Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.5, Oct. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.5.961

가상머신 내 mutex 공유 자원을 이용한 은닉 채널 구현*

고 기 완,† 최 형 기‡
성균관대학교

Implementation of Covert Channel Using Mutex Shared Resources in Virtual Machine*

Ki-Wan Ko,† Hyoung-Kee Choi‡
Sungkyunkwan University

요 약

공유 자원 간섭으로 인해 가상머신 간의 격리가 위반되고 은닉 채널 구현이 가능해서 클라우드 컴퓨팅 환경에서 가상머신 간의 격리는 중요 보안 요소이다. 본 논문에서는 Hyper-V 하이퍼바이저의 구조를 분석하여 가상머신 간의 은닉 채널을 구현한다. Hyper-V는 가상머신 간의 상호 배제를 위해 mutex(mutual exclusion) 기법을 사용한다. Mutex로 인해 가상머신 간의 간섭이 발생하고 은닉 채널 구현이 가능함을 밝힌다. 구조가 복잡한 Hyper-V에 적용 가능한 mutex 자원 탐색 방안을 고안하여 은닉 채널을 다수 구현하였다. Mutex 기반 은닉 채널은 하드웨어 의존적이지 않으며, 은닉 채널이 탐지되거나 방어되는 경우 다수의 은닉 채널 중에서 다른 은닉 채널을 이용하면 방어 기법 회피가 가능하다.

ABSTRACT

Isolation between virtual machines in a cloud computing environment is an important security factor. The violation of isolation between virtual machines leads to interferences of shared resources and the implementation of covert channels. In this paper, the structure of Hyper-V hypervisor is analyzed to implement covert channels between virtual machines. Hyper-V uses a mutex technique for mutual exclusion between virtual machines. It indicates that isolation of virtual machines is violated and covert channels can be implemented due to mutex. We implemented several covert channels by designing a method for searching mutex resources applicable to Hyper-V with complex architectures. The mutex-based covert channel is not hardware dependent. If the covert channel is detected or defended, the defensive technique can be avoided by using the other covert channel among several covert channels.

Keywords: Cloud Computing, Virtualization, Covert Channel

I. 서 론

멀티테넌시(multi-tenancy)란 한정된 자원을 여러 사용자가 공유하여 사용하는 기술이다. 클라우드 컴퓨팅(cloud computing) 서비스 모델의 한 종류인 IaaS(Infrastructure as a Service)는 시스템 자원에 멀티테넌시를 적용한다. IaaS는 물리

Received(08. 02. 2019), Modified(08. 17. 2019), Accepted(08. 19. 2019)
* 이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2016R1D1A1B03936211).
† 주저자, gogil@skku.edu
‡ 교신저자, meosery@skku.edu(Corresponding author)

적으로 한정된 시스템 자원을 여러 사용자가 공유하여 사용한다. IaaS 서비스 사용자는 논리적으로 격리된 가상의 시스템인 가상머신을 제공받는다. IaaS 클라우드 컴퓨팅은 멀티테넌시를 지원함과 동시에 사용자 간의 완벽히 격리된 가상머신을 제공해야 한다.

IaaS 서비스의 가상머신들은 물리적으로 한정되어 있는 시스템 자원에 동시 접근이 불가능하여 순차적으로 접근해야 한다. 한 가상머신이 자원을 사용하면 다른 가상머신의 자원 접근에 대한 지연이 발생한다. 가상머신들이 자원을 공유하여 사용하는 과정에서 가상머신 간의 간섭이 발생한다. 가상머신간 간섭은 클라우드 컴퓨팅 환경에서 공격으로 악용된다. 간섭을 이용하여 다른 가상머신의 상태를 예측하거나, 가상머신 간에 은밀하게 정보를 공유하는 은닉 채널 생성이 가능하다. 은닉 채널은 클라우드 컴퓨팅 관리자의 감시를 피해 가상머신 간의 통신에 사용된다.

은닉 채널은 송신자와 수신자만이 알고 있는 채널이다. 다른 사용자가 접근하거나 채널의 존재를 알 수 없어 송신자와 수신자 간의 은밀한 통신이 가능하다. 클라우드 컴퓨팅 환경에서의 은닉 채널은 가상머신 간의 은밀한 통신 채널이다. 클라우드 컴퓨팅 관리자는 가상머신의 네트워크 트래픽, 리소스 사용률 등의 정보를 모니터링하여 악의적인 행위를 탐지하므로 일반 통신 채널을 이용한 비정규 행위는 대부분 감지된다. 은닉 채널을 사용하면 클라우드 컴퓨팅 관리자와 보안 솔루션의 감시를 피해 민감한 정보를 공격자에게 전송할 수 있다.

은닉 채널은 구현 방식에 따라 스토리지 채널(storage channel)과 타이밍 채널(timing channel)로 나누어진다. 스토리지 채널은 송신자와 수신자가 데이터를 공유하는 메모리 영역을 통신 매개체로 이용한다. 송신자는 공유 메모리에 전달하려는 데이터를 작성한다. 수신자는 공유 메모리에서 데이터를 읽어서 송신자와 통신한다. 타이밍 채널은 공유 자원의 사용량의 정도로 통신한다. 송신자는 공유 자원 사용 정도를 임계치 이상으로 높여 비트 '1'을 전송하고, 이하로 낮춰 '0'을 전송한다. 수신자는 공유 자원의 사용량의 정도를 측정하여 송신자가 전송한 데이터를 읽는다. 스토리지 채널은 메모리 데이터가 기록되어 탐지가 쉬운 반면, 타이밍 채널은 데이터가 간접적으로 전달되어 탐지가 상대적으로 어렵다. 따라서, 타이밍 채널 방식에 관한 연구가 더 활발하다. 공유 메모리 또는 공유 자원이 공격자에게 남용되지 않도록 제조사는 가상화 기술을 설계하도록 요구되지만, 가상머신들은 반드시 시스템 자원을 공유하여 사용해야하기 때문에 공유 자원 간섭이 발생하지 않도록 격리하는 것은 쉽지 않다.

