버퍼를 이용하지 않는 스트링 매칭

곽후근° 정규식
숭실대학교 정보통신전자공학부
{gobarian°, kchung}@q.ssu.ac.kr

String Matching without Buffer

Hukeun Kwak° Kyusik Chung
School of Electronics Engineering, Soongsil University

요 약

전 세계적으로 큰 피해를 주는 웜을 탐지하는 대표적인 방식으로 스트링 매칭을 이용한 방법이 있다. 스트링 매칭은 네트웍상의 패킷을 자신이 가진 시그니쳐(규칙)와 매칭을 해서 웜을 탐지하는 방법으로 동작하는데 매칭시에 필요한 버퍼(메모리)의 사용량으로 인한 단점을 가진다. 즉, 동시에 매칭해야 하는 패킷수가 늘어남에 따라 버퍼(메모리) 사용량도 급격하게 증가하고 버퍼링된 이전 데이터에 대한 매칭으로 속도 지연이 발생하는 단점을 가진다.

이에 본 논문에서는 기존 방식에 비해 적은 메모리 사용량 및 속도 향상을 가지는 버퍼를 이용하지 않는 스트링 매칭 방식을 제안한다. 제안된 방식은 기존 데이터에 대한 매칭 정보만을 기억하고 버퍼링을 하지 않는 방식으로 실험을 통해 기존 방식에 비해 적은 메모리 사용량 및 속도 향상을 가짐을 확인하였다.

1. 서론

인터넷 웜이란 네트웍 상에서 자신을 스스로 복제(Replication)하고 전파(Propagation)할 수 있는 독립된 프로그램으로서 호스트 내의 시스템 자원 혹은 네트웍 대역폭을 소비함으로써 전 세계적으로 큰 피해를 주고 있다.

이러한 웜을 막을 수 있는 방법에는 기존 웜의 유형을 미리 알고 차단하는 침입 탐지 방법(IDS: Intrusion Detection System)과 웜의 유형을 알지 않고 차단하는 침입 차단 방법(IPS: Intrusion Prevention System)이 존재한다. 침입 탐지 방법은 기존의 웜을 식별할 수 있는 시그니쳐(Signature) 정보를 가지고 인터넷 상의 패킷들과 비교하는 방식이고, 침입 차단 방법은 인터넷 상의 패킷들의 통계 정보 등을 이용해서 비 정상 패킷을 사전에 미리 차단하는 방식이다[1].

침입 탐지 방법은 다시 호스트에 기반하여 탐지하는 방식과 네트웍에 기반하여 탐지하는 방식으로 나눌 수 있다. 호스트에 기반하여 탐지하는 방식은 로컬 컴퓨터내의 백신 등을 이용하여 웜을 탐지하는 방식이고, 네트웍에 기반한 방식은 네트웍을 지나가는 패킷을 캡쳐한 후 시그니쳐 정보와 비교하여 웜을 탐지하는 방식이다[2].

스트링 매칭 방식은 네트웍에 기반한 웜 차단 방식에서 캡쳐 패킷과 시그니쳐를 비교하는 데 주로 사용되는 방식이다. 즉, 스트링 매칭 방식은 캡쳐된 패킷 안의 컨텐츠(스트링)와 시그니쳐의 컨텐츠(스트링)를 비교하는 방식으로 동작한다.

본 논문에서는 기존의 버퍼를 이용하는 스트링 매칭 방식이 가지는 문제점을 분석하고 이를 해결할 새로운 버퍼를 이용하지 않는 스트링 매칭 방식을 제안한다. 본 논문의 구성은 다음과 같다. 제 2장에서는 기존의 버퍼를 이용하는 스트링 매칭 방식과 그 문제점을 소개한다. 3장에서는 기존의 버퍼를 이용하는 스트링 매칭 방식의 문제점을 해결하는 새로운 버퍼를 이용하지 않는 스트링 매칭 방식을 설명하고, 4장에서는 실험 및 토론을, 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 연구 배경

2.1 버퍼를 이용하는 스트링 매칭

그림 1과 표 1은 버퍼를 이용한 스트링 매칭은 동작과정을 나타낸다. 매칭되지 않은 패킷을 바로 통과(Accept)시키지 않고 버퍼에 저장하여 이후 패킷과 합쳐 시그니쳐와 매칭하는 이유는 다음과 같다. 시그니쳐에서 매칭하려는 스트링이 하나의 패킷에 모두 들어있을 수도 있지만 두개의 패킷에 나뉘어서 들어오는 경우도 있기 때문이다. 그럼으로 이전 데이터는 항상 버퍼에 저장하고 이후 패킷과 합쳐져서 시그니쳐와 매칭해야 한다[3].

