난독화 서버를 이용한 소프트웨어 보호방식

이현록*, 노한영*, 박재범**, 김광조*

*한국정보통신대학교 암호와 정보보안 연구실
** SK 텔레콤 Service 기술연구원 Application 개발팀

A Software Protection Method using Obfuscation Server

Hyunrok Lee*, HanYoung, Noh*, JaeBum Park**, Kwangjo Kim*

*Cryptology & Information Security Lab, Information and Communications University.
**SK Telecom Service Development Team 1, Service Platform & Core Network R&D Center

요 약

최근 개인 소유의 디지털 정보기기의 수가 많아짐에 따라 필요한 소프트웨어의 수요도 많아지고 있다. 하지만, 소프트웨어의 사용에 있어 불법행위라는 것을 인지하면서도 불법복제가 근절되고 있지 않다. 이러한 소프트웨어 불법복제 문제를 도덕에 호소하거나 법으로 강제하는 것에는 분명 한계가 있으며, 기술적으로 불법복제를 어렵게 하여 사전에 해당 문제를 방지하도록 하는 연구가 반드시 필요하다. 소프트웨어를 보호하기 위해 일련번호의 사용, 코드 난독화 기법의 도입, 추가적인 하드웨어를 사용한 보호기법 등 많은 연구가 이루어져 왔으나 악의적인 사용자를 막기에는 역부족인 실정이다. 본 논문에서는 기존의 코드 난독화 기법들의 문제점을 개선한 난독화 서버 기반의 소프트웨어 보호방식을 제안하고, 기존 방식들과의 비교분석을 통해 제안 방식이 좀 더 효율적이고 안전한 소프트웨어 보호방식임을 보인다.

I. 서론

최근 개인 소유의 디지털 정보기기의 수가 많아짐에 따라 필요한 소프트웨어의 수요도 많아지고 있다. 하지만, 소프트웨어의 사용에 있어 불법행위라는 것을 인지하면서도 불법복제가 근절되고 있지 않아 소프트웨어 저작권자의 피해가 심각해지고 있는 실정이다. 시장조사기관인 IDC 조사에 따르면 2005년 한국의 소프트웨어 불법 복제율은 46%에 달한다고 밝히고 있으며, 국가적인 차원에서 이를 근절하기위해 2007년 4월 컴퓨터프로그램 보호법 개정안이 시행되면서, 불법 소프트웨어 사용 처벌 기준이 기존 '3년이하 징역 또는 5000만원 이하 벌금'에서 '5년 이하 징역 또는 5000만원 이하 벌금'으로 강화됐다. 또한, 지속적인 단속과 교육을 통해 불법 복제 문제를 해결하려고 하지만 이는 기존의 아날로그 매체와 다르게 매우 쉽게 복제 가능하고 증거 확보나 추적이 어렵다는 소프트웨어의 특징상 한계가 있다. 따라서 기술적으로 불법복제를 어렵게 하여 사전에 해당 문제를 방지하도록 하는 연구가 반드시 필요하다.

소프트웨어를 보호하기 위해 일련번호의 사용, 코드 난독화(Code Obfuscation) 기법의 도입, 소프트웨어의 구동 및 복사 방지를 위한 추가적인 하드웨어인 동글(Dongle) 사용 등을 적용하고 있는 추세이다. 하지만 사용자에게 배포된 소프트웨어는 무한의 시간과 무한의 자원을 사용할 수 있는 공격환경에서 역공학(Reverse Engineering)을 수행할 수 있으며, 이렇게 한번이라도 소프트웨어 공격이 성공하면 쉽게 복제하여 재사용할 수 있다는 소프트웨어의 특징으로 인해 불법 복제의 피해를 줄이기에는 역부족이다. 역공학을 근본적으로 방지하는 것은 어렵지만 코드 난독화 등을 통해 이에 드는 비용을 크게 증가시켜 역공학을 어렵게 만들 수는 있으며, 이에 대한 연구들이 지속적으로 수행되고 있다.

