모바일 코드를 이용한 최적적응 침입탐지시스템

방 세 중[†] · 김 양 우[††] · 김 윤 희[†††] · 이 필 우[††††]

요 약

지식사회의 역기능인 정보시스템에 대한 각종 침해행위들로 정보자산의 피해규모는 나날이 증가하고 있다. 이러한 침해행위 중에서 네트워크 보안과 관련된 범죄수사 요구의 강화는 침해행위탐지와 이에 대한 대응 및 보고를 포함하는 다양한 형태의 침입탐지시스템들에 대한 연구개발을 촉진시켜왔다. 그러나 초기 침입탐지시스템은 설계상의 한계로 다양한 네트워크 환경에서 오탐지(false-positive)와 미탐지(false-negative)뿐만 아니라 침입탐지시스템을 우회하는 방법에 대처하기 힘들었다. 본 논문에서는 이런 문제점을 모바일 코드를 통한 최적적응 능력을 갖춘 가상프로토콜스택(Virtual Protocol Stack)을 통해 보완함으로서 침입탐지시스템이 다양한 환경에서 능동적으로 감시중인 네트워크의 상황을 자동학습 하도록 하였다. 또한 본 논문에서는 이를 적용하여 삽입/회피(Insertion/Evasion) 유형의 공격이 적극적으로 탐지될 수 있음을 보였고, 이러한 방법은 보다 다양한 혼성의 네트워크 환경에서도 적응능력을 갖춘 침입탐지 기법으로 확대 적용될 수 있음을 논의하였다.

An Optimum-adaptive Intrusion Detection System Using a Mobile Code

Se-chung Pang[†] · Yang-woo Kim[††] · Yoon-hee Kim[†††] · Phil-Woo Lee[††††]

ABSTRACT

A damage scale of information property has been increasing rapidly by various illegal actions of information systems, which result from dysfunction of a knowledge society. Reinforcement in criminal investigation requests of network security has accelerated research and development of Intrusion Detection Systems(IDSs), which report intrusion detection about these illegal actions. Due to limited designs of early IDSs, it is hard for the IDSs to cope with tricks to go around IDS as well as false-positive and false-negative trials in various network environments. In this paper, we showed that this kind of problems can be solved by using a Virtual Protocol Stack(VPS) that possesses automatic learning ability through an optimum-adaptive mobile code. Therefore, the enhanced IDS adapts dynamically to various network environments in consideration of monitored and self-learned network status. Moreover, it is shown that Insertion/Evasion attacks can be actively detected. Finally, we discussed that this method can be expanded to an intrusion detection technique that possesses adaptability in the various mixed network environments.

키워드 : 가상프로토콜스택(Virtual Protocol Stack), 최적적응(Optimum-adaptive), 침입탐지시스템(Intrusion Detection System), 모바일코드(Mobile Code), 삽입/회피 공격(Insertion/Evasion Attack)

1. 서 론

현대의 정보화 사회에서는 다양한 운영체제(OS)를 탑재한 수많은 컴퓨터들이 인터넷의 성장과 더불어 네트워크로 연결되어 있다. 이들은 각종 정보 서비스를 제공하게 되었고, 이를 이용하여 우리의 사회는 새로운 지식을 창출하는 지식정보사회로 빠르게 이행되고 있다. 그러나 이와 더불어 역기능인 정보시스템의 침해사고가 증가하고 있으며 이와 함께 네트워크 침해사고 또한 급격히 증가하고 있는 실정이다[1]. 이것은 기본 네트워크 환경으로 자리 잡은 인터넷 프로토콜인 TCP/IP 프로토콜의 설계 및 구현 시 취약점과 다양한 운영체제 환경에서 비롯된 보안 결함들로 인해 도청(Eavesdropping), 변조(Tampering), 위장(Impersonation) 등의 침입 공격에 노출되었기 때문이기도 하다. 따라서 인터넷에 연결된 각종 정보시스템을 보호하기 위해 관리자들은 시스템이 갖고 있는 자체 취약성을 분석하여 알려진 보안결함을 제거하거나, 암호화시스템 및 접근제어시스템 등의 보안 시스템을 구축, 운영하기도 한다.

그러나 기본적인 방어시스템을 구축하였음에도 불구하고 외부와의 연결을 위해 허용되어 있는 경로를 통해 형성된 위장 채널(Covert Channel)은 외부로부터의 침입과 내부자에 의한 침해행위를 줄이지 못하여 왔다. 이에 따라 시스템 관리자들은 정보시스템에 대한 단순한 방어뿐

※ 이 연구는 2003학년도 동국대학교 연구년 지원에 의하여 이루어졌음.
† 정 회 원 : 넷시큐어테크놀러지(주) 부설연구소, 수석연구원
†† 종신회원 : 동국대학교 정보통신공학과 부교수
††† 종신회원 : 숙명여자대학교 컴퓨터과학과 조교수
†††† 정 회 원 : 한국과학기술정보연구원(KISTI), 초고속정보망기술지원실, 선임연구원/박사
논문접수 : 2004년 9월 10일, 심사완료 : 2004년 11월 25일

