HARDWARE-ASSISTED SECURE EXECUTION ON REMOTE, UNTRUSTED SYSTEMS

Securing code and data in a cloud computing environment is a quintessential problem. Cloud-based remote systems are untrusted, which is a challenge for applications dealing with sensitive data. A malicious adversary can aim to subvert the execution flow to obtain sensitive data. We show that traditional, license-based, encryption-based software-only solutions using a provable, strong, unbreakable cryptographic primitive are vulnerable to a malicious entity with admin-level privileges. A control flow bending, or CFB, attack does not interfere with the functioning of the defense mechanism but rather with its outcome, rendering the entire defense method ineffective. The industry has attempted to address this challenge using trusted execution environments (or TEEs), where the hardware ensures the code's confidentiality, integrity, and freshness. However, such solutions have faced resistance in adoption primarily due to two key limitations: one, high performance overheads, and second, limited or no accessibility to standard features due to the strict security requirements.

In this thesis, we focus on alleviating both these limitations by enabling secure execution of applications in a TEE with minimal performance overheads, along with access to standard cloud features such as access to a secure file system, a mechanism to control the execution, and live migration. We use Intel Secure Guard eXtension, or Intel SGX, a TEE solution from Intel for our experiments. To solve the first key limitation, we propose a novel mechanism to prevent CFB attacks on applications by *handicapping* a binary by removing a small set of important functions. The binary can only execute when the execution is validated by a secure code inside SGX, either using a license file or a key. We propose novel methods to identify the “important” functions in the application and securely patch the handicapped binary. Subsequently, we extended the idea to enable a low-cost execution control mechanism in SGX, which provides a developer with complete execution control on remote machines.

To solve the second key limitation, we first design a novel secure filesystem for SGX based on a thorough characterization of representative secure applications. The secure filesystem can be integrated with any SGX application without source code modifications. Finally, we combine all the previous contributions to enable the live migration of large SGX enclaves from one machine to another. Prior work in this space took several minutes to migrate a large SGX enclave from one machine to another; we reduced the total downtime to a few seconds.