KIPS Tr. Comp. and Comm. Sys.
Vol.3, No.1 pp.23~30 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2014.3.1.23

An Efficient Agent Framework for Host-based Vulnerability Assessment System in Virtualization Environment

Jin-Seok Yang† · Tai-Myoung Chung††

ABSTRACT

In this paper, we propose an efficient agent framework for host-based vulnerability assessment system by analyzing the operational concept of traditional vulnerability assessment framework and proposed vulnerability assessment agent framework in virtualization environment. A proposed agent framework have concept by using the features of virtualization technology, it copy and execute checking agent in targeted virtual machines. In order to embody a propose agent framework, we design function block of checking agent and describe a vulnerability checking scenario of proposed agent framework. Also we develop pilot system for vulnerability checking scenario. We improve the shortcomings of the traditional vulnerability assessment system, such as unnecessary system load of the agent, inefficiency due to duplication agent installation. Moreover, the proposed agent framework is maximizing the scalability of the system because there is no agent installation when adding a targeted system.

Keywords : Virtualization, Vulnerability Assessment System, Host-based Vulnerability Assessment System

가상화 환경에서 호스트 기반 취약점 분석 시스템을 위한 효율적인 에이전트 프레임워크

양 진 석† · 정 태 명††

요 약

본 논문에서는 가상화 환경에서 호스트 기반 취약점 분석 시스템의 에이전트 프레임워크를 제안한다. 기존 호스트 기반 취약점은 점검 대상 시스템에 에이전트를 반드시 설치하여 운용해야만 했지만 제안한 에이전트 프레임워크는 점검 대상 가상머신에 에이전트를 복사 및 실행하는 개념을 가진다. 본 논문에서는 가상화 환경에서 호스트 기반 취약점 분석 시스템 에이전트 프레임워크를 수립하였으며 점검 시나리오를 기술하고 해당 시나리오에 대한 파일럿 시스템을 개발하였다. 제안하는 에이전트 프레임워크는 점검 대상 시스템 추가 시에도 에이전트의 추가 설치가 없어 시스템 확장성을 극대화하였으며 에이전트의 불필요한 자원 낭비 등 기존 시스템의 단점을 개선하였다.

키워드 : 가상화, 취약점 분석 시스템, 호스트 기반 취약점 분석 시스템

1. 서 론

가상화는 클라우드를 구현하기 위한 필수적인 기술로써 물리적인 컴퓨팅 자원을 논리적으로 나누어 사용자에게 하나의 물리적인 자원을 서로 다른 서버, 운영체제 등의 장치로 보이게 하는 기술이다[1].

가상화 기술을 사용한 컴퓨팅 자원은 논리적으로 CPU, 스토리지, 네트워크 등의 자원을 분할하여 사용하기 때문에 논리적 자원에 대한 관리가 복잡하다. 이에 가상화 관련 업체들은 가상화된 논리적 자원들의 관리 편의성을 위해 가상화 관리 API(Application Programming Interface)를 제공하여 사용자에게 원격으로 관리할 수 있도록 한다.

취약점 분석은 10여 년 전에 이미 개념이 정립되어 다수의 점검 도구가 개발된 상황이고 현재까지도 시스템의 구조 및 개념의 변화 없이 사용되고 있다[2, 3, 4]. 최근 연구되고 있는 가상화와 관련된 취약점 분석은 그 대상이 기존의 인프라가 아닌 하이퍼바이저(Hypervisor)나 데이터 공유 등 가상화 환경에서 새롭게 추가되는 구성 요소와 구조적인 특

※ 본 연구는 산업기술연구회 및 ETRI부설 국가보안기술연구소, 엘에스웨어㈜의 기술인재지원사업의 일환으로 수행하였음.
[ITAH2801120110010001000100100, 기술인재지원사업]
† 정 회 원 : 한국전자통신연구원 부설연구소 선임연구원
†† 종신회원 : 성균관대학교 컴퓨터공학과 교수
논문접수 : 2013년 4월 29일
수 정 일 : 1차 2013년 10월 30일, 2차 2013년 12월 9일
심사완료 : 2014년 1월 3일
* Corresponding Author : Jin-Seok Yang(jsyang@ensec.re.kr)

징을 반영한 취약점 분석 시스템을 제안하고 있다[5, 6, 7].

본 논문에서 제안하는 가상화 환경에서 취약점 분석 시스템을 위한 에이전트 프레임워크는 가상화 관리 API를 이용하여 취약점 점검 모듈을 원격에서 복사 및 실행함으로써 취약점 분석 시스템 관리 편의성 및 확장성을 높이고 논리적 컴퓨팅 자원의 불필요한 사용을 감소시킬 수 있다.

