효율적인 메모리 분석을 위한 자바 카드 스택뷰어 설계 및 구현

하지헌°, 조증보, 정민수
경남대학교 컴퓨터공학과

A Design and Implementation of Visual Stack viewer based on JavaCard Technology for efficient analysis of Memory

JiHeon Ha°, JeungBo Cho, MinSoo Jung
Dept. of Computer Engineering, Kyungnam Univ.

요 약

자바 카드 기술이 나온지 약 8년 정도가 되었다. 그간 많은 업체들과 연구원들에 의해 현재까지 자바 카드 기술이 많이 향상되어 왔다. 본 논문에서 제안하는 개발 도구는 계속해서 발전해 가는 자바 카드 기술개발에 도움이 되고자 자바 카드 내부 스택에 들어있는 바이트 코드를 비주얼하게 접근 가능하도록 해준다. 이러한 이점이 Off-Card영역에서 CAP 파일을 직접 이진 코드로 분석하는 것 보다 조금 더 빠르게 분석이 될 수 있게 하기 때문에 자바 카드 프로그램 개발에 있어서 개발자들에게 많은 도움이 될 것이다.

1. 서론

오픈 소스 자바 카드 기술은 1996년에 처음 소개된 뒤 여러 업체들의 노력으로 스마트 카드 상의 프로그램 상호 호환성을 향상시켜 오고 있고, 현재의 자바 카드 기술의 주체라고 볼 수 있는 SUN사는 자바 카드를 위해 스마트 카드 팀인 인터그래티 아트(integrity Arts)를 젬플러스로부터 인수할 정도로 자바 카드 기술에 열정을 가지고 있다.

이러한 스마트 카드는 통신, 신분 확인, 가까이에는 교통카드 및 전자화폐 등의 여러 응용 서비스에서 널리 사용되고 있으며 점진적으로 그 용도는 확대되어 다양한 장비와 함께 네트워크와 연결 및 사용되는 추세로 이어지고 있다. 물론 빈번한 사용과 사용자의 보안인식 결핍 그리고 오픈 소스라는 점으로 많은 보안 문제가 드러나고 있지만 여기에 발 맞추어 스마트 카드의 하드웨어도 발전하고 있다.

Java Card 플랫폼을 내장한 스마트 카드는 애플릿 형태로 구현되어 있는데 이 같은 애플릿은 CAP (Converted Applet Program)파일 형태로 변환되어 CAD(Card Acceptance Device)를 통해 APDU(Application Protocol Data Unit)에 실어서 카드 측으로 전송된다. 이것을 받은 카드에서는 인터프리터에서 해석을 하여 스택을 사용하여 그 애플릿이 하려고 하는 역할을 하게 된다. 본 논문은 앞에서 사용한 스택 내부를 비주얼하게 보아서 개발 및 연구

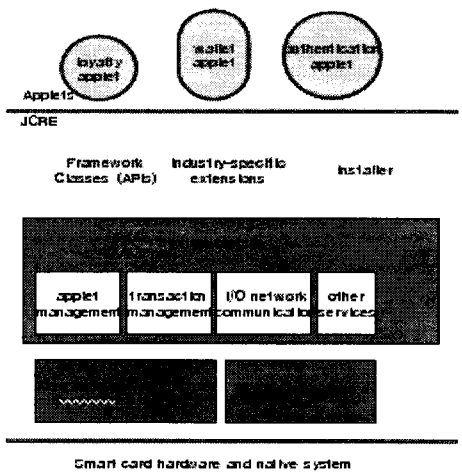
에 도움이 되게 하는 것이 주 목적이다.

이 논문의 구성은 다음과 같다. 2장에는 이 논문에 관련되는 자바 카드 및 관련 파일 연구 부분을 소개할 것이고, 3장에서는 해당 프로그램의 설계와 구현을 기술할 것이다. 마지막으로 4장에서는 결론과 본 논문에서 제안하는 개발 도구의 활용 방안에 대하여 기술할 것이다.

2. 관련연구

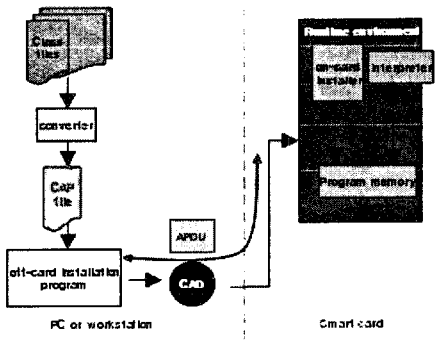
2.1 자바카드

서론에서 알아본 바와 같이 다양한 일을 하는 스마트 카드는 [그림 2-1]과 같이 하위 운영체제(OS) 위에 자바 카드 가상 머신(java card virtual machine)이 위치하며 그 위에 자바 카드 프레임 워크와 산업별 에드온(Add on)클래스 등이 차례로 위치한다. 실제 구현은 애플릿 형태로 이루어진다.