본 논문에서는 기존의 소프트웨어 보호 방식들로는 해결할 수 없었던 알고리즘의 보호, 사용자 인증 문제, 성능 저하 문제 등을 해결할 수 있는 새로운 난독화 서버 기반의 소프트웨어 보호 방식을 제안하였으며, 제안 방식은 사용자의 하드웨어 고유번호를 기반으로 난독화 서버에서 생성한 난

독화된 실행코드의 중요 컴포넌트를 제공하여 사용자 프로그램에서 재결합하게하고, 해당 컴포넌트에 바이너리 난독화 방식을 적용하여 좀 더 효율적이고 안전한 방식이다. 또한 기존의 방식들과의 비교분석을 통해 제안 방식의 우수함을 보인다.

본 논문의 구성은 다음과 같다. II장에서는 대표적인 소프트웨어 보호방식과 기존 연구 결과들을 요약하고, III장에서는 본 논문에서 제안하는 난독화 서버 기반의 소프트웨어 보호 방식을 제안한다. IV장에서는 비교분석을 수행하고, 마지막 V장에서는 본 논문의 결론과 향후 연구 과제를 기술한다.

II. 관련연구

사용자에게 배포된 소프트웨어는 사용자에게 완전히 공개된 상태가 되며, 무한의 시간과 무한의 자원을 사용할 수 있는 공격환경 하에서 공격자는 소프트웨어를 역공학을 이용하여 공격할 수 있는 최고의 권한을 가지게 된다. 따라서 실행코드 자체를 분석할 수 있는 정적분석(Static Analysis), 실행되고 있는 상태를 살펴볼 수 있는 동적분석(Dynamic Analysis) 모두 가능하고, 해당 프로그램을 수정할 수도 있으며 이러한 공격을 방지하기 위한 다양한 방법이 적용되고 있다. 첫째, 일련번호의 입력을 통한 보호가 가장 일반적이다. 이 방식은 일련번호의 노출에 의한 불법 복제가 가능하며, 혹은 역공학을 통해 비교적 쉽게 분석되어 해당 일련번호 검증 모듈을 회피 및 위조 일련번호의 생성 등의 방법을 통해 불법복제 행위를 막기 힘들어 진다. 둘째, 신뢰할 수 있는 추가적인 하드웨어인 동글을 사용할 경우 비교적 안전하긴 하지만, 이 방식 또한 해당 동글을 회피하는 방식이 여전히 존재하며, 사용자에게 추가적인 비용과 불편함을 주게 되는 단점이 존재한다. 셋째, 코드 난독화 기법은 해당 소프트웨어의 역공학 자체를 상대적으로 어렵도록 분석, 공격에 소요되는 시간 및 자원의 비용을 크게 증가시키는 방법으로 난독화 정도와 성능 측면에서의 상호 절충점이 문제로 제기가 되고는 있지만 효과적인 소프트웨어 보호 방식으로 각광받고 있으며, 이에 대한 연구가 활발히 진행되고 있다. 그 외 프로그램을 쉽게 수정하지 못하도록 코드 자체를 감시하는 프로그램 변조방지(Tamper-proof), 임의로 프로그램을 변경했을 때 이를 감지할 수 있는 코드를 삽입하는 방식인 소프트웨어 워터마킹(Watermarking) 기술이 있다.

코드 난독화는 크게 프로그램의 소스 난독화 방식[3, 4, 5]과 바이너리 난독화 방식[6]으로 나눌 수 있다. 첫째, 소스 난독화 방식은 주로 자바(Java)나 닷넷(.NET)과 같은 중간 언어 형태로 바이너리를 표현하는 언어를 사용한 프로그램에서 많이 사용되고 있으며, 소프트웨어 일련번호 입력 부분에 부분적으로 활용되고 있으나 역공학 분석 도구들[7, 8]을 활용하거나 전문적인 공격자에 의해 공격당할 가능성이 높다. 둘째, 바이너리 난독화 방식은 최종 컴파일된 기계어를 직접 수정하여 다양한 역공학 분석도구를 활용하는 것을 더욱 어렵게 하는 방식으로 아직까지 많은 연구가 되어있진 않지만, 악성 코드 제작자들이 백신 프로그램을 우회하기 위해 사용이 점차적으로 늘고 있는 실정이며, 이에 대한 연구가 지속적으로 필요한 기술이다.

