

SUBVERTING BYTE CODE LINKER SERVICE TO CHARACTERIZE JAVA CARD API

Samiya Hamadouche, **Guillaume Bouffard**, Jean-Louis Lanet,
Bruno Dorsemaine, Bastien Nouhant, Alexandre Magloire, Arnaud Reygnaud

guillaume.bouffard@xlim.fr
bruno.dorsemaine@etu.unilim.fr

SAR-SSI 2012



Université
de Limoges



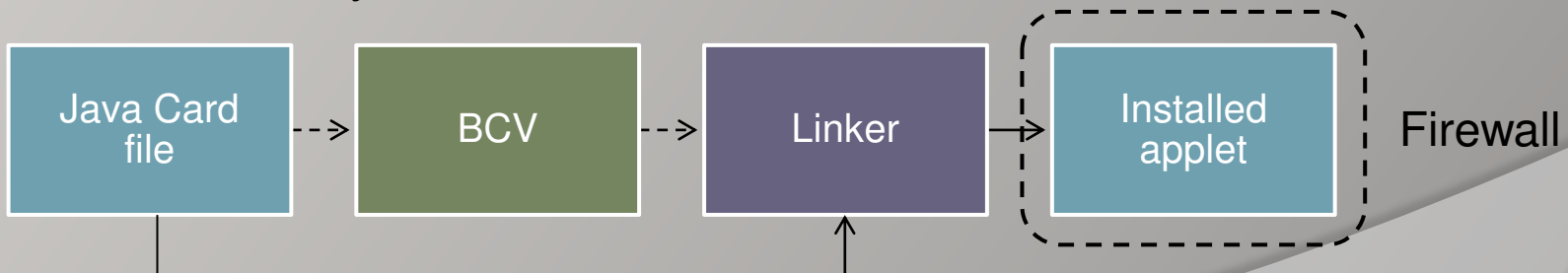
Introduction

Java Card security model

Off-card security model



On-card security model



Introduction

Our objectives

- ⦿ Understand the security of Java Card better
- ⦿ Improve it

Process

- ⦿ Create ill typed files
- ⦿ Load files on the card

Summary

Introduction

Overview

Dr4ccarD & the results

Counter measures

Conclusion

Overview

Goals

- ⦿ Execute arbitrary & rich shell-codes

Problem

- ⦿ The addresses of the methods are not access free

Process

How ?

- ⦿ Modifying the CAP file

What ?

- ⦿ Method Component
- ⦿ Constant Pool Component
- ⦿ Reference Location Component

When ?

