ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 23-03-05
j.inst.Korean.electr.electron.eng.Vol.27,No.3,239~245,September 2023
https : //doi.org/10.7471/ikeee.2023.27.3.239
27

계층화된 쓰레드 생성을 이용한 DLL 삽입 탐지기술 우회 공격 기법

Hierarchical Threads Generation-based Bypassing Attack on DLL Injection Monitoring System

김 대 엽*★

DaeYoub Kim*★

Abstract

Whitelist-based ransomware solution is known as being vulnerable to false impersonation attack using DLL injection attack. To solve this problem, it is proposed to monitor DLL injection attack and to integrate the monitoring result to ransomware solutions. In this paper, we show that attackers can easily bypass the monitoring mechanism and then illegally access files of a target system. It means that whitelist-based ransomware solutions are still vulnerable.

요 약

화이트리스트 기반 랜섬웨어 솔루션이 DLL 삽입공격을 활용한 사칭공격에 취약한 것으로 알려진 후, 이러한 문제점을 개선하기 위하여 DLL 삽입 공격을 활용한 사칭공격을 탐지하고, 랜섬웨어 탐지 및 대응 기술과 연동하는 기술이 제안되었다. 본 논문에서는 공격자가 이러한 탐지기술을 우회하여 불법적으로 공격 대상의 파일에 접근할 수 있음을 보여준다. 이는 화이트리스트 기반 랜섬웨어 솔루션이 여전히 DLL 삽입 공격에 취약함을 의미한다. 특히, 본 논문에서는 랜섬웨어 솔루션을 대상으로 실제 공격을 수행하여, 그 가능성을 증명하였다.

Key words : Ransomware, Whitelist Access Control, Dynamic Link Library Injection, False Impersonation

Ⅰ. 서론

랜섬웨어와 같은 악성코드의 피해가 날이 갈수록 증가하고 있다. 특히, 스마트 자동차, 전력, 공장, 도시와 같이 기반 시설 및 설비가 자동화 되고 네트워크에 연결됨에 따라 이러한 피해는 더욱 증가할 것으로 예상되고 있다[1-3]. 그러나 기존 악성코드 탐지 및 대응 기술은 사전에 수집/분석된 악성코드의 특성을 블랙리스트(Blacklist)로 관리하고, 이 블랙리스트를 기반으로 검사 대상이 되는 실행코드/파일을 비교분석하여 악성코드 여부를 결정

* Professor, Dept. of Information Security, Suwon University
★ Corresponding author
E-mail : daeyoub69@suwon.ac.kr, Tel : +82-31-229-8284
※ Acknowledgment
This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIT)(No. NRF-2021R1F1A1062954).
Manuscript received Aug. 23, 2023; revised Sep. 18, 2023; accepted Sep. 18, 2023.

하는 방식을 채택하고 있다. 이와 같은 블랙리스트 기반의 기술은 False Negative Error 발생 비율이 낮아 알려진 악성코드에 대한 탐지 시, 정확도가 높은 방안으로는 안정적인 대응 방식이라 할 수 있다[4]. 그러나 블랙리스트 기반의 악성코드 탐지를 위해서는 악성코드 샘플 수집이 우선 요구된다. 그러나 악성코드 샘플 수집을 위해서는 악성코드가 공격 대상이 된 시스템에서 실행되어 피해가 발생했음을 의미한다. 악성코드의 대상이 개별 시스템에 한정된 경우, 악성코드로 인한 피해 규모와 대상도 제한적이었다. 그러나 스마트 자동차나 전력과 같은 기반시설이 공격 대상이 된다면, 악성코드의 실행이 직접적인 인명 피해를 야기할 수 있다. 그러므로 악성코드의 실행 전에 선제적으로 탐지 및 대응할 수 있는 방안이 시급히 필요하다.

화이트리스트 기반의 접근통제 기술은 정통적으로 강제적 접근통제(MAC, Mandatory Access Control)와 같은 접근통제 기법에서 주로 활용되어져왔다[5]. 이와 같은 접근통제는 접근 허용 여부를 판단할 때 사용되는 화이트리스트를 구성하기 위하여 접근통제가 요구되는 객체(Object)와 허용된 접근권한(Privilege)을 정의하고, 이를 기반으로 접근요청에 대한 허용(Permit) 또는 금지(Deny)를 결정한다. 이러한 접근통제 기술은 기본적으로 객체에 대한 모든 접근을 금지하는 정책(All-Deny)을 기본으로 설계되었으며, 객체에 대한 접근을 요청한 사용자가 자신이 갖고 있는 권한을 증명한 경우에만 해당 객체에 대한 접근을 허용한다.

