PV204 Security technologies

Labs: JavaCard platform

Petr Švenda svenda@fi.muni.cz

Faculty of Informatics, Masaryk University

Masterplan for today

1. ~~10 minutes project gettogether~~

2. Pre-prepared project for NetBeans
   - Debugging of applet with simulated card (jcardsim.org)

3. Compilation of applet from the command line
   - ant-javacard, GlobalPlatformPro for installation
   - Select applet and obtain random data

4. Quick switch from simulated to real card

5. Automation via gradle template
   - Use from command line
   - Import into IntelliJ IDEA

Setup SimpleAPDU [NetBeans]

- Run NetBeans and import existing project
  - File → Open project → SimpleAPDU
  - (Now contains only host application)

- Add jar library file with simulator
  - Libraries→Add JAR→\lib\jcardsim-3.0.5.5.jar

- Add link to folder with on-card applet
  - File → Project properties → Sources →Add folder
  - Add SimpleApplet\src
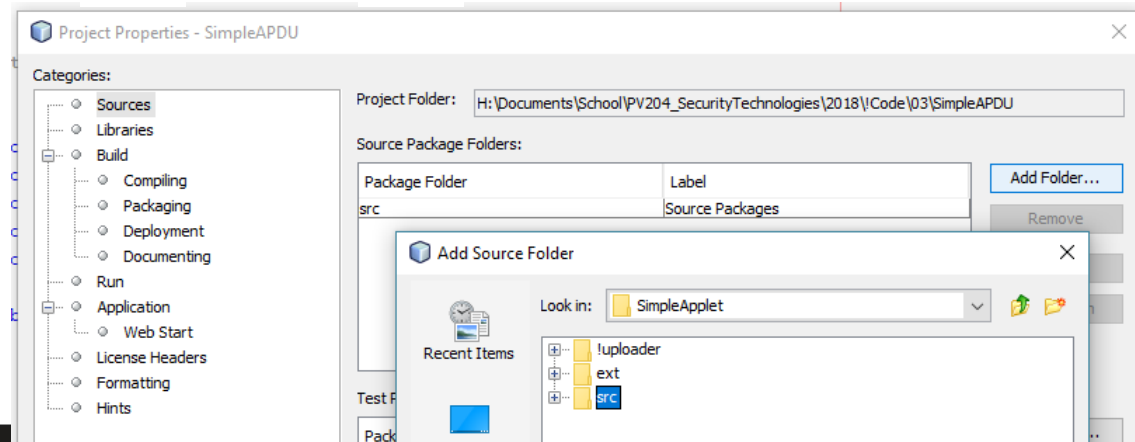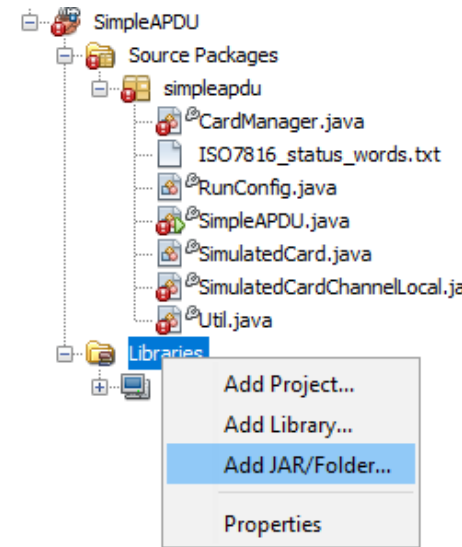
Setup SimpleAPDU [NetBeans]

- Project properties → Run → VM Options
  - `-enableassertions -noverify`
  - (required when custom compilation of jcardsim is used)
  - Not anymore, if you will use provided jcardsim-3.0.5.5.jar

- Project should now compile

- Try to run in debug mode
  - Place breakpoint inside applet's code

SIMULATED CARD (JCARDSIM)

Task: demoGetRandomDataCommand()

- Example code with single APDU command
- Simulated card is used
- Investigate the code, observe the values returned
- Place breakpoints and observe in debugger:
  - SimpleApplet constructor
  - process() method
  - Random() method

Tasks: demoEncryptDecrypt()

- Investigate method demoEncryptDecrypt
- Task 1: Prepare and send APDU with 32 bytes of data for encryption, observe output
- Task 2: Extract the encrypted data from the card's response. Send APDU with this data for decryption
  - Compare data for encryption with the decrypted data
- Task 3: What is the value of AES key used inside applet? Use debugger to figure this out
- Task 4: Prepare and send APDU for setting different AES key, then encrypt and verify

