情報保護學會論文誌
第12券 第2號, 2002. 4

취약성 점검 코드를 자동으로 생성하는 에이전트를 통한 통합 취약성 분석 시스템*

김 수 용**, 서 정 석***, 조 상 현***, 김 한 성***, 차 성 덕****

Integrated Security Manager with AgEnt-based vuLnerability scanner automatically generating vulnerability analysis code(ISMAEL)

Su Yong Kim**, Jeong Seok Seo***, Sang Hyun Cho***, Han Sung Kim***, Sung Deok Cha****

요 약

악의의 침입자는 일반적으로 시스템의 취약성을 분석하고 발견된 취약성을 이용하여 시스템에 침입한다. 그러므로 보안 관리자는 악의의 침입자로부터 시스템을 보호하기 위해서 주기적으로 시스템의 취약성들을 분석하여 가능한 빠른 시간 내에 해당 취약성들을 제거해 주어야 한다. 기존의 시스템 취약성 분석 방법은 주로 네트웍 기반 취약성 분석 도구에 의존해 왔다. 하지만, 네트웍 기반 취약성 분석 도구는 대상 시스템과의 네트웍 통신을 통해 얻을 수 있는 제한된 정보만을 이용하여 취약성을 점검하기 때문에 네트웍 통신을 통해 접근할 수 없는 시스템 내부 취약성에 대한 검사는 불가능하다. 시스템의 모든 취약성을 정확하게 점검하기 위해서는 호스트 기반 취약성 분석 도구를 사용해야 한다. 하지만, 호스트 기반 취약성 분석 도구는 시스템의 운영체제의 종류와 버전에 따라 다르게 구현해야 하기 때문에 많은 호스트들을 관리해야하는 보안 관리자가 호스트 기반 취약성 분석 도구를 사용하는 것은 불가능하다. 본 논문에서는 에이전트 기반 취약성 분석 도구인 ISMAEL을 제안하여 프로토타입을 구현하였다. ISMAEL에서 매니저는 대상 시스템에 설치된 에이전트에게 취약성 여부를 판단할 수 있는 정보를 제공하고, 에이전트는 그 정보를 근거로 자신에게 맞는 취약성 분석 코드를 자동으로 생성하여 취약성을 점검한다. 해당 취약성의 점검 결과는 다시 매니저에 XML 형태로 전해진다. 이와 같은 구조를 통해 여러 운영체제로 에이전트를 포팅하는 것이 쉬워질 뿐만 아니라 취약성 점검 항목들이 추가되더라도 매니저의 수정이나 에이전트의 수정이 거의 필요 없게 된다. 우리는 ISMAEL의 프로토타입을 구현하고 실제로 적용해 봄으로써 이것이 효율적임을 알 수 있었다.

ABSTRACT

Malicious attackers generally attempt to intrude the target systems by taking advantage of existing system vulnerabilities and executing readily available code designed to exploit known vulnerabilities. To the network security administrators, the first and minimal step in providing adequate network security is to identify existing system vulnerabilities and patch them as soon as possible. Network-based vulnerability analysis scanners (NVAS), although widely used by network security engineers, have shortcomings in that they depend on limited information that is available and generally do not have access to host-specific information. Host-based vulnerability analysis scanner (HVAS) can serve as an effective complement to NVAS. However, implementations of HVAS differ from one platform to another and from one version to another. Therefore, to security engineers who often have to maintain a large number of heterogeneous network of hosts, it is impractical to develop and manage a large number of HVAS. In this paper, we propose an agent-based architecture named ISMAEL and describe its prototype implementation. Manager process provides various agent processes with description on vulnerabilities to check, and an agent process automatically generates, compiles, and executes an Java code to determine if the target system is vulnerable or not. The result is sent back to the manager process, and data exchange occurs in XML format. Such architecture provides maximal portability when managing a group of heterogeneous hosts and vulnerability database needs to be kept current because the manager process need not be modified, and much of agent process remains unchanged. We have applied the prototype implementation of ISMAEL and found it to be effective.

keyword : *호스트 기반 취약성 분석 도구, ISMAEL, 에이전트, 보안*

---

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았습니다.
** 국가보안기술연구소(sweetlie@etri.re.kr)
*** 한국과학기술원 소프트웨어공학 연구실
**** 한국과학기술원 교수

Ⅰ. 서 론

1.1 연구 배경

인터넷의 이용이 급증하고 있다. 이런 상황 속에서 Code Red II[1]나 NIMDA[2]와 같은 웜들은 가공할 만한 전염력을 보여주었다. 이런 웜들은 취약한 시스템을 침입할 뿐만 아니라, 침입한 시스템을 이용해 무차별적으로 다른 시스템들을 공격하기 때문에 웜이 전파되는 속도와 규모는 상상을 초월했다. 하지만, 이런 웜들이 이용하는 취약성들은 이미 오래 전부터 널리 알려진 취약성들이기 때문에, 취약성에 대한 패치를 설치하는 것만으로도 이런 웜들의 공격을 무력화시킬 수 있었다. 하지만, 관리자의 부주의로 많은 시스템들이 패치가 되지 않았기 때문에 많은 시스템들의 경우 웜에 감염되었다.

이와 같이 많은 공격들이 이미 알려진 취약성들을 이용하며, 관리자의 부주의로 취약성에 대한 보안 패치가 설치되지 않은 시스템들을 대상으로 삼는다. 따라서, 시스템의 취약성들을 분석하여 해당 취약성들을 제거해주는 것은 대상 시스템의 보안을 위한 최소한의 노력이며, 적은 노력으로 높은 보안 수준을 유지할 수 있다.

취약성 분석은 시스템으로부터 수집되는 정보를 근거로 시스템의 보안 상태를 측정하는 행위를 말한다. 시스템이 일단 침입을 당하게 되면, 사후 조사가 힘들뿐만 아니라 완전한 복구가 불가능한 경우가 많기 때문에 침입이 일어나기 전에 침입에 이용될 수 있는 시스템의 취약성들을 제거해 주는 것이 중요하다.

1.2 취약성 분석 도구

취약성 분석을 자동화해 주는 취약성 분석 도구는 호스트 기반 취약성 분석 도구와 네트워 기반 취약성 분석 도구로 구분된다.

