침입자 역추적을 위한 TCP 연결 매칭 알고리즘 설계

강 형 우[†] · 홍 순 좌[†] · 이 동 훈[††]

요 약

최근의 해킹사고에서 침입자는 피해시스템에서 자신의 IP주소 노출을 피하기 위하여 피해시스템을 직접 공격하지 않고 Stepping stone(경유지)을 이용하여 경유지 우회 공격을 수행한다. 본 논문에서는 현재 네트워크 환경에서 Stepping stone을 이용한 경유지 우회 공격시 공격자의 근원지 IP주소를 추적하기 위한 알고리즘을 설계한다. 침입자 추적은 크게 두 가지 분류로 나뉘어 진다. 첫째는 IP Packet traceback, 둘째는 Connection traceback 이다. 본 논문에서는 Connection traceback에 공격을 다루며, 운영체제의 프로세스 구조를 이용하여 공격자 또는 Stepping stone을 구분하여 침입자의 위치를 추적한다.

키워드 : 역추적, 경유지, 경유지 우회 공격, 백도어

Design an Algorithm Matching TCP Connection Pairs for Intruder Traceback

Hyungwoo Kang[†] · Soonjwa Hong[†] · Dong hoon Lee[††]

ABSTRACT

In the field of network defense, a lot of researches are directed toward locating the source of network attacks. When an intruder launches attack not from their own computer but from intermediate hosts that they previously compromised, and these intermediate hosts are called stepping-stones. There are two kinds of traceback technologies : IP packet traceback and connection traceback. We focused on connection traceback in this paper. This paper classifies process structures of detoured attack type in stepping stone, designs an algorithm for traceback agent, and implements the traceback system based on the agent

Key Words : Traceback, Stepping Stone, Detoured Attack, Backdoor

1. 서 론

최근 인터넷 사용자가 급증하면서 인터넷을 이용한 각종 해킹 및 사이버 범죄가 크게 증가되고 있으며 그와 동시에 침입 탐지 기술도 급속도로 발전하고 있다. 이와 같은 기술은 침입자가 피해시스템에 직접 연결해서 공격하는 공격에 대해서는 매우 유용하게 사용될 수 있다. 하지만 침입자가 경유지를 이용하는 경유지 우회 공격을 시도할 경우, 침입 탐지시스템은 침입자의 근원지 정보를 알아내지 못하며 바로 전단계 경유지의 정보만을 얻을 수 있다. 이런 이유로 경유지 우회 공격이 발생 시 공격 근원지를 추적하는 것은 상당히 어려운 일이다. 따라서 수동적인 대응 기법에서 나아가 적극적인 대응이 필요한데 이를 위해 공격자 및 공격자 시스템에 대한 실시간 추적과 즉각적인 공격 대응이 필요하다.

본 논문에서는 적극적인 공격 대응에 해당되는 공격자에 대한 실시간 추적이 가능한 침입자 역추적 메커니즘을 제안한다. 2장에서는 역추적의 정의 및 그 문제의 어려움에 대해서 살펴보고, 3장에서는 근원지 추적을 위한 기존의 연구를 살펴보며 4장에서는 경유지 우회 공격에 대한 유형을 분류하고, 시스템 운영체제(예, UNIX)의 구조를 이용하여 해킹의 근원지를 실시간으로 추적할 수 있는 새로운 추적 알고리즘을 제안한다. 5장에서는 제안된 추적알고리즘에 대한 구현 및 실험결과가 제공되며, 마지막으로 6장에서는 결론을 내린다.

2. 역추적 정의 및 문제점

2.1 용어 정의

[정의 1] 역추적 : 사이버 범죄를 시도하는 공격자의 네트워크 상의 실제 위치를 탐색하는 기술

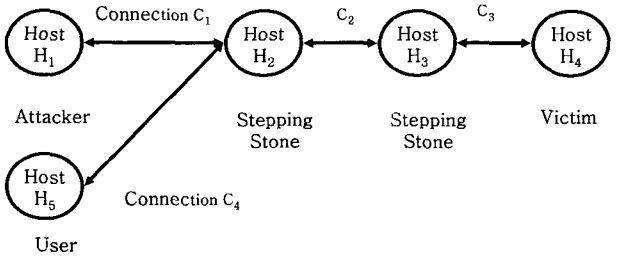
† 정 회 원 : ETRI 부설 국가보안기술연구소 선임연구원
†† 정 회 원 : 고려대학교 전산학과 교수
논문접수 : 2005년 8월 29일, 심사완료 : 2005년 12월 27일

[정의 2] Connection pair 과 stepping stone의 정의 : 어떤 사용자가 한 시스템에 접속하였다가 다른 시스템에 접속하였다면 접속한 순서를 *connection chain*[1]이라 한다. 이때 connection chain 상의 중간 호스트를 stepping stone이라 한다. 어떤 네트워크 connection pair가 connection chain의 일부이면 stepping stone connection pair라 한다.