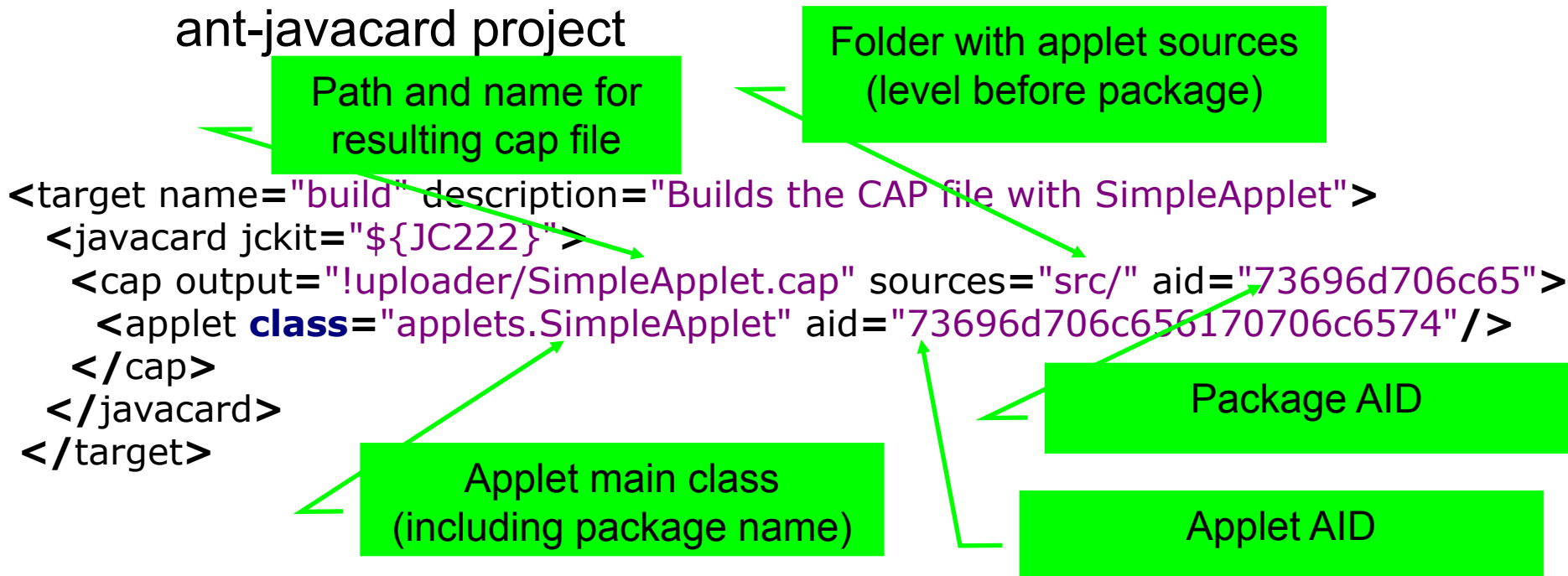This applies only if you have older/unused banking card and smartcard reader

