클라우드 메모리 덤프 가속을 위한 API 설계 및 구현
Deploying Cloud API for Cloud Memory Dump Acceleration

최상훈†, 박기웅‡
SangHoon Choi, Ki-Woong Park

† 대전대학교 정보보안학과 시스템보안연구실, csh0052@gmail.com
‡ 대전대학교 정보보안학과 시스템보안연구실, woongbak@dju.kr

요 약

가상머신 메모리 분석기술은 가상머신에 악성코드를 실행하여 행위를 분석하거나 난독화 된 코드를 추출하는데 사용되고 있다. 가상머신 메모리 분석기술은 클라우드 플랫폼이나, 포렌식 분야에서 응용 될 수 있는 유용한 기술이지만, 기존 가상머신 메모리 덤프기술은 덤프를 수행 하는데 시간적 오버헤드가 크다는 문제로 인해 활용되지 못하고 있었다. 본 논문에서는 기존 가상머신 덤프 기술들을 프로파일링하고 분석하여 문제점을 도출하였고, 도출된 문제점을 개선하여 클라우드 플랫폼에서 활용 가능한 메모리 덤프기술 "Thunder"를 제안한다. "Thunder"는 기존 덤프기술 대비 약 8.5배의 성능 향상을 보인다.

1. 서론

2015년 3월 5일 '클라우드 컴퓨팅 발전법'[1]이 통과됨에 따라, 정부가 공공기관에 민간 클라우드 서비스를 적극적 도입하게 됨으로써 클라우드 시장이 큰 주목을 받고 있다. 하지만 클라우드 기술은 보안성이 미흡하다는 문제점이 항상 제시되어 왔다.[2][3] 이러한 클라우드의 보안성 문제를 해결하기 위해 다양한 솔루션들이 제안되어 왔다.[4][5] 기존 솔루션들은 클라우드 플랫폼에 구동되고 있는 가상머신에 에이전트를 설치하여, 가상머신의 도움을 받아 OS를 모니터링 하거나, 가상머신의 도움 없이는 거시적인 관점에서 모니터링 할 수 있는 수준에 머물러 있었다.

가상머신 메모리 분석기술을 이용하면 클라우드 플랫폼에 구동중인 가상머신을 효율적이면서 OS의 내부까지 투명한 모니터링이 가능해진다. 이러한 메모리 정보를 이용하는 가상머신 메모리 분석 연구는 이미 선행되어왔다. BlackHat 2014[6] 에서는 가상머신의 메모리 덤프 분석기술을 이용하여, 사용자가 원하는 API의 매개변수를 추출 하거나 언 패킹 된 코드를 추출 할 수 있는 기술이 발표되었고, DRAKVUF[7]라는 논문에서는 가상머신에서 에이전트 없이 Host OS에서 가상머신 메모리 분석을 통해 Windows 내부의 커널 함수를 추적하거나, 디스크에는 삭제되었으나, 메모리상에 남아있는 파일들을 추출할 수 있는 기술을 발표하였다.

이와 같이 가상머신의 메모리를 분석기술을 이용하면 가상머신에 에이전트 설치 없이 OS의 상세한 부분까지 투명하게 모니터링을 할 수 있게 된다. 하지만 이러한 메모리 분석기술이 클라우드 플랫폼 모니터링 기술 분야에 적용될 수 없었던 가장 큰 이유는 메모리를 덤프 수행의 시간적 오버헤드가 실시간으로 가상머신을 모니터링하기에는 오버헤드가 크다는 문제점이 있었다. 본 논문에서는 메모리 분석기술을 클라우드 플랫폼 모니터링기술로 응용하기 위하여, 기존 가상머신 메모리 덤프 기술을 프로파일링 및 분석하여 문제점을 도출하였고, 도출된 문제점을 개선한 "Thunder"를 제안하였다. 첫 번째로 가상머신의 덤프 속도문제를 개선하기 위하여 기존 QEMU[8] 메모리 덤프 방식을 디스크 I/O 방식에서 메모리 I/O 방식으로 수정하여, 기존 기술대비 약 8.5배의 성능향상 하였고, 두 번째 문제로 지적한 메모리 덤프가 수행되는 동안 가상머신이 일시정지 되는 문제는 메모리 영역에 가상의 버퍼를 생성하여, 사용자가 요청한 메모리 정보를 버퍼에 미러링 후 미러링 된 데이터를 파일로 쓰이게 함으로서 가상머신이 일시되지 않도록 QEMU를 수정하였다. 마지막으로 문제점이 개선된 "Thunder"를 라이브러리 형태로 제공될 수 있도록 API를 구현하여, 실제 클라우드 플랫폼에 응용 될 수 있도록 확장성을 높였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 메모리 덤프 기술 대해서 소개하고, 기존 메모리 덤프기술을 분석 및 프로파일링 하여 문제점을 도출하였다. 3장에서는 프로파일링을 통해 도출된 기존 덤프기술들에 대한 문제점이 발생한 원인에 대해서 분석하고, 문제점을 해결하기 위한 솔루션을 제공한다. 4장에서는 3장에서 제시된 솔루션을 바탕으로 기존 덤프기술과의 성능 비교를 기술하였다. 마지막으로 5장에서는 결론 및 향후 연구를 서술 하였다.

