침입탐지시스템 평가를 위한
침입패턴 자동 생성에 관한 연구

연세대학교 대학원
컴퓨터과학·산업시스템공학과
노 영 주

침입탐지시스템 평가를 위한
침입패턴 자동 생성에 관한 연구

지도 조 성 배 교수

이 논문을 석사 학위논문으로 제출함

2003년 7월 일

연세대학교 대학원

컴퓨터과학·산업시스템공학과

노 영 주

노영주의 석사 학위논문을 인준함

심사위원 _____ 인

심사위원 _____ 인

심사위원 _____ 인

연세대학교 대학원

2003년 7월 일

감사의 글

이 논문이 완성되기까지 많은 분들께서 도움을 주셨습니다. 도움을 주셨던 고마우신 분들께 감사의 마음을 전해드리며, 항상 최선을 다하는 삶을 살아가겠습니다.

학위를 준비하기 위한 2년 동안 수업과 프로젝트, 그리고 연구실의 일까지 모든 것이 제게는 배워나가는 하나의 과정이었습니다. 자신의 능력을 테스트 하기보담 더 낳은 모습을 찾기 위해 훈련하는 과정이리라 생각합니다. 석사학위 논문을 들고 뒤돌아서서 석사 과정의 자신을 봤을 때 아쉬움이 많이 남습니다. 앞으로 더욱 노력하여 성숙된 모습으로 다시 만날 수 있도록 하겠습니다.

석사과정동안 연구하는 방법과 논제선정에서 완성에 이르기까지 끊임없는 지도와 격려로 이끌어 주신 조성배 교수님께 진심으로 감사드립니다. 또한 바쁘신 중에도 기꺼이 논문의 심사와 세심한 지도를 해주신 차호정 교수님과 이경호 교수님께 감사드립니다.

학위과정 동안 불편하신 몸으로 항상 아들을 걱정해 주시던 아버님과 묵묵히 곁에서 힘이되어 주신 어머님께 이 논문을 바칩니다. 부모님의 크나큰 사랑에 한없이 감사드립니다. 또한 물질적, 정신적으로 많은 보살핌을 주신 외숙부·모님께 감사의 말씀을 올립니다. 제게 소중한 동생 노영경과 현역 복무중인 노영일, 그리고 다른 가족 분들께도 감사의 말씀을 드립니다.

소프트컴퓨팅이란 방제 아래 여러 연구생들과 함께 연구실 생활을 하며 참으로 많은 일이 있었다는 생각이 듭니다. 우선 많은 신경을 써주신 양승룡 형님께 감사드립니다. 저와 함께 침입탐지팀을 이끌어준 한상준, 구자민, 생체인식팀 홍진혁, 윤은경, 바이오인포메틱스팀 박찬호, 원홍희, 유시호, 의사결정분류팀 김경중, 민현정 누나, 모두에게 감사를 드립니다.

또한 정신적 도움을 많이 주셨던 분들도 생각이 납니다. 1년 선배로서 친한 친구로서 지냈던 박혁장과 이종하, 일본에서 구슬땀을 흘리시는 이승익 박사님, 육군장교로 복귀한 이재성 형님, 공군장교로 복귀하신 기균도 박사님, 항상 밝게 웃으시며 말씀을 들어주신 윤성수 박사님, 열심히 일하는 모습이 아름다운 장자인 누나, 새내기 소위 임재경, 까망이 김귀주, 제게 대학원이란 길을 열어주신 이시진 교수님과 항상 아낌없는 충고를 주신 오재학 박사님께 감사를 드립니다.

2003년 7월

차 례

그 림 차 례

표 차 례

국 문 요 약

침입탐지시스템 평가를 위한 침입패턴 자동 생성에 관한 연구

컴퓨터 시스템에 대한 불법적 침입이 증가하고, 해킹 기술의 발달로 인해 침입탐지시스템에 대한 많은 연구와 개발이 이루어졌다. 침입탐지시스템이란 내부자의 불법적인 사용, 오용 또는 외부 침입자에 의한 중요 정보의 유출 및 변경을 알아내는 것으로서 각 운영체제에서 사용자가 발생시킨 키워드, 시스템 호출, 사용시간, 네트워크 패킷 등의 분석을 통하여 침입여부를 결정한다. 침입탐지 기술의 세계적인 현황은 다양한 방법을 통해 침입상황에 맞는 최적의 침입탐지 방법을 찾으려는 노력을 하였고, 제품개발 또한 많이 이루어진 상태이다. 현재는 이러한 침입탐지시스템의 증가에 따른 다양한 평가와 성능 최적화가 중심적인 연구 대상이 되었다. 침입탐지시스템의 평가는 성능평가와 관련하여 기존의 알려진 침입을 주로 사용하므로, 새로운 침입에 대한 탐지능력을 측정할 수 없는 한계점이 있다. 그러므로 새로운 침입에 대한 평가 성능을 시험 할 수 있는 침입패턴 생성에 관한 연구가 필요하다. 따라서 본 연구에서는 호스트 기반 침입탐지시스템의 평가를 위해 알려진 침입 코드를 기반으로 새로운 침입패턴을 자동으로 생성하는 방법을 제안한다.

새로운 침입패턴을 만들기 위한 방법 중 가장 대표적인 방법은 침입탐지시스템이 사용하는 침입척도에 변화를 주는 것으로서, 호스트 기반 침입탐지에서는 시스템 호출 정보가 가장 중요한 척도로 사용된다. 기존 침입 코드를 대상으로 침입척도의 변형을 주기 위해서는 침입코드에 대한 연구, 분석이 필요하다. 본 논문에서

침입코드를 의미모듈이란 가장 작은 기능단위와 비슷한 기능들의 4가지 군으로 나누고 도메인 룰을 이용하여 의미모듈 사이의 상호 의존성을 파악하였고, 이를 토대로 Rulebase 위치변형 방법과 모듈 규칙을 이용한 더미코드 삽입방법을 사용하였다.

위치변형은 rule-set을 사용하여 의미모듈들 사이의 위치이동과 4가지의 의미모듈 군집간 위치이동을 하였다. 또한 규칙 적용의 범용성을 위해 침입코드의 형태에 따라 규칙을 다르게 적용하였다. 침입코드의 형태는 main()함수만으로 최적화 되어있는 형태와 main()함수와 서브 함수들로 구성된 형태로 나누었다. 또한 더미코드 삽입은 시스템을 정상적으로 사용할 때 주로 발생하는 시스템 호출을 넣기 위해 불법 침입이 발생할 때 생성되는 시스템 명령 분석을 통한 정상 시스템 명령을 삽입하는 방법을 제안하였다. 통계적 기법을 사용하는 침입탐지시스템에서는 더미코드의 삽입뿐만 아니라 횟수 또한 중요한 척도로 사용되기 때문에 침입이 실행될 때 나타나는 가장 긴 시스템 호출 정보와 짧은 시스템 호출 정보를 절충하여 5회부터 10회까지로 제한을 두었다.

침입변형에 따른 결과로 Rusebase를 이용한 의미모듈 위치변환 방법에서 rule set은 크게 의미모듈 그룹내 변형과 그룹간 위치변형을 하는데, 실험을 통해 그룹내 변형이 그룹간 변형보다 효과적으로 나왔다. 이것은 의미모듈 사이의 의존성과 관계있는 결과로 그룹으로 위치 이동을 하는 그룹간 변형이 의존성에 더 강한 결속을 보여 오히려 의미모듈의 이동 침입 변형률이 떨어지게 된다. 모듈규칙을 이용한 더미코드 삽입에서는 통계적 기법을 적용한 침입탐지시스템의 적용을 고려하여 더미코드에 사용되는 시스템 명령의 종류와 횟수에 제한을 두었고, 실험결과 더미코드 삽입은 모두 변형에 성공하였다. 솔라리스 버전에 따른 변형률은 2.8기반 침입이 2.7보다 더 다양한 변형을 할 수 있는 결과를 나타낸다. 이것은 침입대상이 되는 OS 버전이 높아짐에 따라 같은 이름의 침입이라도 프로그램 복잡도가 더욱 높아져서 위치변형 가능성이 더 높다는 것을 알 수 있다. 더블어 침입종류 구분이 되는 버퍼오버플로우 침입이 구성설정 오류 침입 보다 더욱 유연하게 위치이동을 통한 침입변형률이 나타난다. 마지막으로 수동변형과 자동변형 시스템에 따른 최종 침입 변형율은 6.1%와 4.8%로 나타나 변환 자동성에 대한 성공도

가 높다는 것을 확인 할 수 있었다. 따라서 수동 변환에서 총 변환 가능한 개수가 작기 때문에 Rulebase 방법이 위치변환을 위한 적절한 방법임을 판정할 수 있다.

핵심이 되는 말 : 침입패턴 자동생성, 침입변형, 의미모듈 위치변환, 더미코드 삽입, 규칙기반 침입변형, 침입탐지시스템 평가

제 1 장 서론

지난 수년간 인터넷은 컴퓨터 환경을 크게 변화시켜 왔으며 정보통신에 대한 관심과 필요성을 더욱 증대시켰다. 컴퓨터 환경의 발전에 따른 관련 시장의 양적, 질적 팽창으로 인해 정보통신은 현대 사회의 중요한 기반으로 자리잡게 되었다. 그러나 이런 결과는 정보통신의 발전에 따른 긍정적인 영향의 증가뿐만 아니라 부정적인 요소 역시 증가한다는 것을 의미하며, 컴퓨터 시스템이 공격대상이 될 가능성이 더욱 커진다는 의미를 내포한다. 실제로 컴퓨터 통신망의 확대는 수많은 사용자들에게 편리하고 다양한 정보를 제공할 수 있게 하였으며 고급 정보의 전송 또한 날로 증가하고 있다. 그러나 이와 같은 정보가 인터넷에 연결되어 비단 전문적인 공격자들뿐만 아니라 단순한 호기심에 의한 공격자들의 표적이 되고 있다.

한국정보보호진흥원에 따르면 국내외 해킹 발생 현황은 인터넷이 보급되기 시작한 99년 이래로 폭발적으로 증가하고 있으며, 한국의 경우 최근 2001년 5,333건의 해킹사고가 접수되었던 것이 2002년에는 15,192건으로 크게 증가하여 연 평균 300% 이상의 증가율을 보인다[1]. 해킹의 대상은 대부분 네트워크 규모가 방대하여 보안 정책 수립이 어려운 기업이나 학교 등이 주된 목표였으며, 상당수가 다른 국가의 시스템을 침입하기 위한 우회 경로로 사용되었다.

표 1. 국내·외 연도별 해킹사고 발생현황

	미국	일본	한국
97년	2,134	498	64
98년	3,734	923	158
99년	9,859	788	572
00년	21,756	4,783	1,947
01년	52,658	40,274	5,333
02년	82,094	112,346	15,192

공격을 위한 도구 또한 예전에는 단순히 시스템의 버그를 이용하는 것들이 주류를 이루었으나, 최근에는 은닉화(stealth), 분산화(distributed), 그리고 자동화(automation)의 특성을 갖는 공격 방법들이 늘어나고 있다[2]. 이에 따라 침입탐지에 대한 요구가 증가되었고 방화벽과 함께 불법적인 침입을 완벽하게 막을 수 있는 침입탐지시스템이 개발, 연구되었다.

침입탐지시스템은 시스템의 불법적인 사용이나 오용, 남용 등에 의한 침입을 탐지해내는 것으로, 단일 컴퓨터는 물론이고 네트워크로 연결된 여러 컴퓨터의 보안에 이용할 수 있다[3]. 이러한 침입탐지시스템은 기본적으로 감사 수집, 분석 및 탐지, 침입탐지 대응, 데이터 저장 기능 요소로 구성되며, 감사기록, 시스템 테이블, 네트워크 부하기록 등의 자료로부터 사용자의 행위에 대한 정보를 분석하여 침입을 탐지한다[4].

침입탐지시스템의 탐지방법은 크게 오용탐지와 비정상행위탐지로 나눌 수 있다. 오용탐지 기법은 알려진 공격에 대한 정보를 구축한 후 사용자나 시스템 또는 프로그램의 현재 행동이 공격패턴과 일치하는지를 검사한다[5][6][7]. 공격패턴 정보를 가지고 있으므로 정상행위를 공격행위로 간주하는 오류(false-positive error)가 낮고, 공격패턴만 검색하면 되므로 경제적인 장점이 있는 반면 알려지지 않은 새로운 공격은 탐지할 수 없다는 단점을 가지고 있다. 이에 비해 비정상행위 탐지 기법은 모델링된 정상행위에서 벗어나는 행동을 공격행위로 간주하기 때문에 공격행위를 정상행위로 간주하는 오류(false-negative)가 낮다[8][9][10]. 그러나 정상행위 모델링을 위해서 다량의 데이터를 분석해야 하므로 구현비용이 높고, 학습되지 않은 정상행위는 비정상행위로 간주되므로 정상행위가 공격행위로 간주되는 오류가 높다[11].