CONVERSION AND UPLOAD TO REAL CARD

Real cards –CPLC of your bank card

- Obtain ATR and CPLC info for your bank card
  - `java -jar gp.jar --info -d`
  - Don't worry, nothing is changed on card
- Can you figure out IC manufacturer and chip type?
- Can you figure out Operating System ID and name?
- Can you figure out fabrication date?
- Try to use cplc.py for reference
  - https://github.com/crocs-muni/JCAlgTest/blob/master/AlgTest_Process/cplc.py
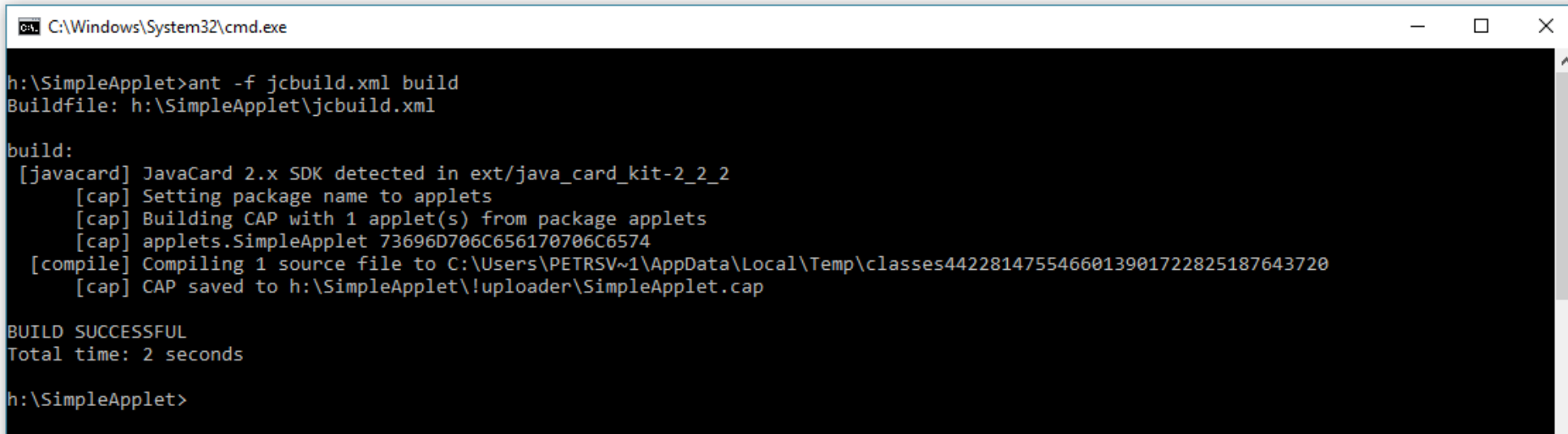- Can you locate CPLC in your project certificate?

Task: Create cap file and upload to card

- Navigate to SimpleApplet folder
  - src folder contains applet's source code in SimpleApplet.java
  - **jcbuild.xml** contains configuration for conversion with ant-javacard project

Path and name for resulting cap file

Folder with applet sources (level before package)

```
<target name="build" description="Builds the CAP file with SimpleApplet">
  <javacard jckit="${JC222}">
    <cap output="!uploader/SimpleApplet.cap" sources="src/" aid="73696d706c65">
      <applet class="applets.SimpleApplet" aid="73696d706c656170706c6574"/>
    </cap>
  </javacard>
</target>
```

Package AID

Applet main class (including package name)

Applet AID

Task: Create cap file and upload to card

- Compile & Convert
  - Execute on cmd line: `ant -f jcbuild.xml build`


```
C:\Windows\System32\cmd.exe                                              —    □    ✕

h:\SimpleApplet>ant -f jcbuild.xml build
Buildfile: h:\SimpleApplet\jcbuild.xml

build:
 [javacard] JavaCard 2.x SDK detected in ext/java_card_kit-2_2_2
      [cap] Setting package name to applets
      [cap] Building CAP with 1 applet(s) from package applets
      [cap] applets.SimpleApplet 73696D706C656170706C6574
  [compile] Compiling 1 source file to C:\Users\PETRSV~1\AppData\Local\Temp\classes4422814755466013901722825187643720
      [cap] CAP saved to h:\SimpleApplet\!uploader\SimpleApplet.cap

BUILD SUCCESSFUL
Total time: 2 seconds

h:\SimpleApplet>
```

- If OK, SimpleApplet.cap is created in !uploader folder

Task: Create cap file and upload to card

- http://github.com/martinpaljak/GlobalPlatformPro

1. List already loaded applets
   - `java -jar gp.jar –list -d`

2. Uninstall previous version of SimpleApplet
   - `java -jar gp.jar -uninstall SimpleApplet.cap -d`

3. Install SimpleApplet.cap
   - `java -jar gp.jar -install SimpleApplet.cap -d`

4. Use applet (commands in SimpleAPDU code)

Problem: what with other applets on card?

