침입탐지 평가를 위한 침입패턴 자동 생성

노영주[0] 조성배
연세대학교 컴퓨터과학과
{yjnoh[0], sbcho}@candy.yonsei.ac.kr

Automatic Generation of Patterns for Intrusion Detection System Evaluation

Young-Ju Noh[0] Sung-Bae Cho
Dept. of Computer Science, Yonsei University

요 약

최근 몇 년 동안 이루어진 네트웍 및 인터넷 시장의 발전과 더불어 빈번히 발생하는 시스템에 대한 침입으로 이를 방어하기 위한 여러 도구들이 개발되어왔다. 이러한 도구들 중 침입탐지시스템은 시스템 방어에 핵심적인 역할을 하는데, 현재까지 이를 평가하기 위한 자동화된 온라인 평가도구는 없는 실정이다. 보안관련 시장이 발달한 미국에서는 DARPA의 지원아래 관련된 연구가 진행되어 1998년부터 2000년까지 대규모의 침입탐지시스템 평가가 이루어졌으나, 이때의 평가들은 당시의 침입 수준만을 고려한 것으로 새로운 침입 환경에 대한 확장은 용이하지 않기 때문에, 급속도로 증가하는 침입 기술에 대응하기 위한 새로운 방법이 필요하다. 본 논문에서는 기존 침입코드를 이용하여 새로운 침입을 만들어 내어 침입탐지 평가도구에 적용할 수 있는 모듈 위치변환과 더미코드 삽입을 제안한다. 모듈 위치변환은 알려진 u2r코드를 모듈 단위로 나누고 나뉘어진 모듈의 위치 변환을 통해 새로운 침입을 만들어낸다. 또한 더미코드 삽입은 침입코드의 모듈 사이에 침입과 관련없이 수행되는 정상 모듈을 삽입하여 새로운 침입을 만든다. 모듈 위치변환을 통해 평균 6.1%의 침입 변환율과 더미코드 삽입을 통해 새로운 침입을 만들었다.

1. 서 론

최근 몇 년 동안 이루어진 네트웍 및 인터넷 시장의 급속한 확장으로 관련 시장의 긍정적인 팽창 뿐 아니라 네트웍 시스템에 대한 침입이나 바이러스 웜 등의 부정적인 요소들이 증가하고 있다. 따라서 이를 방어하기 위한 도구들이 개발되기 시작했는데 그 중 침입탐지시스템은 시스템 방어에 핵심적인 역할을 맡는다.

시스템 보안의 중요성이 커짐에 따라 다양한 침입탐지시스템이 등장하고 있다. 그러나 이에 대한 자동화된 온라인 평가도구는 아직 개발 단계에 머무르고 있으며, 대부분 수동적인 성능평가만이 이루어지고 있다. 국내의 침입탐지시스템에 대한 평가는 한국정보보호진흥원에서 이루어지고 있으나 현재는 초기 단계로 평가를 위한 각종 방안들을 마련하고 있는 상황이다. DARPA의 경우에는 1998년[1]과 1999년[2]에 대규모의 침입탐지시스템에 대한 평가가 이루어졌고, MIT Lincoln 연구소에서 자동화된 온라인에서의 평가에 대해 연구되었다.

현재 주된 연구의 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.

본 논문에서는 새로운 침입 환경을 구축하여 침입탐지 성능평가에 적용할 수 있는 침입패턴 자동 생성에 대한 방법을 제안한다. 침입코드를 모듈 단위로 나누어 모듈 위치를 변환하는 모듈 위치 변환과 침입코드의 모듈 사이에 침입과 관련 없는 더미 코드를 넣는 더미코드 삽입 방법을 제안한다.

2. 관련 연구

초기의 침입탐지시스템에 대한 평가는 적은 수의 시스템에 대해 단순한 트래픽 상황에서 적은 수의 은닉기능이 없는 공격만으로 평가가 이루어졌으나[5, 6], 1998년 DARPA의 1차 평가에서는 10개의 시스템, 38개의 공격, 충분한 트래픽과 일부의 은닉화된 공격을 통한 평가가 이루어졌다[1]. 표 1은 침입탐지시스템에 대한 평가의 변화추이를 나타낸다.

표 1. 과거 침입탐지시스템 평가의 특징[7]

연구자	탐지수	침입수/시스템	은닉	비고
Puketza[5,6]	2	4/1	No	자동화공격. 단순한 텔넷 트래픽
Debar[4]	3	4/1	No	자동화공격. 단순한 FTP 트래픽
Shipley[3]	10	12/4	Yes	10개의 상용제품의 비교
Durst[7]	4	19/4	Yes	1998년 DARPA 실시간 평가
Lippman[1]	10	38/4	Yes	1998년 DARPA 오프라인평가

MIT Lincoln 연구소는 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

았다. 평가 방법은 검사용 데이터에 포함된 각 침입에 대한 탐지 정보에 점수를 부과하는 방식으로 이루어졌다. 부여된 점수는 해당 ID가 침입이라고 확신하는 정도를 나타낸다. 평가 점수를 위한 정보는 1) 공격 탐지와 식별여부 검사, 2) 공격 시작 시간 탐지여부 검사, 3) 공격 목표 탐지여부 검사였다.