Fu 등[1]의 방식은 소스 난독화 방식의 일종으로 기존의 데이터 변환(Data Transformation), 제어 변환(Control Flow Transformation), 문자열 변환(Lexical Transformation)을 조합하여 적용한 방식이 아닌 특정 변수를 패스워드(Password)로 사용하여 패스워드와 임의의 난수 α를 이용해 계산한 Hash(Password+α)의 값으로 미리 만들어진 변수 테이블에서 변수 값을 찾아 해당 변수를 대체하는 방식이다. 하지만, 해당 방식은 변수 테이블이 메모리상에 저장되어 있기 때문에 해당 테이블의 크기가 작을 경우 공격자가 계산을 통해 예측 가능하며, 이를 방지하기 위해 무한정으로 메모리를 많이 잡아 테이블의 크기를 크게 만들 수 없는 공간적인 한계를 가진 방식이다.

NLPK[2]는 Fu 등이 제시한 방식의 단점을 개선 한 방식으로 컴퓨터 하드웨어가 가진 고유한 값인 CPU ID, 하드디스크 ID, 메인보드 ID, NIC의 MAC 주소 등을 조합하여 패스워드 매개 값으로 이용하여 정상적인 경로로 받은 프로그램을 다른 컴퓨터에서 실행시키면 오동작하게 하여 보호하는 방식이다. 또한 NLPK는 난수값과 함께 암호학적인 일방향 해쉬 함수를 적용하여 변수 테이블 값을 예측할 수 없도록 하였다. 하지만, 해당 방식은 사용자 별로 소스코드 전체를 매번 재컴파일해서 배포해야 하는 단점과 소스 난독화 자체가 가지는 공격 가능성이 높은 문제를 여전히 가지고 있다.

III. 제안기법

본 논문에서는 사용자의 하드웨어 고유번호를 기반으로 서버에서 생성한 난독화된 실행코드의 중요 컴포넌트를 제공하여 사용자 프로그램에서 재결합하게하고, 해당 컴포넌트에 바이너리 난독

화 방식을 적용하여 기존 방식들을 개선한 난독화 서버 기반의 소프트웨어 보호 방식을 제안한다. 제안 기법에 사용한 표기법은 표 1과 같다.

표 1 : 표기법

표기법	설 명
u	소프트웨어 사용자
s	소프트웨어 제공자 (난독화 서버)
r_x^i	개체 x가 생성한 난수 $i \ni_R \{0,1\}^k$
$h_k(m)$	키 값 k를 가지는 m의 해쉬값
HID_i	서버에 등록된 i번째 하드웨어 식별자
HID^*	HID들을 임의로 조합한 하드웨어 고유번호
P_u	필수 바이너리 컴포넌트가 제외된 프로그램
$BO(p\|\alpha)$	바이너리 프로그램 p를 매개변수 α를 사용하여 난독화
O_s	서버측에서 난독화된 컴포넌트
$\dotplus$	바이너리 코드 결합 연산자
O	최종 난독화된 소프트웨어 ($O = P_u \dotplus O_s$)

■ **등록 단계**

제안 방식에서 사용자 u와 사용자의 하드웨어 식별자들 HID_i는 소프트웨어를 구매할 때 소프트웨어 제공자의 난독화 서버s에 등록되며, 등록 시 사용자 u의 비밀키 k도 발급된다.

