Karl Koscher – @supersat
Eric Butler – @codebutler

The Secret Life of SIM Cards

Rev: July 8, 2013
Updated slides available from defcon.org and http://simhacks.github.io/

In an alternate reality…

- Applications live on your SIM card
- Phones are dumb hosts – UI and connectivity only
- Telcos own the SIMs, so they control the applications

Actually…

- These are called SIM Toolkit (STK) applications
- Used widely in the developing world
  - Mobile banking, etc.
- Very little documentation on how they work, or how to develop them

An opportunity!

- Toorcamp 2012 had its own GSM network
- SIM cards supported SIM Toolkit, so why not explore it?
- After lots of research, finally figured out how to program the *#$!ing things
- Learn from our misery

Why should you care?

- SIM cards are a lot more powerful than you might have imagined
- Development of STK apps is the same as Java Cards – you can use unlocked STK SIMs as cheap smart card development platforms
- Maybe if people care enough, SIMs can be better utilized (e.g. secure storage of SSH keys, BitCoins, phone decryption keys)

Okay, what can an STK app do?

- Rudimentary UI – display text, menus, play tones, read input
- Send SMSes, initiate calls, initiate and use data services
- Receive and act on events, such as call connected, call disconnected, etc.
- Interact with the rest of the SIM card
- Run arbitrary AT commands on the phone

Technologies involved

- Working our way up the technology stack:
  - Smart cards
  - Java Card
  - GlobalPlatform
  - SMS
  - SIM Toolkit API

Smart Card Standards

- ISO 7816-1: Physical characteristics
- ISO 7816-2: Electrical contacts
- ISO 7816-3: Electrical interface, Transmission Protocol Data Units (TPDUs)
  - T=0: Byte-oriented protocol
  - T=1: Block-oriented protocol
- ISO 7816-4: Standard commands, Application Protocol Data Units (APDUs)
- ISO 14443-4: "T=CL": APDUs over RFID

Smart Card Pinout from Wikimedia Commons: https://en.wikipedia.org/wiki/File:SmartCardPinout.svg

Smart Card Protocols: ISO 7816-4

- Command APDU

CLS Class		INS Instruction	P1 Param 1	P2 Param 2	Optional Command Dependent		
					LC Data Length	DATA	LE Length Expected
MSB	LSB						

8	Standard structure and CLS/INS
1-7	RFU
8-9	Standard structure, custom INS, standard CLS
A	Standard unless spec'd by context
B-C	Standard structure, custom CLS/INS
D-F	Custom

B_3	B_2	B_1	B_0	Meaning when CLS=0X,8X,9X,AX
X	X			Secure Messaging Format
0	X			No 7816-4 SM
0	0			No SM
0	1			Proprietary SM
1	X			7816-4 SM
1	0			Command header not auth'd
1	1			Command header auth'd
		X	X	Logical chan num

http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx

Smart Card Protocols: ISO 7816-4

- Command APDU

				Optional Command Dependent		
CLS Class (MSB / LSB)	**INS** Instruction	**P1** Param 1	**P2** Param 2	**LC** Data Length	**DATA**	**LE** Length Expected

INS	(CLS=0X, AX)
0E	ERASE BINARY
20	VERIFY
70	MANAGE CHANNEL
82	EXTERNAL AUTHENTICATE
84	GET CHALLENGE
88	INTERNAL AUTHENTICATE
A4	SELECT FILE
B0	READ BINARY
B2	READ RECORD(S)
C0	GET RESPONSE

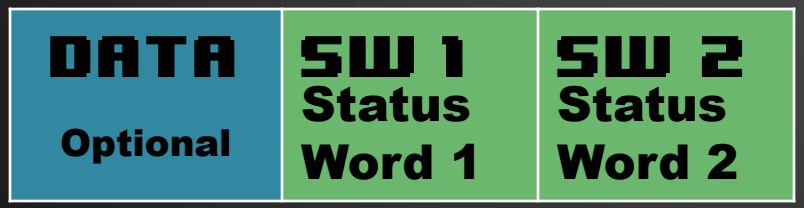
INS	(CLS=0X, AX)
C2	ENVELOPE
CA	GET DATA
D0	WRITE BINARY
D2	WRITE RECORD
D6	UPDATE BINARY
DA	PUT DATA
DC	UPDATE DATA
E2	APPEND RECORD

INS	(Any class)
6X	**Prohibited**
9X	**Prohibited**

http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx

Smart Card Protocols: ISO 7816-4

- Response APDU



SW1	Message
9X	Success
90	Success – no additional info

SW1	Error
61	SW2 bytes still available
62	NVM unchanged
63	NVM changed
64	NVM unchanged
65	NVM changed
66	Security issue
67	Wrong length
68	Functions in CLS not supported
69	Command not allowed
6A	Wrong parameter
6B	Wrong parameter
6C	Wrong Le: SW2 is exact length
6D	INS not supported or invalid
6E	CLS not supported
6F	Other/unknown