2. 기존 메모리 덤프기술 분석 및 프로파일링

본 장에서는 가상머신 메모리 분석기술을 클라우드 플랫폼 모니터링 기술로 활용하기 위해서 기존 가상머신의 메모리를 덤프기술에 대해 알아보고, 덤프기술에 대해 분석하고 프로파일링 하여 문제점을 도출하였다.

2.1 메모리 덤프기술 조사

2.1.1 LibVMI

LibVMI[9]는 Virtual Machine-Introspection 툴로 구동중인 가상머신의 메모리를 보거나, 하드웨어

이벤트를 감지하거나, vCPU의 레지스터를 엑세스 할 수 있는 도구이다. LibVMI를 이용하여 가상머신의 메모리 덤프를 요청할 경우 가상머신이 운영체제를 구동하기 위한 System Map 파일 등을 가상머신의 커널심볼이 가리키는 가상메모리 주소를 획득하고, 하이퍼바이저를 통해 가상머신의 페이지 테이블에 순차적으로 접근하여 커널심볼의 값을 읽어 오는 방법으로 메모리 덤프를 수행한다.

2.12 Libvirt

Libvirt[10]는 하이퍼바이저[11][12]의 기능들을 API를 통해서 라이브러리 형식으로, 가상머신을 관리 할 수 있는 유틸리티이다. Libvirt는 가상머신을 커맨드를 통해, 제어할 수 있다. Libvirt가 제공하는 다양한 기능 중 가상머신의 메모리 정보를 파일로 추출 할 수 있는 명령이 제공된다. Libvirt는 QEMU에서 제공해주는 메모리 덤프 기능을 이용한다. QEMU는 가상머신의 메모리 정보를 페이지테이블 이용하여 가상머신이 사용하는 메모리영역을 찾고, 메모리를 분석 가능한 포맷형태로 가공하여 파일로 만들어준다.

2.1.3 Gcore & GDB - 프로세스 덤프

OS에서는 하이퍼바이저 위에서 구동되는 가상머신을 프로세스로 처리하기 때문에, 프로세스를 자체를 덤프 하는 방법으로 가상머신의 메모리를 정보를 파일로 기록 할 수 있다. 첫 번째 방법으로는 Linux에서 제공해주는 Gcore 명령어를 이용하는 방법이 있다. 가상머신의 Pid를 추출 후, Pid의 값을 Gcore 명령어에 매개변수를 넣어 수행 할 경우, 프로세스의 메모리정보를 파일로 기록 할 수 있다.

두 번째 방법으로 Linux의 대표적인 디버깅 도구인 GDB를 사용하여 가상머신을 프로세스레벨에서 메모리 정보를 덤프 할 수 있다. 가상머신의 Pid 값을 추출하여, 가상머신(프로세스)을 디버깅 모드 상태로 만든 후 덤프를 원하는 메모리 범위를 지정하면 지정된 범위의 메모리 영역을 파일로 기록 할 수 있다.

2.2 기존 덤프 기술의 프로파일링 및 문제점

2.2.1 프로파일링 결과

프로파일링 환경은 CPU Xeon E5-2609(2.5GHz)를 사용하였고 메모리는 32GB를 사용하였으며, 디스크는 SSD를 사용하였으며, Ubuntu 14.04에 KVM을 설치하였다, 가상머신은 OS는 Windows XP SP2 - 32bit를 사용하였으며, 1GB의 메모리를 할당하여 실험을 진행하였다.

프로파일링 결과는 <표1>과 같다. Dump Performance는 가상머신 메모리 1GB에 대해 덤프를 수행할 경우 요청부터 완료까지의 시간을 의미하며, Suspend Time은 메모리 덤프를 수행하는 동안 가상머신이 일시정지 되는 시간을 측정하였다. 마지막으로 다른 플랫폼이나 분야에서 응용가능 여부를 파악하기 위해 라이브러리 제공여부를 표기하였다.

