자바 클래스 보호를 위한 워터마크 변경방지

조 익[0] 이수현
창원대학교 컴퓨터 공학과
chowing@pl.changwon.ac.kr

Tamper-Resistance of Watermark for Java Class Protection

Ik Cho[0] Su-Hyun Lee
Dept. of Computer Engineering, Changwon National University

요 약

자바는 플랫폼 독립적이라는 장점을 가지고 있지만, 쉽게 역-컴파일 하여 소스코드를 얻을 수 있기 때문에 악의 있는 사용자가 개발자의 모듈과 알고리즘, 데이터 구조를 쉽게 얻을 수 있고 불법 사용이 가능하다. 본 논문에서는 자바 클래스를 보호하기 위하여 소프트웨어에 저작권을 삽입하는 워터마킹 기술과 소스코드 변경을 방지하는 변경확인 기술 및 역-컴파일을 어렵게 하는 난독 기술을 적용한 클래스 단위의 자바 클래스 보호 시스템을 제안 한다. 제안 시스템은 워터마크에 대한 왜곡 공격에 강하고, 워터마크에 변경확인 기술을 적용하여 워터마크의 변경을 방지하였다.

1. 서 론

자바의 중요한 특징 중 하나는 플랫폼 독립성이다. 이것은 자바 프로그램이나 애플릿은 자바 가상 기계 위에서 동작하는데, 이 자바 가상 기계가 구현된 어떤 플랫폼에서도 동일하게 동작하기 때문이다. 자바의 이러한 특성은 프로그램의 배포를 쉽게 만들고 여러 프로그램들이 네트워크를 통해서 쉽게 연동될 수 있게 한다. 그러나 자바 애플릿이나 응용프로그램은 인터넷에서 쉽게 얻을 수 있고, 바이트코드로 번역된 코드에는 자바 소스코드의 정보가 그대로 포함되어 있기 때문에 쉽게 소스코드를 얻을 수 있다. 이런 이유로 악의 있는 사용자는 소스코드를 변경하거나 불법 사용이 가능하다.

이에 대한 해결 방법으로는 하드웨어와 연동하는 방법과 소프트웨어적인 방법이 있다. 코드 난독(obfuscate) 기술과 워터마킹 기술, 소스코드 변경확인(tamper-proofing) 기술, 암호화를 이용한 방법은 후자에 속한다[1]. 특히, 이러한 기술들은 소프트웨어의 변경방지(tamper-resistance)를 위해 사용될 수 있고 이와 같은 기술들을 같이 사용했을 때 더욱 효과가 커진다.

본 논문에서는 자바 프로그램에 대한 워터마킹 기술의 적용과 삽입된 워터마크에 변경확인 기술을 적용시켜 워터마크의 변경을 방지 하고, 코드 분석을 어렵게 하는 난독 기술을 적용한 자바 클래스 보호 시스템을 제안한다. 제안 시스템은 워터마크가 변경되었을 때 프로그램이 실행되지 않으므로 악의 있는 사용자로부터 워터마크 변경을 막을 수 있다.

2. 관련연구

2.1 변경방지

변경방지(*tamper-resistance*)란 장치나 소프트웨어에 저장되어 있는 정보를 함부로 읽거나 수정하여 변경시키는 것을 어렵게 하는 것이다[2]. 변경방지 기술은 전자상거래나 컨텐츠 관련 산업 등과 같이 불법 사용자로부터 보호되어야 할 소프트웨어가 사용되는 다양한 응용 분야에서 요구되는 기술이다.

소프트웨어에 대한 위협적인 것은 악의 있는 사용자에 의한 것일 수도 있고, 바이러스에 의할 수도 있다. 이러한 것으로부터 소프트웨어를 보호하는 방법으로는 변경확인, 코드 난독, 암호화, 정보 은닉 등이 있다. 변경방지의 최종적인 목적은 역공학(reverse engineering)으로부터 소프트웨어를 보호하는 것이다.