가령 호스트 H_1, H_2, H_3이 양방향 connection을 맺었다고 가정하고 $H_1 \leftrightarrow H_2$의 connection을 C_1, $H_2 \leftrightarrow H_3$의 connection을 C_2라 표현한다면 호스트 H_2에서 C_1과 C_2는 stepping stone connection pair이다.

2.2 근원지 판단 문제 정의

에이전트 기반의 근원지 추적 문제를 정의하기 위해 (그림 1)을 살펴본다.



(그림 1) 공격자와 일반사용자의 Connection chain

호스트가 5대 있고 각각의 호스트를 H_1, H_2, H_3, H_4, H_5라 하고, 호스트 H_1의 Attacker가 Host H_4의 Victim 컴퓨터를 공격하기 위해서 먼저 Connection C_1을 통해 호스트 H_2에 접속한다. 그 후 다시 호스트 H_3에 Connection C_2를 통해 접속하고 다시 Connection C_3를 통해 목표 호스트인 H_4에 접속하여 공격을 수행하였다고 가정하자.

이 경우 Connection Chain은 [C_1, C_2, C_3]이고, Stepping Stone은 H_2와 H_3가 된다. 이때 *Stepping Stone Connection* Pair는 [C_1, C_2]와 [C_2, C_3]이다.

만약 호스트 H_3에서 호스트 H_4로 공격한 connection C_3를 탐지하고 [C_1, C_2, C_3]와 같은 connection chain이 존재한다면, stepping stone인 호스트 H_3에서 근원지 추적은 C_3에 매칭되는 inbound connection C_2를 찾는 문제로 정의할 수 있다.

마찬가지로 호스트 H_2에서 호스트 H_3로 공격한 connection C_2를 알 수 있다면 stepping stone인 H_2에서 근원지 추적은 C_2에 매칭되는 inbound connection C_1을 찾는 문제이다.

따라서 공격이 이루어진 connection C_n을 안다고 가정하면 호스트 H_n에서 근원지 추적은 C_n의 connection pair인 C_{n-1}을 찾는 것으로 정의 할 수 있다.

만약 호스트 H_n에서 C_n에 매칭하는 connection pair인 C_{n-1}이 존재한다면 이 호스트는 stepping stone이고 그렇지 않다면 이 호스트는 근원지로 판단할 수 있다.

2.3 Connection Pair 매칭

(그림 1)은 정상적인 User와 Attacker가 동시에 Host H_2에 접속한 모습을 표현한 그림이다. H_2에서는 Connection C_2에 해당하는 connection pair를 찾기 위해 inbound connection C_1과 C_4 중에서 적합한 connection을 판단해야한다.

본 논문에서는 H_2의 시스템 프로세스의 성질을 이용하여 inbound connection인 C_1과 C_4 중에서 Attacker와의 연결인 C_1을 찾아내는 추적 알고리즘을 제안한다.

3. 기존의 연구

지금까지 근원지 추적에 관한 연구를 살펴보면 다음과 같이 크게 세 가지로 나눌 수 있다.

- 호스트 기반 추적(Host-based Traceback)
- 네트워크 기반 추적(Network-based Traceback)
- 동적 네트워크 기반 추적(Active Network based Trace back)

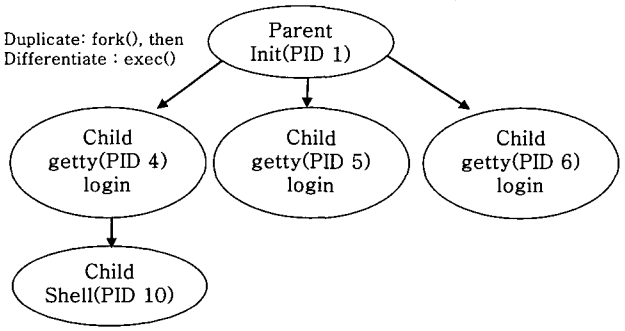
호스트 기반의 추적 메커니즘은 에이전트 기반 시스템으로 모든 호스트에 추적 모듈을 설치하거나 경유지로 사용된 호스트에 역추적 모듈을 설치하여 침입자를 추적한다. 이때 에이전트에서의 추적 목표는 해당 에이전트가 설치된 시스템에서 공격이 시작된 것인지 다른 시스템에서의 접속을 통해 중간 경유지로 사용되었는지 판단하는 것이다. 호스트 기반의 추적 메커니즘은 호스트마다 에이전트를 실행시켜 이 에이전트의 추적 모듈을 통해 경유지/근원지 파악을 하게 된다. 따라서 호스트 단위까지 추적이 가능하며 경유지/근원지 파악도 용이하다. 본 논문에서는 기존의 호스트기반 추적 연구[1, 2, 3]에서 다루지 않은 추적 에이전트에서 사용될 추적알고리즘을 제안한다.

