산업기술연구(강원대학교 산업기술연구소 논문집), 제22권 B호, 2002.
Journal of Industrial Technology, Kangwon Natl. Univ, Korea, No. 22 B, 2002.

역추적 에이전트를 이용한 역추적 시스템 설계 및 구현

Design and Implementation of a Traceback System based on a Traceback Agent

정 종 민* 이 지 율** 이 구 연***
Jeong, Jong-Min Lee, Ji-Yul Lee, Goo-Yeon

Abstract

It is very important to detect and remove original sources of DOS (Denial of Service) attacks or connection oriented/connectionless attacks. In this paper, we implement a traceback system that does not require the reaction of routers and administrators and does not need log data. We bring in a traceback server and traceback agents in each network and use sniffing and spoofing schemes. Finally, the traceback server detects attacking hosts using information transmitted from traceback agents.

키워드 : *역추적서버, 역추적 에이전트, 스니핑, 스푸핑*
Keywords : *traceback server, traceback agents, sniffing, spoofing*

1. 서론

DOS와 각종 스캐닝 공격 등은 원격 호스트 및 네트워크의 자원을 낭비하게 한다. 이러한 공격은 공격자 입장에서 매우 쉽게 구현될 수 있고, 방지하기 힘들며 추적하기 매우 어려운 형태이기 때문에 해결하기 쉽지 않으며,[1] 스캐닝과 DOS 등의 수동적 공격과 아울러 연결 설정을 통해 불법적으로 서버에 침입하여 자원을 도용 및 훼손시키는 전통적인 공격 형태도 여전히 심각한 보안 문제에 해당된다. 특히 IP 스푸핑(spoofing)에 의한 익명성(anonymity)은 공격 근원지 결정을 더욱 어렵게 하고 있다.

공격 근원지를 결정하는 것을 역추적이라 하는데, 공격자는 일반적으로 자신의 위치를 은폐하기 위해 IP 소스 주소를 스푸핑하게 된다. 또한 인터넷 프로토콜 특성에 의해 인터넷 라우팅 상태 정보가 유지되지 않기 때문에 공격 경로를 역추적하여 공격 근원지를 결정하는 것은 많은 작업을 요구하게 된다. 현재 IP 역추적을 위한 몇 가지 구조가 있는데, 대체로 라우터 기능을 이용하는 형태, 로그 기록을 사용하는 것 그리고 링크 테스트 방식으로 구분 될 수 있다.

본 논문에서는 연결형 서비스에서 이행적 신뢰 관계를 악용하여 시스템을 공격하는 경우, 공격 호스트를 검출하기 위해 역추적 서버와 역추적 에이전트를 사용 하는 역추적 시스템을 구현하였다. 본 구현은 기존 라우터 방식의 역추적 시스템과 달리 라우터와 관리자의 동작을 요구하지 않으며, 전체 로그를 저장함으로써 생기는 자원의 오버 헤드를

* 강원대학교 컴퓨터정보통신공학과 박사과정
** 강원대학교 전기전자정보통신공학부 학사졸업
*** 강원대학교 전기전자정보통신공학부 부교수, 공학박사

줄일 수 있다. 동작 절차를 간략히 살펴보면, 침입 탐지 시스템에 의해 공격이 탐지 될 경우 역추적 서버가 서브 네크워크 내의 패킷을 스니핑하여 스푸핑을 한 후 피해 호스트의 주소와 TCP 시퀀스를 가장하여 공격자와 세션을 유지하면서 특정 형태의 패킷을 공격 호스트에 전달한다. 이와 동시에, 역추적 에이전트는 서버가 보낸 특정 패킷을 수신하는 호스트를 추적하면서 공격 호스트의 주소를 탐지하는 형태이다. 다수의 네트워크를 경유할 경우 인접 네트워크에 있는 역추적 에이전트들에 의해 정보를 공유하게 되고, 네트워크 내의 에이전트에 기반 하여 각 네트워크에 포함된 경로 정보를 획득 한 후, 최종적으로 이 경로 정보를 사용하여 역추적 서버는 공격 호스트를 결정하게 된다.

