The Potential of Trusted Execution Environments for Banks

Bruno Bonati

On its foundation, banking is an industry of trust. Therefore, banks and the financial services industry require specific IT solutions to provide secure and confidential services to their clients. The IT landscape of such institutions must ensure the highest possible trustworthiness for the entire banking application portfolio.

However, there are more security problems in banking IT systems than managers are aware of. This paper describes how to conquer these threats fundamentally.

About the Author

Bruno Bonati led the IT and Operations department at Credit Suisse for ten years. After retiring from Credit Suisse in 2005, Bruno worked as an independent consultant in IT and banking. He mandated several boards on companies such as Tieto Helsinki/Stockholm and Swisscom IT Services.

Bruno has further served as the chairman of the Zuger Kantonalbank for ten years. Currently, he is a member of the board and head of the audit and risk committee of ELCA, a large privately owned IT company in Switzerland specializing in software solutions and IT products. Bruno is the co-author of the book *Managed Evolution (A Strategy for Very Large Information Systems)* and advises companies with future-oriented IT solutions. For any questions, please get in touch with Bruno Bonati at bruno.bonati@bluewin.ch

1 The Invisible Problem

Today's Banking IT architecture mainly consists of two layers: an application layer and an infrastructure layer, see figure 1. All business applications in the system form the banking application portfolio, which is partitioned into banking application portfolio which is portioned into banking application domains such as *Finance, Investments & Sales*, *Cash and Asset Operations*, and *Trading and Markets*.

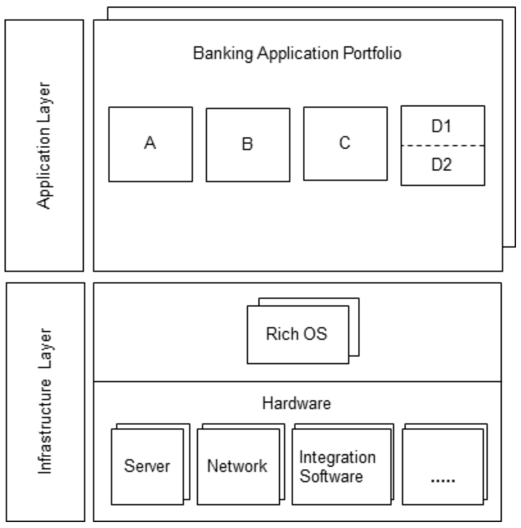

Figure 1: The Big Picture of today's Banking IT architecture.

The infrastructure layer comprises IT elements such as hardware products, networks, service integration and management (SIAM), servers, data centers, application development environments, integration software, data management and databases, and last but not least, general-purpose operating systems.

The *Operating System* (OS) acts as an intermediary between all programs (banking applications) and the computer hardware. For that reason, the general-purpose operating system is qualified as Rich OS. The Rich OS is often the leading cause why the trustworthiness of the entire banking application portfolio is not guaranteed. Today's banking IT managers expect operating systems commonly used in their infrastructure to satisfy increasing security, maintainability, and quality requirements. However, they were never designed to address these challenges.

An operating system consists of two modes. An unprivileged mode called userspace, where applications are executed, and a privileged mode called the kernel, see figure 2. The kernel is the most critical part of the system. Today's commonly used kernels are monoliths, consisting of millions of code lines, and all components have access to all resources, even if it is not necessary. These architectural characteristics lead to the most critical security problems in an operating system.

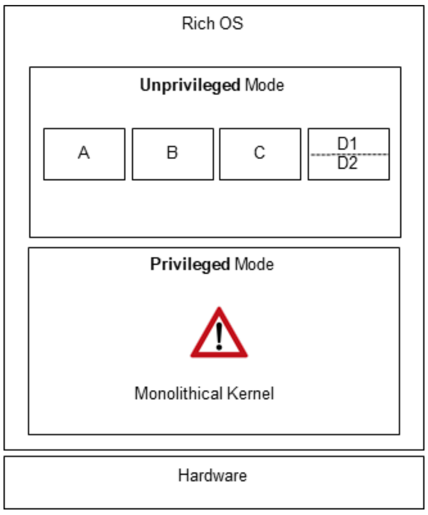

Figure 2: The Rich OS.

