공학석사 학위논문

패킷 로그 분석을 통한 Snort 규칙 추천 기법 연구

Snort Rule Recommendation Technique
through Packet Log Analysis

지도교수 이 장 세

2011년 2월

한국해양대학교 대학원

컴퓨터공학과
박 근 우

KOREA MARITIME UNIVERSITY
1945
한국해양대학교

공학석사 학위논문

패킷 로그 분석을 통한 Snort 규칙 추천 기법 연구

Snort Rule Recommendation Technique through Packet Log Analysis

지도교수 이 장 세

2011년 2월

한국해양대학교 대학원

컴퓨터공학과

박 근 우

이 논문을 박근우의 공학석사 학위논문으로 인준함.

위원장 신 옥 근 인

위 원 김 재 훈 인

위 원 이 장 세 인

2010년 12월 24일

한 국 해 양 대 학 교 대 학 원

목 차

Snort Rule Recommendation Technique through Packet Log Analysis

Geun-woo Park

Department of Computer Engineering
Graduate School of
Korea Maritime University

Abstract

Recently, as using of network service systems providing various services, such as FTP increases, attacks of hacker on them having vulnerabilities are increasing. Thus we need various security tools for detecting and preventing attacks. A IDS(Intrusion Detection System) being one of security tools detects external attacks by using rules for intrusion detection. Because applying whole of rules at a IDS consumes many resources and decrease speed of services, users of IDS need knowledge of security experts to properly configure rules at a IDS. To overcome these problems, this thesis proposes the system recommending Snort rules through packet log analysis. For this, firstly, the proposed system structures Snort rules and analyzes packet log generated from network service systems by using structured rule data. Secondly, the proposed system recommends proper rules based on the result of analysis.

Finally, the proposed system applies recommended rules at Snort and removes them from Snort. The proposed system has an advantage of managing rules at Snort automatically and efficiently without knowledge of security experts.

제 1 장 서 론

국내의 인터넷 이용률은 꾸준히 증가 추세에 있으며 현재 전체 가구의 80%이상이 가구 내에 초고속 인터넷 환경이 구축되어 있고 총 인터넷 이용자 수는 약 3700만명에 육박하고 있다[1]. 이처럼 인터넷 사용자의 증가에 따라 국내 주요 포탈 웹(web) 사이트의 하루 웹 로그(log) 양은 기가바이트 단위가 기록되고 있고 이 로그들은 사용자들의 성향을 분석하기 위한 자료로 사용된다.

또한 인터넷 사용자의 관심도의 증가로 인하여 개인 홈페이지 등의 네트워크 서비스를 제공하는 사용자도 증가하는 추세이다. 하지만 보안 시스템의 사용은 네트워크 서비스를 제공하는 사용자의 증가에 비해 크게 증가하지 않고 있다. 그에 따라 인터넷에서의 공격 시도 및 성공률도 점점 증가하고 있다[2].

이러한 네트워크 서비스 시스템의 취약점을 보완하기 위하여 침입 탐지 시스템(IDS: Intrusion Detection System)이 사용되고 있다. 침입 탐지 시스템은 공격을 탐지하기 위하여 IP 패킷의 정보를 바탕으로 탐지 규칙에 해당 되는지를 검사하고 공격 여부를 판단한다. 이 때 모든 상황에 대한 규칙 파일 적용하면 탐지 시 네트워크 서비스 시스템의 자원을 많이 소모하고 네트워크 서비스 제공 속도가 감소된다. 침입 탐지 시스템을 구성하는데 있어 네트워크 서비스의 상황에 맞는 침입 탐지 규칙을 설정하기 위해서는 보안에 대한 전문가의 지식이 필요하다. 이에 따라 개인 사용자의 경우 침입 탐지 시스템을 적절히 사용하지 못하는 경우가 많고, 네트워크 서비스 시스템에서의 취약점을 가지고 서비스를 제공하는 경우가 많다[3].

이와 같은 문제점들을 해결하기 위하여 이 논문에서는 로그 분석을 통하여 네트워크 서비스 시스템의 성향을 판단하고 이에 맞는 규칙을 추천하여 침입 탐지 시스템에 적용 및 해제하는 기법을 제안한다. 이를 위하여 대표적인 침입 탐지 시스템인 Snort를 이용하며, Snort 규칙을 구조화한 형태로 저장하고 이를 이용하여 Snort에서 생성된 로그를 분석함으로써 적절한 규칙을 추천한다.

제안하는 기법을 적용함으로써 분석 결과로부터 네트워크 서비스 시스템에서의 침입 유형과 침입 발생 빈도, 사용 유형 등의 요소를 이용하여 보안 전문가의 지식이 없이도 상황에 맞는 규칙을 추천하고, 적은 개수의 규칙을 적용하여 침입 탐지 시 시스템 자원을 효율적으로 사용할 수 있다.

본 논문의 구성은 2장에서 IDS와 Snort 그리고 Snort 규칙의 구조에 대하여 설명하고, 3장에서 Snort 규칙을 입력 받아 규칙을 구조화하는 방법과 구조화한 데이터를 이용하여 패킷 로그를 분석한 후 Snort에 규칙을 적용시키는 Snort 규칙 추천 시스템을 설계하고 구현한다. 4장에서는 3장에서 설계 및 구현한 내용을 검증한다. 마지막으로 5장에서는 결론을 내리며 향후 과제를 제시한다.

제 2 장 관련 연구

2.1 침입 탐지 시스템

침입 탐지 시스템은 일반적으로 서비스하는 시스템의 무결성, 기밀성, 가용성을 위협하는 내외부의 불법적인 사용자에 의한 정보시스템의 침입을 탐지하기 위해 구성된 시스템이다. 또한 감사 기록(audit record), 시스템 테이블(system table), 네트워크 트래픽 개요(network traffic summary)와 같은 정보원으로부터 사용자 행위에 대한 정보를 획득하고 분석함으로써 침입탐지를 수행한다[5].

침입 탐지 시스템은 다음과 같은 3가지의 요소로 구성된다.

- 데이터 수집(data collection)
- 데이터 분류(data classification)
- 데이터 보고(data reporting)

그림 1은 위 3가지 요소를 이용하여 외부로부터 들어오는 데이터를 수집하고 분석 및 축약을 하여 침입탐지 결과를 사용자에게 보고하거나 침입에 대응하는 침입 탐지 시스템의 동작 순서를 나타낸다[3].

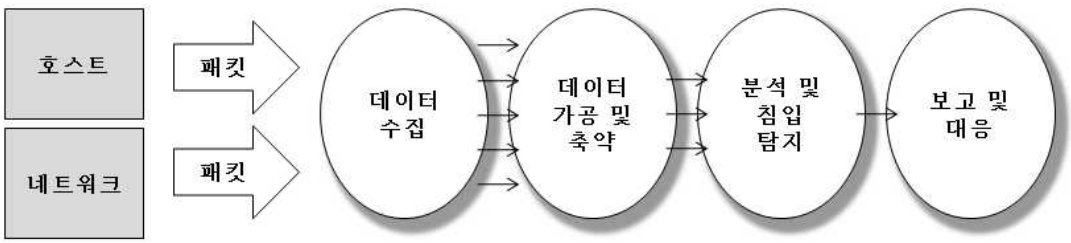

그림 1. 침입 탐지 시스템의 동작 순서
Fig. 1. Flow diagram of IDS operation

침입 탐지 시스템은 관리 방식, 탐지 방법, 네트워크 이용 여부에 따라 여러 가지 형태로 나눌 수 있다. 첫째, 관리 방식에 따라 침입 탐지 기능이 여러 컴퓨터에 분산되어 작동을 하는 분산형 시스템과 하나의 시스템에서 관리하는 중앙 집중형 시스템으로 나눌 수 있다.

둘째, 탐지 방법에 의한 분류는 비정상 침입 탐지(abnormal detection) 시스템과 오용 침입 탐지(misuse detection) 시스템으로 나누어진다. 비정상 침입 탐지 시스템은 평소 사용자 혹은 시스템에서 예상되는 행위에 대한 정보를 이용하여 이로부터 벗어난 범위를 기준으로 침입을 탐지하며 오용 탐지 시스템은 알려진 공격이나 시스템의 취약점을 이용하는 침입을 탐지한다[3,5].

셋째, 네트워크의 이용 여부에 따른 분류는 호스트 기반(host based) 시스템과 네트워크 기반(network based) 시스템으로 분류된다. 호스트 기반 시스템은 한 대의 컴퓨터에서 발생하는 이벤트에 대하여 모니터링을 수행하는데 반하여 네트워크 기반 시스템은 네트워크 전반에 걸쳐 발생하는 이벤트를 모니터링한다. 아울러 후자는 서로 다른 많은 호스트로부터 유입된 감사 추적 내용과 대조된 정보를 모니터링하거나 또는 네트워크 트래픽을 모니터링한다[3,5].

침입 탐지 시스템은 분류 방법에 따라 많은 종류로 표현이 가능하지만, 최근의 IDS는 분류별로 나누어진 모든 방법의 장점을 모아 상호 보완 관계를 이루며 구성된다.

최근 많이 쓰이는 침입 탐지 시스템의 종류로는 Snort[6], Bro[7], Bleeding Edge[8]들이 있으며 이와 같은 침입 탐지 시스템은 침입을 판단하기 위하여 탐지 규칙을 이용하고 들어오는 정보와 비교하여 침입과 오용의 여부를 판단한다. 이 탐지 규칙 집합은 주로 패턴 규칙을 나타내기 위하여 일반 고정 문자열뿐만 아니라 정규표현식을 이용하여 표현된다.

2.2 Snort

Snort는 대표적인 오픈소스 기반의 IDS이다. 주요 모듈으로는 패킷 스니퍼, NIDS, 패킷 로거가 있다. 패킷 스니퍼는 네트워크상의 패킷을 관찰하여 데이터 스트림이 오고갈 때 해당 내용을 읽어오는 기능을 한다. NIDS는 패킷 스니퍼에서 읽어온 정보를 이용하여 분석하고, Snort 규칙 파일로부터 등록된 적용 규칙을 이용하여 침입을 판단한다. 패킷 로거는 패킷 스니퍼에서 읽어온 정보의 내용과 규칙을 이용하여 판단한 내용의 정보를 기록으로 남긴다. Snort의 구조는 그림 2와 같고 Snort는 침입을 판단하거나 패킷 로거에 정보를 남기기 위한 판단 도구로 정형화된 규칙을 사용한다. Snort 규칙은 2.3에서 자세히 설명한다.