침입탐지시스템과 관련된 시장은 97년 4000만불에서 98년 1억불로 성장하는 등 급격하게 성장하고 있어서 이와 관련된 새로운 제품들이 국내외에서 계속 출시되고 있다. 99년을 기준으로 국외의 대표적인 상용 침입탐지시스템은 15종류, 정보용 침입탐지시스템은 8종류, 기타 연구용 침입탐지시스템은 58종류, 국내에서 판매되는 침입탐지시스템은 12종류나 되었으며, 이 수치는 점차적으로 증가하는 실정이다.

다양한 침입탐지시스템이 개발됨에 따라 특정 시스템에 필요한 침입탐지시스템을 선택하는 것은 매우 중요한 일이다. 여러 침입탐지시스템의 특징과 성능을 평가하기 위한 도구의 필요성이 증가하고 있으나 대부분 수동적이 성능평가만이 이루어지고 있으며, 이에 대한 자동화된 평가도구는 아직 개발 단계에 머무르고 있다.

현재 평가의 주된 연구 방향은 공격들에 대한 다양한 환경에서의 대응 평가 및 단순한 통계적인 출력에 초점이 맞추어져 있다. 그런데 새로운 종류의 침입이 급속도로 증가하고 있는 현 상황에서는 특정 침입에 대한 다양한 상황에서의 탐지성능을 측정하는 것 뿐 아니라 새로운 침입에 대한 탐지성능을 측정하는 것 또한 중요한 문제이다.

국내의 침입탐지시스템에 대한 평가는 한국정보보호센터에서 이루어지고 있으나 현재는 초기 단계로 평가를 위한 각종 방안들이 마련되고 있는 상황이다. 미국의 경우 수년간의 DARPA 프로젝트 수행으로 1998년부터 2000년도까지 대규모의 침입탐지시스템에 대한 평가가 이루어졌고, UC Davis와 MIT Lincoln lab 그리고 Trust사에 소속되어 있는 ICSA에서 각각 침입탐지시스템에 대한 평가를 하였다.

침입탐지와 침입탐지시스템의 평가를 연구하는 여러 집단에서 새로운 침입패턴은 매우 중요한 이슈가 되고 있다. 현재 알려진 침입패턴은 오용탐지기법 침입탐지시스템과 침입탐지시스템 평가에 사용되고 있으며, 침입탐지시스템 평가를 위해 알려지지 않은 새로운 침입을 찾기는 매우 어렵다. 또한 새로운 침입패턴을 찾더라도 많은 해킹 집단에서 침입 패턴을 널리 사용하고 있는 상황이여서 침입탐지시스템 평가에 미리 적용하지 못한다. 따라서 더욱 향상된 침입탐지시스템 평가를 위해 새로운 침입패턴을 자동으로 생성하는 연구가 반드시 필요하다.

본 논문에서는 침입탐지시스템 평가를 위한 침입패턴 자동 생성을 위해 다음의 문제에 관해 연구한다.

- ❑ 침입을 나타내는 방법과 분류방법은 어떻게 되는가?
- ❑ 각각 구분된 침입에 따른 변환이 가능할 것인가?

- ❑ 어떤 침입 변환 기법과 생성 기법을 사용할 것인가?
- ❑ 어떤 방법으로 침입패턴 생성 모델을 구축할 것인가?
- ❑ 침입패턴 변형의 자동화는 어떻게 할 것인가?
- ❑ 생성된 침입패턴의 성능 평가 측정을 어떻게 할 것인가?

다양한 침입을 생성하기 위해서는 각 침입 행위 형태를 파악하고 침입 유형을 적절한 척도에 따라 나누어야 한다. 침입의 분류는 과거 연구를 통해 제안되어왔는데 대표적으로 Warwick 분류, Landwehr 분류, Anderson 분류, CERTCC-KR 분류를 들 수 있다. 이러한 분류는 침입을 시스템의 취약점 위주로 분류 하거나 해킹으로 인한 영향, 위험성 등에 따른 분류 결과를 보여주고 있다.

새로운 침입 패턴을 만들기 위한 방법은 기본적으로 나뉘어진 분류 중 평가 대상이 되는 침입탐지시스템의 종류에 따라 주로 사용되는 침입범주를 선택해야 한다. 본 연구에서는 호스트 기반 침입탐지시스템의 평가를 위해 버퍼오버플로우 침입과 구성설정 오류를 Rulebase 기법을 이용한 의미모듈의 위치변환 방법과 모듈 규칙을 이용한 더미코드 삽입방법을 통해 침입패턴을 변형하여 새로운 침입을 만들어냈다. 또한 변형된 침입패턴을 침입탐지시스템 평가도구에 적용하여 침입탐지 평가를 하였다.

제 2 장 배경

2.1 대표적 침입탐지시스템 평가

침입탐지시스템 평가관련 기존 연구는 침입탐지시스템의 성능평가와 관련하여 DARPA, UC Davis[12][13], MIT Lincoln lab[14][15][16], ICSA[17] 등이 있으며 , 보증성 평가와 관련하여 한국정보보호진흥원(KISA)의 정보통신망 침입탐지 기준에 의한 평가[18][19]가 있다.

❑ DARPA

초기의 침입탐지시스템에 대한 평가는 적은 수의 시스템에 대해 단순한 트래픽을 포함한 은닉이 아닌 적은 수의 공격만으로 이루어졌으나 1998년 DARPA의 1차 평가에서는 10개의 시스템, 38개의 공격, 충분한 트래픽과 일부의 은닉화된 공격을 통한 평가가 이루어졌다. 2000년 DARPA의 침입탐지평가는 사용자의 편의성과 성능향상에 중점을 두었다. 특히 사용자 편의성을 위해 GUI 환경을 만들어 적용시킴으로써 사용자로 하여금 공격데이터의 추가와 공격환경의 설정을 가능하게 하였다. 하지만 사용자의 편의성을 향상시킨 반면 다양한 공격데이터의 수용에서 미흡한 면을 보였다.

❑ MIT Lincoln 연구소 평가

1998년부터 DARPA의 지원아래, 침입탐지 관련 연구를 위한 방향을 제시하고 기술에 대한 객관적 교정을 위해서 침입탐지 평가를 주제로 과제를 수행하였다. 각 과제에서 침입탐지시스템들은 훈련용 데이터를 통해 정보를 제공받고 검사용 데이터를 통해 성능을 평가받았다. 훈련용 데이터에 삽입된 공격에 대해서는 관련된 모든 정보가 제공되었으며, 검사용 데이터에 포함된 공격에 대해서는 어떠한 정보도 제공되지 않았다. 평가항목은 침입탐지율, 오판율, 자원사용량 등이며, Denial of Service, Remote to User, User to Root, Remote to Local, Probing

등 공격방법을 유형별로 분류하고 Solaris, Windows, NT, Linux 등 운영체제 특성을 고려한 침입패턴을 생성하여 탐지 시험, 탐지 우회(Evasion) 시험을 수행하였다. 시험에 사용되는 정상 데이터는 tcpdump[20]를 사용하여 실제 패킷을 캡쳐한 후 재생을 통하여 telnet, ftp, SMTP, HTTP 등의 데이터를 생성하고 있으며, 침입패턴은 공격형태를 분류하여 운영체제 특성을 고려한 침입 패턴을 생성하여 Expert[21]라는 도구를 사용하여 침입스크립트를 실행한다. 평가방법은 평가환경 조건 마련을 위한 정상 데이터 생성후 주어진 환경에서 분류한 침입탐지 형태를 기반으로 다양한 운영체제별 특성을 고려한 침입패턴 데이터를 실행하여 침입율 및 오판율 등을 시험하고, 효과적인 탐지를 위한 패턴 별 평가값을 구한다.

❑ UC Davis 평가

평가항목은 침입탐지율, 오판율, 자원사용량 등이며, 각 평가항목별로 테스트 케이스를 제시하여 침입탐지시스템을 실행하여 침입패턴 실행 및 탐지 결과를 측정하는 침입식별 시험(Intrusion Identification Tests), 잡음 또는 CPU 부하 등 스트레스를 가하여 침입탐지 능력을 측정하는 스트레스 시험(Stress Test), 침입차단시스템의 CPU 및 메모리 사용량을 측정하는 자원사용량 시험(Resource Usage Tests)을 수행한다. 시험 데이터 생성은 telnet을 수행하여 정상행위 스크립트를 생상하고, 침입탐지 패턴 생성을 위하여 사전에 스크립트로 작성한 침입패턴을 실행하는 Solaris 기반의 Expert 도구를 사용하고 있다. 평가방법으로 침입탐지 시스템을 수행하고 잡음 또는 CPU 증가 등 해당 시험 조건을 가한 후 침입 스크립트를 실행하여 탐지율, 오판율을 측정하고 잡음 또는 CPU부하가 가하지 않은 조건에서 수행한 순수 침입패턴 시험과 비교 분석하며, 자원사용량을 분석한다.

❑ ICSA 평가

ICSA는 미국의 Trust사에 소속되어 있는 네트워크 정보보호제품 성능 및 취약성 시험을 중심으로 평가하는 사설 시험기관으로서 작성된 침입탐지 시험기준[22]을 근거로 평가를 수행한다. 시험 데이터 생성은 패킷 생성기 등을 사용하며,

침입패턴은 업체 Consortium을 통해 입수된 침입패턴을 사용하여 10M 및 65M bps이상에서 100% 탐지함을 시험한다. 또한 침입 우회 가능성 시험을 위하여 URL encoding, IP Fragmentation 등을 수행하며, DoS 공격 및 악성 코드를 이용하여 취약성을 시험한다. 평가방법은 상용 침입탐지시스템을 설치하여 일정 수준의 환경을 구성한 후 침입패턴을 실행하여 침입탐지율, 오판율, 침입탐지 로그 수준을 시험하며, 침입 우회 가능성 및 취약성 시험은 별도로 수행한다.

❑ 한국정보보호진흥원 평가

기능요구사항과 보증요구사항으로 구성된 평가기준을 근거로 신청 평가등급(K1-K7)별 요구사항에 의해 평가를 수행하며, 평가항목은 침입탐지기능 식별 및 인증, 무결성, 보안감사 등 시스템의 보안기능을 평가하며 개발자가 제시한 환경 범위를 고려한 침입탐지율, 오판율, 침입대응, 침입로그 및 침입 우회 가능성 시험 이외에 침입탐지 엔진 설계 내용 등 개발문서를 병행하여 평가한다. 침입탐지 시험에 사용되는 정상 데이터는 패킷 생성기 및 telnet, ftp 등 실제 데이터를 tcpdump 등 패킷 캡쳐 프로그램을 이용하여 저장한 후 재생을 하며, 침입패턴은 개발 업체가 제공하는 패턴 및 별도로 수집 또는 제작한 침입패턴을 이용한다. 평가방법은 설계 등 문서 평가를 수행하여 요구사항, 설계, 구현 간 일치성을 분석하고 개발자가 제시한 조건에 적합한 평가환경 구성 및 평가대상 침입탐지시스템을 설치하여 침입패턴을 생성 후 네트워크 부하가 없을 경우 및 부하가 있을 경우, 침입탐지 여부 및 IP Fragment등 우회가능성을 평가한다.

1998년부터 2000년까지 DARPA, UC Davis, MIT Lincoln lab, ICSA, 한국정보보호진흥원에서 이루어진 침입탐지시스템의 평가는 현재의 침입 수준에 초점이 맞추어져 있으므로 특정 공격들에 대한 다양한 환경에서의 적응이 주된 평가 방향이었고, 따라서 어떤 종류의 가상 환경을 만드는가에 대해 연구되어 왔다. 그런데 여기에는 새로운 종류의 침입들이 증가함에 따른 침입패턴과, 침입유형 등의 관리 및 결과에 대한 통계적 분석 기능에 대한 개발이 미흡하기 때문에 이에 대한 보완이 필요하다.

2.2 시스템 평가

일반적인 프로그램의 평가 방법은 Black Box 기법과 White Box 기법이 있다. Black Box 기법은 평가대상에 대한 구조나 내부 모듈의 정보를 모르고 평가 요건에 맞추어서 대상을 평가하는 방법이고, White Box 기법은 평가대상에 대한 구조와 내부 모듈을 알고 그것들을 평가시 일부 이용하여 적용시키는 것이다[23]. 본 연구에서는 Black Box기법을 적용한 침입탐지시스템 평가도구의 평가 요건에 맞춘 침입패턴들을 이용하여 침입변형에 대한 연구를 하였다.