제안하는 에이전트 프레임워크는 기존의 취약점 분석 시스템과 달리 가상머신 관리 API를 이용하여 에이전트를 한 번만 설치하여 사용자 개입을 최소화하고 시스템 확장성을 극대화하며 기존 취약점 분석 시스템에서 사용하는 불필요한 자원 낭비를 최소화하는 장점을 가진다.

본 논문의 2장에서는 가상화 기술과 취약점 분석 시스템에 대해서 살펴본다. 3장에서는 기존 취약점 분석 시스템의 운영 개념과 제안하는 취약점 분석 시스템의 운영 개념의 비교를 통해 제안하는 프레임워크의 장점을 기술한다. 4장에서는 새로운 프레임워크의 에이전트 구조 및 시나리오를 기술하고 이를 적용하여 실험적으로 개발한 파일럿 시스템에 대해서 기술한다.

2. 관련 연구

2.1 가상화

가상화는 물리적인 자원을 논리적으로 나누어 사용자에게 여러 개의 자원으로 보이게 하는 기술이다. 가상화에는 하나의 시스템에 여러 개의 운영체제를 실행 가능하도록 구현한 플랫폼 가상화와 메모리, 저장 장치, 네트워크 등을 여러 개의 논리 단위로 나누는 리소스 가상화가 있다.

Fig. 1은 플랫폼 가상화의 종류 중 전가상화를 구현하는 두 가지 방식을 나타낸다. Bare metal 방식은 native virtualization 이라고도 하며 하이퍼바이저가 호스트 운영체제 없이 하드웨어 위에서 실행된다. Hosted 방식은 하이퍼바이저가 윈도우즈, 리눅스 등의 호스트 운영체제 위에서 실행된다. 대부분의 클라우드 컴퓨팅은 상대적으로 성능이 우수한 Bare metal 방식으로 구축한다[8].

가상화 기술 개발 업체들은 논리적으로 공유하는 컴퓨팅 자원의 관리 편의성을 위해 Table 1의 원격 관리 API를 제공하여 가상화 자원들을 쉽게 관리할 수 있도록 한다[9, 10, 11].

Table 1. Virtualization Management APIs

업체명	대표적인 API
VMWare	VixHost_Connect : 호스트 가상머신에 연결 VixVM_Clone : 가상머신을 복제 VixVM_CopyFileFromHostToGuest: 게스트 가상머신에서 호스트 머신으로 파일을 복사 VixVM_RunProgramInGuest:게스트 가상머신에서 프로그램을 실행 VixVM_CopyFileFromGuestToHost: 게스트 가상머신에서 호스트 머신으로 파일을 복사
Citrix	VM.start_on : 정지된 상태의 가상머신을 시작 VM.pause : 실행 상태의 가상머신을 잠시 멈춤 VM.clone : 가상머신을 복제 VM.copy : 가상머신을 복사 VM.reboot : 가상머신을 재시작
LibVirt (오픈소스)	virDomainProcessSignal : 원격으로 가상머신에 kill signal 등을 실행 virDomainShutdown : 가상머신을 셧다운

2.2 취약점 분석 시스템

취약점 분석 시스템은 운영체제, 서비스 등의 알려진 취약점에 대한 취약 여부를 점검하는 시스템이다.

취약점 분석 시스템은 분석을 수행하는 위치에 따라 네트워크 기반의 취약점 분석 시스템과, 호스트 기반의 취약점 분석 시스템으로 나뉜다. 네트워크 기반의 취약점 분석 시스템은 분석 대상 시스템에 열려 있는 포트를 알기 위해 원격에서 포트 스캐닝을 수행하고, 열린 포트별로 명령어를 전송하여 응답된 정보를 기반으로 취약점 분석을 수행한다. 따라서 네트워크 기반의 취약점 분석은 분석 대상 시스템에 점검을 위한 프로그램을 따로 설치할 필요가 없다[3].

