# Java Secure Native Interface

DSP-LAB 2016-17

By: Clindo Devassy and Subhadeep Manna

Guidance By: Marcel Blöcher and Malte Viering

# Security and Computers


**"I'm applying for the Information Security position.
Here is a copy of my resumé, encoded, encrypted and shredded."**

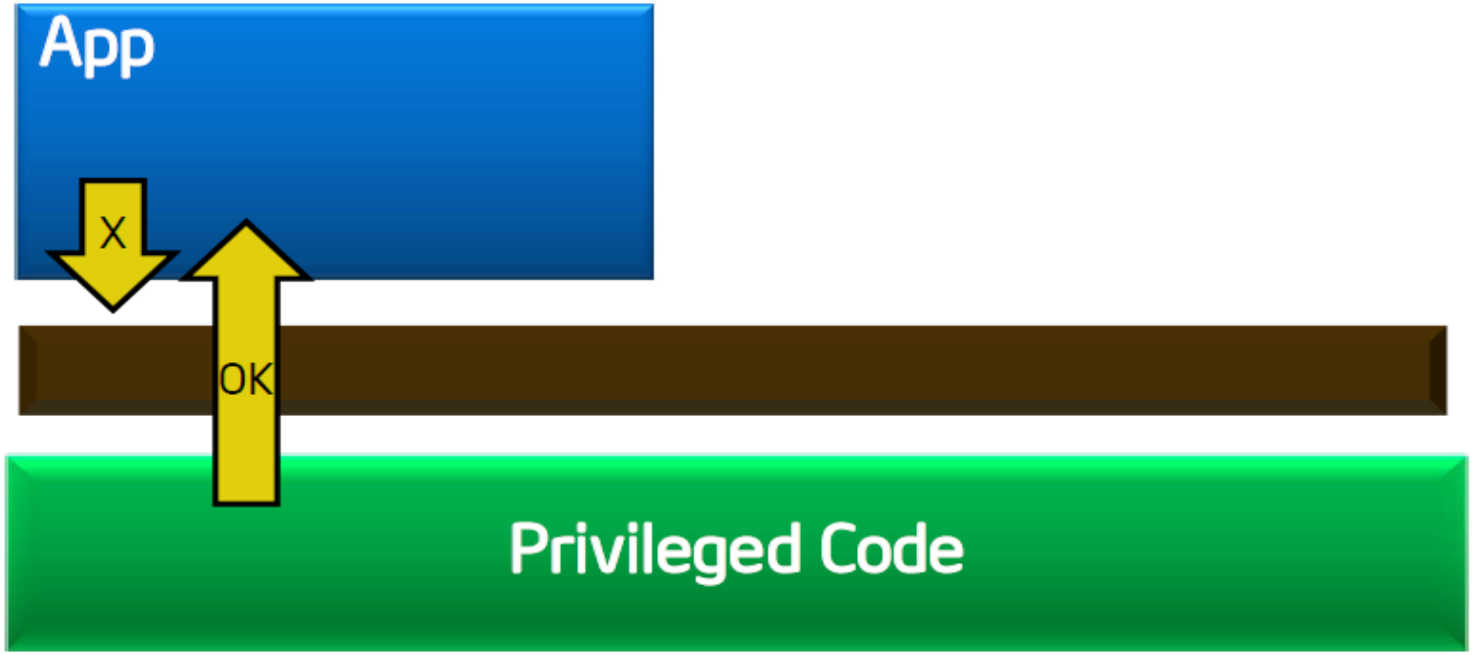
# Security: Primary Concern



## Memory and OS security

Buffer overflow project
Vulnerabilities: control hijacking attacks, fuzzing
Prevention: System design, robust coding, isolation