2.3 침입분류

침입탐지시스템 평가에 사용하기 위한 침입의 종류와 가지수는 매우 방대하여 새로운 침입패턴을 생성하는데 어려움이 있으므로 제안하는 시스템 개발에 앞서 사용될 침입을 분류하고 선택할 필요가 있다. 이러한 사이버 공격 및 취약성에 대한 분류를 여러 조건들에 대해 만족하도록 분류하는 것은 현실적으로 매우 어렵다. 한 가지의 공격은 여러 가지의 취약성을 이용할 수 있으며 다양한 공격결과를 포함하기 때문이다. 보통 해킹기법을 분류하는 방법은 국제적으로 다양한 방법들이 알려져 있다. 단순히 해킹을 당할 수 있는 시스템의 취약점 위주로 분류하는 방법이 있으며 해킹으로 인한 영향, 위험성 등에 의해서도 분류하는 방법이 있다. 대표적 분류방법은 다음과 같다.

표 2. 평가 목록 분류

분류	설명
용어,카테고리 목록방법	가장 간단하고 대중적인 방법
Warwick 분류	현존하는 네트워크 보안 서비스 설명
Landwehr 분류	발생 원인, 발생시간, 발생 위치의 삼차원을 기초로 보안 취약성 분류
Anderson 분류	앤더슨에의해 정의되어진 4가지 대표적 분류법
CERTCC-KR 분류	한국정보보호진흥원에서 10가지로 분류
시스템 취약성	소프트웨어 약점을 기준으로 취약성 분석

❑ 용어, 카테고리 목록 방법

현재 컴퓨터 보안 분야에서 사용하는 분류방법 중 가장 대중적이고 간단한 것은 용어 목록 방법이다. 하지만 용어 목록 방법은 상호배제의 특징에서 문제점을 가지고 있다. 예를 들어 바이러스에 해당하는 공격도구는 논리폭탄에도 포함될 수 있으므로 분류의 중첩을 일으킨다. 이러한 이유로 용어 목록 방법은 적합한 분류 방법이라고 할 수 없다. 카테고리 목록 방법은 용어 목록 방법과 비슷한 방법으로 다소 향상된 방법이지만, 용어 목록 방법과 비슷한 분류방법을 사용하기 때문에 비슷한 문제를 가지고 있다. 카테고리 목록방법은 공격뿐만 아니라 공격 결과의 분류 및 방어 메커니즘, 취약성 분류 등에 사용할 수 있다.

❑ Warwick 분류

Warwick Ford는 그의 저서 “Computer Communication Security" (1994 prentice hall)에서 현존하는 네트워크 보안 서비스를 설명하기 위해 다음과 같은 세 가지 위협의 분류를 정의하였다.

표 3. Warwick 분류 내용

항목	침입분류	내용
Fundamental threats	Information leakage	인가되지 않은 사람이나 매체에 정보가 노출되거나 누출되는 것으로 도청, 스니핑과 같은 정보를 감시하는 공격이 해당된다.
	Integrity violation	데이터의 내용이 인가되지 않은 방식에 의하여 생성, 변경, 파손 또는 삭제되는 것.
	Denial of Service	시스템이 정보나 자원에 합법적인 접근을 허용하는 것을 이용하여 대용량의 불법적인 실패접근을 시도함으로 정상 이용자에게 정보 및 자원 서비스에 장애를 일으키는 것.
	Illegitimate use	시스템 자원이 인증되지 않고 사용되는 것을 의미, 예를 들어 전화를 도용하여 침입하는 것이나 다른 사람의 시스템을 침투하기 위한 거점으로 삼는 것 등.
Primary enabling threats	Masquerade	다른 시스템이나 사람으로 위장하는 행위
	Bypassing Controls	비 인가된 권한을 획득하기 위해 공격자가 시스템 결함이나 보안 취약성을 이용하는 것
	Authorization violation	시스템이나 자원을 사용 가능하도록 권한을 인가 받는 자가 이를 다른 목적을 위해 사용하는 것. 이것을 내부 침입이라고도 한다.
	Planting threat	시스템에 백도어들이 이식되는 위협
Underlying threats	eavesdropping	도청
	traffic analysis	트래픽 분석
	indiscretions by personnel	관리 부주의
	media scavenging	매체 폐기처리 부주의

❑ Landwehr 분류

Landwehr는 발생 원인(고의적인가, 우발적인가), 발생 시간(소프트웨어나 하드웨어의 생명주기에서의), 발생 위치(소프트웨어나 하드웨어)의 삼차원을 기초로 한 컴퓨터 보안취약성 분류를 제시하였다. 이것은 카테고리 목록 방법과 같은 일차원 매트릭스 분류법에 대하여 상호배제의 특징과 포괄성의 특징에서 향상된 방법이다.

❑ Anderson 분류

Anderson 분류의 두 형태 중 하나는 의심스러운 행위를 묘사하는 것이고, 또 하나는 잘 알려진 행위를 묘사하는 것이다. 그리고 이렇게 분류된 침입의 형태를

시간의 흐름에 따라 그 행위를 묘사하는 것을 방법으로 하였다. 하지만, 이 분류 항목에 포함된 침입의 흔적은 실제 시스템을 침입한 침입자가 공격한 자료를 기준으로 하였기 때문에 모든 침입을 묘사하지는 못한다.

Anderson 공격 유형 분류는 네 가지의 큰 분류, Denial of Service, Remote to User, User to Root, Surveillance/ Probe로 되어 있어 구체적으로 공격에 이용되는 시스템 자원의 공통점과 시스템에 미치는 영향의 공통점을 찾아내기 어렵다. 또한 모든 공격의 형태가 순차적으로 나타나지 않는다는 점을 고려하고 있지 않다.

❑ CERTCC-KR 분류

현재 대부분의 해킹 대응을 위한 서비스를 제공하는 각종 업체 및 기관에서 주로 사용하고 있는 해킹 기법 분류안은 Marcus J. Ranum에 의해 분류된 것으로 한국정보보호진흥원 에서는 이 분류 방법을 일부 확장하여 다음의 10가지로 분류하였다[22].

표 4. CERTCC-KR 분류 내용

침입분류	내용
사용자 도용	접근 권한이 없는 정보통신망에 침입하는 행위로 보통 정상이용자의 ID, 패스워드를 도용하여 신분위장 접근하는 방법
SW보안오류	프로그램 설계 및 구현상의 버그로 인한 취약점을 이용하는 방법
버퍼 오버플로우 취약점	지정된 버퍼의 크기보다 더 많은 데이터를 입력해서 프로그램이 비정상적으로 동작하도록 만드는 것
구성설정오류	시스템에서 제공하는 각종 서비스의 설정과 관련된 취약점을 이용한 해킹 기법
악성프로그램	바이러스, 트로이잔, 백도어 등을 이용하여 시스템을 해킹 하거나 시스템에 치명적인 위험을 주는 방법
프로토콜취약점	TCP/IP뿐만 아니라 각종 인터넷 프로토콜(ICMP, ARP, RARP, UDP 등등)의 설계상의 취약점
서비스거부공격	시스템이 정상적으로 동작하지 않도록 만드는 공격 기법
이-메일 관련 공격	E-mail을 통하여 스팸메일 등을 한꺼번에 보내는 것
취약점정보수집	취약점 분석 도구 혹은 시스템 명령어 등을 이용하여 해킹 하고자 하는 시스템에 대한 정보를 수집하는 행위
사회공학	정보시스템 접근에 도달하기 위해 설득이나 속임수를 사용하는 기법

2.4 침입유형

호스트기반 침입탐지시스템 평가를 위한 새로운 침입패턴을 생성하기 위해서는 발생될 수 있는 침입 유형을 분석하고 이를 통하여 각 유형에 적절한 침입 패턴을 선택해야 된다. 현재 업체 및 기관에서 주로 사용하고 있는 침입유형은 한국정보보호진흥원에서 발표한 10가지로 분류를 사용한다. 그 중 호스트에서 발생 가능한 유형은 버퍼오버플로우, S/W 보안오류, 구성설정 오류, 서비스거부공격 등이 있다. 각 유형의 자세한 내용과 해킹 내용을 살펴보면 다음과 같다.

❑ 버퍼오버플로우 : 이 공격은 프로그램을 실행시키는 과정과 메모리구조에 대해서 자세히 알아야 이해할 수 있는 아주 어려운 기술 중의 하나이고 메모리와 스택의 구조나 OS에 따라 다르기 때문에 접하기가 쉽지는 않았다. 하지만 현재 인터넷을 통하여 버그에 대한 소스 코드가 공개되어 있고 누구나 다운을 받아 운영체제에 맞게 사용할 수가 있기 때문에 가장 많이 쓰이고 있는 해킹 방법이다. 이러한 버그는 C 언어에서 데이터가 지정된 버퍼의 크기보다 많이 입력되었는지를 체크하지 않는 것을 이용하여 버퍼가 오버플로우 되는 순간 쉘을 실행시킴으로써 임의의 작업을 수행한다. 특히 SETUID가 설정된 루트 프로그램인 경우 버퍼오버플로우를 이용하여 쉽게 루트 권한의 쉘을 얻을 수 있다[24].

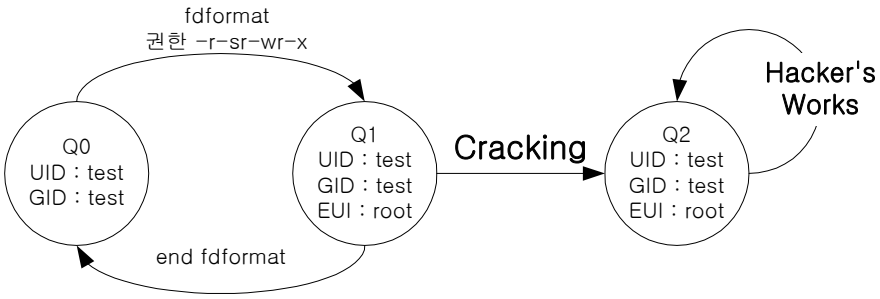

그림 1. 버퍼오버플로우 공격시 권한이동 예

표 5. 버퍼오버플로우를 이용한 해킹의 예

공격 방법	설명
imapd	로그인 버퍼오버플로우 로그인 시 소문자를 대문자로 바꿔주지만 쉘 코드 안에 수정코드 넣을 가능성 있음
samba buffer overflow	패스워드를 비정상적으로 길게 입력할 경우 발생
named buffer overflow	Fake inverse query를 할 때 버퍼오버플로우 가능성
etc	그밖에 로컬상태의 프로그램에서 다수 발생

❑ S/W 보안 오류 : 소프트웨어 보안 오류는 프로그램 설계 및 구현상의 버그로 인한 취약점을 이용하는 방법이다. 프로그램 상의 취약점에는 프로그래밍 상의 보안 오류와 프로그램 동작상의 보안 오류로 인하여 발생하는 취약점이 있다.

표 6. S/W 보안 오류를 이용한 해킹의 예

공격방법	설명
cgichk	CGI 스캔을 통하여 취약성 있는 cgi 검사
sendmail	7비트에서 8비트로의 변환을 제공하는 sendmail 8.8.3 또는 8.8.4에서 원격의 사용자가 루트의 권한을 얻을 수 있는 취약점 발견
IIS 버그	웹 상에서 bat , cmd, exe 파일의 위치를 알아서 그것의 절대 패스를 걸어주면 파일실행되는 버그. ::$DATA ASP가 있는 곳 뒤에 ::$DATA라고 치면 ASP의 소스가 공개되는 버그
FTP bounce 공격	FTP 프로토콜의 허점을 이용한 공격 방법, 전송 목적지 검사가 없을 경우 발생

❑ 구성 설정 오류 : 보안을 고려하지 않는 시스템을 구성함으로써 발생하는 오류로 이 분류에는 시스템 및 시스템에서 제공하는 각종 서비스의 설정과 관련된 취약점을 이용한 해킹 기법이 포함된다. 일반적인 시스템 보안 취약점 분석도구를 이용하여 발견할 수 있고, 해킹을 위한 특별한 해킹 수행 코드가 필요 없는 경우가 대부분이다.