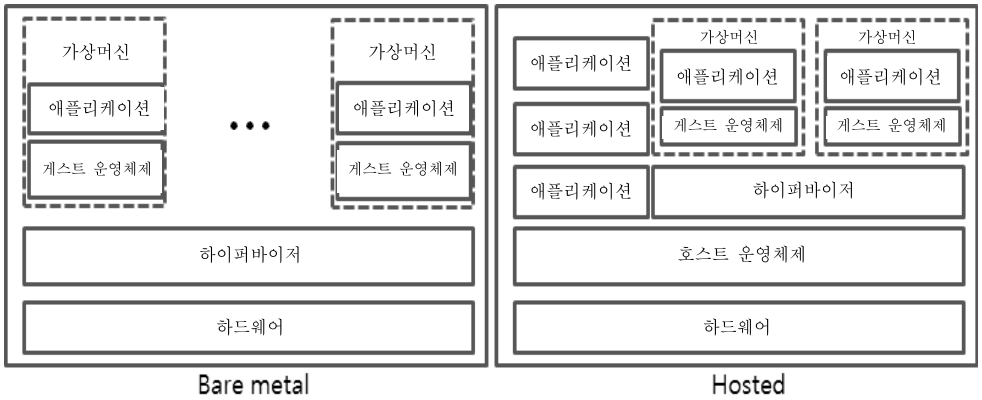

Fig. 1. Full Virtualization Architectures

호스트 기반의 취약점 분석 시스템은 운영체제 및 서비스 등에 대한 취약한 설정을 로컬에서 점검하는 시스템으로 운영체제의 중요한 파일, 디렉토리의 소유자 및 권한 설정을 비롯한 Apache, IIS(Internet Information Service) 등의 웹 서비스, FTP(File Transfer Protocol) 서비스, 메일 서비스 등의 서버에서 실행 중인 서비스의 취약한 설정을 점검하여 분석한다.

취약점 분석 시스템은 이미 10여전부터 그 개념과 이를 구현한 도구들이 다수 개발되었으며 최근에는 기존의 취약점 분석 도구들을 통합하여 자동화하거나 시스템 취약점 분석을 통해 침투 경로 예측을 자동화하는 등 기존의 취약점 분석 프레임워크를 자동화하는 연구를 진행하고 있다[12, 13].

3. 취약점 분석 시스템의 비교 분석

이번 장에서는 기존 취약점 분석 시스템과 제안하는 취약성 분석 시스템의 에이전트를 비교 분석한다. 또한 상용 취약점 분석 시스템과의 비교 분석을 통해 제안하는 프레임워크의 장점을 기술한다.

3.1 기존 취약점 분석 시스템

Fig. 2의 기존 호스트 기반 취약점 분석 시스템은 각각의 서버에 취약점 분석 에이전트를 설치하게 된다. 이는 호스트 기반 취약점 분석 시스템의 특징으로 점검 대상 서버의 취약점을 분석하는 시스템으로 관리자 권한으로 실행되어야 하기 때문이다[3, 14, 15].

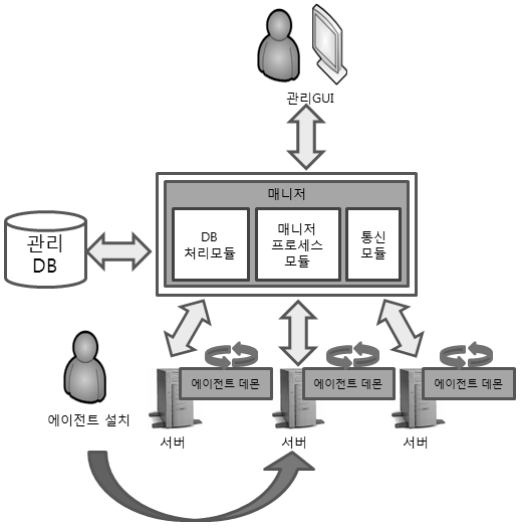

Fig. 2. Concept of Existing Host-based Vulnerability Assessment System

대부분의 점검 서버는 보안성이 높기 때문에 물리적인 보안도 매우 강화된 곳에 위치하기 때문에 설치 및 유지보수를 위해 접근 승인 요청 및 허가 등의 복잡한 절차를 거쳐야 한다. 또한 에이전트 설치 이외에 설정 변경, 업데이트, 삭제 등의 경우에도 동일한 절차를 거치기 때문에 관리 및 유지보수가 매우 불편하다[17].

설치된 에이전트는 매니저의 명령을 수신하기 위해 데몬 형태의 프로그램으로 실행되고 명령을 수신하여 운영체제 및 중요 서비스의 취약점을 점검하게 된다[3, 14]. 데몬 형태의 프로그램은 취약점 점검을 하지 않음에도 불구하고 서버의 자원을 지속적으로 점유하기 때문에 물리적인 자원을 논리적으로 분할하여 쓰는 가상화 환경에서는 단점이 된다. 또한 에이전트가 사용하는 열린 포트로 해킹 공격이 발생 가능하여 24시간 동작해야하는 서버에 위협이 될 수 있다.

3.2 제안하는 취약점 분석 시스템

제안하는 호스트 기반 취약점 분석 시스템은 Fig. 3과 같다.

Fig. 3의 제안하는 취약점 분석 시스템은 가상화 관리 API를 사용하여 원격에서 점검 모듈을 복사 및 전송하여 실행시키므로 에이전트 설치, 업데이트 등 사용자의 개입이 크게 감소한다. 즉, 점검 대상 서버 추가 시 사용자가 복잡한 절차를 거치지 않고도 원격으로 점검이 가능하게 된다. 업데이트 시에도 원격으로 가능하기 때문에 관리의 편의성과 시스템 확장성이 증대된다.

