Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.5, Oct. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.5.1013

윈도우 최대 절전 모드 파일의 메모리 데이터 암호화 기법 연구*

이 경 호,† 이 우 호, 노 봉 남‡
전남대학교 정보보안협동과정

Study on Memory Data Encryption of Windows Hibernation File*

Kyoungho Lee,† Wooho Lee, Bongnam Noh‡
Chonnam National University Interdisciplinary Program of Information Security

요 약

최대 절전 모드는 물리 메모리의 데이터를 비휘발성 매체에 저장하였다가 시스템에 전원이 들어오면 메모리 데이터를 비휘발성 매체로부터 물리 메모리에 복구하는 기능이다. 최대 절전 모드 파일은 메모리 데이터를 정적 상태로 가지기 때문에 공격자가 이를 수집할 경우에 시스템의 물리 메모리에 있던 사용자 아이디와 패스워드 및 디스크 암호화키 등의 주요 정보들이 유출될 수 있는 위험성이 존재한다. 윈도우에서는 최대 절전 모드 파일만을 위한 보호 기능을 지원하지 않기 때문에 최대 절전 모드 파일에 기록되는 메모리 데이터 내용들을 보호하는 방법이 필요하다.

본 논문에서는 최대 절전 모드 파일에 기록되는 프로세스의 메모리 데이터를 보호하기 위해서 최대 절전 모드 파일 내의 물리 메모리 데이터를 암호화하는 방법을 제안한다. 최대 절전 모드 처리 시에 메모리 데이터를 암호화하기 위해 최대 절전 모드 처리 절차를 분석하고, 최대 절전 모드 파일에 기록되는 메모리 데이터에 대한 암호화 과정이 프로세스 각각에 대해 투명하게 동작하도록 구현하였다. 실험을 통해 구현한 최대 절전 모드 파일 내 프로세스 메모리 데이터 암호화 도구는 암복호화 비용으로 약 2.7배의 오버헤드를 보였다. 이런 오버헤드는 전체 디스크 암호화 도구가 지속적으로 10% 이상의 속도저하를 유발하는 상황과 비교해 볼 때, 공격자에게 프로세스의 평문 메모리 데이터가 노출되지 않기 위해 필요한 비용으로 충분히 감내할 수 있다고 판단된다.

ABSTRACT

Windows hibernation is a function that stores data of physical memory on a non-volatile media and then restores the memory data from the non-volatile media to the physical memory when the system is powered on. Since the hibernation file has memory data in a static state, when the attacker collects it, key information in the system's physical memory may be leaked. Because Windows does not support protection for hibernation files only, we need to protect the memory that is written to the hibernate file.

In this paper, we propose a method to encrypt the physical memory data in the hibernation file to protect the memory data of the processes recorded in the hibernation file. Hibernating procedure is analyzed to encrypt the memory data at the hibernating and the encryption process for hibernation memory is implemented to operate transparently for each process. Experimental results show that the hibernation process memory encryption tool showed about 2.7 times overhead due to the crypt cost. This overhead is necessary to prevent the attacker from exposing the plaintext memory data of the process.

Keywords: Physical Memory, Memory Encryption, Hibernation File

Received(08. 09. 2017), Modified(09. 22. 2017), Accepted(09. 26. 2017)
* 이 연구는 ETRI부설 국가보안기술연구소의 지원으로 수행되었음.
† 주저자, koungholee@gmail.com
‡ 교신저자, bbong@jnu.ac.kr(Corresponding author)

I. 서 론

윈도우의 최대 절전 모드는 시스템 전원을 끄기 전에 물리 메모리에 있는 내용을 하드 디스크에 정적으로 기록한다. 이후 시스템에 전원이 들어오면 하드 디스크에 저장된 메모리 데이터가 물리 메모리에 다시 적재되고 시스템은 최대 절전 모드 진입 전 상태와 같이 동작한다.

최대 절전 모드 파일은 운영체제에 의해 자동으로 생성되는 파일로써, 공격자가 최대 절전 모드 상태의 시스템에 접근 및 최대 절전 모드 파일을 탈취하여 파일 내의 물리 메모리 데이터에서 민감한 정보들을 수집할 수 있다.

윈도우는 비트라커의 디스크 전체 암호화 기능을 통해 최대 절전 모드 파일 암호화 기능을 지원하지만, 최대 절전 모드 파일에 기록되는 메모리 데이터 외에 다른 파일들도 암호화하기 때문에 시스템에 지속적인 오버헤드가 발생한다. 따라서 최대 절전 모드 파일에 기록되는 메모리 데이터만을 보호하는 방법이 필요하다.

본 논문에서는 최대 절전 모드 파일에 기록되는 프로세스의 메모리 데이터를 보호하는 방법으로 프로세스 메모리 데이터 암호화 방법을 제안한다.

2절에서는 윈도우에서의 최대 절전 모드와 최대 절전 모드 파일 노출에 대해 설명한다. 3절에서는 물리 메모리 페이지를 암호화하는 이유와 최대 절전 모드 처리 절차를 설명한다. 4절에서는 최대 절전 모드 처리 절차 중 메모리를 암복호화하는 방법을 제안한다. 5절에서는 제안한 방법을 구현한 드라이버와 드라이버에 대한 테스트를 진행한다. 마지막 절에서는 결론을 맺으며 향후 연구 내용을 설명한다.

