인터넷 웜 확산 억제를 위한 거시적 관점의 프레임워크

(A Macroscopic Framework for Internet Worm Containments)

김 철 민 † 강 석 인 †† 이 성 욱 ††† 홍 만 표 ††††
(Cholmin Kim) (Sukin Kang) (Seong-uck Lee) (Manpyo Hong)

요 약 인터넷 웜은 분산 서비스 거부 공격 등을 통해 트래픽 장애를 유발하므로 인터넷 인프라를 사용 불능으로 만들 수 있고, 침입 기능을 갖고 있어 중요 서버를 장악하여 개인의 정보를 유출시키는 등 치명적인 피해를 일으키고 있다. 그러나 이러한 인터넷 웜을 탐지하여 대응하는 기존의 기법은 웜의 지역화된 특징만을 이용하거나 이미 알려진 웜에만 대응 가능한 단점을 지니고 있다. 본 논문에서는 알려지지 않은 웜을 상대적으로 빠른 시점에 탐지하기 위해 웜의 거시적 행위 특성을 추출하여 탐지하는 기법을 제안한다. 이를 위해 웜의 거시적인 행위를 논리적으로 정의하고, IP 패킷 환경에서 호스트간 직접적으로 전파되는 웜을 탐지하는 기법을 제시하며, 이 기법이 실제 적용 되었을 때 웜 확산의 억제 효과를 시뮬레이션을 통해 보인다. IP 패킷 환경은 응답 시간에 대해 민감하므로 제안된 기법이 구현되었을 때 요구하는 시간을 측정하여 제시하고, 이를 통해 상대적으로 큰 오버헤드 없이 제안된 기법이 구현될 수 있음을 보인다.

키워드 : 웜, 시큐리티, 시뮬레이션

Abstract Internet worm can cause a traffic problem through DDoS(Distributed Denial of Services) or other kind of attacks. In those manners, it can compromise the internet infrastructure. In addition to this, it can intrude to important server and expose personal information to attacker. However, current detection and response mechanisms to worm have many vulnerabilities, because they only use local characteristic of worm or can treat known worms. In this paper, we propose a new framework to detect unknown worms. It uses macroscopic characteristic of worm to detect unknown worm early. In proposed idea, we define the macroscopic behavior of worm, propose a worm detection method to detect worm flow directly in IP packet networks, and show the performance of our system with simulations. In IP based method, we implement the proposed system and measure the time overhead to execute our system. The measurement shows our system is not too heavy to normal host users.

Key words : Worm containment, Security, Simulation

· 본 연구는 지식경제 프론티어기술개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 09C1-C5-20S 과제로 지원된 것임

† 정 회 원 : (주)모비안 팀장
cmkim@imobian.co.kr
†† 학생회원 : 아주대학교 컴퓨터공학과
kingksi@ajou.ac.kr
††† 정 회 원 : 신구대학교 인터넷정보과 교수
suleeip@shingu.ac.kr
†††† 종신회원 : 아주대학교 정보및컴퓨터공학부 교수
mphong@ajou.ac.kr

논문접수 : 2009년 2월 9일
심사완료 : 2009년 7월 27일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제9호(2009.9)

1. 서 론

인터넷 웜(internet worm)은 네트워크를 통해 빠른 속도로 전파되면서 네트워크자체나 시스템에 위해(compromise)를 끼칠 수 있는 자동화된 침입 도구이다[1, 2]. 과거에는 웜을 자기복제는 하지만 다른 프로그램에 기생하지 않는 바이러스의 일부로 분류하였지만 현재는 독립적인 악성 코드의 한 부류로 받아들이며, 인터넷을 통해서 빠르게 전파된다는 것과 이를 위한 침입기능을 웜의 대표적인 속성으로 받아들여 인터넷 웜으로 통용하게 되었다[3,4]. 이러한 분류에 의해 본 논문에서는 웜과 인터넷 웜의 두 용어를 구분 없이 사용한다.

웜은 침입행위를 위해 네트워크나 시스템의 취약성을 찾아내어 보호 체계를 무력화시켜야 하므로 상대적으로 높은 프로그래밍 기술을 요구한다. 그러나 웜이 한번 만

들어지면 자동화된 루틴으로 취약성이 있는 호스트를 찾아내며, 무작위로 침입을 시도하게 되므로 상대적으로 강력한 보안 정책을 갖고 있는 네트워크라고 하더라도 모든 행위에 대해 방어를 하지 않는 이상 침투되는 경우가 많다. 이러한 이유로 웜은 해커에 의한 수동적 공격(manual attack) 보다 더 선호되는 공격방법이 되고 있다[1]. 최근에는 웜의 소스나 자동화된 웜 생성도구가 웹상에 공개되어 웜을 제작하기 위해 필요한 프로그래밍 기술의 진입장벽이 더욱 낮아지고 있는 실정이다. 따라서 변종 및 새로운 웜의 발생빈도가 증가하고 있다[5]. 이와 더불어 인터넷 사용자가 늘어남에 따라 웜에 의한 피해 규모도 커지고 있다. 최근에는 국내의 유명 사이트가 인터넷 웜에 의해 해킹 당해 국가적인 중요정보가 유출되기도 하였다[6].

인터넷 웜의 탐지 기법 중 트래픽 분석 기법은 로컬 네트워크의 정보만을 이용하게 되므로 탐지율에 한계가 있고 시그너쳐 방식의 경우 파일의 패턴비교 방식으로 인해 새로운 웜을 탐지하지 못하는 단점을 지니고 있다.

따라서 알려지지 않은 웜을 보다 정확히 탐지하기 위해서는 웜이 네트워크상에서 보이는 행위를 탐지하면서 단순히 로컬 트래픽만을 보는 것이 아닌 전체 네트워크상에서 웜이 가지는 행위적 특징을 규정해 일반적인 트래픽과 구분하는 것이 필요하다. 본 논문에서는 웜이 가지는 거시적인 특성을 이용해 웜을 탐지하는 기법을 제안한다. 또한 제안하는 기법이 적용된 비율에 따라 웜의 확산을 어느 정도 억제할 수 있는지 시뮬레이션을 통해 보이고, 실제 시스템의 구현 방법과 구현 모듈의 측정된 수행시간을 제시해 시스템 부하가 크게 증가되지 않음을 보인다.

2. 기존 연구

일반적으로 받아들여지고 있는 인터넷 웜의 정의는 다음과 같다[2].

정의 1. 인터넷 웜이란 독립적인 자기 복제와 자동화된 감염이 가능한 에이전트로써 감염되지 않은 새로운 호스트를 찾을 수 있는 기능과 네트워크를 통해 이러한 호스트를 감염시킬 수 있는 기능을 갖고 있다.

즉 웜은 호스트에 감염(자기 복제) 기능을 가지는 악성 침입 도구로 볼 수 있다. 따라서 표 1과 같이 자기복제 기능을 가지며 숙주에 기생하는 일반적 악성 코드인 바이러스와는 침입 행위 유무 면에서 차이를 보인다. 또한 악성 침입 기능을 가지는 트로이 목마는 감염 기능이 없으므로 자기 복제를 하는 인터넷 웜과 다르다. 다음의 표는 웜과 다른 악성 코드들을 특성에 따라 구분한 것이다[4].