네트워크 기반의 추적 메커니즘[4, 5, 6]은 네트워크를 구성하는 장치인 라우터나 방화벽에 추적 및 침입 탐지 모듈을 설치하여 네트워크 단위로 추적이 가능하도록 하는 방법이다. 네트워크 기반의 추적 메커니즘의 경우 라우터가 모든 패킷을 검사하는 방법을 취하므로 네트워크 단위의 추적이 용이하지만 라우터의 성능을 떨어뜨리는 요인이 된다. 또한 flooding 공격시 공격 경로를 검증하기 위해 라우터와 밀접한 관계를 가지고 라우터에 새로운 컴포넌트를 추가하는 방식을 취하고 있다. 즉 라우터에 추적 모듈을 설치해야 하는데 현재까지 flooding 공격을 막기 위한 연구 기법은 다수 있으나 상용 추적 모듈은 아직 개발되지 않았고 기존에 설치된 모든 라우터 제품을 교체해야 하는 현실적인 문제점도 있다.

Active Network 기반의 추적 메커니즘[7, 8]은 active 네트워크의 기능을 이용하여 추적 및 대응 모듈을 라우터 단위에서 수행할 수 있도록 하는 방법이다. Active Network 기반의 추적 메커니즘은 Active Network 상에서 동작하므로 현재의 인터넷 망에 적용할 수 없다. 또한 Active Network는 아직 연구 초기 단계이므로 망 구축에 장기간이 소요될 것으로 판단된다.

4. 제안하는 추적 알고리즘

본 장에서는 UNIX 시스템 프로세스 구조의 특성을 이용하여 해당 시스템이 stepping stone인지 아니면 근원지 시스템인지를 판별하고 stepping stone일 경우 근원지를 추적하기 위하여 inbound connection system의 IP 주소와 포트 번호를 찾아내는 알고리즘을 제안한다. 본 논문에서는 기존의 연구[9]를 이용하여 역추적 에이전트를 설계 및 구현하여 제안하는 추적 알고리즘에 대한 실험을 보인다.



(그림 2) UNIX시스템 프로세스 구조

4.1 UNIX 시스템 운영체제의 프로세스 구조

UNIX 시스템[10, 11]의 모든 프로세스는 아래의 요소를 가지고 있다.

- data
- code
- stack
- unique id number(PID)

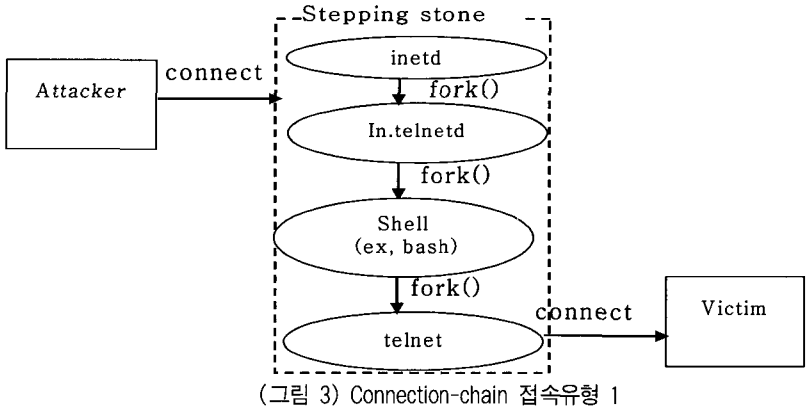
UNIX 시스템이 처음 시작되면 "init"라는 한 개의 프로세스만이 생성되며, 이후의 모든 프로세스는 init 프로세스의 복사에 의해서 생성 된다. 즉, init 프로세스는 모든 프로세스의 ancestor 가 된다. 프로세스가 복사될 때 parent 프로세스와 child 프로세스는 PID를 제외하고는 동일하며 같은 code를 실행한다. 하지만 다른 실행코드를 사용하여 실행하는 code를 변경할 수 있다. 이와 같은 방식으로 UNIX의 프로세스는 생성되고 실행된다.

4.2 connection chain이 발생시 프로세스 구조

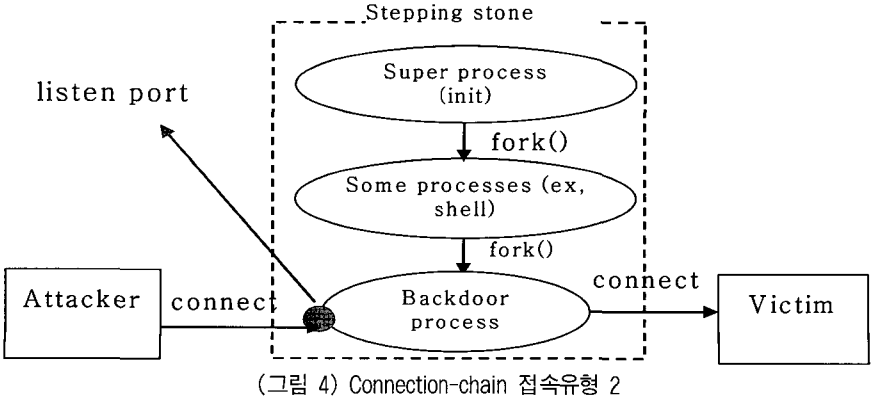
경유지가 UNIX 시스템인 환경에서 Connection pair (chain)가 발생되는 유형은 아래 4가지로 형태로 분류할 수 있다.

