Civet: An Efficient Java Partitioning Framework for Hardware Enclaves

Chia-Che Tsai*, †, Jeongseok Son†, Bhushan Jain‡, John McAvey†, Raluca Ada Popa†, Donald E. Porter‡

*TAMU, †UC Berkeley, ‡UNC Chapel Hill, †Hendrix College
Hardware Enclaves as Root of Trust

An abstraction for bootstrapping users’ trust on untrusted platforms.

✓ Proof of a trustworthy CPU
✓ Isolated from untrusted OS & IO
✓ Memory encrypted in DRAM

Existing solutions:
• Intel SGX
• AMD SEV
• ARM TrustZone
• RISC-V Sanctum & Keystone
Cloud Platforms + Enclaves = Large Trusted Computing Base

Cloud applications are often
(1) complex
(2) multi-principal
(3) written in managed languages like Java.

Ex: Hadoop

\[ \lambda \]

mapper

\[ \lambda \]

reducer

6.3 MLoC
- Scheduler
- HDFS
- Workers
- Other mappers/reducers

+ 2.3 MLoC (JARs)
+ 0.9 MLoC (JVM)
Existing Approaches for Enclave Development

- **Entire Application**
  (Haven’14, SCONE’16, Graphene-SGX’17, SGK-LKL’20)
  - Enclave System API
  - Untrusted OS
  - No code modification but large TCB

- **API Engine Only**
  (VC3’15, SecureKeeper’16)
  - Enclave KVStore
  - Untrusted App
  - Small TCB but little flexibility

- **Partitioned / Partial Re-development**
  (Glamdring’17, GOTEE’19)
  - Partition
  - Enclave func()
  - Untrusted App
  - Small TCB but lack support for partition with object-oriented interfaces
Civet: Partitioning Java-based Applications for Enclaves

- Guided partitioning for experimentation of partition boundary
- White-listing class loading & polymorphism
- Tailored Java runtime for enclave performance patterns (e.g., GC)
Threat Model

Threat 1: Enclave loading unexpected classes/methods

Threat 2: Attacker abusing exposed enclave interfaces (i.e., Iago Attack [ASPLOS’13])

- Encrypted Data
- Enclave
- mapper X
- X.map()
- System API
- Operating System
- IO Devices, DRAM
- Untrusted Components

Civet: An Efficient Java Partitioning Framework for Hardware Enclaves
Partitioning Tool + Java Runtime for Enclaves

mapper X

User Input

Civet Partitioning Tool

Entry: X.map()

Configuration

(Design phase)

Trusted JAR

Untrusted JAR
Partitioning Tool + Java Runtime for Enclaves

Mapper X

User Input

Civet
Partitioning Tool

Entry:
X.map()

Configuration

Trusted JAR

Signature
Verified by
CPU

Untrusted JAR

Proxy for X

Enclave

Trimmed,
shielded
OpenJDK JVM

(Execution phase)
Determining Boundary for TCB

- Determine effective TCB based on partitioning decisions
- Prevent loading unnecessary code

Code Reachability Analysis
(Based on bytecode-level, call graph + points-to analysis)

Class & method shredding
See paper!
Polymorphic Attacks on Enclave Entries

Deep Input Type Checks

See paper!

Behaviors controlled by attackers

Polymorphic objects

Entry: X.map()

Benign

malicious

Trusted World

Untrusted World

Civet: An Efficient Java Partitioning Framework for Hardware Enclaves
Garbage Collection in Enclaves (1/2)

Insight: Memory overhead in enclaves heavily impacted by cache misses & page faults.

“Mark-and-Sweep”
Garbage Collection in Enclaves (2/2)

Insight: Memory overhead in enclaves heavily impacted by cache misses & page faults.

- Young Gen.
  - Fit into cache – 8MB (minimize misses)

- Mid Gen.

- Dead obj

- Fit into encrypted DRAM – 92MB (minimize page faults)

- Avg GC Time
  - 2-gen GC
  - 3-gen GC

Live object ratio
Civet: An Efficient Java Partitioning Framework for Hardware Enclaves

Partitioning Effectiveness + Performance

Hadoop Regular Expression Matching:

Before partitioning:
589K methods, 7.2MLoC

After partitioning:
12K methods, 248KLoC (-96%)

Hadoop latency for regular expression matching in 1GB encrypted data

Civet’s enclave protection + type checking adds 16-22% overhead

Civet’s enclave protection + type checking adds 16-22% overhead

Civet’s enclave protection + type checking adds 16-22% overhead

Before partitioning:
589K methods, 7.2MLoC

After partitioning:
12K methods, 248KLoC (-96%)
Conclusion

- Java workloads don’t fit into enclave programming paradigms
  - Dynamic and polymorphic behaviors
  - Monolithic runtimes and expensive resource management

- **Civet**: partitioning, refining and hardening with reachability analysis, deep type checking, and enclave-specific runtime design.

Questions or feedback: Chia-Che Tsai <chiache@tamu.edu>