SCONE
Secure CONtainer Environment
Secure Docker containers using Intel SGX
INTRODUCTION
CONTAINERS VS. VMS

• performance advantage
  ▫ IO
  ▫ Latency
  ▫ Startup

• weaker security
  ▫ Kernel must protect a larger interface
  ▫ Isolation is only software-based
SGX RECAP

- SGX Enclaves shield code and data from being accessed by other software, especially higher-privileged software
- Enclave memory resides in the Enclave Page Cache
IDEA

- Data should be protected not only from other containers, but also from the kernel and hypervisor
- We use Intel SGX to secure classic Docker containers from the OS by executing the process in the enclave
CHALLENGES

- Minimize TCB to keep the attack vector small
- keep performance overhead low
- Support existing applications
BASIC GOALS

1. Small TCB
   - We only keep a libc library inside the enclave
2. Low performance overhead
   - user-level threading implementation
   - asynchronous syscall queue
3. Transparency to Docker engine
CONTAINERS
MAIN GOAL

Create a secure container mechanism that protects confidentiality and integrity of:

1. Process memory
2. Code
3. External I/O

from attackers with sudo access
DETAILED THREAT MODEL

Attacker has access to:

- sudo
- hardware
- entire software stack, including OS

Outside of scope:

- DoS attacks
- Side-channel attacks
DESIGN
SCONE ARCHITECTURE
INTERFACE SHIELDING

Shields focus on:

- preventing low level attacks
- ensuring confidentiality and integrity of shared data

SCONE provides three kinds of shields:

- File system shield
- Network shield
- Console shield
THREADING MODEL

- M application threads get mapped to N OS threads
- OS threads that enter the enclave get handled by scheduler
- SCONE kernel module reserves hardware threads to queue syscalls
SYSTEM CALLS

• SGX can't execute syscalls directly, so we need helper functions
• Arguments must be copied to non-enclave memory and then be processed
• Transitions are expensive

Solution: asynchronous syscall interface

• Consists of request and response queue
SYSTEM CALL FLOW
WORKFLOW IN DOCKER

- One container consists of one protected process
- Otherwise indistinguishable from regular container
WORKFLOW
BENCHMARK SUMMARY
APP BENCHMARK

- SCONES-async containers achieve almost native performance
- Single-thread applications don't perform well
FILE SYSTEM SHIELD

- Small datasets perform well
- Larger datasets drop to 35% throughput, worst case
SYSCALLS

• SCONC-async achieves almost native syscall frequency, further improvements possible
CONCLUSION

• SCONE TCB Size is 60-200%
• Average throughput is at least 60%, sometimes even better than native
• All we need is static recompilation and the kernel module
QUESTIONS
SOURCES

Research paper here
Intel SGX Documentation here
Slides available soon