4.2.1 Telnet, SSH, rlogin 등 접속(접속 유형 1)

가장 일반적인 접속 유형으로서 경유지 시스템 네트워크 서비스 데몬인 inetd 데몬으로 telnet 접속하면in.telnetd 데몬을 통해 shell이 뜨게 된다[12]. telnet을 통해 shell로 들어온 후, 작업을 하다가 다시 telnet 명령어를 이용하여 피해



(그림 3) Connection-chain 접속유형 1



(그림 4) Connection-chain 접속유형 2

시스템으로 접속하여 피해 시스템을 공격하는 유형이다.

4.2.2 직접 접속형 백도어(접속 유형 2)

설치된 백도어가 port를 열고 listen하고 있으며 근원지에서 원격 접속 시 프로세스 생성 없이 직접적으로 피해 시스템으로 접속하는 유형으로서 원격 DOS(Deny of Service) 등의 일부 백도어 형태의 공격이 이에 해당된다.

4.2.3 프로세스 fork형 백도어(접속 유형 3)

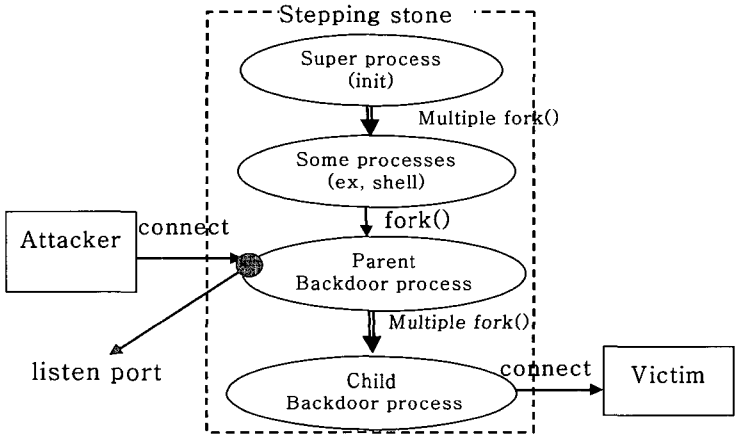
위의 접속 유형 2와 유사하지만 Attacker에서 stepping stone의 백도어에 접속을 한 다음 바로 피해 시스템에 접속하지 않고, child 프로세스를fork 한 후에 피해시스템에 접속하는 유형이다. 여기서 child 프로세스를 fork는 한번만 일어날 수도 있고 여러 번에 거쳐서 일어날 수 있다. port redirect 백도어나 기타 대부분의 백도어가 이와 같은 유형을 가진다.

4.2.4 프로세스간 통신형 백도어(접속 유형 4)

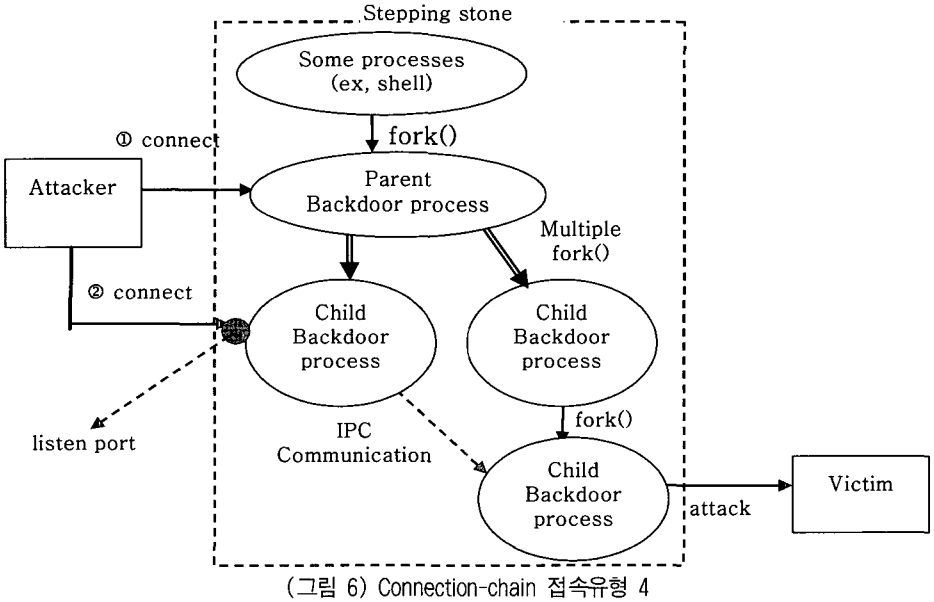
위의 접속 유형 3과 유사하지만 Attacker에서 stepping stone에 접속된 프로세스로부터 여러 개의 자식 프로세스가 fork된 형태이다. 공격자는 그중 한 개의 자식 프로세스에 접속한 후, 또 다른 자식 프로세스에 프로세스간 통신(IPC, Inter-Process Communication)을 이용하여 Victim을 공격하는 유형이다. 실제 이런 유형의 백도어가 등장한 적은 없었지만, 실제 컴퓨팅 환경에서 시스템의 프로세스간 통신을 이용하는 접속 유형 4의 백도어는 충분히 발생될 수 있다.

