Java Secure Native Interface

DSP-LAB 2016-17

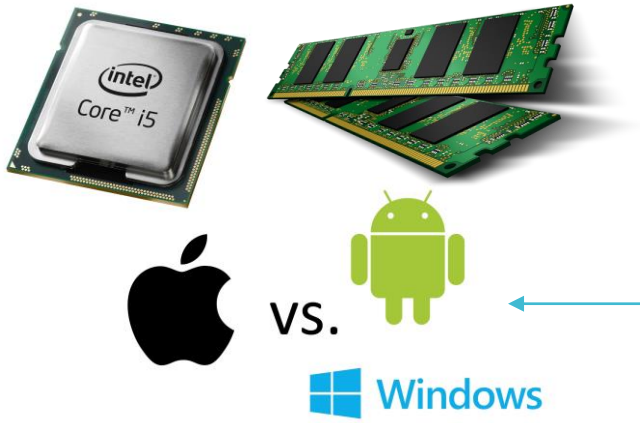
By: Clindo Devassy and Subhadeep Manna

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: Broad Classification



Web security

Web site attack and defences project
Browser policies, session mgmt, user authentication
HTTPS and web application security

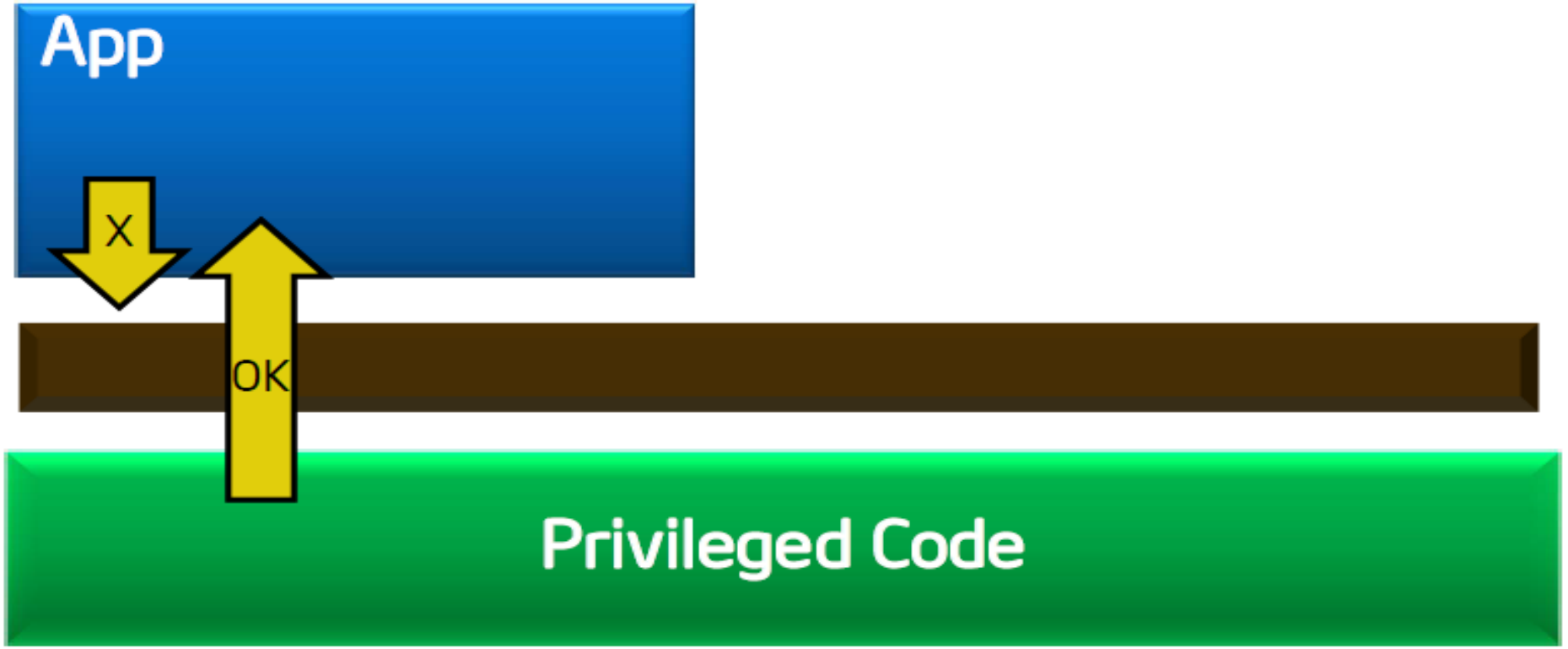
Memory and OS security

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

Network security

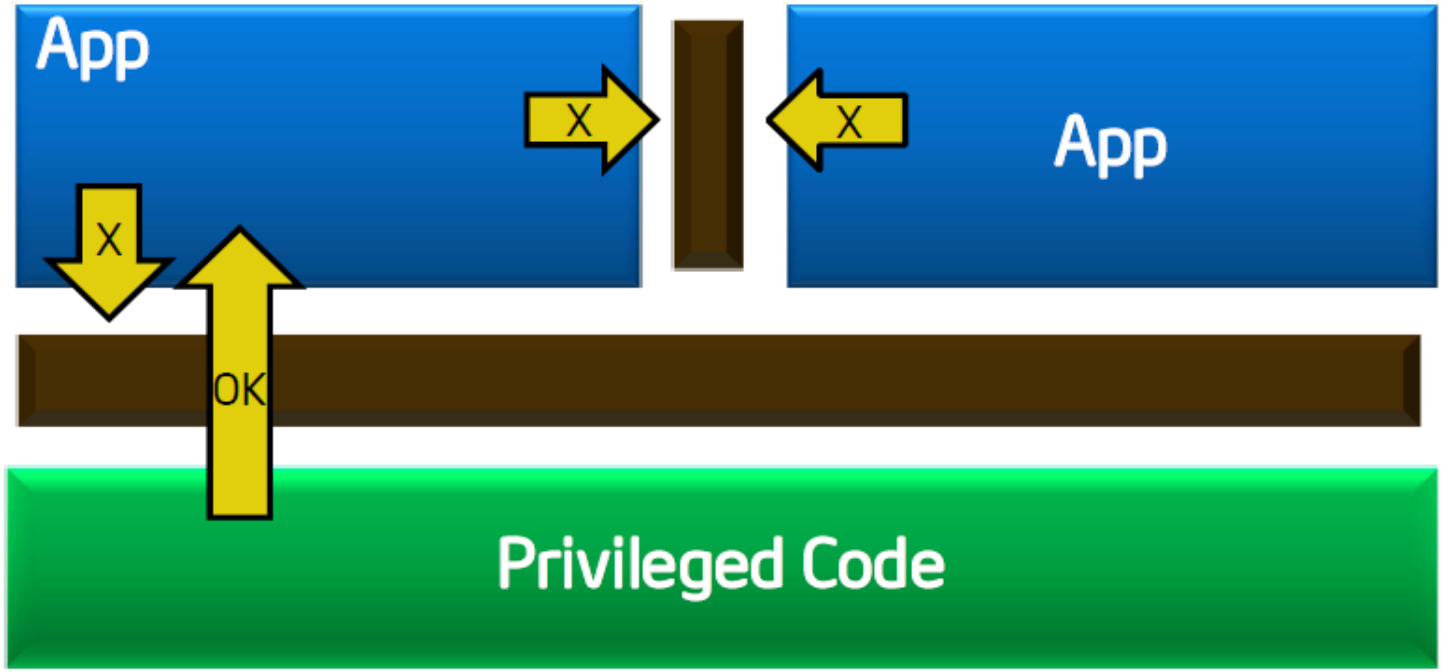
Network traceroute and packet filtering project
Protocol designs, vulnerabilities, prevention
Malware, botnets, DDoS, network security testing