II. 관련 연구

2.1 최대 절전 모드

최대 절전 모드 기능은 ACPI 규격에 슬리핑 모드 S4로 정의되어 있으며, 메인 메모리의 모든 내용을 하드 드라이브같은 비휘발성 메모리에 저장하고 시스템 전원은 차단되는 것을 의미한다[1]. 윈도우는 이 규격에 따라 S4를 슬리핑 모드로 명시하며 최대 절전 모드 또는 하이버네이션(hibernation)이라 한다[2].

윈도우의 최대 절전 모드는 시스템 전원을 끄기 전에 시스템의 물리 메모리에 있는 내용을 하드 디스크와 같은 비휘발성 매체에 기록하는 기능이다. 최대 절전 모드 상태의 컴퓨터에 전원이 들어오면 하드 디스크에 저장된 내용이 메모리에 다시 적재되면서, 최대 절전 모드 진입 전의 원래 상태로 되돌아온다.

일반적으로 최대 절전 모드 파일은 시스템 볼륨 루트에 hiberfil.sys 파일로 저장된다. 시스템이 최대 절전 모드에서 복귀할 때 윈도우 부팅 관리자(bootmgr)는 최대 절전 모드 파일이 있는 것을 확인하고 시스템 재개 전용 로더(winresume.exe)를 실행하여 최대 절전 모드 파일로부터 정적으로 저장된 물리 메모리 데이터를 메모리에 로드한다.

최대 절전 모드 파일은 추가적인 프로그램이나 장비 없이 덤프 장비를 설치할 수 없는 시스템에서 단순히 최대 절전 모드 진입으로 메모리 이미지를 수집하는 유용한 방법이라고 할 수 있다.

2.2 물리 메모리 데이터 노출

윈도우 운영체제에서 프로세스들의 실행 코드와 처리되는 데이터는 메모리 계층에 따라 하드 디스크로부터 물리 메모리에 로드되고 중앙 처리 장치(CPU)에 의해 처리된다. 물리 메모리에서 데이터는 처리 및 가공되기 때문에 하드 디스크에 기록되지 않는 휘발성 정보를 포함한다. 예를 들어, 디스크 암호화키[3], 사용자 문서 입력 정보[4], 로그인 아이디와 비밀번호[5] 등의 민감한 정보들이 물리 메모리에 존재할 수 있다.

공격자는 콜드부트 공격을 이용한 메모리 덤프[6]나 windd, winpmem 같은 소프트웨어 메모리 덤프 도구를 사용하여 메모리 이미지를 수집할 수 있다. 이어서 공격자는 수집한 메모리 이미지에 대해 문자열 기반 탐색이나 volatility[7], rekall[8]과 같은 메모리 포렌식 도구를 이용하여 시스템의 주요 정보를 수집할 수 있다.

최대 절전 모드 파일은 추가적인 메모리 덤프 장치나 소프트웨어 없이 윈도우 운영체제에 의해 자동으로 하드디스크에 생성되는 메모리 데이터 파일이다. 따라서 최대 절전 모드 파일을 일반적인 물리 메모리와 같이 분석하기 위해서 최대 절전 모드 파일 구조에 대한 연구가 이루어져 압축 해제 도구가 공개되었다[9][10]. 물리 메모리 이미지를 분석하고 정보를 얻기 위해 개발된 도구 중 하나인 volatility는 별도의 압축 해제 절차 없이 최대 절전 모드 파일

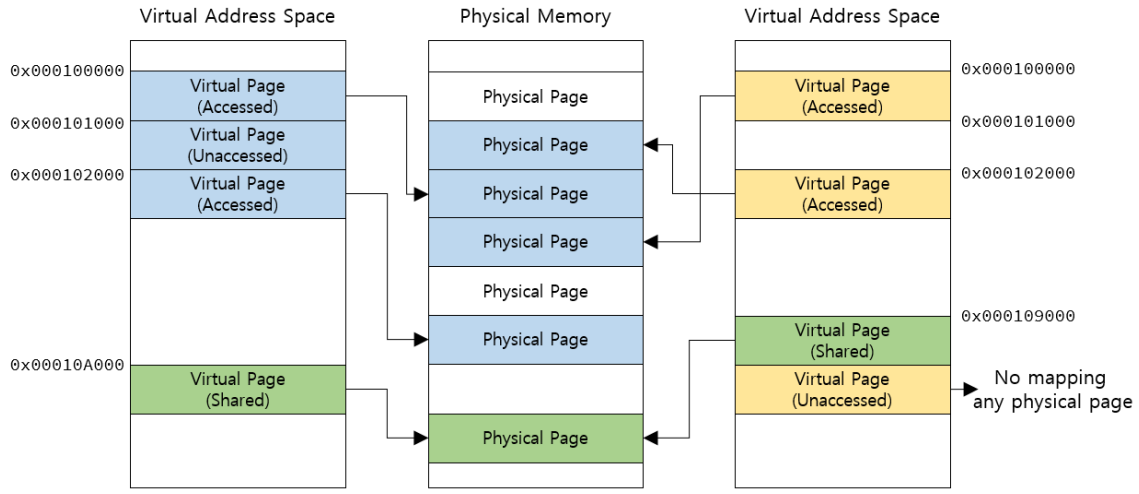


Fig. 1. Mapping Virtual Memory with Physical Memory

구조를 따라 일반 메모리 덤프 이미지와 같이 분석 가능하다[11].

공격자는 최대 절전 모드 상태의 윈도우 시스템에 접근하여 하드 디스크로부터 최대 절전 모드 파일을 수집하여 최대 절전 모드 파일 변환 도구 및 메모리 포렌식 도구를 이용하여 메모리 데이터를 분석하고 공격 대상의 주요 정보들을 수집할 수 있다. 따라서 최대 절전 모드 파일이 공격자에게 노출되었을 때에 대비하여 메모리 데이터를 보호할 방법이 필요하다.