1998년에는 침입탐지 시스템에 대한 최초의 포괄적인 평가가 이루어졌다. 이 평가에서는 네트웍 트래픽 및 유닉스 호스트에 대한 평가가 이루어졌으며, 최초로 false positive 오류를 측정하였다. 평가 데이터에는 7주간 38종류에 대한 300번의 공격 정보가 포함되었다[6]. 1999년의 평가에서는 1998년의 평가를 확장한 것으로 윈도우 NT에 대한 평가와 새로운 공격 등을 추가했다. 또한, 새로운 공격들을 훈련 데이터에 삽입하지 않고 평가하여 해당 공격을 탐지할 수 있는지에 초점이 맞춰졌다. 평가는 탐지 결과의 ROC 곡선을 통해 탐지율과 false positive 오류율로 이루어졌다[7].

DARPA에서 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. 그러나 새로운 종류의 침입들이 증가하는 것과 관련된 침입패턴의 변형 및 적용에 대한 개발이 미흡하기 때문에 이에 대한 보완이 필요하다.

3. 자동 침입 생성

3.1 침입의 분류

침입은 컴퓨터가 사용하는 자원의 무결성, 비밀성, 이용성을 저해하는 일련의 행위이다. 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 취약점을 위주로 분류하는 방법이 있으며 해킹으로 인한 영향, 위험성 등에 의해서도 분류하는 방법이 있고, 침입 행위들이 사용되는 목적에 따라 각각의 평가 척도로서 분류될 수도 있다.

표 2. 침입 발생 지점에 따른 침입분류

침입발생지점	침입분류
시스템 내부	사용자도용(Impresonation)
	S/W보안오류(S/W Vulnerability)
	버퍼오버프로우(Buffer Overflow)
	구성설정오류(Configuration Vulnerability
	악성프로그램(Malicious Codes)
시스템 외부	프로토콜취약점(Protocol Infrastructure Error)
	서비스거부공격(Denial of Service)
	이메일 관련 공격(E-mail Vulnerability)
	취약점정보수집(Vulnerabilities Probing)

본 논문에서는 침입을 구분하기 위해 CERTCC-KR에서 권고하는 분류방법을 기준으로 시스템 내부에서 발생하는 침입과 외부에서 발생하는 침입으로 구분하여 표2와 같이 침입을 분류하였다.

3.2 침입 패턴의 속성

본 논문에서는 시스템 내부 침입자들에 의해 발생하는 u2r 침입을 기반으로 새로운 침입코드를 생성하였다. u2r 침입은 표 3과 같은 종류가 알려져 있다.

표 3. 알려진 u2r 침입코드

OS	침입분류	침입코드명	침입내용
solaris intel 2.8	버퍼오버플로우	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	버퍼오버플로우	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
	구성설정오류	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 있다. 이러한 침입프로그램을 실행시키면 각 모듈이 순차적으로 실행됨에 따라 시스템 내부에서 여러 이벤트를 발생하면서 침입을 한다. 이때 발생되는 일련의 시간정보와 침입에 따른 행위들이 침입탐지 시스템의 척도로 사용되어 침입여부를 판단한다.

3.3 침입패턴의 생성

기존 u2r 침입 코드를 기반으로 새로운 침입을 생성하기 위해서 코드를 작은 모듈단위로 나누어야 한다. 이 모듈은 침입 프로그램의 흐름을 방해하지 않고 위치를 바꿀 수 있는 단위이다. 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 부분이다.

3.3.1 모듈 위치 변환

여러 부분으로 나눌 수 있는 모듈은 침입 코드에서 특정한 일을 수행하는 한 부분이다. 각 모듈의 위치를 변환하기 위해 모듈간 의존성 파악하여 위치 조정이 가능한 모듈과 그렇지 않은 모듈로 나눈다. 이들 나누어진 모듈의 위치 변환을 통해 침입을 실행하면 침입탐지 시

스템의 척도로 사용되는 시스템 호출의 생성 순서가 바뀌어서 침입을 실행한다.

3.3.2 더미코드 삽입

대부분의 침입은 일련의 침입 행위가 계속 발생되지 않고, 시스템의 정상 행위를 수행하면서 특정 행위를 유발하여 침입을 하게 된다. 그림 1은 불법적 파일접근 침입을 나타내는 시나리오로서 step1에서 step6까지는 시스템의 정상적인 수행을 하고, step7과 step8이 실행되면서 파일에 대한 사용 권한을 획득하게 된다.

