정 보 보 호 학 회 지
제24권 제1호, 2014. 2

악성코드 탐지를 위한 물리 메모리 분석 기술

강 영 복*, 황 현 욱**, 김 기 범***, 손 기 욱***, 노 봉 남****

요 약

악성코드는 다양해진 감염 경로를 통해 쉽게 노출될 수 있으며, 개인정보의 유출뿐만 아니라 봇넷을 이용한 DDoS 공격과 지능화된 APT 공격 등을 통해 심각한 보안 위협을 발생시키고 있다. 최근 악성코드들은 실행 후에는 메모리에서만 동작하는 방식으로 파일로 존재하지 않기 때문에 기존의 악성코드 탐지 기법으로 이를 찾기가 쉽지 않다. 이를 극복하고자 최근에는 물리 메모리 덤프를 포함하여 악성코드 분석 및 탐지 연구가 활발하게 진행되고 있다.

본 논문에서는 윈도우 시스템의 물리 메인 메모리에서 악성코드 탐지 기술에 대해 설명하고, 기존 개발된 물리 메모리 악성코드 탐지 도구에 대한 분석을 수행하여 도구별 악성코드 탐지 기능에 대한 특징을 설명한다. 물리 메모리 악성코드 탐지 도구의 분석 결과를 통해 기존 물리 메모리 악성코드 탐지 기술의 한계점을 제시하고, 향후 정확하고 효율적인 물리 메모리 악성코드 탐지의 기반 연구로 활용하고자 한다.

Ⅰ. 서 론

현재 악성코드는 사회적으로 많은 보안 이슈를 발생시키고 있으며, 최근 다양해진 감염 경로를 통해 악성코드에 쉽게 노출될 수 있다. 취약한 시스템에 악성코드가 감염되면 개인정보 유출뿐만 아니라 금융정보 탈취, 봇넷을 이용한 DDoS 공격, 그리고 지능화된 APT 공격 등의 보안 위협이 발생할 수 있다 이러한 악성코드를 탐지하기 위해 기존에는 하드디스크 분석을 기반으로 실행 파일 정보 추출 및 레지스트리 분석 등을 통해 탐지를 수행하였다. 하드디스크 기반 악성코드 탐지 방법에서는 실행되지 않는 파일정보를 분석하기 때문에 패킹된 악성코드나 메모리 상주형 악성코드 탐지에 많은 어려움이 있다. 기존의 하드디스크 기반의 악성코드 탐지의 문제점을 해결하고자 최근에는 물리 메모리 덤프 데이터를 이용한 악성코드 탐지 연구가 진행되고 있다. 메모리 포렌식을 이용하면 언패킹, 루트킷 탐지, 리버스 엔지니어링 등에 많은 도움이 된다[1]. 또한 메모리 포렌식을 이용하면 메모리 덤프 시점에서 실행중인 프로세스 정보, 네트워크 정보, 레지스트리 정보 등 다양한 정보를 활용할 수 있다는 장점이 있다[2]. 최근엔 컴퓨터가 널리 보급되어 이용되어지고, 대용량 저장장치도 널리 보급되었기 때문에 기존 하드디스크 분석 방법과 유사하게 손쉽게 메모리 덤프이미지를 획득하고 분석이 가능하다.

본 논문에서는 윈도우 물리 메모리 분석 절차를 기술하고, 현재까지 연구된 메모리에서의 악성코드 탐지 방법을 설명한다. 또한 다양한 물리 메모리 포렌식 도구에서 제공하고 있는 악성코드 탐지 기능을 비교하여 그 특징을 분석하였다.

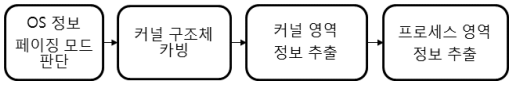
본 논문의 구성은 2장에서는 현재 사용되고 있는 물리 메모리 악성코드 탐지 기술을 설명하고, 3장에서는 기존에 개발된 다양한 물리 메모리 악성코드 탐지 도구에 대한 분석을 설명하고, 마지막으로 4장에서는 결론 및 향후 연구에 대한 내용을 기술한다.

