] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos. SDSecurity: A Software Defined Security experimental framework. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 1871–1876, June 2015.

[3] H. Kang, M. Le, and S. Tao. Container and microservice driven design for cloud infrastructure devops. In *Proceedings of IEEE International Conference on Cloud Engineering (IEEE IC2E)*, pages 202–211, April 2016.

[4] Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. Unikernels: Library operating systems for the cloud. *SIGPLAN Not.*, 48(4):461–472, March 2013.

[5] Maxime Compastié, Ruan He, Mohamed Kassi-Lahlou, Rémi Badonnel, and Olivier Festor. Unikernel-based Approach for Software-Defined Security in Cloud Infrastructures. In *Proceeding of 2018 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2018.