Towards Memory Safe Python Enclave for Security Sensitive Computation

Huibo Wang, Mingshen Sun, Qian Feng, Pei Wang, Tongxin Li, Yu Ding

Abstract

Intel SGX Guard eXtensions (SGX), a hardware supported trusted execution environment (TEE) is designed to protect security sensitive applications. However, since enclave applications are developed with memory unsafe languages such as C/C++, the traditional memory corruption is not eliminated in SGX. Rust-SGX is the first toolkit providing enclave developers with a memory safe language. However, Rust is considered a Systems language and has become a good choice for concurrent applications and web browsers. Many application domains such as Big Data, Machine Learning, Robotics, Computer Vision are more commonly developed in Python programming language. Therefore, Python application developers cannot benefit from secure enclaves like Intel SGX and Rust-SGX.

To fill this gap, we propose Python-SGX which is a memory safe SGX SDK providing enclave developers a memory safe Python development environment. The key idea is to enable a memory safe Python language in SGX by solving the following key challenges: (1) defining a memory safe Python interpreter (2) replacing unsafe elements of Python interpreter with safe ones, (3) achieving comparable performance to non-enclave Python applications, and (4) not introducing any unsafe new code or libraries into SGX. We propose to build Python-SGX with PyPy, a Python interpreter written by RPython which is a subset of Python, has three unsafe parts related to RPython

Challenges

- Memory safety of Python interpreter
  - CPython has 300k lines of C code
  - PyPy, a Python interpreter is written in RPython which is a subset of Python, has three unsafe parts related to RPython
- Build from scratch or on top of Intel SGX SDK
- Porting PyPy interpreter in Intel SGX without introducing extra memory unsafe parts

Overview of PyPy

PyPy

Python Interpreter/JIT

Python Modules

Python Libraries

RPython Type/Translator/JIT

RPython Libraries

Contribution:

- A secure SGX enclave environment for Python software development free of memory corruptions
- Capabilities and rich library support as in native Linux environment
- No significant performance overhead

Objectives

- Memory safety of Python interpreter
- Build from scratch or on top of Intel SGX SDK
- Porting PyPy interpreter in Intel SGX without introducing extra memory unsafe parts

Key Insight

Unsafe PyPy

- Spatial safety risk in RPython type system
- C code in RPython's Translator/JIT backend
- RPython external C libraries

Memory Safe PyPy (MesaPy)

- Security Hardening (list and array index)
- Formal Verification tools (buffer overflow, buffer over-read, null pointer dereference, memory leak)
- Memory Safe Language

Examples of memory safe Python runtime libraries:

- MesaPy
  - PyPy Type/Translator/JIT
  - RPython Libraries

Evaluation

MesaPy normalized against PyPy native execution

Python-SGX normalized against MesaPy native execution

References