제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

소프트웨어 Tampering Attack 방지 기술에 관한 고찰

이기정*, 조인석*, 권태경*, 황성운**, 남도원**
*세종대학교 소프트웨어공학과
**한국전자통신연구원 컴퓨터소프트웨어연구소
e-mail:{leekijung, livehard}korea.com tkwon@sejong.ac.kr
{sohwang, dwnam}etri.re.kr

A Study on Tamper-resisitent Software Development

Kijung Lee*, Inseok Cho*, Taekyoung Kwon*,
Seongwoon Hwang**, Dowon Nam**
*Dept of Software Engineering, Sejong University
**Dept of Computer & Software Research Laboratory,
Electronics and Telecommunications Research Institute

요 약

인터넷 사용자가 지속적으로 증가함에 따라 인터넷에서의 다양한 디지털 컨텐츠의 유통이 점차 확산되고 있다. 이러한 디지털 컨텐츠를 보호하기 위한 기술이 DRM(Digital Rights Management)이며, 본 논문에서는 DRM의 요소 기술에 해당하는 소프트웨어 보안 기술에 대한 기법을 기술한다.

1. 서론

디지털 컨텐츠 유통 환경에서는 컨텐츠 자체 보호와 이에 대한 저작권 보호를 위한 기술이 요구된다. 디지털 컨텐츠에 대한 인가된 사용을 관리하며, 그 사용으로 인하여 컨텐츠의 유효기간 동안에 발생하는 문제를 처리하기 위한 일련의 하드웨어-소프트웨어 서비스와 기술을 일컬어 DRM(Digital Rights Management)[1]이라고 한다. 국내·외에서는 이와 같은 DRM 기술의 중요성이 점점 크게 인식되고 있으며, 특히 미국의 InterTrust, Adobe, Microsoft 등의 회사에서 DRM 관련 기술을 주도하고 있고 향후 발전 전망 또한 밝은 편이다. 디지털 컨텐츠 유통 및 플레이 환경에서는 디지털 컨텐츠 제공자가 컨텐츠의 저작권을 보호하기 위해 DRM-enable된 상태로 컨텐츠를 유통시키면 사용자는 라이선스를 구입하여, 일반 플레이어에 DRM 기능이 합해진 형태의 DRM-enable 컨텐츠 플레이어를 통해서 사용한다. 이와 같은 환경에서는 근본적으로 사용자에 대한 신뢰를 할 수 없으며, 특히 소프트웨어 환경일 경우 reverse engineering, software privacy infringement , tampering attack 등의 소프트웨어 공격에 직접적으로 노출된다. 이에 대한 구체적인 대응 기술은 obfuscation, software watermarking, tamper-proofing 등이 있으며, 미국의 Cloakware, InterTrust사와 Christian Collberg 등이 기술 개발을 주도하고 있고, 특히 자바 obfuscation 기술에 대해서는 Coding Art, Lee Software, Retrologic, Zelix등 많은 회사들이 주도하고 있으며 향후 발전 전망이 밝은 편이다. 이에 본 논문에서는 2장 관련연구에서 DRM 소프트웨어에 대한 공격 형태 및 그에 대한 대응 방안인 tamper-proofing 기술의 obfuscation에 대한 내용[2]을 기술 하였으며 3장 에서는 결론 및 향후 전망에 대해서 논하겠다.

2. 관련연구

디지털 컨텐츠 유통 및 플레이 환경에서는 DRM

보안 모델을 적용해야 한다. DRM 보안 모델이란 클라이언트 관점에서 고려된 일반적인 보안 모델과 달리 소프트웨어 또는 디지털 컨텐츠 저작권자의 관점에서 고려된 보안 모델로서 신뢰할 수 없는 클라이언트 시스템에 배포된 소프트웨어 또는 디지털 컨텐츠 저작권을 보호하는 것을 뜻한다. 이러한 DRM 보안 모델이 적용되는 가장 큰 이유는 디지털 컨텐츠 유통 및 플레이 환경에서 DRM-enable된 디지털 컨텐츠를 사용자 환경에 배포할 경우 이미 사용자 환경에 설치된 DRM-enable된 소프트웨어 에이전트가 이에 대한 확인 및 플레이 (실행)를 처리하도록 하며, 결국 소프트웨어 에이전트의 안전성에 크게 의존하기 때문이다. 하지만 이와 같이 사용자 환경에 설치된 소프트웨어는 사용자의 제어 아래 놓이게 되므로, DRM 보안 모델에서는 reverse engineering 및 tampering attack과 같은 소프트웨어 공격에 대한 안전성 보장이 필요하다.