Essential but complex software, such as device drivers and protocol stacks, have complete access to all the hardware and can run any instruction the machine is capable of executing.

In monolithic kernels, the likelihood of an error is high because of its vast complexity. The impact of an error is also extremely high since every line of code that runs in the kernel is critical, and an exploit results in the takeover of the whole system with fatal consequences. In short, the attack surface is far too big, which results in massive risk for the banking industry.

The security problems in banks are becoming more and more urgent for two reasons: on the one hand, cyber-attacks increase in numbers and are becoming growingly sophisticated and often impossible to detect with conventional reactive security products.

On the other hand, the banking application portfolio is constantly growing with the development of new banking services such as omnichannel management; banking applications and data executed by an outsourcing provider; the execution of product or customer data from different contributors cooperating on an open banking platform; new products; integration of blockchain assets, etc.

Developing and operating products or business applications on top of monolithic operating systems with such a large attack surface requires a constant evaluation of security issues, an effort with high costs and consequently high-efficiency potential. Unfortunately, all security measures taken on the application level (management of access rights, passwords, etc.) and all security features offered by the OS are insufficient to provide the highest possible security and protect against attacks from external hackers or internal adversaries.

The minimal requirement is to ensure the highest possible security for the security-critical components of the banking application portfolio. This requirement is not solved by today's banking IT.

Theoretically, there would be two ways to fulfill this minimum requirement. The most obvious would be to use an operating system with a security-first design. But currently, there is no operating system with these features in use in the existing banking IT landscape.

The realistic alternative is to utilize *Trusted Execution Environments* (TEE) with the following approach: First, define the security-critical banking applications out of the portfolio. Then, in a second step, select the critical components of these applications and execute them with the highest possible security on TEE appliances.

2 Trusted Execution Environment - part of a modern Banking IT architecture

The main goal of *Trusted Execution Environments* (TEE) is to protect critical code and data during runtime. TEE appliances are secure, integrity-protected processing environments running on tamper-proof hardware. TEE technology aims to provide a massive range of functionality. At the same time, the technology should meet the requirements of software developers and service providers who care about privacy, authentication, validation, manageability, attestation, ease of use, and all the other security aspects.

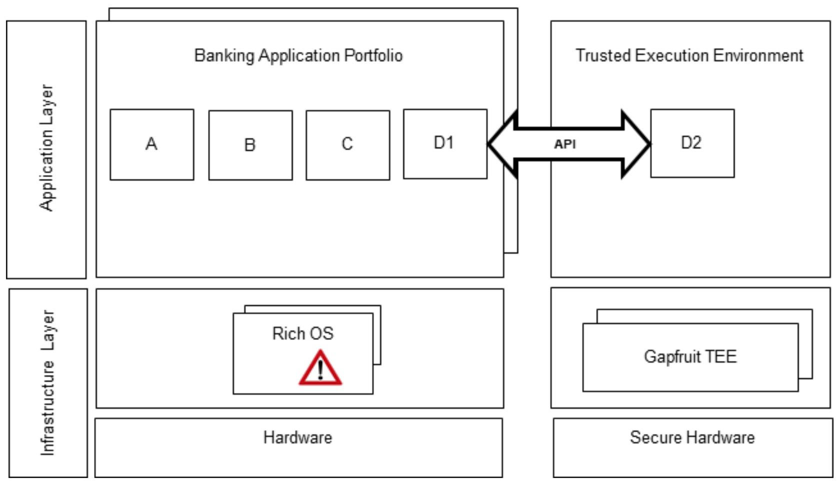

Figure 3: A future-proof modern banking IT system.