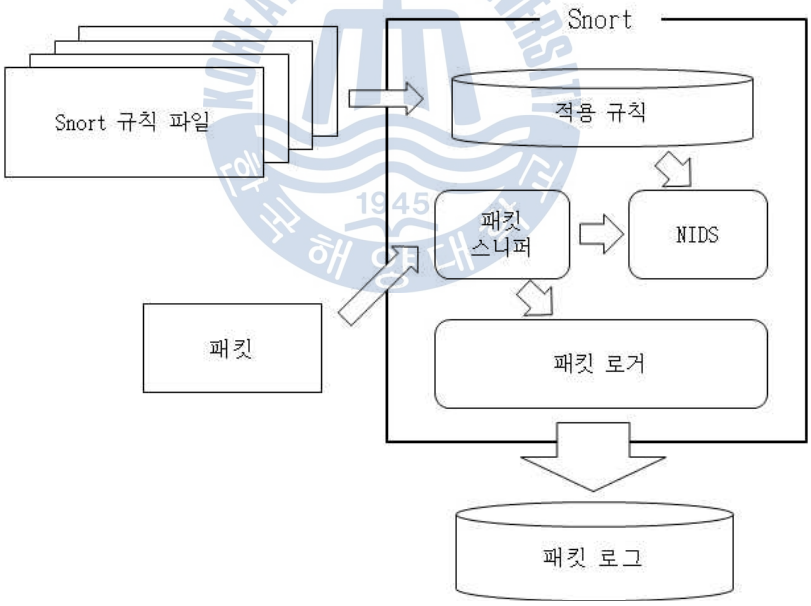

그림 2. Snort의 구조
Fig. 2. Architecture of Snort

2.3 Snort 규칙

Snort 규칙은 2.2절에서 설명한 Snort에서 침입을 판단하기 위한 판단 기준이 되는 규칙의 모음이다[4]. 규칙 형식은 특성화된 정보의 모음으로 구성되며 침입을 판단하기 위하여 protocol의 정보와 IP에 대한 정보, 어떠한 메시지를 출력할 것 인지에 대한 부분, 패킷 내부의 패턴을 판단하기 위하여 사용될 문자열 혹은 정규표현식 등의 내용을 담고 있다.

Snort 규칙은 크게 header와 option으로 구성된다. 표 1은 Snort 규칙의 구조를 나타내고 header와 option에 대하여 각각 2.3.1과 2.3.2에서 자세히 설명한다.

표 1. Snort 규칙 구조
Table 1. The structure of Snort Rule

header	option
• action • protocol • source&destination :: IP/netmask, ports ⋮	• alert message • information ⋮

2.3.1 header

Snort 규칙의 header 구조는 표 2와 같다.

표 2. Snort 규칙의 header 구조
Fig. 2. The structure of Snort rule header

키워드명	action	protocol	source ip/netmask	source port	direction	destination ip/netmask	destination port
예	alert	tcp	1.1.1.1	80	->	2.2.2.2	80

header는 action, protocol, ip/netmask, port, direction의 정보를 포함하고 있다.

· **action** : 해당 규칙에 해당 할 경우에 실행할 행동을 기술한다. action에서 사용할 수 있는 명령어는 다음과 같은 것이 있다.

- **alert** : option에 설정한 대로 사용자에게 경고를 발생한다.
- **log** : 패킷을 로그로 저장한다.
- **pass** : 패킷을 무시한다.
- **active** : 경고를 발생시킨 다음 다른 동적 규칙을 활성화한다.
- **dynamic** : active 키워드에 의해 활성화되기까지는 동작하지 않고, log 액션처럼 동작한다.
- **drop** : iptable을 작성하고 들어오는 해당 패킷을 막고 로그를 남긴다.
- **reject** : iptable을 작성하고 들어오는 해당 패킷을 막고 로그를 남긴다. 만약 UDP 프로토콜을 사용할 때 TCP 나 ICMP에 메시지가 도달하지 않을 경우 TCP 재시작의 명령을 보낸다.
- **sdrop** : iptable을 작성하고 패킷을 drop시키지만 로그를 남기지 않는다.

· protocol : 일반적으로 사용되는 프로토콜의 종류를 기입할 수 있다. tcp, icmp, udp등 미리 지정된 프로토콜의 정보를 기입하여 동작하는 프로토콜에 따라 규칙에 해당하는지 판단하는 기준이 된다.

· IP : 일반적인 이더넷(Ethernet)에서 사용하는 IP 형태의 주소 정보이다. 일반적으로는 하나의 IP를 기입하여 쓰게 되지만 경우에 따라 '!'를 이용하여 negation operator 기능을 수행할 수 있고 ','를 이용하여 복수의 IP의 표현이 가능하며 각 IP 뒤에 '/'를 이용하여 IP정보와 CIDR(Classless Inter Domain Routing) 또한 기술이 가능하다. CIDR에서 '/24'는 Class C network를 의미하고 '/16'은 Class B network를 의미하며 '/32'는 machine address를 의미한다.

사용의 예를 들면 아래와 같다.

- negation operator : !1.1.1.1 (작성한 IP 이외의 모든 IP)
- 복수의 IP 표현 : [1.1.1.0,2.2.2.0] (공백이 포함되어서는 안된다.)
- CIDR 표현 : 1.1.1.0/24

· port : 논리적인 접속 장소로 네트워크상의 특정 서비스를 지정하기 위하여 사용하는 번호이다. 상위 port 번호의 경우는 인터넷번호 할당 허가위원회(IANA:Internet Assigned Numbers Authority)에 의하여 0~1023번 사이의 번호가 특정 서비스에 해당하여 할당이 되어 있고 나머지 부분의 경우 응용 프로그램이 접속할 때마다 새롭게 부여된다.

Port의 사용의 예를 보면,

- 1:1024 ➡ 1번에서 1024번까지의 포트를 모두 지칭한다.
- :1024 ➡ 1024번 이하의 포트를 모두 지칭한다.
- 1024: ➡ 1024번 이상의 포트를 모두 지칭한다.

- !1:1024 ➡ 1번을 제외한 1번에서 1024번까지의 포트를 모두 지칭한다.

· **Direction** : 패킷의 흐름에 대한 정보를 나타내는 것으로 양옆에 있는 ip/netmask 정보들 사이에서 상호 관계에 대하여 정의하는데 이용된다.

- '->' : 왼쪽 ip/netmask에서 오른쪽으로 흐른다는 것을 의미한다.
- '<-' : 오른쪽 ip/netmask에서 왼쪽으로 흐른다는 것을 의미한다.
- '<>' : ip/netmask간에 양방향으로 흐른다는 것을 의미한다.

위에서 설명한 5가지의 정보의 조합으로 구성된 규칙 header와 패킷에서 오는 정보를 비교하여 해당사항이 있을 경우 2.3.2항에서 설명할 option의 검사를 진행하게 된다.

2.3.2 option

Snort 규칙의 option은 패킷에 실린 데이터의 정보를 분석하여 침입을 판단하기 위하여 사용되는 부분이다. option은 Snort 규칙에서 '(' , ')' 사이에 있는 부분을 말하는 것으로 그 형태는 각각의 옵션이 ';'으로 구분되어져 있으며 ':' 기호를 기준으로 앞부분은 option의 항목을 나타내고 뒷부분은 항목에 대한 내용을 나타낸다.

option은 General, Payload, non-payload, post-detection으로 크게 4가지의 형태가 존재한다. General, non-payload, post-detection 항목은 일반적으로 payload 항목이 나타내는 모호성을 피하기 위하여 작성하거나 작성된 규칙의 구분을 위하여 기입하게 된다. 대표적으로 sid, rev, dsize, classtype, priority 등이 많이 쓰인다[10]. sid는 Snort 규칙을 분류하고, 식별하고, 구분하기 위한 유일한 식별자로 사용된다. 100미만의 값은 미리 예약되어 있고 100 ~ 1,000,000 사이의 번호는 Snort에서 배포하는 규칙에서 사용된다.

1,000,000 이상의 번호는 사용자 정의 Snort 규칙에서 사용된다. rev는 원본 규칙이 수정되었을 때에 번호를 증가시키며 해당 규칙이 몇 번의 업데이트가 되었고 수정사항이 몇 번 있었는지에 대한 값을 기입하는 장소이다. dsize는 패킷의 Payload의 사이즈를 검사하는 option으로서 일반적으로는 버퍼 오버 플로우와 같은 비정상적인 패킷의 사이즈를 탐지하는데 유용하게 사용된다. classtype과 priority는 해당 규칙의 취약성을 점수화하기 위하여 사용된다. classtype를 문자열 형식으로 표현된 형태에 따라 등급을 high, middle, low, very low 등급으로 구분을 지어놨고, priority option는 숫자 형태로 표현함으로써 classtype보다 더 자세한 등급을 매길 수 있게 구분된다[10].

규칙에서 주로 사용하는 부분은 Payload 항목이고 Payload 항목 중 대표적인 키워드인 content, uricontent, pcre, flow의 형식은 표 3과 같다.

표 3. Snort 규칙의 Payload(content, uricontent, pcre, flow) 키워드
Table 3. Payload keywords part of Snort rule(content, uricontent, pcre, flow)

키워드	형식
content	[!]"<content string>";
	Content Modifires: nocase, depth, offset, distance, within, rawbytes, http_cilent_body, http_cookie, http_header, http_method, http_uri, fast_pattern
uricontent	uricontent: [!]<content string>;
pcre	pcre: [!]"(/<regex>/ |m<delim> <refex> <delim>) [ismxAEGRUB];
flow	flow: [(established | stateless)][, (to_client|to_server|from_client|from_server)] [, (no_stream|only_stream)];

content, uricontent, pcre, flow 의 각각의 설명은 다음과 같다.

content 키워드는 일반적으로 나열된 문자열을 검사하기 위하여 쓰여지는 구문으로서 큰따옴표(“”) 사이에 문자열의 형태로 사용되고 일반 문자열 및 ‘|’ 기호를 이용하여 바이너리 코드를 포함시켜 작성 가능하다. 문자열 안에서는 ‘;’, ‘₩’기호는 사용이 불가능하다. 단, ‘₩’기호의 경우 그 뒤에 의미를 나타내는 문자와 함께 사용할 수 있다. 또한 ‘!’기호를 이용하여 부정의 의미로 사용될 수 있다. nocase, depth 등과 같은 Content modifiers를 이용하여 좀 더 유동적인 규칙의 패턴을 작성할 수 있다. Content modifiers에 관한 설명은 표 4와 같다.

uricontent 키워드는 URI에 들어 있는 자료를 검사할 수 있도록 해주는 항목으로서, 기본적으로는 content 항목의 작성 요령과 같으나, 패킷 안의 데이터 내용이 아닌 URI에서 검사한다는 점만 다르다.

pcre[9] 키워드는 Perl에서 제공해주는 정규표현식을 이용하여 검사하는 도구로서, content와 같은 행동을 하지만 좀 더 강력한 Rule 작성에 유용하다.

flow 키워드는 규칙의 header와 비교하여 패킷의 흐름을 판단하고 검사하는 도구이다.

표 4 Snort 규칙의 Payload 키워드(etc.)

Table 4. Payload keyword part of S표 4 ort rule(etc.)

키워드	형식 및 설명
nocase	case;
	content에서 대소문자 구별을 하지 않게 한다.
depth	depht: <number>;
	검사할 바이트 수를 지정하고 CPU사이클을 최소화 하여 센서의 속도를 최적화하기 위해 사용한다.
offset	offset: <number>;
	검색을 시작할 패킷의 특정 위치를 지정하고, 다른 문자열의 일부분으로 포함된 문자열을 검색할 때 사용하다.
distance	distance: <byte count>;
	시작점 + 바이트 수 다음부터 해당 문자를 검사한다.
within	within: <byte count>;
	시작점부터 바이트 수 이내 범위에서 해당문자를 검사한다.
byte_test	byte_test:<byte to convert>, [!]<operator>, <value>, <offset>[,relative][,endian][,<number type>,string];
	마지막 패턴이 일치한 이후의 값을 비교하기 위하여 사용한다.
byte_jump	byte_jump: <byte to convert>, <offset> [,relative][,multiplier<multiplier value>][,big][,little][,string][,hex][,dec][,oct][,align][,from_beginning][,post_offset<adjustment value>];
	마지막 패턴이 일치한 이후의 값을 이용하여 이동 하는데 사용한다.

제 3장 Snort 규칙 추천 시스템의 설계 및 구현

이 장은 제안하는 Snort의 패킷 로그를 분석하여 Snort 규칙을 추천하고 적용하는 Snort 규칙 추천 시스템을 제안한다. 그림 3는 제안하는 시스템의 구성과 Snort와의 관계를 나타낸다.

