신뢰할 수 없는 DRM 클라이언트 시스템 하에서 키 보호를 위한 Secure Storage Device에 관한 연구

이기정° 박성호 조인석 진광범 권태경
세종대학교 컴퓨터 공학부 소프트웨어 공학과
fumaro@nownuri.net skyreon@hotmail.com livehard@korea.com jkb1234@hotmail.com tkwon@sejong.ac.kr

A Study on the Secure Storage Device for Protecting Cryptographic Keys in Untrusted DRM Client Systems

Kijung Lee° Seong-Ho Park In-Seok Cho Kwangbum Jin Taekyoung Kwon
School of Computer Engineering, Sejong University

요 약

DRM 유통 시스템 하에서 암호화된 컨텐츠의 데이터는 신뢰할 수 없는 사용자 환경에서 복호화된 이후에 사용자에게 컨텐츠에 대한 서비스를 제공하게 된다. 이러한 신뢰할 수 없는 사용자 환경에서 컨텐츠 데이터를 안전하게 관리하기 위해서는 암호화 키나 라이센스 데이터등과 같은 데이터를 사용자 환경에 안전하게 보관해야 하며 이렇게 보관된 데이터는 사용자에게 절대 노출되지 않도록 보호를 해야 한다. 본 연구에서는 이러한 신뢰할 수 없는 사용자 환경에서 암호화 키나 라이센스 데이터를 안전하게 보관할 수 있는 Secure Storage Device를 개발하여 소개한다.

1. 서 론

디지털 컨텐츠 유통 및 플레이 환경에서는 디지털 컨텐츠 제공자가 컨텐츠의 저작권을 보호하기 위해 DRM-enable된 상태로 컨텐츠를 유통시키면 사용자는 라이선스를 구입하여, 일반 플레이어에 DRM[1] 기능이 합해진 형태의 DRM-enable 컨텐츠 플레이어를 통해서 사용하며 이러한 플레이어를 DRM Client 시스템이라 부른다.

DRM Client 시스템은 디지털 컨텐츠 저작권자의 가장 큰 요구 사항인 저작권 보호와 컨텐츠 보호를 위한 핵심 기능을 수행하며, 신뢰할 수 없는 사용자 환경에 다운로드 및 설치되어 향후 배포되는 디지털 컨텐츠들에 대한 확인 및 플레이(실행)를 지속적으로 처리하게 되며, 이러한 처리 과정 중에 DRM Client 시스템은 사용자에 대한 인증이나 암호화된 디지털 컨텐츠 데이터의 복호화 및 디지털 컨텐츠 사용규칙을 명시한 라이센스 데이터 등은 사용자가 접근 못하도록 안전하게 보호되어야 하며, 본 논문은 이러한 인증키, 암호키, 라이센스 데이터를 DRM Client 시스템 하에서 안전하게 보호할 수 있는 Secure Storage Device를 구현하였다.

2. DRM 시스템 개요

DRM 유통 환경 하에서 컨텐츠 저작권자는 컨텐츠를 Secure Container에 패키징 한 후 유통 시키게 되며, DRM Client 시스템은 이러한 Secure Container에서 컨텐츠를 추출하여 라에센스에 정의된 사용규칙에 따라 사용자에게 서비스를 제공한다.

그림 1은 DRM 유통 시스템의 전체적인 구조를 나타낸 것이며, 이 구조 내에서 DRM Client 시스템은 DRM 유통 시스템의 전체 영역 중에서 각종 부적절한 행위에 노출이 가장 많은 부분으로서 프로그램으로서 완벽한 보안성을 제공해야 한다.

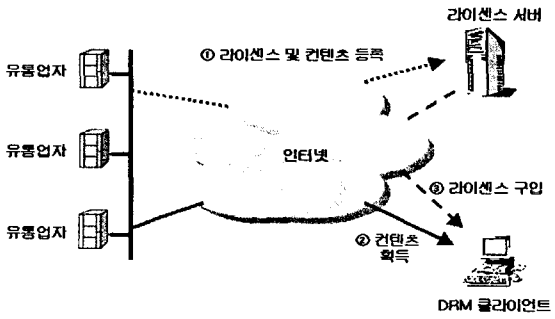

그림 1 : DRM 유통 시스템의 전체 구조

DRM 유통 시스템 내에서 사용자 컴퓨터에 설치되는 DRM 클라이언트 시스템은 유통업자가 Secure Container에 패키징한 디지털 컨텐츠를 주어진 권한 내에서 목적에 맞게 사용자 환경을 제시하는 응용프로그램이며, 그림 2는 DRM 클라이언트 시스템을 구성하는 전체 모

듈 구성을 L-타낸 것이다.

