웹서버를 위한, 서비스 거부 공격에 강한 침입탐지시스템 구성*

김 윤 정**

요 약

오용 탐지(misuse-detection) 기반의 침입 탐지 시스템에서 패턴 비교 부분은 많은 수행시간 및 자원을 필요로 하며, 전체 시스템 성능 상의 병목 현상을 유발한다. 더 나아가 이것은 서비스-거부-공격의 표적이 될 여지도 있다. 본 논문에서는 첫째, 서비스 거부 공격에 강한 침입탐지시스템 구성 방안을 살펴보고, 둘째, 특별히 웹서버 시스템에서 효율적으로 탐지를 수행하는 방안을 제안한다. 이 두 가지 방안을 함께 이용함으로써 웹서버 시스템을 서비스 거부 공격을 포함한 침입에 대항하여 효율적으로 안전하게 유지할 수 있다.

Framework Architecture of Intrusion Detection System against Denial-of-Service Attack, especially for Web Server System

Yoonjeong Kim**

ABSTRACT

The pattern matching part of Intrusion Detection System based on misuse-detection mechanism needs much processing time and resources, and it has become a bottleneck in system performance. Moreover, it derives denial-of-service attack. In this paper, we propose (1) framework architecture that is strong against denial-of-service attack and (2) efficient pattern matching method especially for web server system. By using both of these 2 methods, we can maintain web server system efficiently secure against attacks including denial-of-service.

Key words : Intrusion Detection System, Pattern Matching, Rule File

---

* 이 논문은 2005년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2005-204-D00047).
논문의 기반연구를 함께 진행한 서울여대 정보보호학 전공 학생들과 강성헌 님, 그리고 익명심사위원께 감사드립니다

** 서울여자대학교 정보보호학과

1. 서 론

침입 탐지 시스템(Intrusion Detection System)은 안전한 전산망 환경을 위하여 꼭 필요한 도구 중의 하나이다. 침입탐지시스템은 탐지방법에 따라 크게 오용 탐지(misuse detection) 방법과 이상 탐지(anomaly detection) 방법으로 나눌 수 있다. 오용 탐지 방법에서는 침입에 해당하는 사용자들의 실행을 signature로 추출하고 네트워크 패킷에 이 signature가 존재하는지 여부를 검사함으로써 침입을 판단한다. 이 과정에서 수행되는 signature 비교 방법은 '패턴 탐지' 작업을 필요로 하며, 이것은 침입탐지시스템의 성능 중 많은 부분을 차지하는 과도한 작업으로 침입탐지시스템의 병목현상이 되고 있다. 따라서, 침입탐지시스템에 있어서의 패턴 탐지 방법을 개선하고자 하는 많은 연구들이 있어왔다[1-9]. 여기에는, Boyer-Moore 방법과 Aho-Corassic BM 방법을 이용한 검사방법에 대하여 성능분석을 수행한 것[5], 하나의 단일 알고리즘으로 성능을 개선하는 것이 아니라, 여러 개의 알고리즘을 혼합하여 사용하여 성능을 개선하는 방법에 대한 것[6], 높은 성능을 위한 전용의 하드웨어를 제작하는 것[7], 탐지 규칙을 2-계층으로 구성하여 진행하는 것 등이다[2, 8].

침입탐지시스템의 패턴매칭 오버헤드는 서비스 거부 공격을 발생할 여지도 있다[10]. 이것은 실제 패턴 비교를 수행하는 경우, 패턴탐지규칙들이 균등히 분포하지 않고 어느 한 곳에 몰려있는 경우에 발생할 수 있다.

한편, 요즈음의 컴퓨터 시스템들은 특정한 기능을 위하여 동작하는데, 웹서버의 경우가 특히 그러하다. 일부 회사나 단체에서는 해당 회사의 홈페이지를 구축하고 홈페이지 등과 관련한 웹서버 기능을 한 대의 서버에서 처리하도록 하고, 웹서버 기능 이외의 기능들은 이 서버에서는 수행하지 않는다.

