Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.2, Apr. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.2.357

파일 시스템 모니터링을 통한 클라우드 스토리지 기반 랜섬웨어 탐지 및 복구 시스템*

김 주 환,† 최 민 준, 윤 주 범‡
세종대학교

Ransomware Detection and Recovery System Based on Cloud Storage through File System Monitoring*

Juhwan Kim,† Min-Jun Choi, Joobeom Yun‡
Sejong University

요 약

현대 사회의 정보 기술이 발전함에 따라 시스템의 중요 정보를 탈취하거나 파괴하는 목적을 가진 다양한 악성코드도 함께 발전하고 있다. 그 중 사용자의 자원을 접근하지 못하게 하는 대표적인 악성코드로 랜섬웨어가 있다. 암호화를 수행하는 랜섬웨어에 대해 탐지하는 연구는 최근에 계속해서 진행되고 있으나, 공격을 한 이후에 손상된 파일을 복구하는 추가적인 방안은 제안되지 않고 있다. 또한 기존 연구에서는 암호화가 여러 번에 걸쳐 진행되는 것을 고려하지 않고 유사도 비교 기법을 사용했기 때문에 정상적인 행위로 인식할 가능성이 높다. 따라서 본 논문에서는 파일 시스템을 제어하는 필터 드라이버를 구현하며, 랜섬웨어의 암호화 패턴 분석을 기반으로 검증된 유사도 비교 기법을 수행한다. 이에 접근한 프로세스의 악의적 유무를 탐지하고 클라우드 스토리지를 기반으로 손상된 파일을 복구하는 시스템을 제안하고자 한다.

ABSTRACT

As information technology of modern society develops, various malicious codes with the purpose of seizing or destroying important system information are developing together. Among them, ransomware is a typical malicious code that prevents access to user's resources. Although researches on detecting ransomware performing encryption have been conducted a lot in recent years, no additional methods have been proposed to recover damaged files after an attack. Also, because the similarity comparison technique was used without considering the repeated encryption, it is highly likely to be recognized as a normal behavior. Therefore, this paper implements a filter driver to control the file system and performs a similarity comparison method that is verified based on the analysis of the encryption pattern of the ransomware. We propose a system to detect the malicious process of the accessed process and recover the damaged file based on the cloud storage.

Keywords: ransomware, data similarity, filter driver, file system, cloud storage

I. 서 론

현대 사회의 정보 기술이 발전하면서 동시에 다양한 악의적인 목적을 가진 소프트웨어인 악성코드(malware)도 함께 발전함으로써 악성코드는 정보 보안의 위협적인 측면에서 여전히 헤드라인을 장식하

Received(12. 22. 2017), Modified(03. 06, 2018), Accepted(03. 07. 2018)
* 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2015R1C1A1A02036511)
† 주저자, kjh97852003@naver.com
‡ 교신저자, jbyun@sejong.ac.kr(Corresponding author)

고 있다. 그 중 대표적인 악성코드로는 랜섬웨어(ransomware)를 고를 수 있으며, 랜섬웨어는 최근 현대 사회에서 큰 위협으로 자리 잡고 있다[2]. 랜섬웨어란 몸값을 지불한다는 뜻과 소프트웨어의 결합으로 대상 시스템의 특정 파일(file)을 탐색해 암호화(encryption)를 하고 공격자가 요구하는 금전적인 부분을 만족시키면 복호화(decryption)를 해주는 악성코드이며, 일반적인 악성코드와 달리 행위적인 측면에서 차이가 난다. 일반적인 악성코드는 대상 시스템의 여러 정보를 탈취하는 것을 목표로 하므로 사용자가 의심할 수 없게 행동하지만 랜섬웨어는 사용자에게 감염된 사실을 알려준다[1]. 또한 운영체제의 가용성을 파괴하는 목적의 악성코드는 운영체제와 관련된 파일을 목적으로 접근하지만 랜섬웨어는 사용자의 데이터 파일을 목적으로 접근한다.

국제적으로 2016년도 상반기에는 개인과 기업에 랜섬웨어에 의한 감염이 발생하는 빈도가 각 20초마다 1명, 2분마다 1회였지만 하반기에는 각 10초마다 1명, 40초마다 1회로 큰 증가율을 보였다[3]. 또한 5개의 기업 중 1개의 기업이 금전적인 대가를 지불했지만 파일을 복구하는데 실패하였다. 2017년도에는 더욱 지능화된 새로운 랜섬웨어가 여러 등장하였고 그 중 윈도우의 SMB(Server Message Block) 취약점인 MS17-010[4]을 이용하며, 네트워크를 통해 감염이 확산되는 워너크라이(wannacry) 랜섬웨어가 화제였다[5].

이처럼 랜섬웨어에 의한 감염 사례 및 피해 규모는 지속적으로 증가하는 추세이다. 많은 보안 전문가들은 랜섬웨어의 피해 예방을 위해 물리적, 논리적으로 독립된 저장소에 주기적으로 파일을 백업하는 것이 가장 효과적인 방법이라 평가하고 있으나 모든 파일을 주기적으로 직접 백업하기는 쉽지가 않다.

랜섬웨어의 행위를 탐지하기 위해 본 논문에서는 윈도우 파일 시스템(file system)의 활동을 제어하면서 보호하는 파일에 접근하는 프로세스의 명령에 따라 1차 검증을 하고 파일의 이전 버전 리스트를 기반으로 2차 검증을 하는 시스템을 제안한다. 이 시스템을 이용하면 보호하는 파일에 접근하는 프로세스가 악의적인 접근인지 판단할 수 있으며, 악의적인 접근으로 판단 시에는 해당 프로세스를 차단하고 손상된 원본 파일이 자동 복구됨으로써 랜섬웨어에 의한 피해를 최소화할 수 있다. 2장에서는 랜섬웨어 대응 방안과 데이터 유사도 비교 기법에 대한 기존 연구와 한계점을 살펴보고 3장에서는 랜섬웨어가 파일 시스템에 접근하는 방식에 대한 분석 결과를 설명한다. 4장에서는 본 논문에서 제안하는 시스템의 구성 및 동작 방식에 대해 설명하고 5장은 주로 랜섬웨어의 공격 대상인 파일 유형에 대해 사용자에 의해 파일이 갱신됐을 때와 랜섬웨어에 의해 감염됐을 때에 각각 원본 파일과 유사도 비교 기법을 통해 측정한 결과와 이에 따른 정상적인 유사도 수치의 범위를 제시한다. 또한 다양한 파일 I/O 접근 방식을 가진 랜섬웨어에 제안하는 시스템이 정상적으로 파일을 복구하면서 악의적인 프로세스를 차단하는지 평가한다. 끝으로 6장에서는 결론을 제시한다.