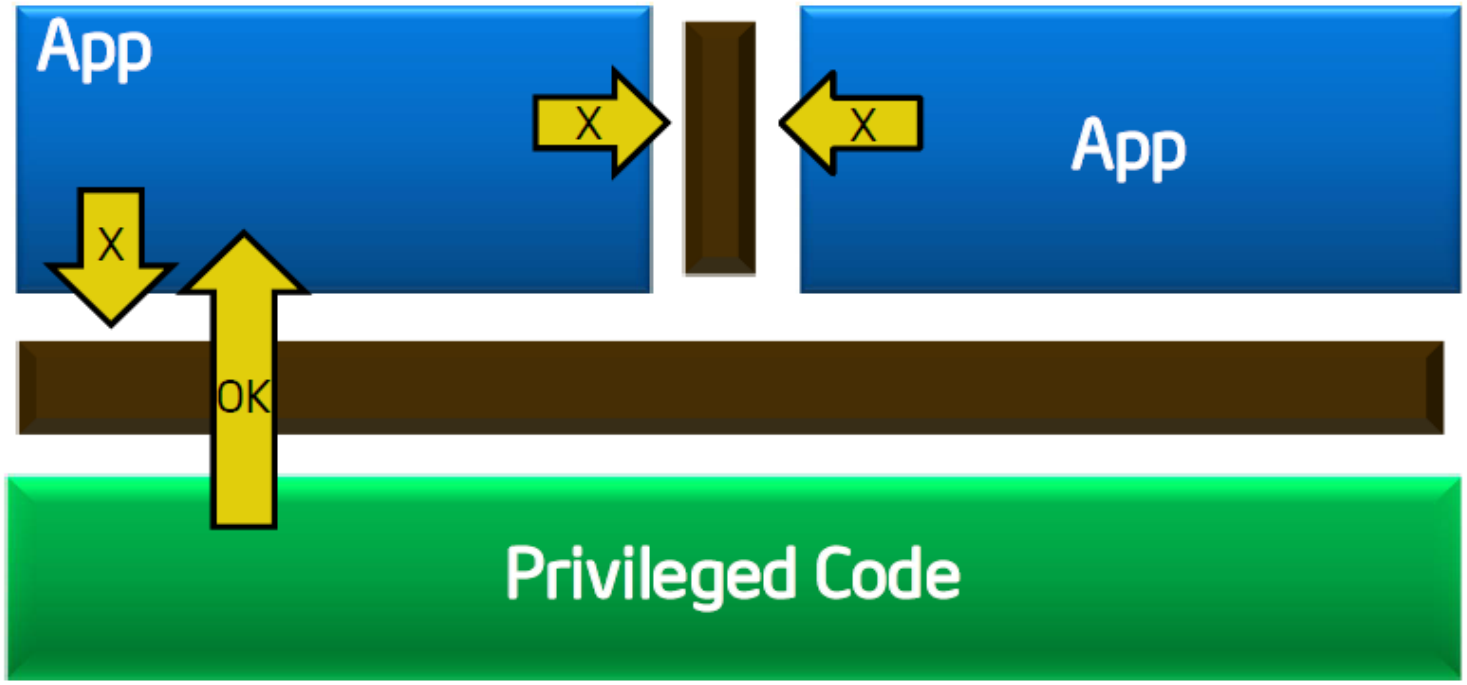
# OS, Application Security and Memory and Permissions




- Protected Mode protects OS from apps

Ref: https://software.intel.com/en-us/documentation/sgx-sdk-developer-reference

# OS, Application Security and Memory and Permissions




- Protected Mode protects apps from other apps

Ref: https://software.intel.com/en-us/documentation/sgx-sdk-developer-reference

# Example Privileged Code

**Main.java:**

```
public class Main
{
  public static void main(String []args) {
    LowLevel.executeLowLevelAction();
  }
}
```

**LowLevel.java:**

```
public class LowLevel
{
  public static void executeLowLevelAction() {
    System.out.println(System.getProperty("test"));
  }
}
```