만 아니라 침입 사실을 탐지, 대응 및 보고를 하는 침입탐지시스템(Intrusion Detection System : IDS)[2]을 도입하여 기존 정보보호시스템을 보완하고 효과적으로 침입과 공격에 대한 피해를 최소화하고자 하였다. 그러나 IDS에는 자체적으로 갖고 있는 한계로 인하여 다양한 환경에 대응하지 못하고 오탐지가 증가하는 문제가 발생하고, 이와 더불어 IDS를 우회하여 공격하는 기술도 같이 발전되어왔다. 즉, 효과적인 침입탐지와 대응을 방해하는 우회공격 중에 대표적인 기술인 IP 프래그먼트(Fragment)[3]를 이용하거나 WEB 서버의 종류에 따른 차이와 고유의 특성[4]을 이용하는 등 여러 방법이 있다. 이 중 하나가 여러 운영체제들이 서로 다르게 TCP/IP 네트워크 프로토콜 스택을 구현하였다는 점을 이용하는 삽입/회피 공격이다[5, 6]. 이와 같은 우회공격은 IDS가 TCP/IP 프로토콜의 패킷에 대한 오류 처리 시, 모니터링하고 있는 네트워크 그룹의 정보시스템과는 다르게 그 패킷을 처리함으로써 같은 패킷에 대해 다른 결과를 보여 줄 수도 있다는 점을 이용한 공격이다. 이런 종류의 우회공격에 대한 피해를 제거하기 위해 우리는 모바일 코드를 통한 최적적응 능력을 갖춘 가상프로토콜스택(Virtual Protocol Stack)을 제시하고 구현하였다. 가상프로토콜스택의 역할은 모니터링하고 있는 네트워크 그룹 내부에서 서로 다른 운영체제로 운영되는 정보시스템들을 사전에 테스트하여 IDS와 모니터링 되는 각각의 정보시스템들이 동일한 방법으로 패킷을 처리하도록 하는 것이다. 이렇게 함으로써 삽입/회피 공격 시도를 방지하고 IDS의 오탐지 또는 미탐지 발생을 줄이도록 하는 것이다.

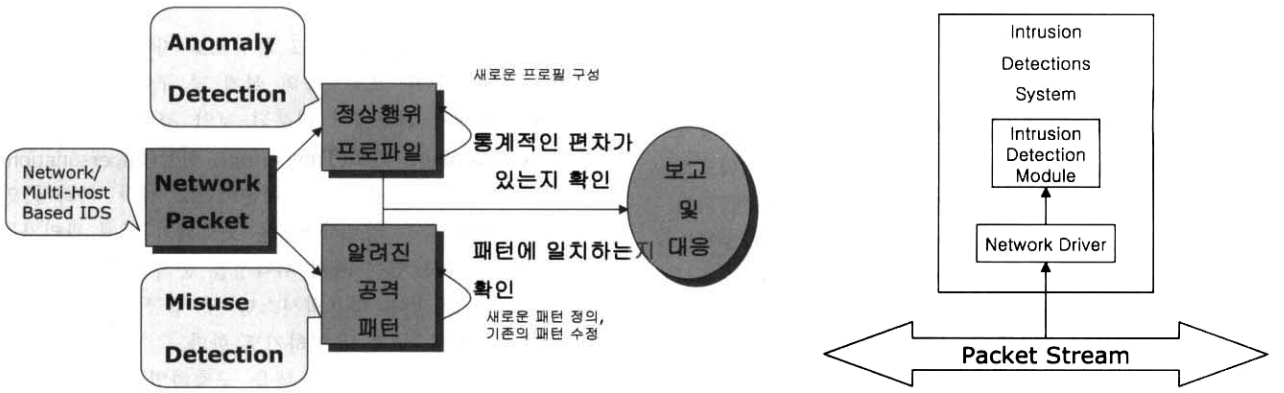
이후 본 논문에서는 우선 기존 관련연구 및 가상프로토콜스택의 개념을 먼저 설명하고, 다음으로 가상프로토콜스택을 포함한 모니터링 시스템의 구성요소들에 대한 설명과 실시한 테스트에 대한 시나리오를 소개하였다. 그리고 이에 대한 결과 및 검토사항과 함께 다른 공격에 대한 확장부분도 기술하였다.

2. IDS의 특징과 구성

침입(Intrusion)은 정보시스템의 무결성(Integrity), 기밀성(Confidentiality), 가용성(Availability)을 저해하는 일련의 행동이나 정보시스템의 보안정책을 위반하는 행위를 말한다[2, 7, 8]. 또한 이러한 침입을 탐지하는 시스템을 침입탐지시스템(Intrusion Detection System : IDS)이라 하며 탐지에 사용되는 데이터의 출처를 가지고 분류하거나 탐지모델에 따른 분류를 할 수 있다. 전자는 크게 네트워크 기반의 IDS(NIDS)와 호스트 기반의 IDS(HIDS)로 구분되며 후자의 분류로는 비정상행위탐지(anomaly detection)와 오용탐지(misuse detection) 방식으로 나눌 수 있다. 비정상행위탐지에 대한 논의도 점점 활발해지지만[9] 아직은 공개 침입탐지시스템인 Snort[10]뿐만 아니라 상용시스템[11]과 많은 연구에서 대부분 오용탐지방식을 채택하고 있다.

