SERECA: Secure Enclaves for Reactive Cloud Applications

Stefan Brenner[‡], Nico Weichbrodt[‡], Dan O'Keeffe[*], Divya Muthukumaran[*], Sergei Arnautov[†]
Thomas Knauth[†], Rüdiger Kapitza[‡], Peter Pietzuch[*], Christof Fetzer[†]

[*]Imperial College London, United Kingdom, {prp}@imperial.ac.uk
[†]TU Dresden, Germany, {firstname.lastname}@tu-dresden.de
[‡]TU Braunschweig, Germany, {brenner,weichbr,rrkapitz}@ibr.cs.tu-bs.de

***Abstract*—We present the Secure Enclaves for REactive Cloud Applications (SERECA) Horizon 2020 project. SERECA is removing technical impediments to secure cloud computing, and encouraging greater uptake of cost-effective and innovative cloud solutions. The consortium develops *secure enclaves*, a new technique that leverages *security extensions in commodity CPUs* for cloud deployments, empowering applications to ensure their own security without relying on public cloud operators. To facilitate secure deployment and coordination of distributed applications, SERECA integrates with the Docker container engine.**

SERECA is validating its results through the development of two innovative and challenging industry-led use cases. The first use case concerns the monitoring of a civil water supply network, a critical infrastructure targeted by malicious attacks. The second use case concerns a commercial software-as-a-service (SaaS) application for analysing the performance of cloud-deployed applications. Such a service collects sensitive performance metrics about live usage, assets that must be protected from industrial espionage and other criminal activities.

I. Introduction

Cloud security is of immediate concern to organisations that must comply with strict confidentiality and integrity policies, including those supporting society's most critical infrastructures, such as finance, utilities and health care. More broadly, security has emerged as a commercial imperative for cloud computing across a wide range of markets. In particular, application vendors have legitimate concerns about the confidentiality and integrity of user data hosted in third-party clouds, with cloud providers struggling to give strong security guarantees that the data will be protected [9]. The lack of adequate security guarantees is a barrier to the broader adoption of cloud computing.

The **Secure Enclaves for REactive Cloud Applications (SERECA)** project is removing technical impediments to secure cloud computing, and encouraging and enabling greater uptake of cost-effective, environment-friendly, and innovative cloud solutions. SERECA focuses on a particularly important and rapidly growing class of applications whose protection in cloud deployments has received little or no attention to date. In stark contrast to traditional throughput-oriented, batch-processing cloud applications, this class is highly *interactive* and *latency sensitive*. Examples are cloud-hosted Internet-of-Things (IoT) applications, Cyber-Physical Systems (CPS), multi-player games, on-line business analytics, real-time data monitoring, and industrial control.

The innovative approach to cloud security pursued in SERECA leverages *secure CPU hardware*. Basing trust in hardware mechanisms offered by commodity CPUs enables a new generation of secure applications. The main technical challenges addressed in SERECA are to (i) integrate this new technology within a standard cloud platform; (ii) extend it for use in a highly distributed multi-data-centre computing environment; and (iii) efficiently develop and manage applications using the secure CPU hardware.

In essence, on the technical side, we set out to achieve the following goals:

1) **Substantially improve the state-of-the-art in cloud security for interactive, latency-sensitive applications** by developing innovative and effective mechanisms to enforce data *integrity*, *availability*, and *confidentiality* based on secure CPU hardware.
2) **Seamlessly integrate the new security features into the standard cloud stack** and its expected characteristics of *scalability*, *elasticity*, and *availability* so as to encourage easy application *migration* to the cloud without also compromising application *responsiveness* nor complicating application *management*.
3) **Convincingly validate and demonstrate the benefits of our approach** by applying it to realistic and demanding *industrial use cases*.

The remainder of the paper is structured as follows: In Section II we give a brief overview of alternative approaches to cloud security. Section III describes the approach taken by SERECA to protect the confidentiality and integrity of applications running in untrusted cloud environments. Section IV summarizes the challenges posed by our industrial use cases. Section V concludes the paper.