OS, Application Security and Memory and Permissions



- Protected Mode protects OS from apps

OS, Application Security and Memory and Permissions



- Protected Mode protects apps from other apps

Example Privileged Code

Main.java:

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

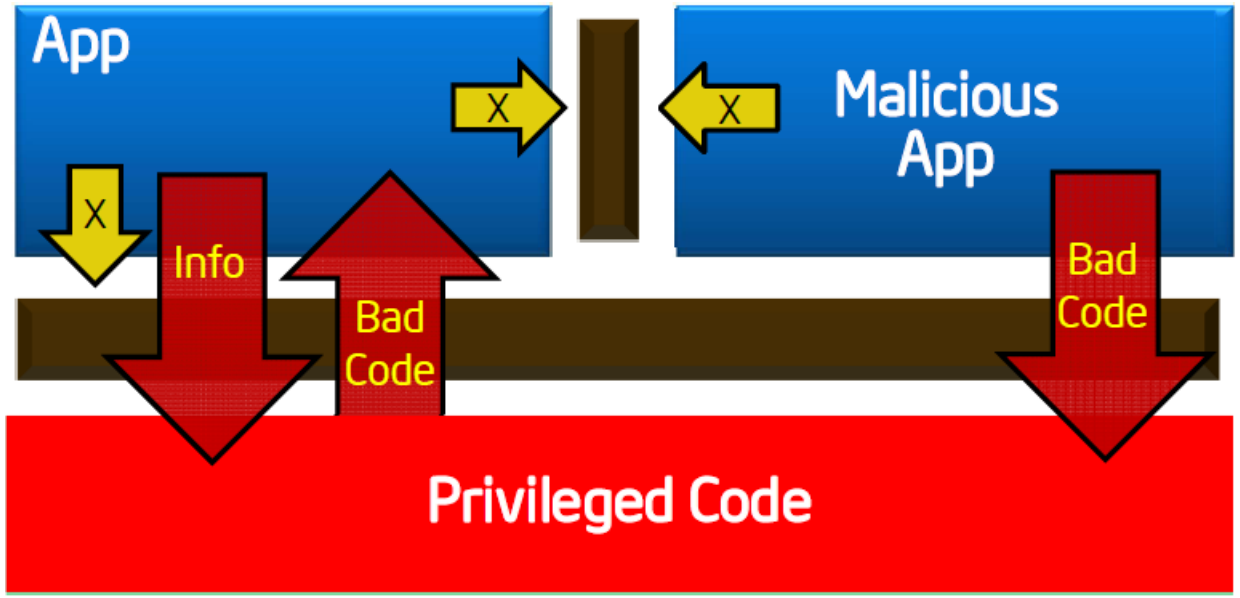
LowLevel.java:

