ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
j.inst.Korean.electr.electron.eng.Vol.26,No.3,380~388,September 2022
논문번호 22-03-07
https://doi.org/10.7471/ikeee.2022.26.3.380
48

동적 DLL 삽입 기술을 이용한 화이트리스트 기반 접근통제 우회공격 대응 방안 연구

A Countermeasure against a Whitelist-based Access Control Bypass Attack Using Dynamic DLL Injection Scheme

김 대 엽*★

Dae-Youb Kim★

Abstract

The traditional malware detection technologies collect known malicious programs and analyze their characteristics. Then such a detection technology makes a blacklist based on the analyzed malicious characteristics and checks programs in the user's system based on the blacklist to determine whether each program is malware. However, such an approach can detect known malicious programs, but responding to unknown or variant malware is challenging. In addition, since such detection technologies generally monitor all programs in the system in real-time, there is a disadvantage that they can degrade the system performance. In order to solve such problems, various methods have been proposed to analyze major behaviors of malicious programs and to respond to them. The main characteristic of ransomware is to access and encrypt the user's file. So, a new approach is to produce the whitelist of programs installed in the user's system and allow the only programs listed on the whitelist to access the user's files. However, although it applies such an approach, attackers can still perform malicious behavior by performing a DLL(Dynamic-Link Library) injection attack on a regular program registered on the whitelist. This paper proposes a method to respond effectively to attacks using DLL injection.

요 약

전통적인 악성코드 탐지 기술은 알려진 악성코드를 수집하고 특성을 분석한 후, 분석된 정보를 블랙리스트로 생성하고, 이를 기반으로 시스템 내의 프로그램들을 검사하여 악성코드 여부를 판별한다. 그러나 이러한 접근 방법은 알려진 악성코드의 탐지에는 효과적일 수 있으나 알려지지 않았거나 기존 악성코드의 변종에 대해서는 효과적으로 대응하기 어렵다. 또한, 시스템 내의 모든 프로그램을 감시하기 때문에 시스템의 성능을 저하시킬 수 있다. 이러한 문제점들을 해결하기 위하여 악성코드의 주요 행위를 분석하고 대응하기 위한 다양한 방안들이 제안되고 있다. 랜섬웨어는 사용자의 파일에 접근하여 암호화한다. 이러한 동작특성을 이용하여 시스템의 사용자 파일에 접근하는 정상적인 프로그램들을 화이트리스트로 관리하고 파일 접근을 제어하는 방안이 제안되었다. 그러나 화이트리스트에 등록된 정상 프로그램에 DLL(Dynamic-Link Library) 삽입 공격을 수행하여 악성 행위를 수행하게 할 수 있다는 문제점이 지적되었다. 본 논문에서는 화이트리스트 기반 접근통제 기술이 이러한 DLL 삽입 공격에 효과적으로 대응할 수 있는 방안을 제안한다.

Key words : Malware, Ransomware, Blacklist, Whitelist, DLL Injection

---

* Dept. of Information Security, Suwon University
★ Corresponding author
E-mail : daeyoub69@suwon.ac.kr, Tel : +82-31-229-8284
※ Acknowledgment
The paper was supported by The research grant of the University of Suwon in 2021.
Manuscript received Aug. 22, 2022; revised Aug. 31, 2022; accepted Sep. 15, 2022.

Ⅰ. 서론

전통적으로 악성코드를 탐지하기 위해 제안된 기술들은 악성 프로그램을 수집한 후, 수집된 악성 프로그램을 코드 수준에서 분석하고, 그 특성을 찾아 정보화 한다. 이렇게 분석된 악성코드 정보를 기반으로 사용자 시스템에 설치되었거나 저장되어 있는 프로그램을 비교분석하여 악성코드를 탐지한다. 이와 같은 기법은 전통적인 접근통제 기술 중 접근 불허 객체 목록인 블랙리스트(Blacklist)를 활용한 방법으로, 이미 알려진 악성코드를 시스템 내에서 탐지하는데 매우 효과적이다. 그러나 블랙리스트 기반의 악성코드 탐지 기술의 경우 알려지지 않은 새로운 악성코드 또는 변종 악성코드를 탐지하기 어렵다는 단점이 있다. 또한, 악성코드를 탐지하기 위해서는 시스템에 저장되어 있는 모든 파일들을 검사해야하기 때문에 시스템 성능 저하의 주된 원인이 될 수 있다[1].

이와 같은 문제점을 해결하기 위한 방안으로 악성코드의 주요 동작 특성에 맞춰 대응하는 다양한 방안이 제안되고 있다. 즉, 프로그램 코드의 특성이 아닌 프로그램이 수행하는 동작을 감시하고, 이를 기반으로 악성코드 여부를 판단하는 방안이다. 대표적으로 머신러닝 기술을 활용하여 알려지지 않은 악성코드를 탐지하려는 시도가 여러 연구에서 진행되고 있다[2][3].

또 다른 방안은 랜섬웨어(Ransomware)와 같이 특정 목적을 달성하기 위하여 악성 행위를 수행하는 프로그램만을 제어하기 위하여, 해당 목적에 따라 수행되는 프로세스만을 대상으로 접근통제를 적용하는 방안이다. 랜섬웨어의 경우, 공격목표에 저장되어 있는 특정 종류의 파일에 접근하여 암호화하는 특성을 갖고 있다. 이러한 랜섬웨어를 통제하기 위하여 특정 파일이나 파일 디렉터리에 대하여 접근통제 기술을 적용시켜 랜섬웨어의 동작을 통제하는 기술이 제안되고 있다[4][5].

공격목표 시스템에 저장된 특정 파일에 접근하는 랜섬웨어의 동작 특성을 고려하여 효율적인 탐지 프로그램을 설계한다면, 다음과 같은 두 가지 사항을 고려해 볼 수 있다.

