Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.2, Apr. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.2.431

프로세스 가상 메모리 데이터 유사성을 이용한 프로세스 할로윙 공격 탐지*

임 수 민,[1†] 임 을 규[2‡]
[1]한양대학교 컴퓨터소프트웨어학과, [2]한양대학교 컴퓨터공학부

Proposal of Process Hollowing Attack Detection Using Process Virtual Memory Data Similarity*

Su Min Lim,[1†] Eul Gyu Im[2‡]
[1]Department of Computer and Software, Hanyang University,
[2]Division of Computer Science and Engineering, Hanyang University

요 약

파일리스 악성코드는 악성 행위를 수행할 페이로드의 흔적을 은닉하기 위해 메모리 주입 공격을 이용한다. 메모리 주입 공격 중 프로세스 할로윙이라는 이름의 공격은 시스템 프로세스 등을 일시정지 상태로 생성시킨 다음, 해당 프로세스에 악성 페이로드를 주입시켜 정상 프로세스인 것처럼 위장해 악성행위를 수행하는 방법이다.

본 논문은 프로세스 할로윙 공격이 발생했을 경우, 악성 행위 실제 수행 여부와 상관없이 메모리 주입 여부를 검출할 수 있는 방법을 제안한다. 메모리 주입이 의심되는 프로세스와 동일한 실행 조건을 갖는 복제 프로세스를 실행시키고, 각 프로세스 가상 메모리 영역에 속해있는 데이터 집합을 퍼지 해시를 이용해 비교한 다음 유사도를 산출한다.

ABSTRACT

Fileless malware uses memory injection attacks to hide traces of payloads to perform malicious works. During the memory injection attack, an attack named “process hollowing” is a method of creating paused benign process like system processes. And then injecting a malicious payload into the benign process allows malicious behavior by pretending to be a normal process.

In this paper, we propose a method to detect the memory injection regardless of whether or not the malicious action is actually performed when a process hollowing attack occurs. The replication process having same execution condition as the process of suspending the memory injection is executed, the data set belonging to each process virtual memory area is compared using the fuzzy hash, and the similarity is calculated.

Keywords: Fileless malware, malware, Process hollowing, malware detection, Process memory, Memory similarity

Received(02. 11. 2019), Accepted(03. 24. 2019)
* 본 논문은 2018년도 한국정보보호학회 동계학술대회에 발표한 우수논문을 개선 및 확장한 것임
* 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.NRF-2016R1A2B4015254).
† 주저자, bingsumin@hanyang.ac.kr
‡ 교신저자, imeg@hanyang.ac.kr(Corresponding author)

I. 서 론

파일리스 악성코드는 악의적인 실행 파일을 파일 시스템에 배치하지 않고 작동시키는 악성코드이다. 악성코드 실행 프로세스에서 특정 단계가 기존의 실행 방법과 다르게 디스크 내에 쓰여지지 않은 상태로 실행되는 경우를 의미한다. 원격 실행을 통해 감염되기도 하고 문서 파일에서 명령 프롬프트 프로세스를 실행시켜 악성 쉘코드를 종속적으로 실행시킬 수 있다. 기존 악성코드의 실행 방법 대비 파일리스의 실행 매커니즘은 고도화되었다고 볼 수 있으며, 이로 인해 파일디스크 및 시그니처 기반의 악성코드 탐지 기술을 통해 파일리스 악성코드를 탐지하는 것은 굉장히 어렵다 [1]. 실제로 Ponemon에 따르면, 2017 년에 파일리스 공격이 전년도 대비 45 % 증가했으며 성공적인 공격의 경우 77 %나 증가한 것으로 나타났다 [2]. 파일리스 형태의 공격은 시간이 지날수록 수와 복잡성이 점차 증가할 것으로 예상되므로 파일리스 악성코드 차단 방법 모색이 시급하다 [3].