악성코드의 동작을 효율적으로 제어하기 위하여 화이트리스트 기반 접근 통제 기술을 적용하는 방안이 제안되었다[6-7]. 특히, 화이트리스트 기반의 접근 통제 기술은 화이트리스트를 구성하는 객체 및 접근권한을 명확히 정의해야 한다. 그렇지 않을 경우, False Negative Error로 인하여 시스템 또는 서비스 장애를 유발할 수 있다. 랜섬웨어 솔루션에 화이트리스트 기반 접근통제 기술을 적용하는 경우, 화이트리스트를 다음과 같이 명확하게 정의할 수 있다:

- 객체 : 접근통제가 요구되는 시스템에 저장되어 있는 파일을 의미한다.
- 권한 : 시스템에 저장되어 있는 파일에 대한 읽기/쓰기로 정의된다.
- 사용자 : 시스템에서 운영 중인 어플리케이션 프로세스 중에서 시스템에 저장되어 있는 파일에 접근을 시도하는 프로세스로 정의한다.

이와 같이 화이트리스트를 구성하는 객체 및 권한을 명확하게 정의할 수 있기 때문에 랜섬웨어 솔루션은 화이트리스트 기반으로 구축하기에 매우 적합한 대응 방안이라 할 수 있다.

화이트리스트 기반의 접근 통제 기술은 사칭공격(False Impersonation Attack)에 취약할 수 있다. 실제로 화이트리스트 기반의 랜섬웨어 솔루션의 경우, 동적 연결 라이브러리 삽입공격(DLL Injection Attack)을 이용하여 시스템에 저장된 파일에 대한 정상적인 접근 권한을 가진 어플리케이션 프로세스에 원격으로 악성 DLL을 로딩 시킨 후, 이를 이용하여 파일에 불법적으로 접근할 수 있음이 지적되었다[8-9].

이와 같은 DLL 삽입 기술을 이용한 사칭공격을 방지하기 위한 기술이 제안되었다[10-12]. 사칭방지 기술은 DLL 삽입공격에서 악용되고 있는 CreateRemoteThread 함수의 동작을 감시하고, 이를 이용하여 악성 DLL이 다른 프로세스에 불법적으로 로딩 되면, 이를 탐지하고 통제하는 방식이다[13].

본 논문에서는 [10-12]에서 제안된 DLL 삽입공격을 탐지하는 절차를 우회하여 DLL 삽입공격을 수행하는 새로운 공격 방법을 소개한다. 소개할 공격 방법은 DLL 삽입공격 과정의 일환으로 계층화된 일련의 쓰레드(Thread)를 생성하고, 생성된 쓰레드 중 최하위 쓰레드를 이용하여 파일에 접근함으로써 DLL 삽입공격 탐지 절차를 우회한다. 이러한 공격을 이용하여 화이트리스트 기반 접근 통제를 무력화시키는 것이 가능함을 보인다.

II. DLL 삽입공격 탐지

2.1 DLL 삽입공격

CreateRemoteThread 함수를 활용하는 DLL 삽입공격이 보편적인 공격 방법 중 하나이다. DLL 삽입공격을 위한 절차는 다음과 같다[14]: tarPID는 악성 DLL을 삽입할 대상 프로세스의 식별자(PID, Process ID), dllName은 악성 DLL 파일의 저장경로를 포함한 파일 이름을 의미한다.

(1) 대상 프로세스의 메모리에 접근하기 위하여 시스템에 디버깅 권한을 요청하여 할당 받는다.
(2) tarPID를 이용하여 대상 프로세스의 핸들(handle)을 획득한다. 이 때, 대상 프로세스에 접근하기 위하여 필요한 권한(=PROCESS_ALL_ACCE)도 요청하여 획득한다.
(3) 대상 프로세스로부터 메모리를 할당 받아 dllName을 할당받은 메모리에 기록한다.

(4) 악성 DLL을 대상 프로세스에 로딩하기 위하여 LoadLibrary 함수의 시작 주소 값을 획득한다. LoadLibrary 함수는 일반적으로 kernel32.dll에서 제공하기 때문에 어플리케이션 동작과 상관없이 운영체제 구동 시 메모리에 로딩 된다.