II. 관련 연구

2.1 랜섬웨어 대응 방안

최근 랜섬웨어의 위협이 계속 증가하면서 이에 대한 다양한 연구가 진행되고 있다.

Kharaz 등[2]은 랜섬웨어의 특정한 행위를 분석해 랜섬웨어 공격을 완화시킬 수 있는 방안을 제안하였다. 사용자의 바탕화면을 잠그는 랜섬웨어는 특정 윈도우 API 함수를 호출하며, 사용자 파일을 대상으로 암호화나 삭제(delete) 행위를 하는 랜섬웨어는 해당 파일의 MFT(Master File Table) entry를 특정하게 변경하는 점을 분석했다. 이점을 활용해 해당 행위들을 식별하고 모니터링한다면 랜섬웨어를 탐지할 수 있음을 설명했다. 또한 디스크 내부 여러 공간에 미끼 파일을 설치해 해당 미끼 파일의 파일 시스템 활동을 모니터링하는 것도 하나의 전략이라고 설명했다.

본 논문과 유사한 연구인 Scaife 등[6]은 사용자 파일의 실시간 변경을 모니터링하여 랜섬웨어를 탐지하는 시스템을 제안하였다. 해당 시스템은 파일 시스템 필터 드라이버를 사용하여 파일에 쓰기 명령을 필터링하고 프로세스를 제어하는 커널 모드와 필터링된 사용자 파일의 변경된 구조를 분석해 접근한 프로세스의 평판 점수를 측정하는 유저 모드로 구성된다. 감시하는 사용자 파일에 쓰기 명령이 발생할 경우 고유한 특정 바이트 값인 매직 넘버(magic number)가 달라지거나 유사도 비교 기법 중 하나인 SDHASH(Similarity Digest Hash)[7]알고리즘을 사용해 이전 버전과의 유사도 수치가 낮거나 엔트로피(entropy)[8]를 측정한 수치가 이전 버전보다 높아지는 의심스러운 특징들이 나타나면 해당 시

스템은 접근한 프로세스에 대해 평판 점수를 증가시킨다. 특정 프로세스의 평판 점수가 정해진 임계값에 도달하면 해당 시스템은 의심 프로세스의 디스크 접근을 일시 중지하고 사용자에게 의심 프로세스의 접근 허가를 요청하게 된다.

Scaife 등[6]이 제안한 시스템은 랜섬웨어의 행위를 탐지하기 위해 유사도 분석 기법인 SDHASH를 사용하는 것이 본 논문과 유사하지만 실제 랜섬웨어에 의해 파일이 감염됐을 때와 사용자에 의해 직접 파일이 갱신됐을 때에 2가지 경우를 명확히 구분할 수 있는 정상적인 유사도 수치의 범위를 제시하지 못하였다. 또한 3장에서 설명할 특정 랜섬웨어는 한 번에 전체 암호화를 수행하는 것이 아닌 점진적인 암호화를 수행하는 경우가 있기 때문에 원본 파일과 유사도 비교 시 수치가 낮지 않게 나와 정상적인 행위로 인식할 수 있다. 이렇다면 원본 파일은 갱신되어 한 번 암호화된 파일이 되지만 이를 탐지하지 못한 상황이 발생하게 된다. 이에 본 논문에서는 기존 연구와 다르게 랜섬웨어와 사용자가 파일에 접근했을 때에 각각 원본 파일과 비교한 유사도 수치를 제시하며, 정상적인 유사도 수치의 범위를 분석한다. 또한 여러 랜섬웨어의 파일 시스템 접근 방식을 분석한 결과를 바탕으로 랜섬웨어의 파일 암호화 방식에 유연하게 악의적 행위를 탐지하며, 손상된 파일을 복구하는 시스템을 제안한다.

2.2 데이터 유사도 비교 기법

2.2.1 Similarity Digest Hash

Roussev[7]이 제안한 SDHASH는 서로 다른 데이터 간에 유사도를 비교할 수 있는 알고리즘이자 개발된 도구의 이름이며, 대략적인 0~100의 범위로 상관관계를 알 수 있다. SDHASH는 입력된 데이터의 특징을 선택하기 위해 입력된 데이터를 각각 64바이트만큼 나눠 엔트로피를 계산하고 우선순위 배열에 순차대로 맵핑한다. 해당 배열의 가장 왼쪽부터 8칸씩 가장 낮은 엔트로피 수치를 가진 특징에 점수를 1씩 증가시키고 오른쪽으로 1칸씩 이동하면서 반복한다. 이후에는 점수의 임계값을 넘은 특징만 선택되며, 해당 특징들은 SHA-1 알고리즘을 사용해 해시값으로 변환되고 블룸 필터(bloom filter)[13]에 삽입된다. 결과적으로 서로 다른 데이터 간에 블룸 필터끼리 비교함으로써 유사도 수치를 산출하는 방식이다.

Roussev[10]은 CTPH(Context-Triggered Piecewise Hash)[9]알고리즘과 SDHASH 알고리즘의 성능을 여러 실험을 통해 비교 평가하였다. 여러 문서 형식의 데이터를 가진 파일들을 대상으로 원본 파일과 여러 방식으로 갱신된 파일의 상관관계에 따른 정밀도와 재현율을 측정하는 실험을 하였고 큰 차이로 SDHASH가 CTPH보다 좋은 결과를 보인 것을 입증하였다. 또한 마이크로소프트 오피스와 호환되는 파일에 대해 필터링 메커니즘을 사용한다면 상당한 정확성의 이점을 제공할 수 있는 결과를 보여주었기 때문에 본 논문에서는 데이터 유사도 비교 기법 중 SDHASH 알고리즘을 채택하였다.

III. 랜섬웨어의 파일 시스템 접근 방식 분석

본 논문에서는 랜섬웨어가 파일에 접근할 시에 파일 시스템에서 일어나는 행위를 정밀하게 분석하기 위해 미니필터 드라이버(minifilter driver)를 구현하였다. 미니필터 드라이버를 통해 여러 랜섬웨어를 분석한 결과에서는 Fig.1.과 같이 5가지의 접근 패턴을 확인할 수 있었고 접근하는 파일의 공유 모드(share mode)에 접근 권한을 없애 랜섬웨어가 파일에 접근하고 있는 동안에는 해당 파일을 분석할 수 없게 하는 현상을 관찰했다. 또한 Fig.2.와 같이 파일에 쓰기 명령이 발생했을 때는 2가지의 패턴으로 암호화를 수행하는 것을 확인하였다.