* 대표저자: 전남대학교 정보보안협동과정 석사과정(k-dupe@nate.com)
** 교신저자: ETRI 부설연구소(hhu@ensec.re.kr)
*** ETRI 부설연구소(kibom@ensec.re.kr, kiwook@ensec.re.kr)
**** 전남대학교 전자컴퓨터공학부 교수(bbong@jnu.ac.kr)

Ⅱ. 물리 메모리 악성코드 탐지 연구

2.1 물리 메모리에서의 커널 정보 추출 방법

윈도우 물리 메모리에서 악성코드를 탐지하기 위해서는 먼저 물리 메모리에 저장된 시스템 정보를 수집해야한다. 물리 메모리에서 시스템 정보를 수집하는 과정은 그림 1과 같다.



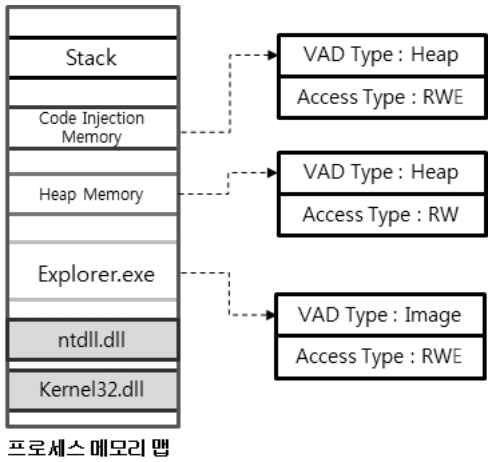
〔그림 1〕 물리 메모리 시스템 정보 수집 절차

총 4가지 단계에 따라 분석 대상 물리 메모리에서 시스템정보를 수집할 수 있다. 1단계 과정에서는 분석 대상 물리 메모리의 운영 체제 정보 및 페이지 모드를 판단한다. 분석 대상 물리 메모리의 운영체제 정보 및 페이징 모드를 파악하기 위해서는 KDBG 구조체 정보를 카빙[3] 및 카빙된 구조체 크기 및 내부 플래그 값을 분석한다[4]. 2단계 과정에서는 OS 버전에 따른 커널 구조체를 메모리에서 카빙하는 것이다. 분석된 OS 정보를 바탕으로 버전별 메모리에 저장된 커널 구조체를 카빙을 수행한다. 카빙하는 주요 커널 구조체는 실행중인 프로세스를 관리하는 구조체와 관련된 카빙가능한 모든 구조체를 카빙 한다. 3단계 과정에서는 카빙된 커널 구조체를 이용하여 커널 메모리 영역에 저장된 정보를 추출한다. 커널 드라이버, 커널 모듈, 네트워크 정보 등을 추출할 수 있다. 4단계에서는 유저 메모리영역에 저장된 주요 프로세스를 정보를 추출한다. 유저영역 프로세스에서 실행중인 쓰레드 정보, 로드 모듈 정보, 소켓 정보 등을 추출한다.

2.2 프로세스 메모리 영역 특징을 이용한 악성코드 탐지

메모리 특징을 이용한 악성코드 탐지 방법에서는 프로세스에서 소유하고 있는 메모리 영역의 플래그 특징을 이용하여 악성 행위를 수행하는 메모리 영역을 탐지하는 방법이다.

물리 메모리 분석환경에서 프로세스 로드 메모리 영역을 검사하기 위해서는 일반적으로 VAD[5] 구조체 정보를 이용한다. VAD 구조체 는 해당 메모리 영역의 타입과 권한 정보를 포함하고 있다. 메모리 영역의 타입은 힙(Heap)과 이미지(Image) 타입으로 구분되며, 이미지 타입의 경우, 실제 파일 데이터가 로드되어 있다.