<표1> 기존 가상머신 메모리 덤프기술 프로파일링

덤프기술	DUMP Performance	Suspend Time	Library
LibVMI	15,254,220ms	Stealth	△
Libvirt	5,110ms	4,447ms	△
GCORE	2,111ms	2,111ms	X
GDB	5,533ms	5,533ms	X

2.2.2 문제점

프로파일링 결과 첫 번째 문제점은 기존 덤프기술은 가상머신 메모리를 파일로 추출하는데 있어서 시간적 오버헤드가 크다는 문제점이 있었다. 가상머신의 메모리 덤프가 실시간으로 이루어 질 수 있게 되면, 클라우드 플랫폼에서 하이퍼바이저 위에 구동되고 있는 가상머신 모니터링도구로 사용 할 수 있게 된다. 메모리분석 기반 모니터링 기술은 가상머신에 별도의 에이전트 설치가 필요 없기 때문에, 가상머신 OS에 국한되지 않고 내부까지 투명하게 모니터링 할 수 있다는 장점을 가지고 있다.

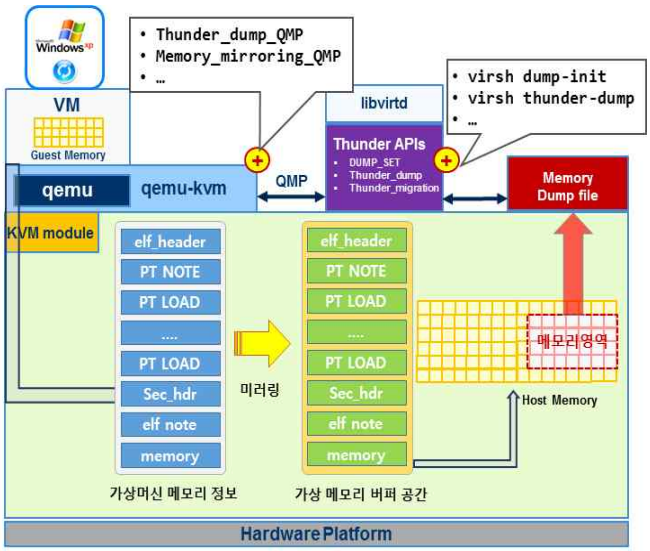
두 번째 문제점은 덤프 수행동안 가상머신이 일시정지 되는 문제이다. 메모리 덤프기술은 사용자가 요구한 시점에 정확히 덤프가 되어야 해당 가상머신으로부터 원하는 정보를 추출 할 수 있기 때문에 사용자가 메모리 덤프를 요청한 시점에는 메모리가 보존되어야하고, 보존된 상태에서 메모리정보를 파일로 써야하기 때문에, 덤프가 수행되는 동안 가상머신이 일시정지 되는 문제가 발생하게 된다. 이러한 문제가 발생하면, 모니터링 되는 동안 사용자는 자신이 이용하고 있는 가상머신이 일시적으로 멈추기 때문에 가상머신을 사용하는 사용자 입장에서는 클라우드 서비스에 대한 불편함을 즉각적으로 느끼게 된다.

세 번째 문제는 클라우드 플랫폼에서 사용할 수 있도록 라이브러리의 추가가 필요하다는 점이다. 기존 하이퍼바이저 도구들은 간단한 API 수준에서 메모리덤프 기능을 제공하나, 클라우드 플랫폼에 특화된 API는 설계되어 있지 않다. 클라우드 플랫폼에서 메모리 덤프 기술을 모니터링 도구로써 활용하기 위해서 다수의 가상화 컴퓨트를 이용하는 클라우드 플랫폼 특징이 고려된 라이브러리가 추가 제공되어야 한다.

3. 프로파일링을 통해 도출된 문제점 해결 및 API 디자인

본 논문에서는 메모리 덤프 가속화에 초점을 맞추었다. KVM(QEMU)을 사용 하였고, 가상머신의 메모리 덤프 방법으로는, 하이퍼바이저와 가장 밀접

한 관련이 있는 API 라이브러리 도구인 Libvirt를 사용하였다.