1.2.1 네트워 기반 취약성 분석 도구

네트워 기반 취약성 분석 기법은 네트워 통신을 통해 대상 호스트로부터 얻을 수 있는 정보들을 이용하여 호스트에 특정 취약성이 존재하는지를 점검한다. 네트워 기반 취약성 분석 도구는 표준화가 잘 되어 있는 네트워 통신을 통해 시스템의 취약성을 분석하기 때문에 취약성을 분석하고자 하는 대상 시스템의 운영체제에 독립적이고, 개발이 쉬우며, 여러 시스템들의 취약성을 동시에 분석할 수 있다는 장점이 있다. 대표적인 네트워 기반 취약성 분석 도구로는 SATAN[3], Nessus[4]등이 있다. 하지만, 네트워 기반 취약성 분석 도구들은 네트워 통신을 통해서 얻을 수 있는 한정된 정보만을 이용하여 취약성 분석을 하기 때문에 많은 한계점을 지니고 있다.

첫째, 네트워 기반 취약성 도구는 내부 사용자에 의해서 악용될 수 있는 시스템의 내부 취약성들을 점검할 수 없다. 가령, 솔라리스의 특정 버전에 존재하는 ufsrestore 버퍼오퍼플로우 취약성[5]과 같이 내부 사용자만이 접근할 수 있는 응용프로그램에 대한 취약성은 점검할 수 없다.

둘째, 네트워 기반 취약성 도구는 모의 공격을 시도해 보고, 그 결과로부터 취약성 여부를 판단하기 때문에 대상 시스템의 정상적인 서비스에 방해를 줄 수 있다. 예를 들면, 버퍼오버플로우에 취약한 데몬이 발견되었다면, 네트워 기반 취약성 분석 도구는 취약한 데몬에 버퍼오버플로우를 일으킬 수 있는 문자열을 보내고, 데몬으로부터의 반응에 따라 데몬의 취약성 여부를 결정한다. 이런 기법은 특정 데몬의 취약성 여부를 판단할 수는 있지만, 취약한 데몬의 정상적인 활동을 방해하게 된다. 특히 최근 이슈가 되고 있는 서비스 거부 공격(Denial-Of-Service)과 같은 취약성을 분석하기 위해서도 실제 서비스 거부 공격 공격을 시도해 보고 그 결과를 통해 취약성 여부를 판단해야 하기 때문에 대상 시스템의 정상적인 활동을 보장할 수 없다.

셋째, 네트워 기반의 취약성 분석 도구에 의한 취약성 분석 결과는 부정확하거나 신뢰성이 떨어질 수 있다. 가령 호스트 기반 취약성 분석 도구라면 시스템의 운영체제의 종류와 버전을 알기 위해서 여러 가지 시스템 콜이나 설정 파일들을 이용해서 정확하게 알아낼 수 있지만, 네트워 기반 취약성 분석 도구에서는 여러 가지 배너나 TCP/IP 스택의 구현을 테스트해 봄으로써 대상 시스템의 운영체제를 "추측"할 수 있을 뿐이다.[6] 그래서 정확도(Accuracy) 나 신뢰도(Reliability)가 호스트 기반 취약성 분석 도구에 비해서 떨어질 수밖에 없다.

마지막으로, 네트워 기반 취약성 분석 도구에 의해 취약성 분석을 할 경우 일반적으로 호스트 기반 취약성 분석 도구보다 더 많은 자원이 필요하다. 가령 네트워 기반 취약성 분석 도구를 통해 서비스 거부 공격에 취약한지 검사하기 위해서 모의 공격을

시도하고자 할 경우에는 시스템의 많은 자원이 필요하며, 심지어는 여러 시스템의 도움이 필요할 수도 있다.

1.2.2 호스트 기반 취약성 분석 도구

호스트 기반 취약성 분석 기법은 시스템의 합법적인 사용자가 시스템에서 얻을 수 있는 정보들을 이용하여 시스템에 존재하는 취약성들을 분석한다. 호스트 기반 취약성 분석 기법의 연구는 크게 두 가지로 나눌 수 있다. 하나는 시스템을 운영하는 과정에서 발생할 수 있는 운영상의 문제점들을 점검하는 연구이고, 다른 하나는 시스템에 존재하는 응용 프로그램들이나 운영체제 자체의 취약성들을 분석하는 연구이다. 지금까지 개발되어 온 호스트 기반 취약성 분석 도구들은 주로 운영상의 문제점을 분석하는 것에 초점을 맞추고 있으며, Kuang[7], COPS[8]를 들 수 있다. 응용 프로그램들의 취약성을 분석하기 위한 도구의 개발이 미흡한 이유는 취약성 분석이 각 응용 프로그램 특성에 많이 의존하기 때문에 구현하기가 어렵기 때문이다. 하지만 점차 시스템 관리자들의 보안 지식 수준이 높아짐에 따라 운영상의 문제점들보다는 시스템에 존재하는 취약한 응용프로그램에 의한 침입이 늘어가고 있기 때문에 응용 프로그램의 취약성을 점검하는 일은 중요하다.

호스트 기반 취약성 분석 도구들의 장점은 취약성을 분석하고자 하는 시스템의 모든 정보를 이용하기 때문에 적은 자원만을 이용하여 높은 정확성과 신뢰도가 보장되는 결과를 얻을 수 있다는 점이다. 또한 어떤 데몬의 특정 버전에 버퍼오버플로우 취약성이 존재한다는 사실이 알려지면, 데몬의 취약성을 조사하기 위해서 해당 데몬에 버퍼오버플로우를 일으킬 필요 없이 데몬이 제공해주는 명령어나 설정 파일들을 이용해 데몬의 버전을 검사하는 것만으로 데몬의 취약성을 점검할 수 있다. 이와 같은 안전한 방법은 시스템이나 데몬의 정상적인 서비스에 전혀 지장을 주지 않는다. 호스트 기반 취약성 분석 도구는 네트워 기반 취약성 분석 도구에서 불가능했던 시스템 내부의 취약성들- 잘못된 환경 설정, 시스템 내부에서만 접근 가능한 취약한 프로그램 등 - 을 점검할 수 있다는 장점이 있다. 하지만, 호스트 기반 취약성 분석 도구는 분석하고자 하는 시스템에 설치되어야 하기 때문에 다음과 같은 문제점들이 존재한다.

첫째, 시스템 내부의 취약성을 점검하는 방법이 운영체제에 따라 다르고, 같은 운영체제에서도 버전에 따라 다를 수 있으며, 심지어 동일한 버전의 운영체제에서조차도 설치된 응용 프로그램에 따라 취약성을 분석하는 방법이 다르다. 여러 요인에 의해 취약성 분석 방법이 달라지기 때문에 취약성을 분석하기 위한 코드도 다르게 된다. 따라서 각 운영 체제의 종류와 버전에 따라 새로운 취약성 분석 도구들을 구현해야 하는 문제점이 있다.

