데이터 마이닝을 이용한 서비스 거부 공격 탐지 기법

박호상*, 조은경, 강용혁, 엄영익
성균관대학교 정보통신공학부
e-mail : {hosanna*, iuno, yhkang1, yieom}@ece.skku.ac.kr

Data Mining based Denial of Service Attack Detection Scheme

Ho-sang Park, Eun-kyung Cho, Yong-hyeog Kang, Young Ik Eom
School of Information and Communication Engineering,
Sungkyunkwan University

요 약

DoS (Denial of Service) 공격은 주로 victim 호스트에 대량의 패킷을 보내거나 비정상적인 패킷을 보냄으로써 정상 사용자가 서비스를 이용하지 못하도록 하는 공격을 의미한다. 이러한 DoS 공격을 탐지하기 위해 다양한 기법들이 개발되어 왔으나, 공격의 종류와 방법은 시간이 흐를수록 매우 다양해지고 있어 이를 탐지하는데 한계가 있다. 본 논문에서는 네트워크 패킷의 헤더정보를 감사 자료로 가지고 있는 NIDS (Network-based Intrusion Detection System)에 데이터 마이닝 기법을 적용시켜 이러한 DoS 공격을 탐지할 수 있는 기법을 제안한다. 이 기법을 이용하면 빠르고 자동화된 방법으로 DoS 공격을 탐지할 수 있다. 본 논문에서는 제안 기법을 이용하여 SYN Flooding 공격과 Teardown 공격에 대한 탐지가 가능함을 보인다.

1. 서 론

최근 몇 년 동안 인터넷은 많은 성장을 하였다. 이러한 인터넷의 성장에 따라서 온라인을 이용한 여러 가지 서비스가 나오게 되었으며, 서비스의 범위는 상업, 금융 등을 비롯하여 군사 시설에 이르기까지 다양해졌다. 이것은 네트워크를 통하여 중요한 정보가 이동하고 있고, 저장되고 있음을 의미한다.

이러한 네트워크상에서 DoS 공격을 하는 공격자는 TCP/IP의 취약한 부분을 공격하거나 서비스를 제공하는 시스템의 자원을 소모시켜 정상적인 사용자가 서비스를 받지 못하게 한다. 이때의 victim은 호스트, 서버, 라우터 혹은 네트워크에 연결이 되어 있는 computing entity가 될 수 있다.

DoS 공격 뿐 아니라 네트워크에서 발생하는 다양한 공격을 탐지하기 위하여 호스트 기반 침입탐지 기법과 네트워크 기반 침입탐지 기법이 알려져 있다. 호스트 기반 침입탐지 기법은 단일 시스템에서 사용자의 행동 양상, 사건 등을 조사하는 반면, 네트워크 기반 침입탐지 기법은 시스템을 통과하는 패킷의 정보를 조사하여 공격을 탐지한다. 특히 데이터 마이닝 기법을 사용한 네트워크 기반의 침입탐지 시스템 (Network-based Intrusion Detection System, NIDS)들은 기존의 공격에 대해서 뿐 아니라 알려지지 않은 공격에 대해서도 빠르고 자동화된 방법으로 공격을 탐지하는 방법을 제공할 수 있다[1][2]. 따라서 본 논문에서는 NIDS에 기반을 두고 데이터 마이닝을 사용하여 DoS 공격을 탐지하는 방법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 DoS 공격에 대한 관련연구를 소개하고, 3장에서는 이들 공격에 대한 탐지 방법을 설명한다. 마지막 4장에서는 결론 및 향후 연구과제에 대해서 기술한다.

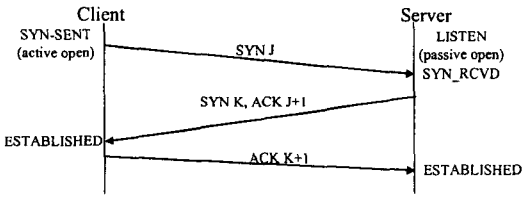
2. 관련 연구

※ 본 연구는 대학 IT연구센터 육성지원사업의 연구결과로 수행되었음

DoS 공격은 주로 victim에 대량의 패킷을 보내거나 정상적이지 않은 패킷을 보내는 공격이 있다. 전자의 경우는 SYN Flooding, UDPstorm, Looping 공격 등이 있고, 후자의 경우는 Smurf, Teardown 등이 있다. 본 절에서는 이중 SYN Flooding 공격과 Teardown 공격을 다룬다.