[그림 2-1]

구현된 애플릿은 JCVM의 Off-Card영역과 On-Card영역을 거쳐 실행되는데 자원의 제약을 받지 않는 Off-Card영역에서 클래스 적재, 바이트코드 검증, 클래스 링킹(linking)과 해석이 이루어져 CAP파일로 변환된다. CAP파일들은 [그림 2-2]와 같이 CAD를 통해 On-Card영역으로 전달된다. 이렇게 전달된 C AP파일들은 인터프리터를 통해 메모리 스택에 적재되어 실행된다.



[그림 2-2]

2.1.2 CAP, SCR

[그림 2-2]를 보면 CAD에서 APDU로 통신하는 것을 알 수 있다. 컨버트(convert)된 파일인 CAP파일을 가지고 통신하는데 이 파일을 On-Card 영역으로 보내기 위해서는 CAP파일을 SCR(Script)파일로 변형해서 사용한다.

CAP 파일은 자바 컴파일러에 의해 생성된 클래스 파일을 컨버터(converter)에 의해 자바카드에서 사용할 수 있는 형태로 변형한 파일이다. 게다가 스마트카드에서 메모리 제한 및 로드(load)시의 안정성을 고려하여 자바 플랫폼에서 post-issuance를 위해 사용하는 파일의 형식으로 패키지 단위의 실행가능한 바이너리 표현을 가지고 있으며 jar constant pool을 통한 resolution 경로를 소유한다.

SCR 파일은 CAP파일을 APDU통신을 위해 프로토콜(protocol) 포멧에 맞게 변형시킨 것이다.

2.2 인터프리터

자바 카드 인터프리터는 자바 언어 모델에 대한 런타임(JCRE) 실행을 지원한다. 즉, 인터프리터는 실제 카드위에서 실행되는 JCVM(Java Card Virtual Machine)의 한부분이 된다.

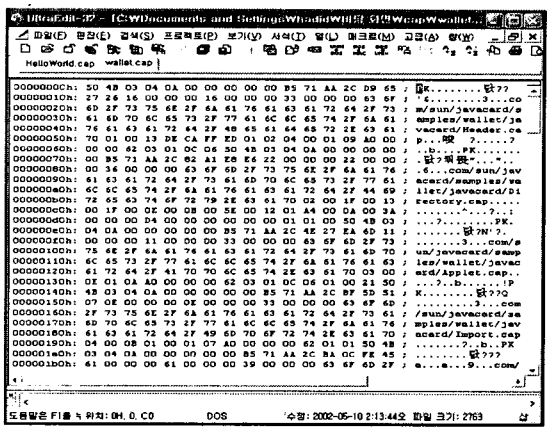
인터프리터는 다음과 같은 작업을 수행한다.

- Bytecode Instruction을 수행함으로써 궁극적으로 applet을 실행한다.
- 메모리 할당을 관리하고 객체를 생성한다.
- 런타임시 보안에 대한 중요한 역할을 한다.

3. 설계 및 구현

3.1 이진 코드로 구성된 CAP파일

CAP 파일 내부를 보면 [그림 3-1]과 같다.



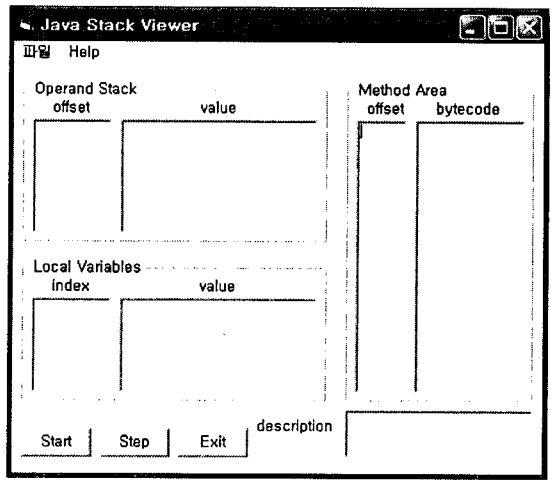
[그림 3-1]

위의 그림은 wallet.cap 파일인데 이진 코드로 되어있어 이것을 보고 바로 분석하기란 쉽지 않다. 우선 살펴보면 01 1b 91 3b가 있다. 여기서 3b가 istore_0을 가르킨다는 것을 2진 코드에 대한 지식이 없으면 해석하는데 많은 노력이 필요하다.

위와 같은 방식으로 CAP 파일을 분석하는 것은 상당한 시간이 걸린다. 그래서 분석하는데 시간을 줄이고 코드 내용을 이해하는데 도움이 되고자 본 논문에서 개발한 방법 즉, 가상 시뮬레이션으로 내용을 분석할 수 있게 시도해 본 것이다.