(그림 1) 덤프 가속화 기술이 적용된 “Thunder” 구조

3.1 메모리 덤프 가속화 기술 “Thunder”

가상머신 메모리 덤프 기술을 클라우드 환경에서 응용하는데 있어서 가장 첫 번째 문제점은 가상머신의 메모리를 파일로 덤프 하는데 걸리는 수행시간의 오버헤드가 커서, 실시간으로 메모리 정보를 분석 할 수 없다는 문제이다. 본 논문에서는 사용한 Libvirt는 QEMU에서 제공해주는 기능을 라이브러리 형태로 사용할 수 있도록 제공해주는 연결도구이기 때문에, 실제 가상머신 메모리 덤프 관련된 코드는 QEMU에 구현되어 있는 기능을 그대로 사용한다. QEMU에 메모리 덤프 코드를 분석하고, 프로파일링 한 결과 덤프 수행시간이 지체되는 주원인은 디스크 I/O 로부터 발생 되었다. QEMU 덤프 기능은 구동되고 있는 머신의 물리 메모리로부터 가상머신이 사용하고 있는 프로세스의 메모리 정보를 분석 가능한 형태의 포맷으로 정리하여 파일로 쓸 수 있게 해준다. 이 과정에서 디스크와 메모리의 I/O 처리 속도 차이로 인해 시간적 오버헤드가 발생하게 된다. 본 논문에서는 디스크 I/O로부터 발생하는 오버헤드를 최소화하기 위하여, QEMU의 구조를 수정하였다. QEMU에서 사용하는 기존 가상머신의 메모리정보를 다른 메모리영역에 쓰이게 함으로써 I/O로부터 발생 하는 오버헤드를 최소화 하여, 메모리 덤프 가속화를 이뤄냈다.

3.2 VM일시정지 솔루션 “메모리 미러링 기술”

기존 가상머신 메모리 덤프기술은 사용자가 덤프를 요청하는 순간부터 덤프가 완료되는 순간까지 가상 머신 일시정지 된다는 문제가 있다. 이러한 문제점은 QEMU에서 가상머신의 메모리정보를 디스크로 쓰기 위하여, 메모리의 상태를 보존 하고, 보존된 메모리가 파일로 추출 될 때까지 가상머신을 강제로 일시정지 시키기 때문에 발생 하는 문제이다. 이러한 문제를 해결하기 위해 본 논문에서는 가상의 메모리 영역을 만들어 메모리 정보를 미러링 하는 방법을 사용 하였다. 사용자가 가상머신의 메모리 덤프를 요청하면 가상의 메모리 버퍼공간을 만들어, 구동중인 가상머신 메모리 정보를 메모리 버퍼공간에 미러링 한다. 미러링 된 버퍼에 있는 메모리정보를 디스크에 파일로 추출되게 함으로써, 메모리 덤프 수행동안 가상머신이 일시정지 되는 현상을 느낄 수 없도록 QEMU의 메모리 덤프 구조를 재설계 하였다.

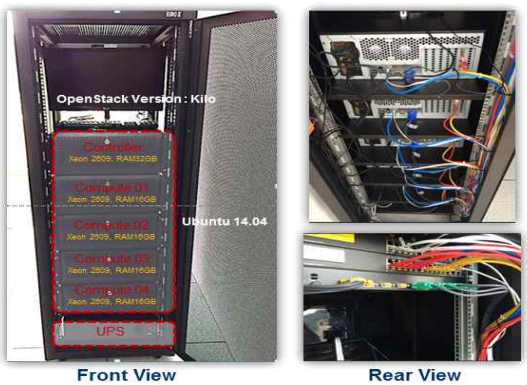
3.3 다양한 플랫폼에서 응용하기 위한 API 설계

클라우드 플랫폼에서 메모리 덤프 가속화 기술 및 메모리 미러링 기술을 사용 할 수 있도록 QEMU 재설계하고 추가된 기술에 대해 타 라이브러리 도구에서 접근 가능하도록 QMP(Qemu Monitor Protocol)를 만들었다. 또한 만들어진 QMP를 클라우드 플랫폼에서 활용 할 수 있도록 라이브러리 형태로 사용할 수 있도록 Libvirt에 API를 추가 해줌으로써, 클라우드 플랫폼이나 다른 플랫폼 환경에서도 응용 가능하도록 확장성을 높였다.

4. 구현된 Thunder 성능 평가

본 장에서는 가상머신의 메모리 덤프 속도 가속화를 위해 설험한 환경과 성능비교 결과를 나타낸다.