Fig.1.과 같이 랜섬웨어의 5가지 접근 패턴은 모두 동일하게 원본 파일을 읽는 것으로 시작하였다.

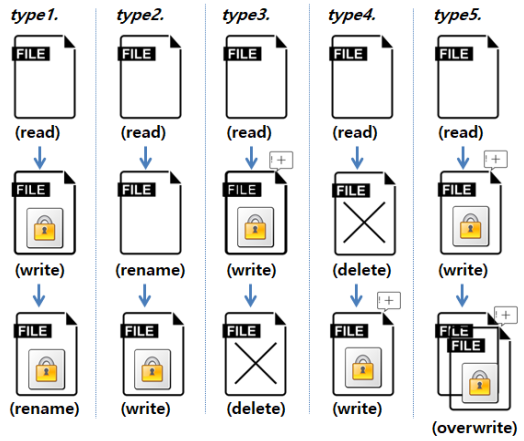


Fig. 1. Access patterns of ransomware

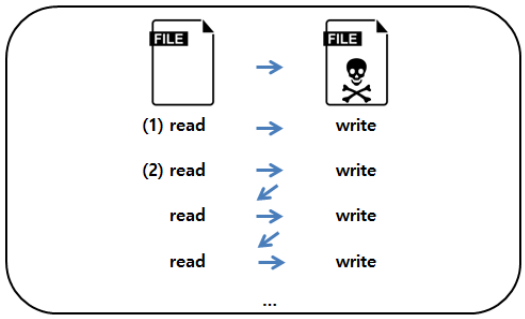


Fig. 2. Encryption patterns of ransomware

파일을 읽어 해당 파일의 내용이 담긴 버퍼를 메모리에 생성하고 암호화하여 파일의 영역에 쓰기를 하는 행위를 보였다. *type1*, *type2*와 *type3*, *type4*의 결과는 같지만 IRP의 일어나는 순서가 공통적인 읽기 이후에는 서로 반대인 경우를 나타냈다. *type1*은 원본 파일의 기존 영역에 미리 읽어놨던 암호화 된 버퍼를 씌운 다음 파일의 이름을 변경하였고 *type2*는 원본 파일의 이름을 먼저 변경하고 암호화 된 버퍼를 기존 영역에 씌웠다. *type5*는 새 파일의 영역에 미리 읽어놨던 암호화 된 버퍼를 씌운 다음 원본 파일에 덮어쓰기를 하는 행위를 보였다. 이러한 경우들은 파일의 영역이 중복되기 때문에 파일 복구가 어렵다.

*type3*은 새 파일의 영역에 미리 읽어놨던 암호화 된 버퍼를 씌운 다음 원본 파일을 삭제하였고 *type4*는 원본 파일을 먼저 삭제하고 새 파일의 영역에 암호화된 버퍼를 씌웠다. 이러한 경우들은 원본 파일의 영역을 토대로 파일을 복구할 수 있지만[1], 파일 복구의 가능성을 없애기 위해 파일의 영역 내에 의미 없는 버퍼로 쓰기 작업을 한 후 삭제하는 행위도 보였다.

Fig.2.와 같이 파일에 쓰기 명령이 발생하면서 암호화가 되는 2가지의 방식을 관찰했다. (1)은 원본 파일에 읽기 명령이 발생하고 생성된 버퍼에 암호화를 한 후 암호화된 버퍼를 목표 파일 영역에 쓰는 작업이 한 번에 이루어졌으나, (2)에서는 원본 파일을 읽고 생성된 버퍼에 암호화를 한 후 암호화된 버퍼를 목표 파일 영역에 쓰는 작업이 여러 번에 걸쳐 수행되었다. 파일의 크기에 따라 쓰기의 횟수가 각각 달랐으며, 크기가 클수록 쓰기의 횟수가 많았다. 이는 단순히 프로그램이 파일을 읽을 때, 읽는 버퍼의 크기를 고정해서 읽었거나 동적으로 파일의 크기만큼 버퍼를 할당받아 한 번에 읽었느냐의 차이지만 이전 버전과의 유사성을 비교할 때는 큰 결함을 가져올 수 있다. 원본 파일이 한 번에 전체 암호화가 수행되는 경우에는 이전 버전과의 유사도 수치가 매우 낮지만 점진적으로 암호화가 수행되는 경우에는 이전 버전과의 유사도 수치가 점차 낮아지기 때문에 부분 암호화가 된 파일을 정상으로 인식하고 갱신될 수 있다.

IV. 제안 시스템

본 논문에서 제안하는 시스템은 Fig.3.과 같이 크게 클라이언트와 클라우드 서버로 구성된다. 클라이언트는 커널 모드에서 동작하는 미니필터 드라이버와 이와 통신하는 유저 모드 프로그램으로 구성되어 있으며, 1차 검증을 수행한다. 필터 드라이버는 보호하는 파일에 대한 명령을 감시하여 여러 정보를 유저 모드 프로그램에게 전송하며, 유저 모드 프로그램은 드라이버로부터 받은 정보로 악의적 행위를 판별하고 반환한다. 이때, 1차 검증이 정상적으로 판단되었다면 클라우드 서버를 통해 2차 검증이 수행되며, 클라우드 서버의 저장소는 특정 파일마다 이전 버전별로 구성된 리스트들을 가지고 있다. 앞서 언급했듯이, 점진적인 암호화를 수행하는 특정 랜섬웨어가 존재하므로 이를 탐지하기 위해 2차 검증을 설계하였다.