일반인에게 주목할만한 영향을 끼친 최초의 인터넷 웜은 모리스 웜(Morris worm)으로 본다[7]. 웜이 유명해지게 된 것은 코드레드(CodeRed)나 님다(Nimda)와 같이 급속도로 많은 트래픽을 유발하는 웜의 등장 이후이다[8]. 이러한 종류의 웜이 발생시킨 변종 웜은 분산 서비스 거부공격(Distributed Denial of Service attack)으로 인터넷 인프라 자체를 사용할 수 없을 정도의 큰 피해를 주었다. 이 외에도 웜에 의한 피해는 끊이지 않고 있으며 그 건수는 계속해서 증가하는 추세이다.

표 1 악성 코드의 유형

구분	숙주(Host)에 기생	자기 복제	네트워크 이용	침입 행위
Internet Worm	X	○	○	○
Virus	○	○	△	X
Trojan Horse	X	X	○	○

2.1 기존의 웜 탐지 기법

기존의 인터넷 웜 탐지 기법은 크게 트래픽 분석, 허니팟(honeypot)과 암흑 네트워크(dark network) 모니터, 시그너쳐 기법의 3가지로 나눌 수 있다[2].

트래픽 분석은 트래픽의 물리적 양이 증가하는 것을 감시하는 것과 스캔 또는 스윕(sweep)의 빈도를 감시하는 것 그리고 트래픽 패턴의 변화를 감시하는 방법이 있다[2,9]. 이 기법이 갖는 장점은 알려지지 않은 웜을 포함하여 대부분의 웜에 모두 적용 가능하다는 점과 다형성(polymorphic) 웜에도 적용이 가능하다는 점이다. 반면 단점은 분석을 위한 시스템을 구성하는 것이 난이도가 높고 많은 노력이 들어간다는 것과 일반적인 웜에 비해 느린 속도로 전파되는 웜은 트래픽 분석을 피해갈수 있다는 점이다. 또한 다른 웜 탐지 기법에 비해 높은 긍정 오류(false positive)를 가진다.

허니팟은 악성으로 의심되는 패킷에 의도적으로 응답함으로써 웜의 악성행위를 유도하는 것을 목적으로 하며 암흑 네트워크 모니터는 존재하지 않는 주소를 목적지로 하는 패킷을 감시하는 기법이다. 허니팟은 매우 상세한 정보를 얻을 수 있다는 장점이 있다. 웜의 실행파일 자체를 얻을 수도 있다. 하지만 셋팅을 위해서 많은 노력이 필요하며 빠른 경고를 제공하지 않는다는 단점이 있다. 암흑 네트워크 모니터는 수동적으로 웜의 패킷을 받아들이므로 데이터의 수집이 쉽고 수집된 패킷의 페이로드를 제공하므로 향후 분석이 가능하다는 장점이 있다. 그러나 최근의 웜들이 존재하지 않는 주소로는 패킷을 전혀 보내지 않도록 발전되고 있어 이 기법으로 탐지하지 못하는 웜이 점차 늘어나고 있다. 또한 네트워크 내에 새로운 주소를 가진 노드가 생성되거나 주소체계에 변경이 있을 경우 이를 일일이 반영해주어야 하는

단점을 지니고 있다.

시그너쳐(signature) 기법은 세부적으로 네트워크 페이로드(payload) 방식, 로그 파일 분석, 파일 시그너쳐의 3가지로 분류할 수 있다[2,9]. 이 기법은 패턴으로 정형화된 데이터베이스를 사용하므로 개발과 적용이 쉽고 수행 속도가 빠르면서도 정확한 탐지가 가능하여 탐지된 악성 콘텐트를 제거하기에 용이하다는 장점이 있지만, 탐지 대상 웜의 정의된 패턴을 갖고 있어야 하므로 새로운 웜을 탐지할 수 없다는 큰 단점을 갖고 있다. 또한 탐지해야 하는 웜의 수가 늘어날수록 유지해야 하는 패턴 데이터베이스의 크기가 커지므로 관리가 어려우며, 탐지하고자 하는 웜의 패턴이 정의 되었다고 해도 그것이 전파 되어서 개개의 탐지 시스템 데이터베이스가 업데이트 되기 전에는 실제로 탐지가 불가능하다는 단점이 있다.

상술된 웜 탐지 기법들을 초기 탐지 가능성과 탐지 정확성의 기준으로 정리하면 그림 1과 같이 나타낼 수 있다. 일반적으로 트래픽 분석 방식에 속하는 기법들은 상대적으로 초기에 웜 탐지를 할 수 있는 장점을 지니지만 탐지의 정확도가 떨어지는 단점을 지니고 있다. 이와 반대로 시그너쳐 기반의 기법들은 정확한 탐지가 가능하지만 새로운 웜을 탐지하기까지는 상대적으로 많은 시간이 소요된다. 이는 신종 웜의 발견부터 웜의 패턴을 분석하여 추출하는 데에 사람이 관여해야 하므로 걸리는 시간이다. 이러한 현상은 트래픽 분석이 기본적으로 탐지하고자 하는 웜에 대한 사전 정보 없이 일반적인 웜 전체를 커버하는 것에 비해 시그너쳐 기반의 기법들은 이미 탐지된 웜에서 추출된 정확한 패턴을 이용하기 때문에 발생한다. 허니팟과 암흑 네트워크 모니터는 관리자의 설정에 따라 다양한 용도로 사용되는데 일반적으로 빠른 웜 탐지보다는 정확한 웜 분석에 활용되는 경우가 많다.

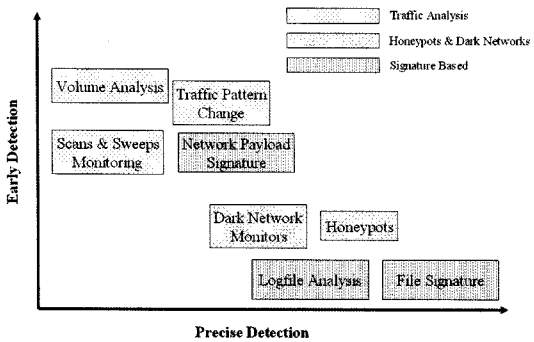

그림 1 기존 웜 탐지 기법

2.2 기존의 거시적 연구

거시적(macroscopic)이라는 표현은 바이러스의 전파를 연구하던 분야에서 언급된 것이다[10]. 이것은 호스트 내에서 바이러스의 행위를 분석하는 미시적(microscopic)연구와 반대로 전체 네트워크의 관점에서 바이러스의 행위를 분석한다는 의미로 사용되었다. 네트워크로 전파되는 특성만으로 비교하면 웜도 바이러스와 같은 모습을 보이므로 이 연구의 결과는 그대로 웜에 적용될 수 있다.

