상관성에 기반한 침입 정보 분석*

이경희, 이순구°, 김형식
충남대학교 컴퓨터과학과
{pupgirl, cyrus, hskim}@cs.cnu.ac.kr

A Correlation-based Analysis on the Intrusion Information

Kyung-Hee Lee, Soon-Koo Lee, Hyong-Shik Kim
Department of Computer Science, Chungnam University

요 약

기존의 침입탐지 시스템(IDS)은 침입 단계를 고려하지 않고 독립적이고 단편적인 공격 정보를 제공하기 때문에 관리자나 침입대응 시스템(intrusion response system)이 정보들을 이해하고 적당한 행동을 취하기가 매우 힘들다.

본 논문은 기존 침입 탐지시스템이 제공하는 정보들이 갖는 한계를 극복하기 위하여 모든 침입은 독립되어 존재하는 것이 아니라 서로 다른 공격의 연속으로 이루어 진다는 점에 근거하여 단편적인 공격 정보들의 상관성을 활용하기 위한 기법을 제안한다. 이미 알려진 공격 단계에 대한 상관성 정보를 그래프 형태로 표현하고 공격정보에 따라 전이하는 토큰을 이용하여 단편적인 공격들의 상관성을 분석함으로써 활용하기 용이한 분석 정보를 제공하는 것을 목표로 한다.

1. 서론

기존의 침입탐지 시스템이 제공하는 침입 정보는 매우 방대하기 때문에 관리자나 침입대응 시스템이 정보들을 이해하고 적당한 행동을 취하기가 매우 힘들다. 정보가 방대한 가장 큰 이유는 의심이 가는 모든 경우에 대하여 보고 정보를 생성하기 때문이다. 기존의 침입탐지 시스템은 침입의 가능성이 있다고 판단되는 모든 경우에 로그를 남기거나 경보를 발생시킨다는 점에서 로그나 경보에 있는 모든 내용이 침입과 바로 연결될 수 있는 것들이라 보기 어렵다. 특히 보안설정단계가 높은 침입탐지 시스템일수록 이런 정보는 더욱 많을 것으로 추정된다.

예를 들면 어떤 사람이 공격의 목적 없이 단순히 호스트가 살아있는지 ping을 했을 경우에도, 대부분의 침입탐지 시스템은 이것을 절대적인 침입준비 단계로 판단하고 정보를 남긴다. 이렇게 생성된 정보들은 관리자들이 이해하고 대응하기에 용이하지 않다.

그러나 만약 ping을 한 뒤에 얼마 지나지 않아 같은 호스트에서 버퍼 오버플로우(buffer overflow) 공격이 온다면 이전 공격인 ping은 침입을 위한 사전공격이었던 셈이 된다. 이처럼 침입은 단독으로 일어나는 것이 아니라 일반적인 침입단계에 따라 공격의 연속적인 다른 단계로 서로 관계를 가지고 있다는 점에서 방대한 내용의 침입 정보를 요약 분석하여 관리자가 이해할 수 있도록 하는 것은 매우 중요한 문제가 된다.

2. 상관성 기반의 침입분석

대부분의 침입탐지 시스템 기술이 무시하고 있는 중요한 점은 대부분의 침입은 독립되어 나타나지 않고, 공격의 연속적인 다른 단계로써 연관되어 나타난다는 것이다 [1].

2.1 전형적인 침입단계

네트워크 공격은 그 절차 및 기법이 이미 잘 알려져 있어 잠재적인 취약점을 방어하기 위한 수단이 많이 강구되어 왔다. 일반적인 공격절차는 가장 먼저 공격대상에 대한 "정보수집 단계"이며, 그 다음 수집한 정보를 바탕으로 "시스템 침입 단계"를 거치게 된다. 그리고 지속적인 침입 및 다른 시스템의 공격을 위한 "공격 전이 단계"를 거치게 된다.

(1) 정보수집

정보수집은 공격의 첫 번째 단계로 공격대상 네트워크에 대한 정보를 파악하는 것이다. 주로 네트워크 토폴러지, 시스템 OS, 네트워크 장치의 종류, 그리고 WWW, FTP 등 공격대상 네트워크가 제공하는 서비스와 그 버전에 대한 정보를 수집한다.