■ **설치 단계**

사용자가 구매한 소프트웨어를 자신의 컴퓨터에 설치할 때 P_u를 설치하게 되고 최초 실행 시에 해당 프로그램에서 사용자 및 디바이스 인증과 서버에서 생성한 난독화 컴포넌트를 전송받게 되는 인증설치 단계를 수행하게 된다.

■ **인증설치 단계**

해당 단계에서는 사용자의 하드웨어 고유번호를 기반으로 서버에서 난독화 모듈 $BO(p)$를 이용하여 O_s를 생성하여 사용자에게 전송하고, 사용자는 이를 바이너리 코드 결합 연산자 $\dotplus$을 이용하여 최종 난독화된 소프트웨어인 $O = P_u \dotplus O_s$를 생성하게 된다. 이때, 난독화 서버는 중요 바이너리 코드 O_s를 생성하여 각 사용자에게 전송하는 역할을 하는 서버로써 O_s는 바이너리 파서를 거쳐 파싱된 정보를 분석하고 바이너리 난독화와 재결합 과정을 거쳐 생성되게 된다. 해당 단계를 상세히 기술하면 아래와 같다.

1) $u \leftarrow s : r_s^i$

2) $u \rightarrow s : r_u^i, h_k(r_u^i, r_s^i, HID^*)$

3) s : 서버에 등록된 HID_i들로 가능한 HID^*의 테이블 작성 후 검증

4-1) 검증 성공 시

$s : O_s = BO(p|HID^*)$

$u \leftarrow s : h_k(r_s^i, r_u^i, u),\ O_s$

$u : O = P_u \dotplus O_s$

인증설치 단계 종료 후 정상사용 □

4-2) 검증 실패 시

인증설치 단계 종료 □

4-1)단계의 검증 성공 시 수행되는 바이너리 난독화의 과정과 사용자가 전체 실행 가능한 최종 프로그램을 받게 되는 과정은 그림 1과 같다.

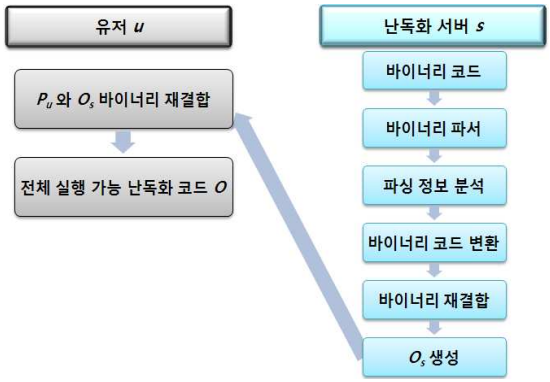

그림 1: 바이너리 난독화 과정 및 최종 SW 생성

또한 $O = P_u \dotplus O_s$에서 P_u와 O_s를 현재 가장 보편적으로 사용되고 있는 윈도우 프로그램의 PE 파일 포맷으로 도식화 하면 그림 2와 같으며 이때 P_u는 SectionHeader1에서 제어하는 Section이고, O_s는 SectionHeader2에서 제어하는 Section으로 $\dotplus$의 연산은 쉽게 이루어 질 수 있음을 보여준다.

IV. 비교분석

본 논문에서 제안한 방식은 기존 NLPK가 사용자 별로 소스코드 전체를 매번 재컴파일해야 한다

는 것과 소스 난독화 자체가 가지는 공격 가능성이 높은 문제점을 해결한 방식을 제안하였으며, 이를 해결하기 위해 사용자의 하드웨어 고유번호를 기반으로 서버에서 생성한 난독화된 실행코드의 중요 컴포넌트를 제공하여 사용자 프로그램에서 재결합하게하고, 해당 컴포넌트에 바이너리 난독화 방식을 적용한 좀 더 효율적이고 안전한 소프트웨어 보호 방식이다. 기존 방식들과의 비교결과는 표 2와 같다.