오용탐지는 1980년 Denning에 의해 처음 제안되었는데 알려진 공격패턴을 이용하여 탐지를 수행하는 형태이다[12]. 특히 이런 형태의 IDS는 네트워크 인터페이스 카드의 설정을 무차별 모드(Promiscuous mode)로 설정하여 해당 네트워크의 모든 패킷을 읽어 들여 패킷의 헤더와 내용을 기존의 침입탐지패턴과 비교하는 검사를 수행하는 방식이다.

그리고 이러한 초기 IDS에서 침입탐지를 정교화하기 위해 다양한 형태의 IDS가 연구되었는데 그중에 일종의 모바일 코드의 구현형태인 에이전트를 적용한 연구가 있다. 이와 관련하여 대표적인 IDS로는 SRI(Stanford Research Institute)에서 수행하는 EMERALD(Event Monitoring Enabling Response to Autonomous Live Disturbance)이다. 이 모델은 에이전트가 로컬 네트워크 트래픽을 관찰하고 엔터프라이즈 모니터에게 분석보



a. IDS의 침입분석 방법

b. NIDS의 구성

(그림 1) IDS의 작동방식

고서를 보내면 엔터프라이즈 모니터는 이러한 보고서들을 종합적으로 관리하는 형태로 에이전트의 기능은 중앙에 자료를 보내는 것만 하는 한정된 형태이다[13, 14, 15].

지금까지 간단하게나마 살펴본 IDS는 침입차단시스템(Firewall : FW)이 갖는 시스템보호 한계를 극복하려고 설계되었으나 이 또한 침입행위가 늘어나면서 실제 침입을 적절히 구분하기 어렵게 만드는 다량의 오탐지 발생 문제와 기존의 침입탐지패턴이 없으면 탐지를 못하는 미탐지에 대한 문제와 함께 네트워크 기반인 NIDS는 침입을 실시간으로 막을 수 없다는 한계를 갖고 있다. 이것에 대한 문제는 근래 침입방지시스템[16, 17]이란 개념으로 활발히 논의되고 있다. 그러나 우리는 오용탐지 방식의 NIDS가 다양한 운영환경과 네트워크 환경을 능동적으로 학습하여 오탐지와 미탐지를 줄여 공격시도를 적절히 구분해 내도록 하는 가상프로토콜스택을 제안하고 적용하여 IDS의 설계상 문제를 보상하도록 하였다.

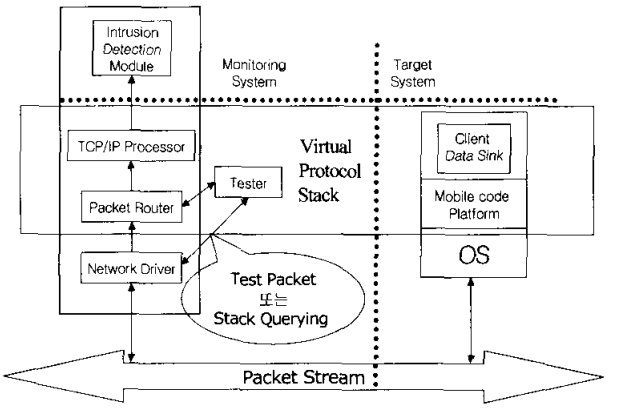
3. 가상프로토콜스택(Virtual Protocol Stack)의 개념

가상프로토콜스택은 NIDS의 적응능력을 강화하는데 사용되며, 침입탐지 모듈을 위해 정확한 정보 자료를 제공한다. NIDS가 적응해야 될 다양한 환경에 대한 한 가지 예를 들면 TCP/IP 프로토콜이 OS마다 구현되면서 발생하는 차이를 이용한 삽입/회피 공격이 있다. 이 공격은 공격자가 IDS와 대상 정보시스템 또는 대상 시스템 간에 전송되는 패킷을 왜곡하여 두 시스템이 같은 패킷을 서로 다르게 해석하게 하여 침입하는 우회공격이다[5]. 즉, 이러한 공격 중 하나는 TCP Header부분의 체크썸을 유효하지 않은 값으로 전송하는 방식이다. 가상프로토콜스택의 중요한 역할은 각각의 다양한 시스템을 모니터 하여 시스템마다 여러 OS에서 서로 다르게 구현된 TCP/IP의 구현 상태를 미리 마련된 테스트 시나리오를 통해 확인 한 후 대상 시스템과 IDS가 서로 같은 관점으로 패킷을 처리하도록 함으로써 삽입/회피 공격으로부터 피해를 최소화 할 수 있도록 하는 것이다.

4. 가상프로토콜스택(Virtual Protocol Stack)의 구성

가상프로토콜스택은 크게 모니터링 시스템과 타겟 시스템으로 나누어진다. 모니터링 시스템은 TCP 프로세서와 IP 프로세서, 패킷 라우터와 테스터(Tester)로 구성되어 있고 부가적으로 네트워크 드라이버와 연동하는 형태이다. 또 타겟 시스템 부분은 모바일 코드 플랫폼과 클라이언트 데이터 싱크(Client Data Sink) 부분으로 구성되어 있으며 각각의 구성요소들이 하는 역할은 다음과 같다.