클라우드 컴퓨팅 환경에서 은닉 채널을 구현한 기존 연구가 존재한다. 기존 은닉 채널은 CPU(Central Processing Unit) 자원을 이용한 연구 [1], [2], [3], [4], [5], 데이터 저장소 하드웨어 자원을 이용한 연구 [6], [7], 가상화 소프트웨어의 자원을 이용한 연구 [8], [9], [10] 가 존재한다. 하드웨어 자원을 이용하는 경우 하드웨어 의존적이며, 하드웨어 모델과 환경에 따라 은닉 채널이 동작하지 않는다. 은닉 채널을 탐지 및 방어하는 기존 연구는 하드웨어 자원을 대상으로 한다 [11], [12]. 하드웨어 자원이 아닌 Xen[13] 가상화 소프트웨어의 자원을 이용한 기존 은닉 채널 연구도 존재한다. Xen의 자원을 이용하는 경우 다른 가상화 소프트웨어에서 동작하지 않는 한계점이 존재한다.

본 논문에서는 Hyper-V[14] 가상화 소프트웨어를 대상으로 하는 mutex(mutual exclusion) 기반 은닉 채널을 제안한다. Hyper-V는 마이크로소프트의 Windows 운영체제를 위한 가상화 기술이다. 본 논문에서 제안하는 mutex 기반 은닉 채널은 Hyper-V의 자원을 이용하는 최초의 은닉 채널이다. Hyper-V는 가상머신들의 공유 자원 접근을 순차적으로 처리하기 위해 mutex 기법을 이용한다. Mutex 기법에 사용되는 mutex 자원을 이용한 은닉 채널 구현이 가능하다. Mutex 자원 탐색 방안을 고안하여 탐색한 결과 은닉 채널로 사용할 수 있는 mutex 자원을 다수 발견하였다. 구현한 은닉 채널의 성능을 평가한 결과 실험 환경에서 250 bps(bit per second) 대역폭과 2% 오류율을 보인다. 높은 대역폭을 가지는 기존 은닉 채널보다 비교적 낮은 대역폭을 가지지만 공격자가 은닉 채널을 악용하여 민감한 데이터를 추출하기에 충분하다. 기존 은닉 채널과 비교하여 하드웨어에 의존적이지 않으며 은닉 채널을 방어하기 어려운 특징을 가진다. 하드웨어 제조사와 모델에 영향을 받지 않고 Hyper-V를 사용 중인 모든 시스템 환경에서 은닉 채널이 동작한다. 은닉 채널에 대응하는 방법으로 공유 자원 간섭을 제거하거나 탐지하는 기법이 존재한다. 제안하는 은닉 채널은 탐색된 mutex 자원들을 이용하여 다수의 은닉 채널 구현이 가능하다. Mutex 기반 은닉 채널에 대응하기 위해서는 모든 mutex 자원을 추적해야 하기 때문에 대응하기 어렵다. 공격자가 은닉 채널로 사용

중인 mutex 자원을 사용할 수 없는 경우, 여러 mutex 자원 중 다른 하나를 선택하여 은닉 채널을 구현하면 우회가 가능하다.

II. 배 경

2.1 Hyper-V

Hyper-V는 Microsoft에서 개발한 타입1 하이퍼바이저 기반 가상화 기술이다. Hyper-V를 사용하면 Windows 운영체제를 사용하고 있는 하나의 시스템에 가상머신을 생성하여 여러 게스트 OS를 구동할 수 있다. 타입1 하이퍼바이저 방식에서는 게스트 OS가 하드웨어에 대한 접근을 하이퍼바이저에게 요청하도록 커널을 수정해야 한다. Microsoft에서 제공하는 integration service[15]를 게스트 OS에 설치하면 Windows뿐만 아니라 Linux 및 FreeBSD의 커널이 수정된다.

Hyper-V의 호스트 OS는 하드웨어 자원에 접근하는 것이 가능하지만 게스트 OS는 하드웨어 자원에 직접 접근하는 것이 불가능하다. Hyper-V는 VSP(Virtualization Service Provider)와 VSC(Virtualization Service Client)를 통해 게스트 OS에서의 하드웨어 사용을 지원한다. 게스트 OS에서 하드웨어 자원을 사용하고자 하는 경우, 게스트 OS에 존재하는 VSC가 호스트 OS에 존재하는 VSP에게 하드웨어 자원 사용을 요청한다. VSP는 다수의 게스트 OS에서 발생하는 하드웨어 사용 요청을 적절하게 처리한다. Network, storage, video, HID(Human Interface Devices), PCI(Peripheral Component Interconnect) 하드웨어 장치에 해당하는 VSP와 VSC 쌍이 각각 존재한다. 각각의 게스트 OS는 자신만의 VSC를 가진다. VSP와 VSC의 통신 프로토콜은 공개되지 않았으나, Linux의 integration service가 오픈 소스로 공개되어 있어 프로토콜 분석이 가능하다.

Hyper-V는 VSP와 VSC의 통신을 위해 VMBus(Virtual Machine Bus)라는 메모리 버스를 사용한다. 게스트 OS들은 개별적인 VMBus를 사용하여 다른 게스트 OS와의 독립성을 보장받는다. VMBus는 VSP와 VSC가 메모리를 공유하는 방식으로 구현된다. 공유 메모리는 가상 주소가 아닌, 물리 주소를 사용한다. 가상 주소를 사용하면, 게스트 OS의 가상 주소를 물리 주소로 변환한 후, 물리 주소를 호스트 OS의 가상 주소로 변환하는 과정이 필요하여 성능이 저하된다. 물리 주소로 메모리를 공유해 VSP와 VSC는 빠른 속도로 통신할 수 있다.