1. List already loaded applets
   - `java -jar gp.jar –list -d`

2. Find package_AID and run:
   - `java -jar gp.jar –deletedeps –delete package_aid`
   - The `–deletedeps` will also delete all applets from target package

- E.g., our SimpleApplet can be also removed by
  - `gp –deletedeps –delete` 73696d706c65

Be aware – real card can be blocked

- Too many unsuccessful authentication requests

```
>gp --list -debug
# Detected readers from SunPCSC
[*] Alcor Micro USB Smart Card Reader 0
SCardConnect("Alcor Micro USB Smart Card Reader 0", T=*) -> T=0, 3BF71800008031F
E45736674652D6E66C4
SCardBeginTransaction("Alcor Micro USB Smart Card Reader 0")
A>> T=0 (4+0000) 00A40400 00
A<< (0018+2) (56ms) 6F108408A000000003000000A5049F6501FF 9000
A>> T=0 (4+0008) 80500000 08 6265E168FB2639C1
A<< (0028+2) (118ms) 00003126960097543174010200103595AC1420213D2969EA8B8C41F3 9
00
```

openkms.gp.GPException: STRICT WARNING: Card cryptogram invalid!
Card: 3D2969EA8B8C41F3
Host: DB1E6E1E71958A15
!!! DO NOT RE-TRY THE SAME COMMAND/KEYS OR YOU MAY BRICK YOUR CARD !!!
    at openkms.gp.GlobalPlatform.printStrictWarning(GlobalPlatform.java:156)

    at openkms.gp.GlobalPlatform.openSecureChannel(GlobalPlatform.java:471)
    at openkms.gp.GPTool.main(GPTool.java:348)

Be aware – real card can be blocked

- Don't write script that executes many authentications at once (cycle, multiple commands)
- If unsuccessful one/two authentication is detected, then as for help, please!!!

USE OF REAL CARD

Tasks: demoUseRealCard()

- Change from simulator to real card
  - runCfg.setTestCardType(RunConfig.**CARD_TYPE.PHYSICAL**);

- Task 5: Obtain random data from real card
- Task 6: Set new key value and encrypt on card

GRADLE TEMPLATE

Usage of gradle template

- Intended for automatic conversion and tests
  - Used by TravisCI…
- IntelliJ IDEA can import project from gradle scripts
  - Interactive debugging, test coverage…
- Download complete zip
  - https://github.com/crocs-muni/javacard-gradle-template-edu/releases/download/v0.2.0-edu/javacard-gradle-template-edu.zip
- Read README.md carefully!
  - ./gradlew buildJavaCard --info --rerun-tasks
  - ./gradlew test --info --rerun-tasks