둘째, 호스트 기반 취약성 분석 도구는 다수의 시스템들을 효율적으로 관리할 수 없다. 초기의 취약성 분석 도구들이 호스트 기반 취약성 분석 기법을 사용했지만, 한 조직에서 관리해야 하는 시스템의 수가 급격히 증가함에 따라 다수의 시스템들의 취약성들을 효율적으로 관리하기 위해서 네트워 기반 취약성 분석 도구가 더 선호되고 있다.

셋째, 호스트 기반 취약성 분석 도구들은 새로운 취약성이 등장하여 취약성 분석 기능을 추가하고자 할 경우 호스트 기반 취약성 분석 도구의 소스 코드를 수정하고 다시 컴파일해야 한다. 하지만 대규모의 시스템에 호스트 기반 취약성 분석 도구들이 설치되어 있을 경우 모든 시스템들에 설치되어 있는 도구들을 일일이 수작업으로 갱신해야 하는 일은 대단히 어려울 수밖에 없다.

1.3 새로운 취약성 분석 시스템 제안 - ISMAEL

네트워 기반 취약성 분석 도구와 호스트 기반 취약성 분석 도구들이 가진 단점을 극복하기 위해서 본 논문에서는 새로운 취약성 분석 시스템을 제안한다. 본 논문에서 제시하는 ISMAEL 시스템은 호스트 기반 취약성 분석 도구[1]를 대상 시스템의 에이전트로 설치하여 취약성을 점검하도록 하고 매니저를 통해 여러 에이전트들을 관리함으로써 호스트 기반 취약성 분석 도구의 단점을 극복하면서 호스트 내의 모든 정보를 이용하여 취약성을 분석한다.

호스트 기반 취약성 분석 에이전트를 모든 운영체제에서 구현해야 하는 문제는 자바를 이용하여 운영체제에 독립적인 모듈로 에이전트를 구현하고, 운영체제에 의존하는 부분들은 라이브러리(.class) 파일

1) ISMAEL에서는 에이전트를 대상 시스템에 설치하기 때문에 호스트 기반 취약성 분석 도구라고 본 논문에서 말하지만, 취약성을 점검할 때, 네트워 통신을 이용할 때 더 높은 신뢰를 얻을 수 있다면 로컬 호스트와의 네트워 통신을 통해 취약성 검사를 수행하기도 한다. 따라서, Shostack와 Black에 의한 취약성 분석의 분류 기준에 따르면 ISMAEL에서 구현된 에이전트는 인가 받은 취약성 분석(credentialed assessment)에 해당한다.[9]

형태로 제공해 줌으로써 다른 운영체제로의 포팅을 용이하게 했다.

이외에도 ISMAEL에서는 에이전트와 매니저간의 통신을 XML을 이용함으로써 차후 다른 보안 제품들과의 연동을 가능하게 하였다.

1.4 논문의 구성

Ⅱ장에서는 기존의 취약성 분석 도구들을 중심으로 관련 연구를 기술하고, Ⅲ장에서는 본 논문에서 제안하는 ISMAEL의 구조와 구현에 대해 기술한다. Ⅳ장에서는 ISMAEL을 이용해서 여러 시스템의 취약성을 분석해 본 실험결과를 통해 ISMAEL의 효과성을 살펴본다. Ⅴ장에서는 본 고에서 제시하는 ISMAEL의 장점과 전망에 대해 언급한다.

Ⅱ. 관련연구

2.1 취약성 점검기술 및 침입시도 탐지기술 개발[10]

한국정보보호진흥원이 주관하고 시큐아이닷컴㈜과 인터넷시큐리티㈜가 공동 연구하여 2000년 12월에 발표한 '취약성 점검기술 및 침입시도 탐지기술 개발'에 관한 1차년도 연구개발 보고서는 취약성 점검과 침입 탐지를 모두 수행할 수 있는 모델을 제시했다. 가장 큰 특징은 네트워 기반 취약성 분석 기법과 호스트 기반 취약성 분석 기법을 통합하는 형태를 취했다는 점이다.

보고서에서 제안하는 모델은 분석 대상이 되는 호스트에 에이전트 시스템을 설치하고, 관리자는 취약성 스캔 관리 시스템을 통해 여러 에이전트 시스템을 통합 관리하는 형태를 갖는다. 에이전트 시스템은 네트워 취약성 점검 모듈과 시스템 취약성 점검 모듈, 그리고 소스코드 보안 취약성 점검 모듈로 이루어진다.

보고서에서는 ISMAEL과 유사하게 호스트 기반 취약성 분석 도구를 에이전트로 시스템에 설치하고 매니저를 통해 관리하는 구조를 사용하고 있지만, 호스트 기반 취약성 분석 도구의 구현에 가장 큰 문제점인 다양한 플랫폼의 지원을 해결하기 위한 방법들이 제안되어 있지 않다.

2.2 COPS(Computerized Oracle and Password System)[8]

COPS는 여러 작은 프로그램들로 구성되어 있고, 각각의 작은 프로그램들이 유닉스 시스템에서 SUID 프로그램 점검, 부적절한 파일 퍼미션 문제, 잘못된 패스워드 등과 같은 특정 취약성 검사를 수행한다.

현재 ISMAEL의 구현은 주로 특정 응용프로그램의 취약성을 분석하는데 초점을 맞추고 있어서 COPS에서 점검하고 있는 부분들을 지원하지 않고 있다. COPS는 동작할 수 있는 운영체제가 유닉스에 제한되는데 반해, ISMAEL은 윈도우나 기타 자바 버추얼 머신이 존재하는 모든 운영체제에서 동작 가능하다는 장점이 있다.

2.3 SHE[2] : 유닉스 시스템을 위한 통합 보안 점검 도구[11]

SHE는 1997년 KAIST 소프트웨어 공학 연구실과 한국통신 멀티미디어 연구소 네트워 보안 연구팀이 공동 연구하여 발표한 통합 보안 점검 도구이다. 기존에 개발된 공개 보안 도구들이 일관성 있는 사용자 인터페이스를 제공하지 않고 시스템의 특정 부분에 대한 점검 기능만을 제공하기 때문에 SHE는 여러 가지 공개 보안 도구들을 통합하여 시스템의 전반적인 보안 관리 기능을 제공하고, 사용자에게 편리한 인터페이스를 제공한다.