역추적 에이전트는 역추적 서버에서 공격 호스트까지의 경로를 구성할 수 있는 정보를 제공하게 되는데 이는 스니핑과 스푸핑을 통해 공격 시스템과 통신하는 역추적 서버의 부하를 줄여 줄 수 있으며, 상이한 네트워크의 역추적 에이전트 간의 경보 공유를 제공할 수 있어, 네트워크가 확장될 경우에도 적용이 가능하다.

2장에서는 역추적 기법에 대한 기존의 연구 및 동작 형태를 정리하며, 3장에서는 구현된 역추적 시스템의 모델 및 구성 요소를 설명한다. 4장에서는 구현된 역추적 시스템의 동작 모습을 제공한 후 5장에서 결론을 맺는다.

2. 관련 연구

현재까지 제안된 역추적 기법은 앞서 언급했듯이 라우터의 기능을 활용하는 방식, 로그 데이터를 저장한 후 이 정보를 이용하여 추후 공격 경로 결정에 사용하는 형태, 그리고 일정량의 트래픽을 전송하여 링크를 테스트 하는 방식, 마지막으로 마킹(marking) 기법이 있다. 이 장에서는 이러한 특성으로 분류 될 수 있는 기존의 역추적 기법에 대해 정리한다.

2.1. DosTracker

1996년 MCI는 시스코 라우터의 이전 링크로 패킷의 흐름을 따라가기 위한 목적으로 Perl 스크립트를 개발하였다. 사용자는 DosTracker를 적용하고자 하는 시작 라우터의 위치, 피해 컴퓨터의 주소 그리고 선택적으로 Dostracker을 실행하기 위한 소스 주소를 입력하게 되며, 프로그램은 시작 라우터에 로그온 하여 디버그 모드로 전환 한 후, 접근 제어 리스트(access control list)를 적용하게 된다. 임의 패킷이 피해 컴퓨터의 네트워크에 진입하게 되면, 디버그 상태 정보에 의해 이 패킷들이 탐지되며 DosTracker은 디버그 정보를 분석하여 이전 경로를 결정하며, 이 절차를 반복적으로 수행하여 최종적으로 공격 호스트를 탐지하게 된다.[2]

2.2 Traceback Logging Protocol

이 프로토콜은 라우터에게 임의 메시지를 표시하도록 요구하지는 않지만, 라우터를 통과하는 모든 메시지를 일정 시간 동안 저장해야 한다. 이러한 접근 형태는 핵심 라우터에서 패킷을 저장한 후에 패킷이 이동한 경로를 결정하기 위해 로그 데이터에 대해 마이닝 기술을 사용 하게 된다. 이 방식은 공격이 발생한 후 상당한 시간 동안 추적할 수 있으나 많은 양의 자원을 요구하게 되는 단점이 있다.[1][3]

2.3 Link Testing Protocol

이 프로토콜은 피해 컴퓨터와 가장 근접한 라우터에서부터 시작하게 되며, 공격 패킷을 발생시키는 시스템을 알아낼 때까지 계속적으로 상위 링크를 검사하게 된다. 하지만 이러한 접근은 공격자가 역추적을 감지하여 일시적으로 공격을 중단하거나 역추적에 대한 대응을 할 경우 정상적인 동작이 이루어지지 않는다[1][3]. 이 방식에는 input debugging 방식과 controlled flooding 이 있다.

(1) Input Debugging

대부분의 라우터들은 관리자가 몇몇 출력 포트에 특정한 패킷을 필터링 하고, 그 패킷이 도착한 입력 포트를 결정할 수 있는 input debugging 기능을 가지고 있다. 이러한 기능이 경로 추적을 구현하는데 사용되는데, 먼저 피해 시스템은 공격을 감지하고, 모든 공격 패킷들에 있는 공통적인 특성을 규정할 수 있어야 한다. 그런 후, 피해 시스템은 input debug 필터를 설치하고 있는 라우터 관리자와 공격 패킷에 대해 상호 협력하여 이 공통적인 특성을 지닌 패킷들을 라우터 출력 링크에서 분별해 내고 이들의 입력 링크를 찾아내어, 원래의 사이트를 검출하거나 ISP의 경계에 도달할 때까지 이를 반복하게 된다. 하지만 이 방식의 문제점은 관리적인 측면에서 상당한 오버헤드가 발생하는 것이다.[2]