2.1 SYN Flooding 공격

SYN Flooding 공격은 TCP connection의 three-way handshaking을 이용한 공격이다. TCP connection을 성립시키기 위해서는 두 개의 시스템은 서로의 sequence number를 교환하는 패킷을 주고받는데 이것이 three-way handshaking이다.



(그림 1) TCP/IP의 three-way handshaking

그림 1은 three-way handshaking을 나타내고 있다. 클라이언트로부터 SYN J 패킷이 서버에 도착하면 서버는 SYN K, ACK J+1 패킷을 클라이언트에게 보낸다. 만약 이 두 번째 패킷이 클라이언트에 도착하지 않으면 ACK K+1 패킷이 서버에 전송되지 못하게 되므로 서버는 일정시간동안 대기 상태에 있게 된다. 이런 대기 상태가 계속된다면 시스템 자원의 낭비를 유발한다. SYN Flooding 공격은 이런 점을 악용한 공격 방법이다. 공격자가 대량의 스푸핑된 SYN 패킷을

victim에 보내게 되면 victim은 SYN, ACK 패킷을 보낸 후, 세 번째 ACK 패킷을 받지 못하게 되어 대기 상태에 있게 된다. 이때 victim은 대량의 SYN 패킷에 대해 ACK 패킷을 받지 못하기 때문에 backlog 큐를 모두 소모하게 되며, 결과적으로 어떠한 새로운 TCP 연결 요청에 대해서도 응답할 수 없게 된다[3][4].

2.2 Teardown 공격

패킷의 fragmentation은 어디서나 발생 가능하다. TCP 연결을 위해 three-way handshaking을 하는 과정에서 두 시스템은 MSS(Maximum Segment Size) 정보를 교환한다. 하지만 두 시스템을 연결하는 네트워크 중간에 MSS보다 더 작은 사이즈의 경로가 존재한다면 fragmentation은 발생한다. Teardown 공격은 이런 fragmentation된 패킷을 이용한 공격이다. 한번 fragmentation이 된 패킷들은 목적지 시스템에서 다시 합쳐진다. 하지만 이때 패킷이 분실된 경우에는 패킷 전체를 버리고 다시 요청을 하게 된다. 그러나 패킷을 다시 모을 때, offset 정보가 맞지 않는다면 시스템은 shutdown되게 된다. 공격자는 *victim*에 *offset*을 변경한 패킷을 보낸다. 이때 새로운 패킷의 offset이 기존의 offset에 포함되는 경우에는 시스템은 shutdown되고 새로운 패킷의 offset이 기존의 offset보다 크게 되면 시스템은 분실된 패킷으로 간주하고 다시 요청을 하게 된다. 이것은 자원의 낭비를 의미한다.

3. 제안 기법

3.1 SYN Flooding

STN Flooding 공격을 탐지하기 위해서는 그림 2의 헤더 정보를 이용한다.

flag	sequence num.	ack num.	dest. IP

(그림 2) SYN Flooding 공격에 이용되는 헤더 정보

첫 번째 SYN 패킷은 누구나 보낼 수 있는 패킷이다. 서버에 서비스를 요청하는 어느 누구라도 SYN 패킷을 보낼 수 있다. 하지만 세 번째 패킷은 서버로부터 두 번째 패킷을 받은 클라이언트만이 보낼 수 있다. 즉, 공격을 받지 않는 상황에서는 두 번째 패킷에 대한 응답으로 세 번째 패킷이 반드시 있어야한다. 본 논문에서는 두 번째와 세 번째 패킷이 TCP 연결을 이루기 위해서는 쌍을 이루어야 함을 이용한다.

<SYN,ACK>와 <ACK>의 쌍을 구별하기 위해서 ISN(Initial Sequence Number)을 이용한다. ISN은 서버에서 고유하게 선택하는 32비트숫자이다. ISN은 서버가 4 microseconds 마다 1을 증가시키면서 connection 요청이 있으면 그때의 숫자를 선택해서 ISN으로 사용한다. 이때의 값은 4.55시간을 주기로 반복되므로 짧은 시간 동안 사용하게 되는 감사 자료에서는 유일하게 나타난다. NIDS가 설치되어 있는 네트워크로 나가는 패킷에 대해서 SYN flag가 on인지를 검사하고 on이 되어있는 경우에는 ISN을 이용하여 해쉬 값을 생성하여 기록한다. 기록을 하면서 <SYN, ACK>의 개수(t)를 증가시킨다.