표 2 : 기존 방식과 비교

	Fu	NLPK	제안방식
불법복제방지	×	○	○
변수보호	△	○	○
알고리즘보호	△	△	○
사용자인증	×	×	○
모듈화기반 성능향상	×	×	○
○: 해당 조건 만족 △: 해당 조건 일부 만족 × : 해당 조건 불만족			

기존 NLPK 방식과 마찬가지로 하드웨어 고유번호를 사용하게 되어 사용자마다 다른 소프트웨어를 가지게 되며, 따라서 BOBE(Break Once, Break Everywhere) 방지도 가능하다. 또한 하드웨어 고유번호를 얻는 과정은 각 프로그램마다 공통적이라서 공격대상이 되기 쉽지만 제안 방식은 사용자 인증과 함께 서버에서 수행되는 바이너리 난독화를 제공하여 공격자가 해당 과정에 대한 공격도 쉽게 할 수 없게 되는 장점을 가진다.

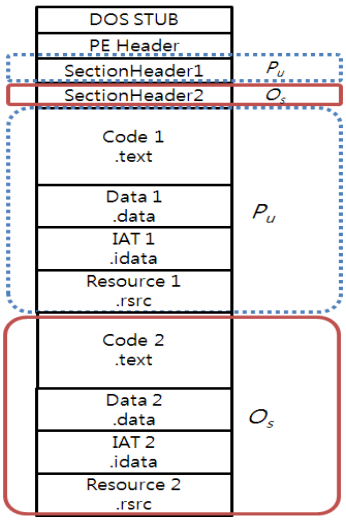

그림 2: PE 파일 포맷

V. 결론

본 논문에서는 악의적인 사용자로부터 법, 제도에 의한 사후 조치 이전에 기술적으로 불법복제를 사전 차단 가능한 방법인 난독화 서버 기반의 소프트웨어 보호방식을 제안하였다. 제안 방식은 사용자의 하드웨어 고유번호를 기반으로 서버에서 생성한 난독화된 실행코드의 중요 컴포넌트를 제공하여 사용자 프로그램에서 재결합하게하고, 해당 컴포넌트에 바이너리 난독화 방식을 적용하였다. 또한 기존의 선행연구들과의 비교분석을 통해 기존 방식의 단점을 개선한 효율적이고 안전한 소프트웨어 보호 방식임을 보였다. 향후 연구 과제로는 제안 방식의 핵심이 되는 바이너리 난독화 기술에 대한 심층적인 연구가 필요하며, 난독화 기술 적용 시 발생하는 시스템 성능저하를 최소로 하고 안전성은 최대로 보장할 수 있는 방식에 대한 연구를 수행할 예정이다.

참고문헌

[1] B. Fu, G. Richard, and Y. Chen, “Some new approaches for preventing software tampering”, In Proc. of the 44th Annual Southeast Regional Conference, Mar. 2006.

[2] 노한영, 이현록, 박재범, 김광조, "하드웨어 고유 번호 기반 소프트웨어 보호 방식", 2007 한국정보보호학회 하계학술발표대회, pp.385-390, 2007. 6. 22.

[3] J. Ge, S. Chaudhuri, and A. Tyagi, "Control flow based obfuscation", In Proc. of the 5th ACM Workshop on Digital Rights Management, Nov. 2005.

[4] C. Collberg, C. Thomborson, "Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection", IEEE Transcations on software engineering, vol. 28, No. 8, Aug. 2002.

[5] C. Collberg, C. Thomborson, and D. Low, "Breaking Abstractions and Unstructuring Data Structures", In Proc. of IEEE Int'l Conf. Computer Languages, May 1998.

[6] G. Wroblewski, "General Method of Program Code Obfuscation", In Proc. of SERP02, pp. 153-159, Las Vegas, USA.

[7] OllyDbg, http://www.ollydbg.de/

[8] DataRescue sa/nv, Liége, Belgium. IDA Pro, http://www.datarescue.com/idabase/