최대 절전 모드 파일을 보호하기 위한 기존 방법은 전체 디스크 암호화 도구를 사용하는 것이다. 전체 디스크 암호화 도구 중 비트라커[12], 베라크립트[13]는 윈도우가 설치된 시스템 파티션에 대한 디스크 암호화를 통해 최대 절전 모드 파일 암호화를 지원한다[14].

시스템 파티션 암호화는 최대 절전 모드 파일만을 암호화하는 것이 아닌 윈도우에서 처리되는 모든 파일들에 대해서 암/복호화를 진행하기 때문에 시스템 사용 중 지속적인 오버헤드가 발생한다. 예를 들어, 마이크로소프트에 따르면 비트라커는 10% 이하의 오버헤드가 발생하며[15], 베라크립트의 전신인 트루크립트[16]는 SSD가 설치된 노트북 환경에서 15%의 오버헤드가 발생하였다[17].

본 논문에서는 최대 절전 모드 파일에 기록되는 메모리 데이터 외에 다른 파일들도 암호화하고, 이에 따른 오버헤드가 발생하는 전체 디스크 암호화 도구의 시스템 파티션 암호화 기능의 한계점을 극복하기 위해 최대 절전 모드 파일만을 암호화하는 방법을 제안한다.

III. 연구 내용

3.1 윈도우 메모리 구조

윈도우는 가상 메모리를 사용하는 운영체제로 프로세스들은 자신만의 가상 주소 영역을 가진다. 프로세스가 가상 주소에 접근할 때 프로세서에 의한 가상 주소 변환이 일어나 매핑된 물리 주소로 접근한다. 윈도우는 4096 바이트 크기의 페이지 단위로 가상 메모리와 물리 메모리를 할당하고 관리하기 때문에 이들 주소 접근은 페이지 단위로 발생한다.

프로세스는 자신만의 가상 주소 영역을 가지기 때문에 서로 다른 프로세스에서 동일한 주소 영역으로 접근하더라도 Fig.1.과 같이 서로 다른 물리 메모리로 연결된다. 또는 프로세스간 공유 메모리 영역을 설정하면 서로 다른 주소로 접근해도 프로세서에 의해 동일한 물리 메모리로 연결된다.

프로세스가 가지는 가상 주소 영역은 실제로 존재하지 않는 메모리 영역도 포함한다. 예를 들어, Fig.1.과 같이 최초 접근되지 않은 메모리(Unaccessed)는 지연 평가로 인해 가상 영역에는 할당되어 있지만 물리 메모리에는 존재하지 않는다. 할당 후 최초 접근 시에 운영체제에서 접근된 페이지에 대한 물리 페이지를 할당한다.

시스템은 또한 물리 메모리가 부족할 시 사용되지 않는 가상 메모리 일부를 페이지 파일이라 불리는 하

드 디스크의 메모리 백업 영역으로 내린다. 이런 메모리를 페이징된 메모리(Paged Memory)라고 부르며 이후 가상 메모리 영역에 접근할 시 페이지 파일로부터 메모리 데이터를 물리 메모리로 불러온다. 따라서 프로세스에 의해 사용되는 실제적으로 존재하는 메모리 데이터는 물리 메모리에 존재한다.

3.2 최대 절전 모드 처리

윈도우의 최대 절전 모드 처리는 Fig.2.에서 보이듯이 크게 최대 절전 모드 파일에 압축할 물리 메모리를 선별하는 메모리 미러링(Memory Mirroring) 단계, 하드웨어 장치들에게 전원 변경을 공지하는 단계, 메모리 압축 저장 단계, 메모리 복구 단계로 나뉜다.

3.2.1 메모리 미러링

최대 절전 모드에 진입하면서 윈도우 커널은 하드 디스크에 저장할 물리 메모리를 선별하는 메모리 미러링을 수행한다. 윈도우는 물리 메모리의 모든 내용을 단순히 최대 절전 모드 파일에 기록하지 않는다. 물리 메모리에 있는 메모리 중 일부 메모리는 하드디스크에 있는 파일과 매핑된 상태이기 때문에 이런 물리 메모리들은 해제한다.

더 자세히 설명하면, 최대 절전 모드 집입 시에 메모리에 매핑된 파일(Memory Mapped File)이나 로드된 이미지(EXE, DLL)와 매핑된 물리 메모리들을 하드 디스크에 다른 파일 형태로 저장하는 것은 비효율적이다. 윈도우는 파일과의 매핑된 물리 메모리 페이지를 해제하고 최대 절전 모드에서 복구된 후에 프로세스가 해당 메모리 접근할 때 본래 매핑된 파일을 물리 메모리로 읽어온다. 따라서 파일과 매핑된 물리 메모리들을 제외한 물리 메모리들이 최대 절전 모드 파일에 저장된다.

3.2.2 전원 변경 전달

최대 절전 모드 파일에 기록될 물리 메모리들을 선정하는 단계인 메모리 미러링을 마친 후, 윈도우는 하드웨어 장치들에게 최대 절전 모드로 진입하는 전원 상태를 알린다. 하드웨어 장치들은 이 알림을 받고 각 하드웨어의 드라이버에서 최대 절전 모드 진입에 대비한다.