- ⦿ Linking step

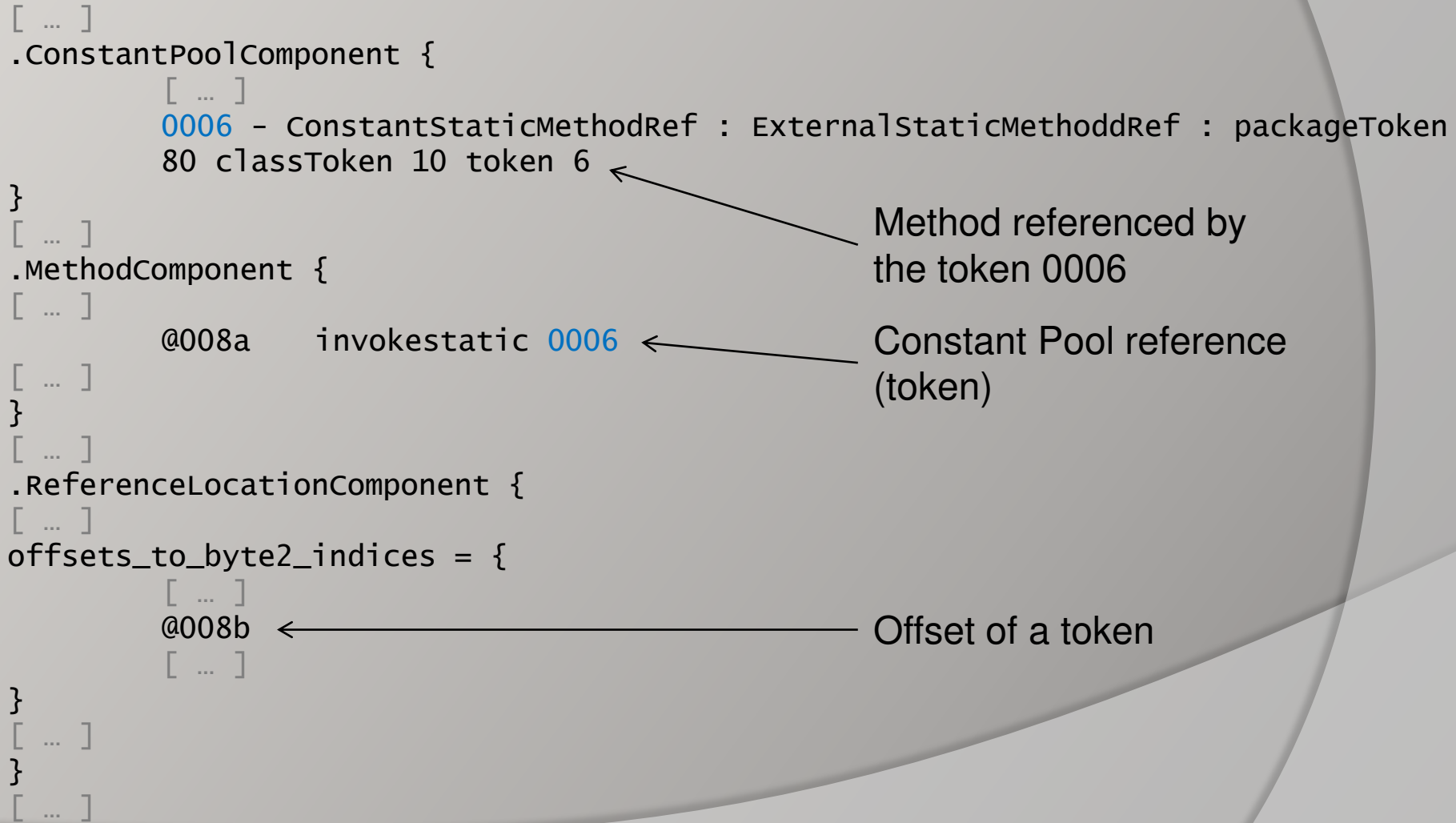
Normal linking step : before

```
[ ... ]
.ConstantPoolComponent {
    [ ... ]
    0006 - ConstantStaticMethodRef : ExternalStaticMethodRef : packageToken
    80 classToken 10 token 6
}
[ ... ]
.MethodComponent {
[ ... ]
    @008a    invokestatic 0006
[ ... ]
}
[ ... ]
.ReferenceLocationComponent {
[ ... ]
offsets_to_byte2_indices = {
    [ ... ]
    @008b ←
    [ ... ]
}
[ ... ]
}
[ ... ]
```

Method referenced by the token 0006

Constant Pool reference (token)

Offset of a token



Normal linking step : after

```
[ ... ]
.ConstantPoolComponent {
    [ ... ]
    0006 - ConstantStaticMethodRef : ExternalStaticMethodRef : packageToken
    80 classToken 10 token 6
}
[ ... ]
.MethodComponent {
[ ... ]
    #8553    invokestatic 0539 ←———— Real address to call the method
[ ... ]
}
[ ... ]
.ReferenceLocationComponent {
[ ... ]
offsets_to_byte2_indices = {
    [ ... ]
    @008b
    [ ... ]
}
[ ... ]
}
[ ... ]
```


The attack

Original code

```
[ ... ]  
@008a  invokestatic 0006  
@008d  bspush 2a  
@008f  sreturn  
[ ... ]
```