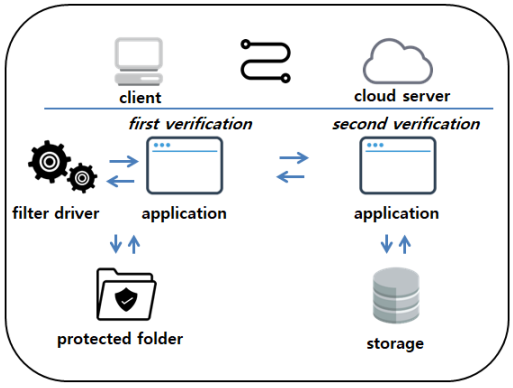


Fig. 3. System overview

4.1 1차 검증 모델

보호하고 있는 파일에 접근이 발생하게 되면 배치되어 있던 클라이언트의 미니필터 드라이버는 IRP

가 파일 시스템 드라이버에 도달하기 전에 전위(preoperation) 함수를 호출하게 되며, 해당 파일의 공유 모드 접근 권한을 확인한다. 파일의 핸들이 생성되면 IRP_MJ_CREATE 패킷이 발생하며, 해당 패킷을 통해 핸들의 공유 모드 접근 권한을 확인할 수 있다. 공유 모드의 읽기 권한이 제한되어 있으면 클라이언트의 유저 모드 프로그램이 특정 프로세스가 접근하고 있는 동안 해당 파일을 분석을 할 수 없으므로 공유 모드의 읽기 권한을 확인하고 접근할 수 있게 파일의 핸들을 제어한다. 그 다음에는 발생한 "IRP 명, 접근한 프로세스 식별자, 접근한 파일의 경로"를 구조체에 담아 유저 모드 프로그램에게 전송하며, 응답이 오기 전까지는 발생한 IRP를 반환하지 않고 대기한다. 응답이 오면 유저 모드 프로그램의 처리에 따라 해당 IRP를 현재 드라이버에서 완료할 것인지, 파일 시스템 드라이버에 도달한 후에 호출되는 후위(postoperation) 함수에 전달할지, 하위 드라이버에게 전달하고 끝낼지의 여부를 반환 값을 통해 정하게 된다. 미니필터 드라이버의 후위 함수에 IRP가 전달이 된다면 전위 함수와 동일한 작업을 수행하며, IRP를 완료한다. 특정 파일의 IRP가 현재 미니필터 드라이버에서 반환되지 않으면 접근한 프로세스의 행위가 잠깐 멈추기 때문에 유저 모드 프로그램이 응답하기 전까지는 다른 파일의 접근은 일어나지 않는다.

Fig.1.의 *type1*, *type2*처럼 원본 파일에 쓰기 명령이 발생한 경우 유저 모드 프로그램은 해당 파일

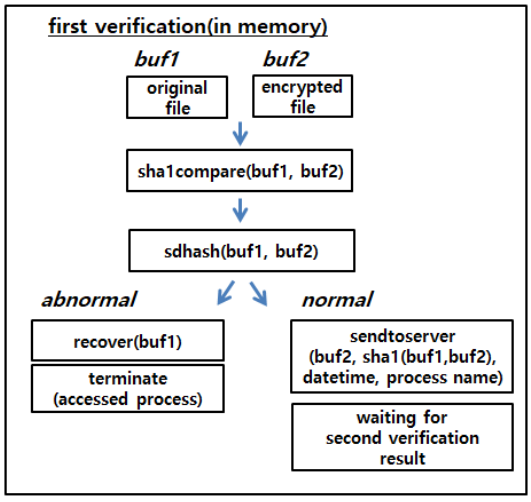


Fig. 4. Detection method of the first verification model

의 크기만큼 동적 할당받은 buf1을 메모리에 생성하고 파일의 내용을 복사한다. 또한 SHA-1 해시 알고리즘에 buf1을 삽입해 해시 값을 추출함으로써 "buf1, 해시 값"을 한 쌍으로 메모리에 가지고 있는다. 이후에 미니필터 드라이버의 후위 함수에 해당 IRP가 전달되도록 요청 및 응답을 한다. IRP는 파일 시스템 드라이버에 도달한 후에 다시 유저 모드 프로그램에게 해당 파일에 대한 정보를 전송하며, 유저 모드 프로그램은 쓰기가 된 해당 파일에 대한 buf2를 만들고 해시 값을 생성해 Fig.4.와 같이 미리 생성한 buf1의 해시 값과 비교한다. 해시 값이 다를 경우 SDHASH의 알고리즘을 사용해 미리 생성한 buf1과 buf2의 유사도를 비교한다. 5장에서 구체적으로 설명할 실험 결과에서 유사도 수치가 9 이하일 경우 악의적인 행위로 판단했으므로 두 파일 간에 유사도 수치가 9이하로 나오면 메모리에 생성한 buf1으로 파일을 복구하고 해당 파일에 접근한 프로세스를 강제 종료한다. 이후 드라이버에 응답과 함께 IRP의 완료를 요청한다. 유사도 수치가 정상적인 범위에 속한 경우에는 2차 검증을 위해 "buf2, buf1과 buf2의 해시 값, 현재 날짜와 시간, 접근한 프로세스 명"을 클라우드 서버에 전송한다. 클라이언트의 유저 모드 프로그램은 2차 검증에 대한 완료 신호를 받기 전까지 드라이버에 응답을 하지 않고 기다린다.

Fig.1.의 *type3*, *type4*처럼 특정 랜섬웨어의 설계에 따라 쓰기가 먼저 발생하거나 삭제가 먼저 발생할 수 있으므로 2가지의 패턴을 모두 고려해 랜섬웨어를 탐지한다. 새 파일에 쓰기 명령이 발생하게 되면 순차적으로 파일이 생성이 되고 쓰기가 진행이 된다. 이때, 쓰기 명령에 대한 IRP가 파일 시스템 드라이버에 도달하기 전에 전위 함수가 호출되므로 이때의 새 파일 크기는 0바이트이다. 마찬가지로 Fig.1.의 *type5*도 원본 파일에 덮어쓰기를 하기 전에 새 파일이 생성되고 쓰기 명령이 발생하므로 이때의 새 파일 크기는 0바이트이다. 이점을 이용해 유저 모드 프로그램은 해당 파일의 내용을 메모리에 복사하기 전에 크기를 확인해 0바이트인 경우 사용자에게 해당 명령이 정상적인 접근인지 요청하고 드라이버에 응답을 한다. 파일에 삭제 명령이 발생하는 경우는 사용자가 시스템을 사용하면서도 자주 발생하는 명령이기 때문에 프로세스 식별자를 사용해 처리한다. 보호하고 있는 파일에 삭제 명령이 발생하면 유저 모드 프로그램은 해당 파일에 접근한 프로세스