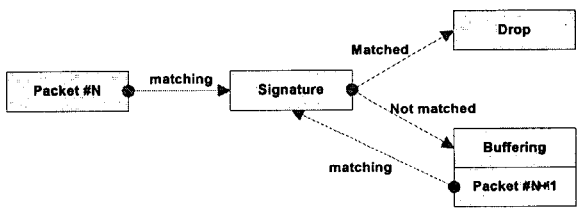

그림 1 버퍼를 이용한 스트링 매칭의 동작 과정

표 1 버퍼를 이용한 스트링 매칭의 동작 과정

단계	동작 과정
1	현재 들어온 패킷이 시그니쳐와 매칭되었다면 패킷을 폐기(Drop) 한다.
2	현재 들어온 패킷이 시그니쳐와 매칭되지 않았다면 이 패킷을 버퍼에 저장한다.
3	이후에 들어온 패킷은 버퍼에 저장된 패킷과 합쳐져서 시그니쳐와 매칭한다.
4	모든 패킷에 대해 1-3단계를 반복한다.

2.2 접근 방식

(1) 버퍼를 이용하는 스트링 매칭의 문제점

버퍼를 이용하는 스트링 매칭의 문제점은 메모리 사용량에 있다. 즉, 버퍼를 할당하기 위해 메모리를 사용하고, 동시 접속 수가 급격히 증가하게 되면 이에 따라 메모리(버퍼) 사용량도 급격히 증가하게 된다. 예를 들어, 하나의 연결(Connection)을 검사하기 위해 2048 바이트의 버퍼가 필요하다면, 이러한 연결이 동시에 10만개 정도 들어오면 메모리 사용량은 204.8 Mbytes가 필요하게 된다. 이것은 필요한 메모리 중 버퍼만을 계산한 것으로 실제로 사용시에는 더 많은 메모리가 필요하게 된다.

또한, 기존 방식은 예전 데이터를 버퍼링하고 이를 현재의 매칭에 다시 이용함으로써 매칭 속도 측면에서도 느리다는 단점을 가진다.

(2) 본 연구의 접근 방식

본 연구에서는 버퍼를 이용한 스트링 매칭의 문제(메모리 소모 및 매칭 속도가 느림)를 해결하는 버퍼를 이용하지 않는 스트링 매칭(메모리를 소모하지 않고 매칭 속도가 빠름) 방식을 제안한다. 제안된 방법은 버퍼를 따로이 할당하지 않고 이전 패킷과 시그니쳐와의 매칭 정보를 가지는 변수를 두어 메모리 사용 및 매칭 속도 지연을 최소로 하도록 하였다.

3. 버퍼를 이용하지 않는 스트링 매칭

그림 2와 표 2는 버퍼를 이용하지 않는 스트링 매칭 방식의 동작 과정을 나타낸다. 제안된 방법은 이전 데이터를 저장하지 않고, 이전 데이터와의 매칭 정보(Offset) 만을 저장하는 것을 제외하고 버퍼를 이용하는 스트링 매칭 방식과 동작 과정이 유사하다. 기존 방식은 동시 연결 수에 따라 메모리 소모 및 매칭 속도 지연이 큰 반면 제안된 방식은 동시 연결 수가 늘어나도 적은 메모리 및 속도 지연으로 스트링 매칭을 할 수 있다는 장점을 가진다.

