제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

정책(Policy)을 이용한 리눅스 기반의 실시간 침입 탐지 및 대응 기법

김미영, 문영성
숭실대학교 컴퓨터학부
e-mail : mizero31@sunnuy.soongsil.ac.kr, mun@computing.ssu.ac.kr

A Study on Policy-based Real-Time Intrusion Detection and Response Method in Linux

Miyoung Kim, Youngsong Mun
Dept. of Computing , Soongsil University

요 약

정보 인프라의 고도화와 인터넷 사용의 폭발적인 증가로 인해 다양한 형태의 정보를 대량으로 교환할 수 있는 환경이 마련되었으며, 정보기술의 보편화를 통해 누구든지 쉽게 기술을 습득하고 이용하게 되었다. 인터넷 사용자는 크게 일반 사용자 및 적대적 사용자로 분류될 수 있으며, 특히 적대적 사용자는 정보의 불법적인 유출, 악용, 파괴할 수 있는 고도의 기술을 지닌 그룹으로서 인터넷의 존재 자체를 위협할 수 있는 수준이며, 이들의 기술은 날로 지능화되고 자동화되는 추세이다. 정보의 가치가 중요해 지면서 고급 정보에 대한 피해 사례가 늘어가고 있으나, 이를 정확하게 발견하고 신속하게 대처하기 위한 기술의 개발은 아직 초보 단계에 머무르고 있다. 본 논문에서는 실시간 탐지 및 효과적인 대응을 지원하기 위한 '정책(policy)' 기반의 처리 기법에 관해 설명한다.

1. 서론

정보 산업 기반의 강화와 정보 접근의 대중성은 다변하는 정보산업에 빠르게 적용할 수 있는 장점을 제공하지만, 그로 인해 파생되는 부작용도 매우 심각하다. 누구나 정보를 엑세스할 수 있고, 누구나 정보를 악용할 수 있게 되었으며, 그러한 기술은 이제 특정 악의적인 해커 그룹뿐 아니라 일반 사용자에게로 급속히 확산되어 일반화되고 있는 실정이다. 그러므로 정보 접근 및 사용에 대한 보안 강화는 필수적이다. 보안이 제공되지 않는 정보는 이미 정보로서의 가치를 잃게 된다.

과거 정보 보호를 위한 보안 접근 방식은 크게 알려진 공격에 대한 커널 및 시스템 전반에 걸친 패치(patch)제공이나, 드라이버 수준의 패킷 수집을 통한 분석(IDS: Interusion Detection System)에 그쳤으나, 이와는 상이한 개념으로 알려진 공격 방법 이외에 알려지지 않은 공격에 대한 학습 및 대응 정책 수립을 목적으로 하는 '하니팟(Honeypot)' 개념의 시스템 보안 기술이 등장하고 상용화를 위해 연구가 진행 중에 있다.

본 논문에서는 하니팟의 주요 기능 중의 하나인 탐지 및 대응 기능에 관한 구현 접근 방법 및 효과적인 사용 방안에 대해 기술한다. 공격 탐지 및 대응은 여러 단계별로 분류된 트리거(Trigger)및 정책(Polciy)에 의해 관리 및 운용되고, 이를 통해 얻은 결과를 바탕으로 적극적(Active)/소극적(Passive) 대응을 행할 수 있다. 또한 알려진 공격에 대해서는 미리 정해진 정책에 따라 처리할 수 있으므로, 대응이 효과적이며 대응상의 오류 및 처리시간을 줄일 수 있다.

2. 하니팟 시스템 기능 블록

본 논문에서 제안하는 하니팟은 크게 5 개의 기능 엔티티로 구성된다. 서비스 관리자, 응답 관리자, 룰 관리자 등을 통해 외부의 침입 방법에 대해 배울 수 있으며, 패킷 분석 엔티티와 대응 엔티티를 통해 주어진 정책에 따라 대응을 할 수 있다. 대응 및 분석은 룰을 기반으로 행해 진다. 위의 구성 요소 중 본 논문에서는 룰 관리자에 대한 구현에 관해 논의한다.