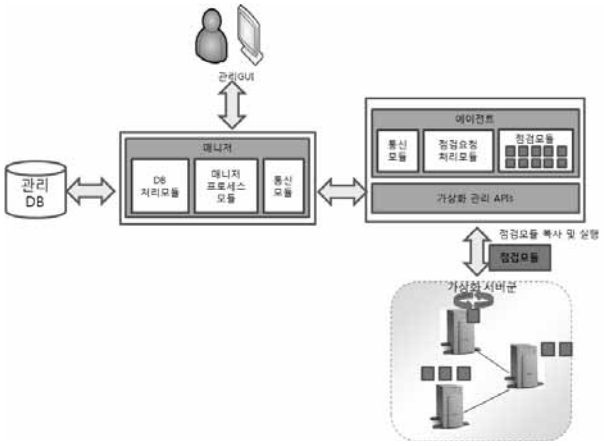

Fig. 3. Concept of Proposed Host-based Vulnerability Assessment System

뿐만 아니라 제안하는 에이전트는 점검 필요시에만 에이전트가 실행되므로 기존 데몬 형태의 취약점 분석 에이전트의 단점인 CPU, 메모리 자원의 불필요한 낭비가 없다.

3.3 제안하는 에이전트 프레임워크의 비교 분석

Table 2는 기존 취약점 분석 시스템과 제안하는 시스템을 자원 및 시스템 확장성 측면에서 비교한 것이다.

먼저 자원 항목을 비교하여 보면, 기존의 취약점 점검 에

이전트는 점검 정책 및 결과 송수신을 위해 점검 대상 서버에 데몬 형태로 실행되어 점검하지 않을 때도 부하를 일으킨다. 그러나 본 논문에서 제안한 구조는 점검 대상 서버에 취약점 점검 시에만 모듈화된 점검 프로그램을 실행하여 불필요한 부하가 없다. 이를 위해 Table 3은 A社에서 개발한 상용 취약점 분석 시스템의 에이전트 실행 파일 크기 및 실행 시 메모리 사용량을 조사하여 기술하였다.

Table 2. Comparison of traditional and proposed agent

	기존 에이전트	제안하는 에이전트
자원 (메모리 및 CPU)	데몬 형태의 프로그램으로 지속적으로 자원 점유	점검 시에만 자원 점유
시스템 확장성	점검 대상 서버가 추가될 때마다 관리자가 직접 설치	점검 대상 서버가 추가되면 원격으로 복사 및 실행

Table 3의 에이전트의 메모리 사용량은 평시 21.4MB이고 실행 파일 크기는 35.9MB이다.

기존 취약점 분석 시스템은 구조적인 특성 상 데몬으로 실행되기 때문에 아무런 작업이 없어도 메모리를 점유하고 있다[6]. 또한 점검 시에는 더 많은 메모리를 소진하게 된다. 평시 메모리 사용량을 기준으로 약 100대의 서버에서 약 2.1G 정도의 메모리 사용량을 절약할 수 있다.

또한 에이전트의 실행 파일 크기는 35.9MB이고 약 100대의 서버를 기준으로 약 4GB의 저장 공간을 절약할 수 있다. 나일소프트社의 경우 에이전트 설치를 위해 필요한 하드디스크가 100MB인데 약 100대의 서버를 기준으로 약 10GB의 저장 공간을 절약할 수 있다[16].

Table 3. Resource usage of traditional agent

A社에이전트		크기(MB)
실행 파일 크기		35.9
메모리 사용량	평시	21.400
	점검시	25.780

가상화 환경에서는 물리적인 메모리 및 하드디스크를 논리적으로 나누어 사용하는 환경이기 때문에 메모리 및 스토리지 자원을 절약하는 것이 매우 중요하다.

시스템 확장성 측면에서 비교해 보면 기존의 취약점 분석 시스템은 점검 대상 시스템이 추가되면 반드시 에이전트를 설치해야만 한다.

본 논문에서 제안한 시스템은 에이전트 설치 없이 대상 시스템의 정보(IP 혹은 호스트네임, 관리자 계정, 관리자 비밀번호)만 알면 점검 모듈 복사 및 실행이 가능하므로 시스템 확장성 및 관리 효율성이 크게 증대된다.

그러나 점검 모듈이 복사 및 실행되는 과정에서 네트워크 대역폭을 점유하는 단점이 존재한다. 만약 점검하는 대상 시스템이 많을 경우 이러한 단점은 사용자의 네트워크 접속이 느려지거나 사용할 수 없게 되는 문제를 발생시킬 수 있다. 이러한 단점은 두 가지 해결책이 있다.