이러한 현실에 비추어, 본 논문에서는, 특별히 웹서버 시스템을 대상으로, 서비스-거부-공격에 대하여 안전하며 효율적으로 탐지를 수행하는 방안을 제안한다. 제안하는 방안은 첫째, 시스템 구성 방안에 대한 것으로, 웹서버 기능만을 웹서버에서 수행하도록 설정하고 웹서비스 및 기본 기능 이외의 기능은 웹서버에서 제거하고 방화벽 등의 기타 정보보호시스템의 도움을 받아 차단하는 것이다. 둘째로, 웹서버 기능만을 수행함에 있어서도 침입을 탐지함에 있어서 현재보다 더 효율적으로 탐지를 수행하는 방안으로서 패턴 비교 트리를 균등하게 구성하도록 하는 것이다.

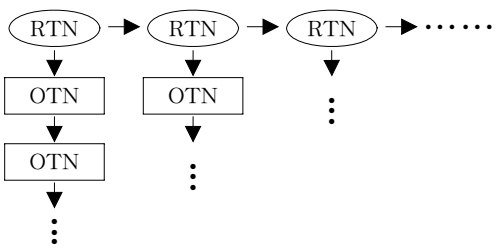
본 논문의 구성은 다음과 같다. 먼저 제 2장에서는 연구의 배경으로서, 기존 연구들을 기반으로 침입탐지시스템의 패턴 탐지 구성 형식을 살펴보고, 다음으로 패턴 탐지 기능의 성능 분포율을 조사한다. 또한 탐지 방법이 서비스-거부-공격의 표적이 될 수 있음을 살펴본다. 제 3장에서는, 웹서버 시스템을 대상으로 서비스-거부-공격에 강하도록 시스템을 구성하는 방안을 살펴본다. 제 4장에서는, 웹서버의 웹 관련한 기능을 탐지함에 있어서 패턴 탐지를 효율적으로 수행할 수 있는 방안을 살핀다. 이것은 현재 존재하는 공개 침입탐지시스템인 snort의 규칙파일 및 수행에 대한 조사를 기반으로 한다. 제 5장에서 성능분석을 기술하고 제 6장에서 결론 및 향후 연구내용을 기술한다.

2. 연구 배경

2.1 패턴 탐지 구성 형식

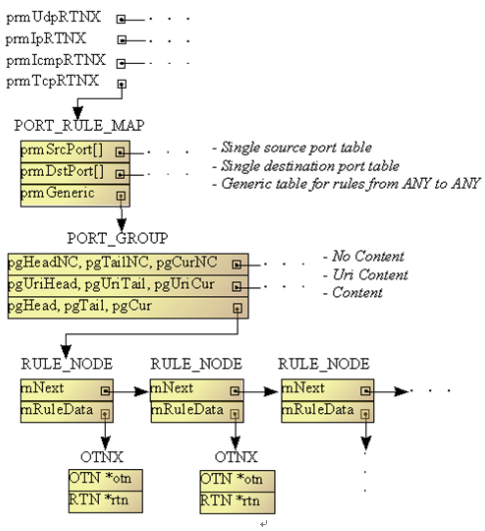
이전 버전의 Snort에서는 (그림 1)과 같이 규칙을 단지 rule tree node와 option tree node의 2차원 rule matrix로 유지하였다. 여기에서 option tree에는 탐지해야 하는 패턴 내용이 들어가게 된다.

최근 버전의 snort에서는 (그림 2)와 같이 이 rule matrix를 포트 그룹에 따라 나누어 분류한다.



(그림 1) Rule Matrix

이것은 패킷이 도착하면 port 번호를 확인하고 포트 번호가 같은 패턴들만을 탐색함으로써 빠른 패턴 탐지를 수행하기 위함이다[11-13].