III. 관련 연구 소개 및 분류

클라우드 컴퓨팅 환경에서 타이밍 채널 방식에 이용되는 자원은 하드웨어 또는 소프트웨어가 사용된다. 하드웨어 자원으로는 가상머신에서 필수 자원인 CPU(Central Processing Unit)와 데이터 저장소를 사용한다. 하드웨어와 마찬가지로 소프트웨어에 공유 자원이 존재하여 은닉 채널로 활용 가능하다. 하이퍼바이저의 한 종류인 Xen 가상화 소프트웨어의 자원을 이용하여 은닉 채널을 구현한 관련 연구가 존재한다.

3.1 CPU 기반 은닉 채널

CPU 기반 은닉 채널은 높은 대역폭의 통신 속도를 보인다. CPU 캐시는 주 메모리보다 빠르게 데이터 입/출력을 처리한다. CPU 구조가 멀티 코어식으로 복잡해지고 연산 속도가 증가함에 따라, 캐시의 종류와 용량도 증가한다. CPU 코어 내부에 존재하는 L1(Level1), L2(Level2) 캐시와 다른 코어들과 공유해서 사용하는 LL(Last-Level) 캐시가 있다. 한 가상머신이 캐시를 점유하여 사용하는 동안 다른 가상머신은 그 캐시에 접근할 수 없어, 공유 자원 간섭이 발생한다. 공유 자원 간섭을 이용하여 타이밍 채널 방식의 은닉 채널 구현이 가능하다.

Okamura와 Oyama[1]의 은닉채널 CCCV(Covert Channels using CPU loads between Virtual machines)는 공유 자원으로 CPU 부하(Load)를 이용한다. 가상머신에서 CPU를 점유하고 있을 때 다른 가상머신에서 CPU 연산 속도가 저하되는 간섭이 발생한다. CPU 부하를 타이밍 채널의 통신 매개체로 사용하여 0.49 bps(bit per second) 대역폭의 은닉 채널을 구현했다.

Xu의 연구팀 [2]은 L2 캐시를 이용하여 타이밍 채널 방식의 은닉 채널을 구현했다. L2 캐시 영역을 두 영역으로 분리하여, 송신자가 비트 '1'을 전송하고자 하면 A 영역에 접근하고, '0'을 전송하고자 하면 B 영역에 접근한다. 수신자는 A 영역과 B 영역의 접근 속도를 기록한다. B 영역보다 A 영역의 접근 속도가 지연되면 '1'으로 인식하고, B 영역이 A 영

역보다 지연되면 '0'으로 데이터를 인식한다.

Wu의 연구팀 [3]은 기존 CPU 캐시 기반 은닉 채널이 멀티 CPU 시스템에서 동작하지 않는 문제를 해결하였다. CPU가 처리하는 명령어의 최소 단위인 원자적(atomic) 명령어를 이용한다. 원자적 명령어가 처리될 때, CPU와 주 메모리를 연결하는 메모리 버스 장치가 동기화된다. 메모리 버스는 CPU들이 공유하여 사용한다. 가상머신에서 원자적 명령어를 연산하여 메모리 버스를 점유하면, 다른 가상머신에서 원자적 명령어의 연산 속도가 저하되는 간섭을 이용하여 타이밍 채널을 구현했다. 746 bps 대역폭 0.09% 오류율의 은닉 채널을 구현했다.

Liu의 연구팀 [4]은 다수의 코어가 사용하는 LL 캐시를 이용한다. L1, L2 캐시는 코어 내부에 존재하여 은닉 채널 송신자와 수신자 가상머신이 동일한 코어에 적재되어야 통신 가능하다는 한계점이 존재한다. LL 캐시는 모든 코어가 범용적으로 사용하여 기존 캐시 기반 은닉 채널의 한계점을 극복할 수 있으며, 용량이 커 높은 대역폭의 은닉 채널 구현이 가능하다. 600 kbps(kilobits per second) 대역폭 1% 오류율을 보였다.

Sullivan의 연구팀 [5]은 인텔 CPU에서 메모리 복사를 수행할 때, 메모리 주소가 4kb(kilobyte)의 배수이면 4K-aliasing이라 불리는 성능 저하를 이용한다. 성능 저하의 시간 차이를 이용하여 타이밍 채널 방식의 은닉 채널 구현이 가능하다. 최신 인텔 CPU에서만 동작하며 다수의 CPU를 사용하는 환경에서 동작하지 않는 한계점이 존재하지만 1.49mbps(megabits per second) 대역폭과 8.7% 오류율의 강력한 은닉 채널을 구현했다.

캐시 기반 은닉 채널은 하나의 시스템에서 다수의 CPU를 사용하는 경우, 송신자 가상머신과 수신자 가상머신이 동일한 CPU에 적재되지 않으면 은닉 채널이 구동되지 않는다. 또한, CPU는 모든 가상머신이 빈번하게 사용하므로 잡음에 취약하며 CPU 모델과 제조사에 따라 캐시가 존재하지 않거나 비활성화 되어 있는 한계점이 존재한다. CPU 기반 은닉 채널을 방어하기 위해 각 가상머신이 서로 다른 영역의 캐시 메모리를 사용하도록 하거나, 임의로 캐시 메모리를 사용하여 은닉 채널에 잡음을 발생시키는 등 방지책이 존재한다 [11], [12].

3.2 저장소 기반 은닉 채널

하드디스크, 플래시 메모리 등의 데이터 저장소는 I/O 속도에 차이를 타이밍 채널로 이용한다. 가상머신에서 파일에 접근하여 I/O를 점유하는 도중에, 다른 가상머신에서 같은 파일에 접근하려면 I/O 속도가 지연된다. 송신자가 비트 '1'을 전송하려면, 저장소에 있는 특정 파일에 접근을 한다. 수신자는 그 특정 파일에 동시에 접근하여 I/O 속도를 관찰한다. 송신자가 파일을 점유하여 I/O 속도가 사전에 정한 임계치 이상이면, 수신자는 송신 데이터를 '1'로 인식하고, 반대일 경우에는 '0'으로 읽는다.