이전 연구에서는 기존 기법의 단점을 보완하기 위해 거시적 관점의 악성 코드 대응 기법을 제안하였고 전자우편 환경 상에서 시행된 시뮬레이션 결과를 보였다[11, 12]. 본 논문에서는 IP 환경에서 알려지지 않은 악성 코드에 대한 거시적 대응 기법을 제안하고 시뮬레이션 결과를 통해 그 효율성을 증명한다.

3. 제안하는 기법

3.1 인터넷 웜의 거시적 특성을 이용한 탐지 기법

2장에서 상술된 기존의 웜 탐지 기법 중 트래픽 분석에 의한 웜 탐지는 알려지지 않은 웜도 탐지가 가능하지만 로컬 네트워크에 의해서 분석된 데이터만으로 웜을 탐지하게 되므로 탐지에 한계를 지닌다. 이러한 단점을 극복하기 위해 본 논문에서는 웜의 거시적인 특성을 이용해 웜을 탐지하는 기법을 제시하고자 한다.

본 논문에서는 웜의 거시적인 특성으로 벌크 전달과 전달 체인을 정의한다.

정의 2. 벌크 전달

웜은 가능한 빠르게 전달되려는 속성이 있으므로 자신의 후손을 최대한 많이 생성한다. 따라서 웜의 전파 경로 상에는 비교적 짧은 시간 동안 하나의 호스트로부터 다수의 호스트로 전달되는 트래픽 패턴이 형성된다. 이것을 벌크 전달로 정의한다.

정의 3. 전달 체인

웜은 단일 세대의 자기 복제에 그치지 않고 계속 해서 다음 세대의 복제를 하게 된다. 따라서 웜 전달로 매개되는 호스트간의 연결이 형성된다. 이것을 전달 체인으로 정의한다.

벌크 전달과 전달 체인에 의해 웜은 그림 2와 같은 전파 행위를 보이게 된다. 본 논문에서 제안하는 웜 탐지 기법은 벌크 전달과 전달 체인을 동시에 보이는 트래픽을 웜으로 탐지 한다. 정상적인 트래픽은 벌크 전달을 여러 체인을 거치면서 하지는 않는다. 서버의 경우 다수의 클라이언트가 동시에 접속할 수 있으므로 벌크 전달 형태를 띠지만 서버에 접속했던 클라이언트가 다시 서버의 행위를 하고 여기 접속한 클라이언트가 다시 서버의 행위를 반복하는 것은 정상적이지 않다.

본 논문에서는 상술된 웜 탐지 기법을 IP기반의 네트워크 환경에서 적용하는 것을 제안한다. IP기반 네트워크는 대부분의 인터넷 네트워크를 포함하므로 이것은

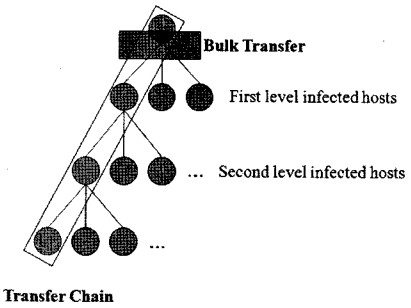

그림 2 벌크 전달과 전달 체인

거시적인 웜 탐지 기법의 일반적인 경우라 할 수 있다. 본 논문은 시스템을 구성하는 요소 중 어떠한 부분이 변경되어야 하는지를 보이며 성능이 중요시되는 모듈의 경우에는 실제 구현을 통한 시간 요구치를 보인다. 제안하는 웜 탐지 기법은 정상적이지 않은 트래픽을 감시하는 것이므로 기존의 웜 탐지 분류기법 중 트래픽 분석 방식에 속한다고 할 수 있다.

3.2 IP 패킷 환경에서의 거시적 웜 억제 기법

벌크 전달은 웜에 의해 감염된 호스트나 네트워크 게이트웨이에 의해서 관찰될 수 있지만, 전달 체인을 알기 위해서는 특별한 도구가 필요하다. IP 헤더의 내용은 헤더 컴프레션(ROHC)과 같은 경우를 제외하면 이종 망을 통과하더라도 헤더의 내용이 변경되는 일은 거의 없다. 본 논문에서는 IP 패킷 환경에서 거시적으로 웜을 탐지하기 위해 IP 헤더의 다소 자유로운 영역인 ID(identification)영역에 특정 정보를 삽입하여 전달 체인을 탐지하는 방법을 제안한다. 그림 3은 IP 헤더의 ID 영역을 나타낸다.

이 영역은 단편화(fragmented)된 패킷의 정렬을 위해서 이용되거나 터널링 등을 통해 IP 패킷이 세부적인 다른 프로토콜상위에 캡슐화(encapsulation) 될 때 사용되는 부분이다. 그러나 이 영역이 반드시 특정 목적으로 사용되도록 강제(mandatory)된 것은 아니다. 따라서 이 영역을 특정 목적에 맞게 이용한 여러 선행 연구들이

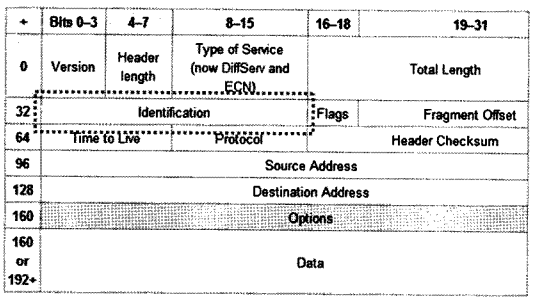

그림 3 IP 헤더의 ID 영역

있었다[13]. 본 논문에서 제안하는 기법은 이 영역에 카운트를 삽입하여 의심되는 플로우에 의한 패킷임을 알리고 의심의 정도(suspiciousness)를 표시한다.

IP 패킷 망에서의 벌크 전달은 특정 호스트에서 다수의 호스트로 동시에 다수의 패킷이 나갈 때 혹은 라우터나 스위치와 같은 네트워크 게이트웨이에서 하위 서브넷의 단일 호스트로부터 외부의(outbound) 다수 호스트로 패킷이 나갈 때 탐지 할 수 있다.

한편 전달 체인은 벌크 전달을 탐지한 호스트나 게이트웨이에서 의심스러운 패킷임을 알리는 카운트를 IP 헤더의 ID 영역에 삽입한 후, 이것이 포함된 패킷을 받은 호스트가 다시 벌크 전달을 시도하는 것을 호스트나 게이트웨이가 탐지하였을 때 성립된다. 단 호스트에서 카운트가 삽입된 패킷을 받았을 때 이것이 웜에 의한 것임을 확신하기 위해 패킷을 받은 프로세스가 다시 파일을 생성하고 생성된 파일이 실행 파일인지 확인한다. 즉 외부로부터 의심스러운 패킷을 받은 프로세스가 실행 파일을 생성하면 웜에 의한 것으로 의심하는 것이다. 전달 체인도중 카운트 값이 일정한 값을 넘으면 웜으로 판단한다.