가 윈도우 환경의 GUI 쉘인 파일 탐색기 프로세스인지 확인한다. 접근한 프로세스가 파일 탐색기 프로세스인 경우에는 정상적인 접근으로 판단하며, 아닌 경우에는 사용자에게 해당 명령이 정상적인 접근인지 요청하고 드라이버에 응답한다. 사용자가 새 파일에 발생한 쓰기 명령이나 보호하는 파일에 삭제 명령을 비정상적인 접근으로 판단한 경우 드라이버는 Fig.5.와 같이 유저 모드 프로그램으로부터 반환 값을 받고 이 값이 TRUE일 경우 해당 파일의 접근을 거부하며, 발생한 IRP를 미니필터 드라이버 스택 아래로 보내지 않고 완료시킨다.

```
if(status == STATUS_SUCCESS)
{
    SafeToOpen = ((PFLT_REPLY)notification)->SafeToOpen;

    if(SafeToOpen == TRUE)
    {
        Data->IoStatus.Status == STATUS_ACCESS_DENIED;
        Data->IoStatus.Information == 0;
        Return = FLT_PREOP_COMPLETE;
    }
    else
    {
        Return = FLT_PREOP_SUCCESS_NO_CALLBACK;
    }
}
...
return Return;
```

Fig. 5. Handling code in case of abnormal accesses

4.2 2차 검증 모델

클라우드 서버의 저장소는 쓰기 명령이 발생한 하나의 특정 파일에 대해 n개의 노드로 구성된 리스트를 가질 수 있으며, 각 노드는 파일이 갱신된 시간에 따라 순차적으로 연결된다. 노드 안에는 "파일, 해시 값, 날짜와 시간, 접근한 프로세스 명"이 포함된다.

쓰기 명령이 발생한 1차 검증의 결과가 정상일 경우 클라우드 서버에서 2차 검증이 수행되며, Fig.6.과 같이 먼저 클라이언트로부터 받은 buf1의 해시 값을 검색해서 해당 해시 값을 가진 노드를 찾는다. 즉, 해당 파일에 대해 마지막으로 갱신된 노드를 찾음으로써 이전 버전이 순차적으로 구성된 리스트를 찾게 된다.

2차 검증으로 악의적인 행위를 탐지하는 방법은 2가지로 수행된다. 1번째는 Fig.7.과 같이 buf2 파일에 접근한 프로세스 이름과 미리 찾은 노드 n번째

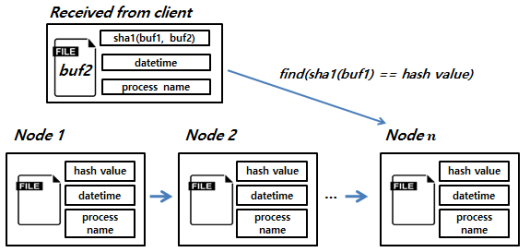


Fig. 6. A linked-list of files

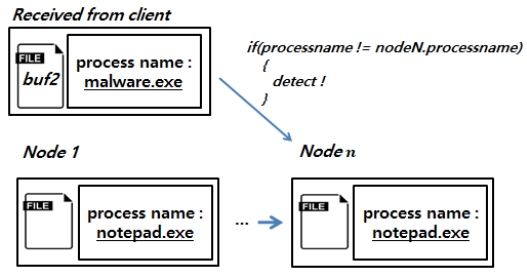


Fig. 7. 1st Detection method of the second verification model

에 저장된 프로세스 이름을 비교한다. Fig.7.의 경우 지금까지 notepad.exe 프로세스가 특정 파일에 접근하였기 때문에 이전 버전의 모든 노드에 notepad.exe로 저장되어 있지만 클라이언트로부터 받은 새로운 정보의 프로세스 명이 malware.exe 이므로 비정상적인 접근으로 탐지하게 된다. 2번째는 Fig.8.과 같이 이전 버전의 모든 파일과 받은 buf2와 유사도 비교를 각각 수행한다. 5장의 실험 결과를 바탕으로 유사도 수치가 9이하인 경우가 발

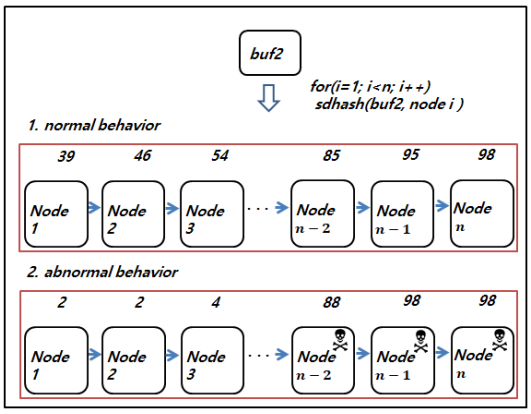


Fig. 8. 2nd Detection method of the second verification model

생하면서 1번째 노드의 파일부터 n번째 노드의 파일까지 유사도의 수치가 급격히 낮아지다 높아지는 통계적 기반의 이상 현상이 보이면 비정상적인 접근으로 판단한다. 예를 들어 Fig.8.의 1번을 보면 오래된 노드일수록 유사도 수치가 낮았지만 계속해서 갱신된 n번째 노드까지는 9이하의 유사도 수치가 출력되지 않았으므로 정상 행위로 판단한다. 2번의 경우는 점진적인 암호화가 진행되고 있으며, 이미 n, n-1, n-2번째 노드의 파일은 부분 암호화가 진행되었지만 전체 암호화가 끝나지 않아 유사도 수치로 탐지하지 못한 상태를 가정했다. 암호화가 진행된 파일끼리는 유사도 수치가 높게 나오기 때문에 부분 암호화가 된 n, n-1, n-2번째 노드의 파일들은 수치가 높았지만 암호화가 진행되지 않은 이전 버전과의 유사도 수치가 9이하인 경우가 발생했으므로 비정상 행위로 판단하게 된다.

2차 검증에서 비정상 행위로 판단했을 경우 기다리고 있는 클라이언트에게 이를 알리며, 해당 파일의 이전 버전을 가진 리스트를 제공한다. 사용자는 노드내 날짜와 시간을 보고 복구 지점을 선택할 수 있게 된다. 이후 복구에 이어 클라이언트의 유저 모드 프로그램은 해당 프로세스를 강제 종료하고 드라이버에 응답하므로 드라이버는 IRP를 완료한다. 정상 행위로 판단했을 경우 클라이언트로부터 받은 정보는 n번째 노드에 이어 새로운 노드로 구성되어 추가된다. 하지만 이는 검증 시마다 정상일 경우 노드가 추가된다면 클라우드 시스템의 특성 상 공간 효율성에 대한 문제가 야기될 수 있다. 이에 Sangmin Ha 등[14]은 최대 용량 제한 및 최대 저장 횟수를 지정해 초기 저장 파일부터 순차적으로 지우고 최근 파일을 저장하는 방식을 적용했다. 본 논문에서 제안하는 시스템도 공간 효율성을 고려해 사용자가 주기를 지정하면 저장소에 보관하고 있는 모든 파일에 대해 각각 최근 5개 노드만 남기고 자동으로 삭제되게 설계했다.