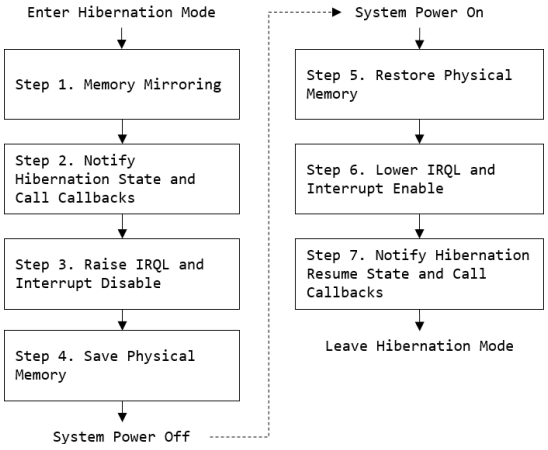


Fig. 2. Hibernation Process

3.2.3 프로세서 선점

윈도우 운영체제는 인터럽트 처리에 우선권을 두기 위해 IRQL(Interrupt ReQuest Level)을 사용한다. 고수준의 인터럽트가 처리되는 동안 저수준의 인터럽트는 프로세서 선점에 실패하고 고수준의 인터럽트가 처리될 때까지 대기하게 된다.

윈도우 스레드 스케쥴러는 최저수준의 IRQL인 패시브 레벨(PASSIVE_LEVEL)에서 각 스레드들이 프로세서를 선점하도록 관리한다. 따라서 커널 코드가 패시브 레벨보다 높이 동작할 경우, 스레드 스케쥴러에 의한 프로세서 선점이 발생하지 않는다.

최대 절전 모드 코드는 최대 절전 모드 파일 생성 준비를 마치면, 메모리에 변경이 일어나지 않기 위해 IRQL을 높이고, 모든 인터럽트를 비활성화 한다. 이 상태에서 메인 프로세서는 오직 최대 절전 모드 파일 생성 코드만을 처리하게 된다.

3.2.4 메모리 압축 저장

최대 절전 모드 파일의 메타 정보를 지닌 파일 헤더를 생성하고 연속되는 물리 메모리 페이지들 압축한다. 윈도우 커널은 최대 절전 모드로부터 복구 시에 정확하게 압축된 물리 메모리 영역에 대한 정보를 저장하기 위하여 메모리 페이지 테이블 구조를 사용한다.

메모리 페이지 테이블은 시스템 재개 시 복구할 물리 메모리 위치 정보를 포함하고, 각 압축 데이터 영역은 압축된 물리 메모리의 크기, 압축된 물리 메모리로 구성된다[9]. 데이터 압축은 허프만 인코딩

알고리즘을 변형한 XPRESS 알고리즘을 사용한다 [18].

3.2.5 최대 절전 모드 복구

메모리 복구 작업은 최대 절전 모드 복구 로더인 winresume.exe에 의해 이루어진다. 이 로더는 최대 절전 모드 파일 구조에 따라 메모리 페이지 테이블과 XPRESS 압축 영역을 참조하여 최대 절전 모드 진입 전과 정확하게 동일한 위치에 압축 해제한 물리 메모리 데이터를 올린다.

모든 압축된 물리 메모리 페이지들을 압축 해제하여 물리 메모리에 올린 후 프로세서 상태를 복구한다. 이후 IRQL 레벨 조정과 인터럽트 활성화를 통해 프로세서 선점을 해제한다. 이 시점부터 시스템은 최대 절전 모드 진입 전의 실행 흐름을 이어간다. 추가적으로 하드웨어 디바이스들에게 최대 절전 모드에서 복구되었음을 알린다.

IV. 제안하는 메모리 암호화 방법

최대 절전 모드 파일에 기록될 어플리케이션 메모리를 암호화하기 위해서는 중요한 점은 첫 번째로 암호화할 대상인 어플리케이션 메모리와 비암호화 대상인 시스템 메모리를 구분해야 하는 것이다. 두 번째는 물리 메모리가 최대 절전 모드 파일에 기록되기 전에 어플리케이션 메모리를 암호화하는 시점이다.

4.1 PFN 데이터베이스 기반 암호화 메모리 선별

윈도우는 각 물리 메모리 페이지에 대한 메타 정보를 PFN 데이터베이스에 관리하며, 각 메타 정보는 Fig.3.과 같은 _MMPFN 구조체로 표현된다 [19].

_MMPFN 구조체의 PteFrame 필드는 상위 페이지 프레임을 가리키고, 이 상위 페이지 프레임이 자기 자신을 가리킬 때 Flink 필드는 물리 페이지를 할당한 프로세스를 가리킨다. 따라서 PteFrame 필드를 통해 상위 페이지 프레임을 따라가면 해당 물리 메모리 페이지를 할당한 프로세스를 알아낼 수 있다.

PteAddress 필드는 가상 주소 변환 구조에서 사용되는 PTE가 있는 가상 주소를 나타낸다. 윈도우는 가상 주소와 PTE간 간편하게 변환하는 인코딩을 사용한다. 아래는 윈도우 내에서 사용하는 PTE 주소를 가상 주소로 변환하는 공식으로 (1)은 32비트 시스템, (2)는 64비트 시스템에서 사용한다.

```
nt!_MMPFN
   +0x000 Flink            : Uint4B
   +0x004 u2               : <unnamed-tag>
   +0x008 PteAddress       : Ptr32 _MMPTE
   +0x008 VolatilePteAddress : Ptr32 Void
   +0x008 Lock             : Int4B
   +0x008 PteLong          : Uint4B
   +0x00c u3               : <unnamed-tag>
   +0x010 OriginalPte      : _MMPTE
   +0x010 AweReferenceCount : Int4B
   +0x018 u4               : <unnamed-tag>
      +0x000 PteFrame          : Pos 0, 25 Bits
      +0x000 PfnImageVerified  : Pos 25, 1 Bit
      +0x000 AweAllocation     : Pos 26, 1 Bit
      +0x000 PrototypePte      : Pos 27, 1 Bit
      +0x000 PageColor         : Pos 28, 4 Bits
```