Smart Card Protocols

- Example: SELECT 1234

CLS Class	INS Instruction	P1 Param 1	P2 Param 2	LC Data Length	DATA	LE Length Expected
00	A4	00	00	02	12 34	

- Response:

DATA Optional	SW 1 Status Word 1	SW 2 Status Word 2
02 12 34	90	00

Java Card

- It's Java!
- … not really.
  - No garbage collection
  - No chars, no strings, no floats, no multi-dim arrays
  - ints are optional
  - No standard API, no threads, etc.
  - But there are Exceptions!
- Instance and class variables are saved in EEPROM, which has limited write cycles

Java Card

- Two mandatory methods:
  - install (static) – creates your application object and registers it with the card manager
  - process – handles APDUs sent to the card
    - Byte array in, byte array out
    - Not all bytes are immediately available due to packing of APDUs in TPDUs!

Java Card

```
package com.degdeg.HelloCard;

import javacard.framework.*;

public class HelloApplet extends Applet {
    private static byte[] msg = { 'H', 'e', 'l', 'l', 'o' };

    public static void install(byte[] bArray, short bOffset, byte bLength) {
        HelloApplet applet = new HelloApplet();
        applet.register();
    }

    public void process(APDU apdu) throws ISOException {
        byte[] buf = apdu.getBuffer();

        if (buf[ISO7816.OFFSET_CLA] != 0x80)
            ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);

        if (buf[ISO7816.OFFSET_CLA] != 0xA5)
            ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);

        Util.arrayCopyNonAtomic(msg, (short)0, buf, ISO7816.OFFSET_CDATA, (short)msg.length);
        apdu.setOutgoingAndSend(ISO7816.OFFSET_CDATA, (short)msg.length);
    }
}
```

Building Java Card Apps

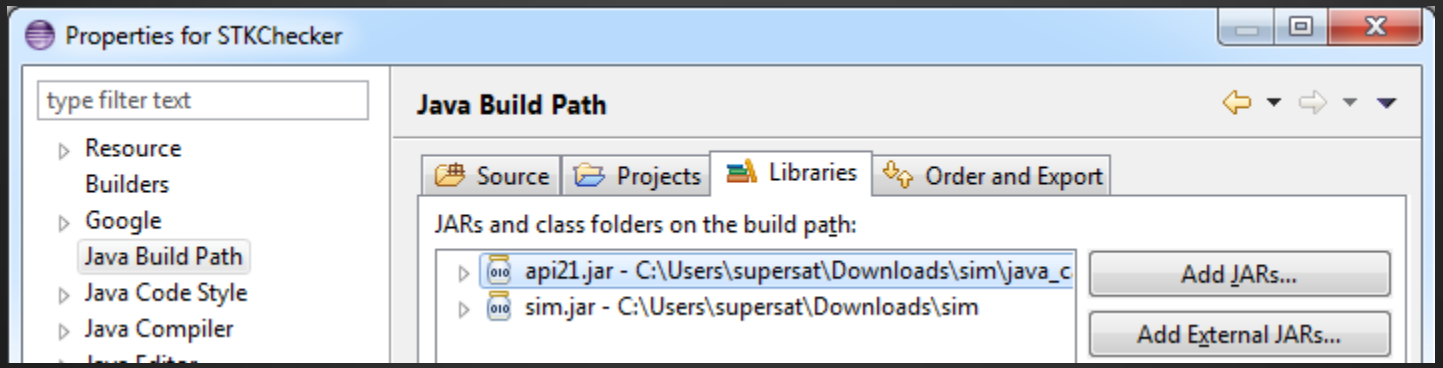
- There are specialized IDEs for this, but you can do without.
- Download the Java Card Development Kit from Oracle (it's free)
- If you're using Eclipse, remove the JRE system library and add the Java Card library