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

Example Privileged Code

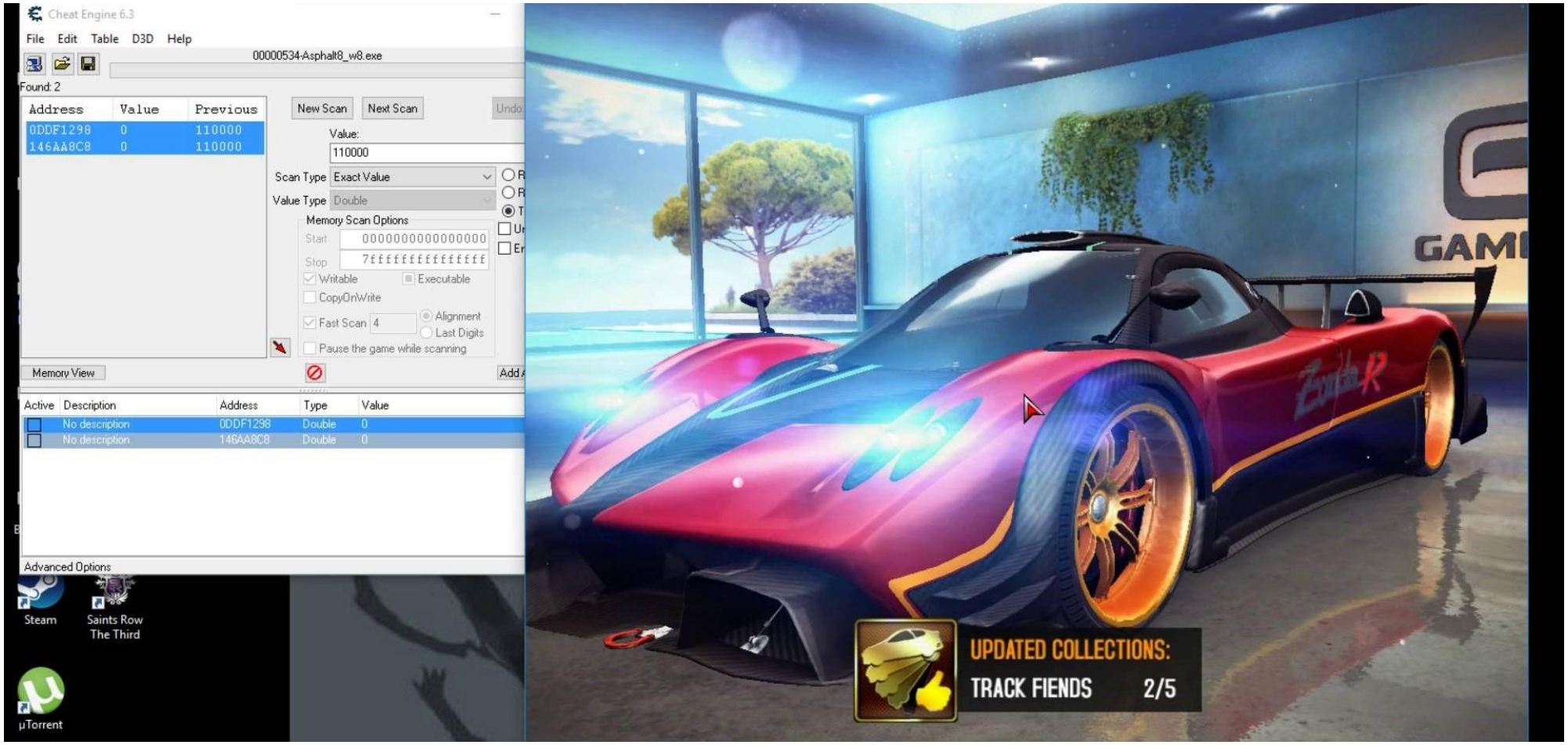
```
C:> java -Dtest="Hello, World!" Main
Hello, World!

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(AccessControlCont
xt.java:195)
        at java.security.AccessController.checkPermission(AccessController.java
403)
        at java.lang.SecurityManager.checkPermission(SecurityManager.java:549)
        at java.lang.SecurityManager.checkPropertyAccess(SecurityManager.java:1