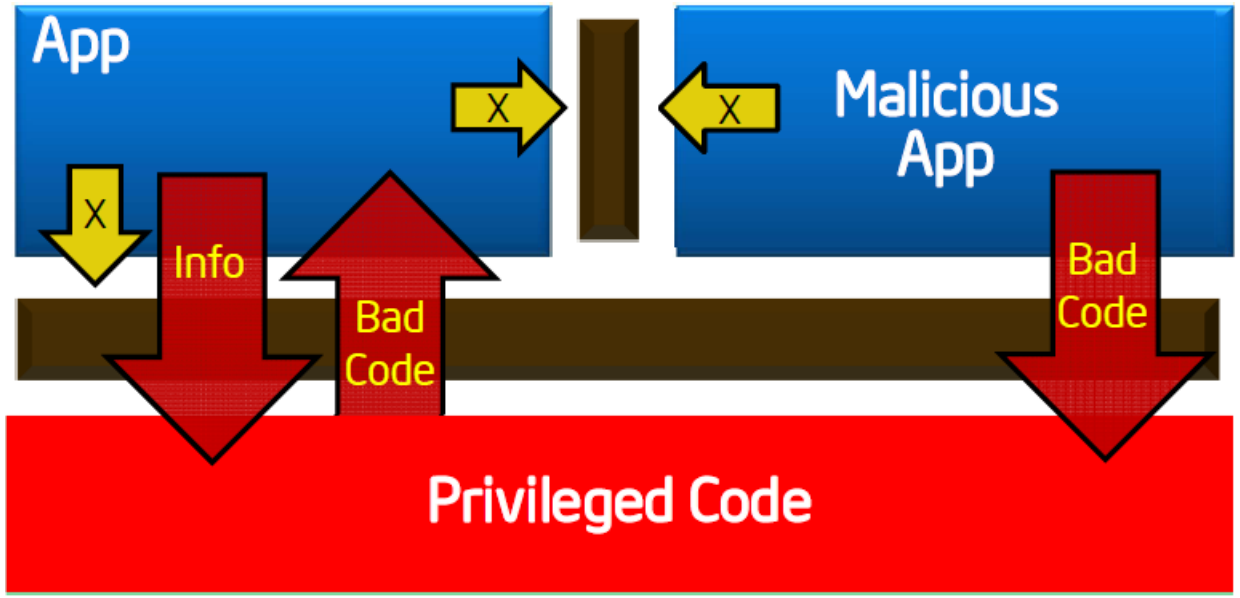
# Example Privileged Code(Software Based Measures)

C:> java -Dtest="Hello, World!" Main
Hello, World!

C:> java -Dtest="Hello, World!" -**Djava.security.manager Main**
Exception in thread "main" **java.security.AccessControlException**: access denied java.util.PropertyPermission test read)

at java.security.AccessControlContext.checkPermission(AccessControlContext.java:195)
at java.security.AccessController.checkPermission(AccessController.java403)
at java.lang.SecurityManager.checkPermission(SecurityManager.java:549)
at java.lang.SecurityManager.checkPropertyAccess(SecurityManager.java:143)
at java.lang.System.getProperty(System.java:539)
at LowLevel.executeLowLevelAction(LowLevel.java:6)
at Main.main(Main.java:4)