SHE는 시스템 보안을 계정 보안, 시스템 보안, 네트워 보안, 파일 변경 검사로 나누어 점검을 수행한다. 계정 보안에서는 COPS를 이용하여 패스워드 파일 검사와 그룹 파일 검사를, Crack을 이용하여 패스워드 크래킹을 시도한다. 시스템 보안에서는 COPS를 이용하여 사용자 검사, 자동 수행 파일 검사, 장치 파일 검사를 수행하며, 네트워 보안에서는 COPS와 SATAN을 이용하여 주요 인터넷 서비스들을 점검한다. Tripwire를 통해 파일 변경 검사에서 파일의 생성, 변경, 삭제를 감시한다. SHE는 이런 네 가지 종류의 검사들을 주기적으로 수행하도록 함으로써 시스템 보안을 일정하게 유지할 수 있도록 한다.

SHE는 여러 공개 소프트웨어들의 기능을 통합하여 사용자에게 편리한 인터페이스를 제공하기 위한 시도였던 반면, ISMAEL은 여러 공개 보안 도구들의 기능을 확장하고, 다양한 운영체제에서 동작 가능하도록 하기 위한 연구이다.

2) "지킴이"라는 이름으로 만들어진 취약성 분석 도구를 한국통신 멀티미디어 연구소에서 제품화하면서 도구명이 "SHE"로 변경되었습니다.

2.4 SATAN(Security Administrator's Tool for Analyzing Networks)[3,12]

SATAN은 "Testing by Exploit" 이라는 기법을 이용하여 대상 시스템의 취약성을 분석한다. "Testing by Exploit"이란 일반적으로 실제 공격에서 리턴되는 루트 셸 대신 상태값(status flag)을 리턴하도록 만든 실험 스크립트를 이용하여 취약성을 점검하고자 하는 대상 시스템에 모의 공격을 수행하고, 상태값(status flag)을 근거로 취약성 여부를 판단하는 기법을 말한다.

하지만, 분산 도스 공격과 같이 실제 공격을 모의실험하기가 곤란한 경우가 많기 때문에 추론 방법(Inference Method)을 병행 사용한다. 주로 사용되는 추론 방법은 운영체제나 응용프로그램의 버전 점검, 포트의 상태 점검, 각 포트에 대한 프로토콜 점검이다. 운영체제나 응용프로그램의 버전 점검은 특정 버전에 취약성이 존재할 경우, 취약성의 존재 여부를 추측하기 위해 서버나 운영체제에서 제공되는 버전 정보나 그 외 특정 버전에 존재하는 특징들을 통해 운영체제나 응용프로그램의 버전을 추측하고 그 버전 정보를 통해 취약성 여부를 판단하는 기술이다. 포트 상태 점검 기술은 특정 포트로 서비스되는 서버의 취약성이 알려져 있을 경우, 해당 포트가 열려있는지 닫혀 있는지를 통해 해당 시스템의 취약성 여부를 판단하는 기법이다. 프로토콜 점검은 포트 상태 점검과 더불어 신뢰성을 높이기 위해 사용되거나 흔히 사용되는 포트 번호 이외의 번호를 이용하여 서비스되는 데몬들의 서비스 내용을 알기 위해 사용된다. 이 방법은 열려 있는 포트에 추측되는 서비스의 프로토콜에 맞는 패킷을 보내고 이에 대한 응답이 추측되는 서비스의 프로토콜과 일치하는지를 점검한다.

추론 방법은 "Testing by Exploit" 보다 공격적이지 않다는 장점을 가지고 있지만, 대상 시스템의 가짜 배너에 속기 쉽고, 한정된 정보를 가지고 다시 추측을 하는 것이기 때문에 "Testing by Exploit" 보다 정확성이나 신뢰도는 떨어지는 편이다.[9]

SATAN은 네트워 통신에 근거하여 취약성을 분석하기 때문에 네트워 통신을 통해 접근 불가능한 시스템 내부의 취약성은 점검할 수 없다는 한계를 가진다. 이런 한계를 극복하기 위해서 ISMAEL과 같이 호스트 기반 취약성 분석 도구를 사용하여야 한다.

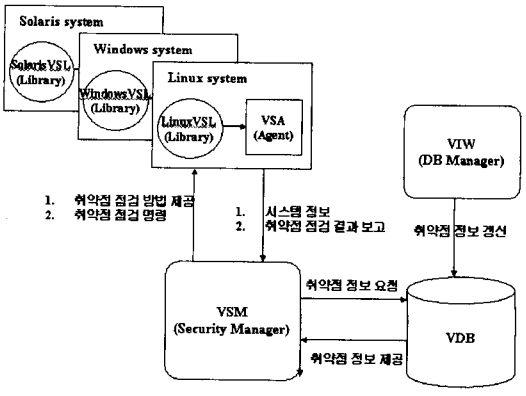
Ⅲ. ISMAEL

3.1 Overview

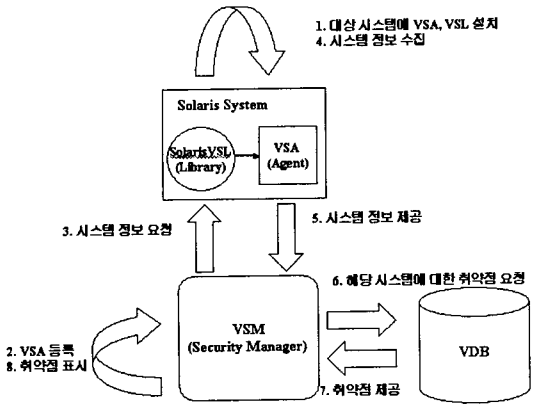
ISMAEL의 전체적인 구조는 [그림 1]과 같다. ISMAEL은 여러 운영체제에서 동작 가능한 호스트 기반 취약성 분석 에이전트와 매니저를 통해 시스템의 취약성들을 관리하는 통합 취약성 분석 시스템이다. ISMAEL은 Vulnerability Scanner Manager (VSM)와 Vulnerability Scanner Agent(VSA), Vulnerability Scanner Library(VSL), Vulnerability Information Writer(VIW) 그리고 Vulnerability DataBase(VDB)로 구성된다.

관리자가 시스템들을 ISMAEL을 이용하여 시스템들을 관리하고자 한다면, [그림 2]에서처럼 운영체제에 독립적인 VSA와 운영체제에 종속적인 VSL을 대상 시스템들에 설치한다. 그리고, VSM에는 VSA가 설치된 시스템의 IP 주소와 포트 번호를 등록한다. VSM은 등록된 VSA로부터 시스템 정보를 받아와서, 시스템의 운영체제에서 점검할 필요가 있는 취약성들을 VDB에 요청한다. VDB는 해당 시스템에서 맞는 취약성들을 VSM에게 넘겨주고, VSM은 이를 나무(Tree) 형태로 관리자에게 보여준다.