파일리스 악성코드는 악성 행위를 수행할 페이로드의 흔적을 은닉하기 위해 메모리 주입 공격을 이용한다 [4]. 고도화된 실행 매커니즘을 통해 자식 프로세스 가상 메모리 공간에 악성 실행 코드를 주입하기도 하고, 기존에 실행되고 있던 합법적인 프로세스 가상 메모리 공간에 악성 실행 코드를 주입하기도 한다. 메모리 주입 공격 중 프로세스 할로윙이라는 이름의 공격은 시스템 프로세스 등을 일시정지 상태로 생성시킨 다음, 해당 프로세스에 악성 페이로드를 주입시켜 정상 프로세스인 것처럼 위장해 악성행위를 수행하는 방법이다 [5]. 또한 악성 페이로드 주입에 희생된 프로세스의 프로세스 환경 블록(Process Environment Block: PEB)의 정보는 악성 페이로드 데이터와 연관된 정보를 띄는 것이 아니라, 주입 공격 전의 정상 프로세스의 정보를 가지게 된다. 따라서 해당 프로세스를 외적으로나 정적으로 검사할 경우, 악성 여부를 탐지하지 못하게 된다.

악성 행위 패턴 등을 이용한 동적 분석을 통해 주입된 악성 페이로드의 존재를 감지할 수도 있지만, 악성코드는 점점 다양하고 복잡한 분석 우회 및 은닉 기술을 사용하기 때문에 악성 행위 기반의 메모리 주입 공격 검출은 불확실하다 [6 , 7]. 또한 메모리 주입 공격 탐지 방법으로 메모리 포렌식 방법이 존재하지만 메모리(Random Access Memory :RAM) 덤프에 필요한 시간이라는 오버헤드가 존재한다는 단점과 감염 즉시 탐지하는 것이 아니라 사후 탐지라는 치명적인 단점이 존재한다 [8].

따라서 본 논문은 프로세스 할로윙 공격이 발생했을 경우, 악성 행위 실제 수행 여부와 상관없이 메모리 주입 여부를 검출할 수 있는 방법을 제안한다. 메모리 주입이 의심되는 프로세스와 동일한 실행 조건을 갖는 복제 프로세스를 실행시키고, 각 프로세스 가상 메모리 영역에 속해있는 데이터 집합을 퍼지 해시를 이용해 비교한 다음 유사도를 산출한다.

II. 관련 연구

박희환 외 1인은 Windows 시스템 파일에 기생하는 악성코드의 치료 방법을 연구하였다 [9]. 사용자의 로그인/로그오프를 담당하는 프로세스인 winlogon.exe 등과 같은 윈도우 운영체제의 관리자 권한을 가진 서비스 프로세스에 메모리 주입 활동이 일어났을 때, 감염된 프로세스를 강제 종료하면 컴퓨터 시스템에 치명적인 블루 스크린 등의 결과가 발생할 수 있다. 따라서 윈도우 운영체제의 시스템 파일을 종료하지 않고 주입된 dll 파일만 제어하기 위해, 종료 함수를 삽입한 스레드를 제어하여 악성 dll의 동작을 종료할 수 있다. 그런데 위 치료 방법은 dll 주입 공격을 통한 메모리 주입의 경우에만 치료가 가능하다. 실제 악성코드가 이용하는 메모리 주입은 프로세스 할로윙, 쉘코드 주입 등 다양한 방법을 통해 이루어진다 [4].

박창욱 외 3인은 퍼지해시를 통해 유사 악성코드 분류 모델에 대해 연구하였다 [10]. PE 섹션 중 .data 영역을 분류 모델의 비교인자로 채택해, 악성코드 간의 유사도를 측정하는 방법이다. 위 방법은 PE 구조가 패킹되지 않아야하며, .data 섹션이 필수적으로 존재하고 무결성을 유지해야한다는 조건이 필요하다. 그런데 최근 악성코드의 동향을 살펴보면, 악성코드 탐지 및 분석을 회피하기 위해 패킹된 바이너리 형태로 유포되는 경우가 많다 [7]. 또한 PE 포맷 형태의 실행파일이 아닌 문서형 악성코드나 원격 실행을 통한 파일리스 형태의 악성코드의 수가 급증하고 있으므로, 해당 방법을 통한 분류는 실제 악성코드 분류에 적합하지 않다 [3].