표 2 버퍼를 이용하지 않는 스트링 매칭

단계	동작 과정
1	현재 들어온 패킷이 시그니쳐와 매칭되었다면 패킷을 폐기(Drop) 한다.
2	현재 들어온 패킷이 시그니쳐와 매칭되지 않았다면 이 패킷을 통과(Accept) 시킨다.
3	현재 들어온 패킷이 시그니쳐와 부분 매칭되었다면 시그니쳐에서 매칭된 부분까지 변수(Offset)에 저장한다.
4	이후에 들어온 패킷은 시그니쳐와 이전 패킷에서의 매칭 정보를 가지는 변수(Offset) 이후부터 매칭한다.
5	모든 패킷에 대해 1-4단계를 반복한다.

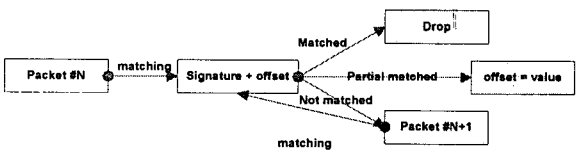

그림 2 버퍼를 이용하지 않는 스트링 매칭

4. 실험 및 토론

4.1 실험 방법

제안된 방법은 Netfilter 모듈에서 어플리케이션(Layer7) 매칭에 사용되는 ipt_layer7 모듈[4]을 수정하여 실험을 수행하였다.

버퍼를 이용한 스트링 매칭 방법은 기존 ipt_layer7 모듈을 그대로 사용하여 실험을 수행하였고, 이때 이전 패킷과 현재 패킷을 버퍼링 하기 위해 2048 바이트의 버퍼(메모리)가 사용됨을 확인하였다.

제안된 버퍼를 이용하지 않는 스트링 매칭 방법은 ipt_layer7 모듈을 수정하여 실험을 수행하였고, 이때 이전 패킷과 현재 패킷을 버퍼링 하지 않고 이전 패킷과의 매칭 정보를 가진 변수만을 두어 매칭을 할 수 있음을 확인하였다. 이때, 사용된 메모리는 이전 패킷과의 매칭 정보를 가진 변수를 위해 4 바이트를 사용하였다.

실험은 메모리 사용량과 속도 측면에서 수행하였다. 메모리 사용량은 1개의 연결에 대한 결과를 N개의 동시 연결에 대한 결과로 확장하였고, 속도는 700 Mbytes의 데이터를 전송할 때 매 패킷에 대한 매칭을 수행한 후의 전송 속도를 비교하였다.

4.2 실험 결과

(1) 메모리 사용량

표 1과 그림 1은 버퍼를 이용하는 스트링 매칭과 이용하지 않는 스트링 매칭의 메모리 사용량을 나타낸다. 버퍼를 이용하는 스트링 매칭은 Netfilter 모듈 중의 어플리케이션 데이터(Layer 7)를 매칭할 수 있는 ipt_layer7 모듈을 사용하였다. ipt_layer7 모듈은 이전데이터와 현재 데이터를 합쳐서 지정된 규칙과 매칭하기 위해 2048 바이트의 버퍼를 사용하고, 이는 동시 접속 수에 따라 (2048 x 동시 접속 수)바이트 만큼 메모리 사용량이 늘어나게 된다.

제안된 버퍼를 이용하지 않는 스트링 매칭은 ipt_layer7 모듈을 수정하여 구현되었으며, 버퍼를 사용하지 않음으로 버퍼에 따른 메모리 사용이 없고, 대신 이전 데이터에서 매칭된 부분을 저장하는 변수에 4 바이트가 할당된다. 즉, 동시 접속 수에 따른 메모리 사용량은 (4 x 동시 접속 수)바이트 만큼 메모리를 사용하게 된다.