Fig. 3. MMPFN structure definition

$$VirtualAddress = PteAddress \gg 9 \quad (1)$$

$$VirtualAddress = (PteAddress \ll 25) \gg 16 \quad (2)$$

MMPFN 구조체를 기반으로 물리 페이지를 소유한 프로세스와 가상 주소를 검사한다. 소유 프로세스만으로 암호화 대상 페이지를 판단하지 않는 이유는 프로세스 영역 중 커널에서 사용하는 영역인 페이지 테이블 영역과 프로세스 관련 환경 블록들이 있기 때문이다. 페이지 테이블 영역은 해당 페이지 테이블을 생성한 프로세스 영역에 속한다. 또한 프로세스 환경 블록(PEB)와 스레드 환경 블록(TEB)들도 각 프로세스 영역에 속한다. 최대 절전 모드 파일 생성 시에도 이들 영역은 사용되기 때문에 물리 페이지에 대한 가상 주소를 알아내어 이러한 주소는 암호화하지 않는다.

4.2 메모리 암호화 시점

4.2.1 전원 정책 변경 콜백

최대 절전 모드로 진입할 때 알림을 받을 수 있는 윈도우에서 지원하는 방법은 윈도우 전원 정책 변경 콜백[20]을 사용하는 것이다. 전원 정책 변경 콜백에 등록된 함수는 시스템의 전원 상태가 변경될 때마

다 호출된다. 호출 시 함수는 전달 인자를 이용하여 최대 절전 모드 진입과 시스템 슬립 등의 다른 전원 상태 변경을 구별할 수 있다.

최대 절전 모드 진입 시 전원 정책 변경 콜백이 호출되는 시점은 Fig.2.에서 2단계에 해당하며, 디바이스들에게 전원 상태 변경을 알리는 시점과 동일하다. 이 시점은 프로세스 스레드 및 커널 스레드들이 프로세서를 선점하면서 동작하고 있고, 인터럽트가 활성화되어 있다.

전원 정책 콜백이 호출될 때 메모리를 암호화하면 동작 중인 프로세스 스레드들이 암호화된 메모리 영역에 접근하여 잘못된 처리로 에러가 발생할 수 있다. 예를 들어, 스레드가 암호화된 코드 영역을 실행하여 에러가 발생할 수 있고, 암호화된 데이터 영역에 접근하여 잘못된 데이터를 처리하여 에러가 발생할 수 있다.

프로세스 스레드들이 암호화된 메모리 영역에 접근하지 못하게 하는 방법으로 스레드들을 정지 상태(Suspend)로 만들고 메모리를 암호화할 수 있지만, 이는 프로세스가 자신의 스레드들이 정지되는 것을 알기 때문에 프로세스에 대해 투명한 작업이 되지 못한다. 따라서 최대 절전 모드 스레드가 프로세서를 선점하고 나서 메모리를 암호화해야 한다.

4.2.2 PopPowerStateHandlers 후킹

최대 절전 모드 진입 시 윈도우 커널은 메모리 미러링을 마치고 프로세스를 선점한 후, 커널 전역 변수 PopPowerStateHanlders에 기록된 함수를 호출하여 최대 절전 모드 파일 생성 함수(PopSaveHiberContext)를 호출한다. 따라서 커널 전역 변수 PopPowerStateHanlders의 값을 메모리 암복호화 함수 주소로 변경하면, 최대 절전 모드 진입 시 메모리 암/복호화 함수가 콜백 형태로 호출된다.

호출된 메모리 암복호화 함수에서 PFN 데이터베이스 기반 방법으로 암호화할 프로세스 물리 메모리들을 선정하고 암호화한다. 암호화 과정이 완료되면 최대 절전 모드 파일 생성 함수를 호출하고, 최대 절전 모드 파일 생성 함수 내에서 최대 절전 모드 파일 생성 후 전원이 차단된다.