H. Pomeranz는 메모리 포렌식 분석을 이용한 악성코드 탐지에 대해 분석했다 [11]. 그는 zeus/zbot이라는 악성코드로 실험했으며, 두 악성

코드군은 winlogon.exe 나 explorer.exe와 같은 합법적인 프로세스에 악성코드를 삽입한다. 코드 삽입 검출을 위해 Mandiant사의 Redline과 Volatility라는 도구를 이용했다. Redline과 Volatility의 코드 검출 방법은 프로세스 가상 메모리 보호 상수 중 PAGE_EXECUTE_READWRITE 플래그를 갖는 메모리 영역을 검출하는 방식을 통해 메모리 주입 의심 영역을 탐지해낸다.

정상적으로 로드된 실행 파일은 PAGE_EXECUTE_WRITECOPY의 메모리 보호를 갖기 때문에, 메모리 보호 속성을 통해 해당 주소의 프로세스의 이상 여부를 감지할 수 있다.

그런데 Windows 운영체제는 호출 프로세스의 가상 주소 공간에서 커밋 된 페이지 영역에 대한 보호를 변경할 수 있는 VirtualProtect 함수를 지원한다. VirtualProtect 함수를 WriteProcessMemory 직후에 호출할 경우, 가상 메모리 공간에 데이터 쓰기에 필요한 권한과 VirtualProtectEx 플래그를 변경할 수 있다. 즉 데이터를 복사 할 때 RW 버퍼가 생성되고 실행될 때 RX 버퍼가 생성되므로, 해당 메모리의 메모리 보호 상수는 PAGE_EXECUTE_READWRITE이 아닌 PAGE_EXECUTE_READ 플래그를 갖는 것이다. 따라서 공격자가 VirtualProtect 함수를 이용한다면, 메모리 포렌식 도구를 통한 메모리 주입 공격의 검출이 어려워진다 [8].

메모리 주입 공격이 일어났을 경우, 피해 프로세스의 가상 메모리 공간에 주입된 실행 코드와 같은 흔적이 존재한다는 사실에 기반해, 가상 메모리 데이터 비교를 통한 메모리 주입 공격 여부를 판단할 수 있다.

III. 프로세스 할로윙

3.1 가상 메모리 구조

소프트웨어 실행시 Fig. 1과 같은 프로세스 구조를 관찰할 수 있다. 프로세스 고유의 메모리 영역이라고 할 수 있는 요소로 스택, 힙, 프로세스 환경 블록, 프로그램 이미지, 스레드 환경 블록 등이 존재하며 이와 같은 요소들은 가상메모리라는 소프트웨어와 물리 메모리 사이의 가상 계층에 존재한다. 물리 메모리는 페이지라는 동일한 크기의 블록 단위로 관리되고, 프로세서는 모든 메모리 접근에 대해 페이지

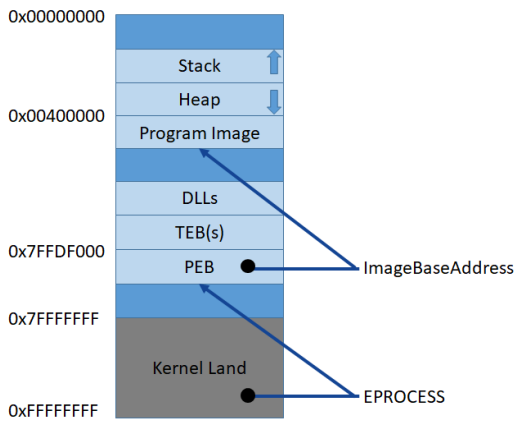

Fig. 1. Process virtual memory structure and addresses call sequences

테이블이라는 테이블을 참고하게 된다.

커널 메모리 영역의 EPROCESS 구조체는 프로세스 환경 블록의 주소를 가리키는 포인터를 가지고 있고, 프로세스 환경 블록의 경우 프로그램 이미지 주소를 가리키는 ImageBaseAddress 포인터를 가지고 있다. 따라서 프로세스 할로윙 공격의 경우 해당 프로세스 고유의 메모리 영역에 악성 프로그램 이미지를 주입한 다음, 프로세스 환경 블록의 ImageBaseAddress 포인터의 값을 새로운 악성 프로그램 이미지의 주소로 변경해 정상 프로세스 내에서 전혀 다른 악성 행위를 수행하게 만들 수 있는 것이다.