II. Existing Approaches to Cloud Security

A modern public cloud is home to a hardware and software stack (Figure 1, left) consisting of thousands of devices, millions of lines of code, and arbitrary number of unknown tenants. The hardware is typically a mix of different models of different hardware generations, purchased in bulk more for their price than for their reliability. The software includes large

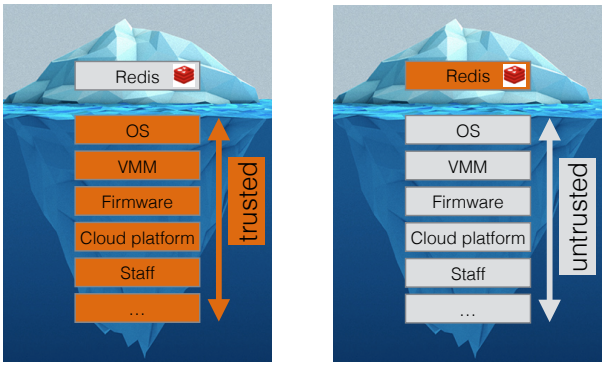

Figure 1. The Trusted Computing Base in established cloud offerings (left) comprises many layers of software, hardware, and "wetware". In SERECA, the cloud provider's hardware and software stack are no longer included in the Trusted Computing Base (right). This is possible due to recent security extensions (SGX) included with modern Intel processors.

frameworks, e.g., OpenStack, that are evolving rapidly and potentially harbor numerous exploitable bugs and configuration errors. The challenge for operators is to convince potential clients that it is safe to execute their applications and store their data in such a dangerous environment.

One approach taken by operators is to define a *Trusted Computing Base* (TCB) within the stack [1]. Typically, the TCB will include most of the basic middleware, operating system (OS), and networking facilities of the data centre, as well as its hardware platform. Establishing the credibility of the TCB amounts to verifying the correctness and security of a large and complex hardware and software system. Aside from the high cost of doing so continously as the hardware and software evolve, the goal of a truly "trustworthy" TCB has proven elusive [8], [10].

Even if it were possible to remove all bugs from the TCB, this alone will not ensure the security of cloud applications, since a malicious system administrator employed by the cloud operator could exploit their privileges to access customer data while the data are, for example, unencrypted in main memory or CPU registers.

An approach focused specifically on securing application data from access by both external and internal malicious agents is based on the use of *homomorphic encryption*, which is a technique intended to allow computations to be carried out directly on encrypted data [4], [7], [12]. The promise of this technique is that the unencrypted form of the data would never need to be present within the data centre, and so never exposed to attack. The technique also extends to the computations themselves, so that they too could be encrypted and thereby protected from exposure. Unfortunately, the goal of homomorphic encryption is proven as elusive as that of a trustworthy TCB [14], [16]. The computations for which the technique is known to work are currently quite limited, making their use in most normal applications infeasible. Moreover, homomorphic encryption is notoriously slow, making its use in reactive applications untenable.

A third approach being investigated for the cloud is the use of *specialised security co-processors*. Such processors consider the chip area as a trust boundary, treating everything outside as subject to attacks and potentially compromised. As in the case of homomorphic encryption, the instructions and data are stored encrypted in the memory. However, once read by the processor, they are decrypted and the instructions carried out on plain-text code and data. Because everything outside the chip can be tampered with, the chip never outputs plain-text, encrypting the data before writing to the bus. While secure processors provide good security guarantees, the notion of investing in specialised hardware is counter to the general principle of data centre "scale out", which advocates the use of large numbers of low-cost commodity components. Another disadvantage is that the use of such processors is not transparent to developers, because they need to specifically tailor their applications to leverage features provided by the hardware [11], [13].

III. SERECA Approach

SERECA advanced the state-of-the art in secure data processing in public clouds in a number of areas. First, we developed tooling to run existing applications with low-effort using Intel Software Guard Extensions (SGX). Second, SERECA enables the secure coordination of distributed applications, i.e., applications spanning multiple enclaves. Last, enclave applications are easy to deploy and use by security laymen due to their integration with the container engine Docker. We describe each aspect in more detail in the following sections.

A. System Support for Secure Enclaves

SERECA leverages the Intel SGX processor extension [6] to create a secure environment, called enclave, inaccessible to other components running on the same system. A major outcome of SERECA is the development of a toolchain to allow existing applications to use the SGX extensions simply by recompiling the application [2]. To this end, we extended an existing C library with SGX-specific features. We successfully applied our toolchain to create SGX-enabled versions of widely used open-source programs such as Memcached (an in-memory cache), Redis (an in-memory database) and the Apache web server. Expanding the set of supported applications is reasonably easy and only requires access to the application's source code. We currently support applications written in C and C++. We are planning to support more modern languages such as Rust and Go in the future.

