Anti-Analysis 기법 탐지를 위한 API 후킹 기반의 데이터 흐름 추적 기법

김민호*, 이광렬*, 조해현†, 이정현*

*숭실대학교, † 숭실대학교

API-Hooking based Data Flow Tracking for Anti-Analysis Technique Detection

Minho Kim*, Gwangyeol Lee*, Haehyun Cho†, and Jeong Hyun Yi*

*Soongsil University, † Soongsil University

요 약

안티바이러스 벤더 사에 많은 악성코드가 보고되고 있다. 많은 악성코드를 분석하기 위해 안티바이러스 벤더 사들은 샌드박스 환경 기반의 자동화 분석 시스템을 사용하고 있다. 이에 맞춰 악성코드들은 샌드박스 환경을 탐지하고 분석을 회피하기 위해 분석 방지 기법을 사용하고 있으며 많은 사례들이 발견되고 있다. 분석 방지 기법을 사용하는 악성코드가 분석되기까지 시간이 소요됨에 따라 유포되는 시간은 길어지고 있으며 사용자들에게 피해를 유발하고 있다. 따라서, 분석 방지 기법을 사용하는 악성코드를 탐지하는 것은 매우 중요하다.

본 논문에서는 분석 방지 기법을 사용하는 악성코드를 탐지하기 위해 동적 분석 방법을 이용한 API 후킹 기반의 데이터 흐름 추적 기법을 제안한다. 제안 기법을 통해 프로그램에서 사용하는 분석 방지 기법을 정확히 식별하고, 분석 방지 기법에서 사용하는 데이터의 흐름을 추적하여 분석 방지 기법에 의해 프로그램의 실행 흐름이 변경되는 것을 탐지한다.

I. 서론

악성코드는 매일같이 안티바이러스 벤더 사에 보고되고 있으며 보고되는 악성코드의 양은 매우 많다. 모든 악성코드를 분석가들이 직접 분석하기에는 물리적으로 한계가 존재하여, 많은 안티바이러스 벤더 사들은 샌드박스 환경의 자동화된 분석 시스템을 사용하고 있다 [1].

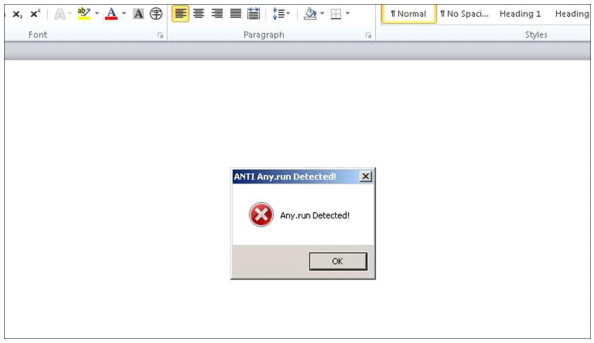
샌드박스를 이용한 자동화 분석 시스템이 보편화됨에 따라 악성코드 제작자들은 분석 방지 기법 중 하나인 샌드박스 환경 탐지 기법을 악성코드에 적용하고 있다. 분석 방지 기법을 사용하는 악성코드는 샌드박스 환경을 탐지하면 실행을 종료하거나 악성 행위를 하지 않고 정상 프로그램으로 위장한다 [2]. 분석 방지 기법으로 인해 악성코드는 샌드박스 기반의 자동화 분석 시스템에서 악성으로 분류되지 않고 정상 프로그램으로 분류되어 보다 오랫동안 유포가 되고 있으며 이로 인해 일반 사용자에게 피해를 유발할 가능성이 있다.

기존의 연구에서는 분석 방지 기법을 사용하는 악성코드를 탐지하기 위해 사용하는 가상화 프로그램의 차이, 분석 환경의 하드웨어 스펙에 차이를 둔 실험 환경에서 악성코드를 실행하여 호출되는 시스템 콜 비교, 네트워크 활동 로그 등을 비교하여 탐지했다 [3]. 그러나 위와 같은 방법은 악성코드가 사용하는 분석 방지 기법을 정확히 식별할 수 없는 문제와 다양한 분석 환경을 탐지하는 악성코드가 존재하는 경우 오탐의 가능성이 존재한다.

따라서, 본 논문에서는 분석 방지 기법을 사용하는 악성코드 탐지와 악성코드에서 사용하

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2021R1A4A1029650).