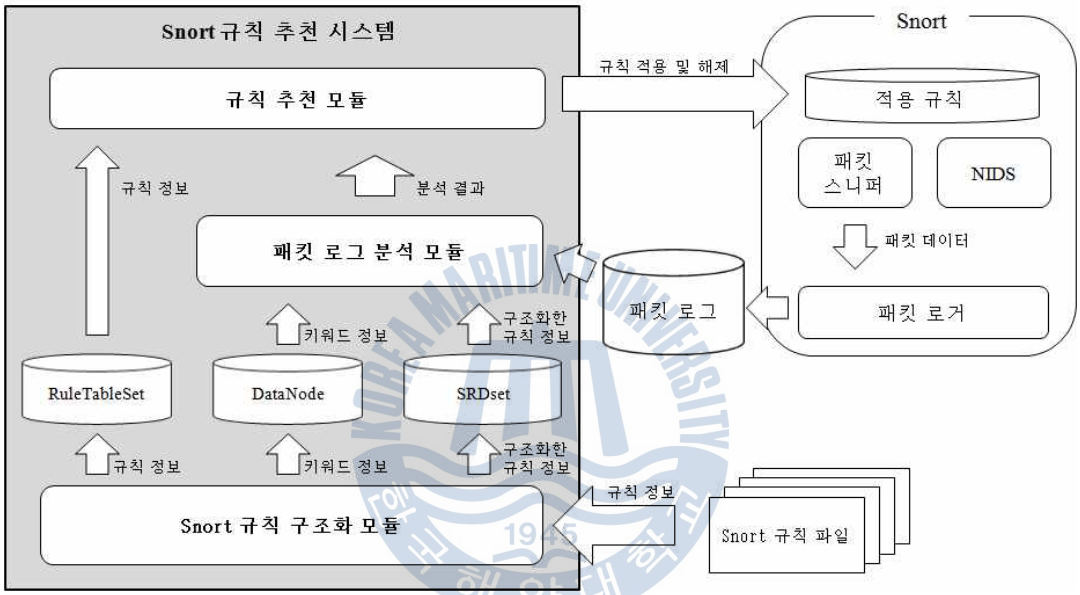

그림 3. Snort 규칙 추천 시스템의 구성 및 Snort와의 관계

Fig. 3. Achitecture of Snort rule recommendation system and its relation with Snort

그림 3과 같이 Snort 규칙 추천 시스템은 Snort 규칙 구조화 모듈, 패킷 로그 분석 모듈, 규칙 추천 모듈로 구성된다. Snort 규칙 구조화 모듈은 패킷 로그의 분석을 위하여 Snort 규칙 파일로부터 Snort 규칙의 정보를 구조화하는 모듈이다. 패킷 로그 분석 모듈은 Snort 패킷 로거로부터 발생하는 패킷 로그 정보를 DataNode 데이터베이스의 키워드 정보, SRDset 데이터베이스의 구조화 규칙 정보와 비교하여 침입을 판단하고 네트워크 서비스 시스템의 사용 성향을

판단한다. Snort 규칙 추천 모듈은 패킷 로그 분석 모듈에서의 분석 결과와 RuleTableSet 데이터베이스의 규칙 정보를 비교하여 침입 또는 성향에 따른 규칙 집합을 추천하고 적용 및 해제하는 기능을 수행한다.

3.1 Snort 규칙 구조화 모듈

Snort 규칙 구조화 모듈은 Snort 규칙 파일에서 읽어 들인 정보 중 필요한 정보를 추출하고 구조화하여 데이터베이스에 저장하는 기능을 수행한다.

앞서 2.3절에서 설명한 바와 같이 Snort 규칙은 header와 option으로 구성되어 있으며 Snort 규칙 구조화 모듈은 각각의 목적에 따라 다음과 같이 6개의 항목으로 Snort 규칙을 구조화한다.

① header의 IP와 Port 정보, 패킷의 흐름을 나타내는 정보

- 패킷 로그의 헤더 정보와 비교에 사용

② option의 content, uricontent, pcre 정보

- 패킷 로그의 데이터 정보와 비교에 사용

③ option의 flow 정보

- 패킷 로그의 헤더에서 정보의 흐름을 비교에 사용

④ ②,③의 정보를 구체화하는 option 정보

- 패킷 로그의 데이터 정보와 비교 시 ②,③의 정보와 함께 구체적인 형식과 구조의 비교에 사용

⑤ Snort 규칙의 구조화 규칙 정보

- 패킷 로그의 침입 탐지에 사용

⑥ Snort 규칙 정보

- Snort 규칙 추천에 사용

① ~ ④항목의 구조화 정보는 DataNode 데이터베이스에 저장되며 ⑤항목의 구조화 정보는 SRDset 데이터베이스에 저장된다. ⑥항목의 구조화 정보는 RuleTableSet 데이터베이스에 저장되며 각각의 데이터베이스에 저장된 구조화 정보는 패킷로그 분석 및 규칙추천에 활용된다.

3.1.1 DataNode 데이터베이스

DataNode 데이터베이스는 Snort 규칙 파일에서 입력 받은 모든 규칙을 구조화하여 저장하는 공간이며 패킷 로그 분석 시 SRDset 데이터베이스에 저장된 구조화 규칙 정보의 ID를 요청하면 그에 해당하는 정보를 제공한다. 이를 위하여 DataNode 데이터베이스는 ProtocolNode 클래스, GroupNode 클래스, ContentNode 클래스로 관리한다.

(1) ProtocolNode 클래스

ProtocolNode 클래스는 Snort 규칙의 header 정보를 저장하고 관리한다. header는 패킷 로그의 헤더와 비교하기 위하여 protocol, IP, port 등의 정보를 가지고 있는 모든 키워드가 필요하다. 각각의 키워드는 공백으로 구분되어 있으므로 쉽게 분리할 수 있다.

표 5는 ProtocolNode 클래스의 구성을 나타낸다.

표 5. ProtocolNode 클래스 구조
Table 5. Structure of ProtocolNode class

맴버 변수	데이터 포맷	비고
PID	string	ProtocolNode 클래스의 구분자로 사용되는 ID
Action	string	규칙에 해당 시 하는 행동
Protocol	string	프로토콜 정보
Destination_IP	string	도착지의 IP 정보
Destination_Port	string	도착지의 port 정보
Source_IP	string	출발지의 IP 정보
Source_Port	string	출발지의 port 정보
Direction	string	패킷의 흐름
CountScore	double	패킷 로그 분석 시 네트워크 서비스 시스템의 성향을 판단하여 규칙을 추천하기 위한 값

특히, 표 5에서 PID는 이전에 저장된 구조화 규칙과 중복여부를 검사하여 유일한 값으로 부여되며 CountScore는 패킷 로그 분석 시 기록하여 네트워크 시스템의 사용 성향을 판단하게 하는 수치정보이다.

(2) ContentsNode 클래스와 GroupNode 클래스

ContentsNode 클래스는 패킷 로그의 데이터 정보와 서로 비교하고 어떠한 규칙에 해당하는지 판단하기 위하여 Snort 규칙의 option 중 content, uricontent, pcre, flow 키워드와 4가지 키워드를 구체화하는 속성 값을 저장하고 관리한다. 패킷의 데이터와 비교 시 option에서 content, pcre, uricontent, flow 키워드 타입으로 모든 규칙 정보 중 대부분에 해당하는 규칙

을 비교할 수 있으므로 4개의 키워드를 적용하였다. 표 6은 ContentsNode 클래스의 구조를 나타낸다.

표 6. ContentsNode 클래스 구조
표 6. Structure of ContentsNode class

맴버 변수	데이터 포맷	비고
CID	string	ContentsNode 클래스의 구분자로 사용되는 ID
Contents_Type	string	content, uricontent, pcre, flow 중 한가지
Contents_Attribute	string	Contnets_Type에 해당하는 내용
Modify_Keyword	vector<string[]>	Contents_Type의 키워드를 구체화하는 정보
CountScore	double	패킷 로그 분석 시 네트워크 서비스 시스템의 성향을 판단하여 규칙을 추천하기 위한 값

또한 GroupNode 클래스는 ContentsNode 클래스의 집합을 가지고 있으면서 ContentsNode 클래스를 대표하는 역할을 한다. option에서 추출한 ContentsNode의 키워드 속성 내용 중 앞의 두 글자를 이용해서 그룹화하여 GID를 부여한다. 표 7은 GroupNode 클래스의 구성을 나타내며 GID, 여러 개의 ContentsNode 클래스, CountScore를 가지고 있다.

표 7. GroupNode 클래스 구조
Table 7. Structure of GroupNode class

맴버 변수	데이터 포맷	비고
GID	string	GroupNode 클래스의 구분자로 사용되는 ID
GroupContents	vector<ContentsNode>	GroupNode 클래스에 속해 있는 ContnetsNode 클래스 정보
CountScore	double	패킷 로그 분석 시 네트워크 서비스 시스템의 성향을 판단하여 규칙을 추천하기 위한 값

(3) DataNode 클래스

DataNode 클래스는 ProtocolNode 클래스, GroupNode 클래스, ContentsNode 클래스를 관리하기 위한 클래스로서 ProtocolNodeSet 클래스, GroupNodeSet 클래스로 구성된다. 표 8은 DataNode 클래스의 구성을 나타내며 ProtocolNodeSet은 여러개의 ProtocolNode 클래스를 관리하며 GroupNodeSet은 여러개의 GroupNode 클래스를 관리한다.

표 8. DataNode 클래스의 구조
Table. 8. Structure of DataNode class

맴버 변수	데이터 포맷	비고
ProtocolNodeSet	vector<ProtocolNode>	ProtocolNode 클래스의 집합
GroupNodeSet	vector<GroupNode>	GroupNode 클래스의 집합

그림 4는 DataNode 데이터베이스에 Snort 규칙을 입력한 예이다. 그림 4에서 윗부분은 ProtocolNodeSet 클래스의 입력 내용으로 여러개의 ProtocolNodeSet

클래스로 구성되고 그 아래 부분은 GroupNodeSet 클래스의 입력 내용으로 여러 개의 GroupNode 클래스를 포함하고 있다. 또한 GroupNode 클래스는 ContentsNode 클래스를 포함하고 있다. 예를 들어 “gid:si” 의 경우 구성요소인 “content:SITE” 와 “content:SIGN” 의 경우 “SI” 2 글자를 이용하여 그룹화 된다. 각 클래스들의 마지막의 숫자는 CountScore로 패킷 로그를 분석할 시 사용하게 되는 요소로 초기값은 0으로 저장된다.