Call to the referenced method

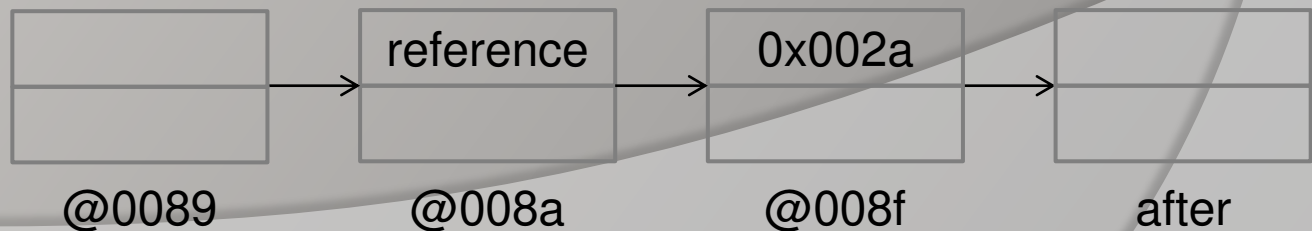
Token

Push the byte 0x2a as a signed short on the stack

Return the top of the stack

Output

0x002a



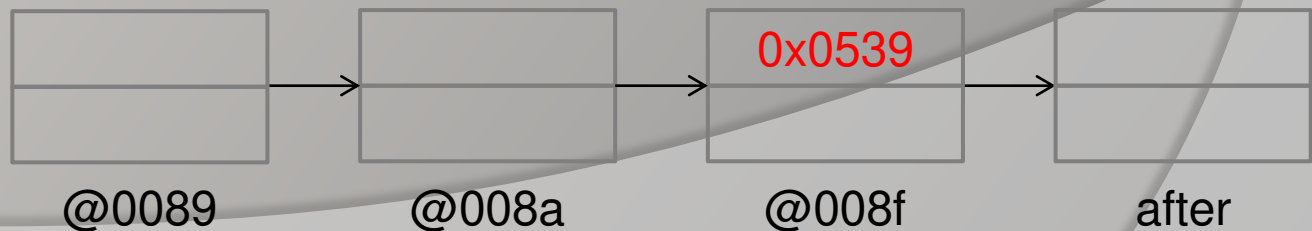
The attack

Modified code

```
[ ... ]  
@008a  sspush 0006 ← Push the token on the stack  
@008d  nop  
@008e  nop  
@008f  sreturn  
[ ... ]
```