4.3 stepping stone 찾는 알고리즘

본 논문에서 제안하는 알고리즘의 기본 아이디어는 두 가지 이다. 첫째, 모든 프로세스의 조상은 init process(PID 0)이다. 둘째, 시스템이 경유지로 판명될 경우에는 inbound connection과 outbound connection이 존재한다. 4.2절에서 살펴본 바와 같이 경유지 시스템에서 inbound connection과 outbound connection은 밀접한 연관성을 가지고 있다.



(그림 5) Connection-chain 접속유형 3



(그림 6) Connection-chain 접속유형 4

4.3.1 자료구조

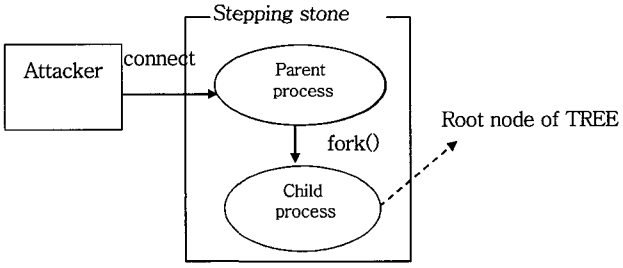
여기서는 connection pair를 찾는 알고리즘에 필요한 자료구조를 소개한다. 먼저 프로세스의 정보를 표현하기 위한 자료구조가 필요하다.

```
STRUCT PROCESS_INFORMATION
{
BOOL            process_existence;
TIME            creation_time;
TIME            termination_time;
BOOL            communication_existence;
HOST_INFO       inbound_connection;
}
```

제안하는 알고리즘에서는 PROCESS_INFORMATION 구조체를 하나의 노드로 갖는 TREE 자료구조가 필요하다. 이 TREE 자료구조는 inbound connection에 의해 생성된 프로세스들의 집합이며, 이 TREE의 생성, 확장, 그리고 소멸은 다음과 같다.

〈표 1〉 PROCESS_INFORMATION 구조체의 의미

필 드	의 미
process_existence	프로세스의 존재 유/무 표시
creation_time	프로세스 생성 시간 표시
termination_time	프로세스의 종료시간 표시 프로세스가 종료되지 않았을 경우, NULL로 표시
communication_existence	프로세스의 통신채널 유/무 표시
inbound_connection	프로세스와 inbound connection을 맺고 있는 시스템의 정보(IP주소, IP 주소) 표시 inbound connection이 없을 경우, NULL로 표시



(그림 7) TREE 생성

-1단계 : TREE 생성

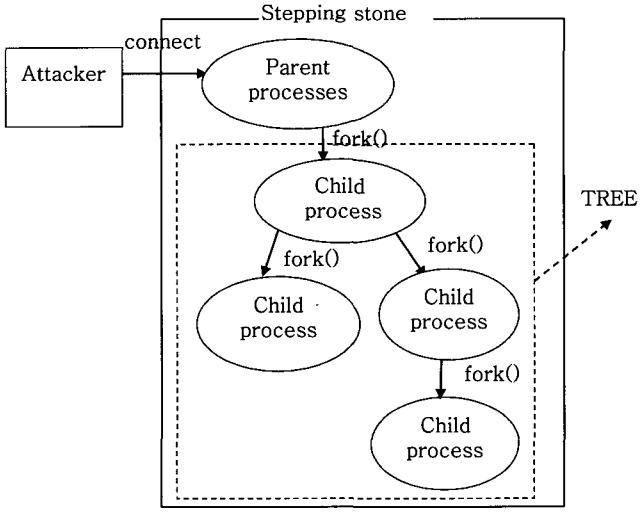
inbound connection에 의해 연결된 프로세스가 자식 프로세스를 fork할 경우, fork된 프로세스를 root 노드로 갖는 TREE 자료구조가 생성된다.