4.1 실험환경



(그림 2) 성능평가를 위한 실험환경

실제 클라우드 환경에서 응용이 가능한지 실험하기 위해 (그림 2)와 같이 IaaS 플랫폼을 구축하여 실험을 진행 하였다. 실험환경은 5대로 서버를 구성하였으며, CPU Xeon E5-2609(2.5GHz)를 사용하였고 5대중 1대는 Memory 32GB를 사용하였으며 4대는 Memory 16GB를 사용한다. Storage는 SSD를 사용하였다. Host OS는 Ubuntu 14.04를 사용하였고, 실 IaaS 플랫폼은 2015년 4월에 릴리즈된 OpenStack

Juno[11]환경에서 실험하였고, 가상머신은 Windows XP SP2 - 32bit를 사용하였고 메모리는 1GB를 할당 하였다.

4.2 분석 결과

본 장에서는 구현된 Thunder와 기존의 기존 메모리 덤프기술의 성능을 비교한다.

<표2> 수정된 Libvirt와 기존 덤프기술 성능비교

덤프기술	DUMP Performance	Suspend Time	Library
Thunder	**595ms**	**Stealth**	**O**
LibVMI	15,254,220ms	Stealth	O
Libvirt	5,110ms	4,447ms	O
GCORE	2,111ms	2,111ms	X
GDB	5,533ms	5,533ms	X

첫 번째 성능비교는 메모리 1GB에 대해 기존 덤프 기술과 덤프 기술 가속화가 적용된 Thunder의 덤프 수행시간을 비교하였다. <표 2> 와 같이 Thunder의 경우 1GB의 메모리를 덤프 하는데 약 0.59초 소요 되었다. LibVMI 대비 약 256배, Libvirt 대비 약 8.5배의 덤프 가속화를 이뤄냈다. 두 번째 실험의 경우 메모리 덤프 시 가상머신이 일시정지 되는 Suspend Time을 측정하였다. 또한 다른 플랫폼에서 개발된 기능을 라이브러리 형태로 사용가능하도록 API를 만들어 응용의 확장성을 높였다.

5. 결론

본 논문에서는 가상머신의 메모리 분석의 응용을 위해 메모리 덤프 가속화 기술에 대해서 제시하였다. 기존 메모리 분석을 통해 모니터링 하거나 악성코드를 분석하기 위한 가장 기본이 되는 기술인 메모리 덤프 기술의 3가지 문제점에 대해서 , 연산처리의 대상을 수정함으로써 문제를 해결 하였다. 결론적으로 기존 덤프기술 대비 약 8.5배의 의 메모리 덤프 가속화를 이뤄 냈고, 메모리를 덤프하는동안 가상머신이 일시정지 되는 문제를 해결하였다. 본 논문에서는 메모리 덤프 가속화 기술 Thunder를 제시 하였고, 실제 클라우드 플랫폼에서 확장성 테스트를 통해 클라우드 모니터링 도구로 사용될 수 있음을 증명하였다. 향후연구에서는 본 논문에서 제시된 기술을 바탕으로 실제 클라우드 플랫폼에서 에이전트 없이 구동중인 가상머신에 대한 모니터링이 가능하도록 실시간 모니터링 기술에 대해 연구하여 클라우드 플랫폼의 보안솔루션으로 제안할 예정이다.

References

[1] 클라우드 발전법: http://www.ddaily.co.kr/news/article.html?no=128017

[2] 클라우드 보안성 미흡문제: http://www.ddaily.co.kr/news/article.html?no=123228

[3] Balachandra R K, Ramakrishna P V, Dr. Rakshit A, 'Cloud Security Issues', 2009 IEEE International Conference on Services Computing, viewed 26 October 2009, pp 517-520.

[4] S. Ramgovind, M. M. Eloff, E. Smith. "The Management of Security in Cloud Computing" In PROC 2010 IEEE International Conference on Cloud Computing 2010

[5] Ciphercloud, http://www.ciphercloud.com/

[6] Tomer Teller, Enhancing Automated Malware Analysis Machines with Memory Analysis Blackhat 2014

[7] DRAKVUF, http://drakvuf.com/

[8] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In Proceedings of the USENIX 2005 Annual Technical Conference, FREENIX Track, pages 41 - 46, 2005.

[9] Haiquan Xiong et al. 'Libvmi: A Library for Bridging the Semantic Gap between Guest OS and VMM'. In: (2012), pp. 549 - 556.

[10] M. Bolte, M. Sievers, G. Birkenheuer, O. Nieh¨orster, and A. Brinkmann. Non-intrusive virtualization management using libvirt. In DATE '10, 2010.

[11] A. Kivity. kvm: the Linux Virtual Machine Monitor. In OLS '07: The 2007 Ottawa Linux Symposium, pages 225 - 230, July 2007

[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164 - 177, New York, NY, USA, 2003. ACM.

[11] Openstack, https://www.openstack.org/