As a future-proof modern banking IT system, we have untrusted applications running on a cluster of Rich Operating Systems and trusted applications executed on Trusted Execution Environments, see figure 3. TEEs enable significantly higher levels of security for banks and financial institutions by executing trusted applications within physically and logically separated TEE appliances that are accessed only through defined interfaces from the already existing banking infrastructure. TEE appliances perform secure execution of online transactions, verification of compliance regulations, confidential multi-party computation, etc. In short terms: Wherever the risk of a compromise results in significant financial loss.

3 Gapfruit TEE

Gapfruit works closely with Hardware Security Modul (HSM) manufacturers to form a hardware/software co-design on top of their secure and tamper-proof hardware. Gapfruit TEE is the toolkit that these HSM manufactures use to build TEE appliances. The toolkit provides a high-security microkernel operating system with different runtimes.

On the one hand, Gapfruit TEE provides the highest possible security properties. On the other hand, its flexibility enables integrating the TEE appliance within any banking IT architecture at a low cost. Gapfruit TEE bridges seemingly incompatible opposites: It is an adaptable fortress, although an adaptable fortress is a contradiction per se.

The following sections explain how to significantly boost the trustworthiness of the entire banking application portfolio and integrate Gapfruit TEE into existing banking infrastructures.

3.1 Trustworthiness through Gapfruit TEE

The main goal is to prove its trustworthiness by guaranteeing the integrity and confidentiality of banking applications and critical information during runtime. The fundamental prerequisite for the proof of trustworthiness for any application is to gain absolute control over all software stacks in the underlying operating system. The core concepts to fulfill this requirement is to isolate, govern dependencies, and simplify. With this approach, you always know the set of software components that are critical to its security. The following sections will explain how Gapfruit uniquely achieves these core concepts.

3.1.1 Strong Isolation

A SLICE represents a *Secure and Light Instance of Contained Enclave*. SLICEs are a USP of Gapfruit TEE to guarantee strong isolation. Contrary to sandboxes and enclaves, which isolate only in one direction, the microkernel enforces the unique isolation properties in both ways (outside in and vice versa). More information: gapfruit.com/technology.

SLICEs contain one or more components such as device drivers, protocol stacks, multiplexers, and applications. Everything runs in unprivileged userspace. A further differentiator is the self-healing mechanism: if one of any components gets corrupted by a bug or an attack, the fault is restricted to the broken component and can be detected and recovered.

Imagine a well-designed ship: If one section springs a leak, the crew isolates this section from the other compartments to protect the whole vessel from sinking.

3.1.2 Governance of Dependencies

The governing of dependencies enables you to know the trust relationship between each part of the system. Relationships between all parts of Gapfruit TEE are tightly structured. As with real-life relationships, there are different dependencies. With the following three methods, we govern these dependencies, enabling provable trustworthiness. Gapfruit distinguishes three types of dependencies:

- Resource distribution is the first type of dependency where a parent provides resources such as RAM and CPU to its children. Further, the parent establishes access to services that other SLICEs may provide, which brings us to the second type, the service topology.
- SLICEs are inter-connected with each other in a service-oriented architecture where a client depends on a server. The configuration of these inter-connections is called topology. The topology inherently governs the access control to the different services. The underlying technique is called capability-based security.
- The final type of dependency is about the software supply chain. Gapfruit provides a powerful package manager that makes software dependencies reliable and reproducible in a secure way.

3.1.3 Simplify

The isolation properties and the governance of dependencies make it possible to identify the set of software components critical to its security and which need to be trusted. The key is to keep these vital components as simple as possible to be verified for correctness.

3.2 Flexibility

Integrating products into a banking IT system has a high impact on the *Total Cost of Ownership* (TCO).

The integration of Gapfruit TEE into all infrastructures is straightforward, with an open and simple-to-use API. Existing applications run inside Gapfruit TEE without or with minimal modifications. Application developers do not need to deal with complex concepts such as attestation or the many enclave frameworks that exist for e.g., Intel SGX.