43)
        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.

Example: Hacking Asphalt 8 using CheatEngine v6.3



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

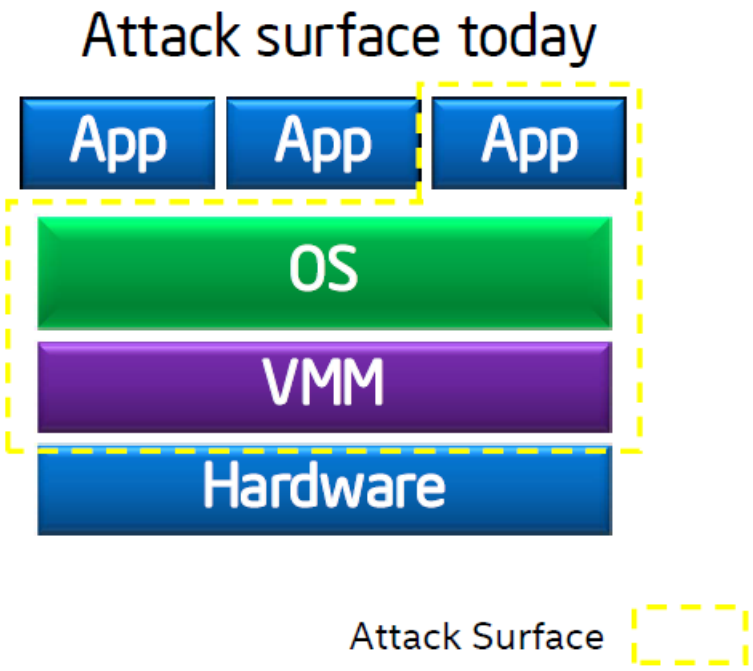
Intel SGX-

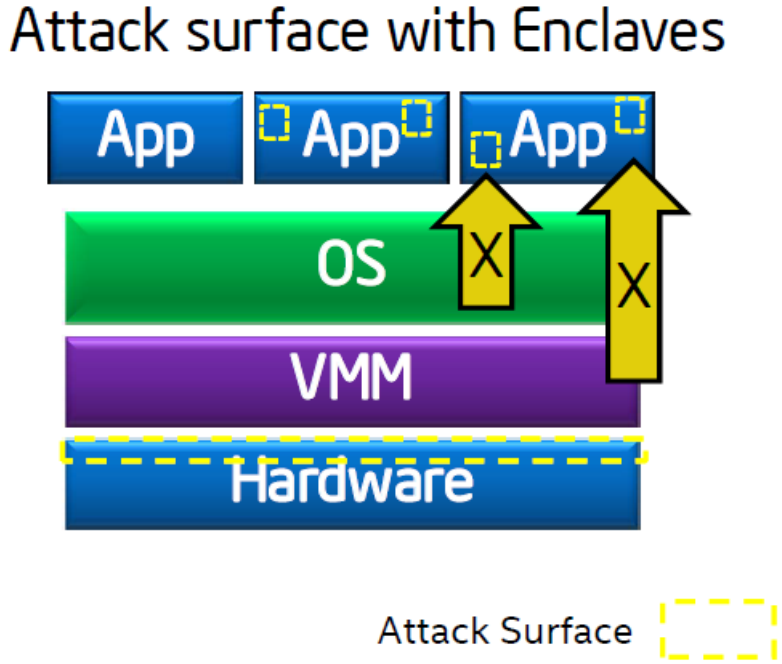
Software Guard Extensions

A brief Introduction to Intel- SGX

- Intel SGX is a ***set of new instructions*** from Intel that allows user-level code to allocate private regions of memory, called enclaves.