표 7. 구성, 설정 오류를 이용한 해킹의 예

해킹 방법	설명
.rhosts 설정 오류	대상 시스템의 $HOME/.rhosts의 내용을 참조하여 아무 인증 없이 홈 디렉토리에 접근 가능
IFS환경 변수이용	외부프로그램을 실행할 때 입력되는 문자열을 나눌 때 기준이 되는 문자를 정의하는 변수를 이용한 해킹
CRON 설정오류	슈퍼 유저의 cron 작업중의 실행 파일 이용

❑ 서비스 거부 공격 : 서비스 거부 공격이란 간단히 말해서 시스템이 정상적으로 동작하지 않도록 만드는 공격기법으로 시스템의 관리자 권한을 얻기 위해 시도되는 일반적인 해킹과는 차이가 있다. 특히 호스트내의 서비스 거부 공격은 시스템의 자원을 모두 고갈시켜 서비스가 전혀 불가능하게 하여 피해를 주기 때문에 큰 문제가 되고 있다[25].

표 8. 서비스 거부 공격을 이용한 해킹의 예

해킹 방법	설명
ICMP flooders	ICMP_ECHO flooder, ICMP_UNREACH flooder를 이용한 방법
SYN flooders	Half-open SYN/ACK 상태를 이용하여 TCP 연결을 방해하는 방법
Udp flooders	Udp 프로토콜 취약점을 통해 패킷들을 무한 루프 돌리게 하는 방법

CERTCC-KR 취약성 분류 중 호스트 기반 침입이 가능한 취약성인 S/W 보안 오류와 버퍼오버플로우, 구성 설정 오류, 서비스 거부공격 등의 방생수를 보면 1월에는 43회, 2월에는 45회가 발생된 것을 알 수 있다(표 9). 그 중 버퍼오버플로우 위약성을 이용한 방법과 사용자 권한 설정 오류를 이용한 방법이 거의 90% 정도를 차지하고 있다. 본 논문에서 새로운 침입패턴 생성과 관련하여 사용하게

될 침입은 실제 해킹에서 침입으로 가장 많이 사용되는 버퍼오버플로우와 구성 설정 오류 관련 침입들이다.

표 9. 최근 해킹 동향

침입유형	2003년 1월	2003년 2월	2003년 3월	2003년 4월
S/W 보안오류	130	34	893	452
버퍼오버플로우	445	28	28	43
구성설정 오류	377	808	2,059	1,960
서비스 거부공격	29	0	0	0
총 갯수	981	870	2,980	2,455

제 3 장 침입탐지시스템 평가를 위한 침입패턴 자동생성

3.1 시스템 개요

본 논문에서는 호스트 침입시 가장 많이 사용되는 침입패턴의 유형별 분석을 통하여 침입변형 모델링 방법을 제안하고 변형된 침입패턴을 침입탐지시스템 평가에 적용하였다. 자동 침입패턴 생성기는 크게 침입 전처리 부분과 침입패턴 자동 생성기 부분으로 나뉜다. 침입패턴 전처리에선 알려진 침입패턴을 의미모듈 단위로 분석하고 Domiain-Rule을 통해 각 모듈간 의존성을 파악한다. 전처리가 끝난 침입패턴은 패턴 자동생성기에서 RuleBase를 이용한 의미모듈 위치변환과 모듈 규칙을 이용한 더미코드 삽입을 통해 침입패턴을 변형한다.

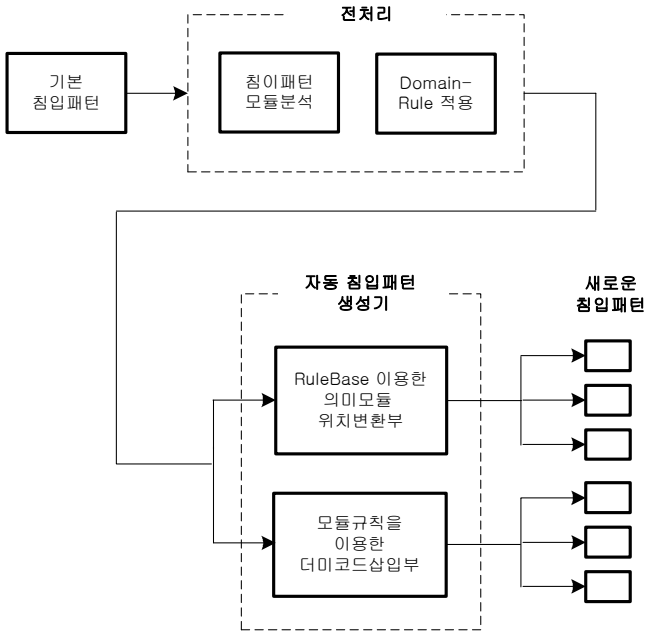

그림 2. 침입패턴 자동생성 시스템 구조도

3.2 침입분석

호스트 기반의 침입탐지시스템에 이용되는 침입은 시스템 내부에서 발생되는 침입으로 범위를 좁힐 수 있고, 버퍼 오버플로우와 구성설정 오류에 의한 root 권한의 획득이 주로 발생한다. root 권한을 획득하는 침입이 실행되면 침입과정을 그림 3과 같이 오토마타를 이용하여 표현할 수 있다.

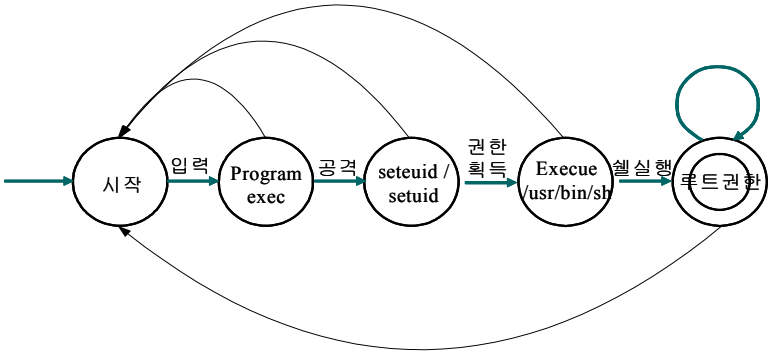

그림 3. u2r(user to root) 침입을 이용한 root 권한 획득과정

본 연구에서는 시스템 내부 침입자들에 의해 발생하는 u2r(user to root) 침입을 기반으로 새로운 침입코드를 생성하였다. u2r 침입은 표 10와 같은 종류가 알려져 있다.

표 10. 알려진 u2r 침입코드

OS	침입분류	침입코드명	침입내용
solaris intel 2.8	버퍼 오버 플로우	kcms_configure_overflow.c	/usr/openwin/bin/kcms_configure overflow
		whodo.c	/usr/sbin/i86/whodo overflow
		xlock.c	xlock heap overflow
		mailx.c	/usr/bin/mailx overflow
solaris sparc 2.7	버퍼 오버 플로우	xlock.c	xlock heap overflow
		xsun.c	/usr/openwin/bin/xsun overflow
		ex_lobc.c	ex_lobc overflow
		dtaction.c	/usr/dt/bin/dtaction overflow
		dtprintinfo.c	ex_dtprintinfo overflow
		ex_admintool.c	admintool overflow
		lpset.c	/usr/bin/lpset overflow
	구성설정 오류	3lc_messages_passwd_stack.c	LC_MESSAGES bug
		eject-fmt.c	locale subsystem format strings bug

u2r 침입프로그램은 코드 내부를 여러 모듈로 나눌 수 있다. 각 모듈은 상황에 따라 한줄의 코드가 되거나 여러줄의 코드가 될 수도 있고, 이 모듈들이 수행하는 일도 여러 가지로 구분된다. 그림 4에서는 모듈들이 침입 코드에서 특정한 일을 수행하는 부분을 나타낸다.

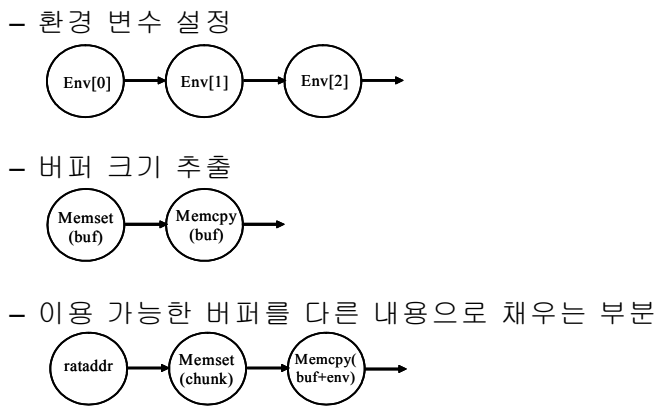

그림 4. 각 모듈의 침입코드 수행 내용

이러한 구성을 가진 침입프로그램을 실행시키면 각 모듈이 순차적으로 실행됨에 따라 시스템 내부에서 여러 이벤트를 발생하면서 침입을 한다. 이때 발생되는 일련의 시간정보와 침입에 따른 행위들이 침입탐지시스템의 척도로 사용되어 침입여부를 판단한다.

3.3 호스트 기반 침입탐지시스템의 침입탐지 척도

침입탐지시스템은 그 규모가 매우 방대하여 탐지 대상척도에 따라 다양한 결과가 가능하며, 특정 상황에서 특정 침입탐지기법에 적합한 척도가 있다. 따라서 다양한 척도에서의 탐지가 적용 가능한 침입탐지기법이 선택된다. 침입탐지에 사용되는 많은 척도들이 과거 연구를 통해 제안되어 왔는데 대표적으로 운영체제에서

발생하는 CPU 사용량, 메모리 사용량, 시스템 호출 정보, 프로세스 척도, 파일 입출력 척도 등 많은 시스템 관련 척도가 통계적인 방법(Statistical Approach), 특징 추출(Feature Selection), 신경망(Neural Network), 데이터 마이닝(Data Mining), 은닉 마르코프 모델(Hidden Markov Model) 등을 사용하여 침입을 탐지한다. 특히 시간 변화에 따른 시스템 호출 관련 척도는 호스트 기반 침입탐지시스템에서 좋은 침입탐지 결과를 내는 중요한 요소이다[26][27]. 대표적인 침입탐지시스템의 척도를 보면 다음과 같다.

표 11. 대표적 침입탐지 시스템의 척도

Measure	IDES	NIDES	SageGuard	CMDS
Activity Intensity Measure				
Directory Activity	○	○		
File Activity	○	○		
Audit Record Distribution Measure				
System call	○	○	○	○
Command invocation	△	△		○
Categorical Measure				
Linear categorical	○	○	○	
Binary categorical	○	○	○	
Ordinal Measure				
CPU, I/O, Memory Usage	○	○	○	
Mail, Editor, Compiler Usage. ETC	○	○		

위에 나타난 침입탐지 척도중 시스템 호출 관련 정보를 이용한 침입탐지시스템의 동작상태를 확인하면 다음과 같다.

1. 시스템의 사용자 레벨에서 활성화된 응용프로그램의 프로세스 중 시스템 호출을 커널에 요구한다.
2. 침입탐지시스템의 감사모듈은 커널레벨에서 시스템 호출요구를 감지하고 엔트리의 내용을 침입탐지시스템 엔진에 전송한다.
3. 침입탐지시스템의 정책엔진은 엔트리의 내용을 확인하고 실행/중지를 내린다.

4. 실행이 되면 커널로부터 호출받은 시스템 호출 내용이 실행되고, 응용프로그램의 프로세스에게 결과를 전달한다.

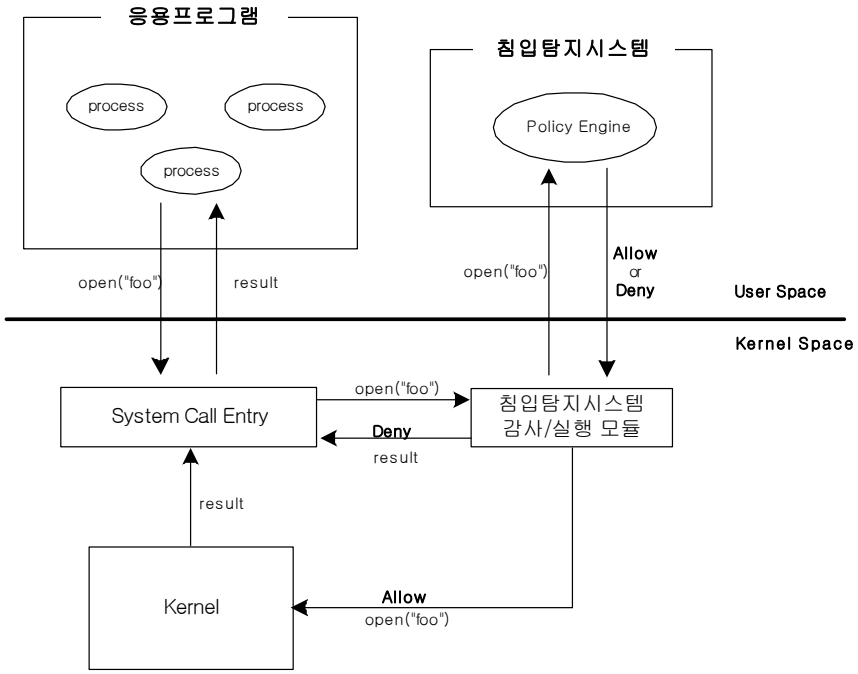

그림 5. 시스템호출 정보를 이용한 침입탐지시스템

특히 침입탐지시스템의 엔진에서 가장 중요하게 받아들이는 정보는 시스템 호출 정보의 순서와 내용이다. 엔진의 구성 원리는 시스템 호출 정보의 순서를 규칙으로 저장한 후 확률적인 계산을 통해 정상패턴과 비정상패턴의 차이를 가려낸다. 이때 침입판단 근거인 시스템 호출 순서와 내용은 침입변형의 중요한 요소로 작용한다[28]. 시스템 로그 데이터 분석을 통하여 호스트 기반 침입 유형의 침입정보를 확인한 결과 구성설정 오류와 버퍼 오버플로우를 이용한 루트권한 획득이 주를 이르고 있다. 구성설정 보안 오류, 버퍼 오버플로우의 경우 시스템 호출 이벤트에서 정상적인 명령어와는 다른 특정 이벤트가 발생하는 것을 확인 하였고 시간 변화에 따른 권한 이동이 탐지되었다. root 권한 획득과 관련된 시스템 호출은 표 12와 같다.