그림 4는 상술된 과정에 의한 탐지 시나리오를 나타낸 것이다. 각 단계에서 호스트에 의해 실행되는 경우는 'H', 게이트웨이에 의해 실행되는 경우는 'G'로 구분하여 표현하였다. 초기 단계에 호스트 H1과 H2가 감염되어 있고, H1, H3, G1, G2에 제안된 웜 방지 시스템이 설치되어 있다고 가정할 때 그림 4는 다음과 같은 6 단계로 설명할 수 있다. 2, 3단계 및 4, 5단계는 시간상 어느 쪽이 먼저 일어나도 상관이 없으며, 둘 중 하나의 단계만 실행되어도 탐지가 가능하다.

1. H1과 H2에 있는 웜은 각각 활동을 시작하여 자신을 복제한 다음세대의 웜을 인터넷에 있는 다른 사용자들에게 보내기 시작한다.

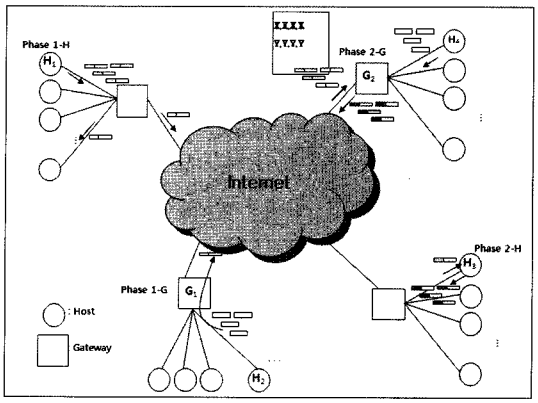

그림 4 IP환경에서의 웜 탐지 시나리오

2. (Phase 1-H) H1에서 동시에 다수의 목적지로 패킷을 보내는 것을 포착한다. H1에 설치되어 있는 웜 방지 시스템에 이 패킷의 ID필드에 카운터를 삽입한다.
3. (Phase 1-G) G1에서 동시에 다수의 목적지로 패킷을 보내는 것을 포착한다. G1에 설치되어 있는 웜 방지 시스템이 이 패킷의 ID필드에 카운터를 삽입한다.
4. (Phase 2-H) H3로 웜이 전파된다. H3에 설치되어 있는 웜 방지 시스템은 마킹된 패킷이 유입된 것으로 의심스러운 플로우가 있는 것을 감지하고 마킹 카운트를 저장해둔다. 웜이 활동을 개시하고 자기 복제를 통해 실행 파일을 생성하여 다시 네트워크로 전파된다. 웜 방지 시스템은 저장되어 있는 카운트를 증가시켜 벌크 패킷의 ID필드에 마킹한다.
5. (Phase 2-G) G2를 통해 마킹된 패킷이 내부 네트워크로 유입된다. G2는 해당 패킷의 수신자 리스트와 카운트를 저장하고(그림 4의 G2 좌측 테이블), 저장된 수신자중 벌크 패킷을 보내는 수신자가 있을 경우 저장된 카운트를 증가시켜 해당 패킷의 ID 필드에 마킹한다.
6. 웜의 전파와 자기 복제로 인해 4 또는 5의 단계가 반복되던 중 카운트 값이 임계값을 넘으면 그 카운트를 지닌 패킷을 웜에 의한 것으로 판단한다.

3.3 제안된 기법의 구현 모델

제안된 기법을 구현하기 위해서는 호스트와 게이트웨이를 향상 시켜야 한다. 호스트에는 패킷 유입을 감시하는 모니터와 파일 생성을 감시하는 모니터가 필요하다. 윈도우의 경우 드라이버 단에서 이러한 모니터를 추가할 수 있는 환경을 제공한다. 제안하는 호스트 시스템은 유입 패킷 모니터(inbound packet monitor), 전송 패킷 모니터(outbound packet monitor), 파일 모니터의 총 3개 모듈로 이루어진다.

유입 패킷 모니터는 외부에서 호스트로 들어오는 패킷을 감시하게 되며 카운트를 가진 패킷이 들어올 때는 해당 프로세스 번호(process identifier)와 카운트 값을 저장해 둔다. 파일 모니터는 파일 생성요청이 있을 때 유입 패킷 모니터에 의해 저장된 번호를 갖는 프로세스인지 확인하고 동일 프로세스이면 카운트 값을 증가 시켜 저장한다. 만약 카운트 값이 임계값을 넘으면 웜으로 판단한다. 전송 패킷 모니터는 외부로 유출되는 패킷이 벌크 전달에 의한 것인지 판단하고 벌크 전달인 경우는 파일 모니터에 의해 저장된 카운트 값을 패킷의 ID영역에 저장한다. 한편 긴 시간이 흐른 후에는 체인으로 연결되지 않은 패킷에 대해서도 마킹 카운트 값을 증가시켜 전송시킬 가능성이 있으므로 마킹 카운트 값은 주기적으로 리셋 한다.

게이트웨이에는 의심스러운 패킷을 수신한 서브넷 내의 호스트 주소를 저장하기 위한 테이블이 필요하다. 외부에서 게이트웨이가 관리하는 서브넷 쪽으로 패킷이 유입되면 게이트웨이는 패킷이 카운트를 가지고 있는지 확인한다. 카운트를 가지고 있는 경우에는 그 패킷의 수신자 주소와 카운트를 저장한다. 만약 내부 단일 호스트에서 외부 다수의 호스트로 패킷 전송이 일어나면 게이트웨이는 발신자의 주소를 저장되어 있는 주소리스트에서 찾아 기존 카운트 값을 1증가시켜 ID영역에 삽입하여 외부로 보낸다. 만약 카운트 값이 임계값에 도달하였다면 웜에 의한 패킷으로 탐지한다.

4. 실험 및 결과 분석

상술된 IP기반 웜 탐지 기법을 시뮬레이션하기 위해 본 절에서는 다음과 같은 가정을 하였다.

① 게이트웨이와 게이트웨이 사이의 망(인터넷)에서 IP헤더의 변경이 일어나지 않는다.
② IP단편화(fragmentation)가 일어나는 경우는 제외한다.
③ 호스트 사용자중 일부가 제안된 시스템을 자신의 PC에 설치한다.
④ 게이트웨이 중 일부에 제안된 시스템이 탑재된다.
⑤ 발견된 웜 정보는 향상된 게이트웨이 간에 공유된다.

첫 번째 가정은 게이트웨이를 지난 패킷이 인터넷 망을 통과하는 동안 IP주소의 변경과 같은 헤더내의 정보가 변하지 않는다는 가정이다. 그러나 실제로는 DHCP 서비스(Dynamic Host Configuration Protocol)등을 통해 유동 IP를 이용해서 망에 접속이 되는 경우가 있기 때문에 헤더의 내용이 변경되기도 한다. 시뮬레이션에서는 이러한 경우를 제외하기로 하였다. 두 번째 가정을 통해 IP헤더의 ID필드가 제안된 기법에 의해서 자유롭게 사용될 수 있다. 일반적으로 종단에서부터 IP패킷이 단편화 되어 전송되는 경우는 드물고 패킷을 전송하는 도중에 MTU(Maximum Transfer Unit)의 제한에 의해 단편화 되는 경우가 있지만 이 경우는 IP - IP의 캡슐화(encapsulation)가 되므로 원래 패킷의 ID필드가 변경되지는 않는다.

