Journal of the Korea Industrial Information Systems Research Vol.27 No.2, Apr. 2022 :1-10
http://dx.doi.org/10.9723/jksiis.2022.27.2.001 ISSN:1229-3741

파일 은닉을 통한 파일 대상 공격 방어 기법

(A Defense Mechanism Against Attacks on Files by Hiding Files)

최 지 원[1], 이 중 희[2]*, 이 규 호[3], 유 재 관[3], 박 아 란[3]
(Jione Choi, Junghee Lee, Gyuho Lee, Jaegwan Yu, and Aran Park)

요 약 기만 기술(deception technology)은 허니팟(honeypot)의 확장된 개념으로, 공격자를 기만하여 공격을 탐지, 방지, 또는 지연시키는 기술을 의미한다. 기만 기술은 네트워크 포트, 서비스, 프로세스, 시스템 콜, 데이터베이스 등에 폭넓게 적용되어 왔다. 파일을 대상으로 하는 공격에도 유사한 개념을 적용할 수 있다. 파일을 대상으로 하는 대표적인 공격으로 랜섬웨어가 있다. 랜섬웨어는 사용자의 파일을 몰래 암호화한 후 값을 지불해야 복원할 수 있게 해 주는 멀웨어의 일종이다. 또 다른 예로는 와이퍼 공격으로 시스템에 있는 모든 또는 타겟 파일을 복구 불가능하게 삭제하는 공격이다. 본 논문에서는 이러한 종류의 공격에 대응하기 위한 방법으로 파일을 은닉하는 방법을 제안한다. 파일을 은닉하는 방법은 기존의 백업이나 가상화를 통한 파일 보호 기법에 비해 디스크 용량을 추가로 소비하지 않고 성능 저하도 최소화할 수 있는 장점이 있다.

핵심주제어: 기만 기술, 파일 대상 공격, 랜섬웨어, 파일 은닉

Abstract Deception technology is an extended concept of honeypot, which detects, prevents or delays attacks by deceiving adversaries. It has been applied to various system components such as network ports, services, processes, system calls and database management systems. We can apply the same concept to attacks on files. A representative example of a file attack is ransomware. Ransomware is a type of malware that encrypts user files and ask for ransom to recover those files. Another example is the wiper attack, which erases all or target files of a system. In this paper we propose a defense mechanism against these kinds of attacks by hiding files. Compared to backup or virtualization techniques, the proposed method incurs less space and performance overheads.

Keywords: Deception technology, Attacks on files, Ransomware, Hiding files

1. 서 론

디지털 기기의 사용이 늘어나면서 주요 정보자산들이 파일의 형태로 저장되고 관리되는 경우가 많아졌다. 이에 따라 디지털 자산에 대한 공격도 늘어나고 있다. 대표적인 예가 랜섬웨어(ransomware)이다. 랜섬웨어는 사용자의 파일을 몰래 암호화하여 사용자가 접근을 못하도록 하고 몸값을 요구하는 멀웨어(malware)이다. 만약

* Corresponding Author: j_lee@korea.ac.kr
Manuscript received February 14, 2022 / revised April 01, 2022 / accepted April 19, 2022
1) 고려대학교 정보보호대학원, 제1저자
2) 고려대학교 정보보호대학원, 교신저자
3) LIG 넥스원

사용자가 파일을 백업해 놓지 않았다면 파일 복구를 위해 몸값을 지불할 수 밖에 없다. 또 다른 예로는 와이퍼(wiper) 공격으로 시스템에 있는 모든 또는 타겟 파일을 복구 불가능하게 삭제하는 공격이다. 마찬가지로, 파일이 백업되어 있지 않으면 복구가 불가능하다.

디지털 자산을 보호하기 위한 기존의 방법은 크게 백업, 접근 제어, 가상화, 디스크 분할로 구분해 볼 수 있다. 백업은 같은 내용의 파일을 별도의 공간에 중복 저장하는 것이다. 이 방법은 중복 저장으로 인해 디스크 공간이 효율적으로 사용되지 못한다는 단점이 있다. 특히, 백업할 파일이 크기가 크거나 많다면 디스크의 상당 부분을 백업에 할당해야 한다. 멀웨어도 진화하고 있어서 최근 발견되는 멀웨어 중에는 백업된 파일도 찾아서 삭제하는 경우도 있다. 접근 제어는 허가 받은 사용자만 파일에 접근할 수 있도록 제어하는 것이다. 그러나, 시스템의 취약점을 이용하여 권한 상승을 통해 접근 제어를 무력화하는 경우가 많이 발생한다. 가상화나 디스크 분할은 파티션 단위로 격리를 하기 때문에 파일 단위의 제어에 비해 효율성이 떨어진다.

