ShieldBox: Secure Middleboxes using Shielded Execution

Bohdan Trach et al.

Symposium on SDN Research (SOSR) 2018

Minkyung Park

mkpark@mmlab.snu.ac.kr

July 21, 2020

Contents

Middlebox in cloud environments

- Middleboxes maintain a wide range of workflows for improving the efficiency, performance, reliability, and security
  - WAN optimizers, caching, proxies, firewalls, IDS
- Many enterprises **outsource middleboxes to the cloud** to reduce deployment and management costs
- Can we trust the cloud environment?
  - If a cloud provider is malicious, it can leak secrets inside of the middleboxes, extract any encryption keys and learn private information of end-users

Secure outsourcing

- State-of-the-art systems advocate network processing **over encrypted traffic**
  - Restrictive type of functionalities
  - High performance overhead

- **ShieldBox** outsources middleboxes **without sacrificing performance** while supporting **a wide range of network functions**

Design overview

- Design principles
  - Security: provide strong confidentiality and integrity against a powerful adversary
  - Performance: achieve near-native throughput and latency
  - Generality: support a wide range of network functions

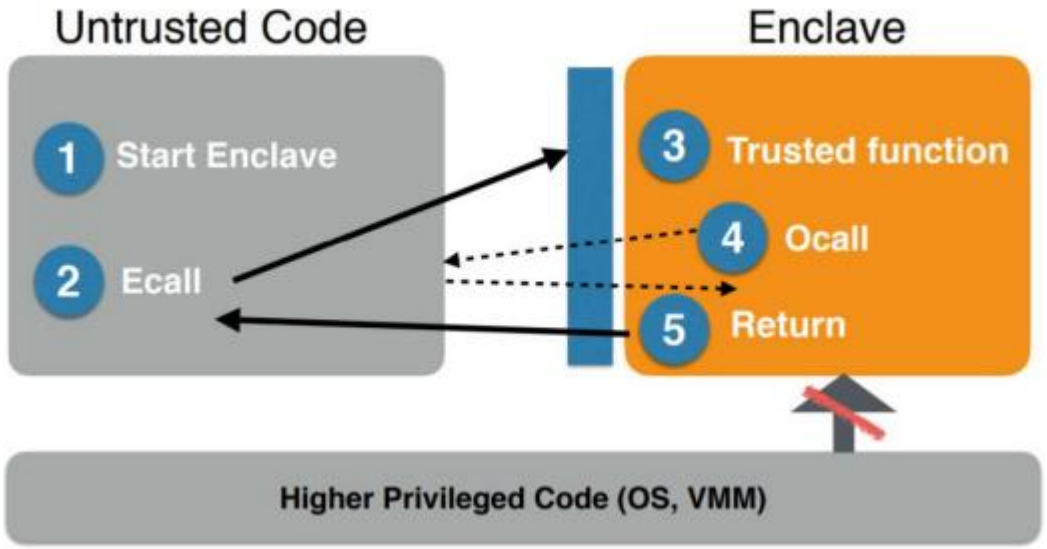
- ShieldBox leverages Intel SGX to shield middleboxes, DPDK to achieve high performance, and CLICK to provide a rich set of network functions

Intel SGX

- A set of instructions added to the Intel architecture to protect selected code and data from disclosure or modification

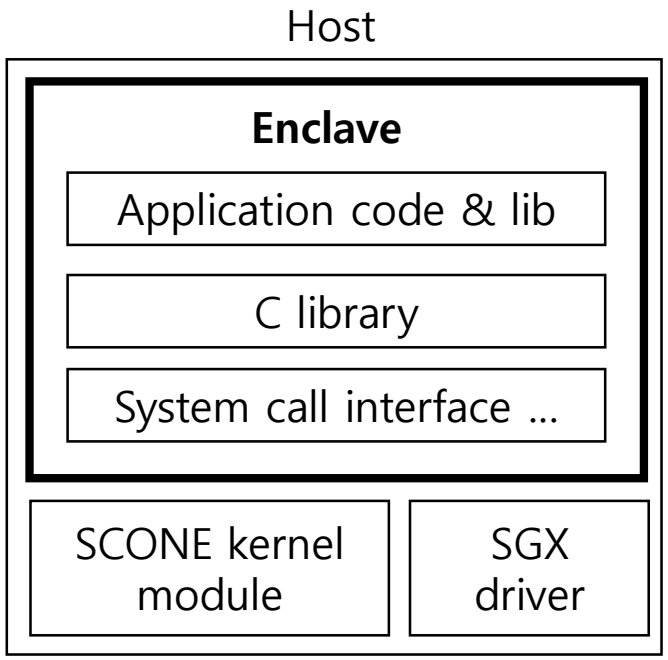
- An application can build a protected container called an enclave
  - Its memory is encrypted by a hardware unit
  - Any privileged software (e.g., OS and BIOS) cannot see its content