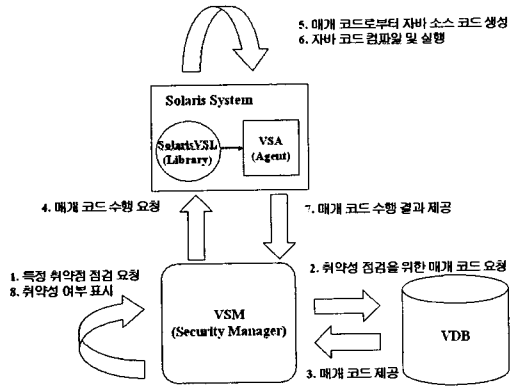
이렇게 등록된 시스템의 취약성을 점검하기 위한 방법은 [그림 3]에서와 같다. 관리자는 취약성들 중 점검하고자 하는 취약성들을 선택하여 검사 수행을 요청하고, VSM은 취약성 점검 방법을 매개 언어(Intermediate Language)로 기술한 매개 코드(Intermediate Code)를 VDB로부터 받아 VSA에게 넘겨주고, VSA는 이 정보를 이용하여 시스템에서 맞는 취약성 점검 코드를 생성한다. 이렇게 생



[그림 1] ISMAEL의 개략도



〔그림 2〕 새로운 대상 시스템의 등록



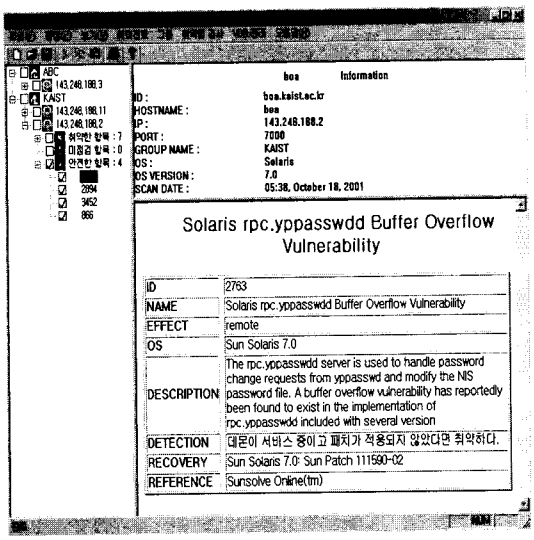
〔그림 3〕 특정 취약성의 취약성 여부 점검

성된 코드는 VSL을 참조하여 컴파일되고, 실행된다. VSA는 취약성 점검 코드의 실행 결과인 취약성 여부를 VSM에 보내준다. VSM은 이 정보를 바탕으로 취약한 항목과 취약하지 않은 항목, 아직 점검하지 않은 항목으로 취약성들을 분류하여 관리자에게 보여준다. 〔그림 2〕는 매니저 프로그램인 VSM을 보여 준다. 왼쪽 창은 여러 에이전트들과 각 에이전트가 설치된 시스템에서 점검해야 하는 취약성들을 나무(Tree) 모양으로 보여준다.

VSM에서는 〔그림 4〕에서와 같이 왼 쪽 창에서는 에이전트들과 해당 에이전트들의 취약성 상황을, 오른 쪽 상단 창에서는 선택된 에이전트가 설치된 시스템에 대한 정보를, 오른 쪽 하단 창에서는 선택된 취약성에 대한 설명을 보여준다.

3.2 운영체제에 독립적인 VSA

ISMAEL의 가장 큰 장점은 에이전트가 다양한



〔그림 4〕 Vulnerability Scanner Manager(VSM)

운영체제에서 동작 가능하며, 취약성이 추가되거나 취약성 여부를 판단하기 위해 점검해야 하는 항목들이 추가되더라도 실행 모듈들인 VSA나 VSM이 수정되는 일이 없다는 점이다.

VSA의 역할은 크게 세 가지이다. VSA가 설치된 시스템의 정보를 모으는 일과 VSM과의 통신, 그리고 VSM으로부터 받은 매개코드를 이용해 실행 가능한 자바 소스 코드를 만들어 수행하는 일이다. 시스템의 정보는 자바에서 제공해 주는 API를 통해 운영체제에 독립적으로 수집할 수 있고, VSM과의 통신 역시 표준화가 잘되어 있는 네트워 프로토콜을 이용하기 때문에 운영체제에 독립적으로 구현이 가능하다. 마지막으로 매개코드로부터 실행 가능한 자바 소스 코드를 만드는 일은 그 과정을 운영체제에 독립적으로 만들었기 때문에 VSA 전체가 운영체제에 독립적으로 구현이 가능했다.

3.3 매개 코드의 사용

매개 코드는 특정 취약성에 대한 취약성 여부를 점검하기 위한 방법이 기술되어 있는 코드이다. 이 코드를 매개 코드라 부르는 이유는 이 코드 자체가 특정 운영체제에서 실행 가능한 소스 코드나 실행 코드가 아니고, 여러 운영체제들에서 생성할 소스 코드의 정보를 담고 있는 중간 단계의 코드이기 때문이다.

매개 코드의 일반적인 형태는 다음과 같다.

```
InspectionElement1 and InspectionElement2 and
InspectionElement3 and ...
```

즉 매개 코드는 점검 항목들의 논리곱의 형태를 띈다. 호스트 기반 취약점 분석 도구인 ISMAEL에서 특정 취약성을 점검하기 위해 필요한 점검 항목들은 특정 파일이 존재하는지를 알기 위한 DoesFileExist, 특정 포트가 열려 있는지를 점검하는 IsPortOpen, 특정 패치가 적용되어 있는지를 점검하는 IsNotPatched, 특정 데몬이 서비스되고 있는지를 점검하는 IsDaemonServed, 특정 응용프로그램의 버전이 취약한지를 점검하는 IsApplicationVersionVulnerable 등이 필요했다.

VDB에 취약성 점검 방법을 저장하기 위해서는 여러 가지 방법들을 사용할 수 있다. 가장 쉬운 방법은 VDB에 직접 VSA에서 수행될 수 있는 소스 코드나 실행 코드를 저장해 두고, VSM으로부터 요청이 올 경우 이 파일을 VSM에 줄 수 있다. 하지만 이 방법은 운영체제마다 다른 실행 파일이나 소스 코드를 일일이 관리자가 만들어 주어야 하기 때문에 많은 취약성을 점검해야 하는 취약성 분석 도구에서는 현실적으로 불가능하다. 다른 방법으로는 Nessus와 같이 취약성 점검 방법을 약속된 스크립트 언어로 기술해 주고, 그 스크립트 파일을 해석하여 취약성 점검을 수행하는 방식이 있다.[13] ISMAEL에서는 매개 언어를 이용하여 취약성 점검 방법의 기술을 NASL보다 상위 레벨에서 이루어질 수 있도록 하고, 기술된 매개 코드를 받은 VSA가 스스로 시스템에 적합한 소스 코드를 생성해주는 기법을 사용한다. 이 방법은 NASL보다 더 상위 레벨에서 취약성 점검 방법을 기술함으로써 취약성 점검 방법 기술에 필요한 시간과 어려움을 현저히 감소시킬 수 있다.