-2단계 : TREE 확장

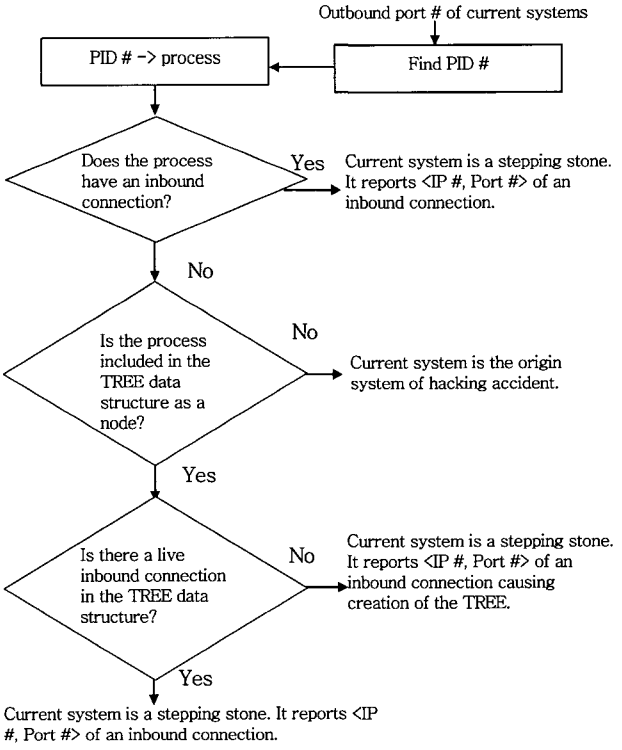
1단계에서 fork된 프로세스가 다른 자식프로세스들을 fork 할 경우, 이 TREE는 확장하게 된다. (그림 8)의 TREE에서 각 노드는 PROCESS_INFORMATION의 자료구조를 가진다.

-3단계 : TREE 소멸

TREE의 leave 노드에 해당되는 프로세스가 종료될 경우, 그 leave 노드는 소멸된다. 이와 반대로, non-leave 노드에 해당되는 프로세스가 종료될 경우, 알고리즘은 TREE에서 해당 노드를 제거하지 않고 구조체의 process-existence 필드 값을 FALSE로 설정한다. 따라서, 모든 자식 프로세스가 종료된 이후에 TREE의 root 노드가 소멸된다.



(그림 8) TREE 확장



(그림 9) Connection pair matching algorithm

4.3.2 알고리즘 flow-diagram

본 절에서는 경유지 우회 공격 발생시 connection pair를 찾는 알고리즘의 진행 절차를 보여준다. 본 알고리즘은 두 가지 state를 가지고 있다.

-Normal state : inbound connection에 연결된 프로세스가 자식 프로세스를 fork 할 경우, 알고리즘은 4.3.1에서 소개된 TREE 자료형을 생성한다.

-Traceback state : 공격이 발생되면, 제안된 알고리즘은 공격자를 추적하기 위하여 아래의 추적절차를 수행한다.

공격시 추적의 입력값에 해당되는 경유지의 source IP 주소가 주어졌을 때 그 IP 주소와 연결되어 있는 프로세스 ID를 찾고 PID에 해당되는 프로세스를 찾아서 그 프로세스에 inbound connection이 있는지 확인한다. 만약 inbound connection이 존재할 경우 그 inbound connection의 IP와 IP 주소가 근원지의 정보가 된다. 만약 inbound connection이 존재하지 않을 경우 그 프로세스가 TREE 자료구조의 노드로 포함되어 있는지 확인한다. 해당 프로세스가 TREE에 포함되어 있지 않을 경우, 현재 시스템이 근원지라고 판단한다. 만약 해당 프로세스가 TREE에 포함되어 있을 경우, 현재 TREE 자료구조에 대해 live inbound connection이 존재하는지 점검하여 live inbound connection이 있을 경우 해당 live inbound connection을 맺고 있는 시스템이 공격의 근원지 시스템이다. TREE에 live inbound connection이 존재하지 않을 경우 TREE의 생성을 유도한 inbound connection을 맺었던 시스템이 공격의 근원지 시스템이 된다.

4.4 제안된 방법의 장점

- 제안하는 역추적 알고리즘은 우회공격이 가능한 4가지 유형의 백도어에 대해서 근원지 추적이 가능함
- 네트워크 단위의 추적 메커니즘의 경우 라우터나 방화벽에 모듈을 설치해야 하는데 이에 비해 개발 및 설치 비용이 줄어듦
- 텔넷과 같은 대화형(interative) 모드의 connection 뿐만이 아니라 거의 대부분의 connection 을 찾아냄
- 기존의 connection pair matching 기법[4]에 비하여 경유지의 양쪽의 데이터가 다르거나 한쪽이 암호화 되어 있어도 잘 찾아냄
- TCP connection 뿐만 아니라 connectionless의 프로토콜도 추적 가능함
- 시스템 프로세스를 이용하므로 기존의 호스트기반[1, 2, 3]의 추적보다 효율적임
- 제안하는 알고리즘이 TREE의 데이터 기록을 로그로 남기면 사후 추적 등 포렌식 도구로 사용가능