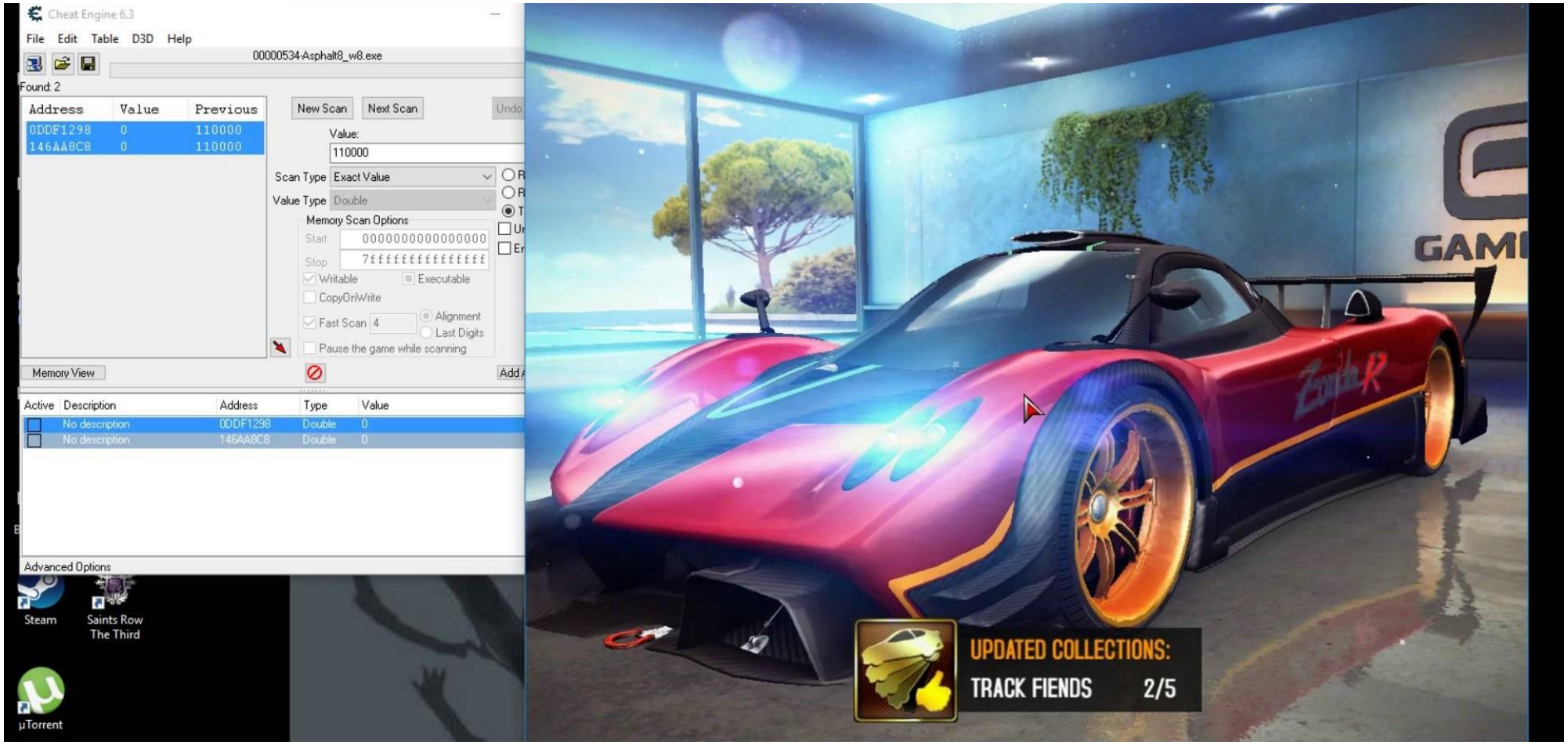
# Well, we still cannot trust the Security




- A malicious app exploits a flaw to gain full privileges and then tampers with the OS or other apps and memory.