4.1 모니터링 시스템(Monitoring System)의 구성



(그림 2) 가상프로토콜스택의 구성도

4.1.1 네트워크 드라이버

네트워크 드라이버는 *Packet Sniffer*와 *Packet Injector*라는 두 부분으로 이루어진다. Packet Sniffer는 네트워크로부터 패킷을 잡아내어 패킷의 헤더 부분을 조사한 후 이 패킷을 패킷 라우터로 보내는 역할을 하며 Libpcap 라이브러리를 이용하여 구성했다. 패킷을 캡처하기 위한 도구는 각 운영체제별로 다양한 도구가 있지만 Libpcap 라이브러리는 운영체제마다 다르게 구현된 데이터링크의 접근 방법과 상관없이 동일한 인터페이스를 갖기에 VPS 설계 개념에 따라 이를 사용하였다[18]. 그리고 Packet Injector는 대상 시스템에 테스터가 보내는 테스트 시나리오를 네트워크를 통해 대상 시스템으로 보내는 역할을 한다. Injector는 Libnet 라이브러리를 이용하였는데 이 또한 Libcpap 라이브러리와 같은 이유로 이기종 운영체제 어디에서도 같은 인터페이스를 갖고 TCP/IP의 몇 개의 프로토콜에 대해 다양한 패킷을 만들어 보낼 수 있어 채택하였다[19].

4.1.2 패킷 라우터

네트워크 드라이버로부터 패킷을 받은 패킷 라우터는 이 수신된 패킷으로부터 대상 시스템의 IP 주소를 알아내어 이 IP 주소가 패킷 라우터 내의 IP 목록에 있는지 검사한다. 이 목록은 테스터에 의해 시나리오에 따른 검사를 마친 시스템의 IP 주소가 등록되어 있는 것으로 만약 이 목록에 이미 패킷의 IP 주소가 있으면 이 패킷은 해당 대상 시스템을 표현하는 IP 프로세서로 전송되어 처리된다. 그러나 목록에 등록이 되어 있지 않다면 패킷 라우터는 테스터에게 해당 대상 시스템에 대해서 테스트가 필요하다고 알린다.

4.1.3 테스터

테스터는 대상 시스템의 TCP/IP 구현상태를 검사하고 검사한 결과를 가지고 IP와 TCP 프로세서를 구성한다. 패킷 라우터가 IP주소 목록에 대상 시스템의 IP 주소가 등록되지 않았다고 테스터에게 통지를 하면 테스터는 대

상 시스템의 모바일 코드 플랫폼에 연결하여 클라이언트 데이터 싱크 애플리케이션을 전송한다. 이후 테스터는 일련의 테스트 시나리오에 따라서 조작된 패킷을 대상 시스템의 클라이언트 데이터 싱크 애플리케이션에게 전송하고 그 테스트 결과는 클라이언트 데이터 싱크가 테스터에게 다시 전송한다. 이 테스트 결과를 가지고 테스터는 대상 시스템의 IP와 TCP 프로세서를 구성하고 패킷 라우터의 IP 목록에 검사한 대상 시스템의 IP 주소를 등록한다.

4.1.4 TCP/IP 프로세서

IP 프로세서의 역할은 테스터에 의해 생성된 규칙을 가지고 대상 시스템과 동일하게 인터넷 프로토콜을 처리하는 것으로 IP 헤더 옵션을 점검하고 IP 프래그먼트를 재조립한다[20]. 그리고 IP 프로세서는 테스터에 의해 생성된 규칙을 가지고 패킷을 폐기해야 할지 아니면 처리해야 할지 결정한다. 이렇게 패킷을 받아 처리한 후에 TCP 세그먼트(Segment) 부분을 해당 TCP 프로세서로 보낸다.

이후 TCP 프로세서는 TCP 헤더 필드 부분을 검사한다. 이러한 필드는 연결 상태를 결정하고, 데이터의 경로와 같은 대상 시스템 상에 존재하는 응용프로그램을 가리키며, 데이터의 순서를 결정하기 위해 필요한 단서들이다. 유효하지 않은 TCP 헤더 필드의 영향은 TCP 표준에 정의되어 있지 않으며 구현에 따라 변할 수 있다[21]. 다시 말해서 TCP 문서에는 오류가 발생 할 경우 그 대응책이 명시되어 있지 않다. 가상프로토콜스택은 이렇게 정의되어 있지 않은 시나리오에 대해 정의를 내리며 IDS에 제공되는 패킷이 일관성을 갖도록 한다.

4.2 타깃 시스템의 구성

4.2.1 모바일 코드 플랫폼

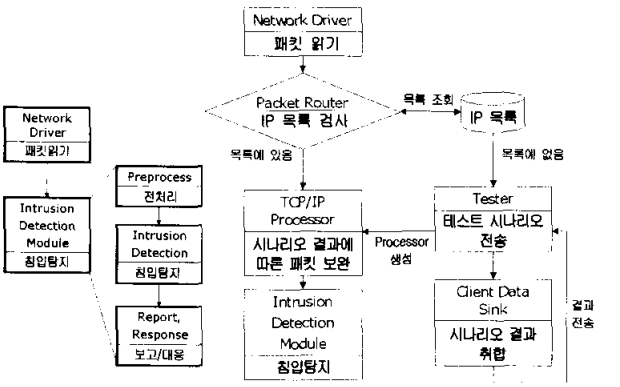
모바일 코드 플랫폼은 테스터에 의해 전송되어지는 클라이언트 데이터 싱크 애플리케이션을 받아들이고 이것이 대상 시스템 상에서 실행될 수 있도록 하는 역할을 한다.