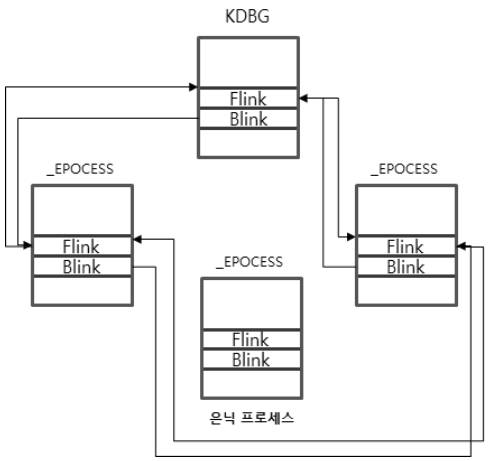
〔그림 2〕 코드인젝션 된 프로세스 메모리 맵 구조

그림 2는 코드인젝션[6] 된 프로세스의 메모리 맵 구조이다. 코드인젝션 된 메모리 영역은 동적으로 할당된 메모리 영역이면서 실행 가능한 영역이므로 힙 메모리 공간에 실행 권한을 가지는 특징을 가지는 것을 알 수 있다. VAD 구조체 분석을 이용하여 물리 메모리에서 코드인젝션이 수행되는 메모리 영역을 탐지할 수 있다.

2.3 커널 구조체 연결 분석을 통한 악성코드 탐지 방법

윈도우 시스템에서는 다수의 정보를 가지는 구조체의 경우 일반적으로 이중 링크드 리스트를 이용하여 복수의 구조체를 관리한다. 악성코드에서는 자신의 바이너리를 은닉하는 목적으로 커널 구조체에서 사용하는 링크드 리스트의 연결을 끊는 작업을 수행한다. 대표적으로 커널에서 사용하는 프로세스 구조체와 모듈 구조체는 각 구조체가 링크드 리스트로 연결되어있다[6].

그림 3과 같이 악성코드에서 프로세스 은닉을 기술을 사용하는 경우 은닉 프로세스의 구조체는 정상 구조체와 링크드 연결이 끊어져있는 특징을 가지고 있다. 물리 메모리 분석 환경에서 링크드 리스트로 연결된 프로세스 커널 구조체 정보와 카빙에서 추출한 커널 구조체 정보와 비교 분석을 수행하면 은닉된 프로세스를 탐지할 수 있다. 은닉 프로세스 탐지 방법과 동일하게 링크

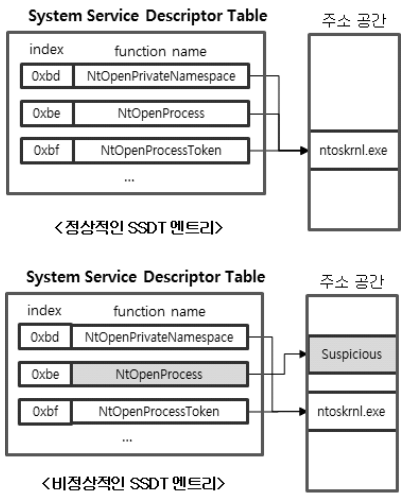


〔그림 3〕 링크드 연결이 되지 않는 은닉 프로세스 구조체

드 리스트 연결 정보와 커널 구조체 카빙 결과와 비교 분석을 수행하면 은닉 모듈 정보도 탐지할 수 있다.

2.4 함수 엔트리 포인터 체크를 통한 악성코드 탐지 방법

윈도우 시스템의 일반적인 모듈에서는 외부 모듈 함수 호출을 제외한 일반적인 호출인 경우 자신의 모듈 데이터가 로드된 메모리 영역에 저장된 코드를 호출한다. 물리 메모리 분석과정에서는 점검 대상 함수 엔트리 포인터와 해당 모듈이 로드된 크기 정보를 비교하여 후킹 행위를 탐지 할 수 있다. 그림 4는 SSDT 후킹이 된 함수 테이블 정보와 정상적인 SSDT 테이블을 비교한



〔그림 4〕 SSDT후킹 후킹 테이블 비교

것이다. 그림 4와 같이 일반적으로 SSDT 후킹이 발생하는 경우 후킹된 함수의 엔트리 포인트는 후킹 모듈의 메모리 영역 값을 가지고 있다[8]. SSDT 후킹 탐지 방법과 동일하게 함수 엔트리 포인터 주소와 메모리에 로드된 정상 모듈의 영역을 비교분석을 수행하면, IAT, IRP 후킹 탐지를 수행할 수 있다.

Ⅲ. 물리 메모리 악성코드 탐지 도구 비교

현재 개발된 물리 메모리 포렌식 도구의 현황은 표 1과 같다.