본 논문에서는 사용자 또는 응용의 권한에 따라 파일을 숨기는 방식을 제안한다. 랜섬웨어나 와이퍼 공격은 공격 대상 파일을 미리 특정하여 공격하는 것이 아니라 시스템에 침입한 후 확장자나 경로를 보고 대상 파일을 찾는다. 따라서 공격 대상이 될 수 있는 파일을 숨기게 되면 공격을 피할 수 있다. 파일을 숨기는 기능은 일종의 기만 기술로 공격자를 속여 공격을 지연 또는 방지하는 기술이다. 본 논문에서 사용하는 방식은 사용자가 응용을 실행할 때 특정 파일뷰를 활성화시키면 그 뷰를 통해 접근할 수 있는 파일들이 보이고, 그렇지 않으면 그 파일들은 보이지 않는 방식이다. 따라서 멀웨어가 침입에 성공하더라도 파일뷰를 활성화시키지 못한다면 공격 대상 파일을 찾을 수 없게 된다. 이 방법은 파일을 추가로 저장하지 않기 때문에 디스크의 공간을 추가로 소비하지 않고 성능 저하도 작다는 장점이 있다.

2. 관련연구

제안하는 기술과 관련된 기존 연구는 Table 1에 요약되어 있다. 가상화는 디스크를 가상 머신 별로 할당하여 한 가상 머신에서는 할당 받은 디스크 공간만 접근하도록 하여 다른 머신의 디스크 공간의 접근을 차단한다 (Zhang, C., Wu, Y., Yu, Z., and Li, Z., 2017; Erulanova, A., Yessenbekova, G., Zhanysbayeva, K., Tlebaldinova, A., Zhantassova, Z., and Zhomartkyzy, G., 2020). 디스크 분할도 유사하게 디스크의 분할하여 특정 파티션은 권한이 높은 사용자 또는 응용만 접근하도록 한다. 두가지 방식 모두 파일 단위로 파일 접근을 통제하는 것이 아니라 파티션 단위로 통제한다. 만약 어느 하나의 파티션이 침해를 당하면 그 파티션에 있는 모든 파일들이 노출된다. 이러한 피해를 최소화 하기위해서는 파티션을 여러 개 생성하여 하나가 노출되더라도 다른 파티션에는 영향이 없도록 해야 한다. 그러나 기존 기술들은 파티션이 다 활용되지 않더라도 공간을 확보해야 하기 때문에 디스크 공간 효율성이 떨어진다. 본 연구에서 사용할 방법은 파일 단위로 통제하므로 디스크 공간을 훨씬 효율적으로 사용하면서도 파일뷰를 여러 개 만들 수 있어 더 안전하다.

백업은 주로 암호학을 이용한 백업 연구가 이루어지고 있었으며, 백업 스케줄러를 최적화하는 아이디어에 관한 연구도 진행되고 있다 (Rizvi, S. S., and Razaque, A., 2016; Qin, Y., Hoffmann, B., and Lilja, D. J., 2018). 백업에서도 가상화를 사용하여 스냅숏(snapshot)을 로컬에 저장하는 단계 없이 바로 원격에 저장하거나 랜섬웨어에 대한 대응책으로 컨테이너(container)를 추가하여 컨테이너에서만 로컬 및 원격에 저장된 소스 및 백업 데이터에 액세스할 수 있는 방식의 아이디어와 컨테이너의 체크 포인트를 사용하여 백업하는 방법 등이 연구되고 있다. 또한, 스토리지를 추가하여 백업하는 주제도 계속하여 연구가 진행되고 있다.

디스크를 활용하는 연구에서는 멀티 캐시나

Table 1 Existing defense techniques to attacks on files

Category	Description	Sub-category	Papers
Virtualization	Files on a virtual machine cannot be accessed by other virtual machines.	Container	Suresh, S. and Rao, L. M. (2017); Walkowski, M., Biskup, M., Szewczyk, A., Oko, J., and Sujecki, S. (2019)
		Policy	Bacis, E., Mutti, S., Capelli, S. and Paraboschi, S. (2015); Samaniego, M., Espana, C. and Deters, R. (2018)
		Hypervisor	Zhang, C., Wu, Y., Yu, Z., and Li, Z. (2017); Erulanova, A., Yessenbekova, G., Zhanysbayeva, K., Tlebaldinova, A., Zhantassova, Z., and Zhomartkyzy, G. (2020)
Backup	Replica of files are retained so that they can be recovered if they were lost by file-targeted attacks.	Algorithm	Rizvi, S. S., and Razaque, A. (2016); Qin, Y., Hoffmann, B., and Lilja, D. J. (2018)
		Virtualization	Chen, X., Jiang, J. and Jiang, Q. (2015); Wang, Z., Zeng, J., Lv, T., Shi, B., and Li, B. (2016); Jin, Y., Tomoishi, M., Matsuura, S., and Kitaguchi, Y. (2018)
		Physical backup	Leterme, W., Pirooz Azad, S., and Van Hertem, D. (2016); Zhao, Y. and Lu, N. (2018); Jeon, W. and Oh, I. (2007)
Disk	The storage device (disk) has an internal mechanism to handle file-targeted attacks.	Flash memory	Yao, J. and Jiang, X. (2020); Md Mehedi Hasan and Biswajit Ray (2020)
		Hardware	Tsai, C. et al. (2020); Angel, S., Kannan, S., and Ratliff, Z. (2020); Zhu, J. et al. (2020)
		Software	Harrison, L., Vijayakumar, H., Padhye, R., Sen K., and Grace, M. (2020); Xiong, A., Wang, T., Li, N., and Jha, S. (2020)
Access control	File access is controlled by additional information or cryptographic techniques.	Permission	Lozhnikov, P. S., Sulavko, A. E., Eremenko, A. V., and Volkov, D. A. (2016); Uddin, M., Islam, S. and Al-Nemrat, A. (2019); Röhling, M. M., Grimmer, M., Kreubel, D., Hoffmann, J., and Franczyk, B. (2019)
		Additional information	Amin, R., Sherratt, R. S., Giri, D., Islam, S. H. and Khan, M. K. (2017); Benedict, A. G. (2019)
		Key protocol	Sanctorum, A. and Signer, B. (2019)