2.1 룰 관리자

서비스 응답 엔티티의 서브 모듈인 실시간 패킷 모니터링 기능에 의해 수집된 패킷이 패킷 분석기로 보내지고, 패킷 분석기는 수집된 패킷의 관심 필드 및 공격 패턴을 검출해서 적절한 대응 동작을 취하게 되는데 이때 관리자를 통해 패킷 분석기가 참조하는 정보를 룰 형태로 생성, 저장 및 삭제할 수 있다. 룰은 다음과 같은 구문 형식을 가진다.

```
Rule_name :: = [A..Z | 0..9]*
Rule_id = [1..9][0..9]*
Protocol_Type = ['tcp'|'udp'|'icmp'|'arp']
Response_Action = {'alert' | 'log' }
Port_Number :: = [1..9][0..9]*
Network ::= [IP addr | MACROS]
Filter ::= ['flags' | 'content' ]
DetectLevel = [Warning(0)|Minor(1)|Major(2)| Critical(3)]
Define {
        Category = ['DT' | 'AT']
        Source = { Network Port_Number };
        Dest   = {Network Port_Number };
        Action = Response_Action;
        Protocol = Protocol_Type;
        Condition = {Filter : value}*
        Level = Detect_Level;
} ::= Rule_name(Rule_Descriptor);
```

'root' 권한으로 Login 하는 경우에 대한 룰을 간단히 기술하면 다음과 같다.

```
Define {
        Category = DT
        Source = {0.0.0.0 any}
        Dest = {192.168.0.1 23};
        Action = alert;
        Protocol_Type = tcp;
        Condition = {content : 'login', content:'root'}
        Level = Critical;
} ::=RootLogin("Telnet Root Loggin Detection");
```

각 룰은 고유의 Rule_id 로 식별되며, 룰 생성시 기존의 룰과 중복되지 않도록 설정해야 한다.

룰을 효과적으로 구현하기 위해 본 논문에서는 '트리거(Trigger)' 및 정책('Policy)'의 개념을 도입하고 상호 연관성 및 동작 과정을 기술한다.

2.1.1 트리거(Trigger)

트리거는 생성된 룰을 기반으로 외부 또는 내부로부터의 공격 가능성에 대한 실시간 탐지를 담당하는 이벤트 기반의 객체이다. 즉, 트리거는 각각의 룰당 하나씩 존재하거나 여러 개의 룰을 묶어서 하나로 처리할 수 있다.

시스템 및 네트워크에 대한 침입은 크게 3 가지로 분류할 수 있다. 먼저, 외부 네트워크로부터의 침입(External Intrusion)은 외부의 침입자가 다른 네트워크를 통해 침입을 시도하는 경우에 해당되는 경우로서, 일반적인 포트 스캔 및 버퍼 오버플로우 방식 등을 사용한다. 내부 네트워크로부터의 침입(Internal Intrusion)은 공격 대상이 되는 네트워크와 동일한 네트워크를 통해 침입을 하는 경우이다. 외부 네트워크로부터의 침입(External Intrusion)과 동일한 침입 방법이 사용될 수 있으며, 또한 id 나 패스워드의 노출 등으로 인해 발생할 수 있다. 마지막으로 호스트 침입(Host Intrusion)은 정당한 사용자 혹은 악의적인 방법을 통해 호스트에 대한 정당한 사용 권한을 얻은 사용자가 호스트의 파일 및 주요 시스템 설정 정보 등을 엑세스 하는 경우를 말한다. 이 경우는 호스트에 접근한 모든 사용자는 탐지의 대상이 된다. 이러한 각 경우에 따른 침입에 대한 효과적인 탐지를 제공하기 위해 트리거는 다음과 같이 분류된다.

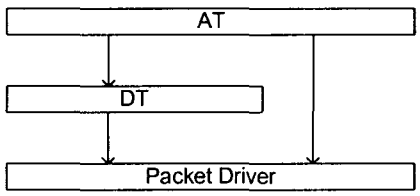
가. 탐지 트리거(DT : Detection Trigger)

탐지 트리거는 External/Internal Intrusion 을 통한 침입이 발생하는 경우 이를 실시간으로 탐지할 수 있는 기능을 제공하는 것으로서 2.1 에서 예로 든 "Telnet Root Loggin Detection" 룰이 이에 해당될 수 있으며, 그 이외에 많은 DT 를 정의할 수 있다.