4.2.2 Client Data Sink

클라이언트 데이터 싱크는 일종의 자바 클래스로서 대상 시스템에서 실행된 테스터 시나리오의 결과를 취합하여 다시 테스터로 보내는 역할을 한다. 테스터는 대상 시스템에서 처리된 테스트 시나리오 패킷들을 수집하여 IP와 TCP 프로세서를 구성하도록 한다.

지금까지 가상프로토콜스택의 역할과 관계와 기존 방법을 (그림 3)처럼 도식할 수 있다.

이렇게 구성된 가상프로토콜스택은 OS 종류가 서로 다른 시스템들에서 (그림 3의 b)와 같이 각각의 구성부분이 동일하게 적용된다. 그리고 앞서 언급한 바와 같이 Libpcap 및 Libnet 라이브러리는 OS의 특성을 따르지 않고 클라이언트 데이터 싱크는 자바 클래스로 구성되어 환경이 다른 시스템과 독립적으로 작동한다. 따라서 가상프로토콜스택은 다양한 시스템의 특성에 영향 없이 독립적으로 구성되고 연동된다.



a. 기존 침입탐지 b. 가상프로토콜스택 적용시 '침입탐지'

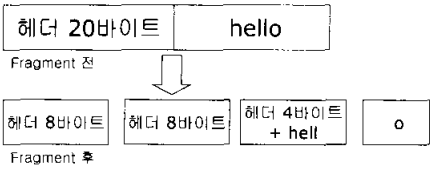
(그림 3) 가상프로토콜스택 적용 전후 동작 순서

5. 시나리오에 따른 가상프로토콜스택 평가와 결과

가상프로토콜스택의 테스트는 IDS와 대상 시스템이 패킷을 똑같이 처리하는지 검사하는 것이다. 가상프로토콜스택은 각 대상 시스템의 특성을 찾아내 보안할 수 있기 때문에 이기종 환경이라도 가상프로토콜스택이 적용된 IDS는 삽입/회피 공격을 탐지할 수 있다. 가상프로토콜스택에 대한 테스트는 패킷을 모니터링하고 있는 동안에 대상 시스템에 'hello'라는 문자열을 전송하여 가상프로토콜스택이 대상 시스템과 같은 방식으로 문자열을 조합한다는 것을 입증하는 것으로 Linux(Kernel버전 2.2.12-X, 2.4.2-X), Windows 98/ME/NT/XP/2000, Solaris 8(SunOS 5.8), FreeBSD4.6, HP-UX 11.0, AIX 4.3.3 등 다양한 OS를 통해 실험하였다.

5.1 IP Fragmentation

이 테스트는 문자열을 8byte의 IP 프래그먼트로 나누고 이것을 대상 시스템으로 전송하는 것으로 대상 시스템이 이러한 IP 프래그먼트를 재조립하는 능력을 갖고 있는지를 결정한다. 대상 시스템이 테스트되면 가상프로토콜스택은 정확하게 대상 시스템과 같은 형식으로 프래그먼트를 처리한다. 그리고 프래그먼트에 대한 테스트는 8byte로 조각을 낸 문자열을 대상 시스템으로 전송하는 방법에 따라 다음과 같이 구분한다.



(그림 4) IP Fragment

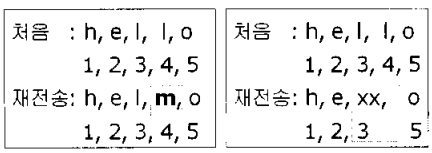
〈표 1〉 IP Fragmentation 테스트 항목

IP Fragmentation 테스트
8byte 프래그먼트를 순서대로 전송
8byte 프래그먼트를 순서 없이 전송
8byte 프래그먼트된 문자열 수정 전송

8byte 프래그먼트된 문자열 수정 전송 테스트에서는 4개의 프래그먼트를 사용하고 패킷을 수정하기 위한 다른 프래그먼트를 하나 더 사용한다. 이 프래그먼트는 'xxxx'의 문자열로서 원래 4개의 프래그먼트를 전송하는 중간에 전송된다. Linux에서는 이 프래그먼트를 받아 'xxxxo'라는 결과가 나오지만, Windows를 비롯한 다른 시스템은 이런 경우를 수락하지 않기 때문에 'hello'라는 문자열 결과가 나온다. 그러나 Ptacek, Newsham의 초기 조사 결과는 Windows와 Solaris 두 개의 OS만 'hello'라는 문자열 결과가 나오고 Linux를 비롯한 다른 OS는 'xxxxo' 문자열 결과를 보인다[5].