(5) CreateRemoteThread 함수를 이용하여 대상 프로세스에 쓰레드를 생성한다. 생성한 쓰레드의 시작주소는 4단계에서 획득한 LoadLibrary 함수의 시작주소이고, 로딩 할 DLL은 3단계에서 대상 프로세스의 메모리에 기록한 dllName이다.

(6) 생성한 쓰레드가 시그널 상태가 될 때까지 강제 대기한 후, 시그널 상태가 되면 생성한 쓰레드를 종료시킨다.

2.2 DLL Injection Monitoring

2.1절에서 악성 DLL을 대상 프로세스에 강제로 삽입하기 위해서는 5단계에서 쓰레드를 생성하여 악성 DLL을 대상 프로세스에 로딩 될 때 악성 DLL의 dllMain 함수가 자동 호출됨을 이용하여 실제 악성코드를 실행한다. 이와 같은 절차를 분석하여 [10]에서는 DLL 삽입공격을 탐지하는 방안을 제안하였다: DLL 삽입을 수행하는 공격 프로세스와 대상 프로세스의 PID가 다르다는 사실에 착안하여 원격 DLL 삽입 공격을 탐지한다. 표 1은 [10]에서 제안된 PID 비교/분석 결과이다. target PID/TID는 생성된 쓰레드의 식별자 (TID, Thread ID)와 해당 쓰레드가 생성된 프로세스의 PID를 의미한다. creator PID/TID는 쓰레드를 생성한 프로세스/쓰레드의 식별자를 각각 의미한다. PID[X]는 프로세스 X의 PID를 의미하고, TID[x]는 쓰레드 x의 TID를 의미한다.

- Case 1은 프로세스 A가 새로운 프로세스 B를 생성한다. 새로운 프로세스가 생성되면 생성된 프로세스의 메인 쓰레드(TID[b])가 자동으로 생성된다.
- Case 2는 프로세스 A가 내부적으로 동작될 새로운 쓰레드 b를 생성한다.
- Case 3은 프로세스 A가 다른 프로세스 B에서 동작될 쓰레드 b를 생성한다.

Table 1. PID/TID comparison result.
표 1. PID/TID 비교분석

	Case 1	Case 2	Case 3
target PID/TID	PID[B]	PID[A]	PID[B]
	TID[b]	TID[b]	TID[b]
creator PID/TID	PID[A]	PID[A]	PID[A]
	TID[a]	TID[a]	TID[a]

Case 2의 경우, target PID와 creator PID가 PID[A]로 동일하다. 이 경우, 생성된 쓰레드 b (TID[b])는 정상적으로 생성된 쓰레드로 간주한다.