4 Areas of Application in Banking

Each player in the banking industry has to define applications that fulfill the greatest possible security requirements. Players are:

- Single banks with inhouse development
- Single banks using standard software (such as Avaloq or Finnova)
- Banks outsourcing its banking IT or parts of IT
- Insourcers providing banking IT services to a bank
- Banking software producers of standard software as well as customized software
- System integrators
- Players of the banking industry exchanging data with each other
- Third-party providers

The following chapters describe some application areas in the banking industry as well as possible concrete use cases.

4.1 Automation with Increased Security

As a general rule of thumb, the older a banking service is, the better it is automated. For example, compliance is a field that is still predominantly done on paper. Compliance processes are really expensive and thus have a high return on investment if they are automated as much as possible. With Gapfruit TEE automating these compliance processes go hand in hand with increasing security.

With Gapfruit TEE, the potential to automate any banking processes increases. As a further example, credit approval processes can execute critical security verifications within a TEE, such as applying the credit authority order and the credit criteria automatically.

How do you prove the trustworthiness of a decision?

4.2 Omnichannel Management

Omnichannel Management means that each channel (mobile, laptop, customer advisor in a bank) has access to the other. The processes from the frontend systems to operations and the IT backend are continuous and uninterrupted. The question for the banking IT manager is whether the interaction of the various channels with each other and with the operations guarantees the highest possible security.

How do you guarantee that an incident of one channel wont affect others?

4.3 Outsourcing and Insourcing

When it comes to outsourcing workloads, providers, as well as tenants, face many security challenges. Often, providers run services from many tenants on shared infrastructure. Generally, the tenant and the provider have the same interests. The provider wants to assure security to the tenant. And the tenant vouches for the security and privacy of the personal data to the banking client. To guarantee the protection of personal information, banks would like to execute selected critical applications in a TEE. On the other hand, the outsourcing provider would like to run some of the clients' applications in a TEE to ensure integrity and confidentiality.

How strong is the separation between the workloads of the different tenants?

4.4 Standard Banking Software

This group has the same potential to increase the security by TEEs as the single bank developing the applications in-house. For example, Avaloq operates on LINUX provided by Red Hat. TEEs are currently not part of this infrastructure.

How do providers of standard banking software guarantee the integrity and confidentiality of the customers' data?

4.5 Integration of Solutions into Core Banking System

Many midsize banks do not develop applications in-house. The architecture of the application layer consists of the core banking system with many subsystems. System integrators unite these subsystems with the core banking system - often with their developed software.

How are requirements for the highest security fulfilled both for applications and data?

4.6 Confidential Computing

Players of the banking industry are aiming to gain market insights without exposing their data to their competition. Players may be regulators (e.g., FINMA), interbank service providers (e.g., SIX), central banks (e.g., SNB), research institutes, and of course the individual banks. As an example, a research institute organizes an annual mortgage study with different banks.

How do you guarantee the confidentiality of sensitive business and banking data?

4.7 Open Banking

Open banking is a new business model that refers to banks' accessibility and parts of their data to third-party providers. Public APIs enable the use and aggregation of information, operations, and transactions on bank accounts triggered by third parties, banks, or non-banks.

How do you fulfill the entirely new security requirements this business model brings?

5 Why Implement Gapfruit TEE in Banks now?

Rising complexity on all levels of the banking architecture, from infrastructure to the application layer, results in an increased attack surface. The resulting vulnerability invites hackers and leads to increased internal threats. With the constantly growing portfolio, banking IT managers should execute more and more applications in a trustworthy manner. Therefore, it is urgent to implement Gapfruit TEE now.

6 About gapfruit

Gapfruit is a deep-tech company based in Switzerland with a proven track record in systems security, software engineering, and innovation. The founding team developed a military-grade operating system fulfilling the requirements set by national governments and security agencies across the world for ironclad security. With this expertise, Gapfruit brings scientifically recognized academic research to real-world products for today's and future challenges. gapfruit.com