# IP Fragment 패킷을 위한 동적 패킷필터링 기법

김영호, 손승원, 박치항

한국전자통신연구원 정보보호연구본부

# Dynamic Packet Filtering for IP Fragments

Young-Ho Kim, Sung-Won Sohn, Chi-Hang Park

Network Security Department, Information Security Division, ETRI

## 요 약

본 논문에서는 IP Fragment 패킷을 이용한 공격에 대해서 소개하고, 현재 사용되고 있는 패킷필터링 방법의 문제점을 극복하기 위한 동적인 패킷필터링 기법을 제안하고 있다. 기존 패킷필터링 방법이 IP 주소 또는 프로토콜 기반의 정적인 규칙에 의한 필터링 방법을 이용하는 반면, 본 논문에서 제안하는 방법은 IP Fragment 패킷에 대해서 단위 시간당 데이터 양으로 표현되는 트래픽에 기반한 패킷필터링 규칙이 동적으로 생성되도록 한다. 이렇게 동적으로 생성된 규칙은 이후 이상 트래픽이 발생되면 자동으로 차단규칙으로 변경되어 IP Fragment 패킷 공격으로부터 네트워크 호스트의 시스템 자원을 보호할 수 있도록 한다.

## I. 서론

현재의 광범위한 네트워크는 다양한 종류의 물리적인 미디어로 서로 연결되어 있다. 이러한 네트워크를 지나는 IP(Internet Protocol) 패킷은 각 미디어를 통과할 수 있는 최대 프레임 크기인 MTU(Maximum Transfer Unit)에 따라 분리(Fragmentation)되었다가 최종 목적지에서 원래의 패킷으로 재결합(Defragmentation)하는 과정을 거치게 된다. 이렇게 패킷이 전송되는 과정에서 분리된 IP Fragment 패킷을 이용한 공격들이 점차 다양한 방법으로 시도되고 있으며 이에 대한 방어 기술이 요구된다. 본 논문에서는 IP Fragment를 이용한 공격과 방어 기술에 대해서 소개하고 특히 IP Fragment 트래픽을 이용한 공격 가능성과 이에 대한 효과적인 방어 기술을 제안하고 있다.

## II. 본문

### 1. IP Fragment 공격 기법

RFC 1858에서는 IP Fragment 패킷을 이용하여 패킷을 차단하기 위해 사용되는 패킷필터링 기법을 공격하는 방법을 Tiny Fragment 공격과 Overlapping Fragment 공격으로 분류하고 있으며 이에 대한 대응 기법을 제안하고 있다.

먼저 Tiny Fragment 공격에 대해서 살펴보면, 모든 인터넷 모듈은 IP 헤더를 제외한 8바이트의 데이터(Payload)를 최소 Fragment 크기를 갖도록 RFC 791에서 명시하고 있다. 그러나 TCP와 같은 프로토콜은 포트 번호와 같은 중요 헤더 정보가 8바이트의 Fragment 패킷 내에 포함된다는 것을 보장하지 못하기 때문에 보안상의 허점이 될 수 있다. Tiny Fragment 공격은 이러한 허점을 이용한 것으로 필터링 규칙 내에 TCP Flag 정보를 이용하여 검사하는 패킷필터를 교묘하게 빠져나가기 위해서 그림 1에서처럼 TCP Flag 정보를 두 번째 Fragment 패킷에 의도적으로 포함시켜서 보낸다. 패킷필터에서는 Fragment된 두 번째 이후의 모든 IP 패킷에 대해서는 검사하지 않고 그대로 통과시키기 때문에 모든 패킷이 그대로 최종 목적지에 도착하게 되고 완전한 하나의 패킷으로 재결합된

다.

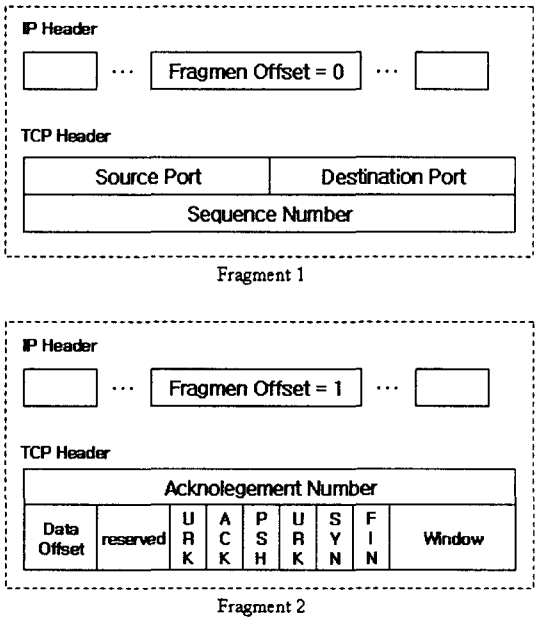


그림 1: Tiny Fragment 공격

또한 RFC 791에 따르면 현재 IP 프로토콜에서는 목적지에 이미 도착된 Fragment와 새로 도착한 Fragment 사이에 오버랩 되는 부분이 있으면 재결합될 때 덮어쓸(overwrite) 수 있음을 기술하고 있다. 이렇게 오버랩 된 Fragment들의 덮어쓰기가 가능하게 되면 공격자들은 패킷 필터를 통과할 수 있는 첫 번째 Fragment 패킷을 만들고 오버랩 될 수 있는 나머지 Fragment 패킷들을 이용해서 최종 목적지에서 새로운 공격 패킷으로 재결합시킬 수 있게 된다. 이러한 허점을 이용한 공격 방법이 Overlapping Fragment 공격이다.