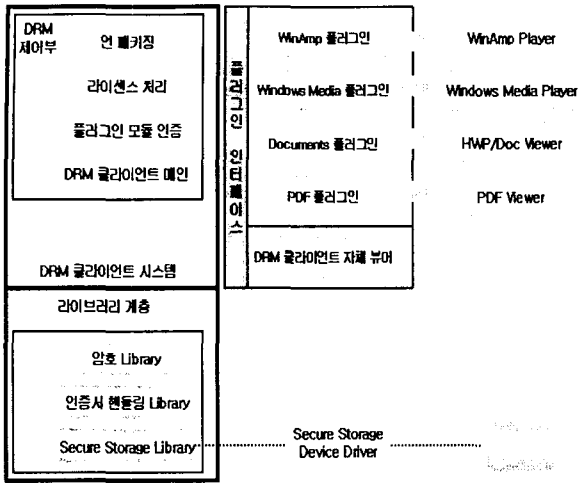

그림 2 : DRM 클라이언트 시스템 모듈 구성

DRM 클라이언트 시스템의 각 모듈들은 암/복호화 기능을 지원함으로써 디지털 컨텐츠에 대한 보안 서비스 강화 및 정당한 절차를 거쳐 라이센스를 구매한 사용자에 대한 인증 기능을 수행한다. 또한, 디지털 컨텐츠의 라이센스에 따라서 복제 방지, 사용 횟수 제한, 사용 기간 제한 등과 같이 다양한 사용 규칙을 적용해서 디지털 컨텐츠의 유료 사용을 위한 과금 관리 기능의 지원 및 영화, 음악, 만화, 문서, 게임등 다양한 디지털 컨텐츠의 특성에 따라서 기존의 플레이어에 플러그인을 추가하거나 특정 컨텐츠만을 재생시킬 수 있는 자체 플레이어를 통해서 사용자에게 이미 전달된 디지털 컨텐츠의 불법 접근을 통한 해킹이나 디지털 컨텐츠 플레이어에서 복호화된 데이터의 복사, 저장, 인쇄, 화면 캡쳐 등을 방지하는 기능을 제공한다.

DRM 클라이언트 시스템에서 디지털 컨텐츠의 보호를 위해 사용되는 인증키, 암호키, 라이센스 데이터 등과 같은 사용자에게 노출되어서는 안될 중요한 데이터는 Secure Storage Device Driver에서 암호화된 이후 하드 디스크에 저장이 된다.

3. Secure Storage Device Driver

3.1 Secure Storage Device Driver 개요

Secure Storage Device Driver가 데이터 암호화를 위해 사용하는 키를 마스터 키라 부르며, 이러한 마스터 키는 윈도우 파일 시스템 및 레지스트리에 저장이 된다. 하지만 이러한 마스터 키를 사용자 모르게 윈도우 파일 시스템 및 레지스트리에 숨긴다는 것은 근본적으로 불가능한 일이다.

Secure Storage Device Driver는 신뢰할 수 없는 DRM 클라이언트 환경에서 사용자가 마스터 키에 접근을 하지 못하도록 하기 위해 파일 시스템 및 레지스트리를 핸들링하는 시스템 서비스 API[2] 함수들에 대한 후킹 기법을 사용해서 마스터 키를 보호하고 있으며, Win9x 계열에서는 파일 시스템 API 및 레지스트리 핸들링 API 함수를 후킹하는 가상장치 드라이버(VxD) 형태로 구현을 하였으며, WinNT 계열에서는 윈도우 커널의 핵심 요소인 NTOSKRNL.EXE에서 제공하는 시스템 서비스 API 함수들 중 파일 및 레지스트리를 핸들링하는 API 함수들을 후킹하는 디바이스 드라이버[3]의 형태로 구현을 하였다. Win9x 계열과 WinNT 계열에서의 후킹 기법은 기본적인 로직은 거의 비슷하므로 본 문서에서는 시스템의 구조가 좀더 복잡한 WinNT 계열의 Secure Storage Device Driver의 구현 기법에 대해 기술한다.