(2) 시스템 침입 단계

시스템 침입단계는 실제 개별 시스템에 침입하는 단계로 정보수집단계에서 수집한 정보를 바탕으로 가장 취약한 부분을 공격하게 된다. 일반적으로 버그가 있는 네트워크 서버를 공격하게 되는데 sadmind, amd, amountd, statd, POP, Imap 등 각종 서버의 원격 버퍼오버플로우 취약점을 공격한다.

(3) 공격전이 단계

"공격전이 단계"는 1차적인 시스템 침입 이후에 일어나는 침입을 말하는데, 1차적인 침입으로부터 얻은 정보 및 추가 작업을 통하여 시스템 침입을 확대하고 다른 시스템에 침입하는 단계이다[2].

2.2 Predicate (Prerequisite 와 Consequence)

연속적인 공격에서 공격들은 독립적으로 나타나지 않고, 공격의 다양한 단계로 서로 관련이 있기 때문에 predicate을 이용할 경우 쉽게 표현될 수 있다. 즉, prerequisite 와 consequence 를 이용하여 공격 정보들간의 연관성을 정형화 할 수 있다. 공격의 prerequisite 는 공격이 성공하기 위해 꼭 필요한 조건이며, 공격의 consequence 는 공격의 가능한 결과이다.

다음은 IPSweep, SadmindPing, SadmindBOF 등의 공격을 predicate 형태로 표현한 것들이다.

- **IPSweep**
  prerequisite={}, consequence={ExistHost}
- **SadmindPing**
  prerequisite={ExistHost},
  consequence={VulnerableSadmind}
- **SadmindBOF**
  prerequisite={ExistHost, VulnerableSadmind},
  consequence={compromised}

위의 예제를 보면, IPSweep 공격이 일어나기 위한 조건은 없으므로 prerequisite 는 null 이 된다. 그리고 IPSweep 공격이 성공한다면 호스트가 존재하는지 알 수 있으므로 consequence 에는 ExistHost 가 들어간다.

* 이 논문은 BK21 충남대학교 정보통신인력사업단의 지원을 받았음.

2.3 침입정보의 상관성

공격 정보들간의 상관성을 활용하기 위하여 선행 공격의 consequence와 후행 공격의 prerequisite가 일치하는 경우를 탐색한다

위의 예제를 보면, SadmindPing의 prerequisite인 ExistHost가 IPSweep의 consequence에 존재하기 때문에, IPSweep이 SadmindPing을 위한 사전 공격임을 알 수 있다. 즉, IPSweep→SadmindPing의 관계가 성립한다는 것을 알 수가 있다. 여기서 →는 IPSweep이 일어난 뒤 SadmindPing이 일어난다는 것을 나타낸다. 각 공격들 사이에 상관성을 적용하면 다음과 같다.

IPSweep→SadmindPing
IPSweep→SadmindBOF
SadmindPing→SadmindBOF
SadmindBOF→Rsh

이들을 통합하여 얻을 수 있는 정보는 다음과 같다.

IPSweep→SadmindPing→SadmindBOF→Rsh
IPSweep→SadmindBOF→Rsh

3. 침입 정보 분석

3.1 그래프 표현 및 활용

상관성 분석 결과를 이용하기 위하여 그래프가 이용된다. 그래프는 그림 1과 같이 나타낼 수 있다. 그래프의 상태는 각각의 공격을 나타내고, 화살표는 각 공격 사이의 관계를 나타낸다. 화살표 앞 공격이 화살표 뒤 공격을 위해 미리 준비하는 단계가 된다.

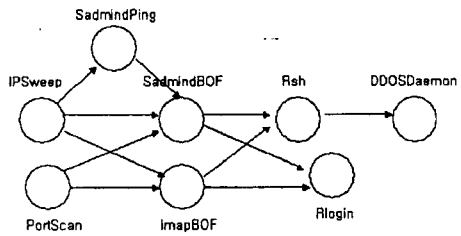

그림 1: 상관성 분석그래프의 예

일반적인 침입탐지 시스템이 생성한 단편적인 침입 정보들의 상관성 분석을 위해 필요한 정보들을 추출하기 위해서는 로그나 경보를 그래프에 대응시켜야 한다. 상관성 분석그래프를 이용하면 단편적인 침입정보들이 어떻게 서로 연관되어 있는지를 찾을 수 있기 때문에 공격자의 의도를 구체적으로 분석할 수 있다. 이때 로그나 경보를 그래프에 대응시키기 위해서는 토큰이 필수적이다.