(1) 시스템 내에 저장되어 있거나 동작하고 있는 모든 프로그램을 대상으로 랜섬웨어 여부를 실시간으로 판단하는 것은 매우 비효율적이다. 그러므로 검사 대상 프로그램의 종류와 수를 제한하는 방안이 필요하다.

(2) 랜섬웨어와 같은 악성 프로그램의 종류는 매우 다양하며, 그 수도 빠르게 증가하고 있다. 그에 반하여 일반 사용자가 시스템에서 사용하는 문서, 이미지, 멀티미디어와 같은 파일의 종류와 사용자가 해당 파일들을 이용할 때 사용하는 프로그램의 수는 일반적으로 많지 않다.

이러한 고려 사항을 반영하여, 수집된 랜섬웨어들을 분석한 블랙리스트를 기반으로 파일 접근을 불허하는 통제방식보다는 특정 종류의 파일에 정상적으로 접근 허용된 프로그램의 목록을 화이트리스트(Whitelist)로 구성하여 해당 프로그램만 접근을 허용하는 접근통제가 보다 효과적이다. 이를 위하여, 운영체제 내에서 파일 접근 정책을 수립할 때, 기본적으로 모든 접근 불허(All Access Deny) 접근통제 정책을 구현하고, 화이트리스트를 이용하여 접근허용 범위를 통제한다. 즉, 시스템에서 보호가 필요한 파일의 종류와 해당 파일들에 정상적으로 접근하기 위해 사용하는 프로그램의 목록(화이트리스트)을 작성하고 관리한 후, 화이트리스트에 포함된 프로그램을 제외한 다른 프로그램들의 파일 접근을 허용하지 않는 방안을 고려할 수 있다[5][6][7]. 이와 같은 접근 통제 방법은 윈도우즈 운영체제의 기본 보안 시스템으로도 제안되었다. 윈도우즈 운영체제에서 랜섬웨어에 대응하기 위해 제안된 제어된 폴더 액세스(Controlled Folder Access, CFA)의 경우, 특정 폴더의 접근을 제어하여 랜섬웨어와 같은 불법 프로세스의 접근을 차단한다[8].

그러나 화이트리스트를 이용한 접근 통제의 경우에도 한 가지 취약점이 있다. DLL 삽입 공격기술(Dynamic-Link Library Injection Attack)을 활용하여 화이트리스트에 포함된 정상 프로그램 내에 악성 코드를 의도적으로 삽입한 후, 해당 악성 코드가 시스템에 저장된 파일에 불법적으로 접근하도록 하는 것이다. 이 경우, DLL 삽입 공격의 공격 대상이 된 프로그램은 화이트리스트에 명시되어 있기 때문에 시스템에 저장된 파일에 대한 접근이 허용되고, 결과적으로 불법적인 파일접근이 허용된다. 실제로 윈도우즈 운영체제의 랜섬웨어 대응 기술로 소개된 CFA도 DLL 삽입 공격으로 무력화 될 수 있음이 알려졌다[9].

그러므로 화이트리스트 기반의 접근통제 기술을 응용하여 랜섬웨어를 탐지하고 대응하는 기술을 실용화하기 위해서는 DLL 삽입 공격을 이용한 우회공격에 대한 대응 방안이 반드시 필요하다.

본 논문에서는 화이트리스트 기반의 접근통제 솔루션에 대한 DLL 삽입 공격을 감시하고, 기존에 파일접근권한을 부여받은 프로그램이 DLL 삽입 공격을 받은 경우, 이를 통제하기 위한 기술을 제안한다. 제안된 기술을 활용하면 화이트리스트 기반의 접근통제 기술을 보다 안전하게 구현할 수 있다. 또한 제안된 기술을 [5]에 실제 적

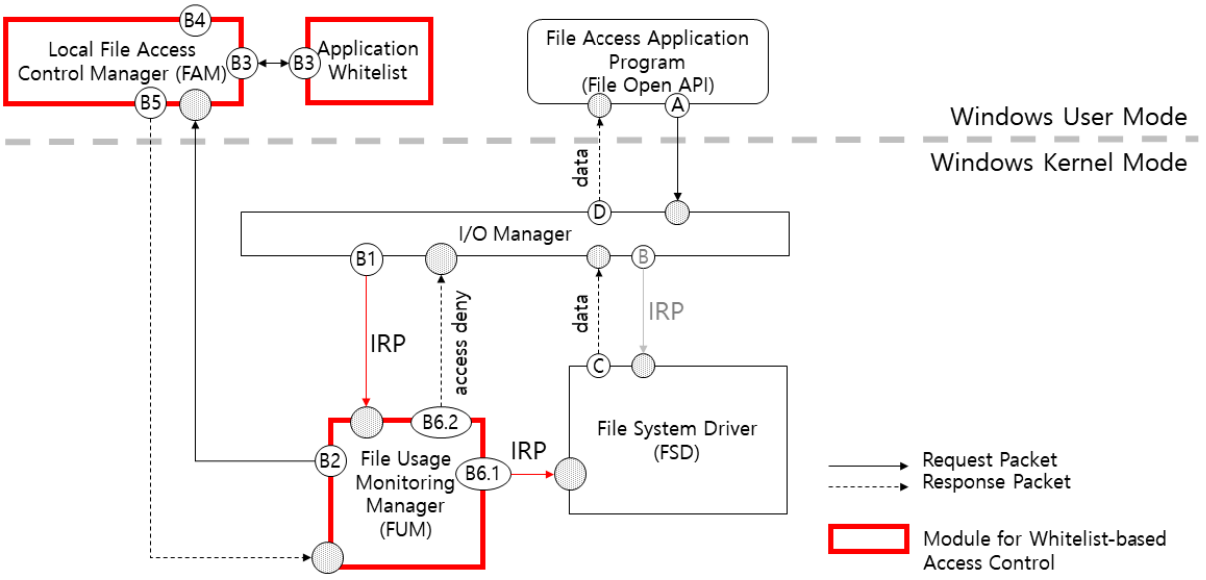


Fig. 1. Whitelist-based Ransomware Prevention System [5].
그림 1. 화이트리스트 기반 랜섬웨어 방지 솔루션 [5]

용하여 제안된 기술의 랜섬웨어 통제 가능성 및 성능을 시험하였다.