나. 접근 트리거(AT : Access Trigger)

접근 트리거는 Host Intrusion 이 발생했을 경우에 실시간 탐지 기능을 제공하기 위한 것으로서 중요 시스템 파일 및 커널 구성요소, 서비스에 대한 잘못된 엑세스를 행하는 경우에 이를 대응하기 위한 탐지 기능을 제공한다. 중요 파일 및 커널/서비스 구성 요소 등은 미리 정의된 보호 정책에 따라 보호된다

DT/AT 트리거의 엑세스 관계를 표시하면 다음 그림과 같다.



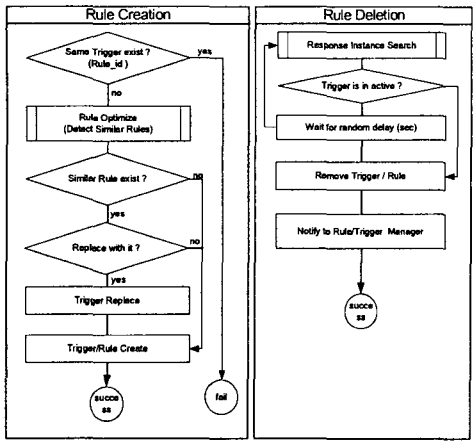
<그림 3> 트리거 엑세스 관계

다. 트리거 추가 및 삭제

트리거는 새로운 Rule 이 생성될 때 다음과 같은 과정을 통해 만들어 진다.

(1)룰 생성 > (2) Rule_id 를 통한 중복성 검사 >
(3)룰 정의를 통한 중복성 검사 > (4)트리거 생성

만일 새로 생성하려는 Rule_id 와 동일한 Rule_id 가 이미 존재하는 경우 룰 생성은 실패한다. 이 경우 구현상 시스템 관리자에게 동일한 Rule_id 가 존재하므로 다른 값으로 설정하도록 권고할 수 있다. 또한 Rule_id 를 룰 생성과 동시에 자동으로 만들 수 있도록 할 수도 있다.



<그림 4> 트리거 생성 및 삭제

트리거는 룰이 삭제되는 경우 함께 삭제된다. 삭제 절차는 다음과 같다.

(1) 룰 제거 > (2) 트리거가 동작중인지 검사 >
(3) 트리거 제거 > (4) 제거 통보

룰이 제거되면 그 룰로 인해 생성된 트리거도 함께 제거되어야 한다. 그러나, 만일 제거하려고 하는 룰의 트리거가 이미 동작 중인 경우, 즉 탐지 및 대응을 위한 Instance 의 트리거 집합 및 체인에 들어 있는 경우, 탐지 및 대응이 완료되고 제거하려고 하는 트리거가 속한 트리거 체인이 엑티브 상태가 아닐 때 까지 룰 삭제는 보류되어야 한다. 트리거가 재거되면 이 사실을 트리거 관리 모듈로 통보함으로써 트리거 집합에서 트리거를 삭제해야 한다.

라. 트리거 중복성 제거

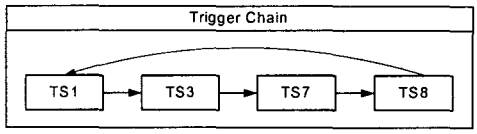
트리거는 룰을 기반으로 생성되므로, 트리거의 중복성을 제거하려면 룰을 기술할 때 주의를 기울여야 한다. 이를 구현하기 위해, 룰 기술의 각 단계별로 기존 트리거의 정의 요소들을 보여 줌으로써, 전체적인 중복을 피할 수 있다. 또한 트리거가 너무 많은 경우, 관리 및 대응이 복잡해 지며, 너무 적은 경우 탐지 기능을 제대로 수행할 수 없으므로 룰 정의(트리거 생성) 시에는 이를 충분히 고려해서 최적으로 유지하기 위해 반드시 중복성을 제거해야 한다.