† 교신저자: 조해현 (haehyun@ssu.ac.kr)



[그림 1] Any.Run 샌드박스 탐지 악성코드의 샌드박스 탐지 출력 메시지 박스.

는 분석 방지 기법을 식별하기 위해 API 후킹 기반의 데이터 흐름 추적 기법을 제안하고 제안 기법의 유효함을 증명하기 위해 샌드박스 탐지 프로그램에서의 실행 결과를 분석하고 평가한다.

II. 연구 배경

2.1 분석 방지 기법

일반적인 악성코드 분석 방법은 사용자 환경과 별도로 격리된 샌드박스 환경에서 악성코드를 실행하고 실행 결과를 분석한다. 분석한 결과를 토대로 실행된 프로그램이 악성인지를 판별한다. 이 과정에서 악성코드는 샌드박스를 기반으로 구현된 분석 환경을 탐지하는 분석 방지 기법을 사용한다. 분석 방지 기법을 사용하는 악성코드는 분석 환경 탐지 시 프로그램의 실행을 종료하거나 악성 행위를 하지 않고 정상 프로그램으로 위장하여 탐지를 회피한다.

[그림 1]은 'Any.Run'이라는 악성코드 분석 목적의 샌드박스 서비스를 탐지하는 악성코드가 샌드박스 환경 탐지 메시지를 출력하는 모습을 보여준다 [2]. 이 악성코드는 일반적인 사용자 환경과 가상 머신에서 악성 행위를 수행하지만 'Any.Run' 샌드박스를 대상으로 실행 환경을 탐지하고 탐지 메시지와 함께 종료하는 것을 볼 수 있다. 이를 통해, 악성코드는 악성코드 분석 목적의 샌드박스를 탐지하여 악성 프로그램으로 분석되지 않기 위해 분석 방지 기법을 사용하는 것을 알 수 있다.

2.2 정적 분석 방법

프로그램 분석 방법으로는 정적 분석 방법과 동적 분석 방법이 있다. 정적 분석 방법은 프로그램을 직접 실행하지 않는 상태에서 프로그램 내에 존재하는 코드와 데이터를 분석하여 프로그램의 행위를 파악한다. 정적 분석 방법으로는 디스어셈블된 실행 명령어 패턴 분석, 데이터 시그니처 분석, 콜 그래프 분석, CFG(Control Flow Graph) 분석과 같이 다양한 방법이 있다. 이러한 정적 분석 방법은 실행 파일 자체에 존재하는 정보를 기반으로 분석하기 때문에 실행 파일 자체를 변형하여 보호하는 패킹, 데이터 암호화, 난독화와 같은 보호 기법에 의해 분석 자체가 불가능한 한계점이 있다.

실제로 VirusShare [4]에서 제공하는 악성코드 샘플(2017-2020)을 대상으로 분석한 결과, [표 1]과 같이 약 30%의 악성코드에서 패킹 기법이 적용된 것을 확인했다.

[표 1] 악성코드 샘플 패킹 탐지 결과.

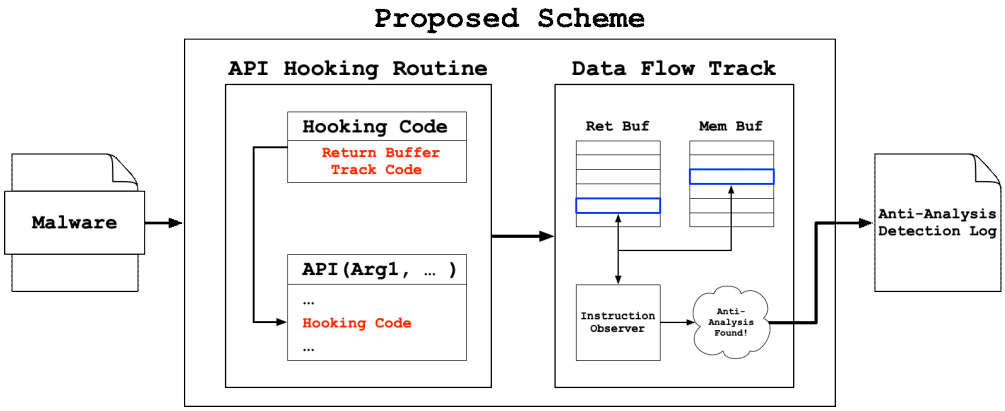
Year	Total	Packed	Ratio(%)
2017	349,256	121,063	34.66
2018	119,952	34,173	28.49
2019	120,942	35,368	29.24
2020	173,785	55,895	32.16

2.3 동적 분석 방법

동적 분석 방법은 정적 분석 방법과 달리 프로그램을 직접 실행하여 얻을 수 있는 정보를 기반으로 프로그램의 행위를 분석하는 방법이다. 널리 사용되는 동적 분석 프레임워크는 DBI(Dynamic Binary Instrumentation)로, PIN[5] 등이 사용되고 있다.