- Inside the enclave, untrusted host functions (e.g., syscall) are restricted
  - The app developer can declare ocall functions to run untrusted code
  - However, its return is not guaranteed as secure



SCONE

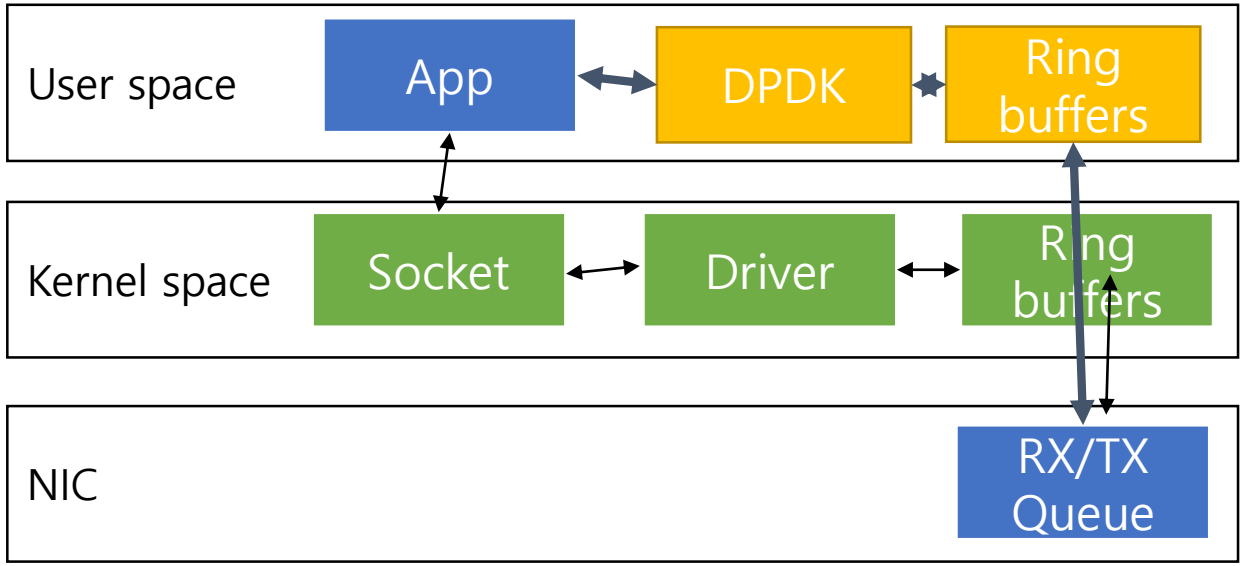
- SGX imposes many restrictions on enclave code
- Adopting SGX necessarily involves significant code changes to applications
- SCONE is a shielded execution framework that enables **unmodified POSIX applications** to be executed inside the enclave



DPDK (Data Plane Development Kit)