본 절에서는 제안된 기법을 탑재한 호스트를 향상된 호스트, 게이트웨이를 향상된 게이트웨이로 각각 명명하였다. 시뮬레이션에서는 향상된 호스트와 게이트웨이의 비율을 변경하면서 각각의 비율이 웜의 전파에 어떠한 영향을 가져오는지 보일 것이다.

시뮬레이션의 시간상 흐름을 나타내는 스크립트는 Tcl로, 시뮬레이션에 사용될 객체 정의는 C++를 이용해서 프로그래밍하였다. 본 절에서 시뮬레이션하고자 하는 인터넷 웜의 프로토타입인 C++클래스는 그림 5와 같다. 그림 5의 클래스는 NS2에서 기본적으로 제공하는 모델

```
class IPWormApp : public WormApp
{
        public :
                IPWormApp() ;
                void timeout() ;
                void PrevMethTimeOut() ;
                int command(int argc, const char*const* argv) ;

        protected :
                void start() ;
                void recv(int nbytes) ;

                bool infected_ ;
                static unsigned long infect_total_ ;

                ReportTimer    *timer_ ;
                PrevMethTimer *PrevMethTimer_ ;
};
```

그림 5 제안된 웜 시뮬레이션 클래스

인 WormApp 클래스에서 상속받은 것이다.

시뮬레이션 위상에는 호스트 노드와 게이트웨이 노드가 존재하며 각 호스트 노드는 특정 게이트웨이 노드에 위상적으로 속해 있는 것으로 표현된다. 시뮬레이션에 사용된 호스트의 수는 총 3000개이며 게이트웨이는 30개이다. 시뮬레이션은 주어진 가상의 IP패킷 환경에서 제안된 기법이 적용된 호스트와 게이트웨이의 비율을 각각 늘려가는 방식으로 진행되었다.

각 호스트에서 웜이 동작하였을 때 웜이 다음 세대를 퍼트리기 위해 찾는 호스트의 수는 평균 10으로 가정하였다. 그림 6은 향상된 호스트와 게이트웨이가 전혀 없을 때 웜의 확산을 발생빈도에 따라 시뮬레이션 한 결과이다. 웜 생성 빈도는 각 호스트에 대해 감염될 확률을 의미한다. 예를 들어 발생빈도 0.2는 감염되지 않은 하나의 호스트가 다음 세대 때 1/5의 확률로 감염이 되는 상태를 말한다.

시뮬레이션에서 가로축은 단위를 가지지 않고, 시간적

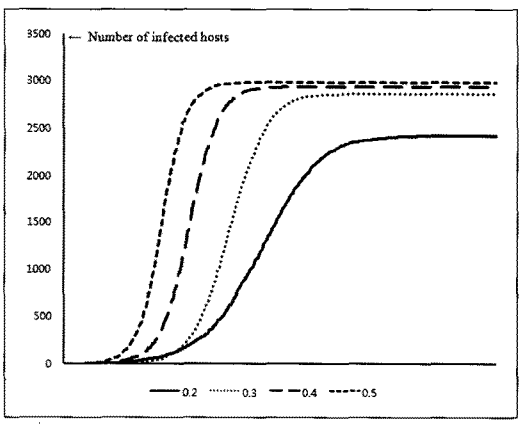

그림 6 향상된 호스트와 게이트웨이를 적용하지 않았을 때 웜 생성 빈도에 따른 확산

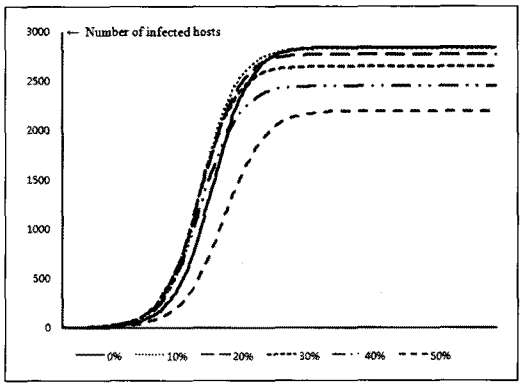

그림 7 웜 생성 빈도가 0.3일 때 향상된 호스트 비율에 따른 확산

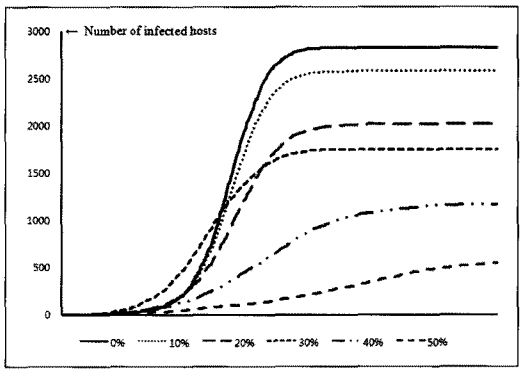

그림 8 웜 생성 빈도가 0.3일 때 향상된 게이트웨이 비율에 따른 확산

흐름으로서의 의미만 가진다.

그림 7은 웜 발생 빈도를 0.3으로 고정하였을 때 향상된 호스트의 비율에 따른 웜 확산 시뮬레이션 결과이다. 그림 7을 통해 제안된 IP기반 웜 탐지 기법이 웜의 확산 속도를 늦추고 평형상태에 도달했을 때 전체 감염된 사용자 수를 감소시키는 것을 알 수 있다.

그림 8은 제안된 기법이 적용된 게이트웨이의 비율이 달라질 때 웜 확산 추이를 나타낸 것이다. 웜 생성 빈도는 0.3으로 고정되었다. 그림 8에 의해 제안된 기법을 적용한 게이트웨이의 비율에 따른 웜 확산 저지 효과를 확인할 수 있다. 그림 7과 비교했을 때 같은 비율의 경우 호스트보다는 게이트웨이를 향상시키는 것이 웜 확산 억제에 효과적인 것을 확인할 수 있다. 이는 인터넷의 위상적 특징이 게이트웨이가 호스트를 거느리는 형상으로 나타나기 때문으로 해석된다. 이러한 현상은 향상된 호스트와 게이트웨이가 동시에 적용되는 경우를 통해 보다 구체적으로 확인할 수 있다.