(2) Controlled Flooding

Burch와 Cheswick는 네트워크 관리자의 지원을 요구 하지 않는 링크 검사 역추적 기술을 개발하였다. 이 기법은 많은 양의 트랙픽을 통해 링크를 폭주시켜서 공격자의 패킷이 어떠한 경로로 관찰되는냐를 검사하는 형태이다. 하지만 이 경우에도 공격자를 탐지하기 위해 발생시키는 트래픽 자체가 DOS 공격이 될 수 있으므로 널리 사용되지는 않는다.

2.4 Traceback Marking 프로토콜

이 프로토콜은 라우터를 통하여 전달되는 메시지 내에 특정 정보를 추가한 후 공격받고 있는 호스트와 송수신하는 호스트의 패스를 결정하여 공격 경로를 알아내는 방식이다.[1][3]

Burch와 Cheswick은 확률적이든 결정적이든 마킹 패킷을 이용하여 공격자를 추적하는 가능성에 대해 언급하였다. 피해 호스트는 공격자를 역추적하기 위해 마크 된 패킷 내의 정보를 사용하게 되며, 이는 잠재적인 많은 이점을 지니고 있다. 이러한 방식은 ISP와의 상호 협력 작용을 요구하지 않기 때문에 input debugging의 과도한 관리적 부담을 덜 수 있으며, controlled flooding과 달리 추가적인 네트워크의 트래픽을 요구하지 않으며 다수 공격자의 추적도 가능하다. 또한 로깅 방식과 마찬가지로 공격이 중단된 후에도 추적이 가능하다.[1]

이 방식의 가장 기본적인 형태는 node append 방식이다. 이는 공격자에서 피해 호스트까지 네트워크 전체를 거쳐 경유하는 패킷의 끝에 단순히 각 노드의 주소를 추가하는 것이다. [표 1]은 node append 알고리즘을 나타내고 있다.

표 1. node append 알고리즘

marking procedure at router R:
for each packet w, append R to w
path reconstruction procedure at victim v:
for any packet w form attacker
extract path(R_i~R_j) from the suffic of w

이에 비해 node sampling 방식은 node append 방식의 전체 경로 기록이 아닌 경로를 간략화하여 라우터의 부담을 줄이고, 패킷 당 요구하는 공간을 줄일 수 있게 된다. 마지막으로 edge sampling 알고리즘이 있는데, IP 옵션 부분을 사용하여 각 패킷 내에 IP 주소 크기와 동일한 두개의 정적 필드인 *start*와 *end* 그리고 한 개의 정적 필드 *distance*를 정의한다. 각 라우터는 확률 p로 패킷을 마크하며, 라우터가 마크를 결정할 경우, 자신의 IP 주소를 *start* 필드에 기입하고 *distance*는 0으로 기입한다. 그렇지 않고 *distance* 필드가 이미 0으로 되어 있을 경우 이는 이전의 라우터가 패킷을 마크 했음을 의미하는데, 이 때는 자신의 IP 주소를 *end* 필드에 기입하여, 자신과 이전 라우터간의 edge를 표현한다. 마지막으로 라우터가 패킷을 마크하지 않을 경우 *distance*를 증가하게 되는데, 이 *distance* 필드가 마크한 라우터에서부터 피해 호스트까지의 라우터 수를 가리키게 된다. 이러한 기법을 통해 공격자에 의해 생성된 패킷은 실제 공격 패스의 길이보다 크거나 동일한 *distance*를 갖게 된다.

