Lab 03 - Hardware Security

Objectives

- Hardware Security Basics
- Side Channel Attacks
- HSMs: Java Card & Simulator

Preparation

You may use the UPB's OpenStack cloud to instantiate a Virtual Machine to be used for this lab! Read these instructions if you wanna know how!.

Overview

Basics

For a long time, hardware had a central role in computer security. Take, for example, the CPU's protection rings model (on x86): they realize a privilege separation between a hypervisor / Operating System kernel and the user applications and is enforced at hardware-level for efficiency.

Nowadays, the security requirements of certain applications has led to the implementation of additional access control or cryptographic functions directly into the hardware, e.g., AES-NI / SHA SSE-based instructions, the Trusted Platform Module cryptoprocessor, Smart Cards or Trusted Execution Environments (ARM TrustZone, Intel SGX, memory encryption etc.).

On a different note, hardware is also susceptible to security bugs: side channel attacks, cryptographic vulnerabilities (e.g., cache or timing attacks or the much recent Spectre / Meltdown speculative execution bugs), hardcoded credentials or even manufacturer-introduced backdoors. These are very difficult (or even impossible) to fix without re-designing the chip and replacing the faulty products.

Side Channel Attacks

A side-channel attack is a type of cyber-attack that targets the unintended *side effects* of a software application / hardware component in a computer system, rather than attacking it directly. These side effects may include signals or data that are generated by the system's physical components, such as its power consumption, electromagnetic emissions, or even sound.

By analyzing these side effects, an attacker can gain information about the system's operations, such as encryption keys or other sensitive data, without directly accessing the system. Side-channel attacks can be executed remotely or locally and are often used to target cryptographic systems that use secret keys.

Java Smart Cards

Java smart cards are small electronic devices that have an embedded microprocessor and memory, which can be programmed with Java Card technology to perform secure transactions and store sensitive information. Java Card technology is a subset of Java that has been designed specifically for smart card environments.

Java smart cards can be found in various forms, such as SIM cards in mobile phones, banking cards, e-passports, and employee ID cards. They can be purchased from smart card manufacturers or vendors, or provided by organizations to their customers or employees.

To program and manage Java smart cards, specialized software development kits and tools are needed, such as Java Card Development Kit (JCDK), Global Platform Card Specification, and Smart Card Integrated Development Environment (IDE). Developers can use these tools to create and test Java Card applets that run on the smart cards and perform various secure operations.

Tasks

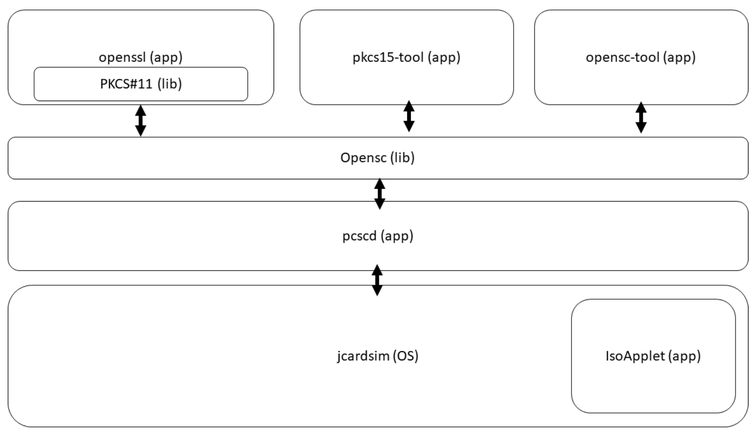
[40p] 1. Python timing side channel attack

Download the side channel demo archive here.

Implement the TODOs and crack the password via a timing attack ;)

Hint: you have LunarVim installed on the VM, use `lvim ${file}` to start it!

[60p] 2. Java Card Simulator



In order to simulate a Java Card, we must install all required Java components (Oracle JavaCard SDKs, jCardSim, IsoApplet, VSmartCard).

Note: you don't need to do this on the ISC VM 2023.1 (has already been setup)!

Click to display ↘

[Re]Start the PCSC service:

```
# the VM does not automatically start this, so do it manually
sudo systemctl restart pcscd
```

Start the simulator:

```
java -classpath "$HOME/jcardsim/target/jcardsim-3.0.5-SNAPSHOT.jar:$HOME/IsoApplet/src" com.licel.jcards
```

Hint: use separate terminals or `tmux`, since the command is blocking ;)

Loading the Smart Card applet:

The APDU Smart Card Application Protocol Data Unit is a communication protocol used for interfacing with smart cards, standardized in ISO/IEC 7816-4.

We use an initial APDU script to install & execute the IsoApplet into the emulated smart card.

```
# install IsoApplet usign a APDU script
opensc-tool --card-driver default --send-apdu 80b800001a0cf276a288bcfba69d34f310010cf276a288bcfba69d34f3
opensc-tool -n

# create PKCS#15 structure on our smart card (also set a PIN and a PUK, for security purposes)
pkcs15-init --create-pkcs15 --so-pin 123456 --so-puk 0123456789abcdef
# generate an RSA key pair to use for signing (note: auth-id is a PIN slot)
pkcs15-init --generate-key rsa/2048  --id 1 --key-usage decrypt,sign --label MyRSAKey --auth-id FF --pin
# download the generated public key to your machine
pkcs15-tool  --read-public-key "1" --output "smartcard-pubkey.pem"

echo "Sunt de acord să cedez toată averea mea asistenților de ISC. Adevăraaat\!" > textToSign.txt
openssl dgst -engine pkcs11 -sign "pkcs11:object=MyRSAKey;type=private;pin-value=123456" -keyform ENGINE

# now everyone can check whether the document is correctly signed using the public key:
openssl dgst -sha256 -verify smartcard-pubkey.pem -keyform PEM -signature textSignature.sig textToSign.t
# modificați fișierul textToSign.txt și re-verificați semnătura digitală... ce se întâmplă?
```

Finally, here's one last challenge: use `openssl` to encrypt / decrypt a file using this key!

Bonus: You can configure OpenSSH to use a private key stored inside a smart card for authentication!

3. Feedback

Please take a minute to fill in the feedback form for this lab.

Lectures
- Lecture 01 - Introduction
- Lecture 02 - Cryptography
- Lecture 03 - Hardware Security
- Lecture 04 - Access Control
- Lecture 05 - Authentication and Key Establishment
- Lecture 06 - Application Security
- Lecture 07 - Operating System Security
- Lecture 08 - Network Security
- Lecture 09 - Web Security
- Lecture 10 - Privacy Preserving Technologies
- Lecture 11 - Forensics

Labs
- **kernel**
- Lab 01 - Introduction
- Lab 02 - Cryptography
- Lab 03 - Authentication and Key Establishment Lab
- Lab 03 - Hardware Security
- Lab 04 - Access control
- Lab 05 - Authentication in Linux
- Lab 06 - Application Security
- Lab 07 - Operating System Security
- Lab 08 - Network Security
- Lab 09 - Web Security
- Lab 10 - Forensics
- Lab 11 - Privacy Technologies
- Lab 12 - Security and Machine Learning

Support
- Useful resources
- Virtual Machine

Table of Contents
- Lab 03 - Hardware Security
  - Objectives
  - Preparation
  - Overview
  - Basics
  - Side Channel Attacks
  - Java Smart Cards
  - Tasks
    - [40p] 1. Python timing side channel attack
    - [60p] 2. Java Card Simulator
    - 3. Feedback