첫 번째, 취약점 분석 시스템은 대부분 점검 대상 서버의 취약점 점검을 자주 하지 않기 때문에 각 서버별로 점검 모듈을 복사하는 시간을 다르게 하거나 대역폭을 비교적 적게 사용하는 시간에 실행하여 문제를 해결할 수 있다.

두 번째, 점검 대상 서버에 최초 한번 복사된 점검 모듈을 삭제하지 않는다면 에이전트가 업데이트되는 경우를 제외하고 복사로 인한 대역폭 점유는 더 이상 발생하지 않는다. 따라서 점검 모듈 복사 이후에 삭제하지 않고 사용하면 대역폭 점유를 피할 수 있다. 부득이하게 점검 모듈을 삭제할 경우에는 첫 번째 해결책을 적용하도록 한다.

상기 기술한 두 가지 해결책은 각각 적용할 수도 있고 동시 적용할 수도 있다. 따라서 이러한 정책을 적용하면 네트워크 대역폭에 대한 문제를 해결할 수 있다.

4. 가상화 환경에서 제안하는 에이전트

4.1 에이전트 아키텍처

본 논문에서 제안하는 취약점 분석 시스템의 에이전트 아키텍처는 Fig. 4와 같다.

가상화 환경에서 취약점 분석 에이전트는 통신 모듈, 점검요청명령분기모듈, 점검모듈관리모듈, 점검모듈 선택모듈, 점검결과 관리 모듈 그리고 대상 서버의 점검 모듈들로 구성된다.

먼저 점검 모듈들은 점검 대상 호스트의 취약점을 점검하는 프로그램이다. 점검 프로그램들은 점검 대상 서버의 알려진 취약점을 점검하는 실행파일로 운영체제 및 서비스 정보 확인 모듈들, 운영체제별 취약점 점검 모듈들, 운영체제별 서비스 취약점 점검 모듈들 등으로 점검 목적 및 대상에 따라 분리하여 개발할 수 있다. 예를 들어 Fig 4의 점검모듈들 중 웹서버 점검 모듈은 리눅스용과 윈도우즈용으로 나누거나 아파치 웹서버용, IIS 웹서버용을 분리된 모듈로 개발할 수 있다. 상기 취약점을 점검하는 모듈화된 점검 프로그램은 점검 후 점검 결과를 가상 머신에 파일로 임시 저장한다. 통신 모듈은 취약점 관리 매니저와 통신을 수행하는 통신 모듈로 점검 명령 및 결과와 점검 정책을 송수신하는 기능을 수행한다. 점검요청명령분기모듈은 매니저에서 송신된 명령들을 요청에 따라 분기시키고 처리 결과를 통신 모듈에 반환하는 기능을 수행한다. 점검모듈관리모듈은 가상화 관리 API를 호출하여 점검모듈들의 복사, 실행, 종료, 삭제 등의 관리를 수행하고 각 가상 머신의 취약점 점검 결과를 취약점 분석 에이전트로 복사해오는 기능을 수행한다. 여기서 가상화 관리 API는 모듈화된 점검 프로그램의 복사,