5.2 TCP Segmentation

침입자들은 유효하지 않은 패킷 시퀀스 넘버(Sequence Number)를 가진 패킷을 전송하여 IDS와 비동기 시키려고 한다. 만약 IDS가 잘못된 시퀀스 넘버를 재 동기화 시키려고 한다면 IDS는 비동기된 데이터의 모니터링에 의해서 탐지 능력이 떨어질 수가 있다. 그래서 모든 시퀀스 넘버를 대상 시스템과 같은 방식으로 처리하기 위해 가상프로토콜 스택을 사용한다. 아래의 테스트는 'hello'라는 문자열을 6개의 1byte 세그먼트로 나눠서 전송하고



a. 세그먼트 수정 b. 세그먼트 중첩 수정

(그림 5) TCP 세그먼트 중첩 수정

〈표 2〉 TCP Segmentation 테스트 항목

TCP Segmentation 테스트
하나의 세그먼트 반복 전송
세그먼트 수정 전송
세그먼트 중첩 수정
순서 없이 세그먼트 전송
무효의 순서번호를 갖는 문자열 끼워 넣기 전송
무효한 TCP/IP 헤더 옵션
무효한 TCP/IP 체크썸

세그먼트의 중첩 수정 테스트는 "--xx-"라는 문자열을 이용하는 세그먼트를 전송하여 원래 세그먼트와 중첩 수정되는 지 알아보는 테스트로서 Windows뿐만 아니라 Linux를 포함한 모든 OS도 중첩 수정 세그먼트를 폐기한다. 이는 Ptacek, Newsham의 초기 조사 결과인 Windows을 제외한 모든 OS가 재전송 된 중첩 세그먼트를 그대로 받아들여서 'hexxo'라는 결과가 나온 것과는 차이가 있다[5].

5.3 TCP/IP 헤더 옵션

대상 시스템이 잘못된 TCP/IP 헤더 옵션과 체크썸을 가진 패킷을 수락하지 않은 반면 IDS는 이러한 패킷을 수락하게 되면 시스템과 IDS간의 패킷을 처리하는 관점이 다르게 되어 삽입/회피 공격을 당할 수가 있게 된다. 따라서 가상프로토콜스택은 테스트 결과를 이용하여 대상 시스템과 IDS가 서로 동일한 방법으로 패킷을 처리하도록 한다.

〈표 3〉 TCP/IP Header Option 테스트 항목

TCP/IP Header Option 테스트
무효의 TCP/IP 체크썸 전송
무효의 IP 버전 전송
무효의 IP Header 크기 전송
무효의 TCP Header 크기 전송

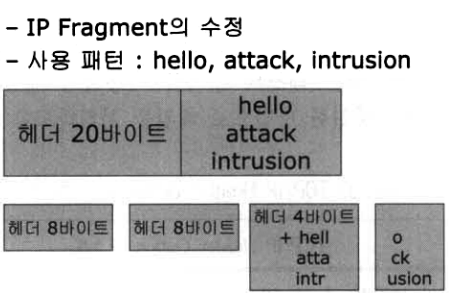
〈표 4〉 테스트 시나리오에 따른 결과

테스트	테스트 문자	Linux 2.2.11-X 2.4.2-X	Windows 98/ME/NT /XP/2000 Solaris 8 FreeBSD 4.6 HP-UX 11.0 AIX 4.3.3
8byte 프래그먼트 순서대로 전송	hell,o	hello	hello
8byte 프래그먼트 순서 없이 전송	hell,o	hello	hello
8byte 프래그먼트 문자열 수정 후 전송	hell,o /xxxx-	xxxxo	hello
하나의 세그먼트 반복 전송	h,e,l,l,o	hello	hello
세그먼트 수정	h,e,l,l,o /---m-	hello	hello
세그먼트의 중첩 수정	h,e,l,l,o /--xx-	hello	hello
순서 없이 세그먼트를 전송	h,e,l,l,o	hello	hello
무효의 Seq. 번호의 문자열 끼워 넣기 전송	h,e,l,l,o	hello	hello
무효의 TCP/IP 체크썸	hello	No message	No message
무효의 IP 버전	hello	No message	No message
무효의 IP Header 크기	hello	No message	No message
무효의 TCP Header 크기	hello	No message	No message

이와 같은 시나리오에 대한 각 OS별 문자열 처리 결과는 <표 4>와 같이 정리할 수 있다. 이는 Ptacek, Newsham의 초기 조사 결과와 다르게 '8byte Fragment 문자열 수정 후 전송' 항목에서만 Linux OS만이 다른 OS와 다르게 처리함을 알 수가 있다.

Comma	: 프래그먼트 되거나 세그먼트 된 문자열 구분
Dash	: 전 메시지에서 사용된 문자
Slash	: 두 메시지 구분(첫 번째/두 번째)

5.4 침입탐지의 변화



(그림 6) IP Fragment에 의한 침입탐지 패턴 테스트

앞에서 얻은 결과를 토대로 다르게 처리하는 것이 확인된 IP 프래그먼트 중 8바이트 프래그먼트된 문자열을 수정하여 전송해서 Linux와 Windows 시스템이 설치된 이기종 네트워크 환경에서 공개 IDS인 Snort[10]와 연동시켜 테스트하여 보았다. 즉, (그림 6)에서처럼 'hello', 'attack', 'intrusion' 3개의 문자열을 가지고 앞서 살펴본 방법으로 삽입/회피 우회공격 테스트를 하였다. '8byte Fragment 문자열 수정 후 전송' 테스트의 경우 가상프로토콜스택을 적용하지 않은 Linux 시스템에 대해서는 Windows 시스템과 다르게 해당 문자열에 대해 침입탐지가 되지 않았으나 가상프로토콜스택을 적용하고 같은 방법으로 테스트 문자열을 전송하였을 때 <표 5>와 같이 'hello', 'attack', 'intrusion'이라는 패턴에 대응되게 만든 침입탐지 룰(Rule)인 'TestHello-1', 'TestHello-2', 'TestHello-3'이 모두 탐지되었다. 이는 가상프로토콜스택에 의해 IP 네크워크 스택이 보완되어 처리됨을 확신할 수 있는 것이다. (그림 7)은 이중 하나인 'TestHelllo-1'의 침입탐지로그이다. 테스트할 때 사용한 211.X.X.83 IP주소의 1054 포트에서 211.X.X.114 IP주소의 80포트로 공격한 것을 알 수 있다.