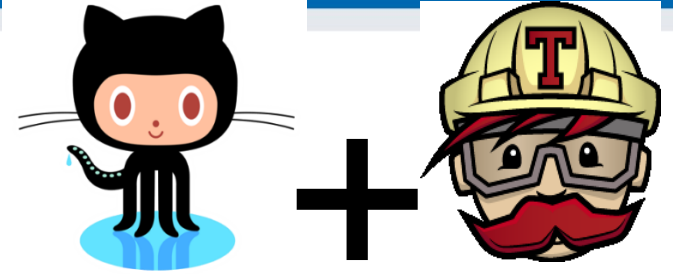# Integration with TravisCI

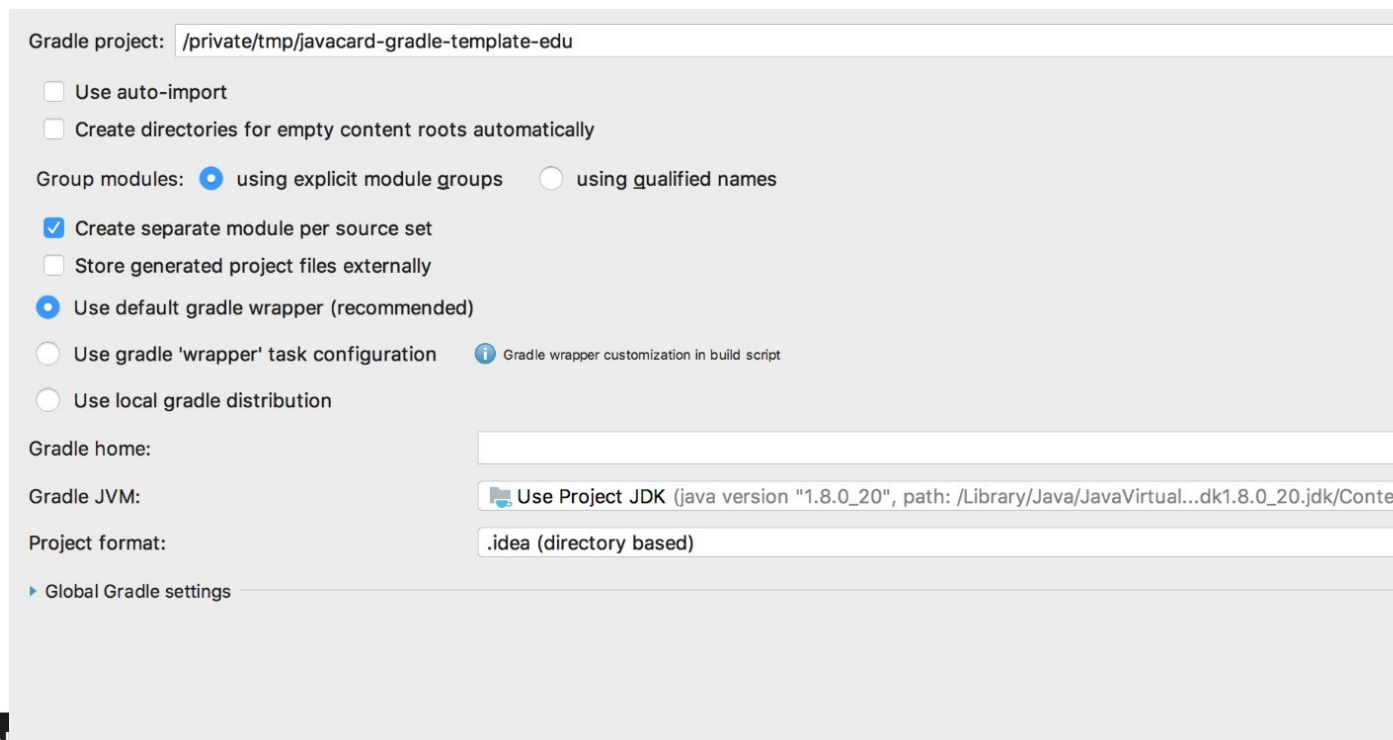- .travis.yml

```
language: java

jdk:
  - oraclejdk8

script:
  - ./gradlew check --info
  - ./gradlew buildJavaCard --info
```

Gradle + IntelliJ IDEA

- IntelliJ IDEA supports import of gradle –based projects (simpler IDE NetBeans not)
- Debugging with simulated card works as in NetBeans

Gradle project: /private/tmp/javacard-gradle-template-edu

☐ Use auto-import
☐ Create directories for empty content roots automatically

Group modules: ● using explicit module groups   ○ using qualified names

☑ Create separate module per source set
☐ Store generated project files externally
● Use default gradle wrapper (recommended)
○ Use gradle 'wrapper' task configuration   ⓘ Gradle wrapper customization in build script
○ Use local gradle distribution

Gradle home:

Gradle JVM: 📁 Use Project JDK (java version "1.8.0_20", path: /Library/Java/JavaVirtual...dk1.8.0_20.jdk/Conte

Project format: .idea (directory based)

▸ Global Gradle settings

Test coverage: Gradle + IntelliJ IDEA

- Go to Gradle plugin in IntelliJ IDEA

- Tasks $\rightarrow$ verification $\rightarrow$ test

- RClick $\rightarrow$ run with coverage

HOMEWORK

Assignment 4: OpenFIPS201 analysis

- Analyze existing JavaCard applet
  - https://github.com/makinako/OpenFIPS201
- Answer the following questions:
  - What cryptographic algorithms are executed on card and with what parameterization? (use JavaCard constants like ALG_RSA_NOPAD)
  - For every algorithm, find the JavaCard specification version which introduced it first (e.g., JC API 2.1.1 for ALG_RSA_NOPAD)
  - Describe purpose of usage (achieved functionality) for a particular algorithm (e.g., OwnerPIN cardPIN is used to protect RAM memory against fault induction)
  - Describe details of security mechanism used to protect APDUs going from and to applet
- Prepare applet to run in jcardsim simulator
  - Demonstrate working setup by description of process, issues encountered and screenshots
  - Send at least one APDU command

Assignment 4: OpenFIPS201 analysis

- Produce short (2xA4) text description of solution
  - Provide answers to questions asked
  - Provide demonstration of applet executed in simulator
- Submit **before 19.3.2020 23:59** into IS HW vault
  - Soft deadline: -1.5 points for every started 24 hours

Troubleshooting – jcardsim simulator

- Don't forget to add jcardsim-3.0.5.5.jar

- Use debugger – insert breakpoint directly into applet's method

- Local vs. remote simulator jcardsim
  - Only single card can be simulated as local one (CAD.getCardInterface())
  - We will use and debug only one card (so local is fine)
  - Multiple cards can be used as remote simulators (sockets)