Ref: https://software.intel.com/en-us/documentation/sgx-sdk-developer-reference

# Example: Hacking Asphalt 8 using CheatEngine v6.3



# So, how to **Stop** the Attacks on memory, OS and Apps ?

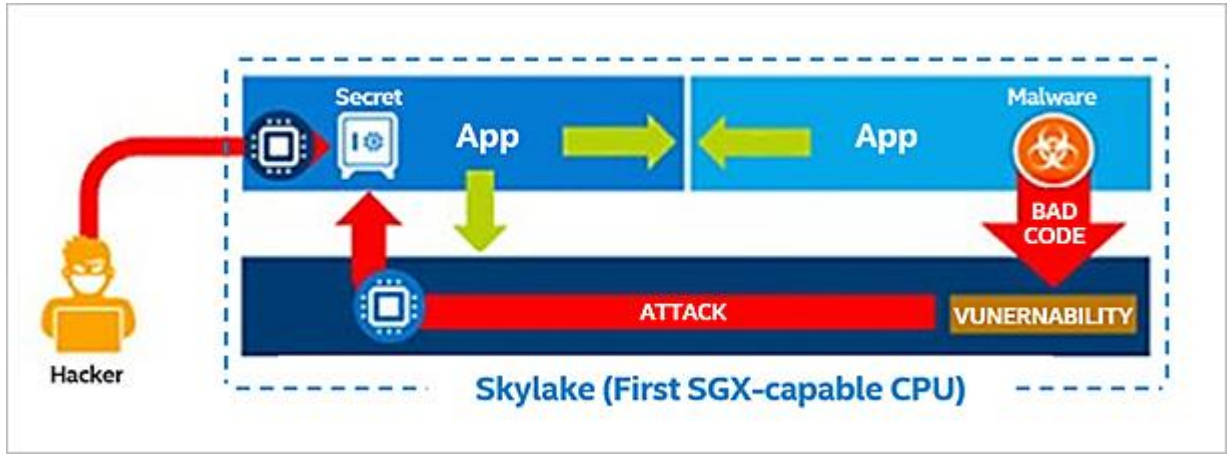
# Intel SGX-

# Software Guard Extensions

# A brief Introduction to Intel- SGX

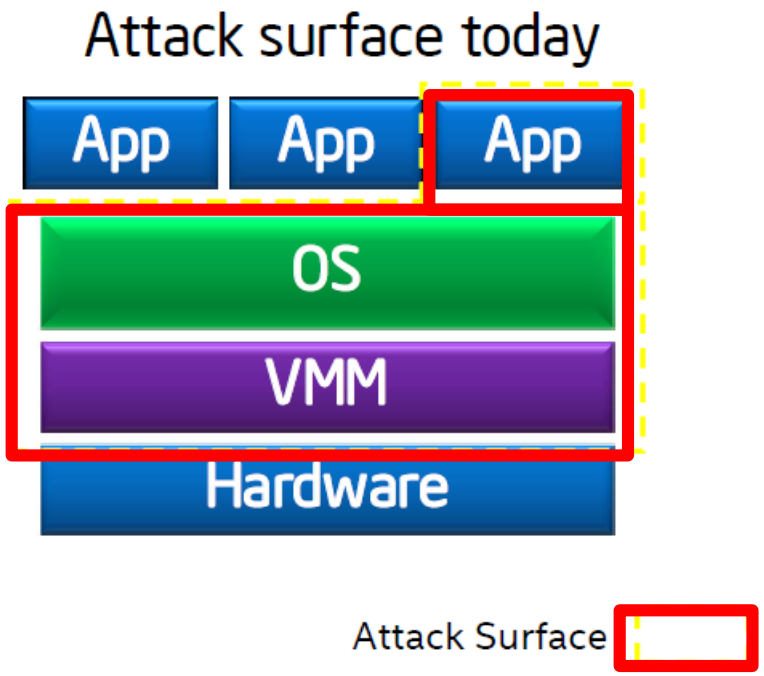
- Intel SGX is a ***set of new instructions*** from Intel.
- It was introduced in 2015 with the **sixth generation** Intel Processors.
- The introduction of SGX has a large impact on the security industry.
- It shifts how security is being achieved and lowers the attack surface area of projects

