Secure execution of unmodified applications on an untrusted host

Andrew Baumann Marcus Peinado Galen Hunt
Microsoft Research

Krystof Zmudzinski Carlos V. Rozas Matthew Hoekstra
Intel Labs

Today's commodity hardware architectures use a hierarchical privilege model, forcing applications to place complete trust in system software (OS, hypervisor, bootloader, BIOS, etc.) and the administrators and management tools that control it. In practice, this means that application users must trust all of that software, its administrators, and any malware that may have subverted the system not to access or modify private data handled by the application, nor to change the application's behaviour. This trust model is increasingly inadequate in the era of large system software stacks, pervasive malware, and distributed applications. Particularly in the latter case, whenever an application component runs remotely (for example, in scenarios such as trusted network services, remote desktop infrastructure, or client-side components of an online service), unknown hardware, software and people must be trusted not to compromise the application. In this work, we show how proposed extensions to the x86 architecture can be employed to run existing unmodified Windows applications efficiently on remote hardware, securing the application's data while removing all trust from the system's software and administrators.

Intel software guard extensions (SGX) [3] are a set of new instructions and memory access changes to the Intel architecture. SGX allows a process to instantiate a protected region of its address space known as an *enclave*, which provides hardware-enforced confidentiality and integrity protection for data and code against potentially-malicious privileged code (including OS and VMM) or hardware attacks such as memory probes. While SGX was designed to enable trustworthy applications that protect specific secrets (e.g., decryption keys) by placing portions of their logic inside enclaves, we instead run entire legacy applications that have not been written to support SGX. Besides the direct technical challenges (enclave memory footprint, execution model limitations, etc.) in achieving this, the biggest security concern arises from "Iago attacks" [2], where a malicious operating system subverts a protected application by exploiting the application's reliance on results of system calls.

The interface between modern applications and operating systems is so complex that we do not seek to protect it. Instead, we rely on a library operating system (LibOS) [4], which executes in the enclave alongside the application and implements all the APIs on which it depends. Beneath the LibOS lies a trusted runtime, which implements core OS primitives inside the enclave. Components of our trusted runtime include a memory allocator and address-space region manager, user-mode thread scheduling and synchronisation, and an encrypted and integrity-protected file system. The enclave must still call out to the untrusted host OS for system services such as committing memory pages and performing I/O, but we have carefully designed this interface to be very narrow, with only a handful of calls, and to permit mutual distrust of host and guest, using the trusted runtime to monitor the correct behaviour of the host. This allows us to ensure confidentiality and integrity of application data; a malicious OS can deny service to the enclave, but cannot trick the enclave into divulging its secrets nor executing incorrectly. Conversely, the host has full control over resource allocation and may protect itself from a malicious guest.

SGX also includes features for CPU-based remote attestation [1] that enable a remote user to verify cryptographically that the software they selected and configured executes as specified on a processor implementing SGX. Together with our LibOS and trusted runtime, these provide a user with an end-to-end guarantee on the security of their application without placing any additional trust in the remote computer or its software.

As an experiment, we have implemented a research prototype based on an emulator for the SGX instructions and protection mechanisms. One limitation of our prototype is that the host OS is trusted to implement the file system, but other key security protections are already implemented. We will demonstrate unmodified Windows applications running within this environment.

References

[1] I. Anati, S. Gueron, S. P. Johnson, and V. R. Scarlata. Innovative technology for CPU based attestation and sealing. In *2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.

[2] S. Checkoway and H. Shacham. Iago attacks: why the system call API is a bad untrusted RPC interface. In *18th ASPLOS*, Mar. 2013.

[3] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. Innovative instructions and software model for isolated execution. In *2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.

[4] D. E. Porter, S. Boyd-Wickizer, J. Howell, R. Olinksy, and G. C. Hunt. Rethinking the library OS from the top down. In *16th ASPLOS*, pages 291–304, Mar. 2011.