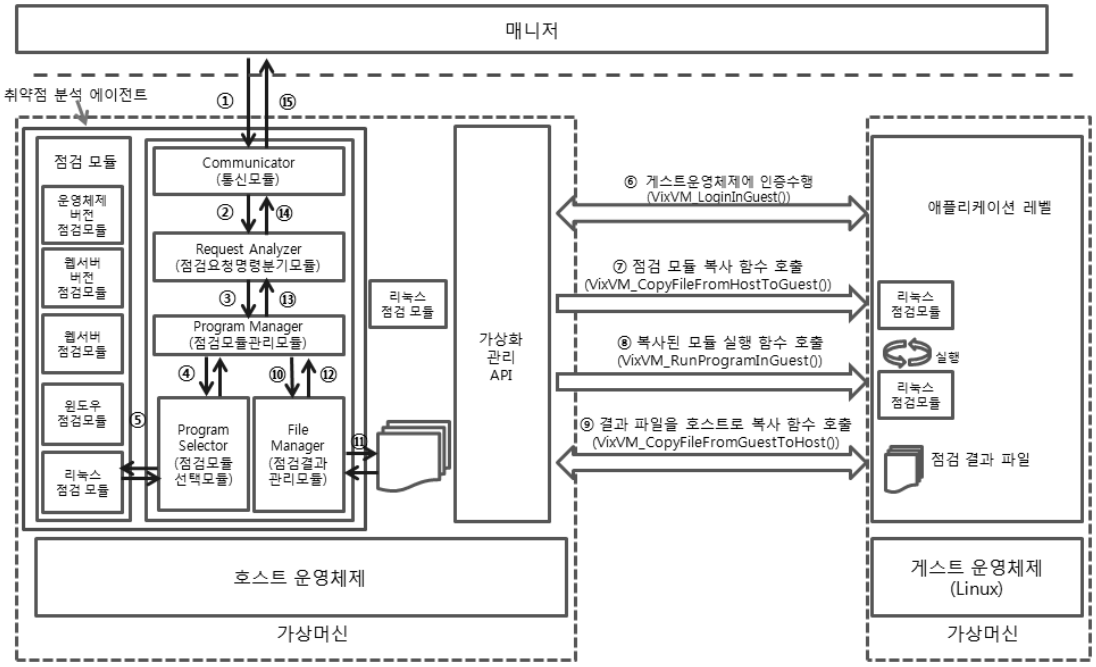

Fig. 4. Architecture of proposed host-based Vulnerability Assessment Agent

실행, 종료, 삭제 등 원격 가상머신 관리를 위한 인터페이스를 제공한다. 점검모듈선택모듈은 점검 모듈들 중 점검 명령에 따른 점검 모듈을 선택하여 점검모듈관리모듈로 반환한다. 점검결과관리모듈은 파일 형태의 점검 결과를 수신하면 점검 결과 파일을 파싱하여 점검모듈관리모듈로 반환하는 기능을 수행한다.

Fig. 4의 구조는 매니저와 관리자 UI의 구조는 기존과 동일하나 취약점 분석 에이전트는 기존의 구조와 달리 점검 대상 서버가 실행되고 있는 각 가상 머신에 설치되지 않고 점검모듈들을 복사하여 실행하는 구조를 가진다.

Fig. 4의 제안하는 에이전트를 중심으로 리눅스 운영체제에 대한 취약점 분석 시나리오는 다음과 같다.

① : 매니저가 게스트 운영체제(Linux)에 대한 운영체제 취약점 점검 요청을 수행

② : 에이전트의 통신 모듈이 요청을 수신하여 점검요청 분기 모듈로 전달

③ : 점검요청분기 모듈은 점검 요청을 분기하여 점검모듈관리 모듈로 전달

④ : 점검 요청에 따른 프로그램을 선택하기 위해 점검모듈선택모듈에 점검 모듈 선택 요청

⑤ : 점검모듈선택 모듈은 다수의 점검 프로그램 중 리눅스 점검 모듈을 선택하여 점검모듈관리모듈로 반환

⑥ : 게스트운영체제에 접근하기 위해 로그인 함수 호출

⑦ : 선택된 리눅스 운영체제 점검 모듈을 게스트운영체제에 복사

⑧ : 복사된 리눅스 운영체제 점검 모듈을 실행

⑨ : 점검이 완료된 후 결과 파일을 호스트 운영체제로 복사

⑩ : 호스트 운영체제에 복사된 결과 파일 요청

⑪ : 결과 파일이 존재하는지 확인한 후 결과를 파싱

⑫ : 점검모듈 관리 모듈에 반환

⑬ : 점검요청명령분기모듈로 반환

⑭ : 전송을 위해 통신 모듈로 반환

⑮ : 점검 결과를 매니저로 전송

윈도우즈 운영체제나 웹서버 점검 등 추가적인 점검 요청은 점검 요청에 따른 점검 모듈만 변경되며 상기 시나리오와 동일하게 실행된다.

4.2 취약점 점검 파일럿 시스템 구현

취약점 점검 파일럿 시스템은 4.1 절에 기술되어 있는 리눅스 운영체제의 취약점 점검 시나리오에 대한 개발 가능성을 증명하기 위해 구축 및 개발하였다.

Fig. 5는 리눅스 운영체제 취약점 점검을 위해 구축된 파일럿 시스템을 보여준다. 호스트 운영체제로 윈도우즈 7이 설치된 PC에 취약점 점검을 위한 가상화 환경을 구축하기 위해 VMWare社의 가상화 환경 에뮬레이션 프로그램인 VMware workstation9.0을 설치하였으며 점검 대상 서버로는 레드햇 엔터프라이즈 리눅스 6을 가상화로 구축하였다.