표 12. root 권한 획득 관련 시스템 호출

이벤트 아이디	시스템 호출	이벤트 아이디	시스템 호출
2	fork	27	setpgrp
11	chown	38	fchroot
21	symlink	39	fchown
23	execve	200	setuid
24	chroot	215	seteuid
25	vfork	6159	su

취약점을 이용한 Exploit 코드를 분석할 경우 유사한 특성을 추출할 수 있는데 대표적 버퍼오버플로우 코드를 분석하여 예를 들어보면 표13과 같은 성질을 나타낸다. 각각의 공격은 코드화 되어있고, 코드를 실행시키면 조사된바와 같이 시스템 호출정보를 발생하며 실행된다. 구성설정 오류와 버퍼오버플로우 공격의 최종 목표는 시스템의 root 권한을 획득하는 것이다.

표 13. 대표적 버퍼오버플로우 공격 분석

공격이름	xlock	mailx	kcms	whodo	lpset
이벤트 발생					
execve 실행횟수	3	4	3	4	8
execve 명령어	xlock, ksh	mailx, sendmail, ksh	kcms, sh, kcms_configure	whodo, sh	lpset, fndestroy fnunbind, sh, fncreate_printer
setuid 실행횟수	2	1	1	1	4
setgid 실행횟수	0	14	0	0	0
EUID 권한이동					
user-root	○	X	○	○	○
user-other user	X	○	X	X	X
RUID 권한이동					
user-root	○	○	○	○	X
user-other user	X	X	X	X	X

그중 xlock의 정상 로그파일과 Exploit 파일의 차이점을 살펴보면 시스템 호출의 변화를 확인할 수 있다. 두 로그 파일의 차이를 확인하기 위해 본 논문에서는 Sun사의 Basic Security Module(BSM)을 이용하여 시스템 호출의 변화를 추적하였다[29].

표 14. 정상파일(/usr/openwin/bin/xlock) 실행시 BSM 파일

Event ID	EUID	RUID	Process ID	Session ID	설명
fork	user1	user1	23517	23516	자식쉘 실행
setpgrp	user1	user1	23517	23516	
setpgrp	user1	user1	23517	23516	
execve	root	user1	23517	23516	/usr/openwin/bin/xlock 실행
open	root	user1	23517	23516	
fcntl	root	user1	23517	23516	
exit	root	user1	23517	23516	xlock 실행 파일 종료
setgrp	user1	user1	23517	23516	
setgrp	user1	user1	23723	23516	부모쉘에 대한 변경
setgrp	user1	user1	23723	23516	
setgrp	user1	user1	23723	23516	

표 15. Exploit 파일인 xlock 실행

Event ID	EUID	RUID	Process ID	Session ID	설명
fork	user1	user1	23517	23516	
setpgrp	user1	user1	23517	23516	
setpgrp	user1	user1	23517	23516	
execve	user1	user1	23527	23516	/exploit/xlock 실행
sysinfo	user1	user1	23527	23516	
execve	root	user1	23527	23516	/usr/openwin/bin/xlock 실행
open	root	user1	23527	23516	
seteuid	root	user1	23527	23516	
old setuid	root	root	23527	23516	
execve	root	root	23527	23516	root shell 실행
setpgrp	root	root	23527	23516	
setpgrp	root	root	23527	23516	
setpgrp	user1	user1	23517	23516	부모쉘에 대한 설정변경
setpgrp	user1	user1	23517	23516	
setpgrp	user1	user1	23517	23516	
fork	root	user1	23527	23516	root shell에서 date 파일 실행
execve	root	user1	23527	23516	
exit	root	user1	23527	23516	

표 14, 표 15에서 정상파일과 exploit 파일의 실행을 비교해보면 EUID 권한의 변경에서 큰 차이점을 발견할 수 있다. 일반 파일의 경우 권한이동이 일어난 후 다시 exit 이벤트의 호출과 함께 EUID가 일반 유저로 바뀌지만 해킹시도 시에는 권한의 이동이 일어난 시점에서 execve 파일로 root 소유의 shell을 실행 시켜 EUID는 root 로 유지된다[30]. 따라서 버퍼오버플로우 공격과 구성설정 오류를 이용한 공격을 탐지 하기위해서는 시간에 따른 시스템 호출 이벤트의 변화와 EUID와 RUID의 변화를 측정하면 좋은 탐지가 가능하다.

시스템 호출 척도는 전통적으로 IDS에서 가장 많이 쓰이는 척도로 시간에 따른 시스템 호출 이벤트의 변화를 측정하는 것이 주를 이룬다. 따라서 앞에서 언급한 대로 일정하게 주어진 시간의 범위안에서 발생된 이벤트들의 상태 변화를 측정하는데 유용하게 사용되고, 침입탐지시스템에서 사용하는 시스템 호출 관련 척도의 변화를 주기위해 시스템 호출에 직접적으로 영향을 미치는 침입 프로그램에 변형을 주었다.

3.4 변형 침입패턴 생성 방법

3.4.1 전처리

u2r 침입의 경우 프로그램을 크게 3부분으로 나눌 수 있다. 선언부에 각 변수와 쉘을 실행시키는 쉘코드의 선언으로 구성되고, 시스템 메모리 관련 연산을 하는 부분과 최종적으로 운영체제의 취약부분을 실행하는 부분으로 나눌 수 있고, 작은 모듈로 나눌 수 있는 부분이 메모리 관련 연산을 수행하는 부분과 쉘을 실행하는 부분이다.

모든 침입 프로그램은 프로그램을 만든 해커들에 의해 최적화된 코드를 유지한다. 이들 프로그램에 변형을 위해 선행적으로 최소단위의 의미 모듈을 나누고, 이들 모듈간의 의존성을 파악하여 변형 가능한 모듈과 그렇지 않은 모듈을 나누어야 한다. 의존성 처리를 위해 모듈간의 위치에 따른 처리 조건을 명시하는 도메인 룰(Domain Rule)을 사용하였다[31]. 도메인 룰은 Action(모듈실행내용), Pre(선행처리모듈), Post(모듈처리결과)로 구성되며 한개의 의미모듈을 나타낸다. 이때 룰은 프로그램을 변형시키기 위한 최소단위인 의미모듈이 된다.

표 16. Domain Rule를 적용한 xlock 침입 프로그램

	Action(모듈실행내용)	Pre(선행처리모듈)	Post(모듈처리결과)
Step 1	arg(variable)	not	not
Step 2	env(variable)	not	not
Step 3	retaddr(addr)	arg, env	address(ret)
Step 4	memset(sizeof_buf)	not	overbuf size
Step 5	memcpy(overbuf)	memset	overbuf mem
Step 6	exec(shell)	memcpy	root_access(user)

이러한 특성을 기반으로 각각의 침입코드 속에서 모듈들이 이동할 위치를 찾기 위해 u2r 침입의 의미모듈 내용을 분석하면 그림 6과 같이 도식화가 된다. 논문에서 사용되는 u2r 침입의 내부 모듈 중 위치 이동을 할 수 있는 모듈을 4개의 부류로 나누었다. 각 부류의 내용은 침입 실행시 환경변수의 변화를 주어 수행되

는 내용을 나타내는 Environment Set과 침입을 하기위해 해당 시스템의 정보를 습득하는 Platform Info, 해킹을 시도할 때 쓰이는 Shellcode를 넣을 수 있는 주소와 Shellcode, 그리고 Shell의 실행 부분으로 나누어진다. 또한 각 군집은 내부에 위치 이동이 가능한 의미모듈 단위의 항목으로 나뉘어 진다. 각 항목의 정의를 내리면 표 17과 같이 나타낸다.

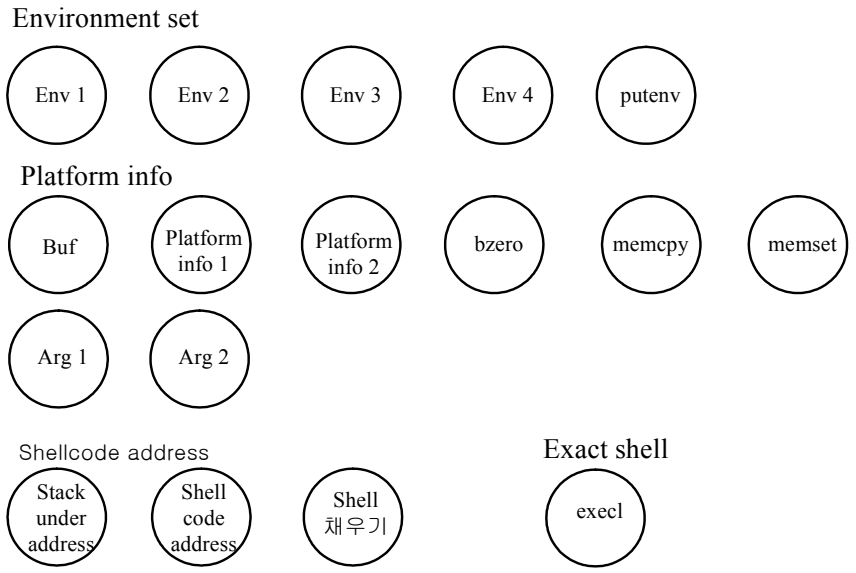

그림 6. U2R 침입 구성 모듈

표 17. 의미모듈 세부설명

의미모듈	설 명
env	침입 코드의 환경 변수를 저장함. 번호에 따라 변수 저장이 다름.
putenv	환경 변수 내용을 넣음.
bof	버퍼의 크기확인 또는 위치 찾기.
platform info	시스템 이용 정보 획득 또는 저장. 번호에 따라 내용 다름.
bzero	버퍼의 내용을 필요 없는 값으로 채움.
memset	메모리의 내용을 셋팅함.
memcpy	메모리에 내용을 복사함.
arg	침입 코드가 사용하는 를 저장. 번호에 따라 내용 다름.
shell under addr	불법적인 쉘이 실행 되는 스택의 아래 번지
shell code addr	불법적인 쉘이 실행되는 스택의 번지
shell put	침입을 위한 쉘의 삽입
execl	침입 실행

3.4.2 Rulebase를 이용한 의미모듈 위치변환

의미모듈의 위치 이동을 통한 침입변형은 각각의 모듈이 이동할 수 있는 위치의 다양성이 존재한다. 특히 모듈이 이동할 수 있는 위치는 매우 많은 반면, 완전한 코드로 완성 될 수 있는 위치는 매우 적게 나타난다. 따라서 침입변형의 자동화를 위해서는 여러 종류의 u2r 침입을 변형하면서 동시에 위치 이동의 다양성에 따른 적절한 위치이동을 보장해야 한다.

Rulebase는 찾으려는 해영역이 작을 때 간단한 Rule Set을 이용하여 해를 찾는 방법으로 침입코드의 모듈 이동위치 다양성 중 적은수의 실행 위치 찾기에 알맞은 방법이다. 새로운 침입코드를 자동으로 생성하기 위한 Rule Set은 표 18과 같은 특성을 만족해야 한다.

표 18. 침입변형 Rule Set의 특성

특성	설명
범용성	생성된 Rule은 모든 u2r 침입에 적용되어야 한다.
자동성	침입변형이 자동으로 되어야 한다.
유용성	침입 코드에 적용되어 변형을 성공해야 한다.
체계성	생성된 Rule는 체계적으로 분류 가능해야 한다.