<표 5> Linux OS에 대한 삽입/회피 우회공격의 IDS 탐지유무

공격패턴(명)	가상프로토콜스택 적용 전	가상프로토콜스택 적용 후
hello(TestHello-1)	미탐지	TestHello-1 탐지
attack(TestHello-2)	미탐지	TestHello-2 탐지
intrusion(TestHello-3)	미탐지	TestHello-3 탐지

```
[**] [1:1000001:0] TestHello-1 [**]
[Classification : Unknown Traffic] [Priority : 3]
11/27-16:06:42.713392 211.X.X.83:1054 -> 211.X.X.114:80
TCP TTL:128 TOS:0x0 ID:2091 IpLen:20 DgmLen:378 DF
***AP*** Seq : 0xFD1C066D  Ack : 0x70AFC1DB  Win : 0x4470
TcpLen : 20
[Xref => test test string]
```

(그림 7) 침입탐지패턴 TestHello-1의 탐지 로그

6. 결 론

서로 다른 종류의 OS를 갖는 정보시스템들로 구성된 TCP/IP 네트워크 그룹에서 네트워크 기반인 NIDS가 갖고 있는 일반적인 문제는 서로 다른 종류의 OS가 갖는 TCP/IP 스택의 구현 차이 때문에 삽입/회피 공격이 가능하다는 것이다. 이 공격은 IDS가 모니터링 하고 있는 네트워크 그룹에 있는 각각의 정보시스템으로 전송된 패킷과 다르게 해석되는 패킷을 받게 하여 침입탐지를 회피하는 공격이다. 그런데 앞선 결과로 OS 공급사들이 Ptacek, Newsham의 초기 조사 결과에서 나타난 취약성을 최신버전에서는 대부분 제거하여 서로 다른 차이는 많이 발견되지 않았다. 이러한 현상은 TCP/IP 프로토콜을 OS 공급사들이 각기 구현하면서 내재되어있던 취약성을 Ptacek, Newsham 등의 조사로 발견[3, 22]되면서 이를 제거한 결과라 추측된다.

그렇지만 본 논문에서 제안한 가상프로토콜스택에 대한 작업이 이러한 형태의 공격, 즉 다양한 환경차이를 갖는 정보시스템을 대상으로 한 공격에 대해 IDS가 능동적으로 학습하여 이런 취약성을 제거함으로써 최적의 성능을 갖도록 하는 것을 확인할 수 있었다. 이것은 IDS가 모니터링 하는 정보시스템에 모바일 코드 플랫폼이 있고 오탐지 및 미탐지를 발생시킬 상황에 대한 검사 시나리오가 준비되어있다면 클라이언트 데이터 싱크를 해당 시스템에 보내 시나리오를 시험하고 시스템마다 다른 시험 결과를 취합하여 해당 시스템에 맞는 TCP/IP 프로세서를 재구성하여 IDS가 침입탐지 기능을 수행하기 전에 차이를 보정할 수 있다. 이렇게 함으로써 IDS와 대상 시스템이 동일한 방법으로 패킷을 처리할 수 있는 것이다. 앞서 든 예를 갖고 부연하자면 WEB 서버가 포함된 다양한 환경에서 침입탐지를 할 경우 모바일 코드를 이용하여 해당 서버를 찾아내고 그에 따라 미리 정해 놓은 시나리오에 의해 테스터가 검사하고 그 결과를 돌려받아 TCP/IP 프로세서를 해당 시스템에 맞게 재구성하여 해당 IP에 대응하는 가상 프로토콜 스택을 만들어 WEB 서버 종류에 따른 취약성을 능동적으로 제거할 수 있다. 따라서 이런 식으로 가상프로토콜스택을 적용함으로써 다양한 환경에서 보다 적응력 높은 IDS를 얻을 수 있는 것이다.

본 연구에서는 모바일 코드에 의한 가상프로토콜스택을 이용해서 IDS가 탐지에 보다 정교화 되고 공격에 능동적으로 대응하는 수 있음을 보였다. 그렇지만 여기서 제시된 가상프로토콜스택 내용만으로 IDS의 오탐지를 일으키는 오탐지와 실제 공격을 탐지 못하는 미탐지를 모두 제거할 수 있는 것은 아니다. 가상프로토콜스택은 이것들을 제거하기 위한 많은 활동 중 하나이다. 향후 연구에서는 다양한 환경에 대한 체계적인 분류와 이에 대응되는 시나리오 개발과 함께 모바일 코드 배포에 따른 보안성을 논의하는 것이다.