3.4 에이전트의 자동 코드 생성과 취약성 점검

특정 시스템의 취약성을 점검하고자 할 때, VSM은 VDB로부터 취약성 여부를 점검할 수 있는 매개 코드를 받아서 VSA에게 넘겨주고, VSA는 이를 이용하여 자바 소스 코드를 자동 생성한다. 솔라리스에서 공격자가 버퍼오버플로우 공격을 통해 루트 권한을 획득할 수 있는 kcms_configure의 취약성의 점검을 예로 들면, 시스템에 kcms_configure의 취약성이 존재하는지를 점검하기 위해서는 kcms_configure 파일이 존재하는 지와 파일이 존재한다면 패치(111400-01[3])가 설치되었는지 점검해야 한다.[5] 이를 위한 매개 코드 는 다음과 같다.

```
DoesFileExist("kcms_configure") and IsNotPatched
("111400-01")
```

매개 코드를 받은 VSA는 논리곱을 제외한 모든 부분을 점검항목들로 인식하고, 논리곱만을 자바의 문법에 맞도록 '&&'으로 수정해 준 뒤 if 문에 추가함으로써 각 항목들을 모두 만족하는지를 조사한다. 해당 자바 소스 코드는 다음과 같다.

```
public class VIDxxxx
{
    public static void main(String[] args)
    {
        VSL vsl = new VSL();
        if(vsl.IsNotPatched( "111400-01")
         && vsl.DoesFileExist( "kcms_configure "))
            return VULNERABLE;
        else
            return NOT_VULNERABLE;
    }
}
```

VSL 클래스를 생성해서, VSL에서 제공해 주는 IsNotPatched와 DoesFileExist 함수를 사용한다. 코드의 의미는 kcms_configure의 취약성 패치가 이루어지지 않았고, kcms_configure 파일이 존재하면 시스템이 kcms_configure 취약성에 취약한 것으로 판단한다.

위의 코드 생성을 주의해서 살펴보면, VDB로부터 받은 매개 코드와 실제로 생성된 자바 소스 코드의 함수 이름이 같음을 알 수 있다. 이는 새로운 점검 함수가 추가되더라도 실행 모듈인 VSA을 수정할 필요 없이 VSL만을 수정하기 위해서 이다. 이런 구조는 VSL이 실행 모듈이 아니기 때문에 실행모듈인 VSA가 VSL의 갱신을 자동화해 줄 수 있다는 장점을 가진다.

3.5 VSL의 운영체8제 종속성

운영체제에서 응용 프로그램의 취약성 여부를 판

3) 솔라리스 사에서 kcms_configure의 취약성을 제거하기 위해 제공한 패치 프로그램의 패치 번호

단하기 위해서는 일반적으로 응용 프로그램의 버전을 알아야 한다는 것은 각 운영체제에서 공통적인 부분이다. 하지만, 특정 응용 프로그램의 버전을 알기 위해서는 각 운영체제마다 그리고 각 응용 프로그램마다 그 방법이 다르기 때문에 응용 프로그램의 버전을 확인하기 위한 세부적인 구현 방법은 운영체제나 응용 프로그램에 종속적일 수밖에 없다. 예를 들면, 응용 프로그램의 특정 버전이 취약한 것으로 알려질 경우 응용 프로그램이 취약한 버전인지를 점검하기 위해 모든 운영체제에서 공통적으로 호출하는 함수는 다음과 같이 표현 가능하다.

IsApplication Version Vulnerable(applicationName, vulnerable Version)

하지만, 리눅스와 솔라리스에서 특정 프로그램 man의 버전을 알기 위한 점검 방법은 서로 다르다. 리눅스에서는 man의 버전을 알기 위해서 /etc/man.config 파일을 통해 man 의 버전을 알 수 있고, 솔라리스에서는 /usr/local/man/sman1/man.1 파일을 통해서 알 수 있다. 하지만, 두 파일에서 버전 정보를 추출하기 위해서는 다르게 파싱 해야 한다. 그렇기 때문에 위의 표현된 함수의 세부 구현은 두 운영체제에서 달라야 한다. 따라서, VSA에서 생성하는 자바 소스코드에서는 위 함수를 호출하도록 하고, VSL에서는 위 함수의 세부 구현을 라이브러리 형태로 제공한다. 이렇게 함으로써 에이전트의 실행 모듈은 모든 운영체제에 독립적으로 구현할 수 있고, 각 운영체제마다 종속적일 수밖에 없는 부분인 VSL의 구현만으로 다른 운영체제로의 포팅이 가능하게 된다.

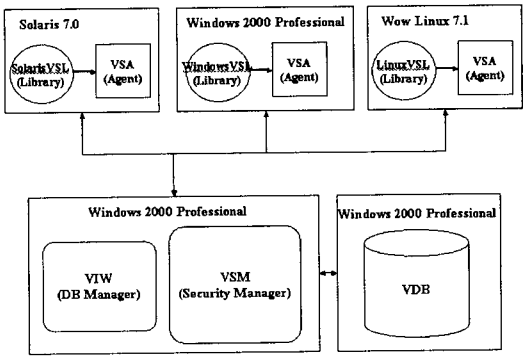
VSL에서 제공하는 모든 함수들이 운영체제에 종속적이지는 않다. 자바 버추얼 머신에 의해서 특정 폴더에 파일이 존재하는지, 특정 포트가 열려 있는지를 점검할 수 있기 때문에 가능하면, VSL에서 제공하는 라이브러리 함수들의 구현도 자바 버추얼 머신에서 제공해주는 기능을 최대한 활용하여 운영체제에 독립적으로 만들고자 했다. 그래서 현재 특정 폴더에 파일이 존재하는지를 검사하는 함수와 특정 포트가 열려 있는지를 점검하는 함수는 운영체제에 독립적이다.