그림 1. 불법 파일 접근 시나리오

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)*

그러나 step7과 step8 두 가지 행위만으로는 침입을 수행할 수 없다. 불법 파일접근 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. 여러 모듈로 나누어진 u2r 침입도 모듈 사이에 침입코드와 무관한 더미코드를 삽입하여 다양한 시스템 호출을 발생하여 새로운 침입을 만든다.

3.4 침입 변환율

모듈 위치변환을 통해 각 나누어진 모듈은 서로 위치를 바꿈으로서 실행 가능한 침입코드를 만든다. 예를 들어 n개의 모듈로 나누어진 침입코드는 $n!$개의 침입 가능한 코드를 만들어 낼 수 있고, 그중 x개의 침입이 성공하였다면 $\frac{x}{n!} \times 100\%$의 침입 변환 성공률을 보인다. 모듈이 바뀐 위치에서 침입실행이 된다는 것은 기존 침입코드에서 발생되던 이벤트의 순서가 바뀌어서 실행된다는 것을 의미한다.

4. 실험결과

실험을 위해 솔라리스 2.7과 2.8 버전 기반의 u2r 침입 13 종류를 기반으로 침입 변형을 하였다. 솔라리스 2.7에서는 버퍼오버플로우 7가지와 시스템 설정오류를 이용한 침입 2가지를, 솔라리스 2.8버젼에서는 버퍼오버플로우 4가지를 변형하였다. 각 침입 코드별 모듈의 위치 변형 가능성을 조사하고 이것을 기반으로 침입의 변형 가능한 수와 실제 침입 변형이 성공한 수는 표4와 같다. 침입코드의 모듈 위치 변형을 통해 평균 6.1%의 침입 변환 성공률을 알 수 있다.

표 4. u2r 침입코드의 침입 변형 성공률

OS	침입코드명	침입변형수/침입가능수	성공률
solaris intel 2.8	kcms_configure_overflow.c	2/56	3.57%
	whodo.c	27/121	22.31%
	xlock.c	11/225	4.88%
	mailx.c	26/110	23.63%
solaris sparc 2.7	xlock.c	10/225	4.44%
	xsun.c	1/81	1.23%
	ex_lobc.c	4/121	3.30%
	dtaction.c	7/121	5.78%
	dtprintinfo.c	5/100	5%
	ex_admintool.c	5/361	1.38%
	lpset.c	2/121	1.65%
	3lc_messages_passwd_stack.c	5/361	1.38%
	eject-fmt.c	4/361	1.10%

5. 결론 및 향후연구

본 논문에서 소개한 침입탐지 평가도구를 위한 자동 침입 생성에 관한 연구는 기존 알려진 u2r 침입을 이용하여 새로운 침입을 만들어 냈다. 사용된 u2r 침입의 종류는 버퍼오버플로우 침입과 시스템설정 오류를 이용한 침입이다. 이들 침입의 변형에서 각 코드내의 모듈을 나눌 때 의미가 있는 곳과 없는 곳을 나누어서 위치 이동을 사람이 의미적으로 파악하고, 모듈 이동을 통한 침입 변형을 시도하였다. 실험에 사용된 13종류의 u2r 침입의 변형은 모듈 위치 변환을 통해 평균 6.1%의 변형률을 보였다.

향후 연구로는 침입 변형의 자동화를 위해 침입 코드의 의미있는 모듈 단위로 나누는 작업을 파서 기능을 통해 제시하고, 코드의 위치변환 문제를 진화 연산을 통해 최적화된 자리를 찾아야 할 것이다.

참고문헌

[1] R. P. Lippmann, et. al, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), vol. 2, IEEE Press, New York, January 2000.

[2] R. P. Lippmann, et. al, "The 1999 DARPA off-line intrusion detection evaluation," Computer Networks, vol. 34, no. 4, pp. 579-595, October 2000.

[3] G. Shipley, "Intrusion detection, take two," Network Computing, November 1999.

[4] H. Debar, M. Dacier, A. Wespi and S. Lampart, "An experimental workbench for intrusion detection systems, ", IBM Research Division, Zurich Research Laboratory, March 1999.

[5] N. Puketza, K. Zhang, M. Chung, B. Mukherjee, and R. A. Olsson, "A methodology for testing intrusion detection systems," IEEE Transactions on Software Engineering, vol 22, pp. 719-729, 1996.

[6] N. Puketza, M. Chung, R. A. Olsson, and B. Mukherjee, "A software platform for testing intrusion detection systems," IEEE Software, pp. 43-51, Sep./Oct. 1997.

[7] R. Dust, T. Chanmpion, B. Witten, E. Miller, and L. Spagnuolo, "Testing and evaluating computer intrusion detection systems," Communications of the ACM vol. 42, pp. 53-61, 1999.