- A framework to provide fast packet processing in data plane applications

- Packets bypass kernel
  - User-level applications handle them directly

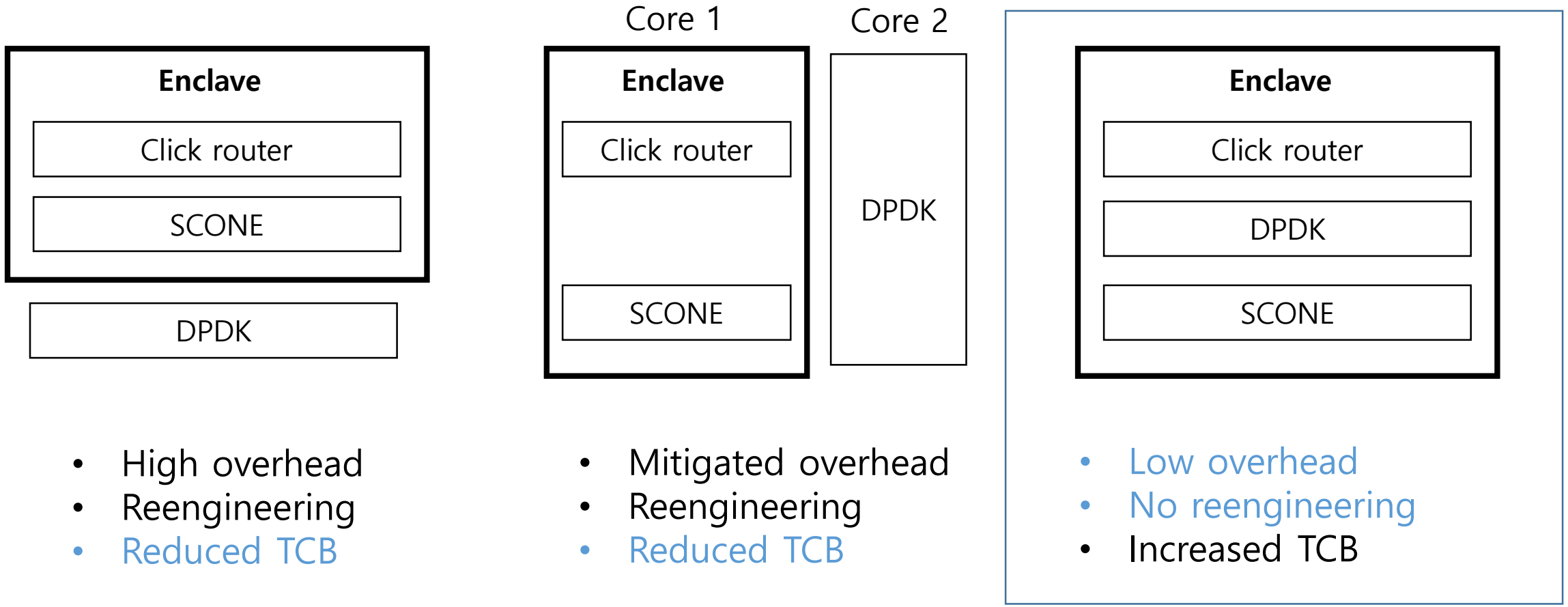


CLICK

- CLICK is a software router
- It provides a large number of *element*s
  - Each element performs a simple packet function
  - e.g., IP look up, TTL decrement, buffering
- A router configuration is a directed graph with elements at the vertices
- Packets flow along the edges of the graph

ShieldBox system overview

- ShieldBox consists of a simple integration of a DPDK-enabled Click that is running inside the SGX enclave using SCONE



- High overhead
- Reengineering
- Reduced TCB

- Mitigated overhead
- Reengineering
- Reduced TCB

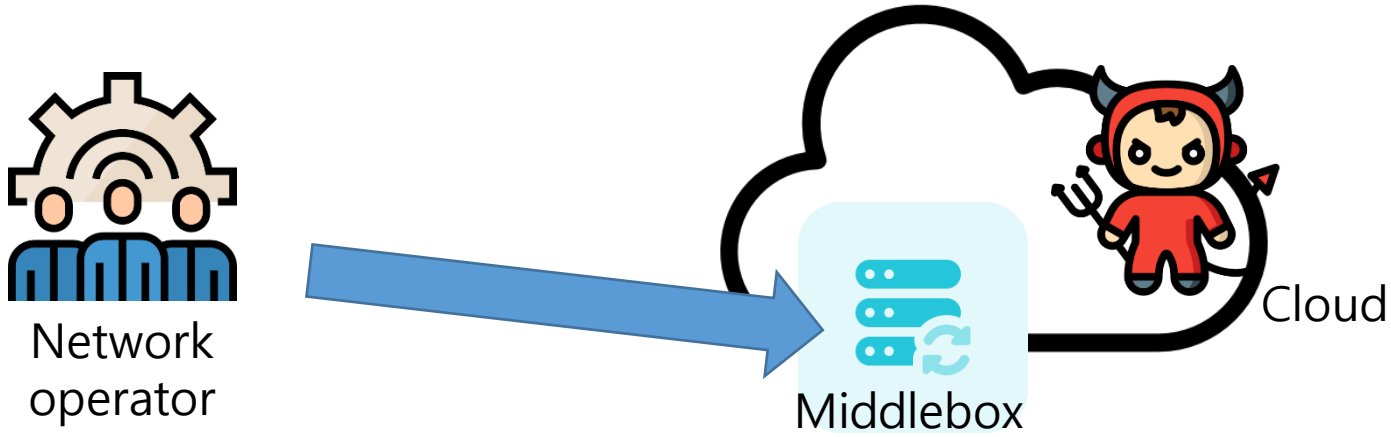
- Low overhead
- No reengineering
- Increased TCB