- Emulation of SGX was added to experimental version of QEMU system emulator in 2014.
- It was introduced in 2015 with the **sixth generation** Intel Core microprocessors based on the Skylake microarchitecture.
- 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

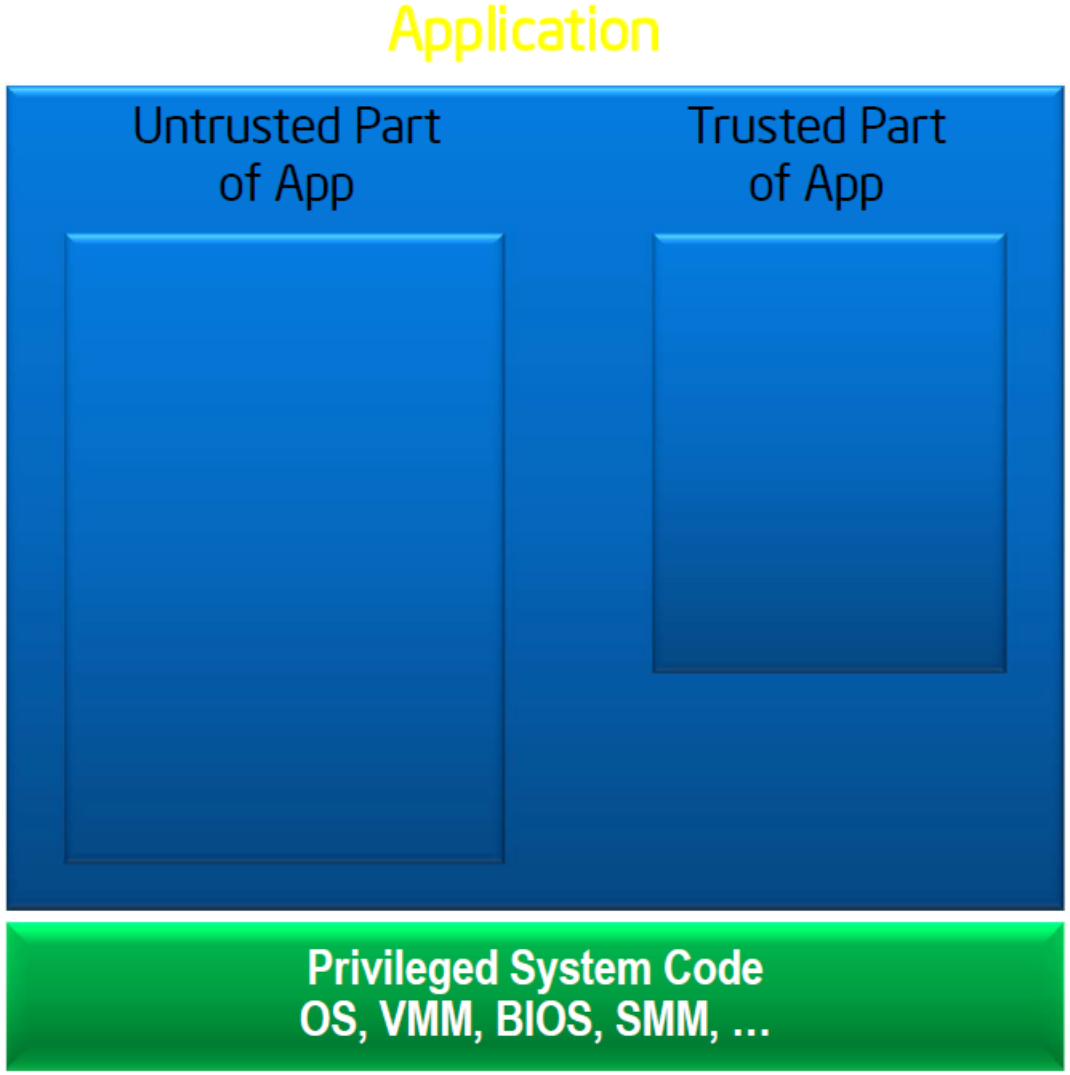
Developing a more secure App – using Intel SGX



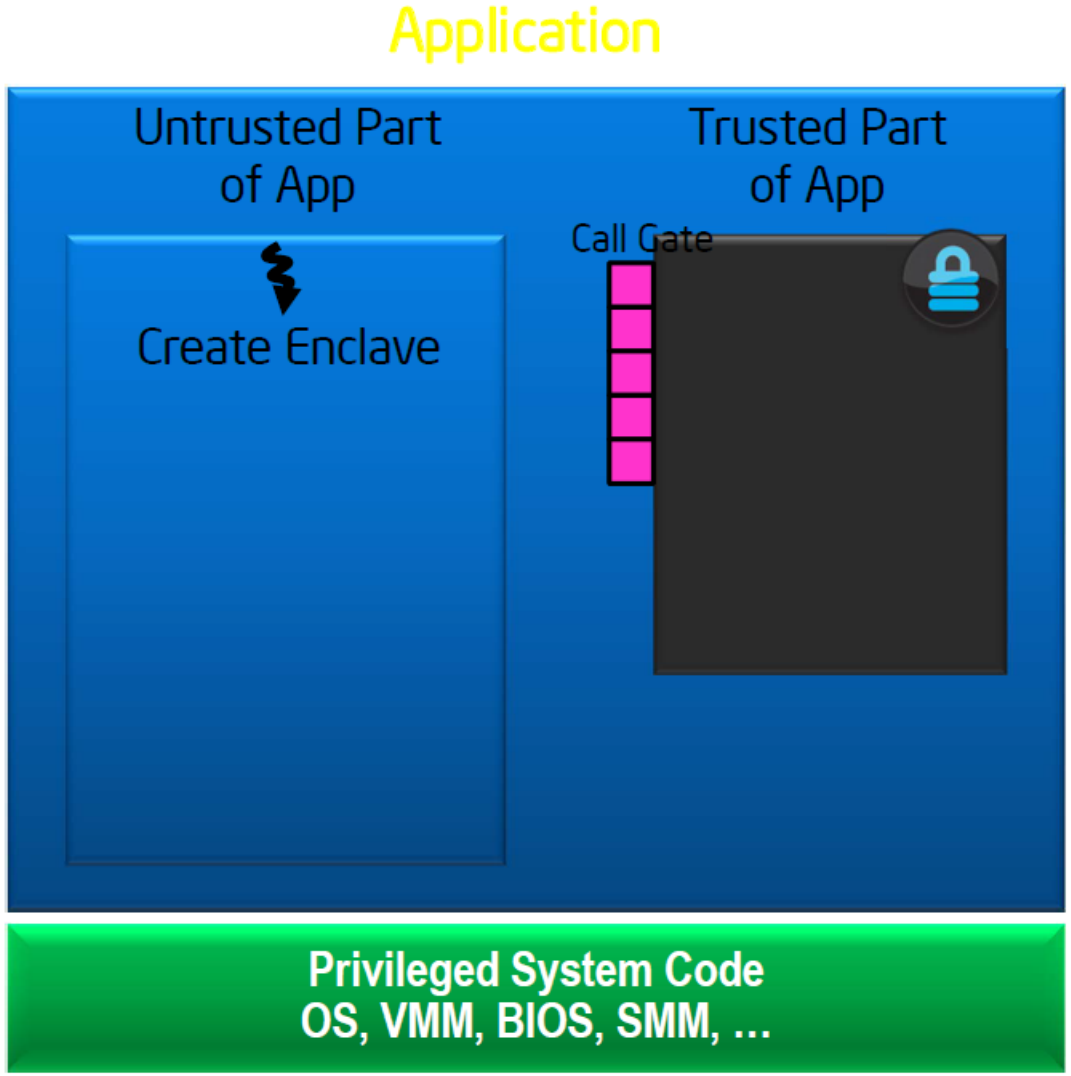


- Application gains ability to defend its own secrets.
- Reduced Attack Surface.

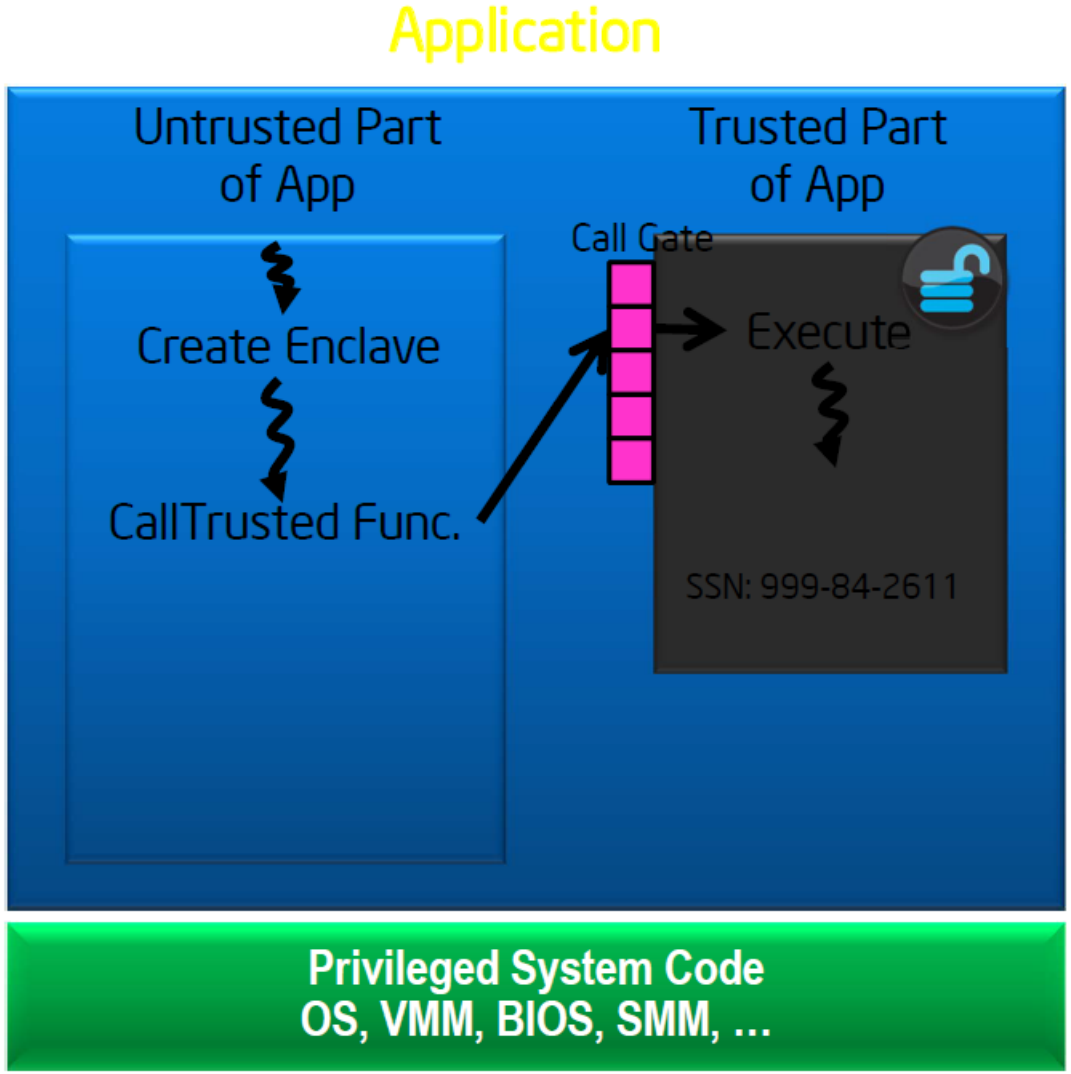