Besides running entire applications within the enclave, we also investigated automated techniques to split applications into a trusted and untrusted portion. Only the trusted portion, manipulating sensitive data, is running inside the enclave. Minimizing the code and data part of the enclave is desirable as it reduces the amount of trusted code running in the secure environment. Further, the first generation of SGX hardware has a stringent limit on how much secure memory an application can use. Beyond the current limit of 96 MB the application may suffer dramatic performance degradations.

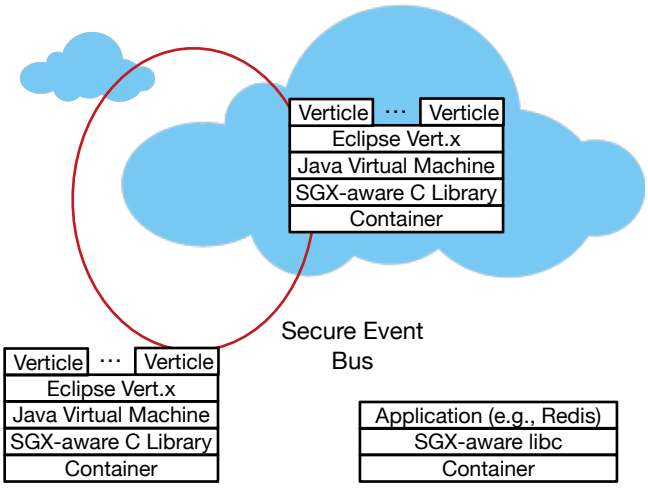

Figure 2. SERECA application runtime. Vert.x applications, a Java framework for reactive applications, run distributed and communicate via a *secure event bus*. Vert.x handles message routing and encryption. A modified C library enables applications to leverage the novel SGX processor extensions.

We also investigated potential attacks on enclaves. While SGX protects the integrity and confidentiality of the enclave's content, side channels and bugs included in the program running within the enclave may still be used to exfiltrate sensitive data. In particular, we looked at how an attacker may use synchronization bugs to subvert the enclave [15].

B. Support for Distributed Secure Enclaves

Besides supporting existing applications, SERECA focuses on *reactive applications* written from scratch in Java using the Eclipse Vert.x framework. Each Vert.x application is structured around the notion of a *verticle* – a unit of program logic communicating with other verticles by exchanging messages on an event bus (Figure 2). A challenging aspect thus far in the project has been to support Java applications within the secure environment. While running a simple Java Virtual Machine within the secure environment is possible, industrial-strength applications often require a particular, complex and highly-optimized JVM, such as the OpenJDK JVM. This led us to explore alternative ways to support Java applications. One possible path is to only run the security-relevant aspects of the Java application within the secure environment and interface the Java and non-Java parts with the Java Native Interface (JNI). We are still exploring the benefits and drawbacks of the mentioned alternatives.

An application consisting of distributed components must coordinate their action. Many recurring tasks of coordination like distributed locks, leader election or configuration management can be encapsulated in a *coordination service* of which ZooKeeper [5] is one example. To support the coordination of secure services, we created a secure variant of ZooKeeper [3] that protects the confidentiality of user data stored as part of the coordination service.

C. Cloud Architecture for Secure Enclaves

To deploy and manage applications securely SERECA advocates the use of *secure containers*. Containers are a lightweight alternative to virtual machines, offering many of the same benefits at a smaller resource footprint and runtime overhead. In particular, SERECA supports the popular container engine Docker. This allows to reuse solutions for orchestration and easy deployment of secure applications from the Docker ecosystem, for example Docker Swarm. Experts create secure versions of their applications with the help of SERECA's toolchain. The secured applications are packaged into containers to make them easily usable by a much larger audience of laymen users.

IV. Application Use Cases

The SERECA platform is evaluated with two industry-led use cases. The first use case transforms an application to monitor a public water supply into a cloud-hosted version. Hosting the application in a public cloud is appealing because the company responsible for monitoring the water supply lacks expertise to run and maintain the vast IT infrastructure required for this task. However, outsourcing into an untrusted cloud environment has been impossible so far due to concerns about data security, service availability and service isolation. To leverage the confidentialy and integrity guarantees provided by the SERECA platform, the application is re-architectured as a reactive Vert.x application.