2.2 변경확인

변경확인(tamper-proofing)은 소프트웨어에 저장되어 있는 정보가 임의로 변경되었는지를 확인하여 변경을 방지 하는 것이다.

Collberg와 Thomborson는 java reflection 메커니즘을 이용하여 자바에 대한 예를 보였다[1]. 변경확인은 객체 생성과 메소드 호출을 이용하여 데이터 간의 의존성을 이용할 수 있고, 암호화를 이용하거나, 자기 검사를 통하여 중간 결과를 검사하여 정보의 변경 여부를 확인할 수도 있다[3].

2.3 코드 난독

실제로 역-컴파일을 완벽하게 막는 방법이 없으므로 코드 난독(obfuscate)이 현실적으로 사용될 수 있다. 코드 난독처리는 프로그램의 역-컴파일과 소스 분석을 어렵게 만드는 기술이다. 코드를 난독처리 하는 방법에는 이름을 변경하거나, 프로그램의 자료구조를 변경하거나, 제어구조를 변경하는 방법 등이 있다[4].

2.4 워터마킹

워터마킹은 컨텐츠에 저작권 정보를 저장하는 것이다. 소프트웨어에 워터마크를 삽입하는 가장 간단한 방법은 초기화 되는 데이터 부분에 저작권을 삽입하는 것이다. 기존의 방법은 정적 워터마킹으로 정적인 코드에 저작권 등을 삽입하였다. Collberg 와 Thomborson은 실행 상태(힙 영역)에 워터마크를 삽입하는 동적 워터마킹 기술을 제안하였다[1]. 자바 클래스에 정보를 삽입하는 간단한 방법으로는 바이트코드로 변환된 클래스 파일의 상수 풀(constant pool) 이나 메소드와 메소드 사이, 그리고 파일의 끝 부분에 정보를 삽입하는 것이다. 그러나 이 방법은 역-컴파일에 의해 쉽게 제거된다.

2.5 암호화

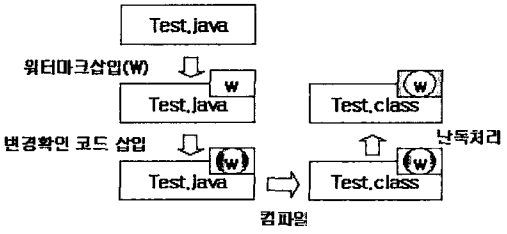
암호화를 이용하는 방법은 실행코드를 암호화하여 비밀키를 이용하여 프로그램을 실행 가능하게 만드는 것이다. 소프트웨어에 암호화 된 특정 함수를 두어 사용자가 이 함수를 사용하기를 원하면 소프트웨어 제공자가 함수를 복호화 하여 결과값을 사용자에게 주는 방법도 있다[5].

3. 제안 시스템

3.1 제안 시스템의 구조

제안 시스템은 자바 소스 파일에 워터마크를 삽입하고 삽입된 워터마크에 메시지요약을 적용하여 변경되는 것을 방지하였다. 그리고 역-컴파일 했을 때 소스 분석을 어렵게 하기 위해서 코드를 난독처리 하였다.

본 논문에서는 워터마크의 정보를 힙 영역에 저장하는 동적 워터마킹을 사용하였다. 제안 시스템의 전체적인 흐름은 (그림 1)과 같다.



(그림 1) 제안시스템 흐름도

3.2 워터마크 삽입과 추출

워터마크의 삽입은 워터마크를 숫자로 변경하여 변경된 숫자를 다시 여러 개의 숫자로 분리시켜 자바 파일에 포함시켰다. 워터마크는 벡터에 포함되어 힙 영역에 저장된다.