3.2 프로세스 할로윙 공격의 동작 방식

악성코드는 CREATE_SUSPEND 플래그를 가지는 CreateProcess 함수를 통해 정지 상태의 양성 프로세스를 생성한다. 호스트 프로그램은 메모리에 로드되었지만, 일시정지 상태이기 때문에 실행코드는 실행되지 않은 상태이다. 그 후 해당 프로세스의 정상 실행코드 영역을 메모리에서 할당 해지하거나, VirtualAllocEx와 같은 가상메모리 할당 함수를 이용해 악성코드가 주입될 영역을 추가적으로 확보한다. 이후 악성코드는 WriteProcessMemory 함수를 사용하여 정상 프로세스에 악성코드를 쓰게 된다.

이와 같은 상태를 거치면, 악성 프로세스 이미지가 주입된 프로세스가 일시정지 상태로 존재하게 된다. 이후 새로 주입된 프로세스 이미지 섹션의 주소

를 가리키는 포인터를 조정하고, SetThread Context함수를 이용한다. 마지막으로 Resume Thread 함수를 사용하여 일시 정지 상태의 프로세스를 재시작 함으로써, 양성 프로세스 모습을 한 악성 프로세스를 실행시킨다.

IV. Windows API

4.1 프로세스 메모리 영역 획득

윈도우 운영체제 보안정책상 특정 프로세스가 타 프로세스 메모리 영역의 데이터에 접근하는 것은 불가능하다. 하지만 윈도우는 디버깅의 이유로 ReadProcessMemory 라는 Windows API (Application Programming Interface) 를 제공한다. ReadProcessMemory 함수를 이용하기 위해 ReadProcessMemory 함수가 필요로 하는 인자를 구해야한다. 아래 Fig 2는 ReadProcess Memory 함수에서 사용되는 인자를 나타낸 것이다.

ReadProcessMemory 함수는 다섯 개의 인자를 필요로 하는데, 다섯 번째 인자는 일반적으로 사용하지 않으며 NULL 값을 넣어 무시할 수 있다. ReadProcessMemory 함수 중 다섯 번째 인자를 제외한 나머지 네 개의 인자는 순서대로 아래와 같다.

(가) 대상 프로세스의 Open Handle
(나) 대상 프로세스에서 읽을 값의 시작 메모리 주소
(다) 메모리에서 읽은 값을 저장할 포인터
(라) 메모리에서 읽을 데이터의 바이트 수

```
BOOL ReadProcessMemory(
  HANDLE  hProcess,
  LPCVOID lpBaseAddress,
  LPVOID  lpBuffer,
  SIZE_T  nSize,
  SIZE_T  *lpNumberOfBytesRead
);
```

Fig. 2. ReadProcessMemory function

4.2 페이지 정보

MEMORY_BASIC_INFORMATION 는 프로세스 가상 주소 공간에 있는 페이지 범위에 대한 정보를 가지는 구조체이다. MEMORY_BASIC_INFORMATION 구조체가 필요로 하는 멤버는 순서대로 아래와 같다.

(가) 페이지 영역의 기본 주소에 대한 포인터
(나) VirtualAlloc 함수에 의해 할당 된 페이지 범위의 기본 주소에 대한 포인터
(다) 영역이 처음 할당되었을 때 메모리 보호 옵션
(라) 기본 주소에서 시작하는 영역의 크기(바이트)
(마) 영역의 페이지 상태
(바) 해당 지역의 페이지에 대한 액세스 보호값
(사) 영역의 페이지 유형

4.3 기타 필수 API

프로세스 핸들을 구하기 위한 OpenProcess 함수의 구조는 위 Fig 3과 같다. 또한 프로세스 메모리에서 유효한 값을 스캔하기 위해, VirtualQueryEX 함수를 이용해 읽을 주소의 페이지 정보를 얻어온다. VirtualQueryEX 함수는 프로세스의 핸들, 값을 가져올 메모리의 주소, MEMORY_BASIC_INFORMATION 구조체, MEMORY_BASIC_INFORMATION의 길이를 인자로 갖는다. 아래 Fig 4는 VirtualQueryEX 함수의 구조를 나타낸 것이다.

```
HANDLE OpenProcess(
  DWORD dwDesiredAccess,
  BOOL  bInheritHandle,
  DWORD dwProcessId
);
```

Fig. 3. OpenProcess function

```
SIZE_T VirtualQueryEx(
  HANDLE                    hProcess,
  LPCVOID                   lpAddress,
  PMEMORY_BASIC_INFORMATION lpBuffer,
  SIZE_T                    dwLength
);
```

Fig. 4. VirtualQueryEX function

V. 프로세스 가상 메모리 비교

본 논문에서 제안하는 실시간 프로세스 할로윙 공격 탐지 방법은 아래와 같다.