모든 마킹 알고리즘은 두개의 요소를 포함하고 있는데, 하나는 라우터에 의해 실행되는 마킹절차와 다른 하나는 피해 호스트에서 구현되는 경로 재구성 절차이다. 최종적인 알고리즘의 수렴 시간은 피해 호스트가 공격 패스를 재구성하는데 관찰해야 하는 패킷의 수이다.[1]

2.5 ICMP Traceback Protocol

이 프로토콜은 라우터에서 생성한 ICMP 역추적 메시지에 기초를 두고 있으며, 이 방식은 모든 라우터들에 대해서 특정 ICMP 역추적 메시지 내의 내용을 복사하여 포워딩하면서 이루어진다. 이 접근은 꽤 효율적이긴 하나 공격자가 ICMP 역추적 메시지를 위장하여 피해 호스트에 전송할 수도 있는 단점이 있다.[3]

2.6 Ingress 필터링

Ingress 필터링이란 불법적인 소스 주소를 갖는 패킷을 라우터에서 차단하는 것을 의미하는데, 이는 라우터가 모든 패킷에 대해 소스 주소를 검사하며, 불법적인 패킷과 그렇지 않은 것에 대한 구분을 할 수 있는 충분한 정보를 포함하고 있어야 한다. ingress 필터링 방식은 현재 일반적인 네트워크나 ISP의 border에서 적용 가능한 것으로 평가되나, 이 방식의 효과가 관리자와 ISP에서의 지원 여부에 따라 상이할 수 있는 문제가 있으며, 관리적인 문제와 라우터의 오버헤드, 그리고 mobile IP와 같은 서비스에서 매우 복잡하다. 또 하나의 단점은 ingress 필터링이 일반적으로 적용이 된다 하더라도 공격자는 여전히 검증된 네트워크의 내부 주소로 위장할 수 있다.

지금까지 기존의 역추적 방식에 대하여 살펴보았는데, 라우터 방식의 경우 관리자의 동작을 요구하게 되며 또한 ingress 필터가 구현된 경우에는 라우터의 오버헤드를 증가시키는 문제가 있으며, 로깅 방식의 경우에는 공격이 발생한 후에야 경로 추적을 시작하게 되며, 로깅 정보의 저장에 따른 오버헤드가 발생한다. 링크 테스트 방식의 경우에도 역으로 발생시키는 트래픽이 네트워크의 과부하를 발생시키는 문제가 있다. 마지막으로 전송 경로를 패킷에 표시하는 마킹 방식의 경우에도 구현의 어려움이 따른다.

이에 본 구현에서는 관리자의 동작을 요구하지 않으며, 적은 양의 로깅 정보를 사용하여, 기본적인 마킹 개념을 적용하여 역추적 시스템을 구현하였는데, 다음 3장에서 시스템에 대한 설계 및 동작 구조를 설명한다.

3. 시스템 설계

이 장에서는 구현된 시스템에 대한 설명과 시스템의 구성 요소, 공격 시나리오에 대한 시스템의 동작 과정을 묘사한다.

3.1 시스템 설계

본 논문에서 구현한 역추적 시스템은 로깅방식과 기본적인 마크방식이 결합된 형태이다. 하지만 일반적인 로깅방식이 모든 패킷 정보를 기록하는 것에 반해 본 시스템은 경로 결정에 이용되는 주소만을 핵심 정보로 간주하고 중복되는 값을 최소화하여 저장하는 양을 줄이게 된다. 또한 마크방식의 기본 생각이 포함되어 있는데, 기존 형태의 마크가 라우터에서 역추적 경로를 재구성 할 수 있도록 사용되는 주소에 관한 추가 정보라면, 본 구현에서의 마크개념은 ICMP 역추적 메시지와 유사한 특정 메시지를 사용하는 것인데, 이는 역추적 서버에서 생성하여 역으로 공격 호스트까지 경로 설정에 사용되게 된다. 본 구현에서는 역추적 에이전트를 이용하여 기존 라우터의 경로 재구성 등의 역할을 생략할 수 있어 네트워크에 투명하게 동작된다.

3.2 시스템 구성

전체 시스템을 구성하고 있는 주요 요소는 역추적 서버, 역추적 에이전트들 그리고 침입탐지 시스템이다.

