Improving Security and Practicality of Modern Trusted Execution Environments

Background and research vision. Cloud computing underpins critical services in healthcare, finance, and AI. Yet, it faces a constant threat of cyberattacks and data breaches. In 2024, these cybercrimes exposed over 280 million records in the US's health sector alone. Data breaches are not only financially devastating for the organizations employing cloud services, costing them up to $4.4 billion in remediation on average in 2025, they also erode customers' trust in cloud providers.

To regain customers' trust, cloud providers are now strengthening the security of their infrastructure. A promising approach is to isolate the customers' applications from the infrastructure to minimize the risk of compromises from permeating to customers. Particularly, providers are increasingly adopting specialized hardware from compute vendors (e.g., Intel, AMD). These are new processors with secure primitives built into the silicon that can create a secure "bubble" on the platform, called a Trusted Execution Environment or TEE (Figure 1), which protects sensitive data and programs from the rest of the platform, including compromised cloud administrators.

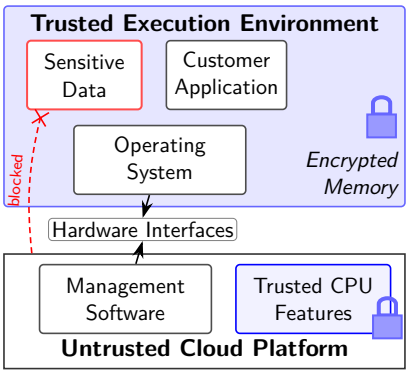

Figure 1: Modern TEE designs include the operating system within.

While several TEE designs have been proposed in the community, the most successful approach has been to isolate an application and all its supporting software into a single TEE. This approach, on the one hand, has enabled companies to migrate cutting-edge applications, such as large language model inference, into TEEs with reduced performance and engineering costs. On the other hand, it requires placing a significant amount of code in the TEE, on which the security guarantees for the customer's application and data depend. Notably, this code, also known as the trusted computing base (TCB) of the application, now includes the operating system (OS), and widely-used OSes like Linux and Windows often consist of several million lines of code that are known to have thousands of vulnerabilities. In summary, placing feature-rich OSes in the TEE eases adoption, but also increases the risks of vulnerabilities.

My vision is to tame these risks in the TEE OSes while further enabling them to provide stronger security and better functionality to TEE applications. My research will focus on two main directions: (i) *compartmentalizing the TEE OS* to reduce the exploitability of vulnerabilities, and (ii) *designing OS-assisted mitigations* for novel threats such as side-channel attacks from a compromised host platform.

Expected impact. The outcome of this proposal is not only more trustworthy cloud infrastructures for running sensitive workloads in healthcare, finance, and AI, but also the broader adoption of TEEs for data protection, thanks to improved usability. The research also presents numerous opportunities for interdisciplinary impact, given the growing need for secure processing of sensitive data in the cloud. Specifically, I plan to work with the Faculty of Medicine to apply the developed techniques for securing sensitive medical data processing workloads, which will also serve as a means to evaluate my research.

Taming TEEs' large TCB with compartmentalization. The large TCB introduced by OSes hinders efforts to prevent and predict bugs that allow attackers to compromise the security of the TEE. Worse, feature-rich OSes expose numerous endpoints (e.g., for storage and external communication) through which information can be exfiltrated from the TEE, in the event of a compromise.

Prior research has tackled these challenges in three broad ways: (i) comprehensively testing the TEE code to detect bugs and vulnerabilities, (ii) reducing the size of TCB by removing extraneous functionalities and endpoints, and (iii) compartmentalizing the large codebase into smaller, isolated compartments to restrict the impact of attacks from spreading to the entire TEE. Unfortunately, (i) and (ii) are prone to missing vulnerabilities in complex TCBs and do not generalize to multiple OSes and applications.