Threat model

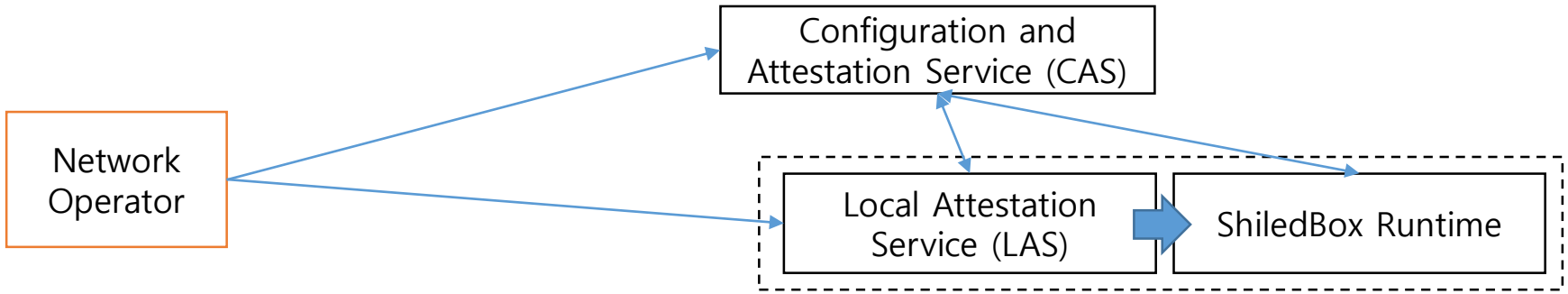
- Middleboxes process **confidential data** and they are deployed in the **untrusted cloud environment**

- Attackers want to learn the contents of encrypted packets and system configuration
  - System configuration: cryptographic keys, filtering and classification rules, etc.

- The adversary can control the entire system software stack including OS or hypervisor