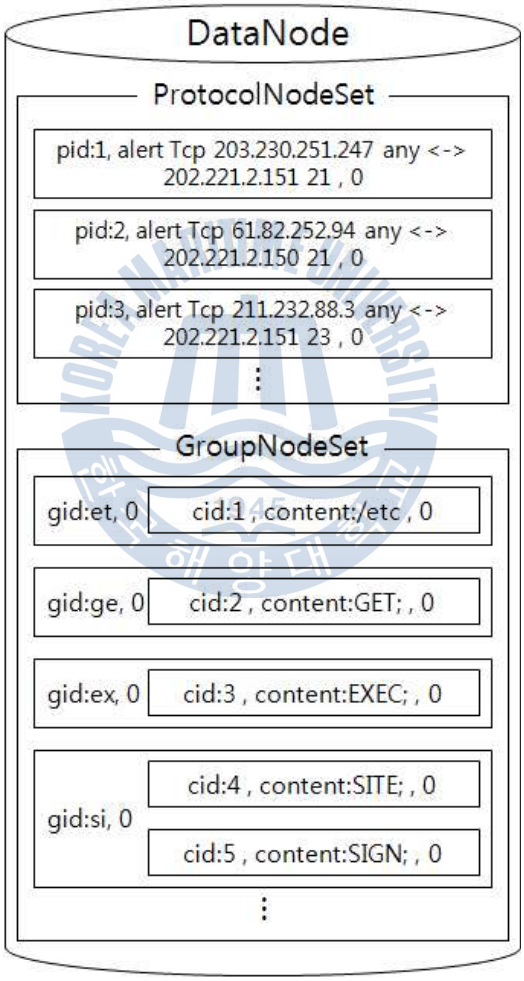

그림 4. DataNode 데이터베이스의 예
Fig. 4. Example of DataNode database

3.1.2 SRDset 데이터베이스

SRDset 데이터베이스는 DataNode 데이터베이스에 저장된 Snort 규칙의 구조화 정보들로부터 각각의 ID를 이용하여 규칙을 표현한 구조화 규칙 데이터(SRD :Structured Rule Data)의 정보를 가지고 있다. 구조화 규칙 데이터를 저장하고 관리하는 SRD 클래스는 Snort 규칙의 고유 번호인 sid와 DataNode 클래스의 ProtocolNode 클래스, GroupNode 클래스, ContentsNode 클래스의 ID의 조합으로 이루어진 클래스이다. ProtocolNode 클래스, GroupNode 클래스, ContentsNode클래스 각각의 ID는 패킷 로그를 분석하는데 있어 DataNode 클래스로부터 정보를 읽어오기 위한 키이고 sid는 패킷 로그 분석을 통한 결과를 규칙 추천 모듈로 보내어 RuleTableSet 데이터베이스에 저장된 해당 규칙을 하기 위한 식별번호다.

SRD 클래스의 구조는 표 9와 같다. SRD 클래스는 하나의 sid와 PID는 반드시 있어야 하고 GID와 CID는 Snort 규칙에 따라 존재하지 않을 수도 있으나 존재할 경우 GID와 CID의 쌍으로 존재한다. 또한 GID와 CID의 쌍은 여러 개가 있을 수 있다. GID와 CID의 쌍은 GCNodeID클래스로 관리된다.

표 9. SRD 클래스의 구조
Table 9. Structure of SRD class

클래스 명	맴버 변수	데이터 포맷
SRD	sid	string
	PID	string
	GroupNodeIDset	vector<GCNodeID>
GCNodeID	GID	string
	CID	string

그림 5는 SRDset 데이터베이스에 Snort 규칙을 구조화 규칙 데이터로 저장한 예를 나타낸다. 각각의 구조화 규칙 데이터는 Snort 규칙의 고유번호인 sid와 ProtocolNode, GroupNode, ContentNode 클래스의 ID의 조합으로 되어있고 sid를 제외한 ID들의 순서를 이용하여 차례대로 패킷 로그를 검사하면서 규칙에 해당되는지 판단하게 된다.

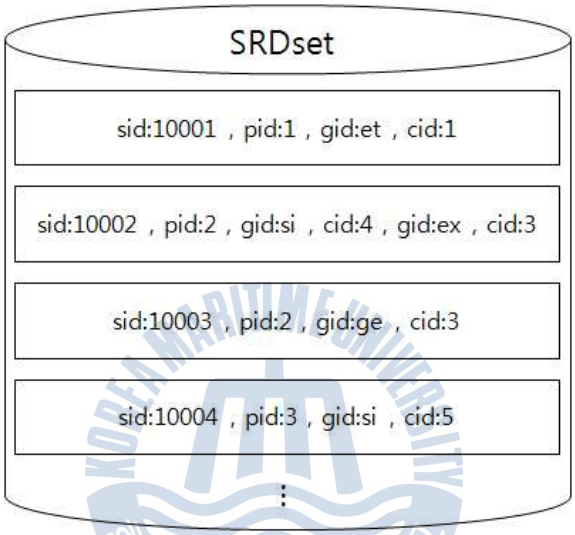

그림 5. SRDset 데이터베이스의 예
Fig. 5. Example of SRDset database

3.1.3 RuleTableSet 데이터베이스

RuleTableSet 데이터베이스는 Snort에 규칙을 적용시키기 위한 실제의 규칙 정보를 저장한다. RuleTableSet 데이터베이스에서 각각의 데이터는 RuleTable 클래스를 이용하여 관리한다. RuleTable 클래스는 표 10과 같이 규칙의 고유한 식별번호인 sid, 규칙 데이터인 RuleData 그리고 남은 적용시간인 Remain_Time으로 구성된다.

표 10. RuleTable 클래스의 구조
Table 10. Structure of RuleTable class

맴버 변수	데이터 포맷	비고
sid	string	규칙의 고유 식별 번호
RuleData	string	실제 규칙 데이터
Remain_Time	double	남은 적용 시간

그림 6은 RuleTableSet 데이터베이스의 예이다. 각 데이터에서 앞의 숫자는 Snort에 규칙 적용 및 해제를 위하여 사용되는 남은 적용 시간이고 뒤의 문자열은 규칙을 적용 및 해제할 때 사용되는 Snort 규칙의 원본부분이다.

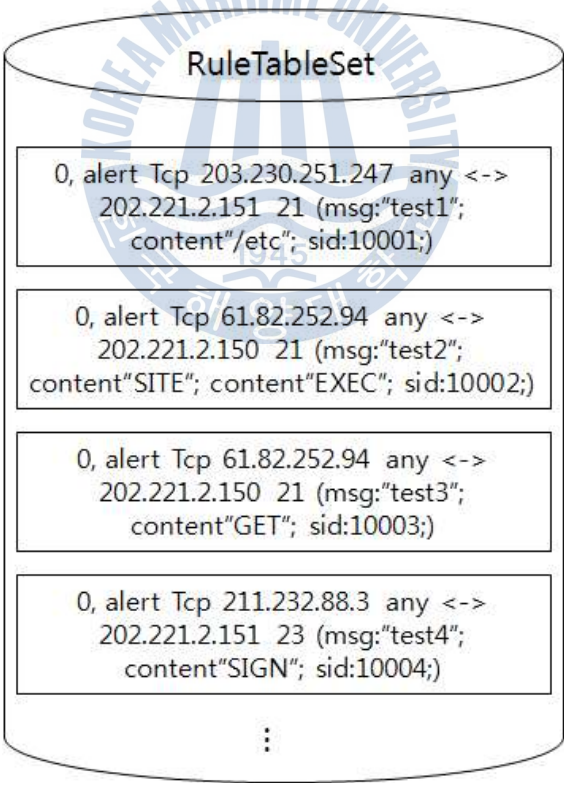

그림 6. RuleTableSet 데이터베이스의 예
Fig. 6. Example of RuleTableSet database

그림 7은 Snort 규칙 구조화 모듈의 처리과정을 나타낸 흐름도이다. Snort 규칙을 입력하여 header, option의 정보를 분리하고 중복검사를 통해 각각 ProtocolNode 클래스, ContentsNode 클래스, GroupNode클래스를 생성하고 이들 클래스의 ID 이용하여 그에 해당하는 구조화 규칙정보를 생성하여 SRDset 데이터베이스에 저장한다. 또한 실제 규칙 정보는 RuleTableSet 데이터베이스에 저장한다.

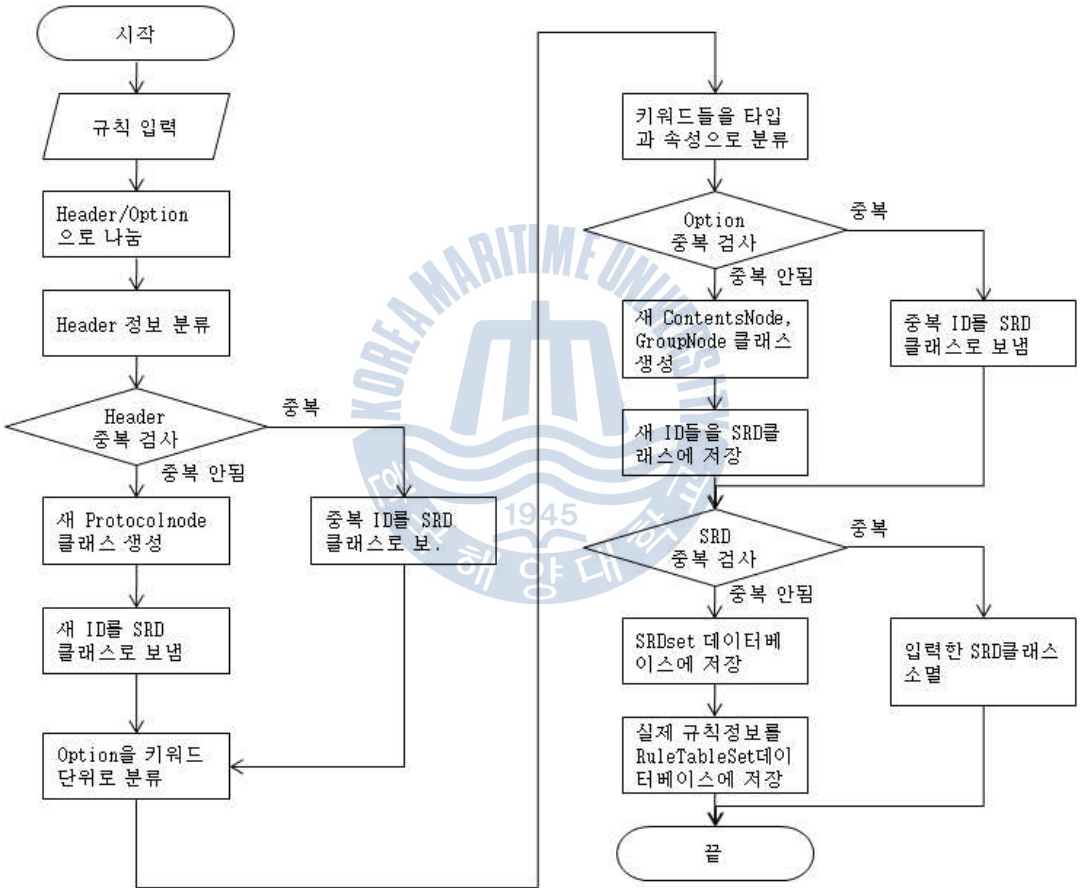

그림 7. Snort 규칙 구조화 모듈 처리과정

Fig. 7. Processing flow of Snort rule structuring module

3.2 패킷 로그 분석 모듈

패킷 로그 분석 모듈은 문자열 형태로 변환된 패킷 로그와 구조화된 Snort 규칙을 이용하여 해당 패킷 로그의 내용이 침입인지 아닌지 판단하고, 패킷 로그 분석을 통하여 해당 네트워크 서비스 시스템의 사용 성향을 판단하는 모듈이다. 패킷 로그의 내용에 대한 침입 여부의 판단을 3.1.2에서 설명한 SRDset 데이터베이스와 3.1.1에서 설명한 DataNode 데이터베이스의 구성요소를 이용하여 그림 8의 상태도에 따라 Snort 규칙에 해당하는지 분석함으로써 결정한다. 또한 해당 네트워크 서비스 시스템의 사용 성향을 판단하기 위하여 그림 8의 상태도에 따라 DataNode 데이터베이스의 관련 데이터의 CountScore를 증가시켜 그 값을 이용하여 판단한다.