본 논문의 구성은 다음과 같다. 2장에서는 화이트리스트 기반의 랜섬웨어 탐지 기술을 간략하게 살펴본다. 3장에서는 DLL 삽입 공격기술을 활용하는 방법을 설명하고 그 대응 기술을 제안한다. 4장에서는 제안된 기술을 적용한 테스트 코드의 시험 결과를 설명하고 5장에서 연구 결론을 기술한다.

II. 화이트리스트 기반 랜섬웨어 탐지

기존의 블랙리스트 기반의 랜섬웨어 탐지 솔루션의 단점을 보완하고 대응 역량을 강화하기 위하여 화이트리스트 기반의 랜섬웨어 탐지 솔루션이 제안되었다[5]. 그림 1은 [5]에서 제안된 화이트리스트 기반의 랜섬웨어 탐지 및 대응 솔루션의 개념을 설명한다. 윈도우즈 운영 체제에서 프로그램의 파일 접근 요청은 일반적으로 I/O Manager가 IRP(I/O Request Packet)를 생성하고, 이 IRP를 File System Driver에게 전달한다(Step B). File System Driver는 해당 IRP를 처리한 후 그 처리된 결과를 I/O Manager를 통해 파일 접근을 요청한 프로그램에게 전달한다(Step C와 D). [5]에서 제안된 기술은 I/O Manager가 발생한 IRP를 File Usage Monitor (FUM)가 File System Driver 보다 앞서 획득하고 접근 권한을 판단하도록 설계되었다. FUM은 윈도우즈 미니필터를 활용하여 구현할 수 있다[10]. [5]에서 제안된 절차는 다음과 같다:

(A) 프로그램이 파일 접근을 요청하면, I/O Manager는 해당 요청을 처리하기 위하여 IRP를 생성한다.

(B1) FUM은 해당 IRP를 File System Driver 보다 앞서 획득한다.

(B2) FUM은 IRP 정보를 분석하여 파일에 접근 하려는 프로세스의 정보를 File Access Control Manager (FAM)에 전달한다.

(B3) FAM은 관리하는 화이트리스트에 해당 프로세스 정보가 기록되어 있는지 확인한다.

(B4) FAM은 해당 프로세스가 화이트리스트에 포함되어 있다면 접근을 허용하고, 그렇지 않다면 접근을 불허한다.

(B5) FAM은 판단 결과를 FUM에게 전달한다.

(B6) 만약 파일 접근 불허 판단을 전달받으면, FUM은 해당 IRP를 삭제하고 처리를 종료한다. 그렇지 않다면 해당 IRP를 File System Driver에게 전달하여 해당 프로그램이 정상적으로 파일에 접근하도록 나머지 절차를 수행할 수 있게 한다.

III. DLL 삽입 공격 및 대응 기술

1. DLL 삽입 공격

소프트웨어 프로그램 개발 시 사용되는 라이브러리

(Library)는 크게 정적 라이브러리(Static Library)와 동적 라이브러리(Dynamic Link Library, DLL)로 구분된다. 정적 라이브러리의 경우는 프로그램의 실행파일 구성 단계에서 실행파일에 포함되어 배포되는 반면에, DLL의 경우 프로그램 실행 중 필요에 따라 라이브러리를 메모리에 탑재하거나 기존에 다른 프로세스에 의해 이미 탑재된 라이브러리를 공유하도록 설계되었다.

DLL 삽입 공격은 DLL의 동적 연결 특성을 악용하여 공격 코드를 실행 중인 프로세스에 불법적으로 삽입하고 실행시키는 공격 기술이다. DLL 삽입 공격은 정상 프로세스에 공격 코드를 삽입하여 접근 권한을 불법적으로 취득하거나 악성 프로세스 탐지 솔루션을 무력화하기 위해서 사용되어져 왔다. DLL 삽입 공격을 수행하는 대표적인 방법으로는 정적 DLL 삽입 기술과 동적 DLL 삽입 기술이 있다.

가. 정적 DLL 삽입 공격

프로그램 내에서 정상적으로 DLL을 활용하는 경우, 운영체제는 프로그램이 메모리에 탑재되는 과정에서 프로그램의 PE(Portable Executable) 파일을 분석하여 필요한 DLL 파일 목록을 확보한다. 이와 같은 DLL 파일 정보는 PE 파일의 IDT(Import Directory Table)를 분석하여 확보할 수 있다. 정적 DLL 삽입 공격은 다음과 같은 두 가지 방안이 가능하다.

(1) PE 파일 수정: 비정상 DLL 파일을 생성한 후, 해당 DLL 파일 정보를 PE 파일의 IDT에 추가한다.

(2) DLL 파일 수정: 정상 DLL 파일과 같은 이름의 비정상 DLL 파일을 생성한다. 이 때, 비정상 DLL 파일은 정상 DLL의 기능도 포함하도록 설계한다.

정적 DLL 삽입기술을 활용할 경우 시스템 내에 정상적으로 설치된 프로세스의 PE 파일 또는 DLL 파일을 수정하기 때문에, 정상 파일의 코드 서명(Code Signature)을 활용하여 탐지할 수 있다.