〔표 1〕 물리 메모리 포렌식 도구 현황

도구이름	지원 운영체제	공개 여부	악성코드 탐지기능	기타 지원 기능
Volatility	Linux/MAC/ Window	공개	제공	Plugin
Responder	Window	상용	제공 (Digital DNA)	Report
RedLine	Window	공개	제공	Report

상용 도구인 Responder를 제외한 나머지 도구는 현재 인터넷에 공개되어있다. Volatility 도구의 경우 현재 오픈 소스로 개발되고 있으며, 윈도우, 맥, 리눅스 메모리 분석 기능을 제공하고 있다. Responder, RedLine 물리 메모리 포렌식 도구에서는 직관적인 GUI를 제공하여 손쉽게 데이터 수집 및 분석을 수행할 수 있는 기능을 제공하고 있다. 반면 Volatility에서는 콘솔 기반으로 제작되어 데이터 수집 및 분석에 있어서 다양한 명령어를 사용자가 직접 입력하여 실행하도록 개발되었다.

현재 개발된 물리 메모리 포렌식 도구에서는 기본적으로 악성코드 탐지기능을 제공하고 있는 것을 알 수 있다. 상용도구인 Responder에서는 다른 물리 메모리 포렌식 도구와 다르게 악성코드 탐지 기능에서 시그니처 기반의 Digital DNA 분석 기능을 제공하고 있다.

3.1 물리 메모리 포렌식 도구 악성코드 탐지 기능 비교

Volatility[9], Responder[10], RedLine[11] 에서 제공하는 물리 메모리 악성코드 탐지 기능 비교는 표 2와 같다. 총 6가지의 악성 행위에 대해 악성 행위 탐지 결

과를 분석하였다. 물리 메모리 악성코드 탐지 기능 비교 과정에서 Responder, RedLine 도구는 오픈소스가 아니기 때문에 특정 기존 분석된 악성코드 또는 악성행위를 수행하는 샘플 코드를 이용하여 실험 테스트를 수행하였다.

Volatility 악성코드 탐지 기능 비교 결과 실험에서 비교한 모든 악성 행위에 대해서 탐지할 수 있는 것을 알 수 있었다. 하지만 Volatility 도구에서는 콘솔 기반으로 분석 결과가 출력되기 때문에 분석 결과에 대해 분석자가 직접 결과를 분석 하고 악성 행위 유무를 판단해야 하는 특징이 있다.

Responder 도구의 코드인젝션 탐지, IAT 후킹 탐지 실험 결과 악성코드 종류 및 실험 테스트 방법에 따라 탐지 결과가 다르게 나오는 것을 확인할 수 있었다. Responder 도구에서는 기존에 안티 바이러스 제품에서 탐지되는 악성코드의 행위에 대해서는 완벽하게 탐지한다. 하지만 신규 악성코드에서 사용하는 악성행위 및 악성 행위를 수행하는 테스트 코드에 대해서는 완벽하게 탐지를 수행하지 못하는 것을 알 수 있었다.

Responder 도구에서는 기존에 분석된 다양한 악성코드를 이용하여 개발한 Digital DNA 통합 분석 기능을 제공하고 있다. 위 기능에서는 분석 대상 메모리에서 추출한 실행 파일(EXE), 모듈 파일(DLL), 메모리 파일을 대상으로 위협 점수를 출력한다. Digital DNA로 출력되는 위협 점수를 이용하여 분석 대상 메모리에서 의심되는 악성행위 바이너리를 추출할 수 있다.

RedLine 도구에서는 Responder 도구의 실험 결과와 유사하게 코드인젝션 및 은닉 모듈 탐지에 대해 악성코드 종류 및 실험 테스트 방법에 따라 탐지 결과가 다르게 나오는 것을 확인할 수 있었다. RedLine 도구에서는 실행중인 프로세스를 중심으로 악성행위 탐지 기능을 제공한다. 은닉 모듈 탐지 부분의 기능에서는 Volatility와 다르게 은닉 모듈 또는 의심 모듈을 탐지를 위한 모듈 로드 순서를 확인할 수 있는 기능을 추가적으로 제공하는 것을 알 수 있었다.