분산 메모리를 통해 파일을 보호하거나 시스템을 보호하였고, 하드웨어적 특징을 통한 논문들도 몇몇 있었는데 하드웨어 인클레이브(enclave)를 이용하거나 다른 클라이언트가 자원을 할당받았는지 아닌지를 클라이언트에게 공개하지 않고 개인 자원 할당 자를 암호화하는 연구와 신뢰 환경 구축에 관한 연구들이 진행되고 있다 (Tsai, C. et al., 2020; Angel, S., Kannan, S.,

and Ratliff, Z., 2020; Zhu, J. et al., 2020). 또한, 에뮬레이션을 통해 TrustZone에서 퍼징(fuzzing) 등 동적 분석이 가능하게 하는 등의 연구가 진행되고 있다.

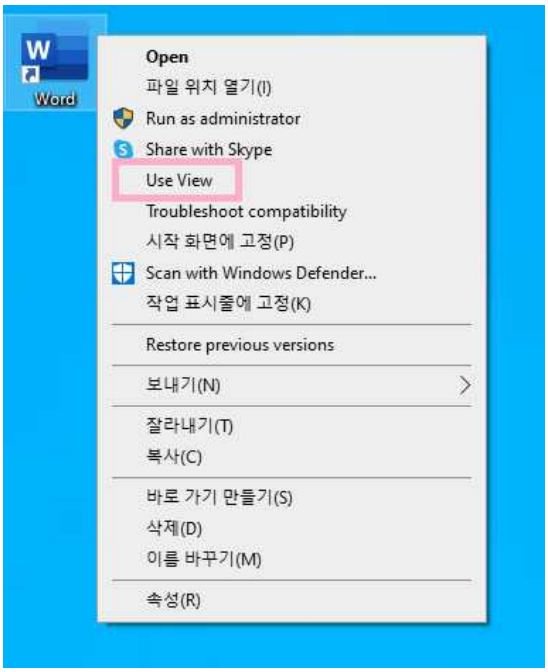


Fig. 1 Activating a view by right-clicking the application.

마지막으로 접근을 통제하는 주제의 논문들은 주로 사용자의 역할에 따른 접근을 통제하는 방법에 대한 논문들이 주를 이루었고 파일에 생체적 특징을 이용하여 워터마크(watermark)를 생성하거나 스테가노그래피를 추가하여 파일을 보호하는 등의 논문들도 있다 (Amin, R., Sherratt, R. S., Giri, D., Islam, S. H. and Khan, M. K., 2017; Benedict, A. G., 2019). 또한, 접근 통제 주제에서도 암호학을 바탕으로 접근을 통제하는 방법은 꾸준하게 연구되고 있다.

3. 제안하는 방법

3.1 기본기능

시스템에 정해진 개수의 파일뷰가 정의되어 있다. 사용자는 응용프로그램을 실행시킬 때 파일뷰를 활성화할지, 한다면 어느 뷰를 활성화할지 결정한다. 파일뷰를 활성화하려면 응용 프로그램의 실행 파일을 마우스 우클릭하여 “파일뷰 활성화”메뉴를 선택하고 파일뷰와 패스워드를 입력한다(Fig. 1). 파일뷰와 패스워드가 맞는다면 해당 파일뷰가 활성화된 상태로 응용이 수행된다. 파일뷰와 패스워드가 맞지 않는다면 파일뷰는 활성화되지 않은 채로 응용이 수행된다.

응용이 파일의 리스트를 읽어 올 때 파일뷰가 활성화되어 있으면 해당 뷰를 통해 볼 수 있는 파일이 포함되어 리스트를 읽어 온다. 그렇지 않으면 일반적인 파일만 리스트에 포함되어 일반적인 파일만 볼 수 있다. 파일이 보이지 않아도 그 파일의 절대 경로를 알면 파일에 접근을 시도할 수 있다. 이런한 경우를 방지하기 위해 파일 접근 시에도 파일뷰 활성화 여부를 체크하여 해당 뷰를 통해서 접근할 수 있는 경우에만 접근을 허용한다.