이때 사용되는 해쉬 함수는 그림 3과 같다. 이제 네트워크로부터 들어오는 패킷에 대해서 ACK number를 조사한다. ACK number에 1을 뺀 값이 해쉬 테이블에 존재하면 s값을 1 증가시킨다.

이제, t(<SYN, ACK>의 개수)와 s(<ACK>의 개수)를 비교하면, 정상적인 환경에서는 t와 s는 거의 비슷한 값을 나타낸다. 연결을 성립시키기 위해서는 두 번째 패킷을 받은 시스템은 세 번째 패킷을 반드시 보내야 하기 때문이다. 하지만 공격을 받고 있는 경우에는 s는 t에 비해서 매우 작은 값을 나타낸다. 이것은 SYN 패킷으로 연결 요청을 하고서 응답을 하지 않게 하여 victim의 자원을 소모시키고 있음을 의미한다.

```
//   : 기준 시간
// t :  동안 <SYN k, ACK>의 개수
// k : 네트워크를 나가는 패킷의 ISN
// s :  동안 <ACK k+1>의 개수
// ack_number : 네트워크로 들어오는 패킷의 ack number
// threshold :침입 탐지의 근거가 되는 임계값

When packet is outgoing to the network
    if(SYN and ACK flags are on)
        if(k is not in the hash table) {
            insert the k into the hash table;
            t++;
            delete the k in the hash table after 4.55 hours;
        }

When packet is incoming from the network
    ack_number = packet's ack number-1;
    if(ack_number is in the hash table)
        s++;

Every   seconds
    if( (t-s) > threshold ) {
        SYN Flooding attack;
        initialize the t and s are zero;
    }
```

(알고리즘 1) SYN Flooding 공격을 탐지하는 알고리즘

```
#define  ISN_HASH_LOG  11
#define  ISN_HASH_SIZE  (1 << ISN_HASH_LOG)

int isn_hashfunc(unsigned long isn)
{
        unsigned int value;
        int hash;

        value = isn;
        hash = 0;

        do {
                hash ^= value;
        } while ((value >>= ISN_HASH_LOG));

        return hash & (ISN_HASH_SIZE - 1);
}
```

(그림 3) 상태정보를 저장하기 위한 해쉬 함수

3.2 Teardown

Teardown 공격을 탐지하기 위해서 그림 4와 같은 헤더 정보를 이용한다.

identification	src. IP	flag	offset	size

(그림 4) Teardown 공격에 사용되는 헤더 정보

패킷의 fragmentation은 어디에서든지 발생할 수 있다. 또한, fragmentation이 발생하게 되면 패킷의 크기는 모두 동일하다. 이것은 fragmentation이 발생한다면, 한 시스템에서 하나의 패킷에 대해서 fragmentation을 수행하기 때문이다. 이

때의 크기는 size 필드를 통해서 알 수 있다. NIDS가 설치된 시스템을 통과하는 모든 패킷에 대해서 fragmentation이 되었는지를 판단한 후에 fragmentation이 발생하였다면 다음과 같은 리스트를 만든다.

```
<src, identification>--<size, offset>--<size,offset>
  |
<A, 20303>--<1470, 0>-<1470, 1470>-<1470, 2540>-<50, 4010>
  |
<B, 23023>--<1000, 0>-<8, 600>
  |
<C, 31303>--<1470, 0>-<1470, 1470>-<1470, 2000>-<270,2540>
  |
 ...
```

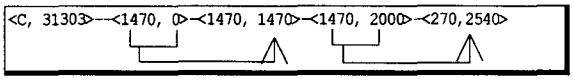
(그림 5)Teardown 공격을 탐지하는데 사용되는 리스트의 예

```
// frag_list : 동안 도착한 fragmentation된 패킷들의 정보가 저장되는 리스트
// item_list : 하나의 fragmentation된 패킷들의 정보(item)가 저장되는 리스트
// size : 현재 item의 size
// offset : 현재 item의 offset
// next_offset : 다음 item의 offset

When fragmented packet is arrived
    if ( <src ip, identification> is not in the frag_list) {
         make the item_list;
         insert item_list into frag_list;
    }
    else {
         update item_list;
    }

Every   seconds
    current_fragment  = frag_list's first element;
    do {
        current_item = current_fragment's first item_list
                       element;
        do {
            current_item's sum = current_item's size +
                                 current_item's offset;
            if (current_item's next_offset ==
                                 current_item's  sum)
                continue;
            else
                Teardown;
        } while(current_item = next item_list element);
    } while(current_fragment = next frag_list element);
    initialize the frag_list;
```