V. 실 험

본 실험에서는 특정 파일을 대상으로 사용자가 실제 작업을 한 듯이 갱신됐을 때와 실제 랜섬웨어에 의해 암호화됐을 때에 각각 원본 파일과의 유사도 수치를 측정한 분석 결과를 제시하며, 유사도 수치의 정상적인 범위에 따른 임계값을 증명한다. 또한 다양한 파일 I/O를 발생시키는 랜섬웨어(20개의 패밀리로 170개의 샘플)에 제안하는 시스템이 파일의 손상을 막으며, 차단하는 것을 평가한다.

5.1 실험 환경

앞서 언급한 실험을 위해서는 여러 확장자를 가진 파일이 필요하기 때문에 저작권이 없고 공개적으로 사용이 가능한 GovDocs[11] 파일 샘플에 대해 Roussev[10]이 축약한 t5라고 불리는 파일 세트를 사용했다. 해당 파일 세트는 .xls, .ppt, .doc, .txt, .jpg, .gif, .pdf, .html 확장자를 가진 파일로 구성되어 있다. 본 실험에서는 .html, .txt 확장자를 가진 파일은 제외하고 여러 크기(114KB~5.40MB)의 파일을 3개씩 수집해 총 18개의 파일 세트를 구성했다.

랜섬웨어 샘플들은 Payload-Security[12]분석 플랫폼에 제출된 파일들에서 수집하였으며, 가상 머신 환경에서 샘플을 실행하여 동작의 유무를 확인하고 총 202개의 랜섬웨어를 패밀리 별로 분류하였다. 이중 32개 샘플은 가상 머신의 환경을 인식하고 종료되거나 암호화가 수행되지 않고 메모리에 오래 상주하거나 운영체제의 버전과 호환되지 않는 현상들을 보여 삭제하였다. 랜섬웨어에 대응하는 좋은 평가를 위해서는 다양한 파일 I/O를 발생시키는 패턴을 확보해야 하므로 샘플의 개수보다 패밀리의 개수가 더 중요하다. 이에 총 분류된 랜섬웨어 패밀리는 20개이다.

시스템의 평가를 하는 실험에서는 가상 머신 환경을 사용하였다. 가상 머신의 운영체제는 윈도우 10(32bit)을 사용하였고 메모리는 3GB를 할당하였다. 또한 랜섬웨어 감염에 손쉽게 노출될 수 있게 관리자 계정으로 샘플을 실행하였고 사용자 계정 컨트롤과 내장된 윈도우 디펜더의 실시간 감시와 방화벽을 껐으며, 일부 랜섬웨어는 닷넷 프레임워크를 요구하므로 설치하였다. 이러한 환경을 설정한 상태의 스냅샷을 만들었으며, 하나의 샘플을 실험하고 해당 상태로 돌려 작업을 반복하였다.

5.2 유사도 수치의 임계값 측정

공통적으로 유사도 수치를 측정할 때는 하나의 확장자마다 서로 다른 크기인 3개의 파일로 실험했다.

사용자에 의한 유사도 측정에서는 .jpg, .gif, .pdf 확장자를 가진 파일은 주로 사용자가 읽는 파

일이므로 제외하였다. 랜섬웨어에 의한 유사도 수치 측정에서는 한 번에 암호화가 되는 경우와 점진적으로 암호화가 되는 경우의 결과가 같으므로 점진적인 암호화를 수행하는 대표적인 랜섬웨어인 Cerber로 실험하였다.

특정 파일이 사용자에 의해 갱신된 경우의 시나리오는 2가지이며, 원본 파일에 계속해서 갱신된 10개의 파일을 만들어 각각 원본 파일과 유사도 비교를 수행하였다. 원본 파일의 기존 양식은 수정하지 않았으며, 기존 내용만 수정하거나 새로운 내용을 처음, 중간, 끝에 삽입만 하였다. Fig.9.의 실험 결과와 같이 왼쪽 열 (1)은 사용자가 원본 파일 내용의 끝에만 문자나 여러 개체를 삽입할 때마다 수정된 버전을 만들어 순차적으로 갱신된 각 버전과 원본 파일을 유사도 비교한 것이고 오른쪽 열 (2)는 원본 파일 내용의 처음, 중간 위치에 문자나 여러 개체를 삽입, 수정, 삭제하여 (1)과 동일하게 수행한 결과이다. 특정 파일이 랜섬웨어에 의해 암호화가 진행될 경우 Fig.10.은 점진적으로 암호화가 된 경우의 실험 결과이며, 부분 암호화가 될 때마다 각각 원본 파일과 유사도 비교를 수행하였다.

Fig.9.와 Fig.10.의 결과에 따른 .xls(5,539KB) 파일은 사용자가 갱신했을 경우 최대 17, 최소 15의 유사도 결과가 산출되었으며, 랜섬웨어에 의해 암호화가 진행될 때는 최대 74, 최소 4의 유사도 결과가 산출되었다. 만약 유사도 수치의 정상 범위 임계값을 분명하게 정하지 않고 판별하였