Intel - SGX : Concept of Enclaves



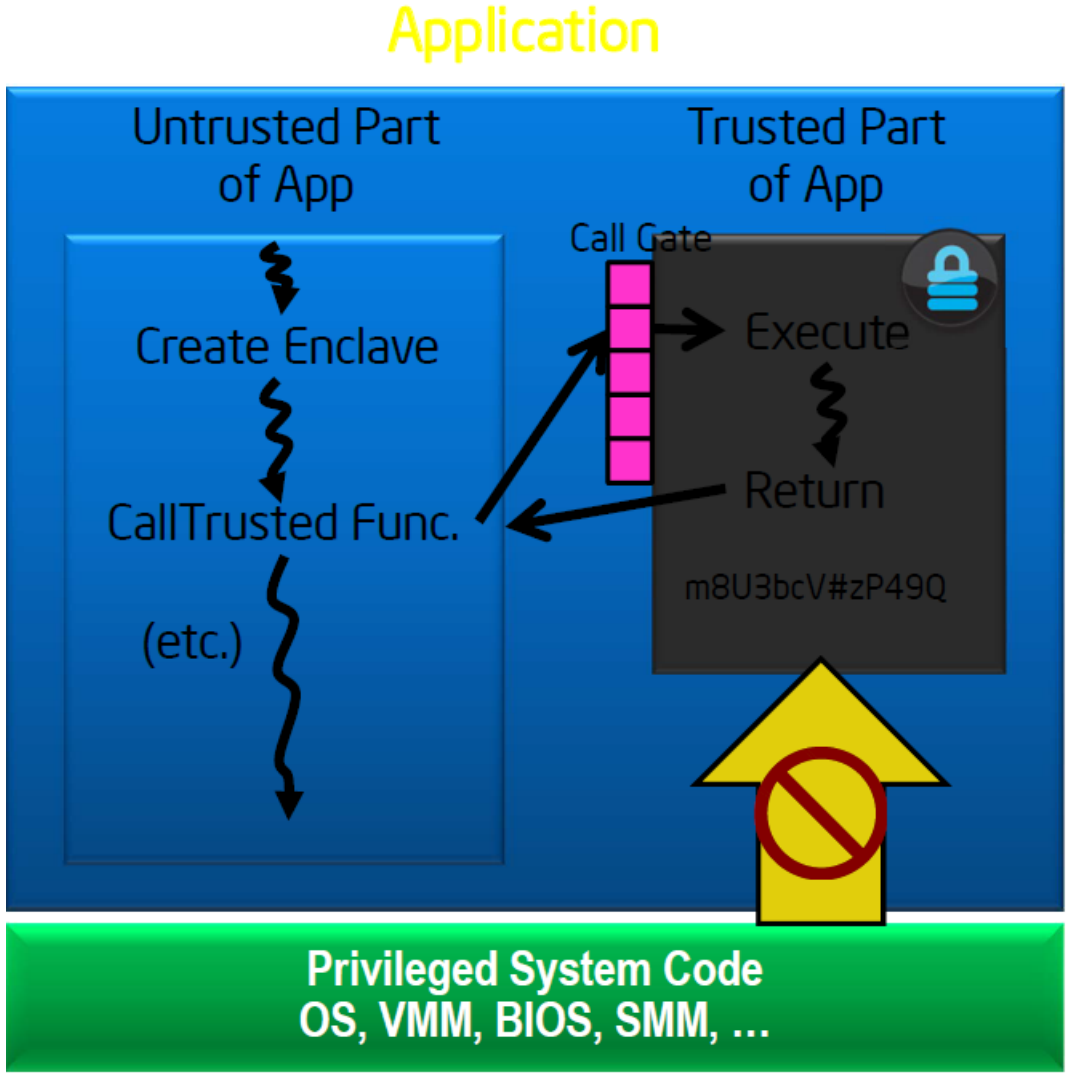
Intel - SGX : Concept of Enclaves



Intel - SGX : Concept of Enclaves

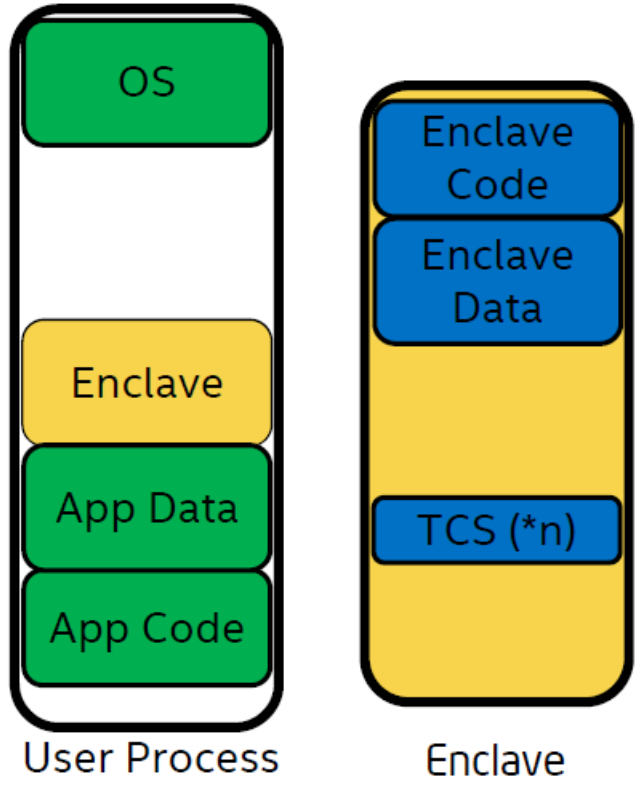


Intel - SGX : Concept of Enclaves

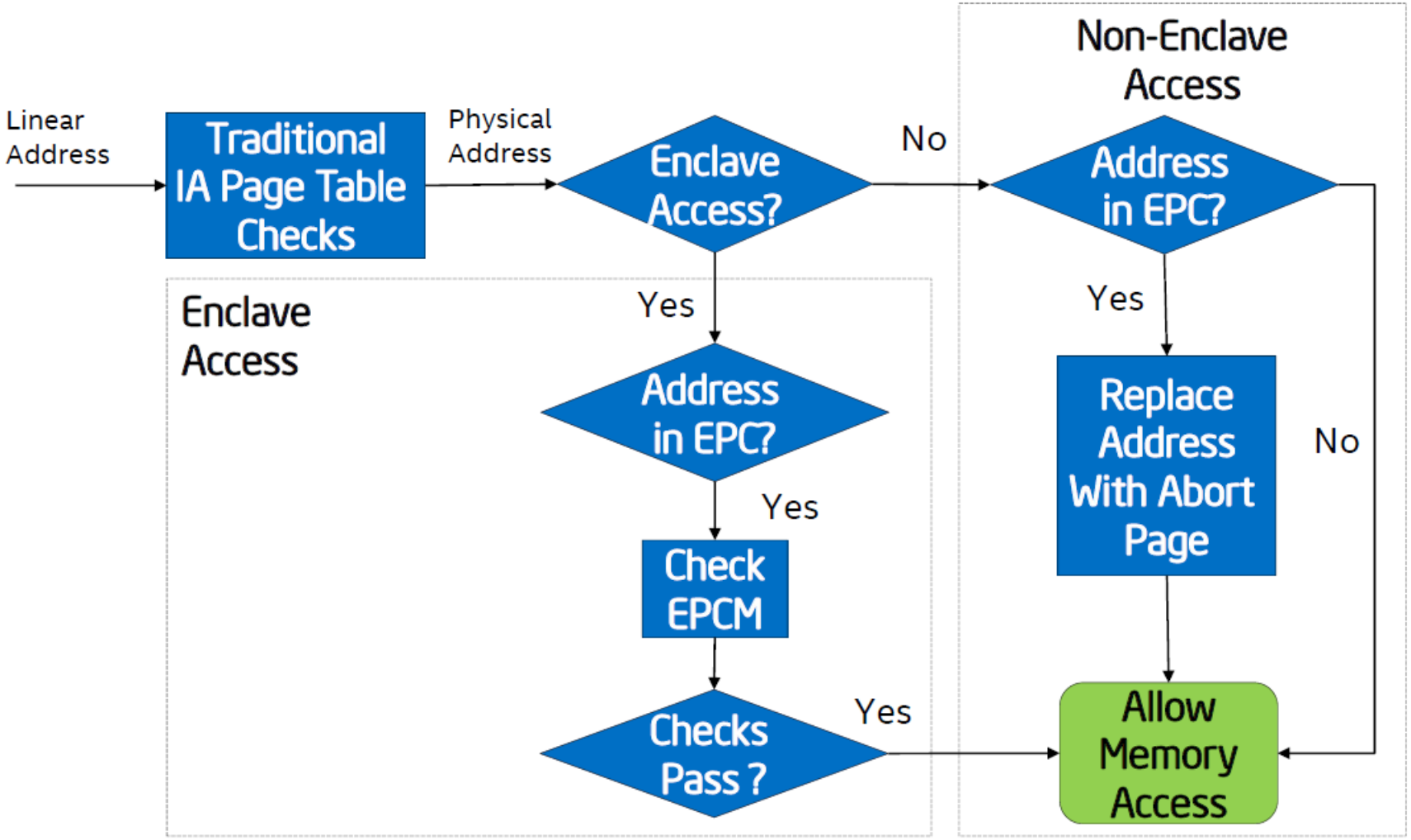


Intel - SGX : Feature of Enclaves

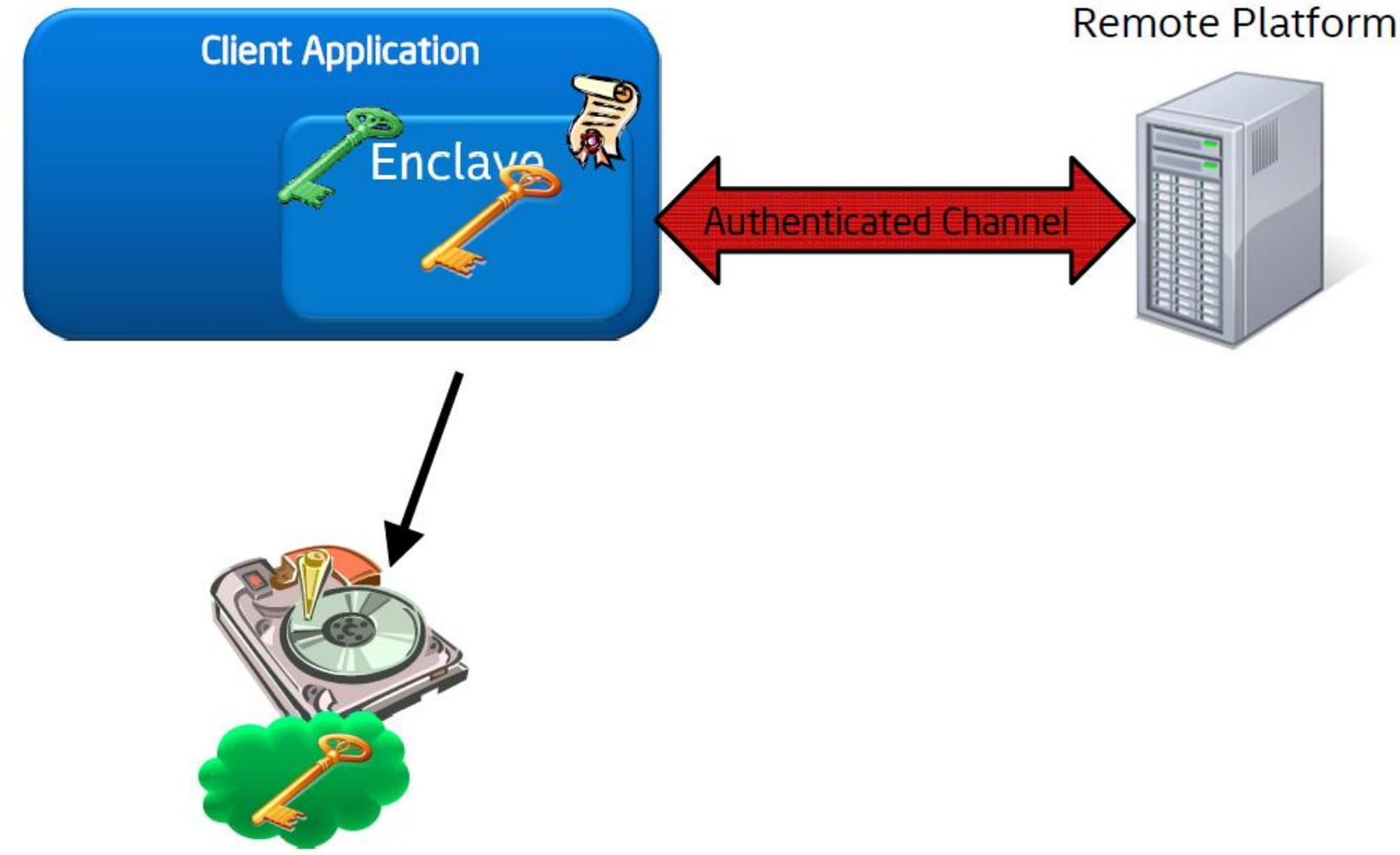
- With its own code and data
- Provide Confidentiality
- Provide integrity
- With controlled entry points
- With full access to app memory