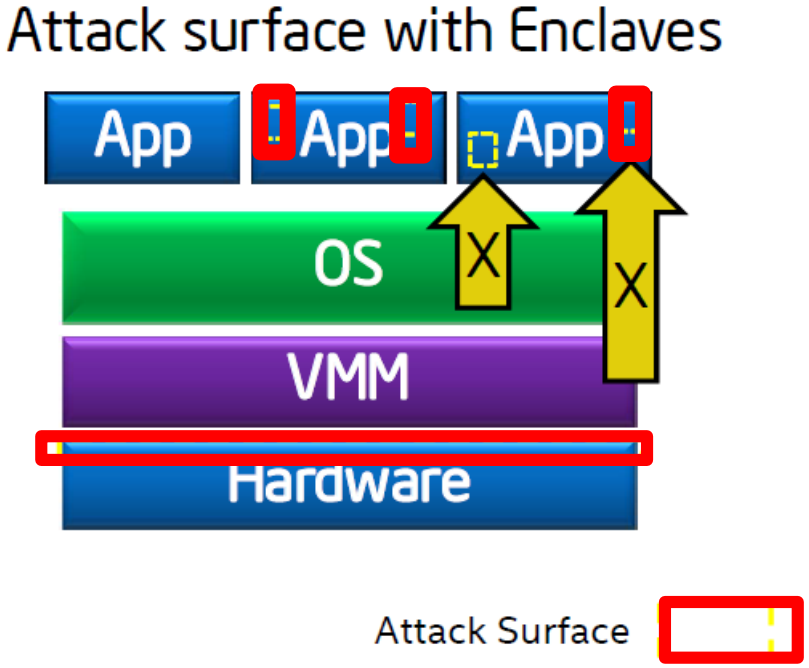
# An SGX Capable CPU




Ref: https://software.intel.com/en-us/sgx/details

# Developing a more secure App – using Intel SGX

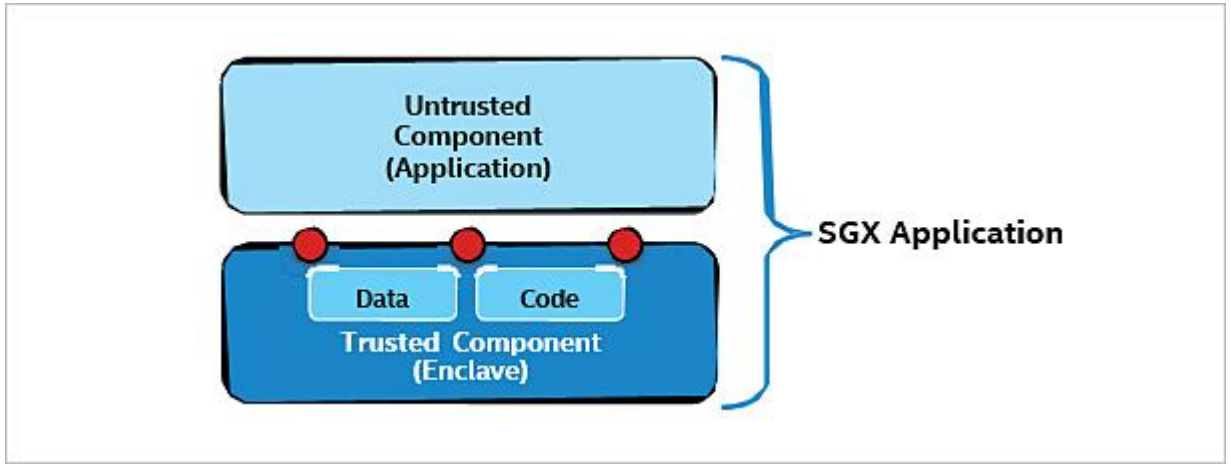


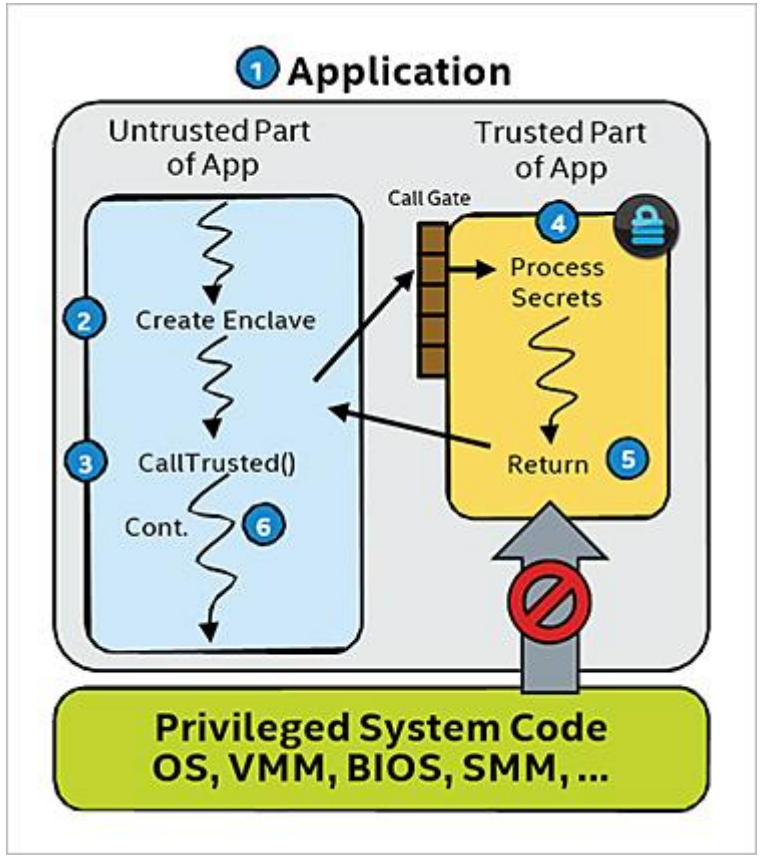




- Application gains ability to defend its own secrets.
- Reduced Attack Surface.