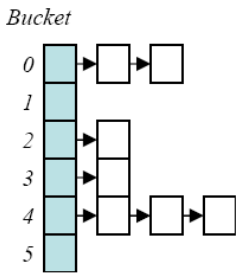
(그림 2) 빠른 패킷 탐지를 위한 자료 구조

2.2 패턴 탐지 기능의 성능 비율

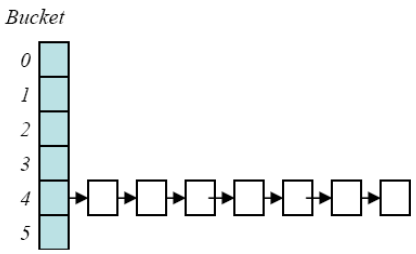
침입탐지시스템에서 어느 부분이 성능의 병목 현상이 되겠는가에 대한 의문을 해결하기 위하여 많은 연구들이 있어 왔다[1-9]. 이것은 이론이나 실험적으로 수행되었으며, 이 때 수행대상으로는 공개침입탐지시스템인 Bro나 Snort가 많이 이용되어 왔다[11-14]. [9]에서는 공개침입탐지시스템인 snort를 대상으로 실제적인 수행성능을 조사한 바 있다. 이들은 snort 소스를 gprof[15]를 이용하여 컴파일하고, DARPA 침입 탐지 평가의 1998년과 1999년의 2-주 실험 데이터를 이용하여 CPU 시간 소비를 측정하였다[16]. 이들의 조사에 의하면, 문자 패턴 비교가 시간과 자원을 가장 많이 소비하는 작업이며 snort는 자신의 전체 CPU 시간의 70%를 이를 수행하는 탐지엔진에 사용하고 있음이 알려졌다.

2.3 패턴 탐지 기능의 서비스-거부-공격 가능성

[10]에서는 해쉬함수를 알고 있는 공격자가 최악의 경우를 예측하고 이로부터 서비스-거부-공격을 수행할 우려가 있음을 지적하였다. (그림 3) (a)와 같이 해쉬 테이블이 정상적으로 구성되는 경우에는 괜찮으나 (그림 3) (b)와 같이 최악의 경우로 해쉬 테이블이 구성된다면 이를 구성하고 탐색하는데 많은 시간이 허비될 수 있음이다. 공격자는 최악의 경우로 해쉬 테이블이 구성되도록 입



(a) 정상적인 경우의 해쉬 테이블 구성



(b) 최악의 경우의 해쉬 테이블 구성

(그림 3) 해쉬 테이블 구성 예

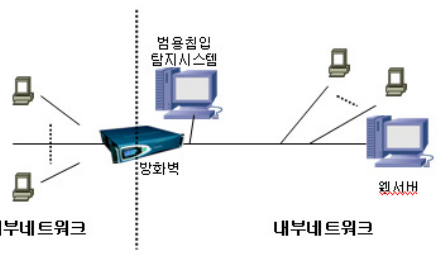
력을 주고 이를 악용하여 침입을 수행할 수 있다.

[10]에서 발표한, 침입탐지시스템에 해쉬함수 구성을 이용한 공격은 공개침입탐지시스템인 Bro[14]를 대상으로 한 것이다. Bro는 port scan detect를 하는 과정에서 얼마나 많은 목적지 포트를 살피는지를 해쉬 테이블을 구성하여 추적하는데, 이를 이용하는 공격이다.

한편, 일반적인 침입탐지시스템에서 패턴 비교 방법은 규칙들을 그룹화하고 들어오는 패킷에 대하여 이 그룹 중 일부 그룹 내에서만 패턴 비교를 수행한다. 이 때, 비교해야 할 패턴의 개수가 일부 그룹에 과도하게 많이 할당되어 있다면, 해쉬 테이블의 예와 마찬가지로 이를 악용한 서비스-거부-공격 등과 같은 공격이 있을 수 있다. 본 논문에서 강화하고자 하는 서비스 거부 공격에 대한 안전성은 이 패턴 탐지 구성 부분에 대한 것이다.