Output

0x0539



Summary

Introduction

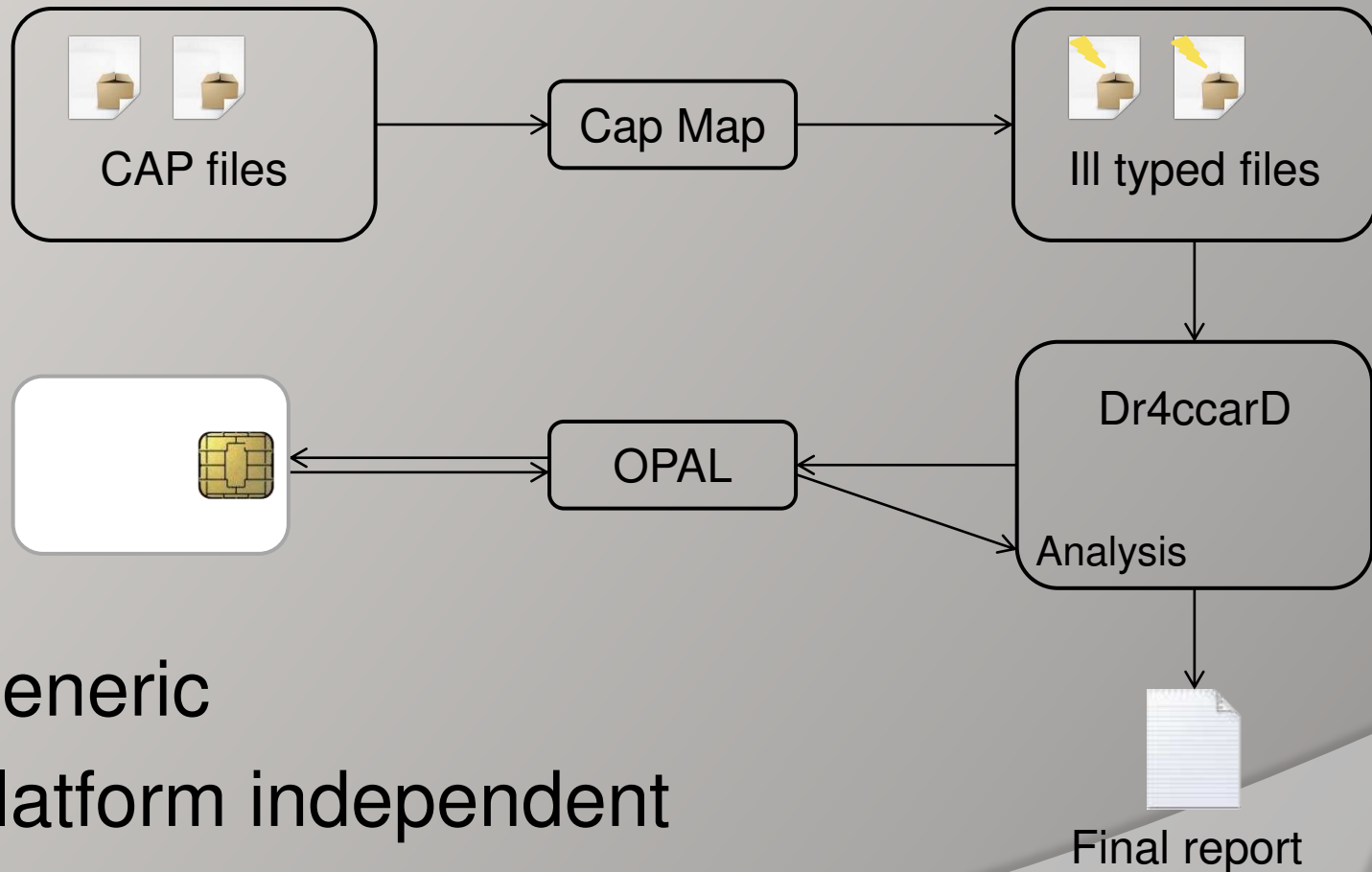
Overview

Dr4ccarD & the results

Counter measures

Conclusion

Dr4ccarD



- Generic
- Platform independent
- API version (in)dependent

The results

Reference	Java Card	GP	Characteristics	Address of <i>getKey</i>
a-21a	2.1.1.	2.0.1.		0x8C08
a-22a	2.2.	2.1.	64k EEPROM	0x080A
a-22c	2.1.1.	2.1.1.	36k EEPROM, RSA	0x020F
b-21a	2.1.1.	2.1.2.	16k EEPROM, RSA	0x3267
c-22a	2.1.1.	2.0.1.	RSA	0x810B
c-22c	2.2.	2.1.1.	72k EEPROM, dual interface, RSA	0x810B
d-21a	2.1.	2.0.1.	32K EEPROM, RSA	0x0003
d-22b	2.1.1.	2.1.1.	16k EEPROM	0x80BA
e-21a	2.2.	2.1.	72k EEPROM	0x142F

Summary

Introduction

Overview

Dr4ccarD & the results

Counter measures

Conclusion

Counter measures

Use an embedded BCV

- ⦿ $O(n * 43 + p)$
- ⦿ n : number of instructions
- ⦿ p : number of tokens

Counter measures

Only link real tokens

- ⦿ $O(p * \log(\log(43)))$
- ⦿ p : number of tokens

```
.ReferenceLocationComponent {
```

```
[ ... ]
```

@008b

```
[ ... ]
```

```
}
```

```
@008a invokestatic 0006
```

Belong to {new, invokestatic, invokevirtual, ...} ?

Summary

Introduction

Overview

Dr4ccarD & the results

Counter measures

Conclusion

Conclusion

- ⦿ Map of the Java Card API
- ⦿ Reverse engineering is easier
- ⦿ Affordable counter measure
- ⦿ Ongoing work : Use a laser beam to bypass an embedded BCV

Thank you for your attention

Do you have any question ?

guillaume.bouffard@xlim.fr
bruno.dorsemaine@etu.unilim.fr
<http://secinfo.msi.unilim.fr/>



Université
de Limoges

