Using Side Channel Information for Improving Data Partitioning Strategy to Test Smart Cards

Benjamin Putt Ethan Putt Jean-Louis Lanet

University of Limoges – Computer Science Department
jean-louis.lanet@unilim.fr
http://secinfo.msi.unilim.fr/lanet/

SAR-SSI 2014

Université de Limoges

Our Motivation

- We have a **black box** approach;
- A smart card application may contain any **hidden feature**;
- If some commands are hidden, **how to find it?**

Université de Limoges

Outline

Université de Limoges

Fuzzing to Discovering Vulnerabilities

- It is a technique used to uncover vulnerabilities
- 3 modules
  - Data Generator
  - Delivery Mechanism
  - Monitoring System

Université de Limoges

State-of-the-art Fuzzing Techniques

Mutation-based fuzzer

- Mutates correct data to invalid one.

Generation-based fuzzer

- More smart that the mutation-based fuzzer
- Commands are generated from the specification

Université de Limoges

Current Fuzzers Limitations

- Tests case combinatorial explosion!
- Limiting the test coverage
  - Generate values around the input
- Partitioning the input domain
  - Specification or implementation is required
  - Does not work on a black box context

Université de Limoges

The Smart Card

- Widely used device
  - Credit Card;
  - (U)SIM Card;
  - Health Card (french Vitale card);
  - Pay TV;
  - ...
- Most of them embed a Java Virtual Machine

Université de Limoges

The Smart Card

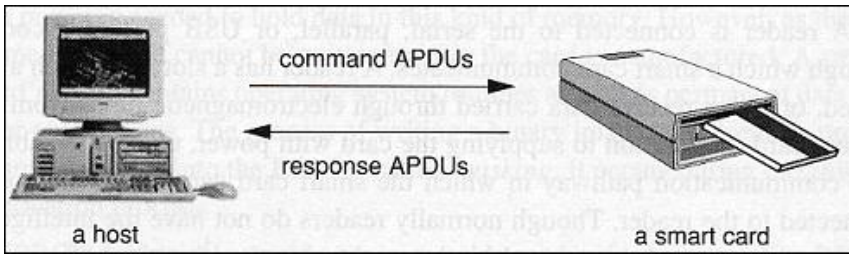
- Widely used device
  - Credit Card;
  - (U)SIM Card;
  - Health Card (french Vitale card);
  - Pay TV;
  - ...
- Most of them embed a Java Virtual Machine

How to test a Java Card Application?

- Model-based approach
- Only dedicated for functional testing ...

Java Card Application

The APDU command



APDU Command

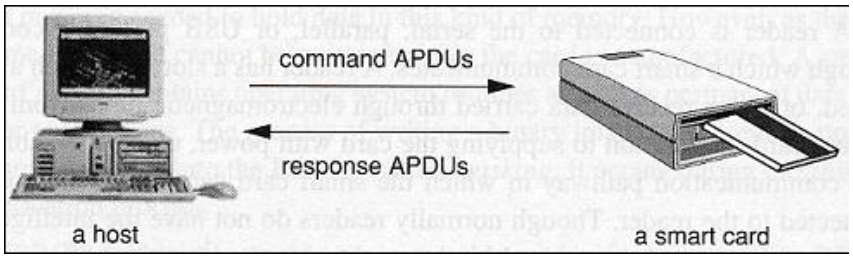
Header				Body		
CLA	INS	Parameter 1	Parameter 2	Lc	Data field	Le

APDU Response

Data field	Status Word

Java Card Application

The APDU command



APDU Command

Header				Body		
CLA	INS	Parameter 1	Parameter 2	Lc	Data field	Le
P_0	P_1	P_2	P_3			

APDU Response

Data field	Status Word

Fuzzing a Java Card Application

- C&SAR 2011: Testing on Java Card Web Server upon several smart card's protocols; (Barreaud *et al.*)
  - Testing an implementation from its specification
- SSTIC 2011: Fuzzing on EMV protocol; (Lancia)
  - Reference Implementation Vs tested implementation
- ECIWS 2010: Fuzzing to discover hidden command; (Guyot)
- Alimi's Master Thesis (2012): Fuzzing based on genetic algorithm