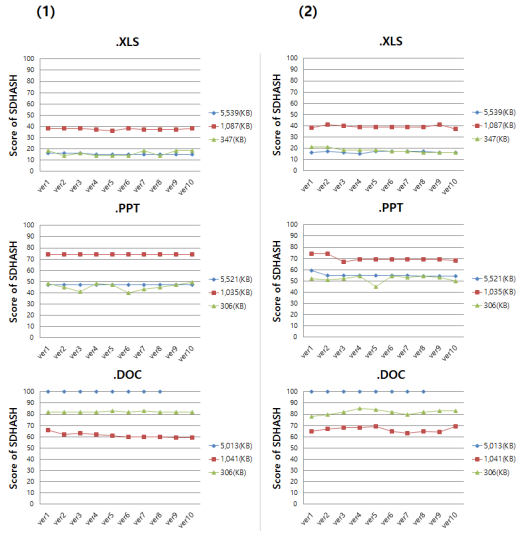


Fig. 9. Files updated by a user

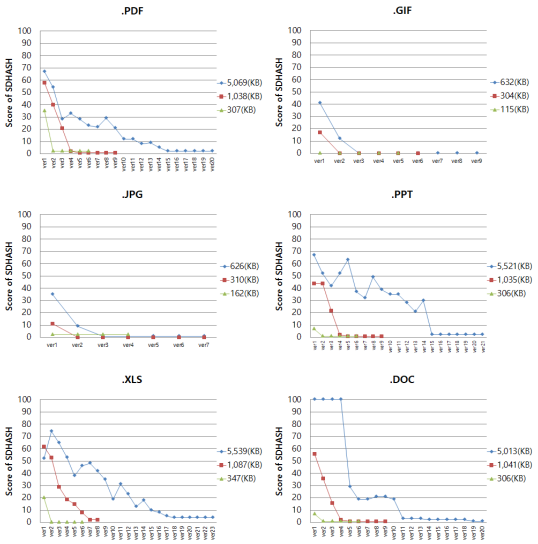


Fig. 10. Files encrypted by ransomware

다면 사용자에 의해 갱신된 경우에도 불구하고 최소 수치가 객관적으로 보았을 때 높은 결과가 아니므로 비정상적인 접근으로 판단할 수 있다. 또한 점진적으로 암호화가 수행되는 경우에는 결과적으로 유사도 수치가 4까지 낮아졌지만 암호화가 진행되는 동안에는 최대 유사도 수치가 74였기 때문에 점진적인 암호화를 고려하지 않고 유사도 비교를 하였으면 정상적인 접근으로 판단할 가능성이 높다. .doc(5,013KB) 파일은 사용자에 의한 10번의 갱신에도 불구하고 원본 파일과 유사도 비교 시에 수치가 100으로 계속 산출되었다. 랜섬웨어에 의해 암호화가 진행될 때도 네 번까지 유사도 수치가 계속 100으로 산출되어 false-positive가 발생할 가능성이 높았지만 다섯 번부터 급격히 수치가 낮아졌고 열 한 번째에는 수치가 10미만으로 낮아졌다.

결과적으로 사용자에 의한 쓰기 명령에 각각의 파일들은 유사도 수치의 최솟값이 15부터 100까지 파일 특성마다 상이했지만 처음 갱신된 버전과 계속해서 갱신된 버전과의 유사도 수치는 크게 차이가 없었다. 랜섬웨어에 의한 암호화에는 유사도 수치의 최솟값이 서로 다른 파일 확장자를 가졌음에도 0부터 4로 산출되었다. 이는 암호화가 되면 파일 특성에 종속적이지 않고 파일의 본질을 확연히 잃는 것을 알 수 있었다.

여러 파일 확장자를 대상으로 랜섬웨어에 의해 유사도 수치를 측정 시, 유사도 수치는 0~4로 점차

떨어지지만 사용자가 갱신 시 유사도 수치의 최솟값은 15였다. 이에 랜섬웨어를 대응하기 위해 유사도 수치의 비정상 범위를 5미만으로 정해도 되지만 점진적으로 암호화가 진행되는 랜섬웨어 특성 상 유사도 수치가 점차 낮아지는 것이기 때문에 5미만이 아니면 계속해서 2차 검증을 수행할 것이고, 노드도 계속해서 추가되므로 공간 효율성도 떨어지면서 시스템에 오버헤드가 커질 수 있다. 이에 더 효율적인 유사도 수치의 비정상 범위를 고려하여 위 실험 결과를 바탕으로 사용자가 접근했을 때 최소 유사도 수치 값인 15보다 작으면서 랜섬웨어에 의해 최종 암호화된 파일들의 유사도 수치 값 중 최댓값인 4보다는 큰 값을 정하기 위해 두 값의 평균인 9이하를 비정상 범위의 임계값으로 정하였다.

5.3 랜섬웨어 대응 평가

본 논문에서 제안하는 시스템은 Table 1과 같이 총 170개의 샘플 중 168개(98.8%)를 악의적으로 탐지했으며, 탐지된 샘플에 의해 감염된 파일을 100%로 복구하였다. 원본 파일에 암호화가 점진적으로 발생했을 때는 2차 검증 모델이 주로 기여를 했으며, 전체 암호화가 한 번에 발생했을 때는 1차 검증 모델이 주로 기여를 했다. 암호화된 파일들은 원본 파일과 유사도 비교 시 0부터 4까지 산출되므로 정상 범위의 임계값이 4보다 크기만 해도 위 Table 1과 동일한 탐지율을 보였겠지만 사용자에 의한 접근을 구분해야 하므로 유사도 수치의 비정상 범위를 고려할 필요가 있다. 이에 유사도 수치의 비정상 범위 임계값은 위 실험 결과에 따라 9이하로 수행하였다.

Table 1. 170 samples of 20 different ransomware families

Ransomware Family	No of Samples	Detection Rate
Cerber	73(42.9%)	73/73
Locky	31(18.2%)	31/31
Troldesh	17(10%)	17/17
Shade	8(4.7%)	8/8
FileLocker	7(4.1%)	7/7
Sage	7(4.1%)	7/7
CryptoMix	5(2.9%)	5/5
TeslaCrypt	5(2.9%)	5/5
Globe	3(1.7%)	3/3
MRCR	2(1.1%)	2/2
YOUGOTHACKED	2(1.1%)	2/2
iRansom	2(1.1%)	0/2
Chimera	1(0.5%)	1/1
CryptoLocker	1(0.5%)	1/1
CryptoWall	1(0.5%)	1/1
CryptXXX	1(0.5%)	1/1
Fantom	1(0.5%)	1/1
Jigsaw	1(0.5%)	1/1
xorist	1(0.5%)	1/1
Maktub	1(0.5%)	1/1
Total	170	168/170

클라이언트는 초기 환경을 구성할 때 보호할 폴더를 C:\ 아래에 배치를 했다. iRansom 랜섬웨어는 바탕화면에 있는 파일들만 대상으로 암호화를 하는 특성을 가졌으며, 이를 탐지하기 위해 바탕화면 폴더를 보호하면 탐지를 할 수 있다. 하지만 바탕화면은 사용자가 자주 접근하는 곳이기 때문에 배치를 하게 되면 시스템의 오버헤드 가능성이 높아지므로 그대로 실험을 하였으며, 본 논문의 시스템은 해당 랜섬웨어만 탐지를 못하였다.