3. 서비스-거부-공격에 강한 시스템 구성 방안

웹서버의 서비스-거부-공격에 대응하기 위한 제안하는 시스템 구성 방안은 (그림 4)와 같다. 일단 외부 네트워크와 내부 네트워크를 분리하고 웹서버와 별도로 방화벽과 범용 침입 탐지 시스템을 두도록 한다. 방화벽을 이용하여 내부 네트워크로 오가는 위험한 패킷을 차단하도록 하고, 범용침입탐지시스템을 이용하여 위험한 패킷은 감시하도록 한다. 그리고, 웹서버에는 웹서비스 기능과 서버 운영에 필요한 기본적인 네크워크 서비스 만이 유지되도록 하고, 그 외 다른 서비스들은 비활성화시켜 놓는다. 이와 같은 지침을 정리한 내용은 (그림 5)와 같다.



(그림 4) 서비스-거부-공격에 강한 시스템 구성 방안(구성도)

1. 외부 네트워크과 내부 네트워크를 분리하고 중간에 방화벽을 설치하여 오가는 위험한 패킷을 차단하도록 하며, 범용침입탐지시스템으로 하여금 위험한 패킷을 감시하도록 한다.
2. 웹서버에는 웹서비스 기능과 서버 운영에 필요한 기본적인 서비스를 제외한 다른 모든 서비스는 비활성화 시켜 놓는다.

(그림 5) 서비스-거부-공격에 강한 시스템 구성방안(지침)

4. 웹서버에 대한 탐지모듈의 효율적 구성 방안

4.1 Snort의 웹서버 탐지 패턴 분석

웹서버의 탐지모듈을 효율적으로 구성하기 위하여, 우선 공개용 침입 탐지 시스템인 snort의 탐지 규칙을 분석하였다. 분석파일은 2006년 10월에 입수한 것으로 총 8210개의 규칙으로 구성되어 있다. 이 규칙들은 여러개의 파일에 나누어 저장되어 있다. <표 1>에 파일이름의 첫문자를 해당 문자로 갖는 파일에 포함된 규칙의 개수가 나타나 있다. n으로 시작하는 파일들에 3107개의 규칙들이 있으며, g, h, j, k, l, q, u, y, z로 시작하는 파일은 존재하지 않거나 규칙이 포함되어 있지 않다. 규칙파일 세기는 <부록 1>과 같이 규칙의 개수를 세는 명령을 이용하였다.

이 중 w로 시작하는 파일들인 web-attacks, web-cgi, web-client, web-misc 등이 웹서버에

〈표 1〉 SNORT 규칙파일에 포함된 규칙 갯수

파일이름의 첫자	규칙갯수
a*	17
b*	594
c*	34
d*	771
e*	150
f*	91
i*	184
m*	94
n*	3107
o*	309
p*	137
r*	142
s*	836
t*	32
v*	6
w*	1704
x*	2
합 계	8,210

대한 것으로 여기에는 1704개의 규칙이 포함되어 있다.

이 규칙을 이용하여 snort 2.8.1에서 수행하게 되면 1704개의 규칙으로부터 1557 개의 옵션 체인이 만들어지며 이 옵션 체인들은 27개의 체인 헤더에 나누어 연결되게 된다. 이러한 분석은 <부록 2>와 같이 웹 관련 규칙만을 snort에 주고 실행시에 snort에서 출력하는 메시지를 확인하여 얻었다.

4.2 웹서버 패턴 탐지 모듈의 성능 개선 방안

웹서버 패턴 탐지 기능에서 가장 중요한 요소는 1557 개의 패턴들이 균등하게 분포할 수 있도록 구성하는 것이다. 이렇게 함으로써 특정 패킷이 왔을 때 최악의 경우의 패턴 탐색 시간이 발생하는 것을 방지할 수 있다.

5. 성능 분석

침입탐지시스템의 성능 측정은, 전체적인 수행 시간을 시뮬레이션이나 실제 측정을 통하여 할 수도 있고, 또는 침입탐지의 병목이 되는 패턴 비교 횟수를 세는 방법도 있다. 본 논문에서는 비교해야하는 또는 패턴 탐지를 수행해야 하는 규칙의 개수를 대상으로 성능 분석을 수행하였다. 이 방법이 실제 수행 시간에 대한 정보를 주는 것은 아니지만, 제안하는 방법의 기존 방법에 대한 효율성은 충분히 나타낼 수 있다.