Université de Limoges

Fuzzing a Java Card Application

- C&SAR 2011: Testing on Java Card Web Server upon several smart card's protocols; (Barreaud *et al.*)
  - Testing an implementation from its specification
- SSTIC 2011: Fuzzing on EMV protocol; (Lancia)
  - Reference Implementation Vs tested implementation
- ECIWS 2010: Fuzzing to discover hidden command; (Guyot)
- Alimi's Master Thesis (2012): Fuzzing based on genetic algorithm

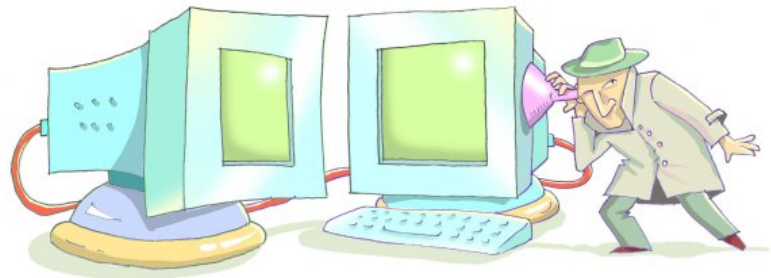
How to choose the input data?

Université de Limoges

Side Channel Attacks

"In cryptography, a side channel attack is any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms (compare cryptanalysis)" (source: Wikipedia)

- Timing attacks;
- Power monitoring attacks;
- Electromagnetic attacks;
- Acoustic cryptanalysis;
- Differential fault analysis;
- Data remanence.



Université de Limoges

How to generate interesting input data?

- Combined timing attack and fuzzing
- Partitioning regarding to the time response

Université de Limoges

Step 1: Get Evaluation Order

Data: P the set of input parameters
Result: C the set of data partition
$Min \leftarrow \emptyset$; $C \leftarrow \emptyset$; $InitParameter()$;
for $i \leftarrow 0$ **to** $MAXPARAMS$ **do**
  $P_i \leftarrow NON\_VALID\_VALUE$;
  $t_{i,j} \leftarrow Send(P_{i,j}, data)$;
end
$SortedOrder \leftarrow Sort(t_{i,j})$;

- Testing incorrect values from the specification.
- Only one incorrect value by parameter is tested

Université de Limoges

Steps 2 & 3: Hidden Command & Build Partition

for $i \in 0$ **to** *MAXPARAMS using the SortedOrder* **do**
  for $j \leftarrow 0$ **to** *MAXBYTE* **do**
    $P_i \leftarrow j$;
    $t_{i,j} \leftarrow Send(P_{i,j}, data)$;
  end
  $InitParameter()$; $Min_i \leftarrow Sort(t_i)$;
  for $j \leftarrow 0$ **to** *MAXBYTE* **do**
    $Mean(t_{i,j})$;
    if $t_{i,j} > Mean(t_{i,j}) + \delta$ **then**
      $C_i \leftarrow CreateNewPartition(j)$;
    end
  end
end

- Each element is tested from the evaluation order.
- The response time depends to a card and reader delay (δ)
- According to the response time, a partition is created

Step 4: Fuzz it!

Sort(*Min*);
DefineCFG(*Min*);
// Fuzzing step
for $i \leftarrow 0$ **to** *MAXPARAMS* **do**
  for $j \in$ *each subdomain of* C_i **do**
    $Res_{i,j} \leftarrow Send(P_{i,j}, data)$;
  end
end

- Resorting the response time to take in count the hidden command
- Generating the real tested application CFG
- Fuzzing each application's subdomain

Université de Limoges

Example: a One Time Password I

- One Time Password (OTP) generates keyed hash using DES algorithm
- The specification defines:
  - $P_0 = 0x00$
  - $P_1 \in \{0x34, 0x36, 0x38\}$
  - $P_2 = 0x00$
  - $P_3 \in [0x50, 0x57]$ for the get and put data command
  - $P_3 \in \{0x0, 0x1\}$ for the verify PIN command