나. 동적 DLL 삽입 공격

동적 DLL 삽입 공격은 시스템 내에서 실행 중인 프로세스(또는 프로그램) 내에 추가로 공격용 DLL 파일이 탑재되도록 유도하고, 해당 DLL 파일이 메모리에 탑재될 때 자동으로 수행되는 DLLMain 함수에 공격코드를 구현함으로써 공격이 자동적으로 수행되게 한다. 윈도우즈 운영 체제 내에서 동적 DLL 삽입 공격을 수행하는 대표적인 방법은 CreateRemoteThread 함수를 악용하여 LoadLibraryA 함수가 공격용 DLL 파일을 공격목표 프로세스에 탑재하도록 유도하는 것이다. 원격 스레드 생성 기술을 이용한 동적 DLL 삽입 공격은 다음과 같은 절차에 따라 수행 된다[11]. 동적 DLL 삽입 공격을 수행하기 위해서 공격자는 다음과 같은 두 개의 공격 파일을 제작 한다:

(1) 공격목표 프로세스에 삽입 될 공격용 DLL 파일: 실제 공격 코드는 DLL 파일의 DLLMain 함수에서 호출되도록 설계한다.

(2) DLL 삽입을 수행할 공격 프로그램(DLL 삽입자, Injector) 파일: 공격자는 DLL 삽입자를 실행시켜 공격목표 프로세스에 앞서 작성한 공격용 DLL 파일을 원격으로 탑재시킨다.

이 때, DLL 삽입자는 다음과 같은 절차에 따라 공격을 수행하도록 설계한다.

(1) 삽입자의 실행권한을 SE_DEBUG_NAME으로 설정한다. 이는 삽입자가 디버깅 권한과 메모리 접근 권한을 소유할 수 있도록 한다.

(2) 시스템에서 실행 중인 공격목표 프로세스의 프로세스 ID (PID)를 획득한다. 획득된 PID를 이용하여 공격목표 프로세스의 핸들러(Handler)를 시스템에 요청할 수 있다.

(3) 공격목표 프로세스에게 메모리를 할당받아 공격용 DLL 파일이 저장되어 있는 위치 경로를 할당받은 메모리에 기록한다.

(4) 시스템 내에서 현재 메모리에 탑재되어 있는 LoadLibraryA 함수의 시작 주소를 탐색한다. LoadLibraryA 함수는 Win32 API 중 하나로 윈도우즈 운영체제의 기본 시스템 DLL 중 하나인 kernel32.dll도 사용하기 때문에, 운영체제 부팅 후 kernel32.dll이 메모리에 탑재될 때 해당 함수가 탑재된 메모리 주소가 결정된다. DLL 파일 운영 특성상 이후 해당 함수를 이용하는 모든 프로세스가 동일한 함수 주소를 이용하게 된다.

(5) CreateRemoteThread 함수의 입력 매개변수로 공격목표 프로세스의 핸들러, LoadLibraryA 함수의 시작주소, 그리고 공격용 DLL 파일 경로를 지정하여 해당 DLL 파일이 공격 목표 프로세스에서 탑재되도록 한다. 공격용 DLL 파일이 공격목표 프로세스에 성공적으로 탑재되면 해당 DLL 파일의 DLLMain 함수가 자동으로 호출된다.

동적 DLL 삽입 공격의 경우 목표 프로세스의 PE 파일을 수정할 필요가 없다. 또한, 시스템의 프로세스들이 이미 사용 중인 DLL 파일을 변경하지도 않는다. 그러므로 실시간 탐지가 어렵다. 그러나 화이트리스트와 같은 접

```
VOID (*PCREATE_THREAD_NOTIFY_ROUTINE)(
        [in] HANDLE ProcessId,
        [in] HANDLE ThreadId,
        [in] BOOLEAN Create
);
```

Fig. 2. Thread notify callback function type.
그림 2. 스레드 통지 콜백함수 타입

근통제 기술을 이용하여 프로세스의 파일 접근을 통제하려고 할 때, 동적 DLL 삽입 공격은 이를 우회하거나 무력화시킬 수 있다.

2. 동적 DLL 삽입 공격 탐지

동적 DLL 삽입 공격을 위해 가장 많이 사용되는 기술이 원격 스레드 생성 기술이다. 그러므로 시스템 내에서 발생하는 스레드 관련 이벤트를 모니터링하고, 의심 스레드를 분별할 수 있다면 DLL 삽입 공격 탐지가 가능하다. 실제 윈도우즈 운영체제는 프로세스나 스레드의 생성/종료, 이미지 로드 같은 이벤트를 실시간으로 감시할 수 있는 기능을 제공한다. 그러므로 이러한 기능을 활용하여 동적 DLL 삽입 공격 시도 여부도 감시할 수 있다.

가. 스레드 이벤트 모니터링

CreateThreadNotifyRoutine은 새로운 스레드가 생성되거나 기존 스레드가 종료될 때 호출되는 루틴을 의미한다. 이 루틴은 프로세스의 주 스레드(Main Thread) 생성/종료 시뿐만 아니라 새로운 스레드를 생성하기 위하여 CreateThread 함수 또는 CreateRemoteThread 함수를 호출할 때에도 수행된다. 이러한 콜백함수는 윈도우즈에서 제공하는 PsSetCreateThreadNotifyRoutine 함수를 이용하여 등록할 수 있다. 이 스레드 통지 콜백함수는 PCREATE_THREAD_NOTIFY_ROUTINE 유형의 매개변수 NotifyRoutine을 이용하여 스레드 이벤트가 발생할 때 호출되도록 등록시킬 함수의 포인터 값을 지정하게 설계되었다. 이 콜백함수의 원시 형태는 그림 2와 같다. 콜백함수를 통해 운영체제로부터 제공되는 ThreadId와 ProcessId는 이벤트가 발생된 스레드의 ID와 해당 스레드가 탑재된 프로세스의 ID를 각각 의미한다.