Case 1과 Case 3의 target PID와 creator PID는 다

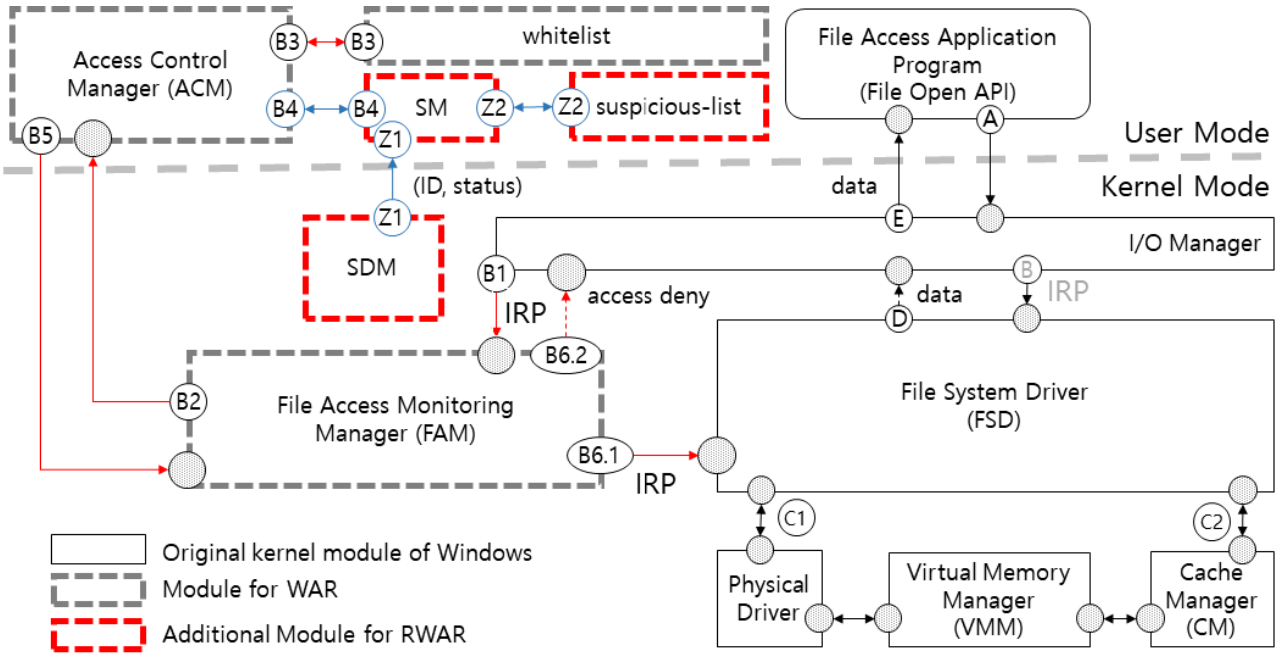


Fig. 1. Whitelist-based ransomware solution.
그림 1. 화이트리스트 기반 랜섬웨어 솔루션

른 값을 갖는다. 그러나 Case 1은 정상적인 쓰레드 생성의 결과이므로 target PID와 creator PID 값을 단순 비교하여 생성된 쓰레드가 비정상 쓰레드 생성이라고 확증할 수 없다. Case 1에서 생성된 쓰레드는 프로세스의 메인 함수를 실행하는 메인 쓰레드이다. 반면에 Case 3에서 생성된 쓰레드는 LoadLibrary 함수를 실행하는 쓰레드이다. 그러므로 Case 3의 경우, 운영체제마다 특정 주소 값, 즉 운영체제가 구동될 때 메모리에 초기 로딩 된 kernel32.dll의 LoadLibrary 함수의 시작 주소 값이다. 이 주소 값은 일반적으로 운영체제의 버전마다 고정된 값이 사용된다. 이와 같은 특성을 기반으로 TID[b]의 시작 주소 값을 확인하여 Case 3을 특정 지을 수 있다.

그림 1은 DLL 삽입공격 탐지 기술을 화이트리스트 기반 랜섬웨어 솔루션에 적용하여 구현한 결과이다 [11]. 제안된 랜섬웨어 솔루션은 2가지 정보를 기반으로 운영된다.

- 화이트 리스트 : 시스템에 설치된 어플리케이션 중에서 파일에 접근이 허용된 어플리케이션 정보를 저장한다.
- 의심 쓰레드 리스트 : 커널 수준에서 구현된 SDM (Suspicious DLL Monitor)은 [10]에서 제안된 기준에 따라 생성된 쓰레드를 탐지하여 원격 DLL 삽입공격을 위해 생성된 쓰레드로 의심되는 쓰레드 정보를 SM(Suspicious-list Manager)에 전달하여 목록을 저장/관리한다.

만약 시스템에서 구동 중인 프로세스가 파일 접근을 시도하면, 다음과 같이 파일 접근 요청을 처리한다.

(1) 파일 접근 요청(IRP, I/O Request Packet)이 커널 수준의 FSD (File System Driver)에 전달된다.

(2) 접근 통제를 위하여 FAM(File Access Monitoring Manager)가 FSD 보다 앞서 IRP를 획득하고, IRP를 사용자 수준에서 구동하고 있는 ACM(Access Control Manager)에 전달한다.

(3) ACM은 다음과 같은 2가지 기준으로 접근 권한을 검토한다.

(3-1) 화이트리스트에 해당 IRP 생성을 요청한 어플리케이션 프로세스 정보가 있는지 확인한다. 해당 정보가 없으면 파일 접근을 불허한다.

(3-2) 화이트리스트에 해당 프로세스 정보가 기재되어 있다면, 의심 쓰레드 목록을 확인하여 IRP 생성을 요청한 쓰레드 정보가 기재되어 있는지 확인한다. 만약 해당 정보가 존재한다면, 파일 접근을 불허한다.

즉, 파일에 접근하려는 프로세스의 정보가 화이트리스트에 기재되어 있고, 동시에 쓰레드 정보는 의심 쓰레드 목록에 기재되어 있지 않을 때에만 접근을 최종 허용한다.