Ristenpart의 연구팀 [6]은 클라우드 컴퓨팅 환경에서 피해자 가상머신과 공격자 가상머신이 하나의 물리적 시스템에서 동작하는 공동 거주(co-residence) 상태일 때 발생하는 보안 위협을 소개한다. 가상머신들이 같은 시스템에 적재되어 실행되는 공동 거주 여부를 확인하기 위해 하드디스크 기반 은닉 채널을 구현했다. 은닉 채널이 동작할 경우, 송신자 가상머신과 수신자 가상머신이 같은 데이터 저장소를 사용하는 공동 거주 상태임을 파악할 수 있다. 데이터 전송이 목적이 아닌 공동 거주 확인이 목적이므로 채널 속도 향상을 하지 않아 0.0005 bps의 낮은 대역폭으로 동작한다. Yang의 연구팀 [7]은 기존 저장소 기반 은닉 채널을 데이터 통신에 적합하게 최적화하여 125 bps 대역폭의 은닉 채널을 구현했다.

데이터 저장소 기반 은닉 채널은 통신 대역폭이 낮고, 은닉 채널이 동작하는 동안 저장소 I/O를 독점하므로 시스템 전반적인 성능이 저하되는 한계점이 있다. 또한, 송신자와 수신자 가상머신이 공동 거주 상태임에도 은닉 채널이 동작하지 않는 경우가 있다. 가상머신이 물리적으로 독립된 저장소를 제공받거나 클라우드 컴퓨팅 관리자 또는 사용자가 가상머신에 I/O 대역폭 한계치를 설정한 경우, 공유 자원 간섭이 발생하지 않는다.

3.3 Xen 하이퍼바이저 기반 은닉 채널

Xen은 클라우드 컴퓨팅에서 널리 사용되는 타입1 하이퍼바이저 기반 가상화 기술이다. 오픈 소스로 공개되어 있어 분석이 용이한 Xen의 특성을 이용한 은닉 채널이 존재한다.

Salau¨n [8]의 XenCC(Xen Covert

Channel)는 메모리 주소 테이블이 가상머신 간에 공유되는 문제점을 이용한다. 가상머신이 메모리 주소 테이블을 수정할 수 있어 송신자 가상머신은 메모리 주소 대신 수신자에게 보내고자 하는 데이터를 삽입한다. 수신자는 메모리 주소 테이블을 읽어 송신자가 보낸 데이터를 수신하는 스토리지 채널 방식으로 452 kbps 대역폭의 은닉 채널을 구현했다.

Wu의 연구팀 [9]은 Xen의 공유 자원을 탐색하여 SMCTC(Sharing Memory Covert Timing Channel)를 구현했다. 공유 자원을 탐색하기 위해 Xen의 소스 코드를 보다 구조화된 중간 언어인 LLVM(Low-Level Virtual Machine) 코드로 변환한다. 중간 언어를 대상으로 공유 자원을 탐색하는 알고리즘을 제안한다. 탐색한 공유 자원을 이용하여 타이밍 채널 방식의 은닉 채널을 구현하여 175 bps 대역폭 2% 오류율을 보였다.

Shen의 연구팀 [10]의 CCECS(Covert Channel using Event Channel State)는 Xen의 가상머신 이 다른 가상머신에 전달하는 인터럽트인 이벤트를 이용한다. 송신자는 비트 '1'을 보내고자 하면 이벤트를 발생시킨다. 수신자는 이벤트가 발생되는 경우 비트 '1'으로 인식하고, 이벤트가 발생하지 않는 경우 비트 '0'으로 인식하여 데이터 통신이 이루어진다. 13 kbps 대역폭 5% 오류율의 은닉 채널을 구현했다.

하이퍼바이저의 고유 특징을 이용한 은닉 채널은 하이퍼바이저의 변화에 따라 동작하지 않을 가능성이 있다. 기존 연구는 최신 Xen을 대상으로 하지 않으며, Xen의 고유한 특성을 이용하기 때문에 다른 하이퍼바이저에서 동작하지 않는 한계점이 존재한다.

IV. Mutex 기반 은닉 채널

하이퍼바이저 내 소프트웨어 공유 자원이 존재할 경우, 공유 자원에 접근하는 상호 배제에 의해 간섭이 발생한다. 이 간섭을 이용하여 타이밍 채널 방식의 은닉 채널 구현이 가능하다. Hyper-V의 공유 자원을 탐색하여 은닉 채널을 구현하고, 통신 프로토콜을 설계하여 데이터 통신을 검증하고자 한다.

4.1 공격 시나리오

클라우드 컴퓨팅에서 두 개 이상의 가상머신들이 물리적으로 동일한 시스템에 탑재되어 실행되는 것을 공동 거주라 부른다. 공격을 준비 단계에서 공격자와 피해자는 공동 거주되어 있는 상태에서 공격자는 피해자 가상머신에 백도어를 설치한다. 피해자 가상머신에 설치된 백도어는 은닉 채널을 이용하여 민감한 정보를 공격자에게 전송하려 하지만 쉽지 만은 않다. 왜냐하면, 클라우드 컴퓨팅 관리자는 위협 탐지 보안 솔루션과 방화벽을 구성하여 가상머신들의 악성 행위를 탐지하고 정보 유출을 방지하고 피해자 가상머신은 네트워크를 비롯한 외부 연결 통로가 차단된 상태이기 때문이다.

4.2 Mutex 기법

Mutex 기법은 다중 스레드 환경에서 공유 자원에 순차적으로 접근하기 위한 스레드 간 동기화 기법이다. 공유 자원에 대한 mutex-object를 생성한 후 접근 시에 잠금 그리고 종료 시에 잠금을 해제한다. 다른 스레드에서 공유 자원에 접근할 때, mutex-object가 잠금 설정된 경우 잠금 해제될 때까지 대기하여 상호 배제가 이루어진다.