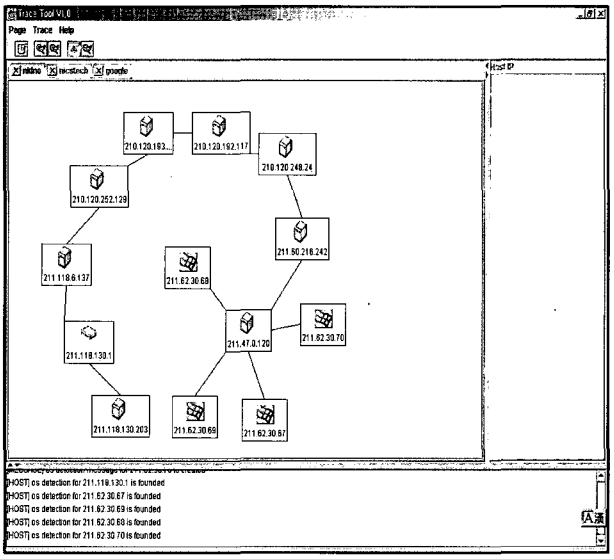
5. 구현 및 실험

본 장에서는 제안하는 알고리즘에 대한 구현과 실험을 보여준다.

5.1 구현

제안하는 추적 알고리즘의 실험을 위하여 (그림 10)과 같이 전체 추적시스템을 구현하였으며, 구성요소는 추적서버, 추적에이전트, 관리시스템으로 구성되어 있다. 추적시스템의 프로토콜 부분은 기존연구[13]을 이용하여 구현하였다.

제안하는 알고리즘이 적용되는 추적에이전트는 SUN Solaris와 Windows 플랫폼에서 구현되었으며, 구현을 위한 자세한 내용은 <표 2>와 같다.



(그림 10) 추적시스템

〈표 2〉 추적 에이전트의 구현

구분	Solaris agent	Windows agent
컴파일러	Gcc 3.0	VC++ 6.0
네트워크 IP 주소에 연결된 프로세스의 정보 획득 방법	Lsof tool	Fport tool
프로세스에 연결된 inbound connection에 대한 정보 획득 방법	Lsof tool	Fport tool
프로세스의 정보 획득 방법	/proc/PID/psinfo file	EPROCESS block

Lsof는 LiSt Open Files의 약자로서, 현재 System에서 돌아가는 모든 Process에 의해서 Open된 파일들에 대한 정보를 보여준다. 즉, 프로세스에서 사용되고 있는 네트워크 IP 주소에 대한 정보를 제공한다. FIP 주소는 Windows 플랫폼에서 Lsof와 동일한 기능을 제공한다. PID, PPID, 파일 이름, 그리고 프로세스 상태 등의 프로세스 관련 자세한 정보를 얻기 위해서 UNIX에서는 /proc/FID/psinfo 파일을 이용하고 Windows에서는 EPROCESS 블록을 이용한다.

5.2 실험 결과

본 절에서는 제안하는 알고리즘이 경유지 우회 공격 발생시

connection pair를 정확하게 찾는 것을 보이기 위하여, 4.2절에서 보였던 4가지 경유지 우회 공격 유형에 대하여 추적 실험과 그에 대한 결과를 보여준다. <표 3>은 4가지 경유지 우회 공격 유형에 대하여 제안하는 알고리즘이 정확히 outbound connection에 대한 inbound connection을 찾음을 보여준다. 즉, 제안하는 알고리즘은 4가지 경유지 우회 공격 유형에 대하여 정확히 공격 근원지를 추적함을 보여준다.

〈표 3〉 실험 결과

경유지 우회 공격 유형	실험에 사용된 도구	connection 존재 유/무	inbound connection에 의해 생성된 프로세스의 개수	실험 결과
접속 유형 1	Telnet	유	2	성공
	SSH	유	3	성공
	Rlogin	유	2	성공
접속 유형 2	DDos agent	무	0	성공
	Bo2k IP 주소 redirector	유	0	성공
접속 유형 3	IP 주소 redirector backdoor	유	1	성공
	Cd00r.c(packet coded backdoor)	유	1	성공
	Httptunnel	유	2	성공
	NetCat	유	1	성공
접속 유형 4	Test backdoor (designed by author)	유	4	성공