3.2 토큰 관리

토큰은 경보에 의해 상태를 전이하면서 경보에서 받은 실제 정보를 저장한다. 경보가 새로운 공격이라면 토큰을 생성하고 경보가 이미 그래프상에 존재하는 토큰의 연속적인 공격이라고 판단되면 상태를 전이하게 된다.

그림 2는 그래프에서의 토큰들을 나타낸다. 각 토큰들은 서로 다른 연속적인 공격들을 나타낸다.

IPSweep 192.168.0.1→192.168.10.1·····①
SadmindPing 192.168.0.2→192.168.10.1·····②
SadmindPing 192.168.0.1→192.168.10.1·····③

침입탐지 시스템이 위와 같은 침입 정보를 생성하였다면, 토큰은 다음과 같이 움직인다. ① 정보를 받았을 때, 존재하는 토큰이 없으므로 새로운 토큰 A 을 생성하여 출발주소 192.168.0.1, 도착주소 192.168.10.1 그리고 시간 등의 정보를 저장한다. 그리고 ② 정보가 도착했을 때, 그래프에 있는 토큰 A 와 비교하면 전이는 가능하나 출발주소가 틀리므로 이것은 연속적인 공격이 아니다. 그래서 새로운 토큰 B 를 만든다. 마지막으로 ③정보가 왔을 때, 그래프에 있는 토큰 A 와 토큰 B 에 대한 비교를 통하여, 토큰 A 와 출발주소, 도착주소가 같고, IPSweep 에서 SadmindPing 으로 갈수 있기 때문에 토큰 A 는 새로 받은 정보의 시간을 새로 저장한 뒤, 다음 상태로 전이한다.

토큰에 관련된 연산은 다음 4 가지가 있다.

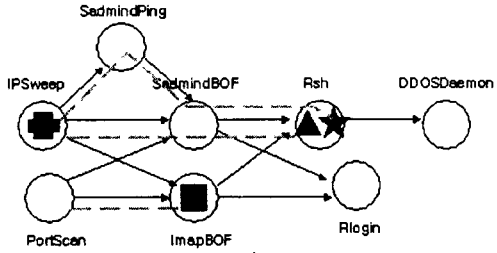

그림 2: 그래프와 토큰

(1) 토큰 생성

각 토큰들은 서로 다른 연속적인 공격들을 나타내기 때문에 같은 호스트에서 다른 연속적인 공격이 일어나면 서로 다른 토큰이 생성되고, 또한 같은 연속적인 공격이라도 다른 호스트에서 일어나면 서로 다른 토큰이 생성된다.

(2) 토큰 복사

서로 다른 일반적인 공격을 특정 침입탐지 시스템에 적용시켰을 경우에 같은 이름으로 탐지하는 경우가 생긴다. 이런 경우에 특정 침입탐지 시스템의 현 경보가 어떤 공격을 나타내는지 알 수 없으므로 가능성 있는 모든 공격으로 전이해야 한다. 그러나 토큰의 나머지 정보는 같으므로 전이 가능한 상태만큼 새로운 토큰을 복사하여 생성한다.

(3) 토큰 소멸

새로운 로그나 경보에 의하여 매번 토큰을 생성할 경우 불필요하게 많은 토큰을 유지하게 된다. 즉 단편적인 공격으로 판단된 경우에 대하여 토큰을 소멸시키기 위한 연산이 필요하다. 이것은 연속적인 공격행위들을 일정한 시간 간격 내에 발생한다는 점을 이용하여 해결될 수 있다. 따라서 일정시간 이상 전이되지 않은 토큰들에 대해서는 소멸연산을 통하여 적정규모로 토큰을 유지할 수 있도록 한다.