RedLine을 이용한 정확한 은닉 모듈 탐지를 수행하기 위해서는 분석자의 추가적인 시스템 정보 분석 및 모듈 로드 순서 분석이 요구된다. RedLine 도구에서는 프로세스 단위 악성행위 의심 정보를 출력하는 MRI(Malware Risk Index)[12][13]통합 분석 기능을 제공하고 있다. MRI에서는 실행 프로세스와 관련된 모든 악성행위, 프로세스에서 로드된 모듈 파일, 윈도우 시스템 파일 시그니처 등 다양한 정보를 통합하여 위협 점수화를 수행한다. MRI기능으로 출력되는 결과를 이용하여 분석 대상 메모리에서 의심되는 프로세스 정보를 추출할 수 있다.

3.2 비교 결과

현재 개발된 물리 메모리 포렌식 도구에서 제공하는 악성코드 탐지 기능을 비교 분석을 수행한 결과 은닉 프로세스 탐지, SSDT 후킹 탐지, IRP 후킹은 탐지는 정확하게 수행할 수 있었다. 하지만 코드인젝션 탐지, 은닉 모듈 탐지, IAT 후킹 탐지는 분석 도구만을 이용한 정확한 탐지가 어렵다는 것을 알 수 있었다. 또한 기존의 물리 메모리에서는 개별적인 악성코드 방법의 문

〔표 2〕 물리 메모리 포렌식 도구 악성코드 탐지 기능 비교

도구이름	메모리 영역 탐지	커널 구조체 기반 악성행위 탐지		함수엔트리 포인트 기반 탐지			통합 분석 기능
	코드인젝션 탐지	은닉 프로세스 탐지	은닉 모듈 탐지	SSDT 후킹 탐지	IAT 후킹 탐지	IRP 후킹 탐지	
Volatility	□	□	□	□	□	□	제공하지 않음
Responder	△	○	X	○	△	○	제공 (Digital DNA)
RedLine	△	○	□	○	X	○	제공 (MRI)

※의미 ○ : 악성행위 탐지 가능
□ : 악성행위를 탐지하기 위한 사용자의 수동적인 분석 및 판단이 요구됨
△ : 악성코드 종류에 따라 탐지 결과가 달라짐
X : 악성 행위 탐지 불가

제점을 해결하고자 Responder, RedLine에서는 악성행위 코드 시그니처화 및 스코링 방식을 이용한 통합 분석 기능을 제공하고 있는 것을 알 수 있다. Responder에서는 시그니처 기반의 Digital DNA를 이용하였기 때문에 신규 악성코드 탐지에 있어 많은 문제점이 있다. RedLine에서는 프로세스 단위 스코링 방식을 적용하기 때문에 타 프로세스에 메모리만 로드하여 상주하는 경우 또는 커널 드라이버를 이용하여 악성행위를 하는 방식에 있어 탐지가 어려울 수 있다는 문제점이 있다. Volatility 도구에서는 악성행위 탐지 결과에 관한 시스템 커널 정보를 콘솔을 통해서 출력되기 때문에 정확한 악성행위 판단을 위해서는 출력 결과에 대한 사용자 분석이 요구된다. 또한 Volatility에서는 통합 분석 기능을 지원하지 않기 때문에 메모리 이미지가 크거나 분석 대상 이미지가 많은 경우 악성코드 탐지에 있어 비효율적이라고 할 수 있다.

Ⅳ. 결론 및 향후 연구

본 논문에서는 물리 메모리 악성코드 분석을 위한 악성행위 탐지 기술 방법에 대해 설명하였다. 또한 물리 메모리 포렌식 도구에서 제공하는 메모리 악성코드 탐지 기능에 대한 비교 및 분석을 수행 하였다. 현재 물리 메모리에서의 악성코드 탐지를 위해 코드인젝션 탐지, 비정상 코드에서 사용하는 커널 구조체 분석 등 다양한 연구가 수행되고 있다. 하지만 기존의 연구에서는 개별적인 악성행위에 대한 탐지를 기초로 악성코드를 탐지하기 때문에 코드 재분석 및 악성행위 탐지 결과에 대한 연관 분석을 수행하기가 어렵다는 문제점이 있다.