DBI를 이용한 동적 분석 방법은 프로그램의 실행에 필요한 값과 환경 정보에 대해 알지 못한 상태에서 실행된 결과를 분석하기 때문에 실제 프로그래머가 의도한 실행 결과와 다른 결과로 인해 분석 과정에서 오탐이 발생할 수 있다.

앞서 언급했듯이 정적 분석 방법은 패킹, 암호화, 난독화된 프로그램에 대한 분석이 불가능한 한계점을 가지고 있다. 동적 분석 방법은 패



[그림 2] API 후킹 기반의 데이터 흐름 추적 기법 개요.

킹, 난독화 등의 보호 기법이 적용된 프로그램에 대한 분석이 가능하기 때문에 DBI 프레임워크를 이용한 언패킹, 역난독화 도구로 널리 사용되고 있다.

III. 제안 기법

앞서 분석 방지 기법을 사용하여 악성코드 분석 목적의 샌드박스 환경을 탐지하는 악성코드와 각각의 프로그램 분석 방법에 따른 장·단점을 설명했다. 따라서 본 논문에서는 동적 분석 방법 기반의 분석 방지 기법 탐지를 위한 API 후킹 기반의 데이터 흐름 추적 기법을 제안한다. 제안 기법은 [그림 2]와 같이 크게 분석 방지 기법으로 의심되는 API에 후킹 코드 삽입, 데이터 흐름 추적, 조건 분기 식별 과정을 거쳐 분석 방지 기법을 탐지한다.

3.1 API 후킹 코드 삽입

악성코드는 분석 방지 기법에 사용되는 API의 실행 결과에 따라 악성 행위를 할지 판단한다. 이를 위해, 악성코드는 API가 실행된 후 API의 실행 결과가 저장되는 버퍼의 값을 비교한다. 악성코드가 버퍼의 값을 비교하는 과정을 식별하기 위해 위 과정에서는 API 실행 결과가 저장되는 버퍼의 위치를 파악하는 후킹 코드를 삽입한다.

3.2 데이터 흐름 추적

악성코드가 버퍼를 비교하는 과정에서 다른 메모리 주소로 백업한 데이터를 사용하거나 스택에 버퍼의 주소를 기록 후 참조하는 방식 등과 같이 다양한 방식을 사용할 수 있다. 분석 방지 기법 탐지를 위해 버퍼와 관련된 모든 실행 명령어를 분석하여 데이터가 메모리에 저장되는 과정에 대한 추적이 필요하다.

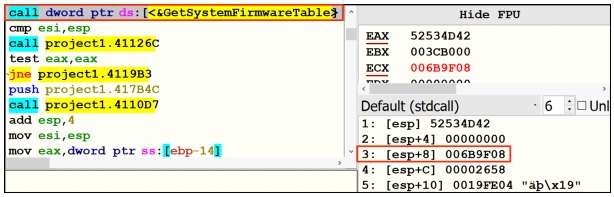
3.3 조건 분기 식별

데이터 흐름 추적을 통해 API의 실행 결과가 저장된 데이터의 흐름을 분석하고 실행 결과에 영향을 끼칠 수 있는 데이터의 위치를 모두 추적했다. 조건 분기 식별 과정은 추적한 데이터를 비교하는 과정과 비교 결과가 영향을 끼치는 조건 분기 명령어를 식별한다. 위 과정에서 조건 분기 명령어를 찾게 되면 이를 분석 방지 기법에 의해 실행 결과에 영향을 끼치는 것으로 판단하고 API를 분석 방지 기법에 사용된 것으로 분류한다.

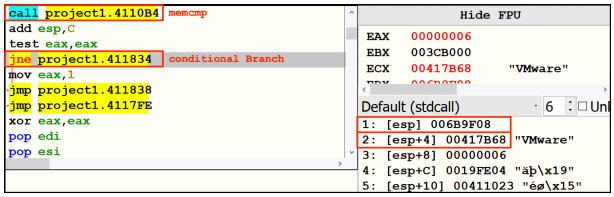
IV. 제안 기법 검증

본 논문에서 제안한 API 후킹 기반의 데이터 흐름 추적 기법에 대해 평가한다. 제안한 기법을 검증하기 위해 샌드박스 환경을 탐지하는 프로그램을 대상으로 직접 분석한 결과와 제안 기법에 의해 분석된 결과를 비교했다.