Ref: https://software.intel.com/en-us/documentation/sgx-sdk-developer-reference

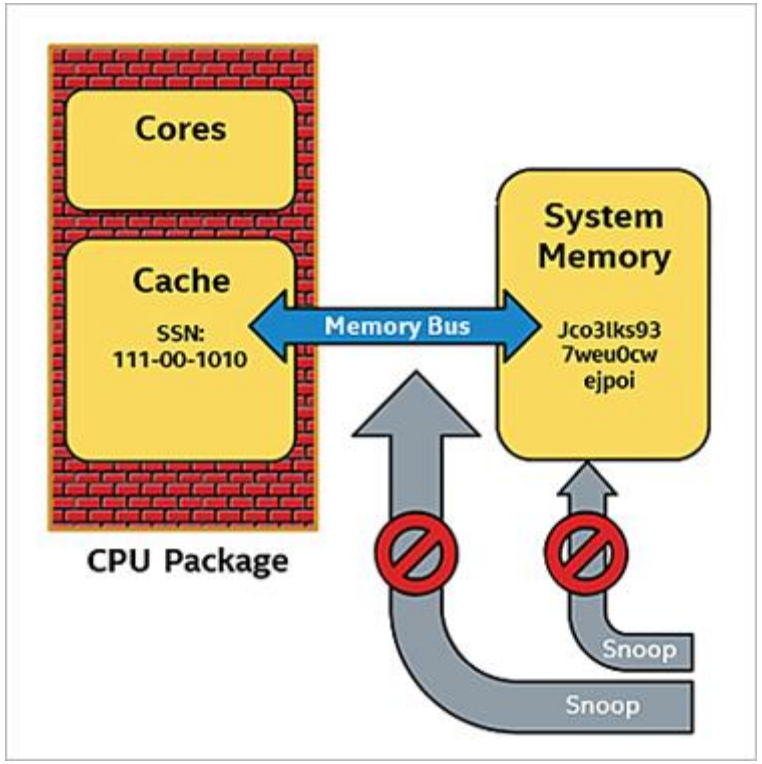
# An SGX Enabled Application




Ref: https://software.intel.com/en-us/sgx/details

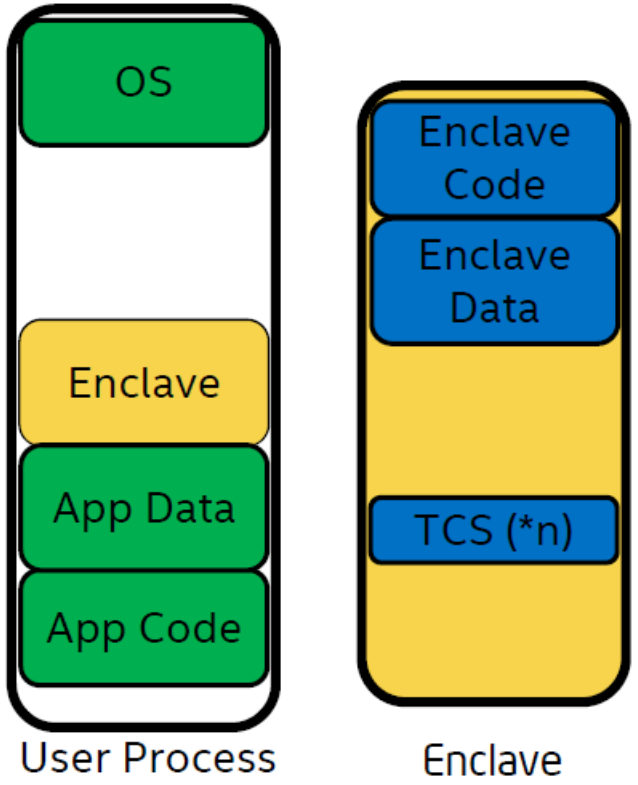
# Snooping Attacks Disabled




Ref: https://software.intel.com/en-us/sgx/details

# Intel - SGX : Feature of Enclaves

- With its own code and data
- Provide Confidentiality
- Provide integrity
- With controlled entry points
- With full access to app memory




Ref: https://software.intel.com/en-us/documentation/sgx-sdk-developer-reference

Currently libraries available for implementation in C/C++.