그림 8은 패킷 로그 분석을 위해 거치는 상태를 나타낸 것으로 상태도의 구성은 패킷 로그를 이용하여 SRDset 데이터베이스의 구조화 규칙을 탐색할 수 있도록 구성된다.

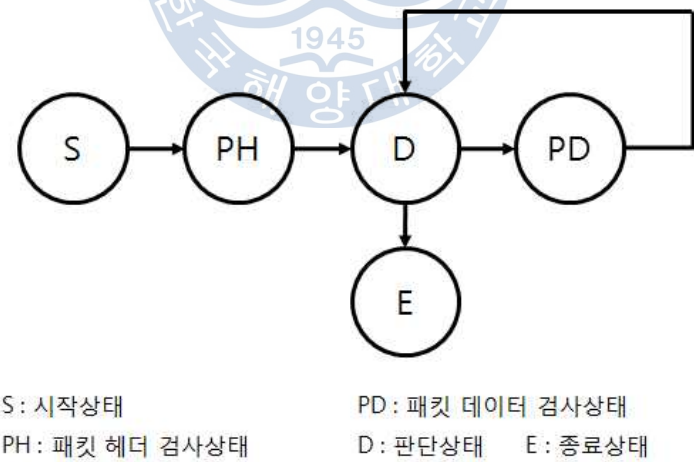

그림 8. 패킷 로그 분석 모듈의 상태도

Fig. 8. State diagram of packet log analysis module

그림 8에서 S는 시작상태, PH는 패킷 헤더 검사상태, PD는 패킷 데이터 검사상태, D는 판단상태, E는 종료상태를 나타낸다.

S 상태에서는 Snort로부터 받아온 패킷 로그 데이터를 시간, 접속자의 기준으로 분할하여 입력한다.

PH 상태에서는 패킷의 헤더 정보와 DataNode 데이터베이스안의 ProtocolNode 정보를 비교하고 해당 내용이 있을 경우 PID를 이용하여 해당하는 SRD의 집합을 찾는다. PH 상태는 반드시 한번 거쳐야 되며 한번 이상의 탐색은 하지 않는다. 그리고 더 이상의 패킷데이터가 없을 경우 패킷 로그 내용의 침입 여부는 패킷 헤더만을 이용하여 D 상태에서 규칙의 해당 여부를 결정한다.

PD 상태에서는 패킷 로그의 데이터 부분과 PH 상태에서 나온 해당 가능성이 있는 DataNode 데이터베이스안의 GroupNode 클래스와 ContentsNode 클래스정보를 비교한다. 그리고 해당 내용이 있을 경우 GID와 CID를 이용하여 해당하는 SRD집합을 찾는다. PD 상태는 D 상태를 거쳐서 여러 번의 탐색을 할 수 있다.

D 상태는 패킷 로그의 내용이 규칙에 해당 되는지 안 되는지에 관하여 판단하는 노드이다. D 상태에서 하는 기능은 첫째, 거쳐 온 노드의 정보를 읽어 들이고 둘째, 거쳐 왔던 노드의 정보를 이용하여 저장된 SRD에 해당 되는 정보가 있는지 판단한다. 셋째, 둘째에서 해당되는 정보가 없을 때 앞으로 나올 수 있는 SRD 클래스의 목록을 추출하여 정보를 저장한다. 만약 SRD에 해당하지 않는다면 다음 해당 가능성이 있는 노드 정보를 불러 오고 SRD 클래스에 해당 하였다면 로그 데이터의 검색을 종료하는 E 상태로 해당 SRD 클래스의 정보를 전달한다. 마지막으로 패킷 로그의 내용에 해당하는 SRD 클래스 정보가 없을 경우 E 상태로 침입 여부가 없다는 결과를 전달한다.

E 상태는 D 상태로부터 전달 받은 정보를 RuleTableSet 데이터베이스와 비교하고 해당하는 RuleTable 클래스의 규칙을 Snort에 적용시키기 위하여 3.3절에서 설명하는 규칙 추천 시스템으로 규칙 정보를 전달한다.

그리고 패킷의 정보에 따라 검사 상태를 거치면서 DataNode 데이터베이스의 해당 클래스에 CountScore를 증가시킨다. 이와 같이 현재 들어오는 패킷의 성

향을 파악함으로써 시스템의 서비스 제공의 유형을 알 수 있고 그에 해당하는 규칙을 적용 시킬 수 있다. 일반적으로 한 번의 잘 못된 행위가 오는 경우가 정상적인 패킷이 올 경우의 1/100정도의 확률로 나타난다[1]. 잘못된 행위란, 침입으로 판단되는 패킷과 오작동으로 들어오는 패킷의 합을 말하는데 오작동의 비율이 많고 침입의 행위는 패킷의 정보 중에서 극히 드물게 발생한다. 따라서 이 논문에서는 침입의 유형이라 판단되는 ContentsNode 클래스에 검색이 될 때 CountScore를 1을 증가 시키는 것을 기준으로 GroupNode 클래스는 0.01, ProtocolNode 클래스는 탐색 시 0.0001의 CountScore를 증가한다.

3.2.1 패킷 로그 변환

패킷 로그는 호스트로 들어오는 패킷 또는 나가는 패킷을 저장한 것이다. 이와 같은 패킷 로그의 내용을 분석하기 위해서 우선적으로 패킷의 구조를 알아야 한다. 패킷의 구조는 그림 9와 같이 16진수 코드로 구성되어있다. 패킷의 구조는 헤더와 데이터로 나누어지는데 헤더는 tcp, udp, icmp, snmp등 프로토콜에 따라 자료의 크기와 내용이 달라진다[11].

```
21:24:49.489956 IP 192.168.108.130.ssh > 192.168.108.1.59887: tcp 52
        0x0000:  4510 005c 78c1 4000 4006 67f6 c0a8 6c82
        0x0010:  c0a8 6c01 0016 e9ef 0d94 6fb7 5480 d602
        0x0020:  5018 041a 64ca 0000 8c45 cbf2 9050 1d45
        0x0030:  d2ac 0578 92f1 7c09 466f 3fde 3821 9e59
        0x0040:  fa45 3e96 48cf 045c f0bb f382 46dd 4acc
        0x0050:  7d1c ddb3 e301 9d3e d00a 6a49
21:24:49.490010 IP 192.168.108.130.ssh > 192.168.108.1.59887: tcp 116
        0x0000:  4510 009c 78c2 4000 4006 67b5 c0a8 6c82
        0x0010:  c0a8 6c01 0016 e9ef 0d94 6feb 5480 d602
        0x0020:  5018 041a edc2 0000 9d65 e26f 05e9 7c93
        0x0030:  3486 e738 5d9f 9878 a0c1 fc51 2813 7c85
        0x0040:  6ff2 575d 1eb0 15fd 52a9 365d de90 f764
        0x0050:  38ea 1ce9 0faa 7a2e 5da6 c5ed c09f b138
        0x0060:  d036 fde4 fcd8 f49b 4cb3 5789 df49 5ce7
        0x0070:  e3f9 4857 3b48 125c 9b48 d3f7 52cf 9ca6
        0x0080:  3f62 3247 bd4c cbd8 8446 630d bf5f 60d9
        0x0090:  fbae ea58 a858 1f28 da13 a914
21:24:49.490015 IP 192.168.108.1.59887 > 192.168.108.130.ssh: tcp 0
        0x0000:  4500 0028 3a97 4000 8006 6664 c0a8 6c01
        0x0010:  c0a8 6c82 e9ef 0016 5480 d602 0d94 705f
        0x0020:  5010 0103 c280 0000 0000 0000 0000
21:25:18.811330 IP 192.168.108.1.59887 > 192.168.108.130.ssh: tcp 52
        0x0000:  4500 005c 3a98 4000 8006 662f c0a8 6c01
        0x0010:  c0a8 6c82 e9ef 0016 5480 d602 0d94 705f
        0x0020:  5018 0103 f9c7 0000 90c9 2209 ee3b 432f
        0x0030:  926f 7dd5 3962 4083 0be7 ed5b ee90 852f
```

그림 9. 패킷 로그의 예

Fig. 9. Example of packet log

16진수로 구성되는 패킷 로그는 헤더 정보를 분리한 후 ProtocolNode 클래스와 비교하고 패킷의 내용적인 측면인 데이터는 GroupNode 클래스와

ContnetsNode 클래스를 이용하여 비교한다.

그림 10은 16진수로 구성된 패킷 로그를 문자열의 형태로 변환한 예다. 변환된 정보를 이용하여 패킷 로그를 분석한다.

그림 10. 문자열형태로 변환한 패킷 로그

Fig. 10. packet log for converted to strings

3.2.2 패킷 로그 분석

패킷 로그 분석은 패킷 로그를 변환하여 문자열 형태의 정보로 표시한 내용을 이용하여 분석한다. 앞서 설명한 그림 8의 상태도에 따라 진행하면서 침입 및 성향을 판단하며 그 결과를 3.3절의 규칙 추천 모듈로 전달한다.

그림 11은 패킷 로그 분석에 사용되는 데이터베이스들 간의 연관 관계를 나타낸 것이다. 그림 11에서 음영으로 처리된 부분은 그림 10의 문자열 형태로 만들어진 패킷 로그 첫 문장의 정보와 서로 연관 관계를 나타내며 sid:10002에 해당되는 것을 알 수 있고 이 패킷 로그의 정보가 침입에 해당되는 것을 판단할 수 있다.

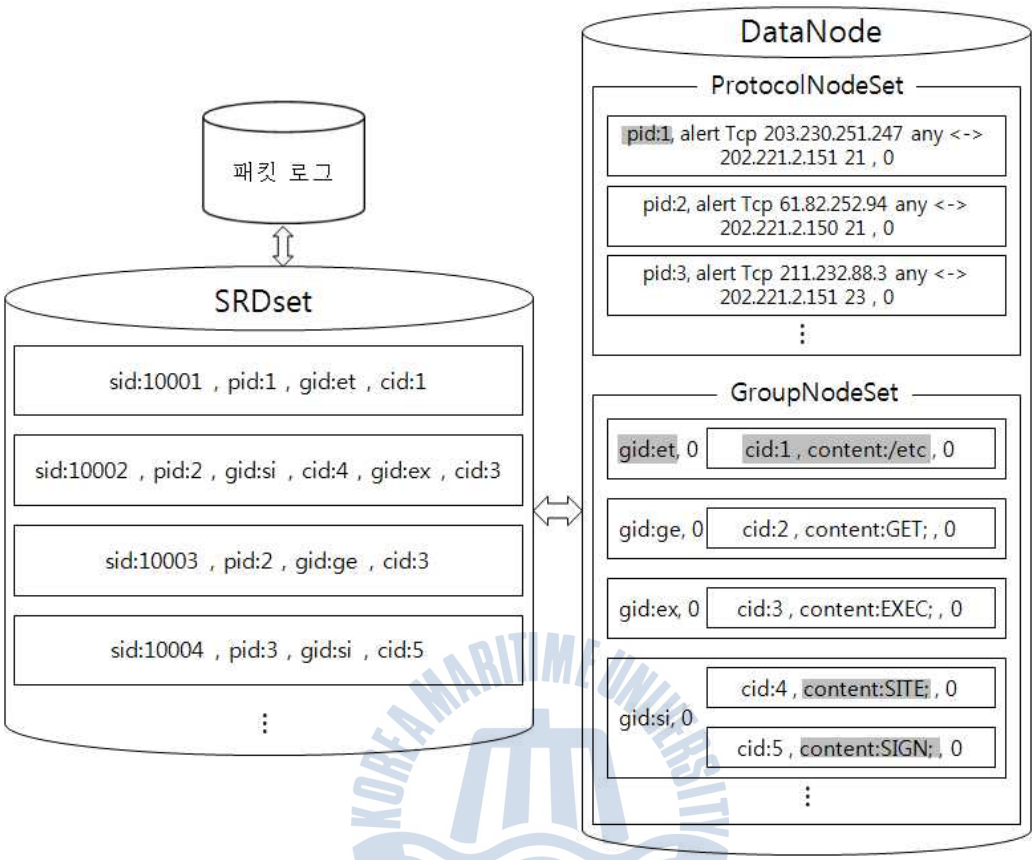

그림 11. 패킷 로그 분석 모듈에서의 SRDset과 DataNode의 관계
Fig. 11. Relation of SRDset and DataNode in packet log analysis module

그림 12는 패킷 로그 분석 모듈의 상태 변환을 통하여 패킷 로그를 분석하는 예를 나타낸다. 그림 11과 같은 SRDset 데이터베이스와 DataNode 데이터베이스의 입력이 Snort 규칙으로부터 입력되고, 그림 10과 같은 로그를 분석할 경우 "S->PH(pid:2)->D-> PD(gid:2,cid:2)->D->E"상태 변화를 통하여 분석이 이루어진다.