하이퍼바이저 소프트웨어에 가상머신 간의 공유 자원이 존재할 경우 여러 가상머신이 공유 자원에 동시 접근하지 못하도록 가상머신 간의 상호 배제가 필요하다. 하이퍼바이저는 게스트 OS에서 발생하는 하드웨어 접근 요청을 소프트웨어적으로 처리한다. 하드웨어 자원은 물리적으로 한정적이므로 하드웨어 접근 요청을 동시에 수행할 수 없다. 하이퍼바이저는 게스트 OS의 하드웨어 접근 요청 작업을 순차적으로 처리해야 한다. 게스트 OS의 하드웨어 접근 이외에도, 가상머신들의 전역 설정을 수정해야할 경우 상호 배제가 필요하다. 전역 설정을 수정할 때 상호 배제를 하지 않으면 경쟁 상태(race condition)가 발생한다. 전역 설정에 가상머신들이 동시 접근하지 않도록 가상머신 간의 동기화가 필요하다.

4.3 Mutex 기반 은닉 채널의 장점

은닉 채널에 대응하는 방법은 세 가지로 나누어진다. 첫째, 공유 자원을 제거하여 은닉 채널을 구현할 수 없도록 한다. 둘째, 송신자와 수신자가 아닌 다른 사용자가 공유 자원을 지속해서 점유하면, 은닉 채널 통신에 잡음이 발생한다. 은닉 채널에 잡음이 발생하는 경우 통신이 불안정하여 은닉 채널을 사용할 수 없다. 마지막으로, 공유 자원의 사용 빈도를 추적하

여 은닉 채널의 존재 여부를 탐지할 수 있다.

공유 자원 간섭과 같은 보안 결함은 제조사의 패치를 통해 제거된다. 보안 결함을 해결하기 위한 패치가 소프트웨어로 제공되어 하드웨어 설계 단계에서 발생하는 결함은 해결할 수 없다. 일반적인 소프트웨어 기반 은닉 채널은 제조사의 패치로 인해 은닉 채널이 제거되는 한계점이 존재한다.

소프트웨어 결함은 크게 설계 결함과 구현 결함으로 나누어진다. 버퍼 오버플로우와 같은 구현 결함은 패치가 용이하다. 설계 결함은 소프트웨어 설계를 재구성해야 하므로 비교해 상대적으로 어렵다. 가상화 기술의 설계 결함을 이용하여 은닉 채널을 구성하는 경우, 결함을 해결하기 위해 하이퍼바이저를 다시 설계해야 하므로 즉각적인 패치를 제공하기 어렵다.

설계 결함에 속하는 mutex 공유 자원 간섭은 공유 자원을 제거하기 위해 가상머신 간의 자원 접근을 재설계해야 하는 어려움이 있다. 은닉 채널에 잡음을 발생시키거나, 공유 자원 사용 빈도를 추적하여 mutex 기반 은닉 채널 탐지가 가능할 수도 있지만, 다른 mutex-object를 이용하여 탐지 기법을 회피할 수 있다.

4.4 Mutex-object 탐색 방안

Muex 기반 은닉 채널을 구현하기 위해 Hyper-V에 존재하는 mutex-object를 탐색해야 한다. Hyper-V의 VSP는 VMBus API(Application Programming Interface)를 이용하여 VMBus 내 메시지를 읽는다. VSC의 요청이 발생했을 때, 콜백 함수를 호출해주는 VMBus API가 존재한다. VSP는 VMBus API를 이용하여 메시지 핸들링 함수를 콜백 함수로 등록하면 VSC의 요청이 발생했을 때, 콜백 함수를 새로운 스레드로 호출한다. VSP에서 VMBus API의 사용 위치를 파악하여 메시지 핸들링 함수를 탐색할 수 있다.

메시지 핸들링 함수는 또 다른 서브 함수 여러 개를 호출하는 복잡한 구조로 이루어져 있다. Mutex-object를 탐색하기 위해서는 역공학 도구를 이용하여 전체 함수 스택을 추출한 후 VSP의 메시지 핸들링 함수 전체에서 mutex API를 탐색하여 mutex 기법이 구현된 위치를 파악한다.

VSP의 메시지 핸들링 함수에서 mutex-object는 가상머신에 귀속되어 사용되거나 전역으로 사용되는 두 분류로 나누어진다. Mutex-object가 가상머신에 귀속되어 사용되는 경우, 가상머신 간의 동기화가 아닌 하나의 가상머신 내에서 동기화가 이루어진다. 예를 들어, VSP에 존재하는 A기능과 B기능이 동시에 수행되지 않아야 할 때 동기화를 위해 사용한다. Mutex-object가 전역으로 사용되는 경우, 가상머신 간의 동기화가 이루어진다. 예를 들어, A가상머신과 B가상머신이 VSP에 존재하는 C기능을 동시에 요청할 때 동기화가 필요한 경우 사용한다. 가상머신 간의 은닉 채널 구현 목적에 부합하기 위해서는 전역으로 사용되는 mutex-object를 사용해야 한다. Mutex-object가 전역 변수로 선언된 경우, 하나의 자원을 전역으로 공유하여 사용한다.

4.5 은닉 채널 구현

Mutex-object를 점유하기 위해서는 VSP의 mutex-object를 사용하는 코드 위치가 역공학으로 파악 되었다고 가정하자. VSC에서 VSP에 보내는 메시지 종류에 따라 서로 다른 mutex-object를 선택하고 따라서 실행되는 코드가 상이하다. 게스트 OS에 존재하는 VSC를 조정하여 메시지를 보내면, 해당 mutex-object 코드가 실행되어 은닉 채널 구현이 가능하다. Hyper-V의 linux integration service는 오픈 소스이므로 VSC를 변경하는 것이 가능하다.