선행 처리에 의해 나뉘어진 의미모듈과 각 군집을 이용하여 rule set의 특성을 고려한 rule을 정리하면 표 19와 같다.

표 19. Rule Set의 내용

구분	Rule Set
군집내의 세부모듈 이동 Rule	Rule1) IF Module Type = env and Module Type = arg THEN Module Transfer
	Rule2) IF Module Type = memset and Module Type = memcpy THEN Module Transfer
	Rule3) IF Module Type = arg and Module Type = platform info(1,2...) THEN Module Transfer
	Rule4) IF (Module Type = memset or Module Type = memcpy) and Module Type = buf THEN Module Transfer
군집간 이동 Rule	Rule5) IF Module Group Type = environment set and Module Group Type = platform info THEN Module Group Transfer
	Rule6) IF Module Group Type = platform info and Module Group Type = shellcode address THEN Module Group Transfer
	Rule7) IF Module Group Type = environment set and Module Group Type = shellcode address THEN Module Group Transfer

침입 Rule Set은 크게 두부류로 나뉜다. 군집내의 세부모듈 이동 Rule은 의미모듈의 4가지 군집속에서 모듈간의 이동을 할 수 있는 규칙을 나타낸다. 군집간 이동 Rule은 4가지 군집간 전체 이동을 시키는 규칙이다. 여러 부분으로 나눌 수 있는 모듈셋은 침입 코드에서 특정한 일을 수행하는 한 부분이다. 생성된 rule을 이용하여 의미모듈의 군집간 이동과 군집내 이동을 적용하게 되는데, 침입코드의 변형 성공을 높이기 위해 Domain Rule를 따라 변형을 시도한다. Rule Set을 통해 각 의미 모듈의 위치를 변환하기 위해 도메인 룰이 적용되어 모듈간 의존성에 따라 위치 조정이 가능한 모듈의 위치 변환을 한다. Domain Rule에 따라 나뉘어진 의미 모듈의 위치 변환은 Rule Set이 적용되는 의미모듈 군집을 기준으로 그림 7에서 나타내는 알고리즘을 이용하여 위치를 변환한다.

```
for (의미 모듈의 처음부터 마지막까지)
{
 if 이번 블록이 앞 블록에 의존적이면
    앞으로 이동 금지
 else 의존적이지 않으면
    앞으로 이동
}
```

그림 7. 모듈 위치변화 알고리즘

또한 Rule 적용의 범용성을 위해 침입변형 선행처리시 침입코드의 유형을 두가지로 분류하고, 이 분류에 따라 각각 다음과 같이 침입변형 Rule을 적용한다. 우선 침입코드는 main() 함수속에 모든 프로그램이 최적화 된 상태와 main() 함수와 기타 다른 함수들의 구성으로 된 프로그램으로 나누고, main() 함수로 이루어진 침입코드는 군집내 세부모듈 이동 Rule과 군집간 이동 Rule 모두를 적용하여 침입변형을 한다. 또한 main() 함수와 기타 다른 함수들로 구성된 프로그램은 서브 함수별 최적화 상태를 보이므로 memset, memcpy, shellcode 관련 모듈의 위치 이동이 불가능하다. 따라서 적용 Rule의 제한이 있기 때문에 rule1, 3, 5를 이용하여 침입모듈의 위치이동을 통한 침입변형을 한다.

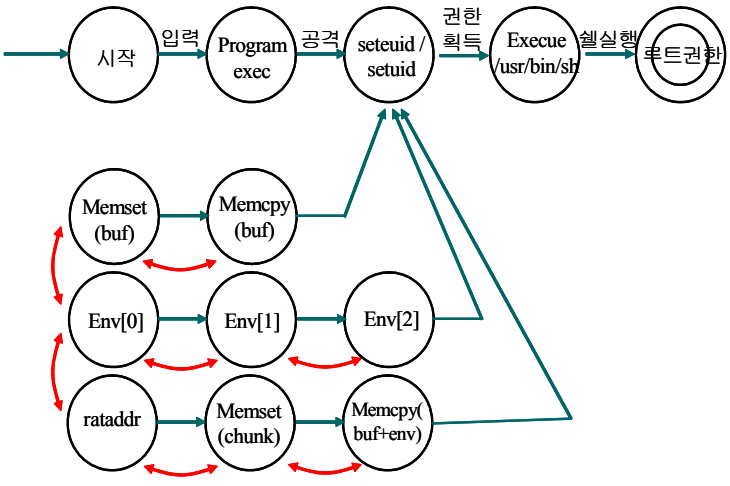

그림 8. 의미모듈의 위치이동

위치 변환 수행모습은 그림 8과 같이 표현할 수 있고, 이 결과로 생성된 침입을 실행하면 침입탐지 시스템의 척도로 사용되는 시스템 호출의 생성 순서가 바뀌어서 침입을 실행한다.

3.4.3 모듈규칙을 이용한 더미코드 삽입

대부분의 침입은 일련의 침입 행위가 계속 발생되지 않고, 시스템의 정상 행위를 수행하면서 특정 행위를 유발하여 침입을 하게 된다. 그림 9는 불법적 파일접근 침입을 나타내는 시나리오로서 Step1에서 Step6까지는 시스템의 정상적인 수행을 하고, Step7과 Step8이 실행되면서 파일에 대한 사용 권한을 획득하게 된다.

Step 1: *touch(bad_guy, guy_file)*
Step 2: *black(bad_guy, ppt)*
Step 3: *lpr-s(bad_guy, ppt, guy_file)*
Step 4: *remove(bad_guy, guy_file)*
Step 5: *ln-s(bad)guy, guy_file, secret_file)*
Step 6: *unblock(bad)guy, ppt)*
Step 7: *print-process(ppt, guy_file)*
Step 8: *get-file(bad_guy, secret_file)*

그림 9. 불법 파일 접근 시나리오

그러나 Step7과 Step8 두 가지 행위만으로는 침입을 수행할 수 없다. 불법 파일 접근 침입의 예제와 같이 많은 침입들이 실행될 때 정상적인 행위와 특정 행위들이 모여 침입을 수행한다. 정상행위를 수행하는 Step1과 Step6 사이에 파일 권한 획득과 관계없는 더미 코드를 삽입하면 시스템은 침입을 수행하면서 다양한 더미 척도를 생성한다.

기존 침입코드에 더미코드를 삽입하기 위해서는 더미코드로 사용될 시스템 호출을 정의해야 한다. 호스트 기반 침입탐지시스템 기법 중 통계적 기법을 이용한 침입탐지시스템에서는 침입 발생시 나타나는 시스템 호출, 사용 명령, 시스템 자원 사용량 등의 이용 빈도를 이용하여 침입을 판단한다[32]. 특히 시스템 호출의

경우 통계적 기법을 이용한 침입탐지시스템은 정상 시스템 호출 패턴과 침입시 나타나는 비정상적 패턴의 통계량을 비교하게 된다[33]. 그러므로 더미코드 삽입은 시스템에서 정상적으로 사용되는 시스템 호출을 위주로 삽입 되어야 한다.

본 연구에서는 시스템에서 일반적으로 사용하는 시스템 호출을 더미코드에 삽입하기 위해 모듈 규칙을 적용하였다. 모듈 규칙은 솔라리스 운영체제에서 일반적으로 사용되는 시스템 명령과 침입시 자주 나타나는 시스템 명령을 구분한 것으로 표 20과 같이 나뉘어 있다[34].

표 20. 시스템 명령어 구분

구분	시스템 명령
일반적 사용 명령군	open, write, close, exit, fork, read, date, cls
침입시 사용 명령군	seteuid, setuid, setpgrp, memcpy, memset rataddr

나뉘어진 시스템 명령중 침입코드 삽입에 이용되는 부분은 정상상태에서 사용되는 시스템 명령군을 택했다. 삽입되는 시스템 명령 종류의 중요함과 마찬가지로 추가되는 횟수 또한 중요하다. 시스템 명령이 더미코드에 삽입되는 횟수는 각 침입코드에 따른 침입 발생시 나타나는 총 시스템 호출의 길이 이용하여 적용하였다. 삽입되는 더미 시스템 명령의 횟수는 가장 길게 나타나는 시스템 호출 정보와 가장 짧게 나타나는 시스템 호출 정보를 절충하여 5회부터 10회까지로 제한을 두었다.

여러 모듈로 나누어진 u2r 침입은 모듈 사이에 침입코드와 무관한 더미코드를 삽입하여 다양한 시스템 호출을 발생하며 새로운 침입을 만든다. 그림 10은 침입코드와 무관한 더미코드 중 fork() 시스템 명령을 이용하여 삽입하는 과정을 나타낸다.

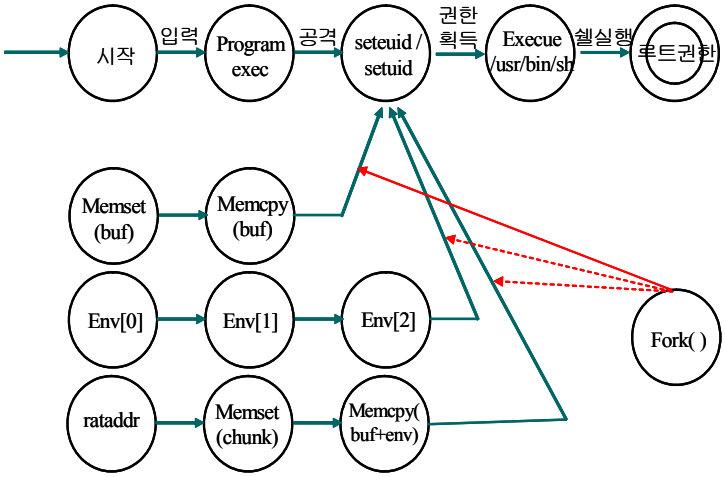

그림 10. fork() 시스템 호출을 이용한 더미코드 삽입

3.5 침입코드 변환률

침입 변환에 따른 침입 변환률은 수식(1)과 같이 나타낼 수 있다. 침입 변환률은 실제 침입 변형수를 총 침입변환 가능수로 나눈 것을 100으로 곱한 것으로 각 침입별 또는 모든 침입을 통합하여 계산 가능하다.

$$\text{침입변환률}(\%) = \frac{\text{침입변형개수}}{\text{총 침입변환 가능개수}} \times 100 \qquad (1)$$

이때 총 침입변환 가능수는 침입 코드의 의미모듈이 위치이동을 통해서 나타낼 수 있는 모든 개수를 의미하며, 만약 n개의 모듈로 나누어진 침입코드가 있다고 가정하면 $n!$개의 위치변형 가능 위치를 만들 수 있다.

수식(2)는 자동 변형의 유용성을 나타내기 위해 상대적 자동변형 비율을 나타낼 수 있다. 상대적 자동변형 비율은 각 침입에 대하여 계산한다. 자동변형 비율이 높을수록 적용된 침입의 변형이 성공적으로 되었다는 것을 나타낸다.

$$\text{상대적 자동변형 비율}(\%) = \frac{\text{자동 침입변형률}}{\text{수동 침입변형률}} \times 100 \qquad (2)$$

위 수식(2)를 그래프로 나타내면 비교본이 되는 그래프는 항상 100%가 되고 비교대상은 상대적 크기를 %로 표시하고, 100%에 가까울수록 좋은 비율을 나타낸다.

제 4장 성능평가 및 결과

본 연구에서는 알려진 침입을 이용하여 변형된 새로운 침입을 생성하였다. 변형을 하기위해 침입을 정확히 분류하고 분류된 침입을 기반으로 호스트 기반 침입에 사용되는 침입코드를 변형하였다. 실험을 위해 가장 널리 사용되는 OS중 하나인 솔라리스 2.7과 2.8 버전 기반의 u2r 침입 13 종류를 기반으로 침입변형을 하였다. 솔라리스 2.7에서는 버퍼오버플로우 7가지와 구성설정 오류를 이용한 침입 2가지를, 솔라리스 2.8버전에서는 버퍼오버플로우 4가지를 사용하였다.