참 고 문 헌

[1] 한국정보보호진흥원, "통계보고서", http://www.krcert.or.kr

[2] Rebecca Basce, Peter Mell, "Intrusion Detection Systems," National Institute of Standards and Technology, Nov., 2001(http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf)

[3] 정현철, "IP Fragmentation를 이용한 공격들", http://www.certcc.or.kr/paper/tr2001/tr2001-03/IP%20Fragmentation. html, 2001.

[4] Whisker Web Scanner, "웹서버 취약성 점검도구", http://www.wiretrip.net/rfp

[5] T. H. Ptacek and T. H. Newsham, "Inserting Evasion, and Denial of Service : Eluding Network Intrusion Detection," Secure Networks Inc., Jan., 1998.

[6] Denmac System, "Network Based Intrusion Detection -A Review of Technologies," Denmac System, 2000.

[7] S. Axelsson, "Intrusion detection systems : A survey and taxonomy," Technical report. Department of Com puter Engineering, chalmers University of Technology, Goteborg, Sweden, 2000.

[8] Stephen Northcutt, "Network Intrusion Detection An Analyst's Handbook-2nd Edition," New Riders Publishing, Sep., 2000.

[9] S. Noel, D. Wijesekera and C. Youman, "Modern Intrusion Detection, Data Mining, and Degrees of Attack Guilt," Applications of Data Mining in Computer Security, Kluwer Academic Publishers, 2002.

[10] Snort, "공개용 침입탐지시스템", http://www.snort.org

[11] 한국정보보호진흥원, "산업지원/시험평가/평가인증제품 현황", http://www.kisa.or.kr

[12] S. Kumar and E. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection," Proceedings of the Seventeenth National Computer Security Conference, Oct., 1994.

[13] G. G. Helmer, J. S. K. Wong, V. Honavar and L. Miller, Intelligent agents for intrusion detection. In Proceedings, IEEE Information Technology Conference, Syracuse, NY, pp.121-124, Sep., 1998.

[14] A. Porras and P. G. Neumann, EMERALD : Event Monitoring Enabling Responses to Anomalous Live Disturbances. In Proceedings of the National Infomation Systems Security Conference, Oct., 1997.

[15] A. Porras and A. Valdes, "Live Traffic Analysis of TCP/IP Gateways," in Networks and Distributed Systems Security Symposium, Mar., 1998.

[16] 정연서 등, "침입방지시스템", ETRI 기술문서, 2002.

[17] 정보홍, "침입 방지 시스템 분석", ETRI 기술문서, 2003.

[18] Pcaplib, "패킷 수신 라이브러리", http://www.tcpdump.org

[19] Libnet, "패킷 전송 라이브러리", http://libnet.sourceforge.net

[20] J. Potel, "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, USC / Information Sciences Institute, Section 3.2, line 1099, September, 1981.

[21] J. Potel, "Transmission Control Protocol - DARPA Internet Program Protocol Specification," RFC 793, September, 1981.

[22] L. Joncheray, "A Simple Attack Against TCP," In 5th USEIX UNIX Security Symposium, June, 1995.

방 세 중

e-mail : neovega@netsecuretech.com

1993년 서강대학교 물리학과 졸업(학사)

1995년 서강대학교 대학원 물리학과 (이학석사)

2003년 동국대학교 산업대학원 컴퓨터공학·전자·정보통신공학과(공학석사)

1995년~2000년 삼성SDI 중앙연구소

2000년~현재 넷시큐어테크놀러지(주) 연구소

관심분야 : 네트워크 보안, 시스템 보안, IPsec, 컴퓨터 포렌식

김 양 우

e-mail : ywkim@dgu.edu

1984년 연세대학교 전자공학과(공학사)

1986년 Syracuse Univ. 컴퓨터공학전공 (공학석사)

1992년 Syracuse Univ. 컴퓨터공학전공 (공학박사)

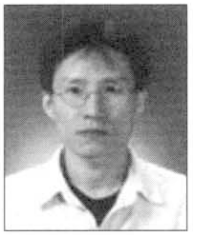
1992년~1996년 한국전자통신연구원 선임연구원

1996년~현재 동국대학교 정보통신공학과

관심분야 : 컴퓨터구조, 병렬 및 분산 그리드 컴퓨팅 시스템

김 윤 희

e-mail : yulan@sookmyung.ac.kr
1991년 숙명여자대학교 전산학과(학사)
1996년 Syracuse Univ. 전산학(공학석사)
2000년 Syracuse Univ. 전산학(공학박사)
1991년~1994년 한국전자통신연구원 연구원
2001년~현재 숙명여자대학교 컴퓨터과학과
관심분야 : 분산컴퓨팅, 그리드 컴퓨팅 시스템, 서비스/네트워크 관리



이 필 우

e-mail : pwlee@kisti.re.kr
1988년 동국대학교 전자공학과(공학사)
1991년 Univ. of Tsukuba 이공학연구과 컴퓨터공학전공(공학석사)
1998년 Univ. of Tsukuba 공학연구과 컴퓨터공학전공(공학박사)
2000년 Univ. of Tsukuba 박사후 연구원
2000년~현재 한국과학기술정보연구원(KISTI) 슈퍼컴퓨팅센터 선임연구원
관심분야 : 컴퓨터구조, 프로그래밍환경, 분산컴퓨팅, 그리드 컴퓨팅