그림 9는 향상된 호스트와 게이트웨이의 비율을 각각

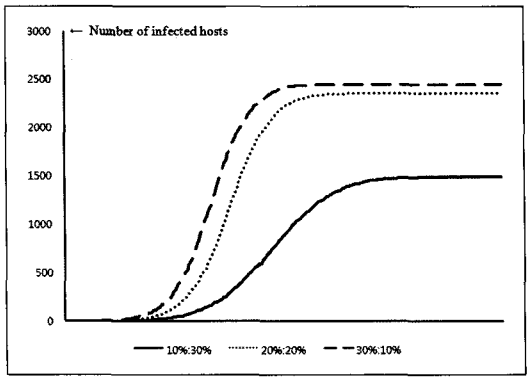

그림 9 향상된 호스트와 게이트웨이의 비율을 달리한 경우의 감염 추이(호스트의 비율 : 게이트웨이의 비율)

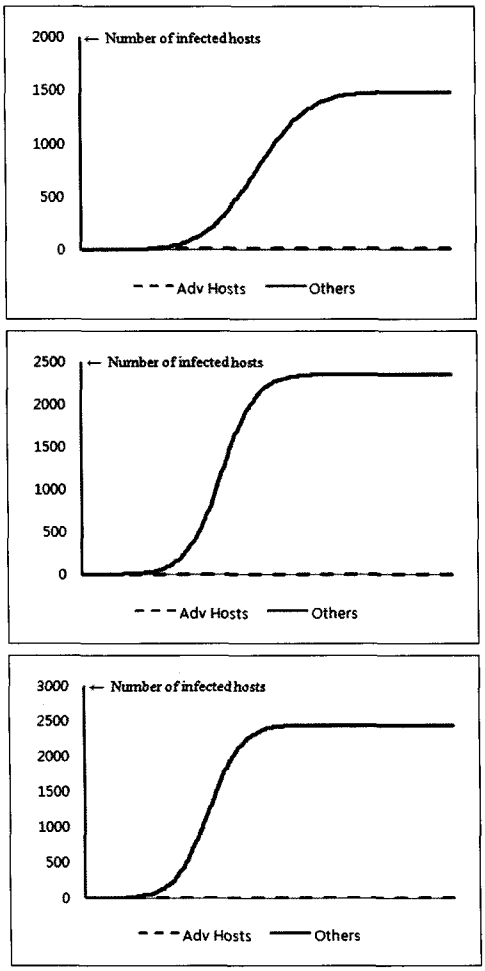

그림 10 향상된 호스트 및 향상된 게이트웨이에 속한 호스트와 나머지 호스트로 분리한 경우의 감염 추이. 그림 위부터 아래로 향상된 호스트 대 게이트웨이 비율 10%:30%, 20%:20%, 30%:10%

달리하면서 시뮬레이션 한 결과이다. 웜 발생 빈도는 0.3으로 고정되었다. 그림 9의 결과를 보면 향상된 호스트의 비율이 높을 때보다 향상된 게이트웨이의 비율이 높을 때 웜 억제 효과다 뛰어난 것을 확인할 수 있다. 그림 10은 그림 9의 결과를 향상된 호스트 또는 향상된 게이트웨이에 속한 호스트와 그렇지 않은 호스트로 나누어 표시한 것이다. 세 경우 모두 향상된 호스트이거나 향상된 게이트웨이에 속한 호스트들이 그렇지 않은 호스트에 비해 웜 감염 비율이 월등히 낮은 것을 알 수 있다. 그림에서 나타난 향상된 경우의 감염호스트 수치는 약 10에서 50정도로 전체 호스트의 수에 비해 미미한 수치이므로 그래프 상에서 거의 나타나지 않고 있다.

이러한 시험결과들은 향상된 호스트와 향상된 게이트웨이의 비율이 결과에 큰 차이를 준다는 것을 보여준다. 그림 7의 경우 향상된 호스트의 비율이 50%일 때 향상된 호스트를 포함한 전체의 절반 이상 호스트가 감염된 반면 그림 9, 10에서 향상된 게이트웨이의 비율이 추가됨에 따라 감염되는 호스트의 비율은 급격히 감소되었으며 향상된 호스트의 감염비율은 더욱 확연하게 줄었다. 이 실험은 향상된 게이트웨이가 웜 전파 억제 효과에 얼마나 큰 영향을 주는지 알 수 있게 해준다.

5. 제안된 기법의 구현 및 수행 시간 분석

제안된 기법은 인터넷 웜을 탐지하기 위한 독립적인 시스템을 필요로 하지 않고 현재 널리 사용 중인 일반적인 PC와 네트워크 게이트웨이 단에 설치되도록 고안되었다. 그런데 네트워크와 이에 연결되어 있는 호스트상에는 항상 다량의 트래픽이 흐르고 있으므로 제안된 기법에 의한 성능저하가 불가피하다. 따라서 본 절에서는 제안된 기법이 실제로 구현된 시스템 상에서 어느 정도의 성능저하가 일어나는지 측정된 값으로 제시하고자 한다.

제안된 시스템의 구현은 윈도우의 드라이버 개발을 위해 사용되는 모델인 WDM(Window Driver Model)에 의해 이루어졌으며 WinDDK 6001.18001을 이용하여 빌드 되었다. 제안된 시스템의 구현을 위하여 WDM에서 제공되는 패스스루(passthru) 샘플 코드가 이용되었다. 이를 이용하여 NDIS(Network Driver Interface Specification)의 중간계층 드라이버(Intermediate Driver)로 제안된 시스템을 구현할 수 있었다. 한편 파일 생성을 감시하는 모듈은 파일감시(filemon)를 응용한 파일 시스템 모니터 드라이버 형태로 제작되었다. 각 윈도우 드라이버에서 보내는 데이터를 화면에 출력하고 관리하기 위한 테스트용 콘솔 어플리케이션을 제작하여 UI(User Interface)로 두었다.

구현된 시스템의 동작 시나리오는 다음과 같다.

NDIS 드라이버 모듈은 외부로부터 유입되는 패킷 경

로를 모니터링하며 패킷이 유입되면 마킹 카운트가 있는지 확인한다. 만약 마킹 카운트가 있다면 해당 패킷을 받은 프로세스의 번호와 마킹 카운트를 UI에 전달해서 저장해 놓는다.

파일 모니터 드라이버는 파일 생성 이벤트가 일어날 때마다 파일을 생성시키는 프로세스의 번호가 현재 UI에 저장되어 있는지 확인한다. 만약 매칭되는 프로세스 번호가 있다면 파일 모니터 드라이버는 그 파일이 실행 파일인지 확인한다. 실행 파일이 맞는다면 UI에 현재 저장되어 있는 마킹 카운트를 증가 시키도록 알린다.

UI는 마킹 카운트 값이 증가 되었을 때 임계값을 넘는지 판단하고 임계값을 넘으면 웜으로 판단하여 사용자에게 알린다. 마지막으로 임계값을 넘지 않고 마킹 카운트 값이 증가되었다면 NDIS는 외부로 발송되는 벌크 패킷 전달에 대해 증가된 카운트를 마킹하여 전송한다.

상술된 웜 탐지 과정에서 파일 모니터 드라이버가 새롭게 생성된 파일이 실행파일인지 확인하는 모듈은 간단한 PE파서를 통해 구현하였다. 윈도우에서 실행되는 모든 실행파일은 PE포멧을 이용하여 저장되므로 파일의 확장자에 상관없이 파일의 시작부분부터 일정 오프셋 떨어진 부분에 PE포멧임을 알리는 시그너쳐 값이 들어 있다. 제안된 시스템에서는 위의 규칙을 확인하는 모듈을 작성하여 새롭게 생성된 파일이 PE포멧인지의 여부를 확인하였다.