3.2 Secure Storage Device Driver 구현 원리

Windows 2000과 같은 WinNT 계열의 OS는 시스템이 크게 User Mode 영역과 Kernel Mode 영역으로 나누어지며 User Mode 영역에 존재하는 시스템 컴포넌트 중 Win32 Subsystem Dlls 들은 윈도우 어플리케이션들에게 C Runtime Library나 MFC Library와 같은 다양한 종류의 API 함수를 제공한다. Kernel Mode 영역에 존재하는 시스템 컴포넌트로는 Executive 컴포넌트, IO Manager, Memory Manager, Device Driver, HAL(Hardware Abstraction Layer) 등과 같은 다양한 컴포넌트들이 있으며, 특히 Executive 컴포넌트에 속하는 NTOSKRNL.EXE 모듈은 윈도우 커널을 구성하는 핵심 모듈의 하나로서 Windows System 컴포넌트에게 다양한 Native API 서비스를 제공한다.[2][4]

다음 그림은 Windows 2000 System에 대한 전반적인 구성 요소를 나타낸 것이다.

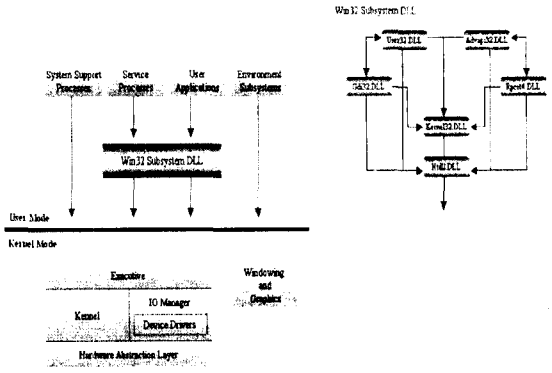

그림 3 : Windows 2000 System Archtecture

Executive 컴포넌트에 해당되는 NTOSKRNL.EXE 모듈은 다양한 시스템 서비스 API 함수를 제공하며, 이들 함수들은 User Mode 영역에 있는 Win32 Subsystem Dlls 모듈들에게 필요한 서비스를 제공하기 위해 사용되기도 한다.

예를 들어 User Mode 영역에 있는 임의의 어플리케

이션에서 파일을 핸들링하기 위해 파일을 열게 될 경우 어플리케이션은 Win32 Subsystem Dlls에서 제공하는 fopen() 함수를 사용하게 되며, 이 파일 오픈 요청에 대한 메시지는 NTOSKRNL.EXE에서 제공하는 ZwCreate File() 함수나 ZwOpenFile() 함수를 호출하게 되는 것이다. 즉, User Mode 영역의 어플리케이션에서 사용하는 대부분의 API 함수에 대한 서비스 요청은 Kernel Mode 영역에 존재하는 NTOSKRNL.EXE 모듈로 전달되어서 User Mode API 함수에 매칭되는 System Service API 함수를 다시 호출하게 되는 것이다.

Secure Storage Device Driver는 NTOSKRNL.EXE 모듈이 제공하는 System Service API 함수를 후킹해서 마스터 키에 대한 접근을 감시하는 것이며, 다음 그림은 Secure Storage Device Driver가 NTOSKRNL.EXE 모듈이 제공하는 System Service API 함수를 후킹하는 과정을 나타낸 것이다.

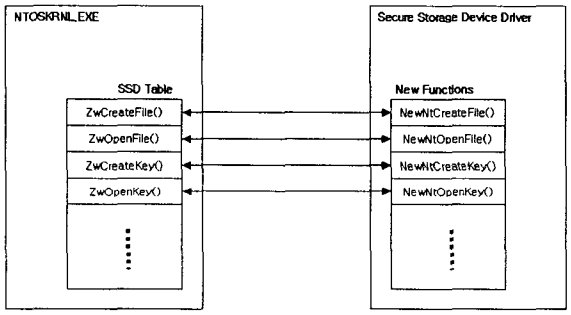

그림 4 : Secure Storage Device Driver의 System Service API 후킹 원리

위 그림에서 NTOSKRNL.EXE 모듈 내부에는 SSD (System Service Descriptor) Table[2] 이라는 어레이 형태의 테이블을 가지고 있으며, 이 테이블 내부에는 NTOSKRNL.EXE 모듈이 제공하는 수많은 System Service API 함수들에 대한 포인터 주소가 저장되어 있다. 여기서 Secure Storage Device Driver는 NewNtCreateFile(), NewNtOpenFile(), NewNtCreateKey (), NewNtOpenKey() 등과 같은 새로운 함수들을 구현한 이후 이들 새로운 함수들에 대한 포인터 주소를 SSD Table 내부의 함수들의 주소와 교체를 시키게 된다.