시스템 재개 시 메모리가 압축 해제되어 본래 위치로 복구되고 프로세서 정보가 복구된 후 커널의 실

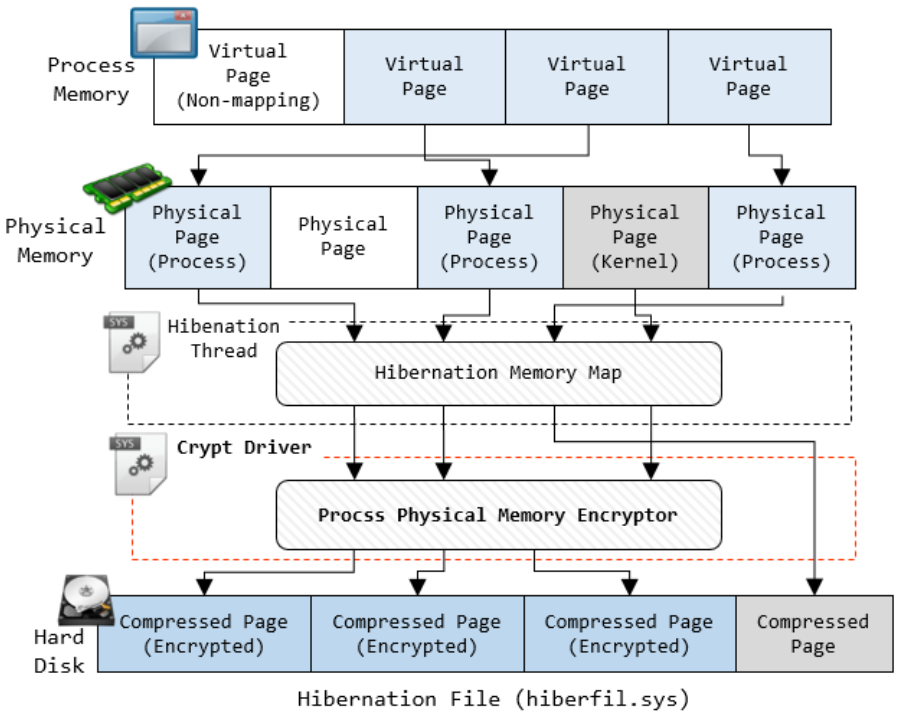


Fig. 4. Encrypting Hibernation Physical Memory Overview

행 흐름은 메모리 암복호화 함수로 돌아온다. 이 시점은 아직 최대 절전 모드 스레드가 프로세서를 선점하고 있기 때문에 메모리 복호화를 수행한다.

PopPowerStateHandlers 후킹 방법은 최대 절전 모드를 처리하는 스레드 외에 다른 스레드들이 프로세서를 선점하지 않고 인터럽트 비활성화 상태이기 때문에 프로세스 개인 메모리 영역뿐만 아니라 프로세스 간 공유 메모리 영역도 암호화 및 복호화할 수 있다. 프로세스의 스레드들은 최대 절전 모드 진입 후부터 최대 절전 모드 복구 완료 시까지 프로세서를 선점하지 못하기 때문에 자기 자신의 메모리가 암호화되는지 모르게 동작한다. 따라서 암호화 과정은 프로세스에 대해 투명하게 동작한다.

V. 구현 및 실험

최대 절전 모드 파일의 프로세스 메모리 암호화 도구는 물리 메모리에 접근할 수 있는 권한을 가져야 하기 때문에 커널에서 실행되는 윈도우 드라이버 형태로 구현하였다. PopPowerStateHandlers에 후킹 함수를 삽입하여, 최대 절전 모드에 진입하고 최대 절전 모드 스레드가 프로세서를 선점한 후에 암호화 코드가 실행된다.

Fig.4.와 같이 최대 절전 모드 처리 스레드가 생성한 최대 절전 모드 파일에 기록할 메모리 맵(Hibernation Memory Map)을 참조하여 PFN 데이터 베이스 기반 프로세스 메모리를 선별하고 암호화 한다. 암호화 후 최대 절전 모드 스레드는 하드디스크 내에 최대 절전 모드 파일을 생성한다.

암호화 코드(Process Memory Encryptor)는 Fig.5.와 같이 최대 절전 모드 진행 시에 시스템이 압축하기 위해 생성한 메모리 맵(*map*)을 인자로 받아 PFN 데이터베이스를 기반으로 프로세스 소유의 메모리 페이지를 선별하고(3라인) 선별된 메모리 페이지들을 AES-128 알고리즘을 사용하여 암호화한다(4라인). 이 때 메모리 복호화를 대비하여 암호화된 페이지 위치를 맵에 기록한다(5라인).

시스템이 재개 되고 로더에 의해 물리 메모리가 복구되고 난 후 실행 흐름은 후킹 함수로 돌아와 복호화 함수(Decrypt Memory)를 호출한다. 복호화 함수는 프로세스 물리 페이지 암호화 시에 기록한 메모리 위치(pmap)를 참조하여 암호화된 페이지들을 복호화한다(25~27 라인).

윈도우 커널은 최대 절전 모드 처리 시에 최대 절

Algorithm 1 Process Physical Memory Crypt Algorithm

```
1: procedure ENCRYPT MEMORY(map)
2:     for all pfn ∈ map do
3:         if CHECK SYSTEM PAGE(pfn) ≠ true then
4:             EncryptPage pfn
5:             SetMap(pmap, pfn)
6:         end if
7:     end for
8: end procedure

9: procedure CHECK SYSTEM PAGE(pfn)
10:    mmpfn ← GetMmPfn(pfn)
11:    address ← SHIFTLEFT(mmpfn.PteFrame, 9)
12:    if address > 0x7F000000 then
13:        return true
14:    end if
15:    while mmpfn.PteFrame ≠ pfn do
16:        pfn ← mmpfn.PteFrame
17:        mmpfn ← GetMmPfn(pfn)
18:    end while
19:    if mmpfn.Flink = SystemProcess then
20:        return true
21:    end if
22:    return false
23: end procedure

24: procedure DECRYPT MEMORY(pmap)
25:    for all pfn ∈ pmap do
26:        DecryptPage pfn
27:    end for
28: end procedure
```

Fig. 5. Physical Memory Encryption Algorithm

전 모드 파일 관련 성능을 파일 헤더에 프로세서 클록 사이클 수로 기록한다. 최대 절전 모드 처리 시에 시스템 시간 관련 함수가 동작하지 않아 정확한 암복호화에 소요된 시간 측정이 불가능하기 때문이다. 따라서 본 논문에서는 최대 절전 모드 파일에 대한 평문 압축 및 압축 해제 시와 암복호화 시에, 실시간 타임 스탬프 카운터 명령어(rdtsc)를 이용하여 프로세서 클록 사이클 수를 측정 및 비교하였다.

테스트는 3.4GHz 프로세서와 4GB RAM을 가진 가상 머신 상의 윈도우 7 32/64비트 환경에서 진행하였다. 가상 머신의 스냅샷 기능을 이용하여 테스트 드라이버 로드를 제외한 동일한 환경에서 메모리 데이터 변경 없이 실험하기 위해서 가상 머신을 사용하였다.