표 2 구현된 PE파서에 의한 오탐율

	파일종류	확장자	검사파일수	오탐
1	이미지 파일	jpg, gif	325	0
2	움직이는 gif	gif	145	0
3	동영상	avi	53	0
4	음악	mp3	127	0
5	플래시	flv, swf	78	0
6	DLL	dll	1593	1
7	EXE	exe	414	1

구현된 PE포멧 파서는 일반적인 시스템 상에 존재하는 실행 파일과 기타 그렇지 않은 파일에 대해 표 2와 같은 탐지 결과를 보였다.

표 2에 의하면 구현된 PE파서는 대부분의 실행파일(DLL 및 EXE 파일)에 대해 실행파일이라는 결론을 내렸으며 실행파일이 아닌 경우의 파일에 대해서는 오탐이 없었다. 탐지에 실패한 DLL파일과 EXE파일은 모두 윈도우XP용이 아닌 파일로 밝혀졌으며 실제로 실행이 불가능한 것들이었다.

한편 제안된 탐지 시스템이 실제로 웜을 탐지하는지 확인하기 위해 웜을 이용해 테스트한 결과는 표 3과 같다. 탐지 결과 비교를 위해 국내에서 많이 이용되고 있는 A사의 안티바이러스 시스템을 사용하였다.

표 3의 결과는 제안된 시스템이 PE 포멧인 웜을 잘 탐지하는 것을 보여주며 PE 포멧이 아닌 웜은 탐지하지 못하고 있음을 나타낸다.

표 3 실제 웜을 이용한 탐지 결과

웜	PE Format	제안된 탐지 시스템	A사 백신
Worm.Win32.Aidid	O	O	O
Worm.Win32.Sluter	O	O	O
Worm.Win32.VB.a	O	O	O
Worm.Win32.Dowin	X	X	X

상용 안티바이러스 시스템도 마찬가지로 PE포멧이 아닌 웜은 탐지해내지 못하는 것을 알 수 있다. 그러나 제안된 시스템은 상용 안티바이러스와 달리 알려지지 않은 웜을 탐지할 수 있다는 데에 큰 차이가 있다. 상용 안티바이러스의 경우 시그너쳐 DB에 포함된 웜 만을 탐지할 수 있지만 제안된 기법은 알려지지 않은 웜도 네트워크를 통해 벌크 전파되는 특성을 이용해 탐지할 수 있다.

표 2와 표 3의 결과를 통해 제안된 시스템이 네트워크를 통해 유입된 웜을 비교적 정확하게 탐지 한다고 할 수 있다. 대부분의 웜은 PE포멧 형태의 파일로 존재한다. 제안된 시스템은 네트워크를 통해 유입되는 파일이 PE포멧인지 판별하고 벌크 전달 감지를 통해 웜을 판단하므로 비교적 정확할 뿐 아니라 알려지지 않은 웜에 대해서도 탐지 가능하다.

상술된 바와 같이 구현된 드라이버 소프트웨어는 일반적인 사용자들이 많이 사용하는 시스템을 가정하여 보급형의 시스템을 통해 성능 측정을 실행하였다. 성능 측정에 사용된 시스템의 사양은 표 4와 같다.

표 4 성능 측정에 사용된 시스템 사양

CPU	Intel Pentium4 2.8GHz
Memory	1792MByte
운영체제	Windows XP (Service Pack 3)

구현된 웜 탐지 모듈은 평상시에는 수신된 패킷 헤더에서 특정 필드의 마킹만을 체크하므로 수행시간이 길지 않고, FTP등을 통해 정상적인 실행파일이 유입될 때에도 시스템과 네트워크에 큰 부하를 주지 않는다. 이를 증명하기 위해 실제 FTP 서버로부터 실행파일들을 전송 받은 후 네트워크 모니터 드라이버와 파일 시스템 모니터 드라이버 UI에서 지연 시간을 측정하였다.

드라이버에서의 지연 시간을 측정할 수 있는 최소 단위는 마이크로초이다. 제안된 시스템에서는 구현된 모듈

의 동작 시간을 윈도우에서 제공하는 API(Application Programming Interface) 함수를 이용하여 마이크로초 단위로 측정하였다. 측정 시점은 네트워크 모니터 드라이버가 송신 패킷을 처리할 때와 수신 패킷을 처리할 때로 나누어 측정하였다. 측정결과는 표 5와 같다.

표 5 네트워크 모니터의 실행 시간 측정(마이크로초 단위)

회수	Send	Receive	회수	Send	Receive
1	101	19	16	38	14
2	67	38	17	69	18
3	97	34	18	46	42
4	103	40	19	66	12
5	22	8	20	47	18
6	47	16	21	71	18
7	22	2	22	78	22
8	47	7	23	108	14
9	48	17	24	69	14
10	47	15	25	69	32
11	30	10	26	70	62
12	50	42	27	69	43
13	72	43	28	70	18
14	97	43	29	117	21
15	98	5	30	117	3
평균시간				68.4	23

각각 30회의 측정결과 송신 패킷, 수신 패킷 처리 시간에 대해 평균적으로 1 밀리 초 이하가 나왔다. 이를 통해 구현된 시스템이 짧은 시간 안에 동작됨을 알 수 있고 사용자가 인식할 만한 부하는 없을 것으로 판단된다.

파일 시스템 모니터 드라이버 또한 네트워크 드라이버와 같이 시스템 시간을 측정하여 마이크로초 단위의 지연 시간을 측정하였다. 측정은 파일 시스템 드라이버의 가로채기(hooking) 루틴에서 실제로 PE 파일 형식을 확인하기 위해 IRP를 가로채는 시점에서 이루어졌다. FTP로부터 실행파일들을 받아 처리하는 파일 시스템 드라이버의 지연 시간을 측정한 결과 역시 1밀리 초 이하가 나왔다. 파일 시스템 또한 마이크로 초 단위의 수행 시간이 소요됨을 알 수 있다.

결론적으로 각 드라이버들과 에이전트 사이의 상호 데이터 교환과 드라이버에서 수행되는 일련의 작업들은 많은 부하를 주지 않는 것을 확인할 수 있다. 네트워크 드라이버는 수신된 패킷의 마킹 카운터 확인, 송신 패킷의 마킹 카운터 삽입과 포트 번호만을 추출하기 때문에 지연 시간이 크지 않으며, 파일 시스템 모니터 드라이버 또한 시스템 상에 쓰이는 파일들 중 PE 형식인 것에 대해서 프로세스 ID만을 추출하기 때문에 많은 부하를 주지 않는다. UI 또한 현재 마킹된 패킷을 수신한 프로세스 ID를 받아서 파일시스템으로부터 보고된 프로세스 ID와 단순히 비교하는 작업과, 마킹 카운터의 값을 증가시키고 마킹 카운터의 값이 임계치 이상인지 확인하는 작업으로만 이루어져 있으므로 수행 시간이 길지 않다.

결과적으로 제안된 시스템은 각각의 모듈이 모두 밀리 초 이하의 수행시간을 가지므로 실제 사용자가 느낄 만큼의 시스템 성능 저하를 가져오지 않음을 알 수 있다.