(알고리즘 2) Teardown 공격을 탐지하는 알고리즘

그림 5와 같이 리스트는 identification, source IP, size와 offset으로 이루어져 있다. 한 source IP에서 발생한 패킷은 고유한 identification 값을 가지므로 이 두 값을 이용하여 각각의 fragmentation된 패킷을 구분한다. Δ 동안 시스템에 유입된 fragmentation된 패킷들의 정보는 frag_list를 이용하여 관리되며, 이는 하나의 fragmentation된 패킷의 정보가 저장되어있는 item_list들로 구성된다. 이때 item_list를 구성하는 각각의 요소(item)들은 offset을 이용하여 패킷이 fragment된 순서대로 배치된다. frag_list에 존재하지 않는 <identification, src IP> 정보를 갖는 패킷이 도착한다면 이는 새롭게 fragmentation된 패킷이므로 item_list를 만들어 frag_list에 추가한다.

리스트가 완성이 되면 offset과 size를 비교하여 정상적인 fragmentation인지 아닌지를 구별한다.



(그림 6) Teardown 공격을 받은 경우 생성되는 리스트의 예

예를 들어, 그림 6과 같은 리스트를 생각해 보자. identification은 313030이고 다음 아이템이 <1470,0>이다. 이것은 fragmentation된 패킷의 첫 번째 패킷의 size는 1470이고 offset이 0임을 의미한다. 두 번째 아이템은 <1470, 1470>이다. 이것은 이 패킷의 size는 1470이고 offset은 1470임을 의미한다. 이때 첫 번째 패킷의 size와 offset의 합은 두 번째 항목의 offset과 같아야 한다. 즉, 정상적인 경우에는 fragmentation이 된 패킷의 offset을 비교하면 서로 정확히 맞아야 한다. 그러나 공격을 받고 있는 경우에는 offset 값이 서로 다르게 된다. 그림 6에서 세 번째 패킷과 네 번째 패킷을 보면, 세 번째 패킷의 size와 offset의 합은 3470이다. 이 값은 네 번째 패킷의 offset과 일치해야 한다. 그러나 네 번째 패킷은 2540으로 3470과 일치하지 않는다. 이 경우에는 Teardown 공격을 받은 것이다.

4. 결론 및 향후 연구 과제

DoS 공격을 실시간으로 탐지하기는 어려우며 현재 가장 현실적인 해결책은 가장 빨리 공격을 탐지하는 것이다. 또한 NIDS에서 공격을 탐지하고자 한다면 대량의 패킷이 센서를 통과하는 동안 패킷에 대한 처리 속도가 민감한 문제로 제기된다. 이때 헤더 정보만으로 공격을 탐지하게 되면 data부분까지 조사해야하는 부담이 줄어들기 때문에 패킷의 처리 속도를 줄이면서 공격을 탐지 할 수 있다.

본 논문에서 제시한 방법은 DoS 공격에서 SYN Flooding 공격과 Teardown 공격에 대한 방법을 제시하였다. 앞으로 연구 내용은 제시한 방법에 대해서 시뮬레이션을 통해 검증하도록 하고, 좀 더 빠른 알고리즘을 통하여 실시간으로 공격을 탐지하는 방법을 찾는 것이다.

참고문헌

[1] Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok, "A Data Mining Framework for Building Intrusion Detection Models," Computer Science Department, Columbia University, 1999.

[2] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, and Diego Zamboni, "Analysis of a Denial of Service Attack on TCP," Proceedings IEEE Symposium on Security and Privacy, 1997.

[3] Haining Wang, Danlu Zhang, and Kang G. Shin, "Detecting SYN Flooding Attacks," Proceedings of IEEE INFOCOM, 2002.

[4] Haining Wang, Danlu Zhang, and Kang G. Shin, "SYN-dog: Sniffing SYN Flooding Sources," Proceedings of IEEE ICDCS, 2002.