SCONE, a Secure CONtainer Environment for Docker that uses SGX to run Linux applications

Razieh Eskandari

May 22, 2019

1 Problem

Recently, container-based virtualization has emerged with the rise of new technologies such as cloud and micro-service architectures. However, the container's security remains a great challenge. For the security-sensitive applications running on an untrusted cloud environment, the operating system, hypervisor, BIOS, and software cannot be trusted.
One possible solution would be through the use of Intel SGX hardware built into SkyLake chips as a Trusted Platform Module (TPM), in which developers could create a hardware-enforced trusted environment to protect their data and code from confidentiality and integrity violation attacks. Intel SGX adds 18 new instructions to the x64 instruction set such that anapplication could create the trusted memory, named enclave.
Enclave is a Trusted Execution Environment (TEE) that blocks access (including super-user, OS, VMM access) to the authorized code and data by CPU access control. Developers can define and partition applications into untrusted and trusted (enclave) parts. In the trusted part (enclave), certain instructions such as system-calls are disallowed to ensure maximum security. An enclave's virtual address space is not accessible by an outside code. Also, its integrity and confidentiality is protected. In addition, the integrity checking for the enclave's memory content is done during the start-up time and remote inter-platform attestation.
Although, SGX provides a good security mechanism by reducing Trusted Computing Base (TCB) to hardware and enclave, using SGX faces the following challenges:

1 Enclave Page Cache (EPC) is limited to 64MB or 128MB. For application larger than EPC size, swapping EPC and untrusted DRAM is very costly, since it involves many encryption/decryption routines to ensure enclave security.

2 Putting the application with all of its dependent library and library OS in the enclave has one significantdrawback: it increases the attack surface; if the system library or library OS has vulnerabilities, the attacker could compromise the security ofthe application. Hence, the code in the enclave (as TCB) must be as small as possible.

3 Executing the enclave code causes a high performance-overhead due to encryption/decryption routines added to ensure confidentiality and integrity of the enclave. This overhead is amplified by eachsystem-call in which the enclave must be exited and reentered to ensure security. Furthermore, every memory or cash miss writes down the performance as it requires encryption and decryption.

Based on the aforementioned challenges, the question is how much functionality to put into the enclave?

2 Solution

One possible solution would be similar to Microsoft's Haven, in which the application and its library along with the most of application-supporting code of the OS(i.e., C library, library OS, and shielding layer) areplaced into the enclave. Putting the library OS in the enclave enables running unmodified applications.Although putting library OS in the enclave reduces the number of system-calls to the untrusted underlying OS which has a positive impact on the performance, it increases TCB size by 5x–Increasing TCB size results into expensive enclave swapping–, the service latency by 4x, and also raises the attack surface.

The other extreme is placing minimal install base in the enclave; i.e., just the application code, its library and, a thin shim C library to wrap C libraries. Though, this design minimalizes the TCB size, it maximizes the external interface/interaction of TCB with the underlying untrusted OS, which raises the likelihood of vulnerability. Moreover, data and operations that fall out of the scope of the enclave are not protected.
Based on the aforementioned extremes, the design of a secure container with SGX is a problem of balancing security against performance. SCONE proposes a suitable middle-ground solution which limits the external interface by putting the heavily used C library in the enclave, thus reducing the number of system-calls. It doesnot use the library OS at all.
SCONE also provides an extra shielding layer to protect security-sensitive data during system-calls (both file system and network system-calls) in a transparent manner. The keys and certificates utilized during encryption or authentication of files, communication channels, and consoles are part of the SCONE container and are determined during secure attestation startup. The shielding layer provides protection against Iago attacks, in which an unchecked return value of a system call compromises the application.
Moreover, SCONE supports a M:N multithreading model, in order to overcome performance reduction due to enclave exits in the case of system calls. This enables execution of another enclave when encountering a system-call in the current enclave. Therefore, it results in better performance compared to single threading.
Another improvement is introducing asynchronoussystem-call in order to reduce the number of enclave transitions during system call. This was performed by executing an OS thread in parallel with enclave thread to manage system call separately, by using two queues for request and response system calls in the SCONE kernel module.
SCONE's implementation is integrated with Docker as the most popular container platform.

3 Evaluation

Evaluations for the set of applications Apache, Redis, and NGINX- show that the throughput is decreased by 0.8, 0.6, and 0.8 respectively, despite an unclear exception for Memcache which runs 1.2 times faster than the native version.
L3 cache miss reduces the performance 12 times while EPC swapping results into the overhead of three orders of magnitude.

4 Future Work

The future work can be mentioned as per following:

- Unmodified applications could not be executed on SCONE framework, since adopting SGX restrictions involvessignificant code changes to the application. Hence, it is recommended to design a system that could run unmodified applications.
- SCONE has a limited threading model in which some system calls such as fork, exec, and clone are not supported. Hence, it is not operative for many applications. Several usability enhancements -including fork and dynamic loading- are required.
- Moreover, the shielding layers in the SCONE protect only the application data. It is necessary to design a system to protect and verify OS data such as file system metadata.
- In addition, it is recommended to integrate SCONE with the open container platform [28].
- As the evaluations confirm, the obtained performance is not very suitable and needs to be improved as future work.
- Since some attacks such as the denial of services, side channels, and controlled-channel attack are not considered in the thread model of SCONE, it is recommended to enhance the thread model.