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, even if the rest of the system, even the cloud administrator, is compromised.

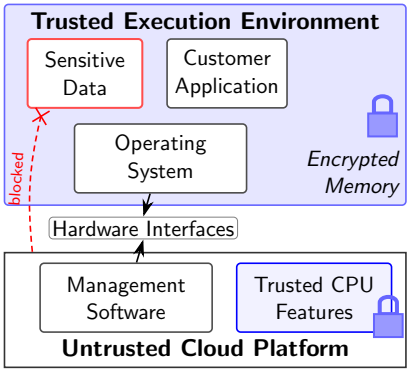

Figure 1: Modern TEE designs include the operating system within.

After the initial exploration of TEE designs, the community has settled on the idea of isolating the application and all supporting software within a single TEE. This design has enabled companies to migrate cutting-edge applications, such as large language model inference, into TEEs with reduced performance and engineering costs. Achieving this requires a significant amount of code to be placed inside a TEE. This code, on which the security guarantees for the customer's application and data depend, is also known as the trusted computing base (TCB) of the application. Notably, the TCB now includes the operating system (OS). Widely-used OSes like Linux and Windows often consist of several million lines of code that are known to have thousands of vulnerabilities. As a result, 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 the TEE applications. Towards this end, 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 to TEE applications, such as side-channel attacks from a compromised host platform. The result of the proposal is not only more trustworthy cloud infrastructures on which sensitive workloads can run securely, but also wider adoption of TEEs for data protection, thanks to the improved usability.

Taming TEEs' large TCB with compartmentalization. The large TCB 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 leaked, allowing sensitive data to be easily exfiltrated out of the TEE. 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 in individual compartments 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 in-TEE compartmentalization mechanisms 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. Adapting this to a feature-rich OS would require significant engineering efforts, which I plan to alleviate by developing automatic compartmentalization tools.

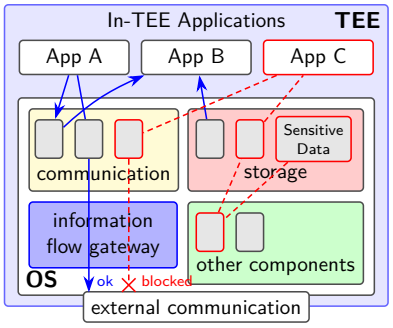

Figure 2: Compartmentalization and sensitive information flow tracking in the in-TEE OS.

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 across the OS compartments (or its information flow). To this end, the work on whole-system provenance analysis, led by Thomas Pasquier and Margo Seltzer at UBC, is particularly promising. Whole-system provenance analysis techniques record all resource accesses made by all applications within the system, allowing them to pinpoint the origin (or provenance) of any resource about to be accessed. I plan to incorporate these techniques into my research to identify and prevent operations that could leak sensitive data while allowing safe operations. This avoids limiting functionalities and renders the solution compatible with a broader range of applications.

Practical and principled side-channel mitigation 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 a compromised cloud platform cannot access a TEE's memory, it can control and observe the order in which a TEE application accesses memory locations during its computations, which is sufficient to reveal secrets in many applications. For example, such attacks have been shown to reveal the image being processed by the TEE or, more critically, its secret encryption key. One way to mitigate such attacks is for the cloud provider to secure the memory management mechanisms in their platforms, 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 on cloud platforms. The second, 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.

~~My proposal explores OS-assisted solutions toward making the second approach more practical and secure. The in-TEE OS can control the resource accesses of applications, allowing it to eliminate side channels from resource usage patterns, all without requiring any modifications to applications.~~ During my PhD, I developed a minimal, proof-of-concept OS with a side channel-eliminating resource management scheme, which can ~~only~~ 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, the complexity of production-scale OSes makes protecting all resource accesses performed by the TEE prohibitively expensive in terms of performance cost and engineering efforts. ~~To this end, I expect the in-TEE sensitive information flow tracking scheme in the previous proposal to be useful. With the information~~ on which compartment is handling sensitive data at the moment, the OS is able to not only efficiently protect the accesses but also determine the best approach to eliminate side-channels based on the type of resource being accessed.

Second, security trade-offs are inevitable when security is applied at scale; however, previous research often achieves them through fragile heuristics that lack a solid foundation. To provide a principled approach to OS-level side-channel mitigations, I plan to explore formal security modeling and analysis, aiming to establish a user-controllable threshold where information leakage is tolerated while maintaining

security. Aastha Mehta, who will be my postdoctoral supervisor, has expertise in applying differential privacy, a formal model of information leakage, for side-channel mitigation in the networking domain. Thus, her experience will be of tremendous help in pursuing this research direction.

Finally, applying protections to resource accesses implies significant changes to how the OS manages its resources, which could easily break compatibility with existing workloads. A way forward is to develop mechanisms that extend OS resource management in a non-intrusive manner. To achieve this, I plan to explore synergies with the work on application-aware memory management by Alexandra Fedorova. These systems provide extension points to the OS, upon which custom resource management policies for specific applications can be implemented. Thus, the side-channel mitigation scheme would only affect the resource management of sensitive applications, while leaving the rest of the system untouched.

Teaching and interdisciplinary collaborations My proposed research will involve many core concepts that are the focus~~es~~ of many courses at UBC: OS resources management for CS436A and CS508, side-channel mitigations and kernel compartmentalization for CPSC538M and CPSC538P. I am looking forward to contributing my expertise to the teaching of these modules. To this end, I have already served as a guest lecturer on CPSC538M, presenting my work on software compartmentalization.

At the same time, the proposed research on enhancing the security of cloud infrastructures presents numerous opportunities for collaboration, given the increasing need for secure processing of sensitive data in the cloud. For instance, I plan to initiate interdisciplinary research with the Faculty of Medicine to apply techniques developed in my research to secure their sensitive data processing.

1 Appendix: Guide (REMOVE IN FINAL)

The proposal should be written in clear, non-technical language that enables a non-specialist to understand the overall content and significance of the work. Members of the Killam Postdoctoral Fellowships and Prizes Committee are from abroad and range in disciplines, and may not have expertise in your area of study.

Although the fellowship may be used to extend or expand upon doctoral work, it must be made clear that you are not intending to use the award to wrap up a thesis. While it is expected that a postdoctoral fellow will be taking the next step beyond the PhD thesis, you must differentiate clearly between the postdoctoral project and the thesis research. Feel free to include hyperlinks to provide links to additional information.

As applicants must make UBC their base, it is essential, particularly for those whose primary research materials are located elsewhere, to indicate the travel involved, its destination, and duration. You should describe how you will deal with the remoteness of the primary materials.

The adjudication committee is very interested in "fit" with the selected UBC department or unit and the university's research programs. You must provide information on how your research relates to that of specific campus programs and advisors. If a colloquium is envisioned, a possible title should be proposed. If you visit the classes, please suggest which ones. Since interdisciplinarity is often a valuable dimension (the Killam Trusts declares that a candidate shall not be "a one-sided person"), specific details on proposed interdepartmental connections are welcome.