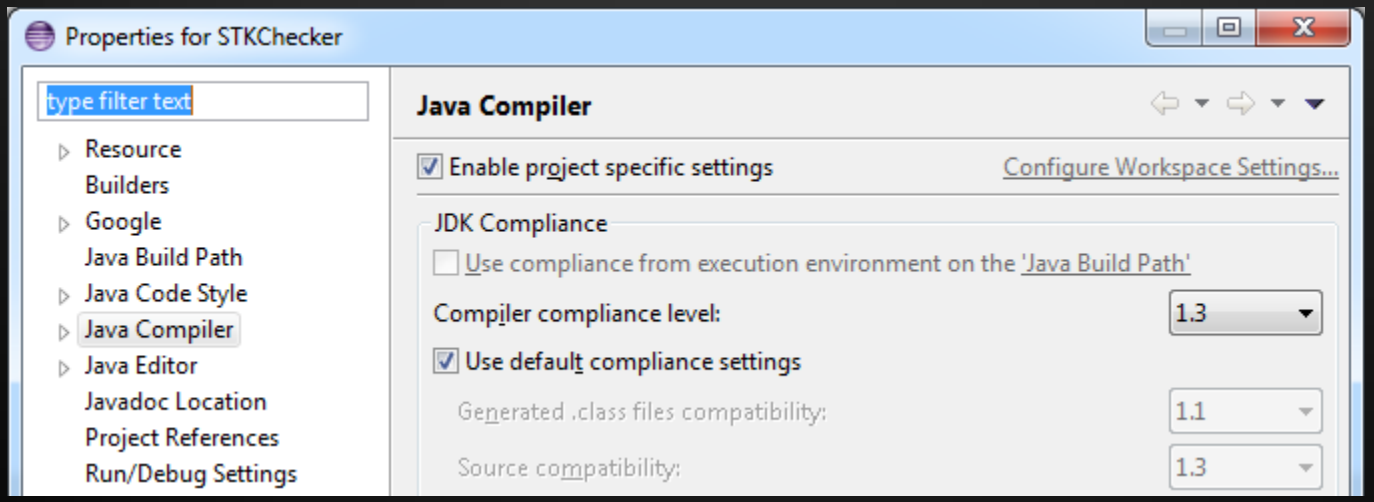
Building Java Card Apps

- You must target Java 1.1 bytecode! 1.3 source code compatibility is okay.
- After you have your .class files, you need to convert them to Java Card bytecode
  - Use the converter tool in the SDK
  - Need to specify application ID (more on this in a minute), API export directory, etc.
- Now you have a CAP file, which is a ZIP archive of CAP components, which define your app

Building Java Card Apps

- Eclipse settings:





Building Java Card Apps

- `$ javac -cp ../sim-tools/javacard/lib/api21.jar -target 1.1 -source 1.3 com/degdeg/HelloCard/HelloApplet.java`

- `java -jar ../sim-tools/javacard/bin/converter.jar -exportpath ../sim-tools/javacard/api21_export_files -applet 0xde:0xfc:0x09:0x20:0x13:0x01 com.degdeg.HelloCard.HelloApplet com.degdeg.HelloCard 0xde:0xfc:0x09:0x20:0x13 1.0`

- We also have Makefiles for your convenience!
  - http://simhacks.github.io

Loading Java Card Apps

- Cards support multiple applications
- Applications are selected by their AID
  - These are officially allocated, but you can make one up
- There is an app on the card that is the *card manager* – use it to load your app
- The card manager is defined by the GlobalPlatform spec

Loading Java Card Apps

- At this point, you might be able to use a tool like GPShell to load your app
  - No such luck on our SIM cards
- Time to dig in to the GlobalPlatform specs!

GlobalPlatform

- All apps are loaded and authorized by the *Issuer Security Domain* – in practice this means that you can't load apps onto a card you didn't issue yourself :(
- On pure GlobalPlatform cards, the ISD is the default app on pre-personalized cards
  - Accessing it on our SIM cards is a lot harder

GlobalPlatform

- Installing an app is a two-step process:
  - Load the binary
  - Instantiate the app
- Loading an app first requires authorization through the INSTALL for load command
- The individual CAP components are concatenated together and sent in blocks with LOAD

GlobalPlatform

- INSTALL for load:

```
80 E6 02 00 12 07 F0 F1 F2 F3 F4 F5
01 00 00 06 EF 04 C6 02 05 00 00
```

- LOAD:

```
80 E8 00 00 6C C4 81 F7 01 00 11 DE
CA FF ED 01 02 04 00 01 07 F0 F1 F2
F3 F4 F5 01 ...
```

- LOAD: `80 E8 00 01 6C ...`
- LOAD: `80 E8 80 02 20 ...`

GlobalPlatform

- To instantiate an app, issue the INSTALL for install command
- There are THREE AIDs involved:
  - Application AID – associated with the load file
  - Module AID – associated with the main class
  - Instance AID – used to select a particular instance

GlobalPlatform

- INSTALL for install and make selectable:

```
80 E6 0C 00 36 07 F0 F1 F2 F3 F4 F5
01 08 F0 F1 F2 F3 F4 F5 01 01 08 F0
F1 F2 F3 F4 F5 01 01 01 00 18 EF 14
C8 02 05 00 C7 02 00 00 CA 0A 01 00
FF 00 10 01 00 00 00 00 C9 00 00
```