이 콜백함수 내에서 PsGetCurrentProcessId 함수와 PsGetCurrentThreadId 함수를 호출하면 해당 스레드를 생성한 프로세스 ID와 스레드 ID를 확인할 수 있다. 단, 획득한 프로세스 ID와 스레드 ID가 해당 함수를 호출한 프로세스/스레드의 ID가 아니라는 점을 유의해야 한다.

나. 의심 스레드 판단 기준

[4]에서는 앞서 설명한 스레드 이벤트 모니터링 기술을 활용하여 의심 프로세스 및 스레드 생성 이벤트를 감시하는 방안을 제안하였다. 이 기술은 악성코드의 프로세스 또는 악성코드에서 일반적으로 악용되는 스레드를 모니터링하고, 대응하기 위하여 제안되었다.

CreateRemoteThread 함수를 사용하여 정상 프로세스에 DLL 삽입 공격을 시도하는 경우, 호출자(Caller)와 피호출자(Callee)에 따라 프로세스 ID(PID)와 스레드 ID(TID)는 표 1과 같이 설정된다.

Table 1. Process/Thread ID.
표 1. 프로세스/스레드 ID

ID \ Case		Case 1 PA→PB	Case 2 PA→TA	Case 3 PA→TB
1	PID	PID[B]	PID[A]	PID[B]
	TID	TID[B]	TID[A]	TID[B]
2	cPID	PID[A]	PID[A]	PID[A]
3	cTID	TID[A]	TID[A]	TID[A]

표 1에서 ID 열의 1항은 스레드 통지 콜백함수에 의해 전달되는 프로세스와 스레드의 ID 값을 각각 의미한다. ID 열의 2항과 3항은 스레드 통지 콜백함수 내에서 이벤트가 발생된 스레드의 생성자 정보를 확인하기 위해 PsGetCurrentProcessId와 PsGetCurrentThreadId 함수를 호출하여 획득한 프로세스와 스레드의 ID 값을 각각 의미한다. PID와 TID는 프로세스 ID와 스레드 ID를 의미한다. PA/PB는 프로세스를, TA/TB는 스레드를 의미한다.

Case 1은 프로세스 A(PA)가 프로세스 B(PB)를 생성하는 경우를 의미하고, Case 2는 PA가 자기 자신이 사용할 스레드(TA)를 PA 내에 탑재하는 것을 의미한다. Case 1과 Case 2는 모두 정상적인 동작이라 판단된다.

Case 3은 PA가 스레드를 PB에 탑재시키는 경우이다. 표 1에서 알 수 있듯이, 이 경우 PID와 cPID 값이 서로 다른 ID 값을 갖게 된다. 즉, 스레드가 탑재된 프로세스와 스레드를 생성한 프로세스가 서로 다른 프로세스임을 알 수 있다. CreateRemoteThread 함수를 이용하여 원격으로 DLL을 삽입하는 경우도 이와 같은 결과가 발생

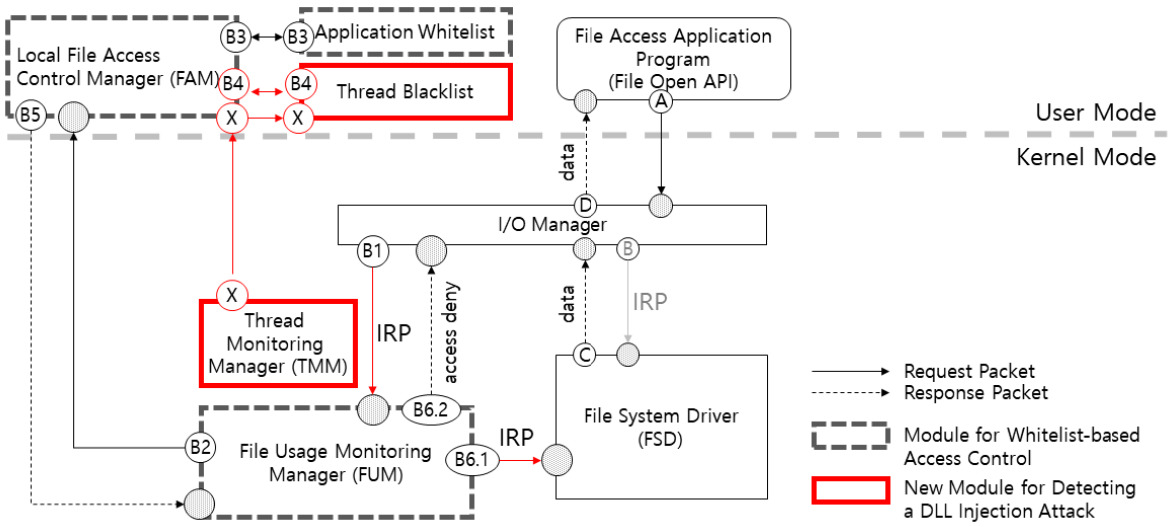


Fig. 3. A countermeasure procedure against a bypass attack using DLL injection skill.
그림 3. DLL 삽입 기술을 이용한 우회 공격 대응 절차

하게 된다. 단, Case 1의 경우도 Case 3과 유사한 결과를 나타내지만, Case 1의 경우는 PB의 메인 스레드가 생성되는 것으로, 프로세스에 생성된 스레드의 수를 확인하면 구분할 수 있다.

그러므로 스레드 통지 콜백함수 내에서 Case 3인 경우가 탐지되면 이를 의심 스레드 생성 이벤트로 간주하고 의심 스레드 목록에 해당 스레드 정보를 추가하여 관리한다.

그러나 [4]에서 제안된 기술은 DLL 삽입공격 여부만을 판단하기 때문에, 삽입된 DLL이 랜섬웨어 공격에 활용됐는지 여부를 판단할 수 없다.