Université de Limoges

Example: a One Time Password II

```
public void process(APDU apdu) throws ISOException {
  if (selectingApplet()) return;
  byte[] cmd_apdu = apdu.getBuffer();
  Util.arrayFillNonAtomic(wy, 0, LEN_WY, 0);
  if (cmd_apdu[ISO7816.OFFSET_CLA]
                    == ISO7816.CLA_ISO7816 /*=0x00*/) {
    switch(cmd_apdu[ISO7816.OFFSET_INS]) {
      case INS_VERIFY /*0x34*/:cmdVERIFY(apdu); break;
      case INS_PUTDATA/*0x36*/:cmdPUTDATA(apdu);break;
      case INS_GETDATA/*0x38*/:cmdGETDATA(apdu);break;
      default : ISOException.throwIt
                  (ISO7816.SW_INS_NOT_SUPPORTED);
  } } else {
   ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
} }
```

Fuzzing Applied to the OTP Implementation I

- We applied the fuzzing approach on the OTP implementation;
- The evaluation order of each parameter is discovered:
  1. P_0 (CLA)
  2. P_1 (INS)
  3. P_2 (P1)
  4. P_3 (P2)
- Two evaluations process: without and with the PIN validation.

Université de Limoges

Fuzzing Applied to the OTP Implementation II

- We search the hidden command & we build the partition;
- We test the 255-value of the P_0 (CLA) parameter:
  - **2 partitions** are discovered
  - One for P_0 (CLA) equals 0
  - One for the rest;
- The same process is done with other parameters:

Université de Limoges

Fuzzing Applied to the OTP Implementation III

```
private void cmdGETDATA(APDU apdu) {
  byte[] cmd_apdu = apdu.getBuffer();
  // if ((P1 != 0) || (P2 < 0x50) || (P2 > 0x57))
  //          => Exception(SW_WRONG_P1P2)
  short le = (cmd_apdu[ISO7816.OFFSET_LC] & 0x00FF);
  switch(tag) { ...
    case (byte) 0x57: // get a new NSU
     if (le != (byte)8)// Exception(SW_WRONG_LENGTH)
     if (!userpin.isValidated())
      // Exception(SW_SECURITY_STATUS_NOT_SATISFIED)
     if (stateMachine!=INIT_DONE)
        // Exception(SW_DATA_INVALID+2)
     generateNSU();  ...
    default: // => Exception(SW_DATA_NOT_FOUND)
  } apdu.setOutgoingAndSend ((short) 0, (short) le); }
```

Université de Limoges

Evaluation

- A learning step is needed to find the δ value (response time).
- Checking the code coverage of our approach
  - Developing a **code coverage API** for Java Card
- Our approach works due to the particular way of programming Java Card applet.
- Our approach is limited to the number of partitions of each parameter: **144 tests** to discover any **hidden command**.
- Testing each APDU header possibilities requires **264 435 456 tests**.

Université de Limoges

Limits of the Approach

- On a **constant time application**, none hidden command will be discovered;
- Incorrect commands can **kill** the targeted card;
- Our approach is currently limited to the **APDU header**;
- Our approach is mainly **limited** to the smart card world . . .

Université de Limoges

To Conclude

- A new method for testing software was presented;
  - Based on Side Channel Information to generate data;
  - Discover hidden command;
- An associated framework was designed to validate our approach;
- A constant time program cannot be analyzed ...

Université de Limoges

To Conclude

- A new method for testing software was presented;
  - Based on Side Channel Information to generate data;
  - Discover hidden command;
- An associated framework was designed to validate our approach;
- A constant time program cannot be analyzed ...

The Next Step

- Testing the APDU data fields;
- Use other side channel information leakages:
  - Electromagnetic field;
  - Analyze oscilloscope curves;
- Reversing Java Card applet using electromagnetic fields to obtain a better CFG.

Thank you for your attention! Do you have any questions?

Using Side Channel Information for Improving Data Partitioning Strategy to Test Smart Cards

Benjamin Putt Ethan Putt Jean-Louis Lanet

University of Limoges – Computer Science Department
jean-louis.lanet@unilim.fr
http://secinfo.msi.unilim.fr/lanet/

SAR-SSI 2014

Université de Limoges