Fig 5의 파일럿 시스템은 호스트 운영체제에 저장된 리눅스용 점검 모듈을 가상화 관리 API를 이용하여 가상화 서

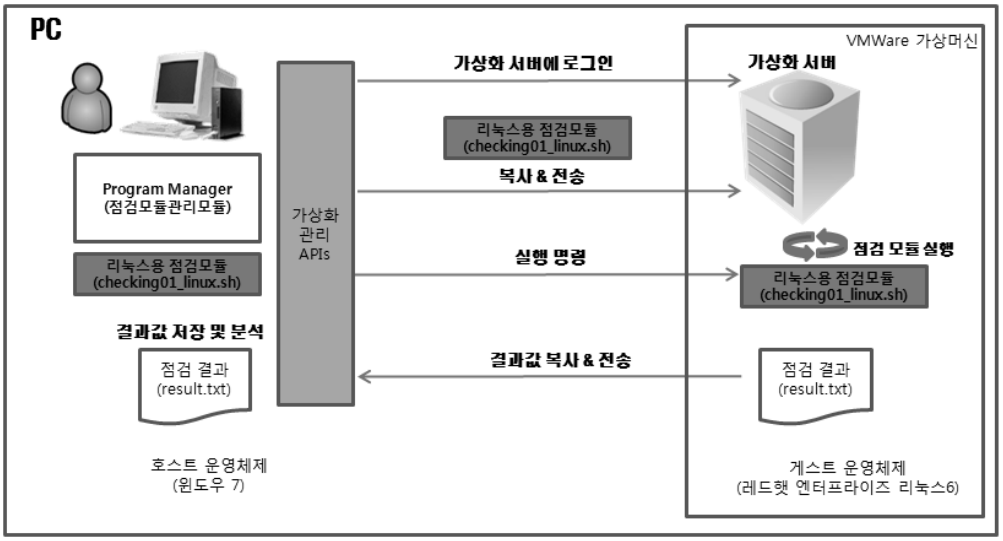

Fig. 5. Pilot system for checking linux vulnerability

Table 4. Major code in Program Manager

```
// 1. 게스트운영체제에 로그인
jobHandle = VixVM_LoginInGuest(vmHandle,
                              "jsyang", // userName
                        "password", // password
                                          0,
                         NULL, // callbackProc
                              NULL); // clientData
   err = VixJob_Wait(jobHandle, VIX_PROPERTY_NONE);
   if (VIX_OK != err) {
       goto abort;}
   Vix_ReleaseHandle(jobHandle);
// 2. 점검 모듈을 게스트운영체제에 복사
jobHandle = VixVM_CopyFileFromHostToGuest(vmHandle,
"D:\\test\\script\\checking01_linux.sh",   // src name
"/home/jsyang/Public/checking01_linux.sh", // dest name
                                          0, // options
VIX_INVALID_HANDLE, // propertyListHandle
                                  NULL, // callbackProc
                                  NULL); // clientData
err = VixJob_Wait(jobHandle, VIX_PROPERTY_NONE);
   if (VIX_OK != err){
   goto abort;}
   Vix_ReleaseHandle(jobHandle);
// 3. 점검 모듈을 게스트운영체제에서 실행
jobHandle = VixVM_RunProgramInGuest(vmHandle,
              "/home/jsyang/Public/",
              "checking01_linux.sh",
                                  0, // options,
         VIX_INVALID_HANDLE, // propertyListHandle,
                                  NULL, // callbackProc,
                                  NULL); // clientData
err = VixJob_Wait(jobHandle, VIX_PROPERTY_NONE);
   if (VIX_OK != err) {
   printf("VIX Error on run program in guest:");
   printf(Vix_GetErrorText(err,NULL));
   goto abort;}
   else {
   printf("Ran program in guest\n");
   }
// 4. 점검 결과를 호스트운영체제로 복사
   jobHandle = VixVM_CopyFileFromGuestToHost(vmHandle,
   "/home/jsyang/Public/result.txt",
   "D:\\test\\results\\result.txt",   // dest name
                                  0, // options
      VIX_INVALID_HANDLE, // propertyListHandle
                         NULL, // callbackProc
                         NULL); // clientData
err = VixJob_Wait(jobHandle, VIX_PROPERTY_NONE);
if (VIX_OK != err) {
    goto abort;
}
```

버인 리눅스 운영체제에 점검 모듈을 복사 및 전송한 후 실행 명령을 수행하고 실행된 점검 모듈의 결과 파일을 호스트 운영체제로 복사해오는 시나리오를 가진다.

Table 4는 상기 시나리오를 구현한 점검모듈관리모듈의 주요 코드이다. 첫 번째(1. 게스트운영체제 로그인)는 취약점 점검 모듈(checking01_linux.sh) 복사를 위해 가상화 서

버에 인증을 수행한다. 두 번째(2. 점검 모듈을 게스트운영체제에 복사)는 점검 모듈을 복사하여 전송한다. 세 번째(3. 점검 모듈을 게스트운영체제에서 실행)는 전송된 리눅스 운영체제 취약점 점검 모듈을 실행한다. 점검이 종료되면 점검 결과가 파일로 저장된다. 네 번째(4. 점검 결과를 호스트 운영체제로 복사)는 취약점 점검 결과를 호스트 운영체제로 복사하여 수신한 후 취약점 점검 절차가 종료된다.

본 파일럿 시스템을 실행하여 본 결과 원격으로 리눅스 취약점 점검 모듈을 복사 및 실행하고 이에 대한 결과 파일을 수신하여 점검 결과를 정상적으로 원격에서 수행할 수 있는 것을 확인하였다.