The second use case enhances an existing cloud service to analyze the performance of Java-based cloud applications with strong security guarantees. Application performance analysis and monitoring is a core component of any cloud-hosted service. Customers can lose substantial revenue and reputation when services perform poorly. While a public cloud infrastructure can result in crucial cost savings, issues exist that prevent cloud deployments with today's technology. The monitoring system collects, processes and stores confidential, business-sensitive customer data in a public cloud environment. The SERECA platform will ensure that no unauthorized party has access to the customer's data. In addition, the posibility to opt-in to encrypt all data leaving and entering a SERECA application, ensures that data is secured while transiting between different instances of the performance monitoring application.

V. Conclusions

We presented the Horizon 2020 project SERECA. SERECA enables, for the first time, secure data processing in public cloud environments. We achieve this by utilizing recently available hardware extentions in commodity processors. The hardware extensions allow us to construct a secure environment independent of the cloud provider's hardware and software stack. SERECA enables existing applications to run within the secure environment with minimal effort. SERECA also facilitates the easy deployment and orchestration of distributed applications by integrating with the Docker container engine.

REFERENCES

[1] Trusted computing base. http://www.trustedcomputinggroup.org/.

[2] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'Keeffe, Mark L. Stillwell, David Goltzsche, Dave Eyers, Rüdiger Kapitza, Peter Pietzuch, and Christof Fetzer. Scone: Secure linux containers with intel sgx. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 689–703, GA, 2016. USENIX Association.

[3] Stefan Brenner, Colin Wulf, David Goltzsche, Nico Weichbrodt, Matthias Lorenz, Christof Fetzer, Peter R. Pietzuch, and Rüdiger Kapitza. Securekeeper: Confidential zookeeper using intel SGX. In *Middleware*, page 14. ACM, 2016.

[4] Craig Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, March 2010.

[5] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. ZooKeeper: Wait-free coordination for Internet-scale systems. In *Proceedings of the 2010 USENIX Annual Technical Conference (USENIX ATC '10)*, pages 145–158, 2010.

[6] Intel Corp. Software Guard Extensions Programming Reference, Ref. 329298-002US. https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf, October 2014.

[7] Seny Kamara and Mariana Raykova. Parallel homomorphic encryption. In *Workshop on Applied Homomorphic Encryption (WAHC '13)*, April 2013.

[8] Jonathan M. McCune, Bryan Parno, Adrian Perrig, Michael K. Reiter, and Arvind Seshadri. How low can you go?: recommendations for hardware-supported minimal tcb code execution. In Susan J. Eggers and James R. Larus, editors, *ASPLOS*, pages 14–25. ACM, 2008.

[9] Peter Mell and Timothy Grance. The NIST definition of cloud computing. Technical Report 800-145, US National Institute of Standards and Technology, September 2011.

[10] Ahmad-Reza Sadeghi, Marcel Selhorst, Christian Stüble, Christian Wachsmann, and Marcel Winandy. Tcg inside?: a note on tpm specification compliance. In Ari Juels, Gene Tsudik, Shouhuai Xu, and Moti Yung, editors, *STC*, pages 47–56. ACM, 2006.

[11] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. Aegis: Architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th Annual International Conference on Supercomputing*, ICS '03, pages 160–171, New York, NY, USA, 2003. ACM.

[12] M. Tebaa, S. El Hajji, and A. El Ghazi. Homomorphic encryption method applied to cloud computing. In *Network Security and Systems (JNS2)*, pages 86–89, April 2012.

[13] David Lie Chandramohan Thekkath, Mark Mitchell, Patrick Lincoln, Dan Boneh, John Mitchell, and Mark Horowitz. Architectural support for copy and tamper resistant software. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS IX, pages 168–177, New York, NY, USA, 2000. ACM.

[14] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[15] Nico Weichbrodt, Anil Kurmus, Peter R. Pietzuch, and Rüdiger Kapitza. Asyncshock: Exploiting synchronisation bugs in intel SGX enclaves. In *ESORICS (1)*, volume 9878 of *Lecture Notes in Computer Science*, pages 440–457. Springer, 2016.

[16] Zhenfei Zhang, Thomas Plantard, and Willy Susilo. Reaction attack on outsourced computing with fully homomorphic encryption schemes. In Howon Kim, editor, *ICISC*, pages 419–436. Springer, 2011.