4.5.1 데이터 송수신 과정

은닉 채널 송신자와 수신자는 사전에 합의된 프로토콜을 이용하여 통신한다. 프로토콜은 Length, Distance, Threshold를 이용하여 정의한다. Length는 송신자가 전송하고자 하는 메시지 비트열의 길이이다. 송신자는 Length만큼 다음 과정을 반복하여 비트를 전송한다. Distance는 송신자와 수신자가 사전에 합의한 값으로 하나의 비트를 판단하는 기준 시간이다. Distance가 1 ms(milliseconds)일 때, 채널 대역폭은 1000 bps가 된다. 전송하고자 하는 비트가 '1'인 경우, 송신자는 Distance 시간 동안 mutex-object를 점유한다. 전송 비트가 '0'인 경우, Distance 시간 동안 mutex-object를 점유하지 않는다. 수신자는 Distance 시간 동안 mutex-object를 사용할 때의 시간을 지속해서 측정하고 지연 시간의 평균을 계산한다. 평균값이 Threshold 값보다 크다면 비트 '1'

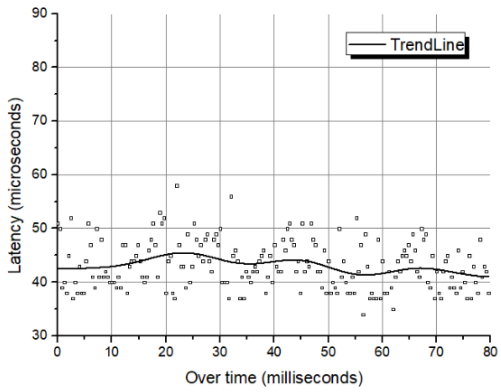

Fig. 1. When the sender is not working, the receiver's measurement result

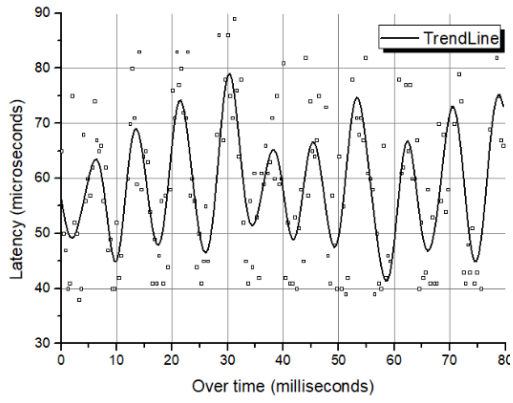

Fig. 2. When the sender repeats 0 and 1, the receiver's measurement result

으로 인식하고, 작다면 비트 '0'으로 인식한다.

수신자가 측정하는 지연 시간은 사용하는 mutex-object와 시스템 환경에 따라 변동된다. Threshold를 고정된 상수로 사용하는 경우, 변동되는 값과 차이가 발생하여 데이터 인식률이 저하된다. Threshold는 시스템 환경에 맞춰 수신자가 동적으로 결정한다. 데이터 송수신을 시작하기 전, 일정 시간 동안 mutex-object를 사용할 때의 평균 지연 시간을 측정한다. 측정된 평균값은 시스템 환경에 따라 상이하다. 평균값을 이용하여 Threshold를 결정하면 시스템 환경에 유동적으로 적응한다.

4.5.2 프로토콜 적용 결과

Fig.1.와 Fig.2.의 x축은 지속 시간이고, y축은 수신자가 측정한 지연 시간을 나타낸다. Fig.1.는 송신자가 아무 작업을 하지 않을 때, 수신자의 측정 결과이다. 송신자가 mutex-object를 점유하지 않는 상황에서 공유 자원 간섭이 발생하지 않아, 수신자의 측정 결과는 일관적이다. Fig.2.는 Fig.1.과 동일한 조건에서 송신자가 0과 1을 반복하여 전송했을 때 수신자의 측정 결과이다. 프로토콜의 Distance는 4ms로 송신자와 수신자가 사전에 합의한 값을 사용한다. 송신자가 0과 1을 반복하여 전송하므로, 수신자의 측정값도 마찬가지로 0과 1로 반복되어 시각적으로 판단이 가능하다.

4.6 대응 방안

공유 자원 간섭이 발생하는 mutex-object를 제거하면 mutex 기반 은닉 채널에 대응이 가능하지만, 불가피하게 가상머신 간의 상호배제를 사용해야 하는 경우 mutex-object를 제거하지 못한다.

Mutex-object를 제거하지 않고 은닉 채널의 통신에 잡음을 발생시키는 대응 방안이 가능하다. Mutex-object 코드가 동일한 코드 실행 시간을 가지지 않고 난수 발생기를 이용하여 무작위의 실행 시간을 가지는 경우, mutex-object가 매번 동일한 시간만큼 점유되지 않고 매번 다른 점유 시간을 가진다. 은닉 채널 수신자의 측정값이 무작위 값으로 측정되어 프로토콜의 Threshold 결정이 무의미해지며 통신에 잡음이 발생한다. 은닉 채널에 대응이 가능하지만, 코드 실행 시간이 증가하므로 은닉 채널 통신이 아닌 일반적인 이유로 mutex-object를 사용하는 경우에도 코드 실행 시간이 증가한다. 코드 실행 시간 증가는 시스템 전반적인 성능 하락으로 이어지는 단점이 있다.

V. 성능 평가

Hyper-V의 VSP에 제안하는 mutex-object 탐색 방안을 적용하였다. 탐색된 mutex-object를 이용하여 가상머신 간에 은닉 채널을 구현하고 프로토콜을 적용하였다. 은닉 채널을 구현한 mutex-object 중 가장 높은 대역폭을 가지는 대상을 선정하여 성능 평가를 진행하였다.

5.1 Mutex-object 탐색 결과

Fig.3.은 Hyper-V의 network, storage, PCI, video, HID VSP를 대상으로

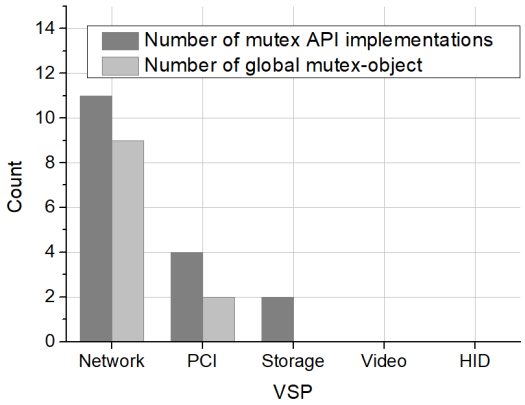

Fig. 3. Mutex-object search result

mutex-object를 탐색한 결과이다. Network VSP에서 mutex API는 11개 사용되며 mutex API에서 mutex-object가 전역으로 사용되는 경우는 9개이다. PCI VSP에서 mutex API는 4개 사용되며 mutex API에서 mutex-object가 전역으로 사용되는 경우는 2개이다. Storage VSP에서 mutex API는 2개 사용되며 mutex API에서 mutex-object가 전역으로 사용되는 경우는 발견되지 않았다. Video와 HID VSP에서 mutex-object가 전역으로 사용되는 경우는 발견되지 않았다.