(가) 현재 실행 중인 프로세스 중 악성 의심이 드는 프로세스의 이름을 찾는다.
(나) (가)의 프로세스 이름과 동일한 이름의 프로세스들의 실행 경로와 실행 매개변수, 해당 프로세스의 pid 를 출력한다.
(다) (가)에서 입력한 이름을 갖는 프로세스 목록 중 비교에 사용할 pid 를 선정한다. 그 다음 개발한 메모리 비교 프로그램에 해당 pid 를 매개변수로 삽입한다.
(라) (다)의 pid 와 동일한 경로, 동일한 조건을 갖는 프로세스를 본 개발물 프로세스의 서브 프로세스로 생성한다.
(마) (라)에서 새로 생성한 프로세스와 (다)에서 선정한 프로세스의 가상 메모리 공간에서 읽을 수있는 영역의 데이터를 스캔해 퍼지 해시 알고리즘을 이용해 비교한다. 프로세스 가상 메모리 데이터 영역의 접근은 4장의 Windows API를 이용한다.

5.1 퍼지 해시(Fuzzy Hash)

해싱(Hashing)은 모든 입력을 고정 크기 출력으로 축소시키는 것이다. 두 데이터의 해시결과가 같으면 같은 데이터라고 판단할 수 있다. 그런데 전체 데이터를 하나의 입력 값으로 해시 알고리즘을 거칠 경우, 바이트 정도의 굉장히 작은 데이터가 일부만 바뀌어도 전체 값이 완전히 바뀔 수 있다. 따라서 퍼지 해시를 이용하면 기존 해시의 무결성을 확보하면서, 비교 대상과의 유사도를 구할 수 있다.

퍼지 해시 알고리즘은 입력 데이터 전체에 대한 해시값을 계산하지 않고, 일정 크기 단위로 구분하여 각 단위 블록에 대한 해시값을 생성하는 방식이다. 메모리 주입 공격이 발생할 경우 동일한 조건의 다른 프로세스와 다른 가상메모리 영역을 포함하고 있어 퍼지 해시 값이 차이날 것이라 예측가능하다.

VI. 실 험

6.1 실험 데이터

본 논문에서는 프로세스 할로윙 공격 여부 탐지를 위해 윈도우 시스템 프로세스인 svchost.exe를 실험 대상 프로세스로 선정하였다. 실험에 사용된 svchost.exe는 총 12개의 프로세스이며, 각 프로세스의 정보는 아래와 같다. 5장에서 제안한 것과 같이 동일한 실행 명령줄을 통해 실험하는 것보다 매개변수 변화를 통해 실험 조건이 다양해진 상황에서, 이미지가 변화한 프로세스와 정상 프로세스 들간의 차이점이 여부가 동일한 명령줄을 가진 경우보다 판별이 어려울 것이라 판단하였다. 아래 Table 1은 실험에 사용된 12개 프로세스의 아이디, 해당 프로세스를 생성한 부모 프로세스 아이디, 프로세스의 소유권을 가진 사용자 그리고 해당 프로세스를 실행시킨 명령줄을 나타낸다.

아래 Table 1을 보면 프로세스 아이디 940과 1232 번만 다른 프로세스와 다른 부모 프로세스 아이디를 갖는다. 해당 두 프로세스는 프로세스 할로윙 공격이 이뤄진 프로세스로서, 프로세스 아이디 940번의 경우 실존하는 IceID 뱅킹 트로이목마 샘플(sha 256: 24503419c21a345173822bd36297c8815e8f91c78c0d46211f33a1738479238a)에서 파생된 svchost.exe 이다. 프로세스 아이디 1232번의 경우 악성행위를 수행하는 것은 아니며, 윈도우 알림창이 뜨는 단순한 프로그램 이미지가 svchost.exe에 주입된 경우이다. 또한 두 경우는 다른 정상 svchost.exe와 다르게 소유권이 일반 유저에게 있는 것을 볼 수 있다. 프로세스 할로윙 공격은 이미지베이스 주소 충돌 여부 등에 따라 기존 이미지를 할당 해제한 다음 악성 이미지를 주입하는 방법과 가상메모리 추가 할당을 통해 생성된 신규 공간에 악성 이미지를 주입하는 방법이 가능하다. 따라서 프로세스 아이디 940번과 1232번의 실행 명령줄이 조금 다른 모습으로 나타났다.