4.1 침입코드 변형에 따른 침입탐지 척도변화

연구의 주된 목적은 침입코드의 변화를 이용하여 최종적으로 호스트 기반 침입탐지시스템에서 사용하는 침입척도인 시스템 호출 정보의 변화를 주어 새로운 침입을 만들려는 것이다. 시스템 호출 정보의 추출을 위해 Sun사의 Basic Security Module(BSM)을 이용하여 시스템 호출의 변화를 추적하였다. 표 21은 정상침입을 실행했을 때 시스템 호출 정보와 제안하는 두가지 방법으로 변형을 시도한 새로운 침입 패턴의 시스템 호출 정보를 비교한 것이다. 3가지 침입 모두 호스트에서 root 권한을 획득하는데 성공하지만, 의미모듈 위치변환을 통한 침입 변형은 시스템 호출 seteuid, open, sysinfo, execve의 순서가 바뀌는 것을 보인다. 또한 더미코드를 삽입을 통한 침입 변형은 더미 시스템 호출 fork, open, date, read, close가 첨가되어 실행되는 것을 볼 수 있다. 따라서 침입코드의 변형을 통해 시스템 호출 정보의 변화를 줄 수 있고, 호스트 기반 침입탐지시스템에서 중요하게 사용하는 침입탐지 척도의 변화를 주면서 침입을 실행하는 새로운 침입을 만들 수 있다. 또한 앞에서 설명한 침입변형 특성 중 침입변환의 유용성에 대한 조건을 만족시킨다.

표 21. 정상침입과 코드변형에 따른 시스템 호출 척도 변화

정상 침입		의미모듈 위치변환 후 침입		더미코드 삽입 후 침입	
시스템 호출	내용	시스템 호출	내용	시스템 호출	내용
fork		fork		fork	
setpgrp		setpgrp		setpgrp	
setpgrp		setpgrp		**fork**	**fork 삽입**
				setpgrp	
execve	/exploit/xlock 실행	execve	/exploit/xlock 실행	execve	/exploit/xlock 실행
sysinfo		**seteuid**	**위치이동**	sysinfo	
execve	/usr/openwin/bin/xlock 실행	**open**	**위치이동**	execve	/usr/openwin/bin/xlock 실행
open		**sysinfo**	**위치이동**	open	
seteuid		**execve**	**/usr/openwin/bin/xlock 실행**	**open**	**open 삽입**
				seteuid	
old setuid		old setuid		old setuid	
execve	root shell 실행	execve	root shell 실행	execve	root shell 실행
setpgrp		setpgrp		**date**	**date 삽입**
				setpgrp	
setpgrp		setpgrp		setpgrp	
setpgrp	부모쉘에 대한 설정변경	setpgrp	부모쉘에 대한 설정변경	setpgrp	부모쉘에 대한 설정변경
setpgrp		setpgrp		**read**	**read 삽입**
				setpgrp	
setpgrp		setpgrp		**close**	**close 삽입**
				setpgrp	
fork	root shell에서 date 파일 실행	fork	root shell에서 date 파일 실행	fork	root shell에서 date 파일 실행
execve		execve		execve	
exit		exit		exit	

4.2 위치변환 Rule Set에 따른 변환률

Rulebase를 이용한 모듈 군집내 위치변형률과 모듈 군집간 위치변형률을 살펴보면 표 22와 같이 나타난다. 그룹내 이동 성공률은 의미모듈 단위의 이동을 지원하는 Rule1, 2, 3, 4번을 이용하여 위치이동을 성공률을 나타낸 것이고, 그룹간 이동 성공률은 의미모듈 군집으로 형성된 Environment Set, Platform Info,

Shellcode 단위의 이동을 지원하는 Rule5, 6, 7번을 이용한 위치이동 성공률을 나타낸다. 그룹으로 묶인 의미모듈은 그렇지 않은 경우보다 더욱 안정된 모듈 의존성을 가지게 된다. 그러나 이런 의존성이 의미모듈 개개의 자유로운 이동을 저해함으로서 변형 성공률을 낮추는 결과를 만든다. 표 22는 수동변환과 자동변환 모두 그룹내 변형 성공률이 높음을 보여주고 있다.

표 22. 의미모듈의 군집내 위치이동과 군집간 위치이동 성공률

구분	수동 변형 성공률	자동 변형 성공률
그룹내 변형	88.99%	94.37%.
그룹간 변형	11.01%	5.63%

Rusebase를 이용한 의미모듈 위치변환 방법에서 rule set의 적용은 실험을 통해 그룹내 변형이 그룹간 변형보다 효과적임을 알 수 있다. 이것은 의미모듈간 의존성과 관계있는 결과로 그룹단위의 위치 이동을 하는 그룹간 변형이 의존성에 더 강한 결속을 보여 오히려 의미모듈의 이동 침입 변형률이 떨어지게 된다.

4.3 더미코드 삽입 방법에 따른 성공률

더미코드 삽입 방법은 정상적으로 시스템을 사용할 때 주로 발생하는 시스템 명령을 더미로 삽입하여 시스템 호출의 길이를 변형시켰다. 이때 더미로 삽입되는 시스템 호출의 길이를 5~10회로 한정하였다. 이 숫자는 정상적인 침입을 실행할 때 시스템 호출의 최대 길이와 최소 길이를 절충하였다.

표 23의 실험결과 더미코드 삽입은 모두 변형에 성공하였다. 그러나 더미코드 삽입방법은 실제 침입탐지시스템에서 매우 높은 False Positive 오류율을 내포하고 있다. 더미 삽입 횟수에 따른 침입변형 오류율을 알기 위해서는 침입탐지평가 도구를 이용하여 실제 침입탐지시스템 평가를 실행해야 한다.

표 23. 더미모듈 삽입 횟수에 따른 변형 성공률

더미 삽입 횟수	침입 변형 성공률
5	100%
6	100%
7	100%
8	100%
9	100%
10	100%

4.4 운영체제 버전에 따른 성공률

각 침입의 자동 변형 성공률은 사용되는 운영체제 버전에 따라 다르게 나타날 수 있다. 솔라리스 2.7과 2.8에 대한 변형 성공률은 식 (3)으로 나타낼 수 있고, 운영체제별 변환률과 침입 프로그램 복잡도는 표 24와 같이 나타낼 수 있다.

표 24. 운영체제 버전에 따른 침입변형 성공률

구분	수동 변형 성공률	자동 변형 성공률	프로그램 평균 라인 복잡도
Solaris Intel 2.8	13.60%	10.87%	63.25
Solaris Sparc 2.7	2.81%	2.10%	90.55
버전별 변형 성공률 차이 (2.8변형률 - 2.7 변형률)	10.79%	8.77%	

$$\text{운영체제별 변환률} = \frac{\text{운영체제에 따른 침입 성공률 합계}}{\text{운영체제별 침입 사용 개수}} \times 100 \qquad (3)$$

프로그램의 복잡도를 표현하는 특징은 프로그램의 특성마다 다르게 나타낼 수 있다. 침입프로그램의 경우 시스템 최적화 유무, 실행 소요시간, 실시간 실행 유무 등의 특성을 대변한 복잡도를 표현해야 한다. 이와 유사한 복잡성을 측정받는 프로그램으로는 군사용 프로그램이 있다. 군사용 프로그램이 복잡도를 측정받기 위해 사용하는 척도로는 프로그램 내부의 클래스 구성인자 Method의 개수, 프로그램 코드 라인의 수, 클래스당 토큰과 심볼의 개수 등을 정량적으로 분석을 한다

[35]. 실험에 사용된 13개의 침입 프로그램 또한 시스템의 정보를 활용하기 위해 시스템에 최적화된 상태를 유지하기 때문에 코드 라인수를 복잡도를 계산하는 척도로 활용하였다.

솔라리스 버전에 따른 침입변환은 표 24에서 나타낸 것과 같이 수동 변환에서 평균 10.79%의 변환 차이를 보이고, 자동 변환에서는 8.77%의 차이를 나타낸다. 이런 결과는 솔라리스의 버전별 침입코드 특성과도 관계가 있다. 솔라리스 2.7의 코드의 평균길이는 2.8버전의 코드보다 상대적으로 길게 구성되었지만 시스템에 더욱 최적화 되어있다. 최적화된 코드 일수록 의미모듈의 위치변환은 더욱 어렵게 된다. 반면 코드의 길이가 조금 더 짧더라도 시스템에 덜 최적화된 2.8 기반의 침입은 더 많은 이동위치를 만들어 낼 수 있다. 이것이 솔라리스 2.8침입이 2.7 침입보다 더 낮은 변형률을 보일 수 있는 이유이다.

4.5 Rulebase 위치변형 방법의 타당성

모듈 위치변환 사용방법인 Rulebase에 대한 적절성을 평가하기 위해 표 25와 같이 각 침입에 대하여 수동변형과 자동변형에 대한 변형 성공률을 비교 하였다. 표 22에서 나타나는 것과 같이 각 침입별 변환 성공률의 차이는 0% ~ 5.45%이다. 수동변형과 자동변형 시스템에 따른 최종 침입 변형율은 평균 6.1%와 4.8%로 나타나 변환 자동성에 대한 성공도가 높다는 것을 확인 할 수 있었다.

그림 11은 모든 수동변환과 자동변환의 평균 비교와 운영체제 버전별, 침입코드 종류별 침입변형 성공률의 평균을 비교해 나타냈다. 그래프는 비교를 편하게 하기 위해 수동변형의 비율을 100%로 표시하고 자동변형 비율을 상대적으로 표시하는 수식(2)의 상대적 자동변형 비율로 나타냈다. 운영체제별 변형비율 그래프에서는 솔라리스 2.8과 2.7에서 각각 79.93%와 74.73%로 비슷한 변환비율을 보이고 있으며, 침입 종류별 그래프에서는 버퍼오버플로우 침입이 77.92%와 구성설정 오류 침입이 79.71%로 나타났다. 이러한 자료를 바탕으로 Rulebase를 이용한 침입코드 위치변형은 운영체제 버전 및 침입종류에 상관없이 상대적 자동변환 비율이 평균 78.69%의 변환률을 보인다. 따라서 수동 변환에서 총 침입변환 개수가 작기 때문에 Rulebase 방법이 위치변환을 위한 적절한 방법임을 알 수 있다.

표 25. 침입코드의 수동변환과 자동변환에 따른 침입변형 성공률

구분	침입코드명	수동변형		자동변형	
		침입변형수/변환가능수	변형 성공률	침입변형수/변환가능수	변형 성공률
solaris intel 2.8	kcms_configure_overflow.c	2/56	3.57%	2/56	3.57%
	whodo.c	27/121	22.31%	22/121	18.18%
	xlock.c	11/225	4.88%	8/225	3.56%
	mailx.c	26/110	23.63%	20/110	18.18%
solaris sparc 2.7	xlock.c	10/225	4.44%	6/225	2.67%
	xsun.c	1/81	1.23%	1/81	1.23%
	ex_lobc.c	4/121	3.30%	4/121	3.30%
	dtaction.c	7/121	5.78%	4/121	3.31%
	dtprintinfo.c	5/100	5%	4/100	4%
	ex_admintool.c	5/361	1.38%	5/361	1.38%
	lpset.c	2/121	1.65%	1/121	0.82%
	3lc_messages_passwd_stack.c	5/361	1.38%	5/361	1.38%
	eject-fmt.c	4/361	1.10%	3/361	0.83%

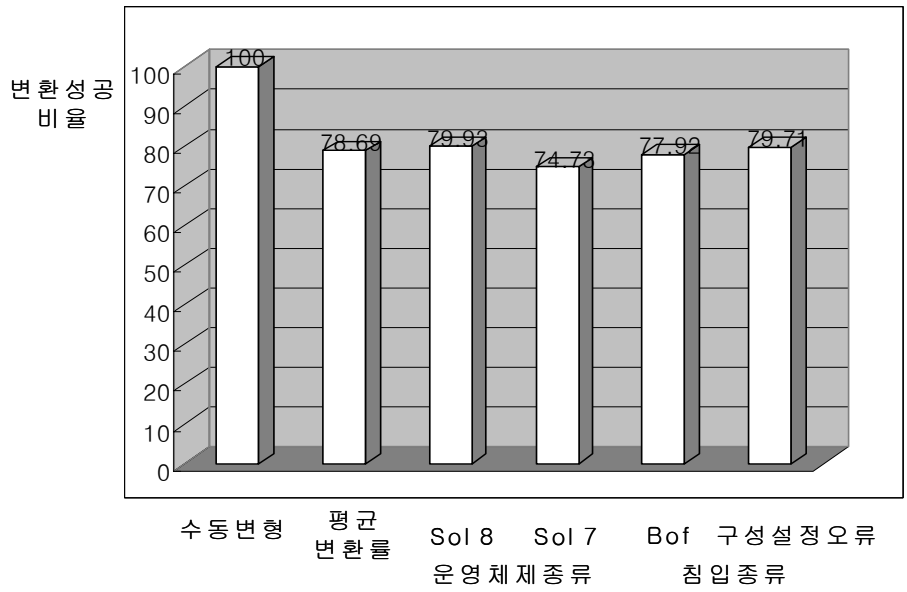

그림 11. 수동변형에 대한 상대적 변환 성공비율

제 5장 결론 및 향후연구

본 논문에서는 기존 침입분류, 호스트 기반 침입탐지 척도 및 침입탐지시스템 평가 연구를 기반으로 침입패턴 자동생성 방법을 제안하였으며, 사용방법은 Rulebase를 이용한 의미모듈 위치변환과 모듈 규칙을 이용한 더미코드 삽입을 이용하였다. 실험을 통하여 사용자가 모든 변환을 수동으로 하던 방법을 이용할 경우 모든 작업을 해야한다는 단점을 극복하고 좀더 자동화된 방법을 찾기위해 침입패턴 자동생성기를 제안하였다.