● 역추적 서버

역추적 서버는 전체적인 동작(역추적 실행 시작/종료, 마크생성/전달, 지문(thumbprint)값 수집, 공격 호스트 결정 등)을 총괄하는 시스템으로 공격을 감지한 IDS로부터 피해 호스트와 공격 호스트의 정보를 수신하여 공격 호스트와 IDS 간의 패킷을 스니핑한 후 TCP 시퀀스를 획득하고 IP 스푸핑을 수행한다. 이와 동시에 역추적 에이전트에게 마크에 해당하는 특정 패킷을 보내어 에이전트의 동작을 요구한다. 후에 에이전트로부터 수신한 지문 값 리스트를 통해 공격 호스트를 검출하게 된다.

이를 위해 역추적 서버에서는 libcap 라이브러리를 사용하여 네트워크의 패킷을 획득한 후 스푸핑을 시작하게 된다.

● 역추적 에이전트

역추적 에이전트는 실제로 마크에 해당하는 메시지를 수신하는 호스트의 경로를 수집하여 공격 호스트의 후보 정보를 서버에게 전달하게 되며, 상이한 네트워크에 존재하는 역추적 에이전트에게 동일한 마크를 전달하여 역추적의 경로가 네트워크의 경계를 벗어 날 경우에 그 네트워크에 포함되어 있는 에이전트에게 자신의 네트워크를 담당하도록 지시한 후 다른 네트워크의 에이전트들도 후에 서버에게 공격호스트의 정보를 전달하게 된다.

● 침입 탐지시스템

본 구현에서 공격을 감지하는 역할을 하는 시스템으로 역추적 서버와 동일한 시스템 내에 있을 수도 있으며, 근접하여 위치할 수도 있다. 본 구현에서는 역추적 서버 내에 존재한다.

[그림 1]은 다수의 네트워크에서, 본 구현이 적용 가능한 형태의 시스템 모델을 나타내고 있다.

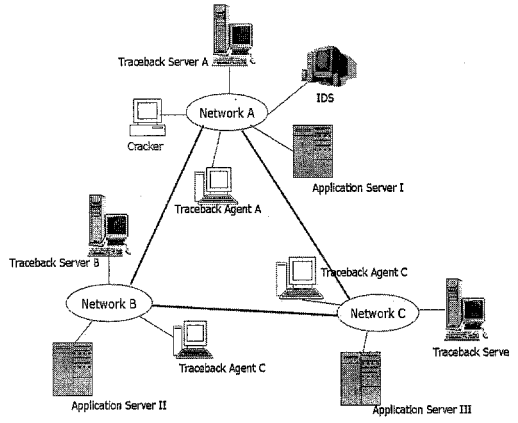

그림 1. 시스템 전체 구성도

핵심이 되는 역추적 서버와 역추적 에이전트가 각 네트워크에 있으며, 이는 라우터의 동작을 요구하지 않은 형태에서 데이터를 스니핑 하기 위한 것이며, 상이한 네트워크에 존재하는 각각의 역추적 에이전트는 마크를 공유하기 위한 것이다.

네트워크 A에 있는 IDS로부터 공격 감지를 통보 받은 역추적 서버 A는 공격자와의 연결을 유지한 채로 마크를 전달하고 멀티캐스트 그룹에 가입된 모든 역추적 에이전트들에게 이에 대한 정보를 전송한다. [그림 1]의 경우 공격자가 네트워크 A에 존재하지만, 만약 상이한 네트워크 B 혹은 C에 존재할 경우 멀티캐스팅을 통해 역추적 에이전트 B 혹은 C에게 동일한 정보를 전달하게 된다. 후에 역추적 에이전트 A, B 그리고 C는 역추적 서버 A에 자신의 네트워크 내의 공격 경로 정보를 전달하게 되면, 최종적으로 역추적 서버가 공격 호스트를 검출하게 된다.