여기서 reverse engineering은 오브젝트 코드를 소스 코드로 변환하여 프로그램의 기능과 해법을 추출하는 공격을 의미하며, tampering attack은 소프트웨어의 내부에 감추어진 정보를 부정으로 불러내거나 기록을 바꾸도록 하는 공격을 의미한다. 이와 같은 공격을 통해서 공격자는 컨텐츠 플레이에 필요한 키를 추출할 수도 있으며, 디지털 컨테이너나 DB를 위조하거나 DRM 관련 중요 컨트롤 로직을 우회할 수 있다. 특히 이 때 컨텐츠 원본의 불법적 유출이나 저작권에 대한 위조가 가능하다.

이러한 공격은 공격 시점에 따라서 정적 분석 (static analysis)과 동적 분석 (dynamic analysis)으로 구분할 수 있다. 정적 분석은 실행중이지 않은 코드 상태에서 이루어지며, 이를 위한 도구로는 역컴파일러 (decompiler), 역어셈블러 (deassembler), 데이터 흐름 분석기 (data flow analyzer) 등이 있다. 동적 분석은 소프트웨어의 실행 상태에서 이루어지며, 이를 위한 도구로는 에뮬레이터 (emulator), 디버거 (debugger), 메모리 덤프 (memory dump), 가상 기계 (virtual machine) 등이 있다.

소프트웨어의 reverse engineering 및 tamper proofing에 대한 안전성에 관한 연구 결과는 University of Auckland의 Christian Collberg에 의해서 비교적 상세히 발표되었으며, Cloakware사와 InterTrust사 등에서 관련 기술 연구 및 상용화가 진행되었다. 특히 JAVA 언어를 중심으로 obfuscator가 많이 연구되었으며, CodingArt의 CodeShield, Lee Software의 Smokescreen, Retrologic의 Retro Guard, Zelix의 Klass Master 등의 자바 obfuscator들은 이미 control transformation과 data transformation을 중심으로 하는 2세대 제품으로 제품군이 바뀌고 있는 추세이다.

이와 같은 기술을 소프트웨어 보호 방법에 따라서 분류하면 다음과 같이 정리할 수 있다.

- 소프트웨어에 대한 불법적인 접근을 직접적으로 차단하는 방법: 예를 들면, 스마트 카드와 같은 tamper-resistant device나 암호 프로세서, 복호 키 위탁, 복호 키 위장, 다양한 종류의 암호 위장, cryptographic wrapping, debugger inhibit 등의 기술들이 이와 같은 방법을 사용한다.
- 소프트웨어를 동일한 기능을 갖지만 reverse engineering이 매우 어려운 구조를 갖는 소프트웨어로 변환 (transformation)하는 방법 : 예를 들면, 다양한 종류의 obfuscation과 tamper resistant software 등이 이와 같은 방법을 사용한다.
- 소프트웨어가 수정될 경우 더 이상 정상적인 기능을 수행하지 않도록 하는 방법 : 예를 들면, tamper-proofing 기술을 가진 프로그래머에 의한 hand coding과 Cloakware의 TRS[3] 등이 이와 같은 방법을 사용한다.