6. 결 론

인터넷 웜은 빠른 속도로 전파되므로 짧은 시간에 큰 피해를 일으킨다. 그러나 기존의 탐지 방식은 빠른 대응이 가능할 경우 정확도가 떨어지고, 정확한 탐지를 위해서는 분석을 위한 시간이 요구되는 단점이 있다.

본 논문에서는 알려지지 않은 웜을 조기 탐지(early detection) 할 수 있는 거시적 기법을 제안하고, 이 기법이 웜을 어느 정도 억제할 수 있는지 시뮬레이션을 통해 보였다. 또한 제안된 기법을 일반적인 네트워크상에서 구현하기 위한 시스템의 모델을 제시하고 이를 구현하였다.

제안된 기법은 웜 탐지를 위해 벌크 전달과 전달 체인을 웜의 일반적인 특성으로 정의하였다. 벌크 전달과 전달 체인을 이용하면 웜에 의한 트래픽을 정상적인 트래픽과 구분할 수 있고, 결과적으로 웜을 탐지 할 수 있다. 제안된 기법은 기존의 분류 방식에 따르면 트래픽 분석 기법에 속한다.

본 논문에서는 제안된 기법을 IP 패킷 네트워크상에서 실제적으로 구현하는 방안을 제시하였다. 또한 기법에 대해 시뮬레이션을 수행하여 제안된 기법의 설치 정도에 따른 웜 확산 억제 효과를 보였다. 시뮬레이션 결과는 제안된 웜 탐지 시스템의 설치 정도에 따라 차별적인 웜 억제 효과가 있는 것으로 나타났다. 특히 동일한 시뮬레이션 환경에서 기존의 웜 탐지 기법과 제안된 기법의 결과를 비교하였을 때 시스템 설치 비율에 비해 제안된 기법이 웜 억제에 효과적인 것을 알 수 있었다.

IP 패킷 네트워크상에서의 구현은 제안된 시스템을 호스트에 탑재하여도 성능 저하가 크지 않다는 것을 보이기 위해 제안된 기법을 수행할 때 소요되는 시간을 측정하고 이 값을 제시하였다. 이후 제안된 기법을 실제 시스템에 구현하여 웜을 탐지하는 방안에 대한 후속연구 및 개발이 가능할 것이다.

제안된 기법을 인터넷에 인프라 규모로 설치하게 되면 웜 확산의 초기 단계에 이를 탐지하고 차단할 수 있으므로 웜에 의한 피해를 최소화하는데 기여할 수 있을 것으로 기대된다.

참 고 문 헌

[1] Jose Nazario, Jeremy Anderson, Rick Wash, Chris Connelly, "The Future of Internet Worms," Cri-

melabs research, 2001.
[2] Jose Nazario, "Defense and Detection Strategies against InternetWorms," *Artech House*, pp.135-208, 2004.
[3] Frederick B. Cohen, "Computer Viruses: Theory and Experiments," Computers and Security 6, 1987.
[4] Eugene H. Spafford, "Computer Viruses as ArtificialLife," *Journal of Artificial Life*, MIT Press, pp.249-265, 1994.
[5] K Alamar, "VBS Worm Generator," http://vx.netlux.org/vx.php?id=tv07, 2000.
[6] B. K. Mun, "Vision-Power Malicious Code Analisys. Increase of Malicious Code for Wresting Information," http://www.etnews.co.kr/news/detail.html?id=200804300118, 2001.
[7] Charles Schmidt and Tom Darby, "The What, Why, and How of the 1988 Internet Worm," http://snowplow.org/tom/worm/worm.html, 2001.
[8] Song, D., R. Malan, and R. Stone, "A Snapshot of Global Worm Activity," http://reserch.arbor.net/up_media/up_files/snapshot_worm_activity.pdf, 2001.
[9] Pele Li, Mehdi Salour, and Xiao Su, "A Survey of Internet Worm Detection and Containment," *IEEE Communication Surveys*, 1st quarter 2008, vol.10, no.1, 2008.
[10] Jeffrey O. Kephart, David M. Chess, Steve R. White, "Computers and Epidemiology," *IEEE SPECTRUM*, vol.30, no.5, pp.20-26, 1993.
[11] K. Lee, C. Kim, S. Lee, M. Hong, "Macroscopic Treatment to Unknown Malicious Mobile Codes," *Journal of KIISE: Computing Practices*, vol.12, no.6, pp.339-348, Dec. 2006. (in Korean)
[12] C. M. Kim, S. U. Lee, M. P. Hong, "Macroscopic Treatment to Polymorphic E-mail Based Viruses," *Proc. of the KIISE Korea Computer Congress 2003*, vol.30, no.1(A), pp.419-421, 2003 (in Korean)
[13] A. Perrig, D. Song and A. Yaar, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," *Proceedings of the 2003 Security and Privacy Symposium*, 2003.
[14] Sarah H. Sellke, Ness B. Shroff, Saurabh Bagchi, "Modeling and Automated Containment of Worms," *IEEE Transaction on Dependable and Secure Computing*, vol.5, no.2, pp.71-86, April 2008.
[15] Sapon Tanachaiwiwat and Ahmed Helmy, "Modeling and Analysis of Worm Interactions(War of the Worms)," *Broadnets 2007, Fourth International Conference on*, pp.649-658, Sept. 2007.
[16] Ram Dantu and Joa˜o W. Cangussu, "Fast Worm Containment Using Feedback Control," *IEEE Transaction on Dependable and Secure Computing*, vol. 4, no.2, pp.119-136, April 2007.
[17] Milan Vojnovic and Ayalvadi J. Ganesh, "On the Race of Worms, Alerts, and Patches," *IEEE/ACM Transaction on Networking*, vol.16, no.5, pp.1066-1079, October 2008.

김 철 민
2000년 8월 아주대학교 정보 및 컴퓨터공학과(학사). 2002년 8월 아주대학교 대학원 정보통신공학과(석사). 2009년 3월 아주대학교 대학원 정보통신공학과(박사). 2008년 1월~현재 (주)모비안 부설 연구소 연구소장

강 석 인
2008년 아주대학교 정보 및 컴퓨터공학부 졸업(학사). 2008년~현재 아주대학교 대학원 컴퓨터공학과 석사과정

이 성 욱
1994년 2월 아주대학교 공과대학 컴퓨터공학과 학사. 1996년 2월 아주대학교 대학원 교통공학과 석사. 2003년 2월 아주대학교 대학원 컴퓨터공학과 박사. 2003년 3월~현재 신구대학 인터넷정보과 조교수

홍 만 표
1981년 서울대학교 계산통계학 전공(학사). 1983년 서울대학교 계산통계학 전공(석사). 1991년 서울대학교 병렬처리 전공(박사). 1983년~1985년 울산공과대학 전임강사. 1993년~1994년 미네소타 대학 교환 교수. 2000년~2001년 조지워싱턴 대학 교환 교수. 1985년~현재 아주대학교 교수