전처리 부분은 침입코드를 분석하기 위해 코드를 이동 할 수 있는 최소 단위인 의미모듈로 나누고 나누어진 의미모듈에 도메인 룰을 적용하여 모듈간 의존성을 처리한다. 또한 위치이동 Rule의 적용 가능한 범위를 정하기 위해 침입코드의 형태(main 함수로만 구성, main()과 서브 함수로 구성)로 나누게 된다. 침입코드의 의미모듈은 크게 4가지로 구분되며, 각각의 내용은 다음과 같이 나누었다. 침입 실행시 환경변수의 변화를 주어 수행되는 내용을 나타내는 Environment Set, 침입을 하기 위해 해당 시스템의 정보를 습득하는 Platform Info, 해킹을 시도할 때 쓰이는 Shellcode를 넣을 수 있는 주소와 Shellcode, 그리고 Shell의 실행 부분으로 구분할 수 있다. 의미모듈 분류를 통해 각 의미모듈이 나타내는 범주를 설명하였고, 이렇게 생성된 침입코드의 의미모듈을 가지고 제안하는 두가지 방법에 의해 침입변형을 한다.

Rusebase를 이용한 의미모듈 위치변환 방법에서 Rule Set은 크게 의미모듈 그룹내 변형과 그룹간 위치변형을 하는데, 실험을 통해 그룹내 변형이 그룹간 변형보다 효과적임을 보였다. 이것은 의미모듈간 의존성과 관계있는 결과로 그룹으로 위치 이동을 하는 그룹간 변형이 의존성에 더 강한 결속을 보여 오히려 의미모듈의 이동 침입 변형률이 떨어지게 된다. 모듈규칙을 이용한 더미코드 삽입에서는 통계적 기법을 적용한 침입탐지시스템의 적용을 고려하여 더미코드에 사용되는 시스템 명령의 종류와 횟수에 제한을 두었다. 실험결과 더미코드 삽입은 모두 변형에 성공

하였다.

또한 실험에 사용된 솔라리스 버전에 따른 변형률은 2.8기반 침입이 2.7보다 더 다양한 변형을 할 수 있는 결과를 나타낸다. 이것은 침입 대상이 되는 OS 버전이 높아짐에 따라 같은 이름의 침입이라도 프로그램 복잡도가 더욱 높아져서 위치변형 가능성이 더 높다는 것을 알 수 있다. 더블어 침입종류 구분이 되는 버퍼오버플로우 침입이 구성설정 오류 침입 보다 더욱 유연하게 위치이동을 통한 침입변형률이 좋은 것을 알 수 있다.

마지막으로 수동변형과 자동변형 시스템에 따른 최종 침입 변형율은 6.1%와 4.8%로 나타나 변환 자동성에 대한 성공도가 높다는 것을 확인 할 수 있었다. 따라서 수동 변환에서 총 변환 가능한 개수가 작기 때문에 Rulebase 방법이 위치변환을 위한 적정한 방법임을 판정할 수 있다. 하지만 이런 변환 방법을 모두 자동화 하기 위해서는 전처리 과정으로 입력된 기본 침입패턴을 의미모듈별 도메인 룰을 정확히 나눌 수 있는 최적의 파서(Parser)를 적용시켜야 하며, 또한 침입의 종류에 따른 침입특성이 나타나므로 특성에 따른 다양한 파서가 필요 할 것이다. 그 외에 보다 다양한 침입패턴에 대한 변형 확장과 침입변형 평가가 이루어져야 할 것이다.

참고문헌

[1] 한국정보보호진흥원, “2001년 정보화역기능 실태조사”, 2001.

[2] 한국정보보호진흥원, “99 국내외 해킹현황 분석”, http://www.certcc.or.kr/statistics /hack/1999/99-hack.htm.

[3] Dorothy. E. Dening, "An Intrusion Detection Model," *IEEE Transactions on Software Engineering*, Vol.13, No.2, pp.222-232, February 1997.

[4] P. Porras, "The Common Intursion Detection Framework Architecture," *CIDF Working Group Document*, September 1999.

[5] Internet Security systems, Introduction to RealSecure version 3.0, January 1999.

[6] V. Paxson, “Bro : A System for Detection Network Intruders in Real-Time,” *In Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, January 1998.

[7] CISCO, *NetRanger Intrusion Detection System*, Technical Information, April 1999.

[8] A. K. Ghosh, J. Wanken, and F. Charron. "Detecting Anomalous and Unknown Intrusions Against Programs," *In Proceedings of the Annual Computer Security Application Conference(ACSAC'98)*, Scottsdale, AZ, December 1998.

[9] H. S. Javitz and A. Valdes "The NIDES Statistical component Description and Justification," *Technical report, SRI International,* March 1994.

[10] C. Ko, M. Ruschitzka, and K. Levitt, "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach," *In Proceedings of the 1997 IEEE*

Symposium on Security and Privacy, pp. 175-187, 1997.

[11] Terran. Lane, "A Survey of Intursion Detection Techniques," *Computer & Security*, vol. 12, no. 4, June 1993

[12] Nicholas J. Puketz et al, "A Methodology for Testing Intrusion Detection Systems," *IEEE Transactions on Software Engineering*, Vol.22, No.10, pp.719-729, October, 1996.

[13] N. Puketza, M. Chung, Ronald A. Olsson and Biswanath Mukherjee, "A Software Platform for Testing Intrusion Detection Systems," *IEEE SOFTWARE*, Vol. 14, No. 5, pp.43-51 , September/October 1997.

[14] John McHugh, "The 1998 Lincoln Laboratory IDS Evaluation a Critique," *RAID 2000,* pp.145-161, October, 2000.

[15] Richard P. Lippmann, David J. Fried, Isaac Graf, and Joshua W. Haines et al, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation," *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, Vol. 2, pp.1012-1027, January 2000.

[16] DARPA Intursion Detection Evaluation, http://www.ll.mit.edu/IST/ideval

[17] http://www.icsalabs.com/html/communities/ids/certifications.html

[18] 한국정보보호진흥원, “침입탐지시스템 평가기준 해설서”, October 2001.

[19] 한국정보보호진흥원, “국가기관용 침입탐지시스템 보호프로파일 V1.0", December 2002.

[20] http://www.tcpdump.org/

[21] Don Libes, "Exploring Expert", O'reilly, December 1994.

[22] CERTCC-KR, Korea Information Security Agency, http://www.certcc.or.kr

[23] T. Ball, D. Hoffman, F. Ruskey, R. Webber, Lee White, “State Generation and Automated Class Testing,” *Software Testing, Verification & Reliability,* April 2000.

[24] David Larochelle, David Evans, "Statically Detecting Likely Buffer OverflowVulnerabilities," *USENIX Security Symposium*, pp.177-190, August 2001.

[25] Felix Lau, Stuart H. Rubin, Michael H. Smith, and Ljiljana Trajovic, "Distrubuted Denial of Service Attacks," *IEEE International Conference on Systems, Man, and Cybernetics*, pp.2275-2280, October 2000.

[26] J. Choy , S.-B. Cho, "Intrusion detection by combining multiple hidden Markov models," *Lecture Note in Artificial Intelligence*, vol.886 pp.829, 2000.

[27] H.-J. Park, S.-B. Cho, "Efficient anomaly detection by modeling privilege flows using hidden Markov model," *Computers & Security*, vol. 22, no. 1, pp. 45-55, Jan 2003.

[28] John Smith, Mark Jones, “Improving Host Security with System Call,” *CITI Technical Report 02-3*, November 2002

[29] Sun microsystems, Inc., 2550 Garcia Avenue, Mountain view, CA, USA. *SunSHIELD Basic Security Module Guide*, November 1995.

[30] B. A. kuperman and E. H. Spafford. "Generation of application level audit data via library interposition," October 1998.

[31] F. Cuppens, F. Autrel, A. Mi`ege, and S. Benferhat, "Recognizing malicious intention in an intrusion detection process," *Second International Conference on Hybrid Intelligent Systems (HIS 2002)*, December 2002

[32] http://www.sans.org/resources/idfaq/statistic_ids.php, “Statistical based approach to Intrusion Detection ”

[33] Lunt, Teresa F. "A Survey of Intrusion Detection Techniques," *Computers and Security,* vol 12, pp. 405-418, June. 1993

[34] Wenke Lee, Salvatore J. Stolfo, "Learning Patterns from Unix Process Execution Traces for Intrusion Detection," *Proceedings of the AAAI97*

workshop on AI methods in Fraud and risk management, pp.50-56,1997.

[35] D. Batory, C. Johnson, B. MacDonald, and D. Van Heeder, "Achieving Extensibility Through Product-Lines and Domain-Specific Languages : A Case Study," *ACM Transactions on Software Engineering and Methodology (TOSEM) archive 2002*, Volume 11 , pp.191-214, April 2002.

ABSTRACT

A study on the automatic generation of attack patterns for evaluation of intrusion detection system

Young-Ju Noh
Dept. of Computer Science
The Graduate School
Yonsei University

The frequency of illegal intrusions on computer systems are increasing and variety of researches and development on intrusion detection systems have been conducted. Intrusion detection system detects illegal usage, abuse by an insider as well as unauthorized extraction and modification of critical information by an outsider where the intrusion itself is determined by several factors such as key words created by the user on each operating systems, system calls, duration of system use, and analysis of network packets. The global trend of intrusion detection technology is to develop an optimal intrusion detection method which can cope with variable intrusion situations and some of these efforts have been converted to manufactured products. Nowaday, many researches and investigation focus on variety of evaluation and optimization of performance issues. Evaluation of intrusion detection system in connection with evaluation of performance has a limitation. Because it makes use of already-known intrusion patterns, the evaluation on new type of intrusions cannot be

performed, therefore to perform an evaluation on new type of intrusions, researches should be conducted on intrusion pattern generation. Therefore this paper proposes a method of generating intrusion patterns from already-known intrusion code for the purpose of evaluating host-based intrusion detection system.

The most typical method of generating a new intrusion pattern is to alter intrusion criterion where system call information is the utmost important criterion in host-based intrusion detection. To apply alteration to existing intrusion code, studies and analysis on intrusion codes are required. In this paper, intrusion code is divided into smallest functional unit of four categories called syntax modules and inter-module mutual dependency is perceived based on domain rule and also insertion of dummy code method is performed using location transformation method and module regulations.

In the process of location transformation, It is used rule set to change the location information of the inter-syntax module group or the four kind of syntex module groups. The rule set is applied intrusion codes using different way for general application. Intrusion codes were divided into two groups: one group where the codes are optimized with a single main() function and the other group containing a main() function with subfunctions. Also, dummy codes were inserted in intrusion codes using normal system calls that are analysised. Intrusion detection system using statistical method utilizes the frequency besides the insertion of dummy codes as an important factor of intrusion determination, so to improve the interposition process, the number of insertion is restricted from five times to ten times.

In the method of location transformation of syntax module using rulebase, rule set mainly performs inner-group transformation and inter-group transformation. And the result of intrusion variation shows that

inner-group transformation was more effective than inter-group transformation. This is due to dependency among syntax modules which reveals that inter-group transformation is more intertwined with dependency and this results in reduction of transformation of intrusion variation. In the insertion of dummy code using module rules, intrusion detection system based on statistical method were considered therefore the limitation was set to the type of system call and frequency of occurrence. In this experiment, entire dummy codes transformed successfully. Variation rate of 2.8-based intrusion was higher than 2.7-based intrusion in Solaris which implies that higher the version of the operating system gets, the higher the possibility of location transformation to occur even with identical intrusion due to the complexity of the program. Also intrusion exploiting buffer overflow which can be categorized by intrusion type has a higher intrusion variation rate than intrusion exploiting configuration error. In conclusion, final intrusion variation rate of manual generation and auto generation system were 6.1% and 4.8% which confirms that success rate of automation of generation was high. Therefore, because the number of possible transformation is small, rule base method is the appropriate method for location transformation.

Keywords : Auto Generation of Intrusion Pattern, Intrusion Variation, Location Transformation of Syntax Module, Insert of Dummy Code, Rulebase Intrusion Variation, Evaluation of Intrusion Detection System