Bootstrapping

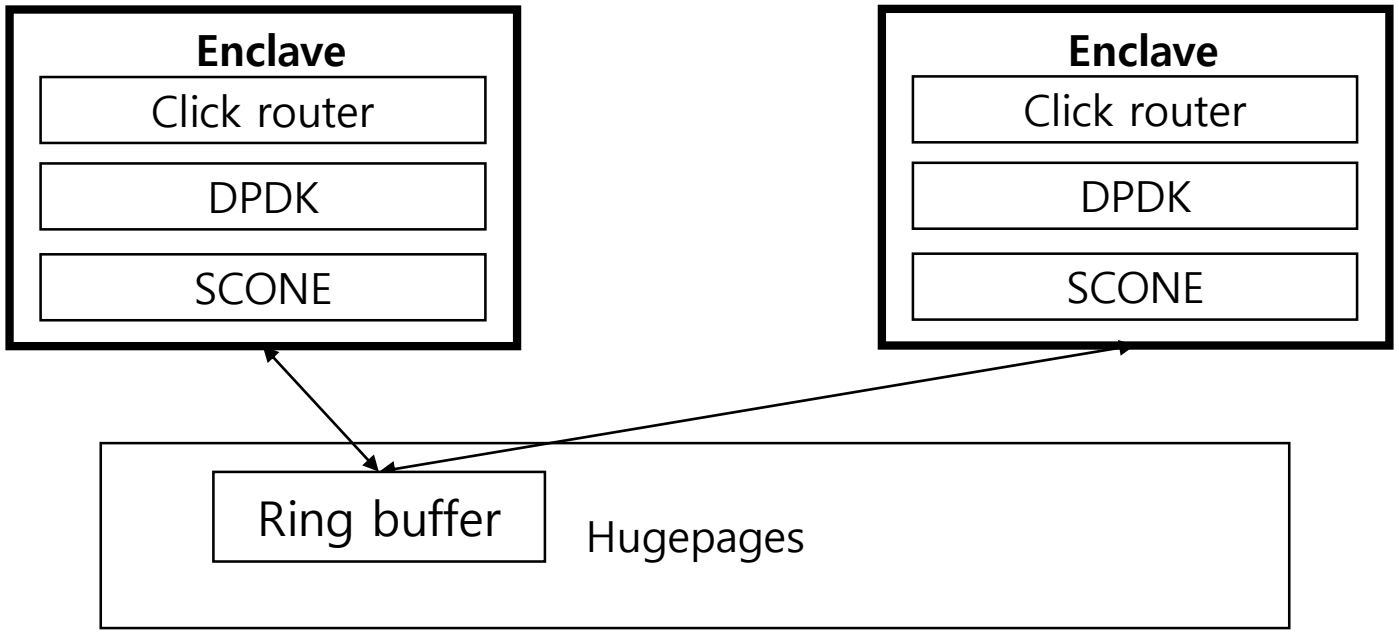


- Initially, ShieldBox instance and its operator has to establish trust
- **Configuration and Attestation Service (CAS)** verifies LAS instead of the operator
- **Local Attestation Service (LAS)** acts as a root of trust once CAS trusts LAS
  - It launches the ShieldBox instance

NFVs chaining

- Typically, NFVs are chained together to build a dataflow processing graph spanning across multiple cores, sockets and machines

- DPDK ring API in hugepage are used for communication between different ShieldBox instances



Secure element

- New Click elements for enclave are introduced
- ToEnclave
  - Copy packet data into enclave
- Seal/Unseal
  - Encrypt/decrypt packet
  - e.g., for VPN
- Persist/StateFile
  - Provide persistent memory outside the enclave and restore it back to the enclave
  - E.g., file

Other aspects and optimizations

- NIC timer source: On-NIC clock is used for the clock source
  - Invoking time-related syscalls (e.g., gettimeofday or clock_gettime) or rdtsc instruction is disallowed and slow
- Memory pre-allocation: run_task function allocates a packet descriptor storage on the stack → the memory is pre-allocated
- Reducing system calls
- Modified scheduler: it allows to remove a system call from scheduler

Evaluation

- Two machines connected using 40 GbE Intel XL-710 network card
  - load generator: Broadwell Intel Xeon D-1540 (2.00GHz, 8 cores) with 32GB RAM
  - SGX-enabled machine: Intel Xeon E3-1270 v5 (3.60GHz, 4 cores) with 32GBRAM
- Micro-benchmark applications: three CLICK elements
  - Wire: sends the packet immediately after receiving
  - **EtherMirror: sends the packet after swapping the source and destination addresses**
  - Firewall: does packet filtering based on PCAL-like rules
- Case study using two applications
  - **A multiport IPRouter**
  - An IDS
- Unless stated otherwise, ToEnclave element is not used in the benchmark

Configuration and attestation

Phase	Average Duration, µsec
Attestation	19467
CAS communication	19301
LAS communication	1474
Configuration	825.6
Total time	26368

Table 3: Overheads of configuration and attestation

- The results show that remote attestation has a negligible effect
  - It is performed once per instance start-up