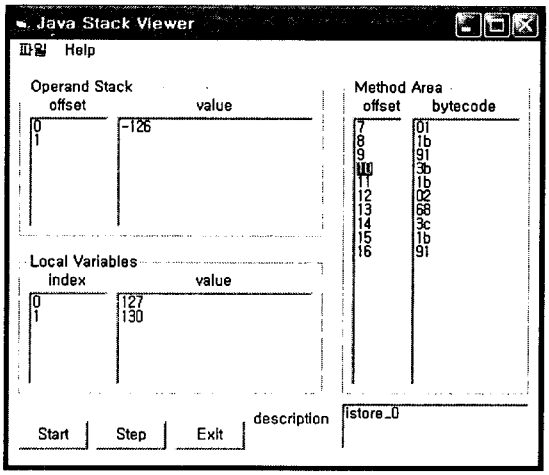
3.2 Java Stack Viewer

아래의 [그림 3-2]를 보자. 프로그램의 구성이 크게 Operand Stack, Method Area, Local Variables, description 네 부분으로 나누어져 있다. Operand Stack에는 연산을 위한 정보가 들어가 있고, Method Area에는 Byte Code를 보여준다. 마지막으로 description에는 Byte Code에 대한 설명을 나타낸다. Local Variables은 해당하는 메소드에 사용되는 변수의 값이 나타나게 된다.



[그림 3-2]

분석한 결과는 [그림 3-3]과 같이 나타난다. [그림 3-3]같은 경우는 Step 버튼을 이용해서 실행했다. Start 버튼을 통해 실행 시킬 경우에는 해당 프로그램의 모든 내용이 각각의 스택 및 변수란에 출력이 된다.



[그림 3-3].

분석 결과를 설명하면 [그림 3-3]의 Method Area를 보면 10번째 byte code가 3b인데 description에는 istore_0를 나타내고 있다. 즉 integer 값을 store한 결과의 값이 operand Stack에 적재되는 것이다. 이러한 연산에 사용된 변수들의 값이 Local Variables에 나타

나있는 것이다.

모든 구현 내용은 각각의 버튼에 구현이 되어있으며 다른 텍스트 컴포넌트는 단지 스택의 내용을 보여 주는 역할만 한다

메뉴의 역할은 분석할 CAP파일을 일어오는 것과 분석된 내용을 저장하는 역할, 프로그램에 대한 간단한 설명을 하고 있다.

4. 결론 및 활용

본 논문에서 구현한 자바 카드 스택 분석기는 애플릿을 실행하는 인터프리터의 내부 스택을 시각적으로 표현함으로써 해당하는 프로그램에 대한 전체적인 이해를 도울 수 있고, 사용자가 작성한 CAP 파일뿐만 아니라 타인이 작성한 CAP파일 또한 바이트 코드로 디버그 할 수 있다.

현재 본 시스템은 Windows 2000 환경에서 Visual Basic을 이용하여 구현하였으나 향후 java로 구현하여 OS와 독립적으로 사용이 가능해 진다면 자바 카드 프로그램을 체계적으로 분석하여 개발하는데 많은 도움이 될 수 있을 것으로 예상된다.

지금은 자바 카드에만 국한된 프로그램이기는 하나 차후 다양한 자바기술(K-Java, P-Java, ..etc)에 접목시키고자 한다.

[참고문헌]

[1] Zhiqun Chen, *Java Card Technology for Smart Cards*, Addison-Wesley, 2000

[2] Sun Microsystems, Inc., *The Java Card 2.2 Virtual Machine Specification*, SUN, 2004

[3] Sun Microsystems, Inc., *The Java Card 2.2 Runtime Environment(JCRE) Specification*, SUN, 2004

[4] Sun Microsystems,Inc., *cJDK_Users_Guides*, SUN, 2004

[5] Venners, *Inside the Java Virtual Machine*, McGraw-Hill, 1997

[6] Uwe Hansmann, Martin S. Nicklous, Thomas Schack, Frank Seliger, *Smart Card Application Development Using Java, Springer*, 2002

[7] T. Lindholm and F.Yellin, *The Java Virtual Machine Specification* ADD-WESLEY, 1997

[8] E. Anuff, *Java Sourcebook*, Wiley, 1996

[9] Gupta M, Choi J-D, Hind M., "Optimizing Java programs in the presence of exceptions", *In Proceedings of the European Conference on Object_Oriented Programming, Cannes, France, 2000.(Lecture Notes in Computer Science,vol.1850) Bertino E(ed)*. Springer Verlag, pp. 422~446,2000

[10] A. Taivalsaari, *Implementation a Java Virtual Machine in the java programming Lunguage*, SUN Lab, 1997

[11] 황욱철, 양윤심, 권오형, 최원호, 김도우, 정민수, *스마트 카드기반의 자바카드 가상기계 최적화 연구*, 한국멀티미디어학회 추계학술발표논문집, 374~377, 2001

[12] http://java.sun.com/, Sun Microsystem,Java Home Page

[13] http://java.sun.com/products/javacard/, SunMicro syste m , Java Card Home Page