GlobalPlatform

- At install time, you can specify:
  - Initialization data
  - App parameters
    - STK uses these extensively
      - Privileges
      - Priority
      - Number of menu items
      - Max menu item size

GlobalPlatform

- Other fun commands:
  - List AIDs, including both modules and instances
  - Delete AIDs
    - You MUST delete instances before deleting the executable!
    - You MUST delete old AIDs before reusing them!
- The spec is freely available

Dealing with #$&!ing SIM cards

- The SIMalliance has a free tool called the **SIM Alliance Loader** that can be used to program SIMs
  - It didn't work out of the box on our SIM cards
  - It is *clearly* designed for experts
  - *If* you know how to configure it, *and* you use Windows, it will work with our SIM cards

Dealing with #$&!ing SIM cards

- The only way to talk to the SIM's ISD is through the over-the-air update mechanism
  - i.e. SMS packets
  - Can also be used to send arbitrary APDUs!
- We don't have to actually send SMSes, but we need to generate commands to the card with SMS packets

Turtles all the way down (GSM 03.48)

- CAT ENVELOPE (A0 C2)
  - SMS-PP Download (D1)
    - Device Identities
    - SMS-TPDU (GSM 03.40)
      - Header
      - User Data
        - Header
        - Command Packet
          - Header (Security parameters, app selection)
            - Uses a 3 byte TAR ID
              - Holy shit powerpoint supports this much nesting
                - This is the actual limit
          - APDU
- http://adywicaksono.wordpress.com/2008/05/21/understanding-gsm-0348/

Turtles all the way down (GSM 03.48)

- Remember this INSTALL command?

```
80 E6 02 00 12 07 F0 F1 F2 F3 F4 F5
01 00 00 06 EF 04 C6 02 05 00 00

A0 C2 00 00 43 D1 41 82 02 83 81 8B
3B 40 08 81 55 66 77 88 7F F6 00 11
29 12 00 00 04 2A 02 70 00 00 25 0D
00 00 00 00 00 00 00 00 00 00 00 01
00 80 E6 02 00 12 07 F0 F1 F2 F3 F4
F5 01 00 00 06 EF 04 C6 02 05 00 00
```

Turtles all the way down (GSM 03.48)

- In case you missed it, you can use this exact mechanism to remotely send APDUs to a SIM card(!!!)

The simhacks toolset

- Open-source Python script to manage apps on SIM cards (plus other SIM card tools)
- http://simhacks.github.io
- DEMO!

Back to STK Apps!

Now that we've figured out how to build and load apps, let's make some!

Life of an STK app

- App is loaded onto the card
- App registers itself with the SIM Toolkit API
- Phone informs STK of its capabilities
- STK informs the phone about registered apps
- Selection of an app will trigger an event to be delivered to the app
- App can then send UI requests back to phone

Anatomy of an STK app

- An STK app is also a Java Card app
  - install
  - process – not normally used
- processToolkit method:
  - Handles STK events

Example STK Apps

- Hello STK
- Toorcamp 2012 Crypto Challenge App
  - Provided hints to the challenge
    - Set tamper-proof hint flags
    - We could remotely query these flags to find out if someone viewed a hint
- DEMOS!

Future Directions

- STK apps are pretty limited, but there is potential for awesomeness
  - SIM card botnet?
- If phones provide an API to send APDUs to SIM apps, things will get really interesting
  - SSH private keys secured on your SIM?
  - Secure BitCoin transactions?
  - What else?
    - Of course, we need carriers to get on board

Future Directions

- Side-stepping the carriers:
  Android's Secure Element

- Yes, it ALSO supports JavaCard/GlobalPlatform!
  - Retail phones have non-default ISD keys :(
  - Come on Google, give us access!

Tools and more!

Learn more and get the tools at:

http://simhacks.github.io/

References

- Java Card 2.1.1 Virtual Machine Specification
- GlobalPlatform card specification 2.1/2.2
- GSM 03.48 – Secure remote SIM access
- GSM 03.40 – SMS standard
- ETSI TS 101 220 – Assigned numbers
- ETSI TS 102 221 – UICC/(U)SIM spec
- ETSI TS 102 223 – Card Application Toolkit
- ETSI TS 102 226 – Remote APDUs
- ETSI TS 102 241 – UICC/SIM API for JavaCard

References

- http://adywicaksono.wordpress.com/2008/05/21/understanding-gsm-0348/
- http://wiki.thc.org/gsm/simtoolkit
- http://randomoracle.wordpress.com/2013/01/28/using-the-secure-element-on-android-devices-33/