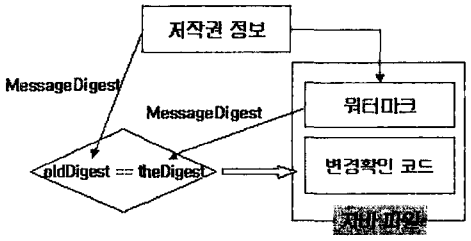
추출은 힙 영역을 조사하여 워터마크를 추출한다. 힙 영역의 분석은 J2SDK의 -Xrunhprof 옵션을 이용하여 조사하였다. 워터마크는 힙 덤프(heap-dump) 파일을 분석하여 추출한다[6].

3.3 워터마크에 대한 변경확인

악의 있는 사용자는 클래스 파일을 분석하여 삽입되는 워터마크를 변경할 수 있다. 본 시스템에서는 워터마크의 변경을 방지하기 위하여 삽입된 워터마크 값을 검사하여 변경여부를 확인한다.

제안 시스템은 자바 API의 MessageDigest 를 이용하여 워터마크에 대해서 변경확인 기술을 적용하였다. 메시지요약은 일종의 데이터 지문이라고 볼 수 있다. 메시지요약의 목적은 데이터가 바뀌지 않았다는 것을 증명하는 것이다. 이는 원본 메시지와 메시지요약을 비교하여 메시지가 손실되었는지를 확인할 수 있다. 메시지요약은 임의의 크기를 가진 메시지를 일정한 크기로 압축해주며, 단방향 해쉬 함수로 역함수가 존재하지 않아 결과값으로는 입력값을 알 수 없다.

변경확인 코드의 삽입은 저장할 워터마크를 먼저 구하고, 이 워터마크에 MessageDigest를 적용하여 메시지요약을 생성한다. 생성된 메시지요약과 삽입할 워터마크를 비교 하는 코드를 생성한 후 이것을 자바 소스 파일에 삽입한다. 변경확인 코드는 (그림 2)와 같이 생성되어 삽입된다.



(그림 2) 변경확인 코드 삽입

클래스가 로딩될 때 삽입된 워터마크에 메시지요약을 적용하여 미리 삽입되어 있는 메시지요약과 비교하여 워터마크의 변경여부를 확인하는 것이다.

3.4 클래스 파일 난독처리

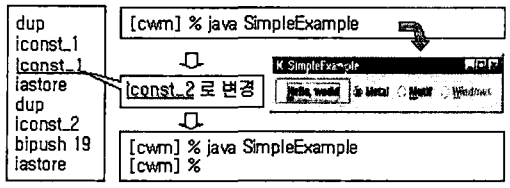
클래스 파일의 유력한 공격은 위치를 발견하여 워터마크를 제거하거나 수정하여 다시 사용하는 것이다. 자바는 역-컴파일이 쉽게 이루어지는 특징 때문에 불법 사용자는

소스코드를 변경할 수 있을 것이다. 난독처리를 적용하면 역-컴파일을 어렵게 하고 역-컴파일된 코드 분석을 어렵게 할 수 있다. 따라서 제안 시스템에서는 제어구조의 변경과 더미(dummy)코드를 삽입하여 코드 분석을 어렵게 하였다. (그림 3)은 포함되는 변경확인 코드와 이 코드에 난독처리를 적용했을 때 나타난 코드이다. 이 코드는 난독처리 전의 코드와 별 차이가 없어 보이나 컴파일이 되면 바이트코드로부터 원래의 소스코드로의 변경은 대단히 어렵다.

MessageDigest md = MessageDigest.getInstance ("MD5"); *md.update(watermark);* *theDigest = md.digest();* *if(!Arrays.equals(theDigest, oldDigest))* *System.exit(0);*	*MessageDigest messagedigest;* *int j;* *messagedigest = MessageDigest.getInstance("MD5");* *j = 0; goto _L1;* *_L3: messagedigest;* *ai; j; JVM INSTR iaload ;* *(byte); update(); j++;* *_L1:* *if(j < ai.length) goto _L3; else goto _L2*

(그림 3) 변경확인 코드와 난독처리 된 코드의 일부

(그림 4)는 악의 있는 사용자가 워터마크를 변경한 후의 실행화면이다. 워터마크가 변경되어지면 프로그램은 실행되지 않는다.