DRM 요소 기술인 소프트웨어 보안 기술은 단순히 암호화만으로 해결할 수 없는데, 그 가장 큰 이유는 암호화에 따른 비용보다는 복호화에 필요한 키 관리의 어려움에서 주로 기인한다. 즉, reverse engineering에 대한 대책이 없이는 복호화에 필요한 키를 소프트웨어에 포함할 수 없으며, 결국 부가적인 방법을 필요로 하게 된다. 따라서 reverse engineering에 대한 대책 마련은 복호화 키에 대한 안전성을 제공하며, 이후 tampering attack에 대한 안전성, ease-of-use 등의 요구 사항을 만족하기 위한 초석이라고 할 수 있다. 이러한 reverse engineering 기술을 막기 위한 기술들은 다음과 같이 크게 3종류로 분류할 수 있으며 각 기술은 물론 소프트웨어 에이전트를 포함한다.

- 하드웨어 기반 기술 (hardware-based)
  - Tamper-resistant device (예, 스마트 카드)
  - 암호 프로세서 (cryptographic processor)
- 네트워크 기반 기술 (network-based)

- 복호 키 위탁
- 복호 키 분산
- 암호 위장 (cryptographic camouflaging)

● 소프트웨어 독립적인 기술 (software-only)

- Obfuscation
- Cryptographic wrapping
- Tamper resistant software (Cloakware TRS)
- Debugger inhibit
- Hand coding

여기서 하드웨어 기반 기술이나 네트워크 기반 기술은 소프트웨어적인 대책 이외의 부가적인 기술이 요구되는 것에 해당하며, 소프트웨어 독립적인 기술은 소프트웨어만으로 해결하는 것을 의미하며 이러한 기술은 Trusted software 기술이라고도 한다. 특히, obfuscator의 경우 JAVA 언어 쪽에서 개발이 활발히 이루어졌으며 그 이유는 JAVA 언어의 경우 바이트 코드에 대한 역 컴파일이 중요한 취약점으로 작용하였기 때문이다. 다음은 이러한 Trusted software 기술의 세부 기술사항을 나열 하였다.[4]

● Layout transformations: lexical transformations 이라고도 하며 Java에서 주로 사용된 obfuscation 기술이 레이아웃 변환 기술이며 프로그램 내부의 변수의 이름이나 함수의 이름과 같은 identifier에 대한 변환을 통해서 reverse engineering이 어렵도록 만드는 기술을 일컫는다. Java와 같이 객체 코드인 바이트 코드 단계에서 변환을 수행하거나, 또는 소스 코드의 컴파일 단계에서 변환을 수행할 수 있다. 어휘 변환은 obfuscation을 위해서 반드시 활용되어야 하는 기술이다.

● Control transformations: 프로그램 제어 흐름의 변환을 통한 obfuscation 기술이다. 제어 흐름의 중간 중간에 opaque predicate이라고 하는 control transformation 코드를 삽입하여, 주로 정적 분석이 매우 어렵도록 만든다. 다양한 변환 코드를 구성할 수 있으며, obfuscation을 위해서 반드시 활용되어야 하는 기술이다. 예를 들면 다음과 같은 변환을 고려할 수 있다.

```
int main() {                int main() {
  S1 ;                        S1 ;
  S2 ;          ⇨             if (7y^2 - 1 ≠ x^2)^T S2 ;
  S3 ;                        S3 ;
}                           }
```

예제에서 변환된 코드의 opaque predicate인 $(7y^2-1\neq x^2)^T$가 항상 TRUE (T)이므로 if 조건문과 상관없이 S2는 항상 수행된다. 하지만 정수론 지식이 없이 판단하기 어렵도록 만든 간단한 예제이다. 이와 같이 control transformation을 위한 안전한 opaque predicate의 선택은 매우 중요하다.

● Data transformations: 데이터 구조 및 기본적 데이터 유형에 대한 변환이다. 매우 어려운 변환 방법이며 또한 많은 비용을 요구한다. 하지만 다른 종류의 변환 방법과 혼용될 경우 유용할 수 있다. 주요 방법으로는 변수 분할 (variable splitting), 정적 데이터 변환 (static data conversion), 스칼라 변수 결합 (scalar variable merge) 등이 있다.

● Abstraction transformations: 데이터 추상화에 대한 변환이며, 객체지향 언어에서 고려되어야하는 방법이다. 여기에는 상속 관계 수정 및 배열 재구성과 같은 방법이 있다.