(4) 토큰 병합

경우에 따라서는 복사된 토큰들이 서로 다른 경로를 거쳐 같은 노드에 도달할 때가 있기 때문에, 복사된 토큰 중 일부는 공격이 진행됨에 따라 다른 토큰에 병합될 수 있다. 즉, 이 경우에는 출발주소와 목적주소 등에 대한 비교를 통하여 일치할 때에만 병합 연산을 적용한다.

4. Snort 에이전트를 이용한 구현

4.1 시스템구성

3장에서 제안된 모델을 설계하고 구현하기 위해서 snort 에이전트를 이용한 간단한 시스템 도구를 사용하였다. 이 시스템의 목적은 공격의 논리적인 단계(전략)를 그래프로 나타내고, 그래프를 이용하여 탐지한 침입을 효과적으로 분석하는 것이다.

시스템은 크게 Correlator 부분과 Analyzer 부분으로 나눌 수 있다. Correlator 의 역할은 공격들간의 논리적인 관계를 찾아내어 그래프 형태로 구성하는 것이다. 그리고 Analyzer 의 역할은 Correlator 에서 만든 그래프를 가지고, 실제 침입탐지 시스템의 로그나 경보와 비교하여 효과적인 분석을 이끌어 내는 것이다. Correlator 의 입력은 그래프를 생성하기 위한 Knowledge base 이고 출력은 Knowledge base 를 변형한 그래프이다. 그리고 Analyzer 의 입력은 실제 침입탐지 시스템의 로그나 경보이며, 출력은 종합적으로 분석한 결과이다. 그리고 Analyzer 의 입력은 실제 침입탐지 시스템의 로그나 경보, 출력은 종합적으로 분석된 결과이다.

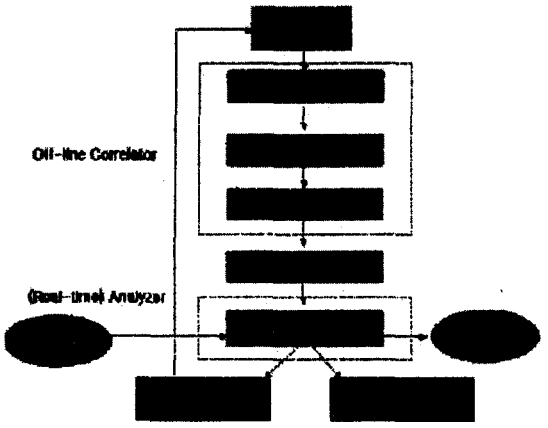

그림 3: 시스템 구조

4.2 Knowledge Base

Knowledge Base는 Correlator에 들어가는 입력으로 공격들의 prerequisite와 consequence, level과 failure_counter가 정의되어 있다. 여기서 level은 다양한 보안수준의 분석을 위한 것으로 침입 위험도에 따라 부여된다. 그리고 failure_counter_name은 특정 상태를 몇 번이나 반복했는지 기록하기 위한 것이다. 그리고 어떤 공격에서는 출력하기 위한 output_current와 output_future도 포함된다. 최근까지 몇 번까지의 반복적인 시도가 발견되는데 이런 경우를 탐지하기 위하여 활용될 수 있다.

DirectoryTraversal: prerequisite="NotBackdoor", consequence="Compromised", level=3, failure_counter_name ="cmd", output_current="DirectoryTraversal 공격을 %failure_counter 회 시도 중이다.", output_future ="Directory Traversal 공격에 대한 침입을 받았다."

위에 예는 DirectoryTraversal 공격에 대한 Knowledge Base 예이다. DirectoryTraversal 공격이 일어나기 위한 선행조건은 NotBackdoor이며, 성공할 때에는 시스템이 위태로워지게 된다. level은 침입의 단계를 나타내는데 3이라는 숫자를 통해 공격 전이 단계라는 것을 표현한다. output_current는 현재 공격이 진행되는 상황을 출력해주기 위한 형식이고, output_future는 앞으로 예측 되는 상황을 출력할 필요가 있을 때 이용된다.