또한, 규칙 파일의 개수는, 실제 널리 이용되고 있는 대표적 침입탐지시스템인 snort를 대상으로 하여 현실성을 높였다.

<표 2>에는 웹서버가 기존 snort를 이용하는 경우의 규칙 개수와 방화벽을 이용한 시스템 구성 시의 규칙 개수가 나타나 있다. 방화벽을 이용한 시스템 구성 시에는 기존에 처리해야 하는 규칙의 20.7%에 해당하는 규칙만을 처리하면 된다. 즉, 79.3%의 규칙파일 감소 효과를 얻을 수 있다.

〈표 2〉 웹서버가, 방화벽 이용한 시스템 구성시 처리해야 하는 규칙 개수 및 규칙갯수감소율

현재 snort	방화벽 이용한 시스템 구성 적용	규칙갯수 감소율
8210	1704	79.3 %

다음으로 웹서버 패턴 탐지 모듈을 균등하게 유지함으로써 얻을 수 있는 성능상의 장점을 분석하면 다음과 같다. 제 4.1장에서 밝힌 바와 같이, 현재 snort는 1704 개의 웹 관련 규칙으로부터, 1557 개의 패턴 탐지를 수행하는 옵션 체인을 생성한다. 이 1557개의 옵션 체인들은 현재 27개의 체인 헤더에 나누어 연결된다. 최악의 경우에 어느 한 개의 체인 헤더에 연결될 수 있는 옵션의 개수는 1531개이다. 제 4.2장에서 제안한 바와 같이 체인 헤더에

균등하게 옵션 체인이 연결되도록 한다면, 최상의 경우에 한 개의 체인 헤더에 평균적으로 58개의 옵션 체인이 연결된다. 즉, 균등하게 옵션 체인이 연결되는 방안 이용 시에는 기존에 처리해야 하는 규칙의 평균적으로 3.8%에 해당하는 규칙만을 처리하면 된다. 즉, 최악의 경우와 비교하여 96.2%의 패턴 비교 횟수 감소 효과를 얻을 수 있다.

〈표 3〉 웹서버가, 웹서버 패턴탐지 모듈을 균등하게 유지함으로써 얻게 되는 옵션 체인 개수 및 감소율

최악의 경우	체인 헤더에 균등하게 옵션 체인이 연결되는 방안 적용	패턴 비교 감소율
1531	58	96.2 %

6. 결 론

본 논문에서는 웹 서버를 대상으로 하여, 오용탐지기반의 침입탐지시스템에서 패턴 비교 횟수를 줄일 수 있는 방안에 대한 연구내용을 소개하였다. 제안하는 기법은 2가지로, 하나는 방화벽 및 범용침입탐지시스템과 공조하여 웹서버는 웹 관련 규칙만 다루는 것이며, 다른 하나는 탐지 규칙 행렬에서 한 헤더에 연결되는 옵션체인들의 개수를 균등히 하는 것이다. 이렇게 함으로써, 전체적으로 웹서버 시스템에서 동작하는 침입탐지시스템의 성능을 개선할 수 있고 나아가서 서비스-거부-공격 등에도 더 안정적으로 동작하게 된다.

향후, 옵션체인의 개수를 균등히 하는 방안을 자세히 설계하고 이에 대한 성능분석을 추가하고자 한다.

참 고 문 헌

[1] 박유미, 김윤정, “침입탐지시스템의 탐지모듈 성능개선 방안에 대한 연구”, 정보보증 논문지, 제3권, 제4호, pp. 15-22, 한국사이버테러정보전학회, 2003년 12월.

[2] 김윤정, “2계층 구조의 탐지 규칙을 사용하는 탐지 기법 및 SNOR에의 구현”, 정보기술논문지, 제5권, 제1호, pp. 79-85, 서울여자대학교 컴퓨터과학연구소, 2007년 12월.

[3] 이재국, 김형식, “규칙 적용 성능을 개선하기 위한 다중 패턴매칭 기법”, 정보보호학회논문지, 제18권, 제3호, pp. 79-88, 한국정보보호학회, 2008년 6월.