이 밖에 predictive transformations 기법은 이미 알려진 reverse engineering에 대한 특정 변환 기법을 개발하는 것을 의미한다. Cloakware사에서는 이와 같은 obfuscation 기술을 자동화하고 randomization 등의 기능을 추가하여, TRS라는 상품을 발표하였다. 그 구성도는 그림 1과 같다. 이와 같이 소프트웨어를 소스 단계에서 자동적인 obfuscation을 통하여 reverse engineering이나 tampering attack에 강인한 소프트웨어로 변환한다.

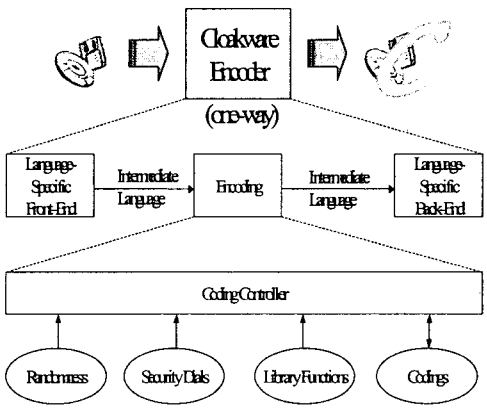

그림 1 TRS 구성도

3. 결론 및 향후 전망

DRM 기반하의 안전한 디지털 컨텐츠 유통을 위해서는 사용자 환경에서의 디지털 컨텐츠 및 플레이 환경 보호가 반드시 이루어져야 하며 다음과 같은 요구 사항을 갖는다.

- Reverse engineering에 대한 안전성 (비밀성 보장)
- Tampering attack에 대한 안전성 (무결성 보장)
- Portability (다양한 플랫폼 지원, 현재 설치된 플랫폼에도 쉽게 적용)
- Ease-of-use (다운로드 후에는 오프라인 사용 지원)

구체적으로 앞의 두 가지는 안전성 요구 사항에 해당하며, 뒤의 두 가지는 기능성 요구 사항에 해당한다고 할 수 있다. 현존하는 기능들도 위의 요구 사항을 모두 만족하기는 쉽지 않으며, DRM 기반하의 안전한 디지털 컨텐츠 유통 시스템에서 수용할 수 있는 수준의 기술 제시가 이루어져야 한다.

또한 향후 도래될 디지털 컨텐츠 유통 환경은 오프라인과 연계된 온라인 환경으로 예상되는바, 온라인 비즈니스 환경에서는 거래하는 상대방에 대한 인증 및 거래에 대한 무결성/부인 봉쇄 기능이 필수적으로 필요하며, DRM 기반의 보호 기술과 연계하여 개발을 진행하고 있는 중이며, 특히 Tampering Attack을 막을 수 있는 강인한 암호 라이브러리를 개발 하고 있다.

본 연구의 목적은 디지털 컨텐츠 유통에 있어서 안전한 플랫폼을 제공할 수 있도록 하는 핵심기술을 개발하는 것이며, 이 연구에서 추구하는 결과물은 기본적으로 오프라인 형태의 안전성을 제공하지만, 네트워크 기반 안전성 강화나 하드웨어 기반 안전성 강화를 염두에 둔 Tampering Attack 방지 기술을 개발하는 것이다.

참고문헌

[1] Christian Collberg and Clark Thomborson. Software watermarking: Models and dynamic embeddings. In Principles of Programming Languages 1999, POPL'99, San Antonio, TX, January 1999
http://www.cs.auckland.ac.nz/~collberg/Research/Publications/CollbergThomborson99a/index.html

[2] Christian Collberg, Clark Thomborson "Water marking, Tamper-Proofing, and Obfuscation-Tools for Software Protection"

[3] "Introduction to Cloakware Tamper-Resistant Software(TRS) Technology
http://www.cloakware.com

[4] Christian Collberg, Clark Thomborson Douglas Low "A Taxonomy of Obfuscation Transformations"