시작노드(S)에서 그림 10과 같은 패킷 로그를 입력받아 패킷 헤더 검사상태(PH)로 넘어가고, 패킷 로그의 헤더부분의 정보를 비교하여 "pid:2"라는 정보를 추출하고 DataNode에 "pid:2"에 해당하는 ProtocolNode 클래스의 CountScore을 0.0001 증가시키고 판단상태(D)로 넘어 간다. 판단상태에서는 해

당하는 SRD 클래스가 있는지 판단하고 패킷 로그의 데이터가 존재하므로 나올 가능성이 있는 "pid:2"의 정보를 가진 “sid:10002”와 “sid:10003”의 정보를 저장한다. 그 다음 PD상태로 넘어가서 패킷 로그의 데이터부분과 나올 가능성이 있는 SRD 클래스의 정보를 이용하여 “GET"란 정보를 가진 "gid:ge" 와 "cid:2”의 정보를 추출하고 DataNode에 해당하는 GroupNode 클래스의 CountScore와 0.01, ContentsNode 클래스의 CountScore를 각각 0.01과 1을 증가시킨다. 다시 판단상태로 넘어가서 해당하는 SRD 클래스의 정보가 있는지 검사하여 패킷 로그의 정보가 “sid:10003"에 해당한다는 정보가 나오면 종료상태(E)상태로 넘어가 결과를 규칙 추천 모듈로 전달한다.

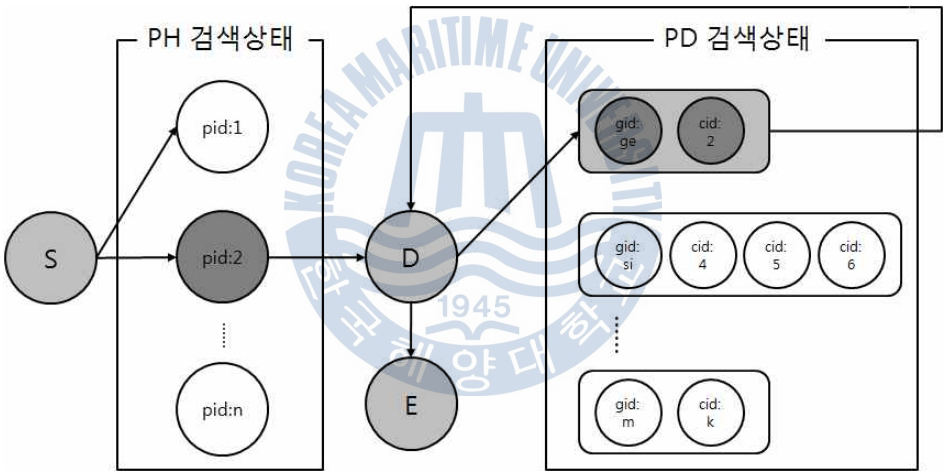

그림 12. 패킷 로그 분석 모듈의 상태 변화의 예

Fig. 12. Example of state transition in packet log analysis module

그리고 패킷의 정보를 따라 검색 상태를 거치면서 해당 클래스의 CountScore을 증가시킴으로서 현재 들어오는 패킷의 성향을 파악함으로써 시스템의 서비스 제공의 유형을 알 수 있고 그에 해당하는 규칙을 적용 시킬 수 있다.

3.3 규칙 추천 모듈

규칙 추천 모듈은 패킷 로그 분석 모듈의 분석결과를 이용하여 네트워크 서비스 시스템의 특성을 고려한 규칙을 추천하고, Snort에 해당 규칙을 설정하는 기능을 수행한다. 이를 위하여, 첫째, 로그 분석을 통하여 규칙이 추천되면 실제 Snort 규칙에 포함된 사항을 검사하여 규칙 설정 기간, 규칙의 중요도 등을 판단하고, 둘째, 추천된 규칙을 Snort에 적용한다. 셋째, 적용하도록 설정된 기간이 만료된 규칙에 대하여 규칙을 확인하여 해제한다.

그림 13은 규칙 추천 모듈에서의 자료의 흐름을 나타낸 그림이다. 패킷 로그 분석 모듈로부터 sid 목록을 전달받아 RuleTableSet 데이터베이스에 저장된 데이터의 sid와 비교하여 해당하는 규칙을 추천한다. 규칙을 추천하는 과정에서 SRDset 데이터베이스와 DataNode 데이터베이스의 CountScore정보를 이용하여 해당 규칙의 적용 시간 계산한다.

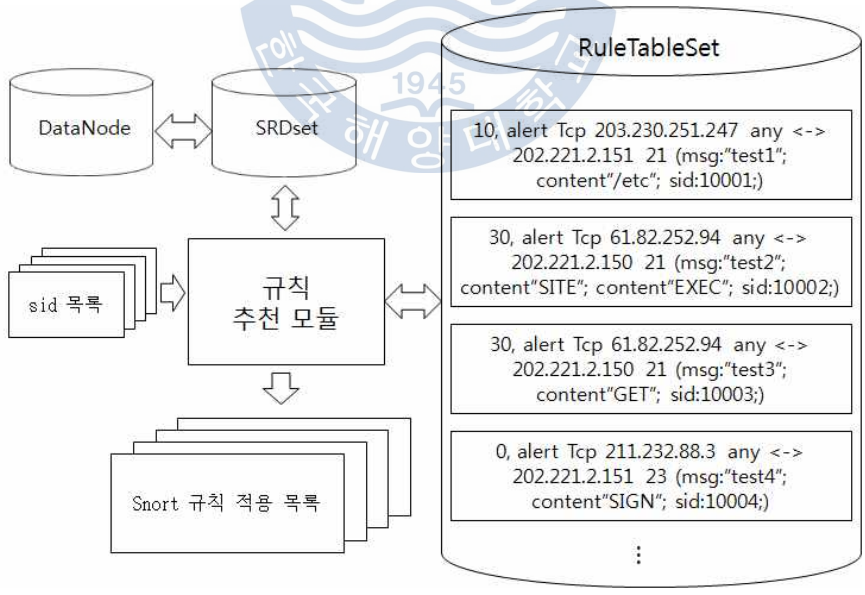

그림 13. 규칙 추천 모듈에서의 자료의 흐름

Fig. 13. Flow of datas in Rule recommendation module

3.3.1 규칙 추천 방법

규칙 추천 방법은 두 가지의 방법으로 나누어진다. 첫 번째는 패킷 로그 분석 모듈로부터 침입으로 판단되어 전달 받은 sid목록을 RuleTableSet 데이터베이스에 저장된 규칙의 sid와 비교하여 해당 규칙을 Snort에 적용하는 방법이고 두 번째는 패킷 로그를 분석할 때 발생하는 CountScore을 사용하여 시스템 사용 유형을 판단하고 그에 해당하는 규칙을 Snort에 적용하는 방법이다.

개별 규칙의 CountSocre는 식 (1)로 구할 수 있다. Rcs_i는 규칙 i의 CountScore을 나타내며 규칙 i를 구성하는 ProtocolNode 클래스의 CountScore Pcs_i 의 값과 GroupNode 클래스의 CountScore인 Gcs_i와 ContentsNode 클래스의 CountScore인 Ccs_i 합의 평균값을 더하여 나타낸 값이다.

$$Rcs_i = Pcs_i + \left(\frac{\sum_{j=0}^{n} (Gcs_{ij} + Ccs_{ij})}{n} \right) \quad \cdots\cdots\cdots \text{식 (1)}$$

패킷 로그의 분석 시 발생하는 CountScore를 사용하여 시스템 사용 유형을 판단하고 그에 해당 규칙을 추천하기 위한 조건은 식 (2)와 같이 해당 규칙의 CountScore 값이 모든 규칙의 CountScore의 평균값 이상인 경우에 Snort에 규칙을 추천한다. 여기서 모든 규칙의 CountScore의 평균값은 식 (3)과 같다.

$$Rcs_{avg} \leq Rcs \quad \cdots\cdots\cdots \text{식 (2)}$$

$$Rcs_{avg} = \left(\frac{\sum_{i=0}^{n} Rcs_i}{n} \right) \quad \cdots\cdots\cdots \text{식(3)}$$

두 가지 방법은 순차적으로 실행되며 그림 14는 규칙을 추천하는 방법을 나타낸 알고리즘이다.

0. RuleTableSet 데이터베이스의 모든 규칙의 적용시간을 시스템 수행 시간만큼 감소한다.
1. 패킷 로그 분석 모듈로부터 침입으로 판단된 sid 목록을 전달받는다.
2. sid 목록을 이용하여 RuleTableSet 데이터베이스의 sid와 비교한다.
3. sid 목록에 해당하는 RuleTable 클래스의 목록을 저장한다.
4. 각 RuleTable 클래스의 CountScore을 계산한다.
5. RuleTable 클래스의 규칙을 시간 계산법에 의하여 적용 시간을 계산한다.
6. 해당 규칙을 Snort에 적용한다.
7. RuleTable 클래스의 모든 목록이 다 검사될 때까지 4~6번을 반복한다.
8. 모든 RuleTable 클래스의 CountScore 평균을 구한다.
9. sid 목록에 해당하지 않는 RuleTable 클래스에 대하여 CountScore가 평균 이상인지 검사한다.
10. 평균 이상인 경우 해당 규칙을 시간 계산법에 의하여 적용 시간을 계산한다.
11. 해당 규칙을 Snort에 적용한다.
12. 모든 RuleTable 클래스이 검사될 때까지 9~11을 반복한다.
13. 규칙 적용 후 모든 ProtocolNode 클래스, GroupNode 클래스, ContentsNode 클래스의 CountScore를 절반으로 감소시킨다.

그림 14. 규칙 추천 알고리즘
Fig. 14. Rule Recommendation algorithm

3.3.2 규칙 적용 시간 계산법

그림 14의 규칙 추천 알고리즘에 의하여 규칙을 추천하고 적용하기 위하여 식(4)와 같이 규칙 적용 시간을 계산한다. 규칙을 적용할 때 계산된 시간은 RuleTableSet 데이터베이스에 저장된 규칙에 기록하며 기록된 시간은 규칙을 Snort에 적용시키거나 해제하는 용도로 사용된다.

규칙 적용 시간을 계산하는데 필요한 요소는 3가지로 구성된다.