Ⅲ. DLL 삽입공격 탐지 우회 기술

3.1 탐지 우회 기술

DLL 삽입공격에 대한 탐지기술은 공격자가 CreateRemoteThread 함수를 이용할 때, 쓰레드를 생성하는 프로세스와 대상 프로세스의 PID 값이 다르게 관찰된다는 사실에 기반을 두고 있다. 이렇게 관찰된 의심 쓰레드 정보를 의심 쓰레드 리스트로 관리하고, 의심 쓰레드가 시스템의 파일에 접근하려고 할 때, 이를 통제한다.

본 논문에서는 이러한 탐지기술을 우회하기 위하여 파일 접근을 시도하는 공격 쓰레드와 DLL 삽입 공격 사이의 연관성을 찾기 어렵게 하는 방법으로 탐지기술을 우회하는 공격 기법을 제안한다.

우회 공격은 DLL 삽입 공격 시 다음과 같이 계층화된 새로운 쓰레드를 최소 3개 이상 계층적/순차적으로 생성한다.

- 1 계층 : CreateRemoteThread 함수 호출을 통하여 LoadLibrary 함수를 시행하여 공격용 DLL을 대상 프로세스에 로딩하기 위한 쓰레드를 생성한다. 이렇게 생성된 쓰레드를 Thread[1]로 표시한다.
- 2 계층 : 공격용 DLL의 DllMain 함수에서 CreateThread 함수를 호출하여 대상 프로세스 내에 쓰레드를 생성한다. 이 쓰레드를 Thread[2]로 표시한다.
- 3 계층 : Thread[2]의 동작코드는 충분한 대기 시간 후, 새로운 쓰레드를 생성하도록 구현한다. 이와 같이 Thread[2]가 대상 프로세스 내에 생성한 쓰레드를 Thread[3]로 표시한다.

Thread[n]이 Thread[n+1]을 생성하는 방법으로 하위 계층 쓰레드를 추가로 생성할 수 있다. 단, Thread[n]의 동작코드는 충분한 대기 시간 후, Thread[n+1]을 생성한다. 이 때 대기 시간 설정은 Thread[1]이 소멸된 이후까지로 설정한다. Thread[1]은 [10]에서 제안된 탐지 시스템의 판별 정책에 따라 의심 쓰레드 리스트에 기재된다. 그러므로 해당 쓰레드 정보를 확인하면, Thread[1]의 소멸 여부를 확인할 수 있다.

생성된 쓰레드 중에서 최하위 계층 쓰레드에서 시스템의 파일에 접근을 시도한다. 그림 2는 총 3 계층으로 이뤄진 공격 시나리오를 설명한다. 이와 같은 공격 시나리오에서 쓰레드 생성 시 관찰되는 PID/TID는 표 2와 같다.

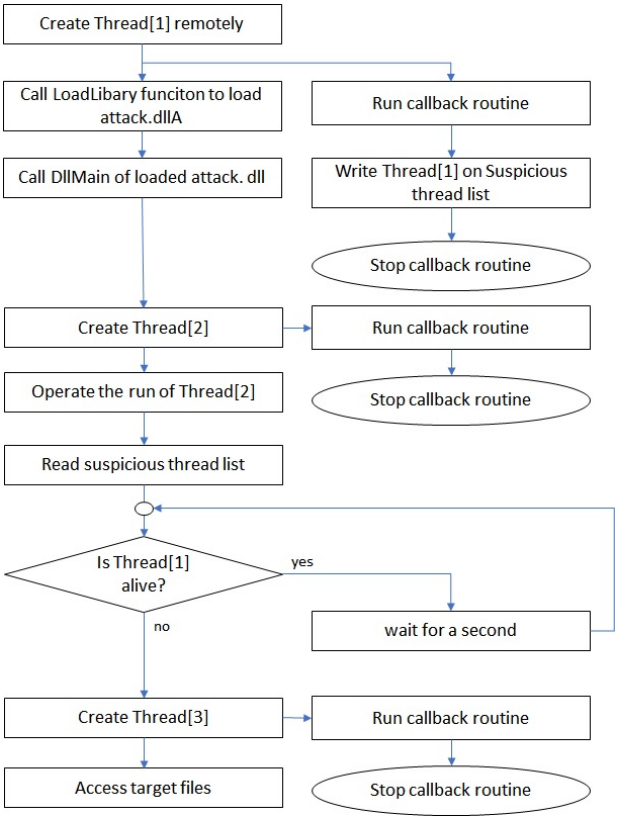


Fig. 2. Suspicious Thread Creation Monitoring.
그림 2. 의심 쓰레드 생성 탐지

Table 2. Hierarchical thread creation result.
표 2. 계층화된 쓰레드 생성 결과

	Thread[1]	Thread[2]	Thread[3]
target ID	PID[B]	PID[B]	PID[B]
	TID[b]	TID[c]	TID[d]
creator ID	PID[A]	PID[B]	PID[B]
	TID[a]	TID[b]	TID[c]