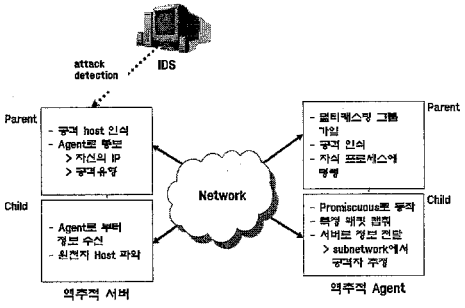

그림 2. 역추적 서버와 에이전트의 역할

[그림 2]는 역추적 서버와 역추적 에이전트의 역할을 나타낸 것으로 역추적 서버는 IDS와 역추적 에이전트 간에 상호 동작하여 공격 호스트의 정보 및 공격 호스트까지의 경유 정보를 전달받아 최종적으로 공격 호스트를 검출하게 된다. 역추적 에이전트는 역추적 서버가 보낸 특정 메시지를 포함하는 패킷을 수신하는 호스트에 대한 송수신 데이터를 획득하여 역추적 서버가 공격자를 결정하기 위한 정보를 제공하게 된다.

3.3 역추적 시스템 동작 시나리오

[그림 3]은 공격자가 여러 경유지를 이용하여 최종적으로 특정 서버를 공격할 경우를 가정하여 본 구현에 대한 시나리오를 묘사한 것이다. 구체적인 동작 절차는 다음과 같다.

① Connect: 공격자는 이행적 신뢰 관계를 악용하여 여러 경유지를 거쳐 목적지에 연결을 설정한다.

② Attacking: 목적 호스트에 DOS 혹은 기타 공격을 시도한다.

③ Cracking detection: IDS가 공격을 감지한다.

④ Send information of attacking and cracked host : 공격을 감지한 IDS는 역추적 서버에게 공격 호스트의 주소와 포트 등의 정보를 전달하게 된다.

⑤ Sniffing and spoofing: 역추적 서버는 IDS로부터 공격 감지를 전달받자마자 네트워크를 스니핑하여 공격 호스트의 송수신 패킷을 획득 한 후 TCP 시퀀스를 가로채어 스푸핑을 시도하여 공격자에게 송신 패킷에 특정 표시를 삽입한다.

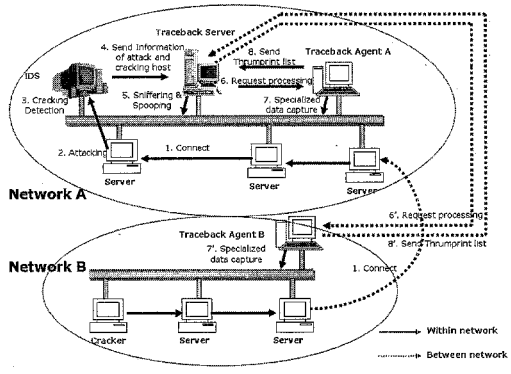

그림 3. 시스템 동작 시나리오

⑥ Request processing: 스푸핑과 동시에 역추적 에이전트에게 특정 패킷에 대한 정보를 전달한 후 이를 수신 하는 경로를 획득하도록 지시한다.

⑦ Specialized data capture: 역추적 에이전트는 서버가 보내온 데이터를 수신하는 호스트에 대한 정보를 획득하여 리스트를 구성한다.

⑧ Send thumbprint list: 역추적 에이전트는 ⑦을 통해 획득한 리스트를 역추적 서버에 전달하게 되며, 최종적으로 역추적 서버는 이를 이용하여 근원지를 탐지하게 된다.

4. 구현 및 고찰

4.1 시스템 구현

본 시스템이 구현된 환경은 펜티엄III PC 3 대에 리눅스 레드햇 v7.0/7.2(커널 v2.4)을 사용하였다.

아래 [그림 4]는 3.3절에서 묘사하고 있는 시나리오에 대한 실제 구현 코드에 대한 전체적 흐름도를 나타내고 있다.