5.2 은닉 채널 평가

탐색된 mutex-object를 이용하여 은닉 채널을 구현하고 프로토콜을 적용하여 속도를 측정한다. 은닉 채널을 구현하기 위해서는 VSC를 조정하여 VSP에 알맞은 메시지를 전송하여 VSP에 존재하는 mutex-object 코드가 실행되어야 한다. VSC에서 어떤 메시지를 전송해야 하는지 알 수 없는 경우 은닉 채널을 구현할 수 없다. 탐색된 mutex-object 중 VSC에서 알맞은 메시지를 전송하여 mutex-object 코드를 실행시킬 수 있는 대상을 선정하여 은닉 채널을 구현하였다.

송신자는 임의의 메시지를 수신자에게 전송하고, 송신자가 보낸 메시지와 수신자가 받은 메시지를 비

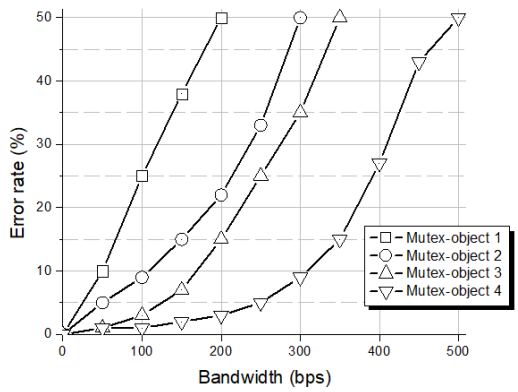

Fig. 4. Bandwidth measurement results of covert channel

교하여 오류율을 측정한다. 통신 대역폭은 프로토콜의 Distance 값에 따라 변화한다. 평가를 진행한 시스템의 CPU는 Intel i7-9700K 이다. 은닉 채널이 동작하는 가상머신에 메모리 4GB와 CPU 2코어를 할당하였으며, 게스트 OS는 CentOS 7 이다.

Fig.4.는 각 mutex-object를 이용하여 은닉 채널을 구현하고 대역폭에 따른 오류율이다. 각 mutex-object에 따라 은닉 채널 대역폭이 다른 결과를 보인다. VSP에서 mutex 기법을 사용할 때, mutex-object를 잠금 설정하고 기능을 수행한 후 잠금을 해제한다. 각 기능에 따라 처리되는 속도 차이가 존재하여 은닉 채널 대역폭에 차이가 발생한다. 모든 은닉 채널은 대역폭이 높을수록 오류율이 증가하는 결과를 보인다.

5.3 잡음 평가

공유 자원을 이용하는 타이밍 채널 방식의 은닉 채널은 송신자와 수신자 이외의 가상머신에서 공유 자원을 사용하는 경우 잡음이 발생한다. Mutex 기반 은닉 채널이 잡음에 어떠한 영향을 받는지 측정한다. 가장 높은 대역폭과 낮은 오류율을 보이는 은닉 채널을 이용하여 송신자와 수신자 이외의 가상머신에서 잡음을 발생시키고 은닉 채널 통신에 대한 영향도를 측정한다. Table 1.은 은닉 채널에 잡음 영향을 끼칠 수 있는 환경에서 250 bps 대역폭으로 통신을 할 때 오류율을 측정한 결과이다. 파일을 다운로드하여 인터넷 통신을 하거나 대용량 메모리를 할당하여 메모리를 점유하는 작업은 mutex 기반 은닉 채널에 영향을 주지 않는다. 파일 I/O를 사용하여 데이터

Table 1. Measurement of error rate in noise environment

Noise	Bandwidth	Error rate
Download internet files	250bps	2%
Memory occupation	250bps	2%
Storage occupation	250bps	3%
CPU occupation	250bps	14%

Table 2. Communication speed measurement result according to system environment

CPU	OS	Bandwidth	Error rate
Intel i7-9700K	Windows 10	250bps	2%
Intel Xeon E3-1230 v3	Windows 10	250bps	6%
ADM Ryzen 7 2700	Windows Server 2016	250bps	5%

저장소를 사용하는 경우 은닉 채널 통신에 미세한 영향을 준다. CPU를 점유하여 잡음을 발생시키는 경우 은닉 채널에 영향도가 크다. CPU를 점유할 경우 mutex-object 코드의 실행 속도가 저하되어 은닉 채널에 영향을 준다. 평가 결과 mutex 기반 은닉 채널은 CPU 점유를 제외한 다른 작업에 대한 영향도가 적다.

5.4 시스템 환경 평가

하드웨어 기반 은닉 채널은 하드웨어 제조사와 모델에 따라 은닉 채널이 구동되지 않는다. Mutex 기반 은닉 채널이 하드웨어 제조사와 모델에 어떠한 영향을 받는지 측정한다. 가장 높은 대역폭과 낮은 오류율을 보이는 은닉 채널을 이용하여 다양한 시스템 환경에 대한 평가를 진행한다. Table 2.는 각 CPU에 따라 은닉 채널의 대역폭과 오류율을 측정한 결과이다. Intel i7-9700K에서 250 bps 대역폭 2% 오류율을 보이고, Intel Xeon E3-1230 v3에서 250 bps 대역폭 6% 오류율을 보인다. AMD Ryzen 7 2700에서 250 bps 대역폭 5% 오류율을 보인다. CPU 제조사와 모델에 따른 실험 결과, CPU 모델에 관계없이 동작 가능하며 하드웨어 의존적이지 않은 특징을 보인다. CPU 모델에 따른 은닉 채널 대역폭과 오류율의 차이는 존재한다. Mutex-object 코드가 실행될 때 CPU 연산 속도에 따라 코드 실행 속도의 차이가 발생한다. CPU 성능이 낮을 경우 mutex-object 코드가 실행되는 속도가 저하되어 은닉 채널 속도에 영향을 준다.