표 1 버퍼를 이용하는 스트링 매칭과
이용하지 않는 스트링 매칭의 메모리 사용량 (Bytes)

동시 접속 수	버퍼를 이용하는 스트링 매칭 (2048 x 동시 접속)	버퍼를 이용하지 않는 스트링 매칭 (4 x 동시 접속)
1	2048	4
10	20480	40
100	204800	400
1000	2048000	4000
10000	20480000	40000

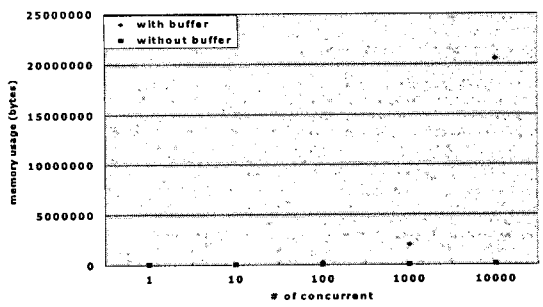

그림 3 버퍼를 이용하는 스트링 매칭과
이용하지 않는 스트링 매칭의 메모리 사용량

(2) 속도

표 2는 버퍼를 이용하는 스트링 매칭과 이용하지 않는 스트링 매칭의 속도를 비교한 것이다. 제안된 방식이 버퍼를 사용한 방식에 비해 속도가 더 빠른 것을 알 수 있다. 이는 버퍼를 사용하는 방식이 기존 데이터를 버퍼링하고 이를 다시 매칭에 사용함으로써 제안된 방식에 비해 느림을 알 수 있다. 버퍼를 사용하지 않는 제안된 방식은 부분 매칭을 검사하는데 약간의 시간이 소비되나 기존 데이터를 버퍼링 하지 않음으로(기존 데이터를 매칭하는데 시간을 소비하지 않음) 기존 방식에 비해 속도가 빠름을 알 수 있다.

표 2 버퍼를 이용하는 스트링 매칭과
이용하지 않는 스트링 매칭의 속도 비교

	버퍼를 이용하는 스트링 매칭	버퍼를 이용하지 않는 스트링 매칭
평균 속도	72.16 Mbps	75.84 Mbps

4.3 토론

제안된 버퍼를 이용하지 않는 스트링 매칭 방식이 기존 방식에 비해 갖는 장점은 이전 패킷을 위해 버퍼(메모리)를 할당하지 않아도 된다는 점에 있다. 버퍼를 이용하는 기존 방식이 동시 접속자수가 늘어남에 따라 기하급수적으로 메모리 사용량 및 매칭 속도 지연이 증가하는 반면 제안된 방식은 최소의 메모리 및 매칭 속도 지연을 사용하여 이전 데이터와 스트링 매칭을 할 수 있도록 하였다.

제안된 방식의 단점은 버퍼를 사용하지 않기 위해 기존 패킷과의 매칭 정보를 갖는 변수를 두어 모든 동시 연결마다 이 변수를 관리해야 하는 부담을 가진다.

5. 결 론

이전 패킷과의 매칭을 위해 버퍼를 이용하는 기존 스트링 매칭의 단점은 동시 연결 수가 늘어남에 따라 버퍼(메모리) 사용량 및 매칭 속도 지연이 급격히 증가하는데 있다. 이에 본 논문에서는 이전 패킷과의 매칭시에도 버퍼(메모리)를 사용하지 않는 방식을 제안하였다. 제안된 방식은 이전 패킷과의 매칭 정보를 담은 하나의 변수만을 두고 버퍼를 사용하지 않는 방식으로 실험을 통해 제안된 방식이 기존 방식에 비해 메모리 사용량 및 속도 지연을 급격히 감소시켰음을 확인하였다.

스트링 매칭과 관련하여 향후 연구 방향을 정리하면 다음과 같다.

● 해싱을 이용한 스트링 매칭 : 규칙의 개수가 늘어남에 따라 매 패킷마다 모든 규칙에 대한 매칭을 수행하면 매칭 속도 지연이 크다는 단점을 가진다. 패킷의 스트링(컨텐츠) 부분에 해싱을 적용하고 이 결과를 매칭에 이용하면 규칙(시그니처) 개수가 늘어나도 매칭 속도 지연을 작게 유지할 수 있다.

참고문헌

[1] IDS and IPS, http://www.kisa.or.kr
[2] Network- vs. Host-based Intrusion Detection System, http://www.documents.iss.net/whitepapers/nvh_ids.pdf
[3] Netfilter, http://www.netfilter.org
[4] Application Layer Packet Classifier for Linux, http://l7-filter.sourceforge.net