① 식 (1)으로 계산된 개별 규칙의 CountScore
② 규칙의 취약성을 나타내는 classtype 또는 priority 값
③ 남은 적용시간(Remain Time)

식 (4)는 3가지 요소를 이용한 계산식으로서 식 (4)에서 나오는 용어는 표 11에서 설명한다.

$$RT_{\text{new}} = RT_{old} + ((Rcs \times Rps)) \times C \quad \cdots\cdots \text{ 식 (4)}$$

표 11. 식(4)에서의 용어 설명
Table 11. Explanation of terms in eq. 4

용어	내용	비고
RT_{new}	새로 적용되는 시간	· 식 (3)에서 계산한 결과 값 · 0 ≦ RT_{new} ≦ 100
RT_{old}	남은 적용시간	· RuleTable에 기록되어 있는 남은 적용 시간 · 0 ≦ RT_{old} ≦ 100
Rcs	개별 규칙의 CountScore	· 식 (1)으로 계산된 값
Rps	classtype 또는 priority	classtype : very low, low, middle, high 4개의 값으로 구분되는 취약성점수이며 0, 2.5, 5, 10으로 표현 priority : 0~10까지의 Snort에서 제공하는 취약성 점수
C	상수계수	· 시간 단위로 변환을 위한 계수 · 네트워크 서비스 시스템의 로그 양에 따라 결정

제 4 장 시 스 템 검 증

제안한 시스템 검증을 위하여 윈도우 기반의 Snort를 이용 하였고, Snort 제공 규칙 1개와 사용자 정의 규칙 4개를 작성하여 테스트 하였다. Snort의 설정에서 $HOME_NET은 “61.82.252.94”의 IP로 설정하였고 $EXTERNAL_NET은 $HOMENET이외의 모든 IP로 설정하였다.

그림 15는 테스트에 사용되는 Snort제공 규칙이다. 하나의 규칙은 하나의 라인으로 구성되며 구분자로 키워드를 구분하면서 구성되어 있다.

```
 alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP SITE
EXEC attempt"; flow:to_server,established; content:"SITE";
nocase; content:"EXEC"; distance:0; nocase;
pcre:"/^SITE\s+EXEC/smi"; metadata:service ftp;
reference:arachnids,317; reference:bugtraq,2241;
reference:cve,1999-0080; reference:cve,1999-0955;
classtype:bad-unknown; sid:361; rev:17;)
```

그림 15. Snort 규칙의 예
Fig. 15. An Example of Snort rule

그림 16은 그림 15의 Snort 규칙을 입력 받아 Snort 규칙 추천 시스템의 규칙 구조화 모듈을 통하여 구조화한 결과화면이다. 우선 규칙을 header와 option으로 나누고 header를 ProtocolNode 클래스에 저장한다. 그리고 option의 정보는 불필요한 키워드는 삭제하고 필요한 부분을 추출하여 pcre, content, flow, uricontents 키워드들의 정보를 저장하고 그에 따른 속성 정보를 ContentsNode 클래스와 GroupNode 클래스에 저장한다. 저장한 클래스의 정보 중 ID정보를 읽어 SRD를 구성한다. 그리고 원본은 RuleTable에 저장하여 관

리한다. Snort 2.9.X버전의 규칙 10874개 중 9349개의 규칙을 구조화하여 저장하였다. 제안한 구조화 방법은 content, uricontent, pcre, flow 키워드를 사용하여 패킷 로그의 데이터 부분을 비교하는 규칙을 구조화 하였다. 패킷의 데이터와 직접적으로 비교하지 않는 키워드를 이용하는 규칙의 경우 구조화에 오류가 발생하는 것으로 판단된다.

```
testrule.rules

입력 받은 Rule

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP SITE EXEC attempt"; flow:t
o_server,established; content:"SITE"; nocase; content:"EXEC"; distance:0; nocase
; pcre:"/^SITE\s+EXEC/smi"; metadata:service ftp; reference:arachnids,317; refer
ence:bugtraq,2241; reference:cve,1999-0080; reference:cve,1999-0955; classtype:b
ad-unknown; sid:361; rev:17;)

header
alert tcp $EXTERNAL_NET any -> $HOME_NET 21

option
msg:"FTP SITE EXEC attempt"; flow:to_server,established; content:"SITE"; nocase;
 content:"EXEC"; distance:0; nocase; pcre:"/^SITE\s+EXEC/smi"; metadata:service
ftp; reference:arachnids,317; reference:bugtraq,2241; reference:cve,1999-0080; r
eference:cve,1999-0955; classtype:bad-unknown; sid:361; rev:17;

P1 : alert tcp $EXTERNAL_NET any -> $HOME_NET 21
G1 : to , C1 : flow : to_server,established
G2 : SI , C2 : content : SITE , properties : nocase
G3 : EX , C3 : content : EXEC , properties : distance : 0 , nocase
G2 : SI , C4 : pcre : /^SITEs+EXEC/smi
R1 : P1 G1 C1 G2 C2 G3 C3 G2 C4 : 361
```

그림 16 . 그림 15 규칙의 구조화 결과
Fig. 16. Result of structuring Fig. 15

이와 같은 구조화 과정을 통하여 만들어진 규칙들의 정보는 각각의 파일로 관리하고 그림 17, 그림 18, 그림 19는 테스트를 위한 4개의 규칙을 관리하는 파일의 나타낸다. 그림 17은 5개의 규칙 정보를 입력받아 키워드들을 구분하여 DataNode 데이터베이스에 저장한 예이다.

```
PID P1 : alert tcp $EXTERNAL_NET any <-> HOME_NET 21
PID P2 : alert tcp $EXTERNAL_NET any <-> HOME_NET 80
PID P3 : alert tcp HOME_NET any ->  $EXTERNAL_NET 80
GID : to
GID : si
GID : ex
GID : et
GID : ge
GID : ht
CID C1 : flow : to_server,established
CID C2 : content : SITE , properties : nocase
CID C3 : content : EXEC , properties : distance : 0 , nocase
CID C4 : pcre : /^SITE\s+EXEC/smi
CID C5 : content : /etc
CID C6 : content : get, properties :offset : 0
CID C7 : content : http, properties :offset : 0
CID C8 : content : sign, properties : offset :0  , nocase
```

그림 17. DataNode 데이터베이스 관리 파일의 예

Fig. 17. Example of DataNode database management file

그림 18은 SRDset를 관리하고 있는 파일의 모습이다. 그림 17의 과정에서 읽어온 구조화한 자료의에서 sid와 키워드의 ID로 규칙을 구성하여 저장한다.

```
sid:361 P1 to C1 si C2 ex C3 si C4
sid:10001 P1 et C5
sid:10002 P2 ge C6
sid:10003 P2 ht C7
sid:10004 P3 si C8
```

그림 18. SRDset 데이터베이스 관리 파일의 예

Fig. 18. Example of SRDset database management file

그림 19는 RuleTableSet 데이터베이스를 관리하는 파일의 모습이다. 앞의 숫자는 현재 남은 적용시간을 의미하는 것으로 적용되지 않는 경우 0으로 설정한다. 그 뒤에 실제 규칙의 정보가 저장되며 마지막으로 규칙의 고유번호인 sid가 표기된다.

```
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 21 (msg:"FTP SITE EXEC attempt";
flow:to_server,established; content:"SITE";nocase;content:"EXEC"; distance:0; nocase; pcre:"/^SITE\s
+EXEC/smi"; metadata: service ftp; reference:arachnids,317; classtype:bad-unknown; sid:361; rev:17;)
sid:361
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 21 (msg:"test1 rule"; content:"/etc"; classtype:string-
detect; sid:10001; rev:6;) sid:10001
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 80 (msg:"test2 rule"; content:"GET"; classtype:system-
call-detect; sid:10002; rev:4;) sid:10002
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 80 (msg:"test3 rule"; content:"HTTP"; classtype:system-
call-detect; sid:10003; rev:6;) sid:10003
0, alert tcp $HMOE_NET any -> $EXTERNAL_NET 80 (msg:"test4 rule"; content:"sign"; priority:1; sid:10004;
rev:6;) sid:10004
```

그림 19. RuleTableSet 데이터베이스 관리 파일의 예

Fig. 19. Example of RuleTableSet database management file

5시간째에 들어온 패킷 로그의 정보를 분석한 결과 총 2209번의 명령어 단위의 로그가 남았고 규칙에 해당하는 패킷 로그의 정보는 없었다. 그림 20은 5시간째에 들어온 패킷 로그 중 일부분이다. 하지만 pid:P1이 2209번 검사하게 되고 sid:361, sid:10001의 CountScore의 값이 0.2209가 되어 평균 CountScore인 0.0863이상의 값을 가지게된다. 따라서 sid:361, sid:10001 규칙이 Snort에 적용되고 RuleTableSet에 시간을 저장한다. sid:361의 적용시간은 CountScore값에 classtype이 bad-unknown인 medium(5.0)의 값을 곱하여 1.1045이고 sid:10001의 적용시간은 classtype이 string-detect의 값인 low(2.5)를 CountScore에 곱하여 0.5522가 된다.

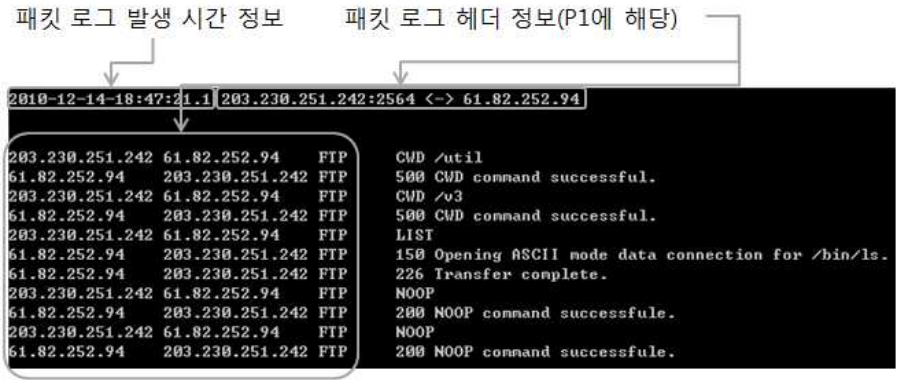

그림 20. 5시간 경과 후 들어온 패킷 로그 중 일부

Fig. 20. Parts of packet log after 5 hours

그림 21는 그림 20의 로그 분석 후 시간 계산 결과를 적용하여 sid:361, sid:10001의 시간이 각각 1.1045와 0.5522로 기록된 RuleTableSet 데이터베이스 관리 파일의 모습이다.

```
1.1045, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 21 (msg:"FTP SITE EXEC attempt";
flow:to_server,established; content:"SITE";nocase;content:"EXEC"; distance:0; nocase; pcre:"/^SITE\s
+EXEC/smi"; metadata: service ftp; reference:arachnids,317; classtype:bad-unknown; sid:361; rev:17;)
sid:361
0.5522, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 21 (msg:"test1 rule"; content:"/etc";
classtype:string-detect; sid:10001; rev:6;) sid:10001
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 80 (msg:"test2 rule"; content:"GET"; classtype:system-
call-detect; sid:10002; rev:4;) sid:10002
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 80 (msg:"test3 rule"; content:"HTTP"; classtype:system-
call-detect; sid:10003; rev:6;) sid:10003
0, alert tcp $HMOE_NET any -> $EXTERNAL_NET 80 (msg:"test4 rule"; content:"sign"; priority:1; sid:10004;
rev:6;) sid:10004
```

그림 21. 5시간 경과 후 RuleTableSet 데이터베이스 관리 파일
Fig. 21. RuleTableSet database management file after 5 hours

10시간째에 들어온 패킷 로그의 정보를 분석한 결과 총 1127번의 명령어 단위의 로그가 남았고 sid:10001 규칙에 해당하는 패킷 로그의 정보가 3번 발생하였다. 그림 22는 10시간째에 들어온 패킷 로그 중 일부분이다. 그림 22와 같은 패킷 로그의 내용일 경우 pid:P1을 검사하게 되고 탐지될 가능성이 있는 규칙 데이터는 sid:361, sid:10001이 검색되므로 sid:361,sid:10001을 이용하여 검사를 한다. 검사를 마치면 총 3번의 /etc를 찾았다는 결과가 나오고 pid:P1은 총 1127번을 탐색하였기 때문에 남아있던 0.1104의 값에 0.1127의 값을 더하여 0.2231이란 값이 되고, gid:et은 0.03, cid:C5은 3의 값이 된다. sid:10001의 CountScore 계산법은 3.3절에서 설명한 방법으로 계산한다. Rcs_{10001}의 값은 0.2231 + ((0.03 + 3) / 1) = 3.2531이고 Rcs_{361}의 값은 0.2231이 되며 나머지 규칙은 0의 값을 가진다. Rcs_{10001}의시간 계산은 취약성 점수를 나타내는 classtype이 string-detect의 값인 low(2.5)를 곱하고 상수 계수인 1을 곱하여 7.5777시간이란 결과가 나오고 RuleTable 클래스에 저장한다. Rcs_{361}은 평균 이하의 CountScore이기 때문에 규칙을 적용하지 않는다.