예를 들면, 리눅스나 솔라리스에서는 "find/exam/ -name vulnerable_program"과 같은 명령어를 이용하여 exam이라는 디렉토리에 vulnerable_program이라는 이름을 가진 파일이 존재하는 지를 점검할 수 있다. 하지만 이와 같은 명령어를 사용하여 해당 파일이 존재하는 지를 점검하는 것은 윈도우즈에서는 사용할 수 없기 때문에 운영체제에 종속적이다. 반면에 ISMAEL에서는 자바에서 제공하는 File 클래스의 exists()라는 API를 사용함으로써 자바 버추얼 머신이 존재하는 모든 운영체제에서 동작 가능하면서 파일의 존재 유무를 점검할 수 있다.

VSL은 실행 모듈이 아니기 때문에 VSA에 의해 업데이트가 가능하다. VSA는 매개 코드를 VSM으로부터 받을 때, 매개 코드가 수행 가능한 VSL의 버전을 함께 받는다. VSA는 시스템에 설치되어 있는 VSL의 버전이 매개 코드가 수행 가능한 버전보다 낮으면, VSM에 해당 버전의 VSL을 요청하여 VSL을 업데이트한다.

Ⅳ. 적용사례

ISMAEL을 실험하기 위해서 [그림 5]와 같이 세 가지 다른 운영체제가 설치된 시스템들에 호스트 기반 취약성 분석 에이전트를 설치하였다. 솔라리스 7.0이 설치된 스팍 머신과 윈도우 2000 Professional 버전이 설치되어 있는 PC, 그리고 와우 리눅스 7.1이 설치된 PC에 VSA와 해당 운영체제의 VSL을 각각 설치하고, 윈도우 2000 Professional 버전이 설치되어 있는 PC에 VSM과 VDB를 설치하였다. 솔라리스 7.0이 설치된 스팍 머신과 와우 리눅스 7.1이 설치된 시스템들은 관리자에 의한 보안 패치가 거의 이루어지지 않은 상태였으며, 윈도우 2000 Professional 버전이 설치된 시스템만이 관리자에 의해 계속해서 보안 패치가 이루어지고 있었다.



[그림 5] 실험 구성도

〔표 1〕 취약성 분석 결과

취약성	수행한 운영체제	점검 결과
ufsrestore 버퍼오퍼플로우 취약성[5]	솔라리스	취약
lpd 원격 명령어 수행 취약성[5]	솔라리스	취약
kcms_configure 버퍼오버플로우[5]	솔라리스	안전
in.fingerd 정보 누출 취약성[5]	솔라리스	취약
IN.FTPD의 사용자명 누출 취약성[5]	솔라리스	취약
텔넷 서비스 불허	솔라리스	보안 정책 위반
Man에서 악의의 캐시 파일 생성에 의한 취약성[5]	리눅스	취약
Sendmail 디버깅 기능에서 임의의 코드 실행 취약성[5]	리눅스	취약
Code Red II 감염 확인[5]	윈도우즈	안전
Code Blue 감염 확인[5]	윈도우즈	안전
FTP 서비스 불허	윈도우즈	보안 정책 위반

〔표 1〕은 몇 가지 취약성들을 이용하여, 각각의 취약성을 해당 시스템에서 분석한 결과이다.

실험을 통해 점검한 결과 솔라리스 7.0에서는 텔넷 서비스를 불허하는 보안 정책을 세웠음에도 불구하고 이런 보안 정책이 제대로 지켜지지 않고 있음을 알 수 있었다. 더 큰 문제는 솔라시스 7.0이 설치된 시스템이 시스템 내부 사용자의 정보를 누출시키는 in.fingerd 정보 누출 취약성과 In.FTPD에서 CWD에 의한 Username 누출 취약성에 취약하며, 시스템 내부 사용자가 시스템의 루트 권한을 획득할 수 있는ufsrestore 버퍼오퍼플로우 취약성, 외부 침입자가 시스템의 루트 권한을 획득할 수 있는 lpd 원격 명령어 수행 취약성을 가지고 있다는 점이다. 이런 취약성 분석 결과를 토대로 솔라리스 7.0이 설치된 시스템의 적절한 패치를 설치함으로써 시스템의 보안을 높일 수 있었다.

와우 리눅스 7.1이 설치된 시스템에 대한 취약성 분석을 통해 시스템이 man의 악의의 캐시 파일 생성에 의한 취약성과 Sendmail 디버깅 기능에서 임의의 코드 실행 취약성에 취약하여 악의의 내부 사용자가 루트 권한을 획득할 수 있음을 알 수 있다.

윈도우 2000에 대한 취약성 분석은 최근 유행했던 Code Red II나 Code Blue에 감염되었는지를 점검하였다. 실험 결과는 두 웜 모두에 감염되어 있지 않다는 것을 알 수 있었다. 하지만, 윈도우 2000이 설치된 시스템에도 보안 정책에서 허용하지 않는 FTP 서비스가 제공되고 있음을 알 수 있었다.

특히 〔표 1〕에서 이탤릭체로 쓰여진 취약성들은 내부 사용자에 의해 악용될 수 있는 응용 프로그램들의 취약성들이기 때문에 기존의 취약성 분석 도구들을 사용해서는 탐지할 수 없고, ISMAEL에서만 점검할 수 있다.

또한 특정 운영체제에서 다른 운영체제로 호스트 기반 취약성 분석 에이전트를 포팅하는 것이 대단히 쉽게 가능했다. 솔라리스에서 VSA와 VSL을 구현한 뒤, 윈도우즈 시스템으로 포팅할 때는 약간의 VSA의 구조의 변경이 필요했다. 예를 들어 솔라리스와 리눅스에서는 디렉토리 분리 기호로써 '/'를 사용하는데 반해 윈도우즈에서는 '\'을 사용한다. 일반적으로 문자열에서 '\'는 특수 문자로 인식되기 때문에 이에 대해서 예외 처리를 해 줄 필요가 있었다. 이와 같이 운영체제에 독특한 몇 가지 문제점들을 일반화시키기는 작업이 필요했다. 그러나 이 작업을 마친 뒤, 다시 리눅스로의 포팅은 VSA의 수정이 전혀 필요 없이 해당 VSL 모듈의 추가만으로 포팅이 가능했다. 구체적으로 솔라리스에서는 패치 여부 점검, 파일 존재 여부 점검, 포트 점검의 기능을 가진 VSL이 제공되고 있었고, 이 중 파일 존재 여부 점검과 포트 점검은 자바의 API를 이용하였기 때문에 실제로 다른 운영체제에서 사용하기 위해 수정할 필요는 없었다. 다만 리눅스에서는 응용프로그램 man과 sendmail의 버전을 확인하기 위한 함수의 추가가 필요했다. 이 두 함수의 추가만으로 솔라리스에서 리눅스로 에이전트의 포팅이 가능했다.