마. 트리거 집합(Trigger Set)

트리거 집합은 유사한 트리거 및 알려진 침입에 대한 탐지를 위해 미리 구성한 관련 트리거들의 집합으로서 TS1 = { DT1, DT3, AT2, AT3, ... } 와 같이 정의될 수 있으며, 트리거 집합은 대응기능 모듈에게 정책(Policy)를 결정할 수 있는 정보를 제공한다.

바. 트리거 체인(Trigger Chain)

트리거 체인은 탐지 단계별로 발생할 수 있는 트리거 집합의 사건 발생 순서에 기반한 관계성을 표시한 것으로서 'TC1 = TS1> TS3 > TS7 > TS8'과 같이 표시되며 최종적인 정책(policy)을 수립하고 검색할 수 있도록 하는 정보를 제공한다.



<그림 5> 트리거 체인

2.1.2 정책(Policy)

정책(Policy)은 하니팟 시스템의 기본 기능 블록인 'Reactor' 에게 효과적인 대응을 위한 최적화된 정보를 제공한다. 정책은 미리 설정될 수 있으며, 새로운 상황에 맞게 동적으로 정의될 수도 있다. 정책은 다음과 같이 기술될 수 있다.

Policy_name ::= [a-zA-Z]*[0..9]*
Policy_id = [1..9][0..9]*
Priority = [0-64]
Trigger_chain ::= {e : TC list }
Correlation_option ::= ['yes' | 'no']
BP_context ::= BP Index
Policy_descr ::= ASCII set

Policy_name 은 정책의 이름을 정의한 것으로서 수행하고자 하는 정책과 연관성을 가지도록 이름을 부여해야 한다. Policy_id 는 설정된 정책을 식별하기 위한 값으로서 각각의 정책은 고유의 값을 가져야 한다. Priority 는 설정된 정책의 우선 순위를 나타내는 값으로서 0 인 경우 최상위 위선 순위를 가지며, 64 인 경우 최하위 우선 순위를 가진다. 정책의 우선 순위는 정책의 중요도를 통해 지정할 수 있다. Trigger_chain 은 정책 결정과 관련된 트리거 이벤트의 흐름을 정의한 것으로서, 트리거 체인을 통해 최종 정책이 선택되며, 이 정보는 'Reactor'에게 전달되어 능동적/수동적 대응의 기본 자료가 된다. Correlation_option 은 설정한 정책이 우선 순위에 따른 Correlation 대상이 되는지를 지정한다. BP_context 는 설정한 정책이 속하는 블록킹 프로파일을 나타낸다. 블록킹 프로파일은 Reactor 에서 대응을 위해 필요로 하는 정보를 말한다. Policy_descr 은 수립된 정책에 대한 간단한 설명을 기술한다. 최대 64 자로 이내로 제안한다.

가. 정책 생성 및 삭제

정책은 시스템 관리자의 필요에 의해 하나 이상의 트리거 체인을 정의함으로써 생성할 수 있다. 일단 정책 수립을 결정하는 단계에서 우선 데이터베이스를 통해 기존의 정책을 검색 및 참조한 후 어떤 블록킹 프로파일에 속하는 정책인지를 결정하고 선택한 BP 에 정책을 추가한다. 정책의 순서는 다음과 같다.

(1) 정책 결정 > (2) 트리거 체인 선택 >

(3) BP 검색 및 선택 > (4) BP 추가

먼저 정책 결정 단계에서는 기존의 정책을 수정하거나 필요한 정책이 존재하지 않는 경우 새롭게 정책을 설 결정한다. 다음, 정책을 구성하기 위해 하나 이상의 트리거 체인을 지정할 수 있는데, 만일 원하는 트리거 체인이 존재하지 않는 경우, TS 및 AT/DT 검색을 통해 트리거 체인을 먼저 구성할 수도 있다. BP 검색 단계에서는 생성된 정책이 포함될 BP 를 선택한다. 만일 BP 가 존재하지 않는다면, 새롭게 BP 를 생성할 수도 있다. BP 가 선택되면 새로 만든 정책을 BP 에 추가한다. 만일 Reactor 가 이미 선택된 BP 를 통해 능동적/수동적 대응을 진행 중인 경우, 새로 설정한 정책은 그 다음 번부터 효력을 가진다.