Fig. 2는 두 개의 뷰를 사용하는 예를 보여준다. 같은 바탕화면의 파일 리스트를 읽더라도 뷰 1이 활성화되어 있는 경우에는 view1.docx, 뷰 2가 활성화되어 있는 경우에는 view2.docx, view2_2.docx가 보이지만 아무 뷰도 활성화되어 있지 않은 경우에는 어떤 세 파일 모두 보이지 않는다.

파일뷰가 활성화되어 있는 상태에서 응용에서 파일을 생성하면 그 파일은 활성화되어 있는 뷰를 통해서만 접근할 수 있는 파일로 생성된다. 공개 파일을 생성하려면 응용을 닫고 뷰를 활성화하지 않은 상태로 응용을 다시 시작해야 한다.

3.2 구현

프로토타입의 개발은 윈도우10 (32bit) 환경에서 진행되었으며, 뷰를 인증하는 윈도우 애플리케이션 개발과 뷰를 실행하는 윈도우 device driver의 개발로 진행되었다. 현 구현은 3개의 파일의 입출력을 통해 뷰의 사용 여부와 뷰를 통해 생성된 파일을 선별하도록 구성하였다. 3

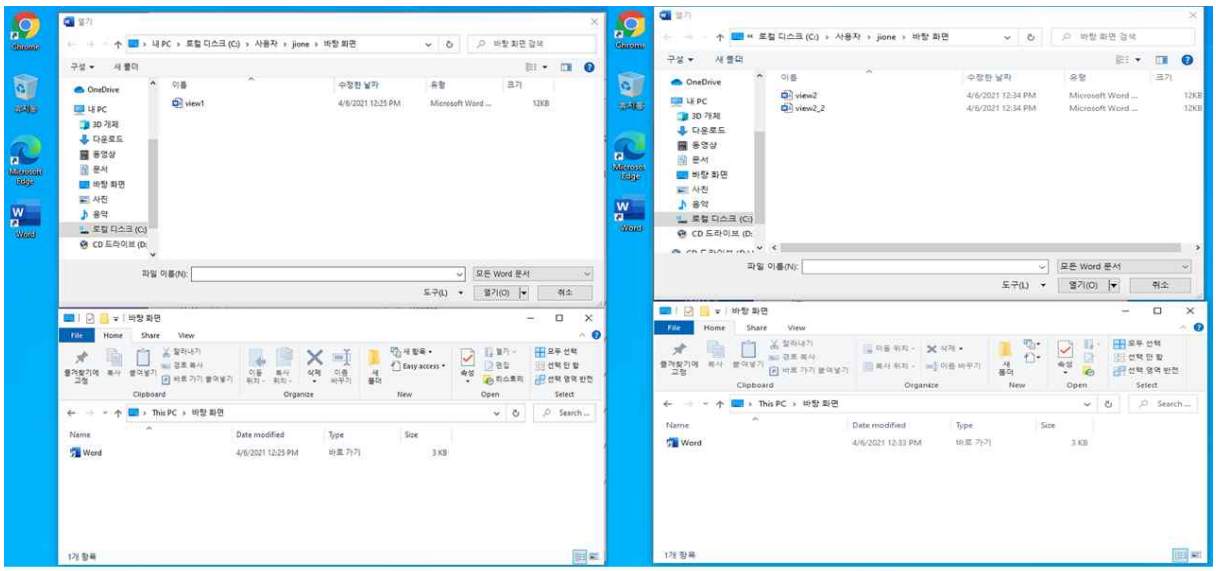


Fig. 2 An example of showing difference files depending on which view is activated.

개의 파일은 다음과 같다. 아래 3개의 파일은 모두 어떤 뷰에서도 보이지 않는 파일이다. 이를 구별하기 위해 “hide_”로 시작되는 이름을 사용하였다. 현재 구현에서는 “hide_”로 시작되는 이름을 가진 파일은 어떠한 뷰에서도 보이지 않는다. “hide_”라는 패턴은 우연한 중복 방지를 위해 더 복잡한 임의의 문자열을 사용할 수 있다.

- hide_ID.txt: 뷰의 ID와 password가 기록된 파일. 이 파일은 애플리케이션에서만 사용하며, 해당 파일과 입력된 ID, password의 일치 여부를 확인하는 데 사용됨.
- hide_pid.txt: 뷰를 사용하고 있는 process의 ID와 해당 뷰 ID가 기록된 파일. 이 파일의 작성은 애플리케이션에서 이루어지며, 작성된 파일을 device driver에서 읽어 요청한 프로세스가 어떠한 뷰를 사용하고 있는지 확인하는 데 사용됨.
- hide_fid.txt: 뷰를 사용하여 만들어진 파일의 ID와 해당 뷰 ID가 기록된 파일. 이 파일은 작성과 접근이 모두 device driver에서 이루어짐.