본 논문에서는 제안된 스레드 모니터링 기술을 랜섬웨어 통제 솔루션에 응용할 수 있도록 개선하고, 이를 [5]에 실제 적용하여 그 성능을 확인한다. 이를 위하여, 의심 스레드의 파일 접근 여부를 추가 감시하고 파일접근 통제에 활용하는 방안을 제안한다. 또한, 기존의 화이트리스트 기반 접근통제 기술에 제안된 기술을 접목시켜 구현할 수 있도록 통제절차를 제안하였다.

다. 화이트리스트 기반 접근통제 우회공격 대응

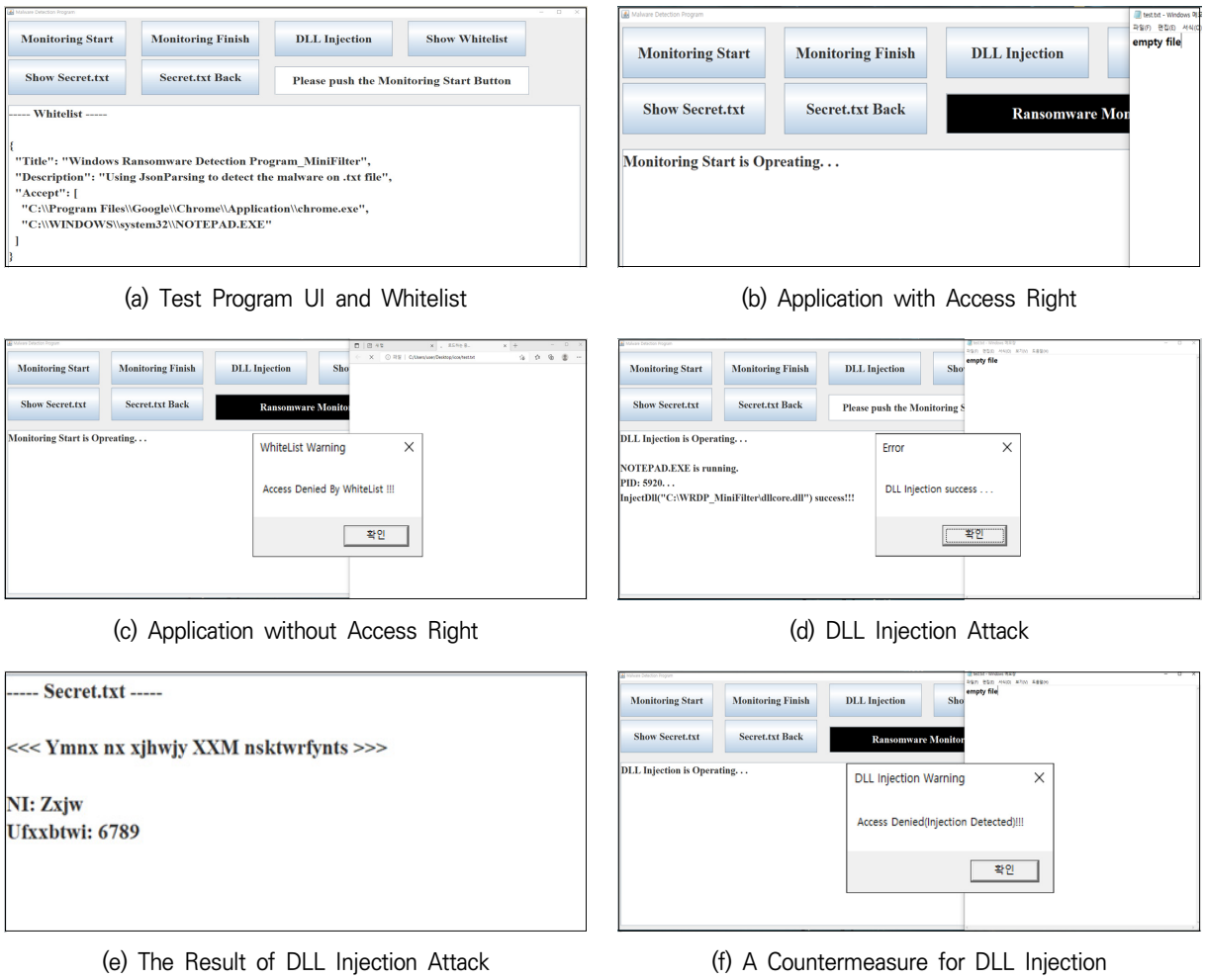
DLL 삽입 공격에 대응하기 위하여 앞서 설명한 화이트리스트 기반 랜섬웨어 대응 모니터링 기술을 다음과 같이 개선한다.

(1) 화이트리스트 기반 랜섬웨어 대응 모니터링에서 사용하는 미니필터의 I/O 콜백함수 내에서 PsGetCurrentThreadId 함수를 호출하여 파일에 접근하려는 스레드의 생성자 ID 값을 획득한다. 이렇게 획득된 스레드의 생성자 ID 값을 앞서 작성된 의심 스레드 목록의 ID 값들과 비교한다.

(2) 만약 해당 스레드의 생성자 ID와 동일한 ID가 의심 스레드 목록에서 발견되면, 해당 스레드는 DLL 삽입 공격으로 생성된 것으로 간주하여 파일 접근을 거부한다. 그림 3은 화이트리스트 기반 접근통제 우회공격을 탐지하고 대응하기 위한 절차를 설명한다.