이와 같이 운영체제에 독립적으로 작성된 VSA는 어느 정도 일반화 작업을 거친 후에는 다른 운영체제로의 포팅 시 전혀 수정이 필요 없으며, VSL의 함수 추가만으로 충분했다.

V. 결과 및 향후 연구계획

ISMAEL은 호스트 기반 취약성 분석 도구를 에이전트로 대상 시스템에 설치하고 매니저를 통해 여러 에이전트들을 관리하여 시스템들의 취약성들을 분석 관리한다. 또한, 매개 언어로 기술된 취약성 점검 방법을 에이전트에게 제공하고 에이전트는 자신에게 맞는 코드를 자동으로 생성하여 취약성 점검을 수행함으로써 운영체제에 따라 다른 에이전트를 구현해야 하는 부담을 최소화시켰다. ISMAEL의 장점은 다음과 같다.

첫째, 효율적으로 많은 시스템들의 취약성들을 사전에 파악하여 이를 보완함으로써 시스템의 안전성을 높일 수 있다.

둘째, 매니저를 통해 여러 호스트 기반 취약성 분석 에이전트들을 동시에 관리하여 호스트 기반 취약성 분석 도구의 한계를 극복하였다.

셋째, 운영체제에 독립적인 모듈을 에이전트로 개발하여 운영체제에 종속적인 호스트 기반 취약성 분석 도구의 단점을 개선하였다.

넷째, 새로운 취약성 점검 기법이 추가되더라도 에이전트와 매니저의 변경 없이 취약성 데이터 베이스와 새로운 취약성 점검 코드를 라이브러리 모듈인 VSL에 추가해 주면 되기 때문에 새로운 취약성 점검 기법의 갱신이 용이하게 했다.

넷째, 그래픽 유저 인터페이스를 통해 취약성 분석 기법(넓게 보면, 각 사용자에게 특화 된 보안 정책)을 기술할 수 있기 때문에 새로운 취약성이 발견되더라도 여러 보안 사이트에서 제공하는 보안 권고문을 토대로 새로운 취약성 분석 기법을 일반 사용자들도 쉽게 기술할 수 있게 하였다.

ISMAEL이 보안 제품으로써 사용되기 위해서 필요한 일들은 VSM과 VSA 간의 통신 암호화와 VSL 모듈에서 취약성 점검 항목들에 대한 함수들의 추가, 그리고 VSL 모듈의 자동 갱신 기능을 추가하는 것이다. 그 외에도 현재까지 알려진 취약성들을 취약성 데이터 베이스에 계속해서 추가해 주는 작업을 진행하고 있다.

참 고 문 헌

[1] 이현우, 하도윤, 전익수, CERTCC-KR, Oct,2001,http://www.certcc.or.kr/paper/incident_note/2001/in2001_010.html

[2] 전익수, 이완희, CERTCC-KR, Nov, 2001, http://www.certcc.or.kr/paper/incident_note/2001/in2001_015.html

[3] Farmer, D. and W. Venema, "Security Administrator Tool for Analyzing Networks," http://www.fish.com/satan/Renaud

[4] Renaud Deraison, http://www.nessus.org/

[5] SecurityFocus, http://www.security focus.com/

[6] Flodor, "Remote OS detection via TCP/IP Stack FingerPrinting," http://www.insecure.org/nmap/nmap-fingerprinting-article.html

[7] Baldwin, R. W., "Rule-Based Analysis of Computer Security," Massachusetts Institute of Technology, Cambridge, MA, June 1987.

[8] Daniel Farmer, Eugene H. Spafford, "The COPS Security Checker System," Purdue University Technical Report CSD-TR-993, Jan. 1994.

[9] Shostack, A., S. Blake., "Toward a Taxonomy of Network Security Assessment Techniques," Proceedings of the 1999 Black Hat Briefings, July 1999.

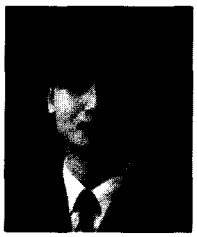
[10] 박정현외 35명, "취약성 점검기술 및 침입시도 탐지기술 개발 제 1차년도 연구개발 보고서," 정보통신부, 2000. 12.

[11] 채홍석, 이남희, 김형호, 김내희, 차성덕, 백석철, 임규건, 박승민, 정종윤, "지킴이 : 유닉스 시스템을 위한 통합 보안 점검 도구," 한국 정보보호 학회 논문지 Vol. 7, No. 3, pp. 23~40, Oct. 1997.

[12] Rebecca Gurley Base, *"Intrusion Detection,"* Macmillan Technical Publishing, pp. 135~154, 2000.

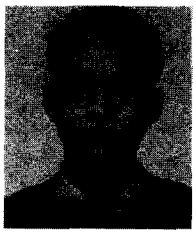
[13] Renaud Deraison, "The Nessus Attack Scripting Language Reference Guide Version 1.0.0pre2," http://www.nessus.org/doc/nasl.html, Apr 2000.

〈著者紹介〉



김 수 용 (Su Yong Kim)
2000년 2월 : 한국과학기술원 전산학과 학사 졸업
2002년 2월 : 한국과학기술원 전산학과 석사 졸업
2002년 3월~현재 : ETRI 부설 국가보안기술연구소 연구원
〈관심분야〉 취약성 분석, 침입탐지시스템

서 정 석 (Jeong Seok Seo)
2001년 2월 : 인하대학교 전자계산공학과 학사
2001년 3월~현재 : 한국과학기술원 전자전산학과 전산학전공 석사과정
〈관심분야〉 소프트웨어 공학, 정보보호



조 상 현 (Sang Hyun Cho)
1997년 2월 : 고려대학교 이과대학 컴퓨터학과 졸업
1999년 2월 : 한국과학기술원 전산학과 석사 졸업
1999년 3월~현재 : 한국과학기술원 전산학과 박사과정
〈관심분야〉 네트워크 보안, 침입 탐지 시스템

김 한 성 (Han Sung Kim)
1990년 3월 : 육군사관학교 전산학과 졸업
1995년 9월 : 웨스턴 온타리오대학(캐나다) 전산학 석사
2001년 3월~현재 : 한국과학기술원 전자전산학과 전산학전공 박사과정
〈관심분야〉 소프트웨어 공학, 정보보호

차 성 덕 (Sung Deok Cha)
1983년 : UC Irvine 전산학과 학사 졸업
1986년 : UC Irvine 전산학과 석사 졸업
1991년 : UC Irvine 전산학과 박사 졸업
1994년~2001년 : 한국과학기술원 조교수
2001년~현재 : 한국과학기술원 부교수
〈관심분야〉 소프트웨어 공학, 정보보호