4.3 공격 이름 변환

제안한 시스템에 여러 침입탐지 시스템의 로그나 경보를 적용시키기 위해서, Knowledge base 의 일반적인 공격 이름을 적용시키고자 하는 침입탐지 시스템의 공격 이름으로 바꾸어주어야 한다. 표 1은 일반적인 표현을 snort 에서의 표현으로 변경하기 위한 예이다.

일반적인 표현	Snort 에서의 표현
SadmindPing	RPC portmap request sadmind
Rsh	RSERVICES rsh root

표 1: 공격이름 변환

4.4 그래프와 토큰의 활용

제안한 시스템에서 그래프를 표현하기 위하여 그림 4 와 같은 배열을 사용하였다. 각 원소는 상태를 나타낸다. 상태 안에는 전이 가능한 다음 상태들의 정보와 여러 기술 정보들이 있다. 기본적으로 토큰은 그림 4 와 같이 트리를 변형시킨 이원적 링크리스트로 저장된다.

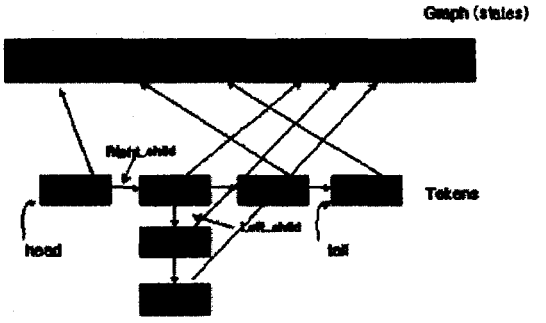

그림 4: 그래프와 토큰의 구조

(1) 토큰 생성

침입탐지 시스템으로부터 경보를 받으면 그래프 위에 이미 생성되어 움직이는 토큰과 비교한다. 경보가 이미 존재하는 토큰의 연속적인 공격이 아닐 경우에 새로운 토큰을 생성하여 기존 토큰의 오른쪽에 삽입하는데, 이것을 right_child 토큰이라고 부른다.

(2) 토큰 복사

서로 다른 일반적인 공격을 특정 침입탐지 시스템에 적용시켰을 때, 같은 이름으로 탐지하는 경우가 생긴다. 이 때, 하나의 경보에 의해 전이 가능한 상태만큼 새로운 left_child 토큰을 생성한다. 이 토큰(들)은 원래의 토큰의 정보와 같으므로 원래의 토큰을 복사한다.

(3) 토큰 소멸

새로 생성한 토큰과 최근에 전이한(움직인) 토큰은 헤드 쪽으로 옮긴다. 그러면 오랫동안 움직이지 않은 토큰들은 저절로 테일 쪽으로 옮겨지게 된다. 경보가 들어올 때마다 경보의 시각과 테일 쪽에 있는 토큰의 마지막 접근시간을 비교하여 일정시간 이상 차이가 나는 토큰들은 제거된다.

(4) 토큰 병합

복사본이 있는 토큰이 전이할 경우에 left_child의 토큰들을 검사하여 조건을 만족할 경우 병합연산을 적용하여 일부 left_child 토큰을 제거한다.

5. 결론

기존의 침입탐지 시스템의 문제점은 단편적인 공격이나 독립적인 공격에 초점을 두고 있을 뿐, 이런 공격들 뒤의 논리적인 단계나 전략은 찾아낼 수 없다는 것이다. 그리고 세부적인 내용의 로그를 생성하지만, 누구나 그러한 로그를 보고 상황을 파악하기에는 로그 내용이 방대할 뿐만 아니라 구체성이 결여되어 있다.

본 논문에서는 관리자와 침입대응 시스템이 경보들을 이해하고 적당한 행동을 취하기가 용이하도록 기존 침입탐지 시스템이 제공하는 정보들을 분석하기 위한 모델을 제안하였다. 향후 다양한 종류의 침입탐지 시스템들이 서로 독립적으로 동작하는 경우에도 적용될 수 있도록 확장할 계획이다.

참고문헌

[1] P. Ning, D.Reeves, and Y. Cui. Correlating Alerts Using Prerequisite of Intrusions. Technical Report TR-2001-13, North Carolina state University, Department of Computer Science, Dec. 2001.

[2] http://cert.certcc.or.kr/paper/paper20000404.html.