그림 1에서 파일 접근통제 관리자는 단계 B2와 단계 B3에서 수집된 프로세스 정보를 단계 B4에서 비교 분석하여 파일에 접근하려는 프로세스의 접근권한을 판단하고, 판단 결과를 파일 사용 모니터링 관리자에게 통보하여 파일 접근 요청 프로세스의 접근 허용 여부를 처리하도록 제안되었다. 본 논문에서 제안하는 방안은 단계 X에서 스레드 감시 관리자(Thread Monitoring Manager, TMM)는 의심 스레드 정보를 파일 접근통제 관리자에게 전송하고, 관리자는 스레드 블랙리스트에 해당 정보를 추가한다. 단계 X는 기존의 파일 접근 절차의 단계들과는 독립적으로 수행된다.

(3) 프로그램이 파일 접근을 시도하면, 해당 프로그램의 파일접근요청 프로세스 정보를 기반으로 화이트리스트를 활용한 접근통제 절차를 단계 B1부터 단계 B3까지 동일하게 수행한다[5]. 만약 화이트리스트 정보에 따라 해당 요청 프로세스가 정당한 접근권한을 소유하고 있다고 판단되면, 단계 B4에서 파일 접근통제 관리자는 스레드 블랙리스트 정보를 분석하여, 접근을 시도하는 스레



(a) Test Program UI and Whitelist

(b) Application with Access Right

(c) Application without Access Right

(d) DLL Injection Attack

(e) The Result of DLL Injection Attack

(f) A Countermeasure for DLL Injection

Fig. 4. Implementation.
그림 4. 구현결과

드가 블랙리스트에 포함되어 있는지 여부를 확인한다. 이 때, 프로그램의 파일접근 요청을 처리하기 위해 생성된 IRP를 분석하여, 스레드 블랙리스트 목록에 포함된 스레드가 해당 IRP 정보에 포함되어 있는지를 판단한다. 만약, 해당 IRP에 기록되어 있는 프로세스의 스레드가 블랙리스트에 포함되어 있다면, 파일 접근을 불허하도록 파일 사용 모니터링 관리자에게 통보한다.

Ⅳ. 구현 결과 및 성능 평가

1. 구현 결과

그림 4는 제안된 DLL 삽입 공격에 대한 대응 기법의 구현 결과로, 화이트리스트에 포함된 프로그램을 대상으로 DLL 삽입 공격이 수행된 경우, 해당 프로그램이 랜섬웨어의 숙주가 되어 파일을 암호화 한 결과를 보여준다: (a)는 [5]에서 제안된 랜섬웨어 탐지 프로그램이다. 탐지 프로그램은 .txt 파일의 접근권한을 정의한 화이트리스트에 chrome.exe와 notepad.exe가 포함되어 있다. (b)는 notepad.exe를 이용하여 테스트 파일에 접근한 결과를 보여준다. (c)는 화이트리스트에 지정된 프로그램 이외에 다른 프로그램을 이용하여 테스트 파일에 접근을 시도한 결과, 접근이 불허된 상태를 보여준다. (d)는 notepad.exe에 공격용 DLL이 성공적으로 삽입된 상태다. (e)는 notepad.exe에 삽입된 공격 DLL에 의해서 파일이 암호화된 상태를 보여준다. (f)는 공격용 DLL이 삽입된 notepad.exe가 파일에 접근하려고 할 때, 해당 접근 요청이 거부된 결과다.

2. 성능분석

랜섬웨어를 탐지하고 대응하는 기존의 접근 방식과 달리 [5]는 화이트리스트 기반 접근통제 기술을 적용하여, 접근권한이 부여된 프로그램을 제외한 모든 프로세스의 파일접근을 통제함으로써 랜섬웨어의 동작도 통제되도록 설계되었다. 그러므로 [5]의 경우 랜섬웨어 데이터베이스

에서 선택한 랜섬웨어뿐만 아니라 테스트를 위해 새로 구현된 랜섬웨어까지도 악성행위 성공률이 0%로 측정되었다.

본 논문은 [5]에서 제안한 화이트리스트 기반의 랜섬웨어 탐지 솔루션의 성능 개선을 목적으로 개발되었기 때문에, 화이트리스트에 기록이 없는 모든 프로세스의 파일 접근을 통제한다. 또한, 화이트리스트에 기록된 프로그램의 경우에도 해당 프로그램에 대하여 원격 스레드 생성을 악용한 DLL 삽입 공격이 수행된 경우를 가정하고 시뮬레이션 한 결과 DLL 삽입 기술을 악용한 파일 접근 또한 효과적으로 통제하였다.

그림 5는 시스템 성능 분석 결과를 나타낸다. 성능 분석을 위하여 3개의 경우를 가정하여 시스템의 CPU 점유율을 30회 측정하였다.

(1) 경우 1은 시스템에 노트패드(notepad)와 악성코드 탐지 프로그램을 실행시킨 후 측정한 결과이다.

(2) 경우 2는 노트패드와 본 논문에서 제안된 기술로 구현된 랜섬웨어 탐지 솔루션을 실행시킨 후 측정한 결과이다.

(3) 경우 3은 워드프로세스와 본 논문에서 제안된 기술로 구현된 랜섬웨어 탐지 솔루션을 실행시키고 DLL 삽입 공격을 수행한 후 측정한 결과이다.