Table 1. Informations of tested processes

pid	parent pid	ownership
	command line (run command and parameter)	
592	469	SYSTEM
	C:\Windows\system32\svchost.exe -k DcomLaunch	
696	469	NETWORK SERVICE
	C:\Windows\system32\svchost.exe -k RPCSS	
784	469	LOCAL SERVICE
	C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted	
828	469	SYSTEM
	C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted	
860	469	SYSTEM
	C:\Windows\system32\svchost.exe -k netsvcs	
940	3140	USERS
	C:\Windows\system32\svchost.exe	
1000	469	SYSTEM
	C:\Windows\system32\svchost.exe -k GPSvcGroup	
1036	469	LOCAL SERVICE
	C:\Windows\system32\svchost.exe -k LocalService	
1128	469	NETWORK SERVICE
	C:\Windows\system32\svchost.exe -k NetworkService	
1232	1572	USERS
	svchost	
1428	469	LOCAL SERVICE
	C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork	
1448	469	SYSTEM
	C:\Windows\System32\svchost.exe -k secsvcs	

6.2 실험 결과

본 논문에서 제안한 퍼지 해시를 이용해 각 프로세스 가상메모리 데이터 간의 유사성을 측정하였다. 실험에 사용된 데이터는 가상메모리에 속하는 읽기 가능 페이지 영역 데이터이며, 프로세스에 속한 페이지 영역을 집합으로 묶고 각 집합을 퍼지 해시 알고리즘으로 비교하였다. Table 2는 0% 나 100% 유사도를 갖는 극댓값과 극솟값을 제외한 다음, 유사성이 보이는 페이지를 얼마나 공유하고 있는지 평균낸 결과 값이다. Table 2를 보면 프로세스 할로윙 공격이 일어난 프로세스와 다른 프로세스 간의 공통적인 페이지 수는 평균 1.18개이다. 유사성을 가진 페이지 수의 범위는 최소 0개에서 최대 2개 값의 범위를 보였으며 0의 값은 프로세스 할로윙이 일어난 두 프로세스 940, 1232 번간의 유사성 측정에서 측정된 결과 값이다. 정상 프로세스의 경우 평균 2.18 ~ 3.91사이의 결과 값을 보였으며 프로세스 할로윙 공격이 일어난 프로세스의 결과 값과 비교했을 때 2배 이상의 값을 보였다. 정상 프로세스 기준으로 비교한 공통 페이지 수의 범위는 1~10의 값을 가졌고, 프로세스 할로윙 공격이 일어난 실험군과 달리 0의 값을 가지는 경우는 존재하지 않았다.

Table 2. Results of experiments

pid	average
592	2.36
696	2.64
784	2.27
828	2.27
860	3.91
940	1.18
1000	2.18
1036	2.27
1128	2.27
1232	1.18
1428	2
1448	3.64

VII. 결 론

본 논문은 임의의 프로세스를 대상으로 동일한 실행 조건을 가진 복제 프로세스를 생성하고, 두 프

로세스 가상 메모리 데이터 비교를 통한 프로세스 할로윙 공격 여부를 판단하는 방법을 제안하였다. 실행 파일이 메모리에 로드되어 실행되면 메모리상에 흔적 정보가 남기 때문에, 메모리 주입 공격 탐지의 경우 메모리 포렌식 도구를 이용한 메모리 분석 방법이 많이 활용된다. 그러나 실제 사용자 환경에서의 메모리 덤프 시간이 굉장히 오래 걸리기 때문에 즉각적인 감염 대처가 불가능하다는 치명적인 단점이 존재한다. 또한 최근 악성코드들은 악성 행위를 은닉하기 위해 Sleep 함수 등을 통해 악성 행위의 시간을 지연시키는 경우가 잦다. 이와 같은 상황에서 본 논문이 제안하는 방법을 이용하면, 악성 행위 등이 이루어지지 않아 탐지에 이용할 특징 정보를 뽑아내기 힘든 시점에도 합법적인 프로세스로 위장한 악성 프로세스의 탐지가 가능하며 즉각적인 차단이 가능해진다. 따라서 메모리 주입 공격이 발생되어져 있는 시스템 환경에서 프로세스 가상 메모리 데이터 비교로 실시간으로 감염여부를 판단한다면, 메모리 포렌식 방식의 단점을 극복할 수 있다.