기존의 에이전트는 사용자가 직접 서버에 설치하고 데몬으로 실행되는 방식인데 반해 제안한 에이전트는 점검 모듈을 원격으로 복사하여 실행하고 종료하기 때문에 시스템 확장성 및 자원 점유 측면에서 장점이 있는 것을 알 수 있었다.

향후 구현된 파일럿 시스템을 기반으로 점검 모듈을 추가하고 다수의 서버를 대상으로 점검을 수행하기 위한 통신 모듈 및 정책을 추가하여 개발할 예정이다.

5. 결 론

본 논문에서 제시한 가상화 환경에서 호스트 기반 취약점 분석 시스템의 에이전트 프레임워크는 기존의 취약점 분석 시스템의 단점을 분석하였으며 가상화 관리 API를 이용하여 기존의 단점을 개선한 에이전트를 제안하였다.

본 논문에서 제안한 에이전트 프레임워크는 점검 대상 시스템 확장 시 에이전트의 추가 설치가 없어 확장성 및 관리 편의성을 증가시킨다. 또한 데몬 형태의 프로그램이 아니기 때문에 메모리 및 하드디스크의 시스템 자원에 대한 비효율성을 개선하였다.

본 논문에서 제안한 에이전트 프레임워크는 국내 특허 등록을 완료하였으며 취약점 분석 시스템 솔루션 업체가 상용화를 목표로 개발 중에 있다.

참 고 문 헌

[1] wikipedia, 가상화의 정의, http://ko.wikipedia.org/wiki/가상화

[2] Jae Seung Lee and Sang Choon Kim, “Design of the Security Evaluation System for Decision Support in the Enterprise Network Security Management”, Journal of KIISE(Korean Institutes of Information Scientists and Engineers), Vol. 30, No.6, pp.776-786, Dec., 2003.

[3] KISA, “Guide for Selecting system Vulnerability Assessment Tools”, white paper, Sep., 2002.

[4] Sung-Kyong Un, “Trend of Cloud Computing Security Technology”, Review of KIISC(Korea Institute of Information Security and Cryptology) Vol. 20, No. 2, pp.27-31. Apr., 2010.

[5] CSA, “Security Guidance for Critical Areas of Focus in Cloud Computing v3.0”, white paper, 2011.

[6] KISA, “Security Guidebook for Cloud Service”, white paper, Oct., 2011.

[7] Tae-Hyoung Kim, et al.,“Trend of Cloud Computing Security Technology”, Review of KIISE, pp.30-38, Jan., 2012.

[8] Karen Scarfone, Murugiah Souppaya and Paul Hoffman, “Guide to Security for Full Virtualization Technologies”, NIST SP800-125, Jan., 2010.

[9] Citrix, “Citrix XenServer Management API”, Sep., 2012.

[10] Libvirt API reference, http://www.libvirt.org/html/libvirt-libvirt.html

[11] VIX API Reference, http://www.vmware.com/support/developer/vix-api/vix112_reference/index2.html

[12] Jun Yoon and Wontae Sim, “An Automatic Network Vulnerability Analysis System using Multiple Vulnerability Scanner”, Journal of KIISE Vol. 14 No. 2, Apr., 2008.

[13] Ji-Hong Kim and Whi-Kang Kim, “Automated Attack Path Enumeration Method based on System Vulnerabilities Analysis”, KIISC Vol. 22, No. 5, pp.1079-1090, Oct., 2012.

[14] NileSoft, “보안취약점 분석도구 선택 기준”, http://www.nilesoft.co.kr/sub/vulnerability/vulnerability_02.html

[15] SECUI, “SECUI SCAN V2.0”, SECUI SCAN product brochure.

[16] NileSoft, “Secu Guard SSE, system vulnerability assessment scanner”, Secu Guard SSE product brochure.

[17] KISA, “정보보호관리체계 통제사항 가이드”, white paper, 12월 2004년.

양 진 석

e-mail : jsyang@ensec.re.kr

2003년 성균관대학교 정보공학과(학사)

2005년 성균관대학교 컴퓨터공학과(석사)

2011년 성균관대학교 컴퓨터공학과(박사수료)

2005년~현 재 한국전자통신연구원 부설연구소 선임연구원

관심분야 : 클라우드 컴퓨팅 보안, 네트워크 보안

정 태 명

e-mail : tmchung@ece.skku.ac.kr

1981년 연세대학교 전기공학과(학사)

1984년 일리노이주립대학 전자계산학과 (학사)

1987년 일리노이주립대학 컴퓨터공학과 (석사)

1995년 퍼듀대학교 컴퓨터공학과(박사)

1995년~현 재 성균관대학교 컴퓨터공학과 교수

관심분야 : 통합보안관리, 네트워크, 무선망