함수들의 주소가 교체된 이후에는 Secure Storage Device Driver가 제공하는 새로운 시스템 서비스 함수가 호출되며, Secure Storage Device Driver는 이들 새로운 서비스 함수 내부에서 마스터 키에 대한 접근 요청을 검사한 후 마스터 키에 대한 서비스를 거부하게 된다.

Secure Storage Device Driver는 윈도우 시스템이 초기 부팅될 때 함께 실행되고 이후 윈도우 시스템이 종료되면서 함께 종료가 된다. 따라서 마스터 키에 접근하는 시스템 서비스 API 함수들에 대한 후킹은 항상 실행되고 있는 상태이므로 DRM 클라이언트 시스템의 사용자는 마스터 키의 정보를 획득할 수 없게 되며, 만일 사용자가 SCM(Service Control Manager)을 사용해서 Secure Storage Device Driver를 강제로 언로드 시킬 경우에는 윈도우 시스템이 자동으로 다운되도록 구현되어 있다.

Secure Storage Device Driver와 DRM 클라언트 시스템 사이에는 IOCTL 코드를 사용해서 서로 통신을 하게 되는데, 이 경우 동일한 IOCTL 코드를 사용하는 외부의 어플리케이션이 Secure Storage Device Driver에게 서비스를 요청할 경우 문제가 될 수 있다. 이를 해결하기 위해 Secure Storage Device Driver는 서비스를 제공하기 전에 먼저 서비스를 요청한 어플리케이션에 대한 프로세스 인증이 선행되어야 하며, 프로세스 인증을 위해 Secure Storage Device Driver는 현재 실행되고 있는 프로세스의 ID값을 획득한 다음 이 프로세스 ID 값을 사용해서 현재 프로세스의 실행 이미지 파일의 전체 경로명을 획득해서 실행 이미지 파일의 데이터를 읽어 들인 후 이 데이터의 MD5 해쉬 값을 계산해서 기존에 저장되어 있는 실행 이미지의 해쉬 값과 비교를 해서 두 값이 동일할 경우에 서비스를 제공한다.

4. 결 론

DRM Client 시스템은 디지털 컨텐츠 저작권자의 가장 큰 요구 사항인 저작권 보호와 컨텐츠 보호를 위한 핵심 기능을 수행하며, 이러한 기능을 수행하기 위해서는 컨텐츠 보호를 위해 요구되는 암호키나 라이센스 정보와 같은 중요한 데이터는 Secure Storage Device Driver와 같은 안전한 보관소에 저장이 되어야 하며, 이렇게 저장된 데이터는 DRM Client 시스템의 사용자 조차도 접근을 허락해서는 안 된다. 이러한 기능을 제공하기 위해서는 정보 은닉 기법, 모듈 인증 메카니즘, reverse engineering 방지를 위한 obfuscation[5] 기법 등과 다양한 기술이 요구되며, 본 연구에서는 이러한 다양한 기법들에 대한 연구를 진행하였으며, 향후 연구에서는 obfuscation 기법을 Secure Storage Device Driver에 적용하는 방안 및 메모리 덤핑을 통해 마스터 키와 같은 중요한 데이터의 누출을 막기 위한 기법을 개발해야 할 것이다.

5. 참고 문헌

[1] Joshua Duhl, and Susan Kevorkian, "Understanding DRM Systems" An IDC White Paper, 2001

[2] Prasad Dabak, Sandeep Phadke, and Milind Borate, "UNDOCUMENTED WINDOWS NT", M&T BOOKS

[3] Edward N. Dekker and Joseph M. Newcomer, "Developing Windows NT Device Drivers: A Programmer's Handbook" Addison-Wesley 1999

[4] David Solomon, Mark Russionovich, "Inside Microsoft Windows 2000" Microsoft

[5] Christian Collberg, Clark Thomborson Douglas Low "A Taxonomy of Obfuscation Transforma tions"