본 시스템은 역추적 서버와 역추적 에이전트로 구분되어 나타내고 있는데, 역추적 서버는 실행 후 침입탐지 시스템으로부터 공격 감지를 통보 받으면, sniper()를 통해 네트워크를 스니핑한 후 피해 호스트의 주소와 TCP 시퀀스를 스누핑하게 된다. 그런 후에 자식 프로세스를 생성 한 후 order()를 통해 역추적 에이전트의 동작 요구와 동작에 필요한 정보를 전달하게 된다.

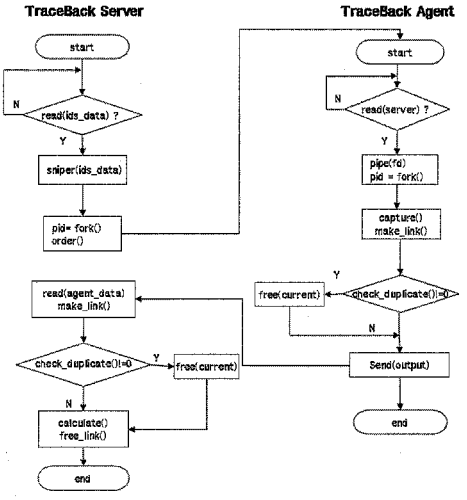

그림 4. 시스템 동작 흐름도

실행 대기로 있던 에이전트는 역추적 서버로부터 동작을 지시 받자마자 데이터 획득을 위한 자식 프로세스를 생성한 후 서버가 공격 호스트와 통신 중에 삽입되는 특정 메시지를 수신하는 호스트를 감지하게 된다. check_duplicate()와 free()로 중복된 호스트 정보를 제거하고 send()를 통해 서버로 결과 값을 전달하게 된다.

이를 수신한 서버는 다수의 에이전트로부터 수신한 정보 중 중복된 것을 필터링 한 후 calculate()를 통해 최종적으로 공격 호스트를 탐지하게 된다.

4.2 역추적 시스템 동작 과정

(1) 역추적 에이전트 동작

[그림 5]는 역추적 에이전트에서 역추적 서버가 발생시킨 특정 데이터를 수신하고 있는 호스트를 검출해 내는 모습을 나타내고 있으며, 동일한 메시지를 반복해서 송신하므로 하나 이상의 동일한 주소와 포트가 탐지되고 있음을 알 수 있다.

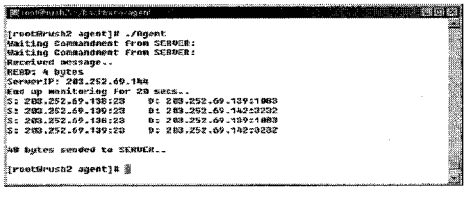

그림 5. 역추적 에이전트 실행 결과

(2) 역추적 서버 동작

[그림 6]은 역추적 서버의 실행 모습을 나타내고 있는데, IDS로부터 공격감지를 기다리고 있다가, 공격 감지 정보를 전달받으면, 스니핑을 통해 공격 호스트와 피해 호스트간의 TCP 시퀀스를 획득한 후 스푸핑을 통해 공격 호스트로 특정 데이터를 전달한 후 역추적 에이전트에게 공격자까지의 경로를 구성하도록 요구한다.

후에 에이전트로부터 수신한 공격자 리스트를 획득한 후 최종적으로 역추적 서버가 공격자를 검출해 내는 모습을 나타내고 있다.

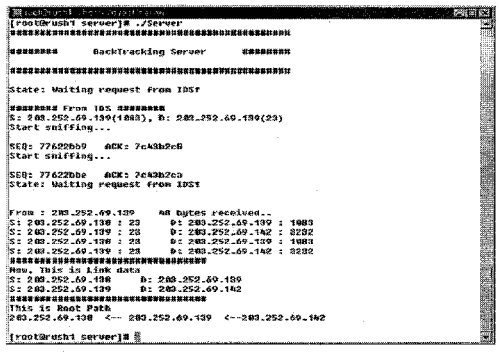

그림 6. 역추적 서버 실행 결과

4.3 구현 고찰

흔히 IP 역추적 개념은 DOS 등의 불특정 다수의 근원지에서 발생하는 공격을 추적하기 위해 주로 언급되고 있으나, 일반적인 네트워크의 공격에는 telnet, rlogin 등의 연결형 서비스 및 ftp 등의 비 연결형 서비스에 관한 사건도 무수히 발생하게 되는데, 본 구현에서는 연결형 서비스에서 효과적인 적용이 가능하다.