VI. 한계점

본 논문에서 제안한 시스템은 랜섬웨어의 행위를 탐지하기 위해 유사도 비교 기법을 부분적으로 사용하기 때문에 false-positive의 가능성이 있다. 5장에서는 원본 파일이 사용자에 의해 갱신될 때 시나리오를 정하고 실험했지만 이는 원본 파일의 폼 양식은 접근하지 않고 부분적인 내용만 접근했었기 때문에 내용뿐만이 아니라 폼 양식까지 수정하거나 내용을 초기화하고 다시 작성하는 등의 대량 수정 시에는 사용자가 접근했을 지라도 비정상 범위의 임계값이 출력될 수 있다. 또한 확산 기능이 있는 파일에 대해 사용자가 갱신 시 전체 내용이 바뀔 수 있기 때문에 이 경우도 false-positive가 발생할 수 있다. 예를

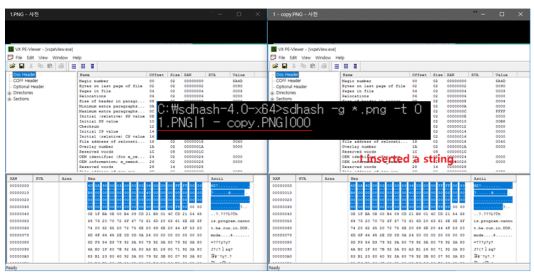


Fig. 11. Write command to file with diffusion function

들어 Fig.11.과 같이 그림 파일 복사본에는 문구를 삽입하고 원본 파일과 유사도 비교 시 0의 수치가 산출되었다.

VII. 결 론

최근 보안의 헤드라인을 장식하고 있는 랜섬웨어는 공격 기법이 나날이 발전하고 있으며, 그로 인한 피해 또한 계속해서 증가하는 추세이다. 이에 본 논문에서는 암호화를 수행하는 랜섬웨어에 대해 파일 시스템을 모니터링 및 제어하는 메커니즘과 유사도 비교 분석 기법을 결합하여 보호하는 파일에 발생하는 명령이 악의적인 접근인지 유무를 판단할 수 있는 랜섬웨어 대응 시스템을 제안하였다. 악의적인 접근으로 판단 시, 탐지뿐만이 아닌 파일의 복구가 가능하다. 또한 실험을 통해 파일이 사용자에 의해 갱신되었을 때와 랜섬웨어에 의해 갱신되었을 때에 각각 원본 파일과의 유사도 비교를 통해 정상적인 유사도 수치 범위를 제시하였으며, 실제 랜섬웨어(20개의 패밀리로 170개의 샘플)의 공격에 본 논문에서 제안하는 시스템이 정상적으로 파일을 복구하면서 악의적인 프로세스를 차단하는지 평가하였다. 본 논문에서 제안하는 시스템은 사용자의 중요한 파일들을 보호하고 공격자의 행위를 차단하므로 향후 랜섬웨어 대응 연구에 활용될 수 있을 것으로 기대한다.

References

[1] A. Kharraz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "Unveil: a large-scale, automated approach to detecting ransomware," 25th USENIX Security Symposium, pp. 757-772, Aug. 2016.

[2] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: a look under the hood of ransomware attacks," Detection of Intrusions and Malware, and Vulnerability Assessment:12th International Conference, LNCS 9148, pp. 3-24, 2015.

[3] Boan News, http://www.boannews.com/media/view.asp?idx=52688

[4] Microsoft Tech Report, https://technet.microsoft.com/en-us/library/security/ms17-010.aspx

[5] AhnLab Tech Report, http://asec.ahnlab.com/1067

[6] N. Scaife, H. Carter, P. Traynor, and K.R.B. Butler, "Cryptolock (and drop it): stopping ransomware attacks on user data," 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp.303-312, June. 2016.

[7] V. Roussev, "Data fingerprinting with similarity digests," Sixth IFIP WG 11.9 International Conference on Digital Forensics, pp.207-226, Jan. 2010.

[8] J. Lin, "Divergence measures based on the shannon entropy," IEEE Transactions on Information Theory, vol. 37, no. 1, pp.145-151, Jan. 1991.

[9] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," Digital Investigation, vol. 3, no. 9, pp. 91-97, Sep. 2006.

[10] V. Roussev, "An evaluation of forensic similarity hashes," Digital Investigation, vol. 8, no. 8, pp. 34-41, Aug. 2011.

[11] Govdocs1 Report, https://digitalcorpora.org/corpora/files

[12] Malware Analysis Site, https://www.payload-security.com/

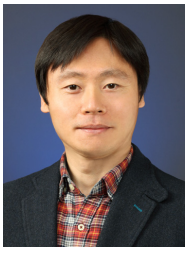
[13] B. Bloom, "Space-Time Trade-offs in Hash Coding with Allowable Errors," Communications of the ACM, vol. 13, no. 7, pp. 422-426, Nov. 1970.

[14] Sangmin Ha, Taehoon Kim and Souhwan Jung, "Design and Implementation of a Cloud-Based Recovery System against Ransomware Attacks", Journal of The Korea Institute of information Security & Cryptology, 27(3), pp. 521-530, Jun. 2017

〈저자소개〉

김 주 환 (Juhwan Kim) 학생회원
2016년 2월 : 학점은행제 정보보호학 전공 학사
2017년 3월~현재 : 세종대학교 일반대학원 정보보호학과 석사과정
〈관심분야〉 악성코드 분석, 시스템 보안, 소프트웨어 취약점

최 민 준 (Min-Jun Choi) 학생회원
2014년 2월: 인천대학교 메카트로닉스공학과, 컴퓨터공학부 복수전공 학사
2016년 9월~현재: 세종대학교 일반대학원 정보보호학과 석사과정
〈관심분야〉 네트워크 보안, 임베디드 보안



윤 주 범 (Joobeom Yun) 종신회원
1999년 2월: 고려대학교 컴퓨터학과 학사
2001년 2월: 서울대학교 컴퓨터공학과 석사
2012년 2월: KAIST 전산학과 박사
2001년 3월~2015년 2월 : ETRI부설연구소 선임연구원
2015년 3월~현재: 세종대학교 정보보호학과 조교수
〈관심분야〉 네트워크 보안, 시스템 보안, 클라우드 컴퓨팅 보안