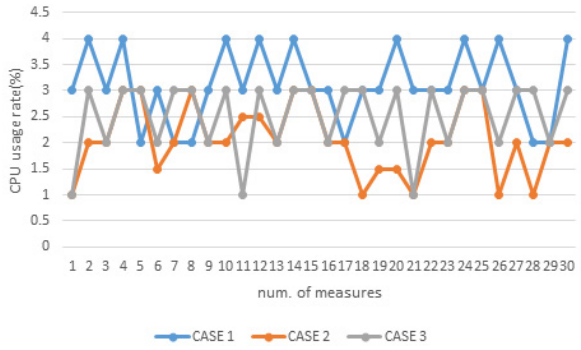


Fig. 5. Evaluation (CPU usage rate) result.
그림 5. 성능분석 (CPU 점유율)

측정 환경은 다음과 같다: CPU(Intel i7-10700F), RAM(32GB), OS(Windows 10, 64 bit). 그림 5의 경우 2의 결과에서 확인할 수 있는 것처럼 의심 스레드를 감시하는 기능을 추가하더라도 CPU 평균 점유율이 2.05%로, 경우 1의 범용 악성코드 분석 솔루션의 평균 점유율 3.1% 보다 효율적임을 확인할 수 있다. 또한, 경우 3에서와 같이 실제 DLL 삽입공격이 수행되는 중에 이를 식별하고 대응하는 경우에도 평균 점유율은 2.5%로, 경우 1의 3.1% 보다 효율적임을 확인할 수 있다.

추가/개선된 모듈의 구현 성능은 다음과 같다: 경우 3과 같은 조건에서

(1) 노트패드에 DLL 삽입공격이 시도될 때, 그림 3의 TMM이 이를 탐지하고 의심스레드 목록을 갱신하는데 소요되는 시간은 평균 110 ns이다.

(2) 그림 3의 FUM과 FAM이 의심스레드 목록과 화이트리스트를 기반으로 프로그램의 파일접근 권한을 판단하고, 접근통제 정책에 따라 악성행위를 통제하는데 소요되는 시간은 평균 170 ns이다.

V. 결론

[5]와 [8]은 접근통제 기술을 활용하여 랜섬웨어로 의심되는 프로세스의 파일 접근을 통제하는 새로운 방식의 랜섬웨어 대응 솔루션을 제안하였다. 특히, 이러한 기술들은 화이트리스트를 기반으로 접근통제 정책을 수립하고, 정당한 권한이 부여된 프로세스 이외의 모든 프로세스의 파일 접근을 통제함으로써 랜섬웨어도 통제할 수 있었다. 그러나 DLL 삽입 기술을 활용하여 이러한 통제 정책을 우회할 수 있다는 취약점이 지적되었다.

본 논문은 이러한 취약점을 개선하기 위하여 DLL 삽입공격을 감시하는 기술을 화이트리스트 기반의 랜섬웨어 솔루션에 접목시켜 안전성을 강화하였다. 효율적인 구현을 위하여 특정 어플리케이션에 대한 DLL 삽입공격만을 모니터링하고, 스레드 블랙리스트를 관리하도록 제안하였다. 또한, 화이트리스트 기반 접근 통제 솔루션과 연동시켜 프로그램의 파일접근 요청 처리를 위한 IRP와 스레드 블랙리스트를 분석하여 공격 여부를 판단하도록 제안하였다.

제안된 기술을 [5]에 적용하여 DLL 삽입공격이 실행된 것으로 의심되는 프로그램이 파일에 접근하는 것을 봉쇄하여 화이트리스트 기반의 랜섬웨어 솔루션의 보안성을 개선하였으며, 추가된 DLL 삽입공격 대응 기술로 인한 시스템 자원 소모 정도를 측정한 결과 기존의 악성코드 탐지 솔루션보다 효율적임을 확인하였다.

References

[1] S. Chakkaravarthy, D. Sangeetha, and V. Vaidehi, “A Survey on malware analysis and mitigation techniques,” *Computer Science Review*, vol.32, pp.1-23, 2019. DOI: 10.1016/j.cosrev.2019.01.002

[2] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol.153, no.1, 2020.
DOI: 10.1016/j.jnca.2019.102526
[3] B. Khammas, "Ransomware Detection using Random Forest Technique," *ICT Express*, vol.6, no.4, 2020. DOI: 10.1016/j.icte.2020.11.001
[4] BS K, WH C, and DJ J, "A Study on the Tracking and Blocking of Malicious Actors through Thread-Based Monitoring," *Korea Institute of Information Security and Cryptology*, vol.30, no.1, pp.75-86, 2020. DOI: 10.13089/JKIISC.2020.30.1.75
[5] D. Kim and J. Lee, "Blacklist vs. Whitelist-Based Ransomware Solutions," *IEEE Consumer Electronics Magazine*, vol.9, no.3, pp.22-28, 2020.
DOI: 10.1109/MCE.2019.2956192
[6] T. McIntosh, A. Kayes, Y. Chen, A. Ng, and P. Watters, "Ransomware Mitigation in the Modern Era: A Comprehensive Review, Research Challenges, and Future Directions," *Computer Science ACM Computing Surveys (CSUR)*, vol.7, 2021.
DOI: 10.1145/3479393
[7] S. Kim, I. Hwang, and D. Kim, "A Study on Creation of Secure Storage Area and Access Control to Protect Data from Unspecified Threats," *Journal of the Society of Disaster Information*, vol.17, no.4, pp.897-903, 2021.
DOI: 10.15683/kosdi.2021.12.31.897
[8] Microsoft Docs, "Enable controlled folder access," https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/enable-controlled-folders?view=o365-worldwide
[9] L. Abrams, "Windows 10 Ransomware Protection Bypassed Using DLL Injection," https://www.bleepingcomputer.com/news/security/windows-10-ransomware-protection-bypassed-using-dll-injection/
[10] Microsoft Docs, "Filter Manager and Minifilter Driver Architecture," 2020, https://docs.microsoft.com/ko-kr/windows-hardware/drivers/ifs/filter-manager-concepts

BIOGRAPHY

DaeYoub Kim (Member)

1994 : BS degree in Math., Korea University.
1997 : MS degree in Math., Korea University.
2000 : PhD degree in Math., Korea University.
2000~2002 : Research Engineer, SECUI
2002~2012 : Senior Researcher and Project Manager, Samsung Electronics.
2012~ : Professor, Dept. of Information Security, Suwon Univ.