향후 연구로는 본 논문에서 조사한 물리 메모리 악성코드 탐지 도구 기능의 개선점을 기반으로 메모리 기반의 악성코드 분석 및 연관된 악성행위 분석을 수행하기 위한 프레임워크를 개발할 예정이다

참고문헌

[1] Michael Hale Ligh, Steven Adair, Blake Hartstein, Mathew Richard, "Malware Analyst's Cookbook and DVD", 에이콘 출판사, pp.715-749, May 2012.

[2] Mariusz Burdach, “Finding Digital Evidence In Physical Memory”, Black Hat USA, Feb 2006.

[3] Volatility, https://code.google.com/p/volatility/wiki/CommandReference23#kdbgscan, 2013.

[4] James Okolica, Gilbert L. Peterson, "Windows operating systems agnostic memory analysis", DIGITAL INVESTIGATION 7, pp.48-56, May 2010.

[5] Brendan Dolan-Gavitt, "The VAD tree: A process-eye view of physical memory", DIGITAL INVESTIGATION S4, pp.62-64, Jun 2007.

[6] Raashid Bhat, "Code Injectin on Window", Strudent Computer Security 2BE, Sep 2011.

[7] Elia Floio, "When Malware Meets Rootkits", Symantect Security Response, 2005.

[8] Muteb Alzaidi, Ahmed Alasiri, "The Study of SSDT Hook through Comparative Analysis between Live Response and Memory Image", Master of Information Systems Security Research 2012 Convocation, 2013.

[9] Volatility, https://code.google.com/p/volatility/, 2013.

[10] HBGary, http://hbgary.com/products/responder_pro, 2013.

[11] MANDIANT Redline, http://www.mandiant.com/resources/download/redline, 2013.

[12] MANDIANT Redline, "Redline User Guide", MANDIANT, 2012.

[13] Michael J. Graven, "Finding Evil In Memory", Ninjacon 11, Jun 2011.

〈저자소개〉

강 영 복 (YoungBok Kang)
학생회원

2012년 2월: 전남대학교 전자컴퓨터공학부(공학사)
2012년 2월~현재: 전남대학교 정보보안협동과정 석사과정
<관심분야> 디지털 포렌식, 악성코드 탐지

사 진

황 현 욱 (Hyunuk Hwang)
정회원

2000년 2월: 조선대학교 정보통신공학과 졸업(공학사)
2002년 2월: 조선대학교 전자공학과 졸업(공학석사)
2004년 8월: 전남대학교 정보보호협동과정 졸업(이학박사)
2004년 9월~현재: ETRI 부설연구소 선임연구원
<관심분야> 디지털 포렌식, 악성코드, 사이버보안

사 진

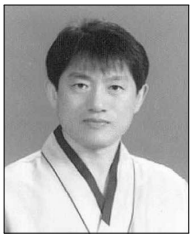
김 기 범 (Kibom Kim)
정회원

1994년 2월: 제주대학교 정보공학과 졸업(공학사)
1996년 8월: 고려대학교 전산과학과 졸업(이학석사)
2001년 2월: 고려대학교 전산과학과 졸업(이학박사)
2004년 1월~2004년 7월: (주) 이씨오 개발부장
2004년 8월~현재: ETRI 부설연구소 선임연구원
<관심분야> 디지털 포렌식, 사이버보안, 정보보호

사 진

손 기 욱 (Kiwook Sohn)
정회원

1990년 2월: 성균관대학교 정보공학과 학사
1992년 2월: 성균관대학교 정보공학과 석사
2002년 8월: 성균관대학교 전기전자컴퓨터공학과 박사
1992년 1월~1999년 12월: 한국전자통신연구원 선임연구원
2000년 1월~현재: ETRI 부설연구소 책임연구원/본부장
<관심분야> 디지털 포렌식, 사이버보안, 정보보호



노 봉 남 (Bongnam Noh)
종신회원

1987년: 전남대학교 수학교육과 (이학사)
1982년: KAIST 전산학과 (이학석사)
1994년: 전북대학교 전산과 (이학박사)
1983년~현재: 전남대학교 전자컴퓨터공학부 교수
2000년~현재: 시스템보안연구센터 소장
<관심분야> 디지털 포렌식, 시스템 및 네트워크 보안, 정보사회와 사이버 윤리