(그림 4) 워터마크 변경 전·후의 실행 화면

4. 고찰

소프트웨어를 모든 공격으로부터 보호하는 것은 힘들다. 공격의 형태는 알려진 버그를 이용하는 방법과 분석도구를 이용하여 코드를 분석하여 변경할 수 있고, 두 프로그램을 비교하여 변경확인 코드 부분을 찾아서 제거하는 방법도 있다. 그리고 제어 구조를 변경하여 코드를 왜곡 할 수 있다. 이러한 공격에 대해서 Wang 등은 인증되지 않는 곳에서 사용자가 프로그램을 분석하여 소프트웨어를 변경하고자 할 때 프로그램의 제어 흐름과 데이터 흐름을 복잡하게 하여 정적 분석을 어렵게 하는 방법을 제안하였고[7], Horen 등은 코드의 여러 곳에 검사기를 삽입하여 실행 중에 자기 검사를 통하여 코드 변경을 방해하는 방법을 제안하였다[3]. Cloakware 시스템은 수학적인 방법으로 코드의 복잡성을 증가시켜 분석을 어렵게 하였다[8].

제안 시스템은 메시지요약을 이용하여 변경확인을 하기 때문에 분석에 의하거나, 코드를 왜곡했을 경우에 삽입된 워터마크 자체가 변경되면 프로그램은 실행되지 않는다. 그리고 소스코드에 난독처리를 하였기 때문에 정적인 분석은 어렵다.

5. 결론 및 향후 과제

최근 불법 소프트웨어의 사용이 증가하고 있고, 이에 따라 저작권 문제와 소프트웨어 보호 문제가 대두되고 있다. 특히, 자바는 쉽게 역-컴파일이 가능하기 때문에 불법 사용이 가능하다. 제안 시스템은 동적 워터마킹 기술을 적용하였고, 이에 대한 변경확인 코드를 삽입하였으며 난독처리를 하여 소스코드 분석과 재-컴파일을 어렵게 하였다.

향후 연구 방향은 워터마크에 대한 것 뿐만 아니라 소프트웨어 전체에 대한 변경방지 기술을 연구하여 소프트웨어를 효과적으로 보호해야 할 것이다.

참고문헌

[1] C. Collberg and C. Thomborson, "Software watermarking: Model and dynamic embeddings," The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages(POPL' 99), 1999.

[2] M. Mambo, T. Murayama and E. Okamoto, "A Tentative Approach to Constructing Tamper-resistant Software," New Security Paradigms Workshop Langdale, ACM, 1997.

[3] B. Horne, L. Matheson, C. Sheehan and R. E. Tarjan, "Dynamic Self-Checking Techniques for Improved Tamper Resistance," Workshop on Security and Privacy in Digital Rights Management 2001.

[4] C. Collberg and C. Thomborson, "Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection," Technical Report, 2000-03, 2000.

[5] T. Sander, C. F. Tschudin and D. Aucsmith, "On Software Protection via Function Hiding," Proceeding of the second workshop on Information Hiding, vol 1525 of Lecture Notes in Computer Sicence, pp. 111-123, 1998.

[6] 조익, 이수현, "자바 클래스 보호를 위한 동적 워터마킹," 한국정보처리학회 추계학술발표논문집, 제8권 제2호, pp. 915-918, 2001.

[7] C. Wang, J. Hill, J. Knight and J. Davidson, "Software Tamper Resistance: Obstructing Static Analysis of Programs," CS Technical Report, CS-2000-12, Department of Computer Science, 2000.

[8] Cloakware Corporation, "An Introduction to the Mathematics of Hiding Data in Software," 2001 (http://www.cloakware.com).