우선 애플리케이션은 윈도우 시스템에 register를 추가하여 워드나 크롬과 같은 일반 프로그램의 우클릭을 통해 실행한다. 이 애플리케이션은 “hide_ID.txt” 파일을 읽어 애플리케이션에 입력된 view의 ID와 password가 일치하는지를 판별하는 로직과 일치한다면 뷰를 사용하겠다고 요청한 일반 프로그램 프로세스를 생성하여 뷰의 ID와 생성된 프로세스의 PID를 “hide_pid.txt” 파일에 기록하는 로직으로 구현된다. PID에 따라 뷰를 활성화 하기 때문에 PID가 다른 자식 프로세스에 자동으로 뷰가 상속되지 않는다. 현재 프로토타입은 hide_fid.txt가 단순한 리스트로 구현되어 있어 파일의 수가 늘어나면 이에 비례하여 탐색 시간이 늘어난다. 실제 구현에서는 트리와 같은 자료 구조를 사용하여 많은 수의 파일 탐색도 빠르게 처리할 수 있다.

다음으로 device driver는 hooking의 개념을 통해 개발을 진행하였다. Device driver가 처음 시작될 때 기존의 윈도우 기존 system call의 주소들을 계산한 후 system call을 현 프로젝트에서 새롭게 구현한 system call 주소로 변경하여 변경된 system call을 사용할 수 있도록 조

작하였다. 현 구현에서 hooking 한 system call 은 다음과 같다.

- ZwCreateFile: 기존의 파일을 읽거나 새로 생성하는 system call.
- ZwQueryDirectoryFile: 디렉토리의 파일 리스트를 읽는 system call.

ZwCreateFile은 파일을 생성하는 system call로, NewZwCreateFile이라는 이름의 새로운 system call로 hooking을 하였다. NewZwCreateFile 은 파일 생성 요청을 한 프로세스가 “hide_pid.txt” 파일에 작성된 PID라면, 즉 view를 사용하고 있는 프로세스라면 새로 생성된 파일에 대한 FID 와 해당 뷰에 대한 정보를 “hide_fid.txt”파일에 작성한다. 만약 파일 생성 요청을 한 프로세스가 뷰를 사용하고 있지 않다면, 기존의 ZwCreateFile과 동일하게 파일을 생성하고 파일 생성에 대한 결과를 반환한다.

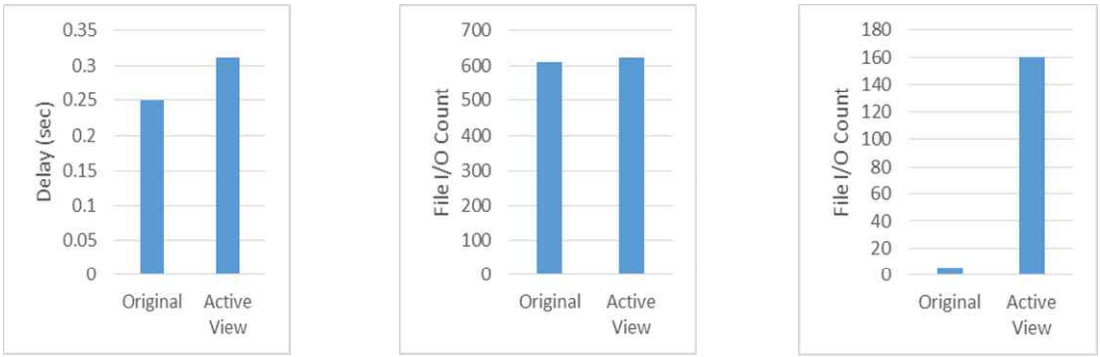
ZwQueryDirectoryFile 은 디렉토리 내에서 파일의 목록을 보여주는 system call로, NewZwQueryDirectoryFile이라는 이름의 새로운 system call로 hooking을 하였다. NewZwQueryDirectoryFile는 3 가지 상황에서 파일의 존재를 숨긴다. 첫번째는 “hide_”라는 패턴의 이름으로 저장된 파일이나 디렉토리는 뷰의 사용 여부와 관계없이 파일 디렉토리 리스트에서 표시되지 않는다. 다음으로 파일의 리스트를 요청한 프로세스가 뷰를 사용하고 있는 프로세스라면 “hide_fid.txt”에서 해당 뷰외에 다른 뷰를 통해 생성된 파일의 경우 디렉토리 리스트에서 표시되지 않는다. 마지막으로 뷰를 사용하지 않는 일반적인 프로세스에서의 파일 리스트 요청의 경우 “hide_fid.txt”에 작성된 FID를 갖는 파일이 모두 표시되지 않는다.

Table 2 Experimental setup

Item	Value
Operating system	Windows 10
Processor	Intel i7
Memory	2 GB
Disk	100 GB
Hypervisor	VirtualBox 5.2.26

4. 성능평가

제안하는 기술의 성능 평가를 위해 윈도우 10



(a) UI delay (b) File creation (c) Read directory

Fig. 3 Performance comparison between the original Windows where the view is not implemented (Original) and the augmented Windows where the view is activated (Active View).