[4] 박용수, Network Intrusion Detection System 관련 최근 연구 동향, Workshop on Dependable Computing System(WDCS 2008), Institute of Computer Technology, Seoul National Univ., 2008.

[5] C. Jason Coit, Stuart Stainford, and Joseph McAlerney, “Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snort”, Proceedings of the 2nd DARPA Information Survivability Conference and Exposition(DISCEX II), June 2001.

[6] M. Fisk and G. Varghese, “Applying Fast String Matching to Intrusion Detection”, Technical Report in Preparation, Successor to UCSD TR CS2001-0670, University of California, San Diego.

[7] Lin Tan, “A High Throughput String Matching Architecture for Intrusion Detection and Prevention”, Proceedings of the 32nd International Symposium on Computer Architecture(ISCA’05), 2005.

[8] Tzu-Fang Sheu, Nen-Fu Huang, and Hsiao-Ping Lee, “A Novel Hierarchical Matching Algorithm for Intrusion Detection Systems”, Proceedings of IEEE Globecom, 2005.

[9] A. Yoshioka, S. H. Shaikot, and MinSik Kim, “Rule Hashing for Efficient Packet

Classification in Network Intrusion Detection", International Conference on Computer Communications and Networks(ICCCN 2008), IEEE, 2008.

[10] Scott A. Crosby, and Dan S. Wallach, "Denial of Service via Algorithmic Complexity Attacks", USENIX Security Symposium, 2003.

[11] Martin Roesch, "Snort-Lightweight Intrusion Detection for Networks", USENIX LISA Conference, November 1999.

[12] Snort, http://www.snort.org.

[13] Andres Felipe Arboleda and Charles Edward Bedon, Universidad del Cauca-Colombia, SnortTM diagrams for developers, April, 2005.

[14] Vern Paxson, "Bro : A System for Detecting Network Intruders in Real-Time", Computer Networks, Vol. 31 No. 23-24, pp. 2435-2463, 14 Dec. 1999.

[15] S. L. Graham, P. B. Kessler, and M. K. Mckucick, "Gprof : A call graph execution profiler", Proceedings of 1982 SIGPLAN sysmposium on Compiler Construction, 1982.

[16] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA offline intrusion detection evaluation", Computer Networks, Vol. 34, No. 4, pp. 579-595.

김 윤 정

1991년 서울대학교 컴퓨터공하과 (공학사)

1993년 서울대학교 컴퓨터공학과 (공학석사)

2000년 서울대학교 전기·컴퓨터 공학부(공학박사)

2000년～2001년 (주)엔써커뮤니티 제품개발연구소 차장

2001년～2002년 (주)데이타게이트 인터내셔널 보안기술연구소 차장

2002년～현재 서울여자대학교 컴퓨터학부 정보보호학전공 조교수

〈부록 1〉 규칙파일에서 규칙의 개수를 세는 명령

유닉스 시스템의 규칙파일폴더에서 다음을 수행한다. 예를 들어 a로 시작하는 파일에 있는 규칙의 개수는 다음 명령어로 얻을 수 있다.

```
grep sid a*.rules | wc -l
```

여기서 sid는 규칙마다 부여되는 고유번호로 주석 등을 제외한 규칙의 개수를 알게 한다. b와 c 등으로 시작하는 파일들의 개수도 동일한 방법으로 얻을 수 있다.

〈부록 2〉 web 관련 규칙 중 패턴 탐지를 수행하는 옵션 개수 및 헤더 개수 얻기

snort의 규칙파일에 web 관련 파일들만을 포함시키고 snort를 수행하면 다음과 같은 메시지가 얻어진다.

```
++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
1557 Snort rules read
    1557 detection rules
    0 decoder rules
    0 preprocessor rules
1557 Option Chains linked into 27 Chain Headers
0 Dynamic rules
++++++++++++++++++++++++++++++++++++++++++++++++++
```

이것은 web 관련 규칙 1704개 중에서 패턴 비교를 수행해야 하는 옵션 트리가 1557개 만들어졌으며 이들이 27개의 헤더에 나누어 연결되어 있음을 의미한다.