나. 정책 순서화(Relocation / Ordering)

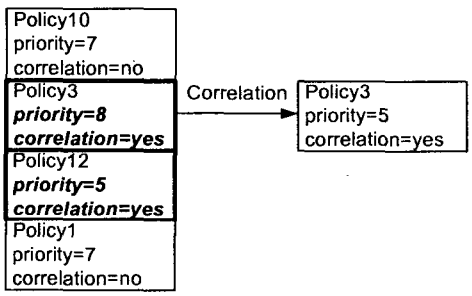
효과적인 정책의 관리 및 적용을 위해 정의된 정책은 우선순위를 가진다. 정책에 우선 순위를 부여함으로써 다음과 같은 이점을 얻을 수 있다.

(1) 정책관리의 효율성을 높일 수 있다. 우선 순위별로 정책을 참조할 수 있다.
(2) 정책 Correlation 을 가능하게 한다.

시스템에 따라 많은 정책이 수립되고 삭제 된다. 그러므로 이러한 정책을 효율적으로 관리하기 위해서는 지정된 Priority 별로 정책을 순서화하는 것이 중요하다. 그리고 다. 에서 서명할 정책 Correlation 기능은 트리거 체인에 의해 여러 개의 정책이 선택되었을 때, 선택된 정책들의 최적화를 제공하기 위해 대표 정책을 사용하고자 하는 경우 유용하다.

다. 정책 Correlation

정책 Correlation 은 트리거 체인을 통해 여러 개의 정책이 선택되었을 경우, 그 중에서 대표 정책을 선택하도록 하는 개념이다.



<그림 7> 정책 Correlation

먼저 Correlation 대상이 되려면 정책을 정의할 때 'correlation=yes'로 지정해야 한다. 선택된 정책 중에서 가장 높은 우선 순위를 가지는 정책을 대표 정책으로 결정한다.

3. 실시간 탐지 및 대응

패킷 드라이버를 통해 수신된 패킷을 실시간으로 수집하고 이를 DT(Detection Trigger) 및 AT(Access Trigger)로 전달한다. 이를 통해 각 트리거의 상태는 Active 상태가 되며, 이 정보는 트리거 체인으로 전달된다. 트리거 체인은 트리거의 발생 순서를 나타내는 것으로서 각각 기술된 룰의 발생 상관 관계를 나타내고 있다. 미리 정해진 순서대로 트리거가 발생되는 경우 이는 알려진 침입에 대한 Reactor 의 대응을 위한 정보로 제공된다.

정책 Correlation 을 통해 정책을 최적화할 수 있으며, 미리 정의된 침입의 경우 빠른 대응이 가능하다.

4. 결론 및 향후 연구과제

근래에 들어서 공격의 유형이 호스트를 목표로 하기보다, 네트워크를 목표로 진행되고 있으므로, 본 시스템을 확장해서 Firewall 이나 Router 등의 시스템과 함께 연동을 하도록 함으로써 정책(Policy)를 사용한 네트워크 차원의 HoneyPot 시스템을 구축하기 위한 연구에 적용해야 할 것이다.

5. 감사의 글

이 논문은 정보통신부 대학정보통신 연구센터 지원사업의 지원 및 한국소프트웨어진흥원의 관리로 수행되었음

참고문헌

[1] Snort Users Manual Snort Release: 1.8.1 Martin Roesch 10th August 2001
[2] http://project.honeynet.org/
[3] http://all.net/ "Deception Toolkit"
[4] 김병구,김동성,정태명,"계층적 구조를 갖는 침입탐지 통합 시스템 설계", 한국정보처리학회 추계학술대회 논문집, Vol. 6, No.2, Jan.,1999, pp. SEC137-SEC142
[5] 정훈조,김병구,정태명, "침입의 유형과 탐지 시스템의 분류", 한국정보처리학회 추계학술대회 논문집, Vol. 6, No.2, Jan.,1999, pp.SEC163-SEC168
[6] 김미영, 문영성, "리눅스 기반의 실시간 침입탐지를 사용한 보안 기술에 관한 연구", 한국정보처리학회 춘계 학술 발표논문집, vol 9, No1, April, 2002, pp.134-136
[7] Miyoung Kim, Youngsong Mun, "The Development of HoneyPot System", SAM'02, June, 2002, pp. 484-489