에서 프로토타입을 구현하여 Table 2의 사양을 가진 가상 머신 환경에서 성능을 측정하였다.

성능 실험은 기존의 윈도우와 뷰를 활성화했을 때에 대하여 윈도우에서 제공하는 Windows Performance Recorder를 사용하여 측정하였다. 성능 실험은 3가지로 진행하였으며 측정값 중 최소값을 통해 성능을 비교하였다. 3가지 실험은 다음과 같다. 처음 UI가 활성화될 때까지의 지연 시간, 파일을 생성할 때 File I/O 횟수, 그리고 디렉터리 목록을 요청하였을 때 File I/O 횟수이다. 뷰를 사용하여 파일을 생성하거나 디렉토리를 읽게 되면 파일 접근이 늘어나 시간 지연이 발생한다. 이에 대한 시간 지연의 시작과 끝을 특정하기 어려워 파일 접근 횟수를 측정하였다. 측정 결과는 Fig. 3에 있다.

윈도우의 경우 사용자 편의성을 기반으로 하므로 뷰를 활성화했을 시 사용자가 사용에 불편함이 있는지를 측정하기 위하여 응용을 실행하였을 때 응용이 실행 창이 나타날 때까지의 시간을 측정하는 UI Delay 값을 사용하였다. UI Delay는 초 단위로 프로그램 실행 요청 시 화면이 나타나기까지의 지연 시간을 측정한 값이다. 본 실험에서는 MS Word실행 시 UI Delay에 대하여 평가하였다. 뷰를 활성화하면 Word 실 Virtualization 시 발생하는 temp 파일 생성으로 인한 hide_fid.txt에 대한 파일 입출력으로 UI Delay가 발생할 것으로 예측하였으나, 뷰가 구현되어 있지 않은 기존 윈도우와 같은 수준 정도의 지연 시간이 발생한다. 또한, 이 값이 1초 미만으로 매우 작다.

다음으로 파일 생성 시 File I/O에 대한 횟수를 비교하였다. File I/O의 횟수는 파일 생성 요청 시 생성되는 모든 File I/O에 대한 이벤트의 총 개수로 temp 파일 생성, temp 파일 삭제, 디렉터리 읽어오기와 같은 요청들이 모두 포함된다. 파일 생성 시 발생하는 File I/O는 기존의 값이 611, 뷰 활성화시의 값이 623으로 hide_fid.txt에 write 하는 I/O에서의 차이가 있었지만, 그 차이가 크지 않았다.

마지막으로 디렉토리 리스트를 요청하였을 때의 File I/O 횟수를 비교하였다. File I/O는 각각 기존 6, 뷰 활성화시 160으로 차이가 컸다. 160개 중 155개는 hide_fid.txt파일을 생성하는 것과 파일을 읽는 것이었으나, 이것의 실행시간이 354 μs 정도로 1초 미만으로 매우 작기 때문에 실제 사용자가 느낄 수 있는 수준은 아니었다.

제안하는 방법과 기존 방법들이 Table 3에 비교되어 있다. 가상화는 파티션 단위로 접근을 제어하나 제안하는 방법을 포함하여 다른 기법들은 파일 단위로 제어한다. 백업이나 디스크 내부 방어 기법에서는 파일을 복제하여 보관하지만 다른 기법들은 같은 파일을 중복 저장하지 않는다. 가상화와 제안하는 방법은 보호 대상 파일이 공격자로부터 보이지 않게 해 주지만 다른 기술들은 공격 대상 파일이 어떤 파일인지 식별할 수 있다. 성능 오버헤드는 가상화와 접근 제어가 부가적으로 확인해야하는 정보가 많아 상대적으로 크다.

5. 결 론

파일 대상 공격에 대한 방어 기법으로 기만 기술의 일종인 파일 은닉 기술을 제안하였다. 뷰를 활성화 시키지 않으면 그 뷰를 통해서만 볼 수 있는 파일을 찾을 수 없기 때문에 사용자

Table 3 Comparison with other defense mechanisms.

	Granularity	Redundancy	Exposure	Overhead
Virtualization	Partition	Single	Hidden	High
Backup	File	Redundant	Exposed	Low
Disk	File	Redundant	Exposed	Low
Access control	File	Single	Exposed	High
Proposed	File	Single	Hidden	Medium

의 주요 파일들을 멀웨어로부터 숨길 수 있다. 제안하는 기술을 윈도우 10에 구현하여 성능을 평가해 보았을 때 기존 윈도우에 비해 큰 성능 저하는 발생하지 않았다. 기존 방어 기술들과 비교하면 파일 단위로 관리가 가능하고, 파일을 중복 저장할 필요가 없어 저장 공간의 낭비가 없고 공격자가 공격 대상이 되는 파일을 식별조차 할 수 없게 한다는 장점이 있다.