Micro-benchmark measurement

- Throughput, latency, and ToEnclave throughput are measured

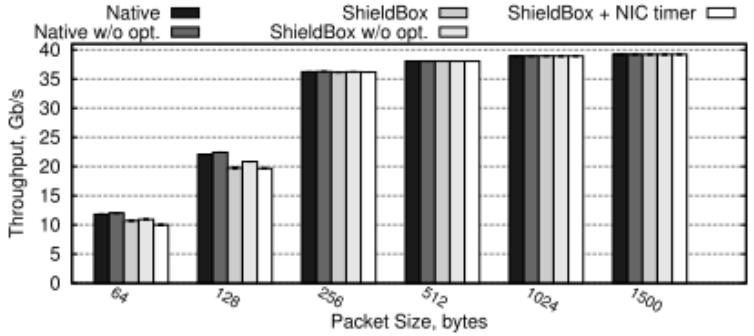


Figure 6: Throughput: EtherMirror w/ varying packet size

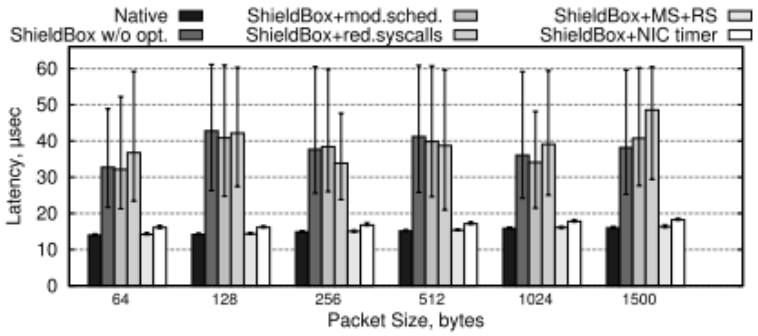


Figure 8: Latency: EtherMirror measurements

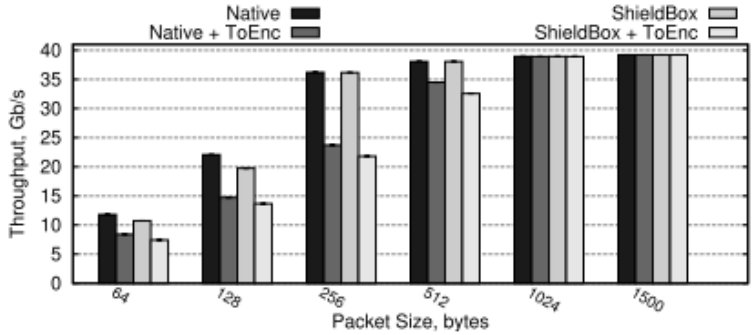


Figure 10: ToEnclave Throughput: EtherMirror measurements

Case study measurement

- IPRouter classifies all packets into three categories
  - ARP requests are answered
  - ARP replies are dropped
  - Other packets are passed to a routing table element

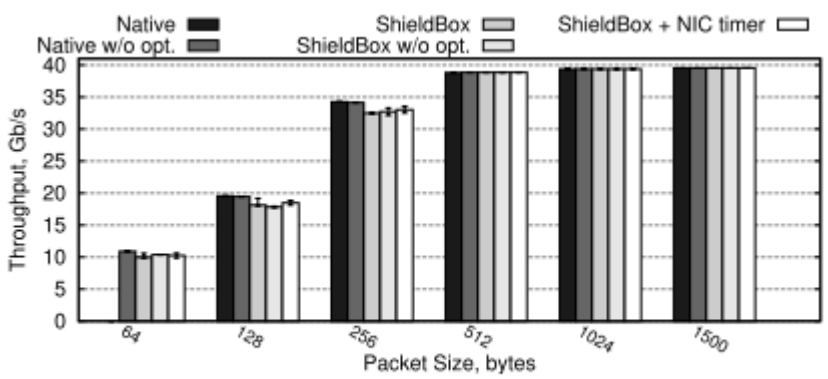


Figure 13: IPRouter: Throughput measurements

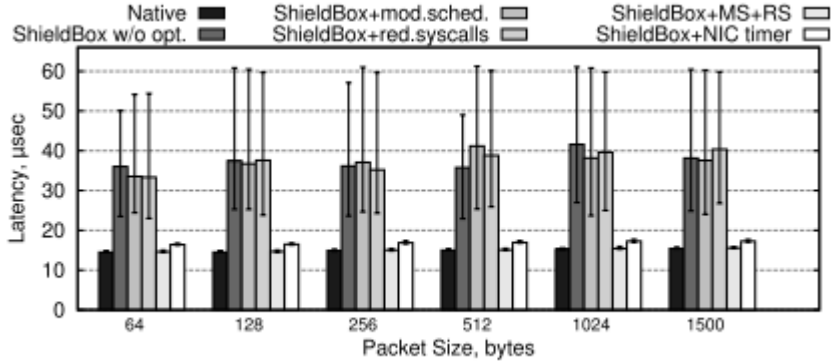


Figure 14: IPRouter: Latency measurements

Conclusion

- ShieldBox is a secure middlebox framework for high-performance network functions
- It protects secret information from untrusted commodity servers
- It integrates a high-performance I/O processing library (Intel DPDK) with a shielded execution (Scone) framework based on Intel SGX
- The evaluation show that ShieldBox achieves near-native throughput and latency