그림 22. 10시간 경과 후 들어온 패킷 로그 중 일부
Fig. 22. Parts of packet log after 10 hours

그림 22는 그림 21의 로그 분석 후 시간 계산 결과를 적용하여 sid:10001의 시간이 8.1327으로 기록된 RuleTableSet 데이터베이스 관리 파일의 모습이다.

```
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 21 (msg:"FTP SITE EXEC attempt";
flow:to_server,established; content:"SITE";nocase;content:"EXEC"; distance:0; nocase; pcre:"/^SITEWs
+EXEC/smi"; metadata: service ftp; reference:arachnids,317; classtype:bad-unknown; sid:361; rev:17;)
sid:361
8.1327, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 21 (msg:"test1 rule"; content:"/etc";
classtype:string-detect; sid:10001; rev:6;) sid:10001
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 80 (msg:"test2 rule"; content:"GET"; classtype:system-
call-detect; sid:10002; rev:4;) sid:10002
0, alert tcp $EXTERNAL_NET any <-> $HMOE_NET 80 (msg:"test3 rule"; content:"HTTP"; classtype:system-
call-detect; sid:10003; rev:6;) sid:10003
0, alert tcp $HMOE_NET any -> $EXTERNAL_NET 80 (msg:"test4 rule"; content:"sign"; priority:1; sid:10004;
rev:6;) sid:10004
```

그림 23. 10시간 경과 후 RuleTableSet 데이터베이스 관리 파일
Fig. 23. RuleTableSet database management file after 10 hours

그림 24는 패킷 로그 분석을 하면서 시간의 경과됨에 따라 규칙이 적용되고 해제되는 모습을 나타낸다.

5시간째에 들어온 패킷 로그의 정보를 분석한 결과 총 2209번의 명령어 단위의 로그가 남았고 규칙에 해당하는 패킷 로그의 정보는 없었다. 10시간째에 들어온 패킷 로그의 정보를 분석한 결과 총 1127번의 명령어 단위의 로그가 남았고 그림 22에서 나타난 sid:10001 규칙에 해당하는 패킷 로그의 정보가 3번 발생하였다. 15시간째에 들어온 패킷 로그의 정보를 분석한 결과 총 302번의 명령어 단위의 로그가 남았고 sid:10002, sid:10003 규칙에 해당하는 패킷 로그의 정보 13번씩 발생하였다. 20시간째에 들어온 패킷 로그는 총 124번의 명령어 단위의 로그가 남았고 sid:10002, sid:10003 규칙에 해당하는 패킷 로그의 정보가 4번씩 발생하였다.

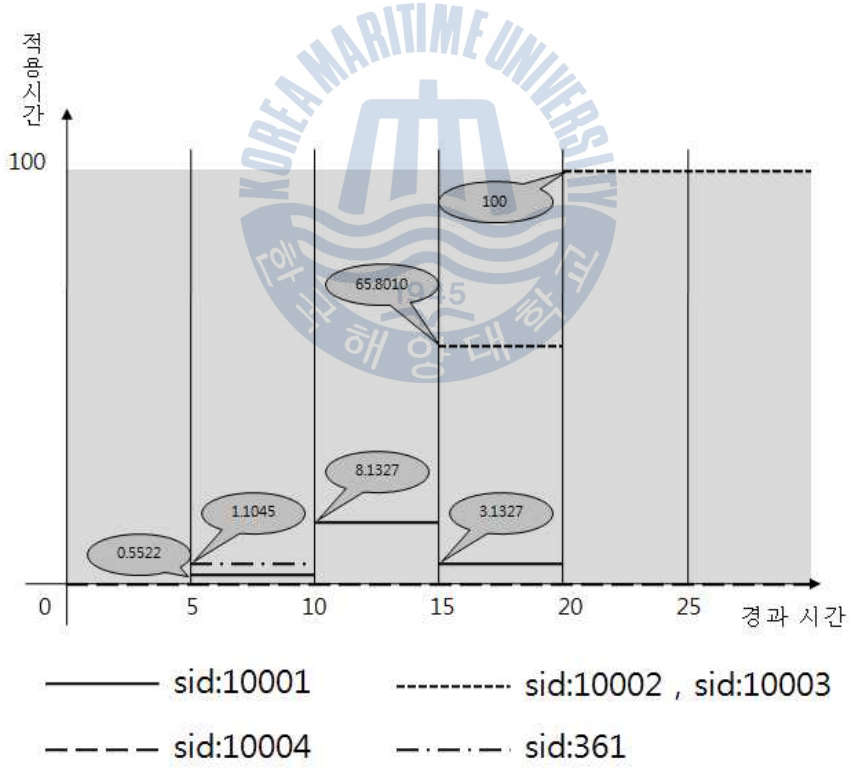

그림 24. 패킷 로그 분석을 통한 규칙 적용시간 변화 예
Fig. 24. Example of rule remain time throught packet log analyze

표 12는 그림 24에서 시간에 흐름에 따라 규칙의 적용시간을 구하기 위한 각 규칙의 CountScore값과 규칙의 평균 CountScore값을 나타낸다.

표 12. 그림 24에서 시간의 흐름에 따른 각 규칙의 CountScore
Table 12. Each rule CountScore throught time in Fig. 24

경과 시간	규칙	규칙의 CountScore	적용 시간	평균 CountScore
5	sid:361	0.2209	1.1045	0.0883
	sid:10001	0.2209	0.5522	
	sid:10002	0	0	
	sid:10003	0	0	
	sid:10004	0	0	
10	sid:361	(0.2209/2) + 0.1127 = 0.2231	0	0.6952
	sid:10001	(0.2209/2) + 3.1427 = 3.2531	8.1327	
	sid:10002	0	0	
	sid:10003	0	0	
	sid:10004	0	0	
15	sid:361	0.2231 / 2 = 0.1115	0	5.6145
	sid:10001	3.2531 / 2 = 1.6265	3.1327	
	sid:10002	13.1602	65.8010	
	sid:10003	13.1602	65.8010	
	sid:10004	0.0142	0	
20	sid:361	0.1115 / 2 = 0.0557	0	4.8294
	sid:10001	1.6265 / 2 = 0.8132	0	
	sid:10002	(13.1602/2) + 4.0524 = 11.6325	100	
	sid:10003	(13.1602/2) + 4.0524 = 11.6325	100	
	sid:10004	0.0142 / 2 + 0.0062 = 0.0133	0	

그림 25와 표 11에서 처음 5시간째에 FTP 로그를 통하여 sid:361과 sid:10001이 직접적으로 패킷 로그에 발생하지는 않았지만 규칙의 CountScore 평균값이상의 CountScore을 가지면서 규칙이 적용되었다. 10시간째에 sid:361과 sid:10001의 경우 시간 경과로 해제가 되었지만 sid:10001은 패킷 로그 분석 후 규칙에 해당하는 정보로 인하여 재적용 되었다. 15시간째에 HTTP 서비스를 시작하면서 새로운 로그를 받고 패킷 로그 분석 후 sid:10002, sid:10003이 규칙에 해당하는 정보로 결정되었다. 따라서 두 규칙을 적용하였고 sid:10004의 경우 ProtocolNode 클래스의 CountScore 증가로 인한 값은 나왔지만 평균값 미만으로 적용되지 않았다. 20시간째 또한 15시간째와 같은 방법으로 적용하고 sid:10002, sid:10003은 남은 시간에 계산한 적용한 시간을 더하여 값이을 저장되어 관리함을 알 수 있다.

제 5장 결론 및 향후 과제

이 논문에서는 로그 분석을 통하여 네트워크 서비스 시스템의 성향을 판단하고 이에 맞는 규칙을 추천하여 침입 탐지 시스템에 적용 및 해제하는 기법을 제시하였다. 이를 위하여 대표적인 침입 탐지 시스템인 Snort의 규칙을 이용하여 구조화한 형태로 저장하였고 이를 이용하여 Snort에서 생성된 로그를 분석함으로써 네트워크 서비스 시스템의 사용 성향을 파악함으로써 적절한 규칙을 추천하도록 하였다.

제안하는 기법을 적용함으로써 분석 결과로부터 네트워크 서비스 시스템에서의 침입 유형과 침입 발생 빈도, 사용 유형 등의 요소를 이용하여 보안 전문가의 지식이 없이도 상황에 맞는 규칙을 추천하고, 적은 개수의 규칙을 적용하여 침입 탐지 시 시스템 자원을 효율적으로 사용할 수 있다.

이 논문에서는 Snort 2.9.X버전의 규칙 10874개 중 9349개의 규칙을 구조화하여 패킷 로그의 분석이 가능하지만 향후 모든 Snort 규칙이 적용 가능한 자료구조의 연구가 필요하다. 또한 실험을 통하여 탐색 시 발생하는 CountScore에 관한 정확한 수치를 증가시키는 개선이 필요하다.

참고 문헌

[1] 한국인터넷진흥원, 2009년 인터넷 이용실태조사 최종보고서, 2009

[2] 케이티하이텔(주), 2006년 하반기 CRM 1단계 개발 제안 요청서, 2006

[3] R. Drum, IDS and IPS Placement for Network Protection, Centre for Sensor Signal and Information Processing, 2006

[4] M. Roesch, et al., Snort Users Manual Snort Release: 2.8.5, 2009

[5] G. Vigna, et al., "A Stateful Intrusion Detection System for World-Wide Web server", Proceedings of the Anual Computer Security Applications Conference, pp. 34-45, 2003

[6] Snort web site, http://www.snort.org

[7] Bros Intrution Detection System web site, http://bro-ids.org

[8] Bleeding Edge Threats web site, http://bleedingthreats.net

[9] PCRE - Perl Compatible Regular Expressions web site, http://www.pcre.org

[10] B.C. Brodie, et al., "A scalable architecture for high-throughtput regular-expression pattern matching", Computer Architecture News, Vol. 34, No. 2, pp. 191-202, 2006

[11] 임철수 역, TCP/IP 인터넷 네트워킹, 도서출판 그린, 2003