파일 은닉을 적절히 사용하면 파일 대상 공격을 방어할 수 있다. 본 논문에서는 어떻게 활용하여야 파일 은닉의 효과를 극대화할 수 있는지, 어떤 공격 유형은 방어할 수 있고 어떤 것은 안 되는지, 어떤 특성들을 보장할 수 있는지 등에 대한 연구는 포함되어 있지 않다. 파일 은닉 기술의 활용 방안에 대한 추가 연구가 지속된다면 파일 대상 공격의 피해를 줄이는데 크게 기여할 수 있을 것이다.

References

Zhang, C., Wu, Y., Yu, Z. and Li, Z. (2017). Research and implementation of file security mechanisms based on file system filter driver. 2017 *Annual Reliability and Maintainability Symposium* (RAMS), Orlando, FL, pp. 1-6, doi: 10.1109/RAM.2017.7889772.

Zhu, J. *et al.* (2020). Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment. *IEEE Symposium on Security and Privacy* (SP), San Francisco, CA, USA, pp. 1450-1465, doi: 10.1109/SP40000.2020.00054.

Amin, R., Sherratt, R. S., Giri, D., Islam, S. H., and Khan, M. K. (2017). A software agent enabled biometric security algorithm for secure file access in consumer storage devices. *IEEE Transactions on Consumer Electronics*, vol. 63, no. 1, pp. 53-61, doi: 10.1109/TCE.2017.014735.

Angel, S., Kannan, S., and Ratliff, Z. (2020). Private resource allocators and their applications. *IEEE Symposium on Security and Privacy* (SP), San Francisco, CA, USA, pp. 372-391, doi: 10.1109/SP40000.2020.00065.

Bacis, E., Mutti, S., Capelli, S., and Paraboschi, S. (2015). DockerPolicyModules: Mandatory Access Control for Docker containers. *IEEE Conference on Communications and Network Security* (CNS), Florence, pp. 749-750, doi: 10.1109/CNS.2015.7346917.

Benedict, A. G. (2019). Improved File Security System Using Multiple Image Steganography. *International Conference on Data Science and Communication* (IconDSC), Bangalore, India, pp. 1-5, doi: 10.1109/IconDSC.2019.8816946.

Chen, X., Jiang, J., and Jiang, Q. (2015). A Method of Self-Adaptive Pre-Copy Container Checkpoint. *IEEE 21st Pacific Rim International Symposium on Dependable Computing* (PRDC), Zhangjiajie, pp. 290-300, doi: 10.1109/PRDC.2015.11.

Erulanova, A., Yessenbekova, G., Zhanysbayeva, K., Tlebaldinova, A., Zhantassova, Z. and Zhomartkyzy, G. (2020). Hardware and Software Support of Technological Processes Virtualization. *International Conference on Electrical and Electronics Engineering* (ICEEE), Antalya, Turkey, pp. 333-337, doi: 10.1109/ICEEE49618.2020.9102506.

Harrison, L., Hayawardh, V., Padhye, R., Sen, K., and Grace, M. (2020). PARTEMU: Enabling Dynamic Analysis of Real- World TrustZone Software Using Emulation. *USENIX Security Symposium* (USENIX Security 20), pp. 789－806.

Jeon, W. and Oh, I. (2007). A Study for Detection of the Kernel Backdoor Attack and Design of the restoration system. Journal of the Korea Society Industrial

Information System Vol. 12 No. 3 pp.104-115

Jin, Y., Tomoishi, M., Matsuura, S., and Kitaguchi, Y. (2018). A Secure Container-based Backup Mechanism to Survive Destructive Ransomware Attacks. *International Conference on Computing, Networking and Communications* (ICNC), Maui, HI, pp. 1-6, doi: 10.1109/ICCNC.2018.8390376.

Leterme, W., Pirooz Azad, S., and Van Hertem, D. (2016). A Local Backup Protection Algorithm for HVDC Grids. *IEEE Transactions on Power Delivery*, vol. 31, no. 4, pp. 1767-1775, doi: 10.1109/TPWRD.2016.2543306.

Lozhnikov, P. S., Sulavko, A. E., Eremenko, A. V., and Volkov, D. A. (2016). Method of protecting paper and electronic text documents through a hidden biometric identifier based on a signature. *Dynamics of Systems, Mechanisms and Machines* (Dynamics), Omsk, pp. 1- 5, doi: 10.1109/Dynamics.2016.7819037.

Md Mehedi Hasan and Biswajit Ray (2020). Data Recovery from Scrubbed NAND Flash Storage: Need for Analog Sanitization. *USENIX Security Symposium* (USENIX Security 20), pp. 1399-1408.

Qin, Y., Hoffmann, B. and Lilja, D. J. (2018). HyperProtect: Enhancing the Performance of a Dynamic Backup System Using Intelligent Scheduling. *IEEE 37th International Performance Computing and Communications Conference* (IPCCC), Orlando, FL, USA, pp. 1-8, doi: 10.1109/PCCC.2018.8711182.