본 구현 시스템은 역추적 서버와 역추적 에이전트간의 역할을 분담하여 적용시켰기 때문에 각 시스템의 부하를 조절 할 수 있으며, 라우터 지원 방식에 비해 라우터의 연동을 요구하지 않게 되어 기존 네트워크 구성 요소의 수정 없이 적용이 가능하며, 로그 기록 방식에 비해 전달되는 패킷 모두를 저장하지 않고 주소만을 사용하므로 기록되는 로그 양 및 로그에 필요한 절차를 최소화하여 자원의 오버헤드를 줄이게 된다. 또한 인접 네트워크의 역추적 에이전트 간의 통신을 통해 특정 데이터의 정보를 공유하여, 이를 포함하는 패킷을 획득하므로 여러 경계를 벗어나는 네트워크들에 적용이 가능하다. [표2]는 기존의 IP 역추적 방식들과 본 구현에서 사용한 형태를 비교한 것이다.

5. 결론

본 논문에서는 역추적 서버와 역추적 에이전트를 통해 다수의 네트워크를 경유하여 크래킹을 시

도하는 공격 호스트의 근원지를 파악하기 위한 역추적 시스템을 개발하였다.

표 2. 기존 방식과의 성능 비교

	네트워크 부하	라우터 부하	관리 부하	멀티 네트워크 적용
Ingress filter	L	M	M	N
logging	L	H	H	H
Input debugging	L	H	H	M
controlled flooding	H	L	L	L
marking	L	L	L	H
본 논문 구현 방식	L	N	L	H

(L : Low, M: Moderate, H: High, N:Nothing)

본 구현에서는 역추적 서버가 피해 호스트의 주소와 TCP 시퀀스를 스푸핑하여 공격 호스트와 연결을 유지하여 패킷을 송수신 함으로써 최종적으로 공격 호스트의 위치를 파악하게 되는데, 공격 호스트가 역추적 동작을 감지하거나 임의의 사건으로 인해 피해 호스트와 연결을 종료시킬 경우 공격 호스트를 탐지하기 어렵게 된다. 즉 연결형 공격에 효율적이긴 하나 그렇지 않은 경우에는 문제점이 따르게 된다. 추후 비 연결형 공격에 적용이 가능하도록 여러 보완 방안 및 수정에 관한 연구가 요구된다.

참 고 문 헌

[1] Stefan Savage, David Wetherall, Anna R.Karlin, and Tom Anderson. "Practical Network support for IP traceback", Proceedings of ACM SIGCOMM, pp. 295-306, 2000

[2] Tom Dunigan, "Backtracking Spoofed Packets", ORNL/TM-2001/114, Oct, 2000, http://www.csm.ornl.gov/~dunigan/oci/back.ps

[3] Chun He, "Formal Specifications of Trace back Marking Protocol", Honors Thesis, May 2002, http://www.cs.utexas.edu/users/UTCS/techreports/tr02-42.pdf

[4] Alex C. Snoeren, Craig Partridge, Lusi A.Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer, "Hash-based IP Trace back", Proceddingsof SIGCOMM 2001, Aug, 2001

[5] D.Song and A.Perring," Advanced and authenticated marking schemes for IP Traceback", Proceedingsof IEEE INFOCOM, vol. 2, April2001

[6] D.Dean, Matt Franklin, Adam Stubblefield, "An Algebraic Approach to IP Traceback", http://www.isoc.org/isoc/conferences/ndss/01/2001/papers/dean01.pdf

[7] S.M.Bellovin, ICMP traceback messages, http://search.ietf.org/internet-drafts/draft-bellobin-itrace-00.txt, Mar,2000

[8] CERT coordination center denail of service attack, http://www.cert.org/tech_tips/denial_of_service.html, Feb, 1999