Intel - SGX : Access Control



Intel - SGX : Attestation and Sealing

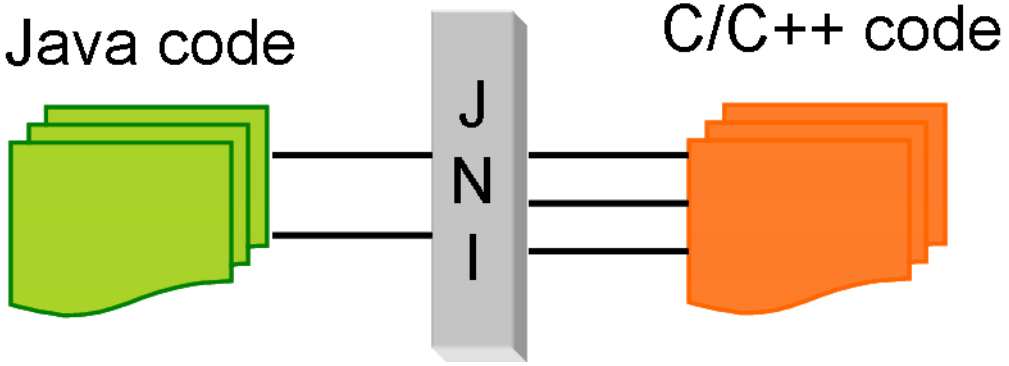


Currently libraries available for implementation in C/C++.

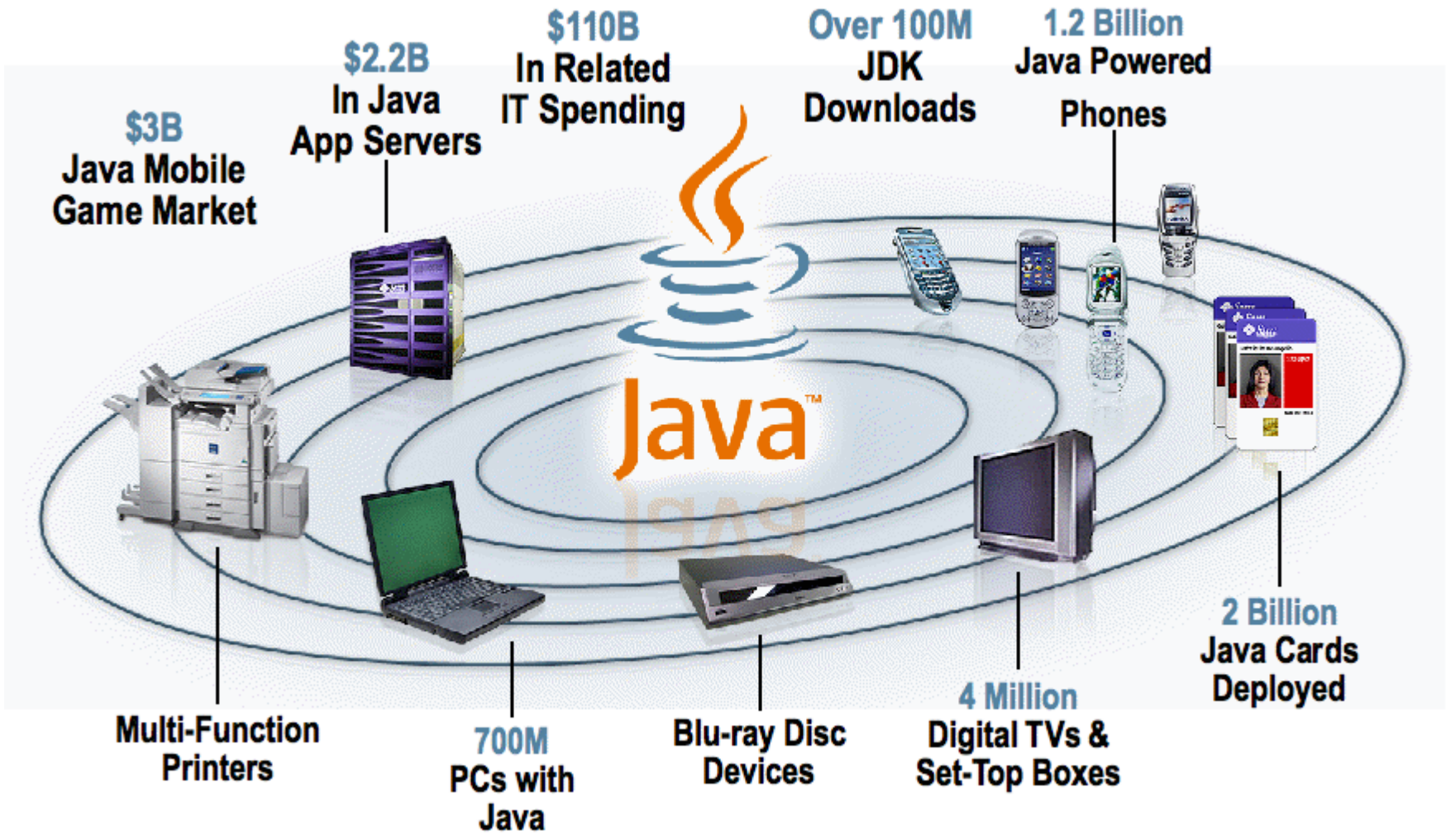
How do we implement it into our old friend Java ?

Java Native Interface (JNI)

Our Current Objective



Java is widespread, needs no introduction



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.

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 applications .(programs specific to a hardware and operating system platform).