Rizvi, S. S. and Razaque, A. (2016). Black-box: A new secure distributed peer-to-peer file storage and backup system. *International Multi-Topic Conference* (INMIC), Islamabad, 2016, pp. 1-6, doi: 10.1109/INMIC.2016.7840163.

Röhling, M. M., Grimmer, M., Kreubel, D., Hoffmann, J., and Franczyk, B. (2019). Standardized container virtualization approach for collecting host intrusion detection data. *Federated Conference on Computer Science and Information Systems* (FedCSIS), Leipzig, Germany, pp. 459-463, doi:10.15439/2019F212.

Samaniego, M. , Espana, C., and Deters, R. (2018). Smart Virtualization for IoT. *IEEE International Conference on Smart Cloud* (SmartCloud), New York, NY, pp. 125-128, doi: 10.1109/SmartCloud.2018.00028.

Sanctorum, A., and Signer, B. (2019). A Unifying Reference Framework and Model for Adaptive Distributed Hybrid User Interfaces. *International Conference on Research Challenges in Information Science* (RCIS), Brussels, Belgium, pp. 1-6, doi: 10.1109/RCIS.2019.8877048.

Suresh, S. and Rao, L. M. (2017). A container based collaborative research cloud framework with a hybrid access control approach for enhanced isolation. *International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques*, Mysuru, pp. 1-6, doi: 10.1109/ICEECCOT.2017.8284605.

Tsai, C. *et al*, (2020). Civet: An Efficient Java Partitioning Framework for Hardware Enclaves. *USENIX Security Symposium* (USENIX Security 20), pp. 505-522.

Uddin, M., Islam, S., and Al-Nemrat, A. (2019). A Dynamic Access Control Model Using Authorising Workflow and Task-Role-Based Access Control. *IEEE Access,* vol. 7, pp. 166676-166689, doi: 10.1109/ACCESS.2019.2947377.

Walkowski, M., Biskup, M., Szewczyk, A. Oko, J., and Sujecki, S. (2019). Container Based Analysis Tool for Vulnerability

Prioritization in Cyber Security Systems. *International Conference on Transparent Optical Networks* (ICTON), Angers, France, pp. 1-4, doi: 10.1109/ICTON.2019.8840441.

Wang, Z., Zeng, J., Lv, T., Shi, B. and Li, B. (2016). A Remote Backup Approach for Virtual Machine Images. *IEEE 3rd International Conference on Cyber Security and Cloud Computing* (CSCloud), Beijing, pp. 252-255, doi: 10.1109/CSCloud.2016.41.

Xiong, A., Wang, T, Li, N., and Jha, S. (2020). Towards Effective Differential Privacy Communication for Users' Data Sharing Decision and Comprehension. *IEEE Symposium on Security and Privacy* (SP), San Francisco, CA, USA, pp. 392-410, doi: 10.1109/SP40000.2020.00088.

Yao, J., and Jiang, X. (2020). Research on multi cloud dynamic secure storage technology. *International Conference on Computer Engineering and Application* (ICCEA), Guangzhou, China, pp. 72-78, doi: 10.1109/ICCEA50009.2020.00022.

Zhao, Y., and Lu, N. (2018). Research and Implementation of Data Storage Backup. *IEEE International Conference on Energy Internet* (ICEI), Beijing, pp. 181-184, doi: 10.1109/ICEI.2018.00040.

최 지 원 (Jione Choi)
- 동덕여자대학교 컴퓨터학과 공학사
- (현재) 고려대학교 정보보호대학원 정보보호학과 석사 박사 통합과정
- 관심분야: 파일시스템 보안, 하드웨어 보안

이 중 희 (Junghee Lee)
- 정회원
- 서울대학교 컴퓨터공학과 공학사
- 서울대학교 컴퓨터공학과 공학석사
- 조지아공과대학교 전자공학과 공학박사
- (현재) 고려대학교 정보보호대학원 부교수
- 관심분야: 하드웨어 보안

이 규 호 (Gyuho Lee)
- 인하대학교 컴퓨터공학과 공학사
- 인하대학교 정보통신공학과 공학석사
- (현재) LIG넥스원 C4I연구소. 사이버전연구팀 수석연구원
- 관심분야: 무기체계 사이버보안, 하드웨어 보안, 안티탬퍼링

유 재 관 (Jaegwan Yu)
- 2015년 2월 고려대학교 전자정보공학과 학사
- 2017년 2월 성균관대학교 플랫폼소프트웨어공학과 석사
- (현재) LIG넥스원 C4I연구소, 사이버전연구팀 선임연구원
- 관심분야: 시스템보안

박 아 란 (Aran Park)
- 고려대학교 정보보호학과 공학석사
- (현재) LIG넥스원 C4I연구소. 사이버전연구팀 선임연구원
- 관심분야: 디지털포렌식, 사이버보안