Instead, my focus is on (iii), improving compartmentalization techniques to make it highly challenging for attackers to exfiltrate sensitive information. Building on my experiences in compartmentalization, I

will explore compartmentalization methods to establish in-OS compartments based on their functionalities and likelihood of compromise (Figure 2). For instance, the code that communicates between the untrusted and trusted worlds is highly vulnerable to attacks and thus should be placed inside an isolated component. Furthermore, adapting compartmentalization to a feature-rich OS would require significant engineering efforts, which I plan to alleviate by developing automatic tools for compartmentalization.

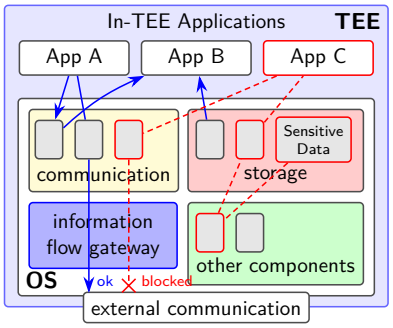

Figure 2: Compartmentalization and sensitive data tracking in the in-TEE OS to minimize exploitability.

Compartmentalization by itself is sometimes insufficient to prevent information leaks; an attacker can exploit the in-TEE communication between compartments (red, dashed arrow in Figure 2). To address this, I plan to track how sensitive data travels (or its information flow) across applications and the OS compartments using whole-system provenance tracking, a technique pioneered by Thomas Pasquier and Margo Seltzer at UBC. I will apply provenance tracking to identify and prevent operations that could leak sensitive data, while allowing safe operations to proceed. This approach avoids functionality limitations and improves compatibility with a broader range of applications.

Practical side-channel mitigation with OS support. Side channels are a critical yet persistent class of vulnerabilities at the software-hardware boundary of TEEs, allowing attackers (e.g., in a compromised cloud platform) to leak sensitive information from sensitive computations even without direct access. For instance, even though the cloud platform cannot read a TEE's memory, it can control and observe the order in which a TEE application accesses memory locations. These memory access patterns have been shown to reveal the image being processed by the TEE and even its secret encryption key.

One way to mitigate such attacks is for the cloud provider to secure their resource management mechanisms, for instance, by eliminating dynamic and flexible memory management across multiple co-located tenants. However, such an approach not only involves substantial engineering costs for the provider but also raises concerns about the efficiency of resource utilization. The less intrusive approach is to modify the resource usage patterns generated from the sensitive application, such that they can no longer be used to infer secrets. Unfortunately, existing solutions require a significant software rewrite, which is often not scalable in terms of engineering and performance costs.

I will develop OS-assisted solutions to eliminate side-channel leaks via an application's resource usage patterns, all without requiring modifications to applications. During my PhD, I developed a minimal, proof-of-concept OS that mitigate side channels through its resource management, which can secure the resource usage patterns of simple cloud applications. I plan to integrate the scheme into a production-scale, feature-rich OS like Linux to support a larger class of applications, which will require addressing several fundamental challenges in security, performance, and engineering costs due to scale.

First, in addition to the applications' resource accesses, the OS introduces its own operations, potentially introducing new or amplifying existing side channels. However, protecting access to all resources in complex OSes will be prohibitively expensive. To efficiently protect only access to sensitive resources, I expect the sensitive data tracking scheme proposed in the previous direction to be useful. Moreover, modifying a complex OS's resource management may break compatibility with existing workloads. To address this, I will explore synergies with Alexandra Fedorova's work on application-aware memory management to provide non-intrusive extension points for secure resource management policies.

Second, security trade-offs are inevitable when implementing security measures at scale. To provide a principled foundation for the OS-assisted side-channel mitigations scheme, I plan to explore formal security modeling and analysis, aiming to establish a user-controllable threshold where information leakage is tolerated while maintaining security. In this regard, Aastha Mehta, who has profound experience in the formal modeling of side-channel leaks, will provide valuable insights as my postdoc supervisor.