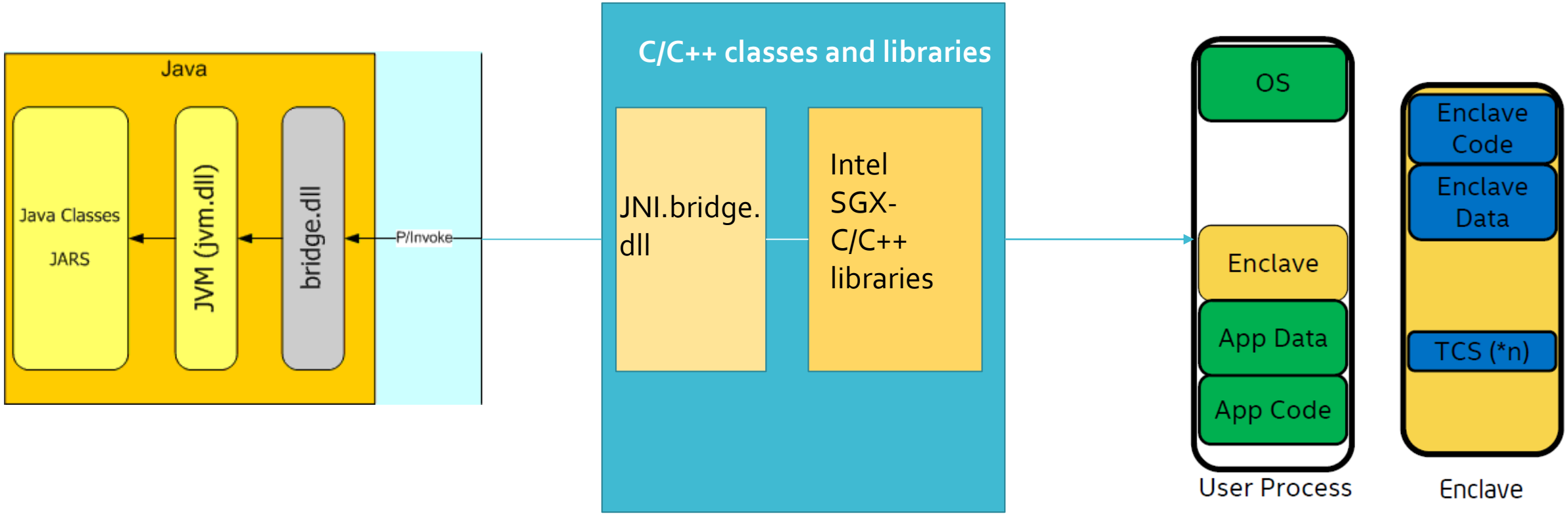
- In the JNI framework, native functions are implemented in separate .c or .cpp files. (Intel SGX has native functions)

- JVM invokes the function, it passes a JNIEnv pointer, a jobject pointer, and any Java arguments declared by the Java method.

Our Objective : Approach for creating secure Interpreter

- 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 : Creating a secure interpreter in Java



Thank You!