하지만 퍼지 해시라는 방법의 특성 상, 단순히 값만 비교할 경우 6장의 실험 결과처럼 큰 차이를 만들어내진 못한다는 단점이 존재한다. 이와 같은 단점을 보완하기 위해 향후 연구로서, 조금 더 데이터를 작게 분할해 비교해보고 뽑아낸 데이터를 파싱하거나 분석하는 전처리하는 과정을 추가할 예정이다. 또한 2장에 기술한 것과 같이 메모리 보호 상수는 메모리 주입 공격에 큰 영향을 끼치는 중요한 특징정보 이므로 메모리 보호 상수를 분석하는 단계도 추가할 예정이다. 마지막으로 프로세스를 생성한 부모 프로세스 등의 환경 정보, 프로세스의 생성 시간 및 시간당 중앙처리장치 사용량 등을 시각화한 그래프 이용 등의 방법을 시도해 정확도를 증가시킬 예정이다.

References

[1] A. Magnusardottir, "Fileless ransom ware. How it works and how to stop it [Online]", infosecurity, Available: https://www.infosecurityeurope.com/__novadocuments/483997?v=636650015234830000(downloaded 2018, OCT.28)

[2] M. Gorelik, R. Moshailov, et al, "Fileless Malware: Attack Trend Exposed", Morphisec Ltd, 2017

[3] J. Smelcer, "The rise of Fileless malware", Utica College, ProQuest Dissertations Publisjing, 10642524, 2017

[4] P. Black, I. Gondal, R.Layton, "A survey of similarities in banking malware behaviours", Computers & Security, vol 77, pp. 756~772, 2018

[5] Ligh, H. Michael, A. Case, J. Levy and A. Walters, "The art of memory forensics: detecting malware and threats in windows, linux, and Mac memory", John Wiley & Sons, 2014

[6] T. Tomer, "Detecting the one percent: Advanced targeted malware detection", *Proceedings of the RSA Conference*, San Francisco, USA, Vol. 8, 2013

[7] J. A. Marpaung, M. Sain, Lee. H. J, "Survey on malware evasion techniques: State of the art and challenges", In Advanced Communication Technology (ICACT), *2012 14th International Conference*, pp. 744-749, 2012

[8] Lim. S. M and Im. E. G, "Proposal of Process Memory Injection Verification Method Using Memory Protection Constants", *Proceedings of the 2018 Conference on International Confer-ence on Convergence Content*, pp. 55-56, 2018

[9] Park. H. H and Park. D. W, "A Study on Treatment Way of a Malicious Code to injected in Windows System File," KSCI Review, Vol. 14, No. 2, pp. 255-262, 2006

[10] Park. C. W, Chung. H. J, Seo. K. S and Lee. S. J, "Research on the Classification Model of Similarity Malware using Fuzzy Hash," *Journal of the Korea Institute of Information Security & Cryptology*, Vol. 22, No. 6, pp. 1325-1336, 2012

[11] H. Pomeranz, "Detecting malware with memory forensics", SANS Institute, 2015

〈저자소개〉

임 수 민 (Su Min Lim) 정회원
2018년 2월: 경기대학교 컴퓨터과학과 학사
2018년 3월~현재: 한양대학교 컴퓨터·소프트웨어학과 석사과정
〈관심분야〉 악성코드 정적분석, 소프트웨어 보안, 컴파일러

임 을 규 (Eul Gyu Im) 정회원
1992년: 서울대학교 컴퓨터공학과 학사
1994년: 서울대학교 컴퓨터공학과 석사
2002년: University of Southern California, Computer Science Ph.D.
2005년~현재: 한양대학교 컴퓨터공학부 정교수
〈관심분야〉 제어시스템 보안, 악성코드, 정보 보호, 소프트웨어 취약 점검