32비트 시스템 실험 결과인 Table 1.과 64비트 시스템 실험 결과인 Table 2.를 보면 메모리 암호화 코드가 추가되어 기존 최대 절전 모드 진입 시에 소요된 사이클보다 오버헤드가 32비트 시스템에서 2.86배, 64비트 시스템에서 2.69배로 발생하였다. 메모리 압축에 소요된 사이클 수는 기존 최대 절전 모드 진입할 때보다 메모리 암호화 시에 32비트 시

Table 1. 32-bit System Hibernation File Encryption/Decryption Performance Results(1K Cycle Count)

Process	Normal Hibernation	Encrypting Hibernation	Over head	Process	Normal Hibernation	Decrypting Hibernation	Overh ead
Encryption	-	27,539,157	-	Decryption	-	24,505,760	-
Compress	20,121,841	29,918,528	1.49	Uncompress	20,922,312	21,321,762	1.02
Total	20,121,841	57,457,685	2.86	Total	24,017,952	45,827,522	1.91

Table 2. 64-bit System Hibernation File Encryption/Decryption Performance Results(1K Cycle Count)

Process	Normal Hibernation	Encrypting Hibernation	Overh ead	Process	Normal Hibernation	Decrypting Hibernation	Over head
Encryption	-	26,583,024	-	Decryption	-	33,858,681	-
Compress	20,717,276	29,206,835	1.41	Uncompress	35,184,199	27,386,215	1.41
Total	20,717,276	55,789,859	2.69	Total	24,017,952	61,244,896	2.69

스템에서 1.49배, 64비트 시스템에서 1.41배 증가하였다. 이는 페이지 암호화로 인해 압축해야할 데이터의 크기가 증가하여 처리한 명령 코드가 증가하였음을 의미한다.

메모리 복호화 시에는 기존 최대 절전 모드 진입 시에 비해 32비트 시스템에서 1.91배의 오버헤드가 발생하였고, 64비트 시스템에서 2.69배의 오버헤드가 발생하였다. 메모리 압축 해제 시에 소요된 사이클 수는 32비트 시스템에서는 1.02배로, 64비트 시스템에서는 1.41배로 증가하였다.

3.4Ghz 프로세서에서 테스트를 진행하였으므로 사이클 수 기반으로 소요된 시간을 계산해보면 32비트 시스템에서 메모리 암호화 시 최대 절전 모드 진입 시에 5.9초에서 16.8초로 증가하였으며, 메모리 복호화 시 최대 절전 모드 재개 시에 7.0초에서 13.5초로 증가하였다. 64비트 시스템에서는 최대 절전 진입 시에 6.1초에서 16.4초로 증가하였으며, 최대 절전 모드 재개 시에 7.1초에서 18.0초로 증가하였다. 이런 최대 절전 모드 진입 및 재개 시간 증가는 메모리를 보호하기 위한 암호화 비용으로 기존 단순 데이터 압축 시간과 압축 시간에 더해지는 메모리 보호용 암호화 시간 간의 상호 교환 관계(trade-off)이기 때문에 필수적이다.

64비트 시스템에서 최대 절전 모드 파일에 기록된 180,349 개의 페이지 중 57,347 페이지가 암호화되었다. 이는 기록된 전체 페이지 중 약 31%가 프로세스 소유의 물리 메모리이며 나머지 69%가 커널 소유의 물리 메모리 페이지인 것을 나타낸다. 32비트 시스템에서는 전체 139,255 페이지 중 48,087 페이지가 암호화되어 전체 물리 페이지 중 34%가 프로세스 소유의 물리 페이지임을 나타내었다. 이 비율은 실행 중인 프로세스 수에 따라 또는 프로세스에서 사용하는 물리 메모리양에 따라 달라질 수 있다.

동일한 시스템에서 전체 디스크 암호화 도구의 오버헤드를 측정하기 위해, 베라크립트를 이용한 전체 디스크 암호화 전과 후의 디스크 성능을 벤치마크 프로그램[21]을 통해 측정하였다. 암호화 전의 시스템은 순차 읽기 시에 133.3MB/s, 순차 시에 138.3MB/s의 성능을 보였고, 암호화 후에는 순차 읽기 시에 115.6MB/s, 순차 쓰기 시에 124.8MB/s의 성능을 보였다. 이 결과로 전체 디스크 암호화로 읽기 시에 15.6%, 쓰기 시에 11.3%의 오버헤드가 발생한 것을 알 수 있다.

VI. 결론 및 향후 연구

본 논문에서는 최대 절전 모드 파일에 기록되는 프로세스의 메모리 데이터를 보호하는 방법으로 최대 절전 모드 파일 내에 기록되는 프로세스 메모리 데이터 암호화 방법을 제안하였다. 프로세스 메인 메모리

데이터를 암호화하기 위해 프로세서 선점 이후 메모리 암호화 시점과 PFN 데이터베이스 기반 암호화 메모리 선별 방법을 연구하였다.

본 논문에서 제안한 방법은 공격자가 하드 디스크 내의 최대 절전 모드 파일을 탈취하더라도 프로세스 소유의 메모리 데이터가 암호화되어 있기 때문에 프로세스 메모리 데이터가 평문 상태로 공격자에게 노출되지 않는다.