[그림 3]과 [그림 4]는 샌드박스 탐지 프로그램을 직접 분석한 결과로 [그림 3]을 통해 분석 방지 기법과 관련된 API의 결과가 저장되는 메모리 주소가 0x6B9F08인 것을 확인한다. [그림 4]는 미리 확인한 메모리 주소에 저장된 값과



[그림 3] 샌드박스 탐지 프로그램 디버깅 결과 (버퍼 주소 확인).



[그림 4] 샌드박스 탐지 프로그램 디버깅 결과 (버퍼 비교와 조건 분기).

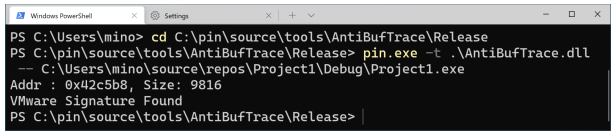
샌드박스 탐지에 사용되는 시그니처 중 하나인 “VMware” 문자열과 비교 후 조건 분기 명령어(JNE 0x411834)를 호출하는 것을 알 수 있다.

[그림 5]와 [그림 6]은 본 논문에서 제안한 기법의 실행 결과 모습이다. [그림 5]를 통해 샌드박스 탐지 API의 결과가 저장되는 메모리 주소가 0x42C5B8인 것을 알 수 있다. [그림 6]에서는 미리 확인한 메모리 주소와 문자열을 비교를 식별하고 조건 분기 명령어(JNZ 0x411834)까지 식별한 것을 알 수 있다.

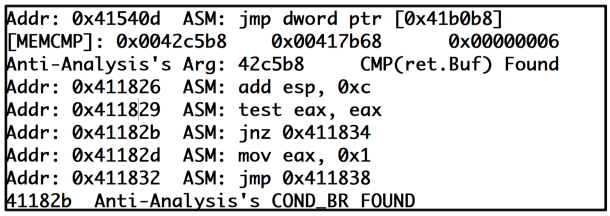
비교 검증을 통해 본 논문에서 제안한 기법이 프로그램에서 사용하는 분석 방지 기법과 관련된 데이터 흐름을 추적하고 추적한 결과를 토대로 분석 방지 기법에 의해 프로그램 실행 흐름에 영향을 미치는 명령어를 찾을 수 있음을 확인했다.

V. 결론

본 논문에서는 분석 방지 기법을 사용하여 악성코드 분석 목적의 샌드박스 환경을 탐지하는 악성코드를 탐지하기 위해 동적 분석 방법을 이용하여 API 후킹 기반의 데이터 흐름 추적 기법을 제안했다. 샌드박스 탐지 프로그램에 대한 비교 분석을 통해 제안 기법이 유효함을 증명했다. 추후 연구를 통해 실제 악성코드를 대상으로 분석 방지 기법을 사용하는 악성코드에 대한 탐지 및 동향 분석을 진행할 예정이다.



[그림 5] 제안 기법 실행 결과 분석 (버퍼 주소 확인).



[그림 6] 제안 기법 실행 로그 분석 (버퍼 비교와 조건 분기).

[참고문헌]

[1] eSecurityPlanet, "Sandboxing: Advanced Malware Analysis in 2021", https://www.esecurityplanet.com/endpoint/sandboxing-advanced-malware-analysis-in-2021/, Accessed November 5, 2021.

[2] BLEEPINGCOMPUTER, "Malware adds online sandbox detection to evade analysis", https://www.bleepingcomputer.com/news/security/malware-adds-online-sandbox-detection-to-evade-analysis/, Accessed November 5, 2021.

[3] D. Kirat, G. Vigna, and C. Kruegel. Barecloud: Bare-metal analysis-based evasive malware detection. In 23rd {USENIX} Security Symposium ({USENIX} Security 14), pages 287 - 301, 2014.

[4] VirusShare, "VirusShare.com", https://virusshare.com/, Accessed Novembr 5, 2021.

[5] Intel, "Pin", https://software.intel.com/content/www/us/en/develop/articles/pin-a-dynamic-binary-instrumentation-tool.html/, Accessed November 5, 2021.