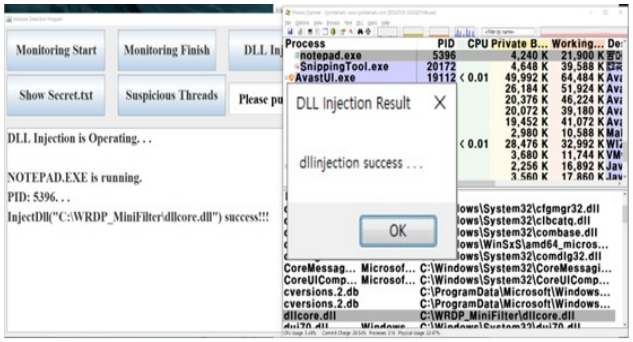
Thread[2]와 Thread[3]는 대상 PID와 생성자 PID가 모두 PID[B]로 동일하다. 그러므로 [10]에서 제안된 의심 쓰레드 탐지 기준에 따라 정상 쓰레드 생성으로 간주되어 콜백 루틴이 호출되더라도 의심 쓰레드 리스트에 쓰레드 정보가 기재되지 않는다. 단, 제안하는 공격 기법에서는 Thread[2]에서 직접 파일 접근을 시도하지 않는다. 그 이유는 Thread[1]과의 연관성을 유추할 가능성이 있기 때문이다. 즉, 의심 쓰레드로 감시 대상이 된 Thread[1]이 생존하는 동안, 대상 PID 내에 생성되는 모든 쓰레드를 의심 쓰레드로 간주하여 파일 접근을 통제할 수 있다[11]. 그러므로 제안된 공격은 Thread[1]이 소멸되고 의심 쓰레드 리스트에서 삭제된 이후에 생성된 하위 쓰레드에서 실제 공격을 수행하도록 설계하였다.

3.2 공격 구현 결과

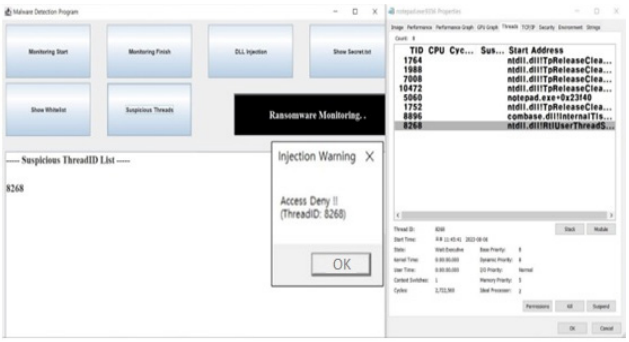
그림 3은 우회공격 구현 결과를 나타낸다. 공격은 파일 접근 권한을 갖는 응용프로그램(노트패드)에 CreateRemoteThread 함수를 이용하여 악성 DLL을 삽입한 후, 응용프로그램이 파일에 접근을 시도하도록 설계되었다.

그림 3-(A)는 [10,11]에서 제안된 화이트리스트 기반 랜섬웨어 솔루션으로 DLL 삽입공격을 탐지하고 그 결과를 의심 쓰레드 목록으로 관리한 결과이다. (A-1)은 공격자가 노트패드에 dllcore.dll을 삽입한 결과이다. (A-2)는 DLL 삽입공격 후, 노트패드 프로세스에서 공격 쓰레드(TID=21860)가 파일(Secret.txt)에 접근하려는 시도를 탐지 하여 파일 접근을 통제하고 불허한 결과이다. 노트패드가 화이트리스트에 기재되어 있음에도 불구하고, 의심 쓰레드 탐지를 통해서 접근을 성공적으로 통제했다.