구현한 프로세스 물리 메모리 페이지 암호화 도구는 기존 최대 절전 모드 파일 생성보다 2.7배 정도의 오버헤드를 가진다. 이런 오버헤드는 전체 디스크 암호화 도구가 최대 절전 모드 처리뿐만 아니라 시스템 동작 중에도 10% 이상의 속도저하를 유발하는 상황에 비교하여, 공격자에게 프로세스의 평문 메모리 데이터가 노출되지 않기 위해 필요한 비용으로 충분히 감내할 수 있다고 판단된다.

본 논문에서 구현한 도구는 프로세스 물리 메모리 페이지를 암호화하기 위해서 사용한 암호화키가 커널 메모리에 남는 한계점이 있다. 향후 부팅 단계에서 유저가 암호화를 입력하여 암호화키를 생성하는 전용 로더를 만들거나 TPM 칩 기반 메모리 암호화, USB 키 기반 암호화키 생성 방법을 이용하여 키 문제를 해결할 예정이다. 또한 향후 본 연구 범위에 포함하지 않은 최대 절전 모드 파일에 기록되는 커널 소유 물리 메모리 암호화에 대한 연구를 진행할 예정이다.

References

[1] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation, "Advanced Configuration and Power Interface Specification," Nov. 2013.

[2] Mark E. Russinovich, David A. Solomon, and Alex Ionescu, Windows Internals, 6th Ed., Acorn Publishing Co., Volume 2, pp. 150-153, Apr. 2014.

[3] S. Mrdovic and A. Huseinovic, "Forensic Analysis of Encrypted Volumes Using Hibernation File," 19th Telecommunications forum TELFOR, pp. 1277-1280, Nov. 2011.

[4] F. Olajide, N. Savage, G. Akmayeva, and C. Shoniregun, "Digital Forensic Research - The Analysis of User Input on Volatile Memory of Windows Application," IEEE World Congress on Internet Security, pp. 231-238, Aug. 2012.

[5] S.M. Hejazi, C. Talhi, and M. Debbabi, "Extraction of forensically sensitive information from windows physical memory," Digital Investigation, vol. 6, Supplement, pp. 121-131, Sep. 2009.

[6] J.A. Halderman, S.D Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Felten, J. Appelbaum, and E.W. Felten, "Lest We Remember: Cold Boot Attacks on Encryption Keys," 17th USENIX Security Symposium, pp. 45-60, Jul. 2008.

[7] The Volatility Foundation, "The Volatility Framework," http://www.volatilityfoundation.org/, Dec. 2016.

[8] The Rekall Team, "The Rekall Memory Forensic Framework," http://www.rekall-forensic.com/, Aug. 2017.

[9] M. Suiche, "Windows hibernation file for fun 'n' profit," Black Hat, USA, Aug. 2008.

[10] Comae Technologies, "Hibr2Bin," https://github.com/comaeio/Hibr2Bin, Apr. 2017.

[11] B. Dolan-Gavitt, "Add Support for Inactive Hiberfiles to Hibinfo," https://github.com/volatilityfoundation/volatility/commit/552c1d813b05a0bf8d3d1ec1f64b3ba5f98403cc, Apr. 2009.

[12] Microsoft Technet, "BitLocker Drive Encryption," https://technet.microsoft.com/en-us/library/a2ba17e6-153b-4269-bc46-6866df4b253c, May 2010.

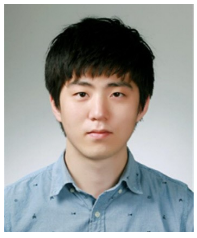
[13] idrix, "VeraCrypt," https://www.veracrypt.fr/en/Home.html, Oct. 2016.

[14] idrassi, "Hibernation File," https://veracrypt.codeplex.com/wikipage?title=Hibernation%20File, Nov. 2014.

[15] B. Lich and J. Tobin, "BitLocker frequently asked questions (FAQ)," https://docs.microsoft.com/en-us/windows/device-security/bitlocker/bitlocker-frequently-asked-questions, Apr. 2017.
[16] Truecrypt Foundation, "TrueCrypt: Free Open-Source Disk Encryption Software for Windows, Mac OS and Linux," http://truecrypt.sourceforge.net, May 2014.
[17] M. Loginova, E. Trofimenko, O. Zader eyko, and R. Chanyshev, "Program-technical aspects of encryption protection of users' data," 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), pp. 443-445, Feb. 2016.
[18] Microsoft, "[MS-XCA]: Xpress Compression Algorithm," https://msdn.microsoft.com/en-us/library/hh554002.aspx, Jun. 2017.
[19] Mark E. Russinovich, David A. Solomon, and Alex Ionescu, Windows Internals, 6th Ed., Acorn Publishing Co., Volume 2, pp. 404-408, Apr. 2014
[20] Microsoft MSDN, "ExRegisterCallback routine," https://msdn.microsoft.com/en-us/library/windows/hardware/ff545534(v=vs.85).aspx, 2017.
[21] hiyohiyo, "CrystalDiskInfo," https://crystalmark.info/download/index-e.html, Aug. 2017.

〈저자소개〉

이 경 호 (Kyoungho Lee) 학생회원
2013년 8월: 전남대학교 전자컴퓨터공학부 졸업(공학사)
2015년 8월: 전남대학교 정보보안협동과정 졸업(이학석사)
2015년 9월~현재: 전남대학교 정보보안협동과정 박사과정
〈관심분야〉 정보보호, 디지털 포렌식, 악성코드



이 우 호 (Wooho Lee) 학생회원
2016년 2월: 순천대학교 정보통신공학과(공학사)
2016년 3월~현재: 전남대학교 정보보안협동과정 석사과정
〈관심분야〉 정보보호, 사물인터넷 보안

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