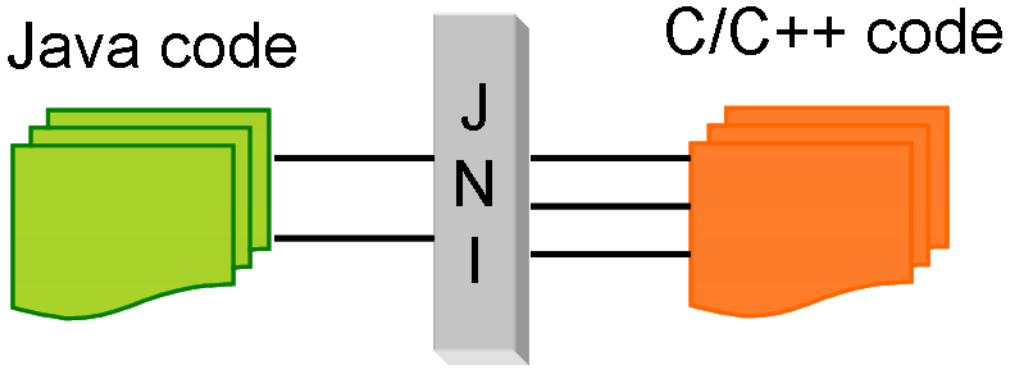
How do we implement it into our old friend Java ?

# JNI in short

- At times, it is necessary to use native codes (C/C++) to overcome the memory management and performance constraints in Java.

- Java Native Interface (JNI) is a programming framework.

# Our Current Objective: Using JNI



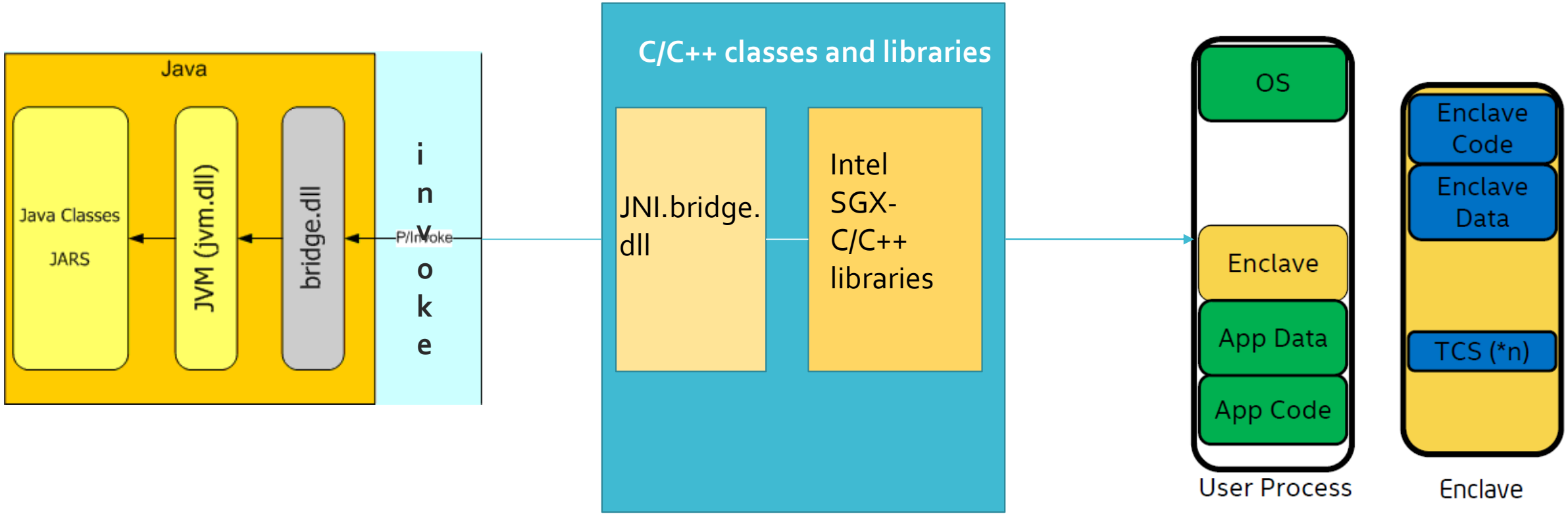
# Problem Statement: How to make Java App more secure ?

- Java code running in a Java Virtual Machine (JVM) to call and be called by native libs.
- In the JNI framework, native functions are implemented in separate .c or .cpp files.

# Our Objective : Security Approach

- Interpreter code written in Java must not be accessed by other applications.
- Should not be prone to attack by malicious apps and users.(Using SGX native C/C++ libraries).
- So we make the trusted code run in Enclave and not trusted code outside the enclave of the application.

# Our Objective : Secure App using JNI



# Thank You!