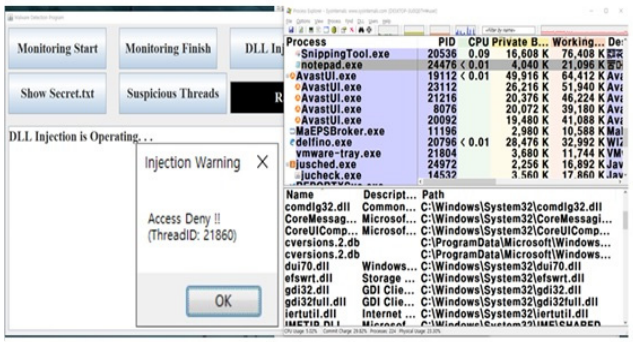
그림 3-(B)는 [11]에서 제안된 의심 쓰레드 탐지 기술을 우회하여 파일에 접근한 결과를 보여준다. (B-1)은 탐지 기술을 적용해서 수집한 의심 쓰레드 리스트를 보여준다. 이 때, CreateRemoteThread 함수 호출로 생성된 쓰레드는 소멸되어 의심 쓰레드 리스트에서 삭제되었고, 노트패드에 삽입된 DLL의 DllMain 함수에서 추가로 생성한 쓰레드(TID=8268)가 의심 쓰레드 목록에 기재된 상태를 보여준다. 해당 의심 쓰레드가 파일에 접근을 시도한 결과, 파일 접근은 불허된다. (B-2)는 의심 쓰레드 탐지 시스템절차를 우회하기 위하여 계층적으로 3개의 쓰레드를 생성하고, 최하위 쓰레드가 파일에 접근을 시도한 결과이다. 이 경우에도 (A-2)에서와 같이 CreateRemoteThread 함수를 이용하여 생성한 Thread[1]은 소멸되어 해당 쓰레드의 정보가 의심 목록에서 삭제되었다. 그러나 Thread[1]이 노트패드에 삽입한 DLL의 DllMain 함수가 생성한 Thread[2]의 TID(=1068)이 의심 쓰레드 리스트에 기재되어 있다. 그러나 실제 파일 접근에 사용된 쓰레드는 Thread[3] (TID=11208)이다. Thread[2]는 대기 시간 후 Thread[3]을 생성하였다. 의심 쓰레드 리스트에는 Thread[3]에 대한 정보가 기재되어 있지 않다. 그러므로 Thread[3]이 파일 접근을 요청하면, 노트패드가 화이트리스트에 기재되어 있고, Thread[3]이 의심 쓰레드 리스트에 기재되어 있지 않기 때문에 Thread[3]의 파일 접근이 최종적으로 허용되었음을 보여준다.



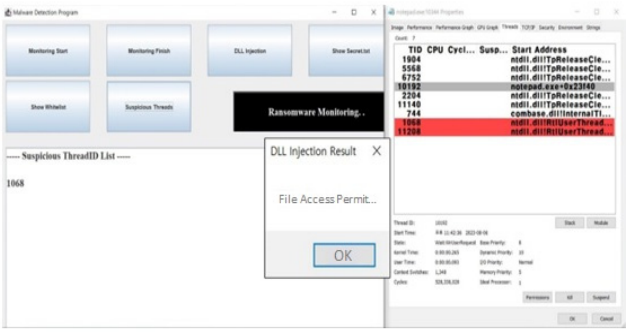
(A-1) DLL Injection



(B-1) Injection Attack Fail



(A-2) Access Deny



(B-2) Bypassing Success

Fig. 3. Implementation Result.
그림 3. 구현 결과

Ⅳ. 결론

스마트 어플리케이션/서비스가 날로 대중화 됨에 따라 이러한 새로운 서비스를 대상으로 하는 악성코드의 공격이 심각한 문제로 대두되고 있다. 특히, 스마트 어플리케이션/서비스의 경우, 악성코드 공격으로 인한 피해가 인명 피해와 연결될 수 있기 때문에, 악성코드 사전 탐지 기술의 중요성이 더욱 부각되고 있다. 블랙리스트 기반의 탐지 기술의 경우, 악성코드의 수집 및 분석 절차를 통해 수집된 악성코드 정보를 바탕으로 탐지를 진행한다. 그러므로 악성코드가 어플리케이션/서비스에 공격을 감행한 후에야 해당 악성코드 수집이 가능하기 때문에 스마트 어플리케이션/서비스에 적용 시, 앞서 언급한 문제가 야기된다.

화이트리스트 기반의 랜섬웨어 탐지 및 대응 기술은 이러한 문제를 해결하기 위하여 제안된 기술이라 할 수 있다. 시스템에 저장된 파일에 접근을 시도하는 어플리케이션을 화이트리스트에 규정하고, 규정되지 않은 어플리케이션 또는 프로세스가 파일에 접근하려고 할 때, 이를 커널 단계에서 통제하고 불허한다. 그러므로 랜섬웨어와 같은 악성코드는 화이트리스트에 규정되어 있지 않기 때문에 파일 접근이 불허되어 공격에 효과적으로 대응할 수 있다.