VI. 결 론

본 논문에서는 은닉 채널을 구현하기 위해 소스코드가 공개되어 있지 않은 Hyper-V의 구조를 분석하였다. Hyper-V의 VSP에 존재하는 mutex-object 탐색 방안을 고안하여 은닉 채널을 다수 구현하고 성능을 평가하였다. 실험 환경에서 250 bps 대역폭 2% 오류율의 통신을 달성하였다. 기존 은닉 채널은 하드웨어 공유 자원을 사용하여 하드웨어 의존적이며, 잡음에 민감하다는 한계점이 존재한다. Mutex 기반 은닉 채널은 하드웨어 의존적이지 않고 잡음에 대한 영향도가 적으며 은닉 채널로 사용할 수 있는 mutex-object에 한계가 없어 탐지 또는 방어하기 어렵다.

본 논문에서 제시하는 mutex-object 탐지 방안은 Hyper-V의 API를 기준으로 탐색을 수행한다. Hyper-V가 아닌 다른 하이퍼바이저에 적용할 경우, 대상 하이퍼바이저의 API를 기준으로 변경하면 동일한 과정으로 탐색 가능하다.

클라우드 컴퓨팅 서비스에서 평가를 진행하지 못한 한계점이 존재한다. 클라우드 컴퓨팅 환경에서 은닉 채널을 평가하기 위해 공동 거주가 선행되어야 한다. Hyper-V를 사용하는 Microsoft Azure 클라우드 서비스에서 가상머신을 공동 거주하는 공격 방법이 알려지지 않아 추가 연구가 필요하다.

References

[1] K. Okamura and Y. Oyama, "Load-based covert channels between xen virtual machines," Proceedings of the 2010 ACM Symposium on Applied

Computing, pp. 173-180, Mar. 2010.
〔2〕 Y. Xu, M. Bailey and K. Joshi, "An exploration of L2 cache covert channels in virtualized environments," Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, pp. 29-40, Oct. 2011.
〔3〕 Z. Wu, Z. Xu, and H. Wang, "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud," Proceedings of the 21st USENIX Conference on Security Symposium, pp. 9-9, Aug. 2012.
〔4〕 F. Liu, Y. Yarom and Q. Ge, "Last-level cache side-channel attacks are practical," Proceedings of the 2015 IEEE Symposium on Security and Privacy, pp. 605-622, May. 2015.
〔5〕 D. Sullivan, O. Arias and T. meade, "Microarchitectural Minefields: 4K-Aliasing Covert Channel and Multi-Tenant Detection in IaaS Clouds," Proceedings of the Network and Distributed System Security Symposium, Feb. 2018.
〔6〕 T. Ristenpar, E. Tromer and H. Shacham, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 199-212, Nov. 2009.
〔7〕 Z. Yang and P. Chen, "Exploring virtual machine covert channel via i/o performance interference," Proceedings of the 2013 International Conference on Cloud Computing and Big Data, pp. 232-239, Dec. 2013.
〔8〕 M. Salaün, "Practical overview of a Xen covert channel," Journal in Computer Virology, Vol. 6, No. 4, pp. 317-328, Nov. 2010.
〔9〕 J.Z. Wu, L. Ding and Y. Wang, "Identification and evaluation of sharing memory covert timing channel in Xen virtual machines" Proceedings of the IEEE 4th International Conference on Cloud Computing, pp. 283-291, Jul. 2011.
〔10〕 Q. Shen, M. Wan and Z. Zhang, "A covert channel using event channel state on xen hypervisor," Proceedings of the International Conference on Information and Communications Security, pp. 125-134, Nov. 2013.
〔11〕 Y. Zhang and M.K. Reiter, "Düppel: retrofitting commodity operating systems to mitigate cache side channels in the cloud," Proceedings of the 2013 ACM Special Interest Group on Security, Audit and Control, pp. 827-838, Nov. 2013.
〔12〕 F. Liu, Q. Ge and Y. Yarom, "Catalyst: Defeating last-level cache side channel attacks in cloud computing," Proceedings of the 2016 IEEE International Symposium on High Performance Computer Architecture, pp. 406-418, Mar. 2016.
〔13〕 P. Barham, B. Dragovic and K. Fraser, "Xen and the art of virtualization," Proceedings of the ACM Special Interest Group in Operating Systems operating systems review, pp. 164-177, Dec. 2003.
〔14〕 A. Velte, and T. Velte, "Microsoft Virtualization with Hyper-V," McGraw-Hill, Inc., New York. USA, 448 pages, 2009.
〔15〕 Microsoft Hyper-V Integration Service, "https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/integration-services"

〈저자소개〉

고 기 완 (Ki-Wan Ko) 정회원
2013년 7월~2014년 12월: 라온시큐어 연구원
2015년 2월~현재: 스틸리언 선임연구원
2017년 2월: 국가평생교육진흥원 컴퓨터공학과 학사
2019년 2월: 성균관대학교 전자전기컴퓨터공학과 석사수료
〈관심분야〉 시스템보안, 클라우드 컴퓨팅

최 형 기 (Hyoung-Kee Choi) 정회원
1992년 2월: 성균관대학교 전자공학과 졸업
1996년 2월: Polytechnic University in Brooklyn, NY 석사
2001년 2월~현재: Georgia Institute of Technology in Atlanta, GA 박사
2001년~2004년: Lanscope 근무
2004년 3월~현재: 성균관대학교 소프트웨어대학 교수
〈관심분야〉 네트워크보안, 리버스 엔지니어링