6. 결 론

컴퓨터와 인터넷의 보급으로 인하여 우리 생활에 있어서 많은 혜택이 있음과 동시에 컴퓨터 해킹 및 바이러스와 같은 정보화에 대한 역기능이 발생하였다. 침입 발생 시 그것을 탐지하는 IDS는 지금까지 매우 유용하게 사용되어 왔지만 경유지를 이용하는 징검다리 공격에 대해서는 경유지의 정보만을 제공할 뿐 실제 근원지인 공격자의 정보를 얻어올 수 없어서 시스템 해킹에 대한 대응책으로서 그 한계를 나타내고 있다. 이런 이유로 말미암아 해킹에 대한 능동적인 대응 방안의 일환으로 침입자의 근원지까지 추적하는 새로운 역추적 메커니즘을 개발하여 제안하였다. 4장에서 보인 새로운 역추적 메커니즘은 기존의 연구에 비교하여 4.4절과 같은 장점을 가지고 있다. 본 논문의 연구는 전산시스템에서 경유지 우회 공격에 대한 유형 분석, 에이전트 기반의 connection pair matching 알고리즘을 설계 및 구현하였고, 추적실험을 수행하여 제안하는 알고리즘이 다양한 경유지 우회공격에 대해서 정확히 근원지 추적을 할 수 있음을 보였다. 향후 이와 같은 에이전트 기반의 추적 알고리즘이 안정적으로 구현 및 적용되도록 지속적인 연구가 필요하다.

참 고 문 헌

[1] S. Staniford-Chen and L.T. Heberlein. "Holding Intruders Accountable on the Internet," In Proceedings of the 1995 IEEE Symposium on Security and Privacy, 1995.

[2] Steven R. Snapp, James Brentano, Gihan V. Dias, "DIDS (Distributed Intrusion Detection System) Motivation, Architecture, and An Early Prototype," Proceedings of the 14th National Computer Security Conference, 1991.

[3] H.T. Jung et al. "Caller Identification System in the Internet Environment" Proceedings of the 4th Usenix Security Symposium, 1993.

[4] Y. Zhang and V. Paxson, "Detecting Stepping stones," Proceedings of 9th USENIX Security Symposium, Aug., 2000.

[5] K. Yoda and H. Etoh, "Finding a Connection Chain for Tracing Intruders," In F. Guppens, Y. Deswarte, D. Gollamann, and M. Waidner, editors, 6th European Symposisum on Research in Computer Security-ESORICS 2000 LNCS -1985, Toulouse, France, Oct., 2000.

[6] D. Schnackenberg, "Dynamic Cooperating Boundary Controllers (http://www.darpa.mil/ito/ summaries97/E295_0.html)", Boeing Defense and Space Group, 1998.

[7] X. Wang, D. Reeves, S. F. Wu, and J. Yuill, "Sleepy Watermark Tracing : An Active Network-Based Intrusion Response Framework", Proceedings of IFIP Conference on Security, Mar., 2001

[8] Chongwoo Woo, Suntae Hwang, Jinwoo Choi, Sangyoung Kim, Hyungwoo Kang, Jaewoo Park, Gunwoo Nam, "Multi-agent based Intruder tracing System in the Active Network Environment", Proceedings of the ICACT 2003, pp.719~723, 2003.1.

[9] H.W. Kang, S.J. Hong, D.H. Lee : "Matching Connection Pairs", PDCAT 2004, LNCS 3320, pp.642~649.

[10] Graham Glass, "UNIX for Programmers and Users : A Complete Guide", Prentice Hall, 1993.

[11] W. R. Stevens, "Unix Network Programming," Prentice Hall, 1998.

[12] W.R. Stevens. TCP/IP Illustrated, Vol.1, Addison Wesley, 1994.

[13] B. Carrier, C. Shields : A Recursive Session Token Protocol for Use in Computer Forensics and TCP Traceback, IEEE INFOCOM 2002.

강 형 우

e-mail : kanghw@etri.re.kr

1997년 고려대학교 전산학과(학사)

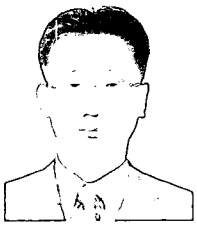
1999년 고려대학교 전산학과(석사)

2004년 고려대학교 정보보호대학원 박사과정 수료

1999년~2000년 한국전자통신연구원 연구원

2000년~현재 국가보안기술연구소 선임연구원

관심분야 : 네트워크보안, 시스템보안, 소프트웨어분석



홍 순 좌

e-mail : hongsj@etri.re.kr

1989년 숭실대학교 전산학과(공학사)

1991년 숭실대학교 대학원 전산학과(공학석사)

2005년 충남대학교 대학원 컴퓨터과학과(이학박사)

1991년~2000년 국방과학연구소 선임연구원

2000년~현재 국가보안기술연구소 팀장

관심분야 : 네트워크보안, 시스템보안, 소프웨어분석

이 동 훈

e-mail : donghlee@korea.ac.kr

1984년 고려대학교 경제학과

1987년 Oklahoma Univ. 전산학과(석사)

1992년 Oklahoma Univ. 전산학과(박사)

1993년~현재 고려대학교 전산학과 교수

2000년~현재 고려대학교 정보보호대학원 교수

관심분야 : 암호이론, 암호 프로토콜, 정보이론