이러한 화이트리스트 기반의 탐지 및 대응 기술은 사칭공격(False Impersonation Attack)에 취약하다. 특히 윈도우즈 운영체제에서 동작하는 어플리케이션/서비스의 경우, DLL 삽입 공격을 이용한 사칭공격이 가능하다. 이러한 사칭공격으로 화이트리스 기반의 탐지를 우회할 수 있음이 지적되었다. 이러한 취약점을 해결하기 위하여, DLL 삽입공격을 탐지할 수 있는 방안이 제안되었다.

본 논문에서는 DLL 삽입공격을 탐지하는 대응 방안을 우회할 수 있는 새로운 DLL 삽입공격을 제안한다. 제안된 공격 방식은 윈도우즈 운영체제가 쓰레드 생성 시, 부모 프로세스 정보는 확인할 수 있지만, 부모 쓰레드에 대한 정보를 확인할 수 없다는 단점을 악용한다. 이를 위하여 DLL 삽입공격 시, 계층적으로 쓰레드를 생성한 후, 최하위 계층 쓰레드를 이용하여 파일 접근을 시도함으로써 DLL 삽입공격에 대한 탐지기술을 우회할 수 있음을 보여주고 있다. 그러므로 화이트리스트 기반의 랜섬웨어 대응 기술은 여전히 DLL 삽입공격에 취약하며, 이에 대한 개선이 요구된다.

References

[1] F. Aloul, "Smart grid security: Threats, vulnerabilities and solutions," *International Journal of Smart Grid and Clean Energy,* vol.1, no.1, pp.1-6, 2012. DOI: 10.12720/sgce.1.1.1-6

[2] K. Toh, "Security for smart cities," IET Smart Cities, vol.2, no.2, pp.95-104, 2020.

[3] N. Tariq, "Blockchain and smart healthcare security: a survey," Procedia Computer Science, vol.175, pp.615-620, 2020.
DOI: 10.1016/j.procs.2020.07.089

[4] R. Bold, H. Khateeb, and N. Ersotelos, "Reducing False Negatives in Ransomware Detection: A Critical Evaluation of Machine Learning Algorithms," *Appl. Sci. 2022,* Vol.12, no.24, pp.12941, 2022.
DOI: 10.3390/app122412941

[5] S. Osborn, "Mandatory access control and role-based access control revisited," *Proceedings of the second ACM workshop on Role-based access control.* 1997. DOI: 10.1145/266741.266751

[6] Microsoft Docs, "Enable controlled folder access," Online: https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/enable-controlled-folders?view=o365-worldwide

[7] 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

[8] A. Klein and I. Kotler, "Windows Process Injection in 2019," *Black Hat USA 2019,* 2019. Online: https://i.blackhat.com/USA-19/ Thursday/us-19-Kotler-Process-InjectionTechniques-Gotta-Catch-Them-All-wp.pdf

[9] L. Abrams, "Windows 10 Ransomware Protection Bypassed Using DLL Injection," Online: https://www.bleepingcomputer.com/news/security/windows-10-ransomware-protection-bypassed-using-dll-injection/

[10] B. Ko, W. Choi, and D. Jeong, "A Study on the Tracking and Blocking of Malicious Actors through Thread-Based Monitoring," *Journal of the Korea Institute of Information Security & Cryptology*, vol.30, no.1, pp.75-86, 2020.
DOI: 10.1016/j.jcss.2014.02.005

[11] S. Cheon, G. Choi, and D. Kim, "A Cheating Attack on a Whitelist-based Anti-Ransomware Solution and its Countermeasure," *2023 IEEE International Conference on Consumer Electronics (ICCE),* 2023.
DOI: 10.1109/ICCE56470.2023.10043480

[12] S. Ramachandran, J. Rami, A. Shah, K. Kim, and D Rathod, "Defence against crypto-ransomware families using dynamic binary instrumentation and DLL injection," *International Journal of Electronic Security and Digital Forensics,* vol.15, no.4, pp.424-442, 2023.
DOI: 10.1504/IJESDF.2023.131961

[13] Microsoft Docs, "CreateRemoteThread Function," Online: https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethread.

[14] J. Seong, "A Study on Injection Attacks and Defenses on Microsoft Windows," *Journal of Software Assessment and Valuation,* vol.16, no.2, pp.9-23, 2020. DOI: 10.29056/jsav.2020.12.02s

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.