위의 두 공격에 대응하기 위해서 RFC 1858에서 제시한 기법은 Fragment된 첫 번째 IP 패킷의 데이터 크기를 보고 패킷의 폐기 유무를 판단하는 것이다. 예컨대 TCP의 경우 패킷 필터링 검사의 대상이 되는 필드들이 TCP 헤더 부분의 20 바이트 범위 내에 있기 때문에 TCP 데이터를 포함한 첫 번째 IP Fragment 패킷이 최소한 20 바이트의 데이터를 갖고 있지 않다면 그 패킷은 폐기시키게 된다. 이렇게 첫 번째 IP Fragment 패킷이 최종 목적지에 도달하지 못하면 비록 나머지 IP Fragment 패킷이 최종 목적지에 도달하더라도 재결합을 할 수 없기 때문에 이 대응 기법이 공격을 효과적으로 차단할 수 있다. 이러한 내용을 간단한 알고리즘으로 표현하면 다음과 같다.

**if FO=0 and PROTOCOL=TCP and TRANSPORTLEN<TMIN then DROP packet**

첫 번째 IP Fragment(FO=0) 패킷이 TCP 프로토콜인 경우 데이터 크기(TRANSPORTLEN)가 최소한의 TCP 헤더 크기(TMIN)보다 작게 되면 패킷을 폐기(DROP)한다. 간접적인 방법으로 공격을 차단하는 다른 알고리즘은 두 번째 이상의 IP Fragment(FO>0) 패킷이 TCP 프로토콜인 경우 Fragment Offset(FO)의 값이 최소한의 TCP 헤더 크기(TMIN)보다 작으면 패킷을 폐기(DROP)한다. 간접적인 방법을 간단한 알고리즘으로 표현하면 아래와 같다.

**if FO>0 and PROTOCOL=TCP and FO<TMIN then DROP packet**

RFC1858에서 제안하는 IP Fragment 공격에 대한 위의 대응 기법은 IP Fragment 패킷들이 최종 목적지에서 재결합되는 것을 막기 위한 것으로 모든 IP Fragment 패킷을 차단하지는 않는다. 그러므로 첫 번째 IP Fragment 패킷을 제외한 거의 대부분의 IP Fragment 패킷이 그대로 최종 목적지에 도달하는 것은 고려하지 않는다. 그러므로 공격자가 의도적으로 IP Fragment 패킷만을 다량으로 전송할 경우 다량의 패킷이 그대로 최종 목적지로 전달될 수 있게 된다.

## 2. 호스트에서의 IP Fragment 처리

최종 목적지 호스트에서는 분리된 IP 패킷을 원래 하나의 IP 패킷으로 재결합시키기 위해서 도착하는 IP Fragment 패킷들을 버퍼에 저장한 후에 모든 IP Fragment 패킷이 도착하면 재결합을 시작한다. 이 때 IP Fragment 패킷을 임시로 저장하는 버퍼는 제한된 크기의 메모리이기 때문에 많은 양의 IP Fragment 패킷이 도달하면 버퍼 수위를 유지하기 위한 메커니즘을 적용하여 계속해서 유입되는 IP Fragment 패킷을 받게 된다. 예컨대 공개 운영체제인 리눅스에서는 이러한 IP Fragment 버퍼의 조절을 위해서 크게 두 가지 메커니즘을 동시에 적용하고 있다.

IP Fragment 패킷이 처음 도착되면 하나의 큐가 할당되고 모든 IP Fragment 패킷이 도착해서 재결합을 해야하는 한계 시간의 타어머가 설정된다. 만약 설정된 타이머가 종료된 후에도 모든 IP Fragment 패킷이 도달하지 않으면 그 동안 이 큐에 저장되었던 모든 IP Fragment 패킷과 큐에 사용된 메모리는 반환된다. 현재 리눅스에서는 큐가 할당되는 순간에 타이머의 기본 설정 값으로 30초를 할애하고 있다. 또한 proc 파일시스템 형태인

ipfrag_time을 통해 기본 설정 값을 다른 값으로 변경이 가능하도록 구현하고 있다. 만약 ipfrag_time을 짧게 설정하면 IP Fragment 패킷들이 재결합해야 하는 시간이 촉박해지고 대신 보다 많은 큐를 확보할 수 있는 장점이 있다.

위의 Timeout 메커니즘에서 처음 IP Fragment 패킷이 도착하면 새로운 큐를 할당하고 타이머가 종료될 때까지 큐는 메모리에 보존된다. 그러나 타이머가 종료되기 전에 많은 양의 새로운 IP Fragment 패킷이 수신되면 계속해서 큐가 할당되고 나중에는 큐로 할당된 메모리의 제한을 넘어서게 된다. 리눅스에서는 이렇게 큐로 할당된 메모리 이상의 IP Fragment 패킷이 수신될 때에는 가장 오래된 큐부터 강제적으로 반환시키는 LRU 알고리즘을 사용하게 된다. 만약 IP Fragment 패킷을 저장하는데 사용된 메모리(ip_frag_mem)가 제한값(sysctl_ipfrag_high_thresh)을 넘어서면 강제적으로 메모리를 반환시키기 위한 함수인 ip_evictor를 호출하게 된다. 현재 리눅스에서 기본값으로 설정된 메모리 제한값은 256KB이며 ipfrag_time과 마찬가지로 proc 파일시스템을 통해 값의 변경이 가능하다. 현재 큐 중에서 메모리 반환을 위해 희생되는 큐는 LRU에 의해 가장 오래된 큐가 선택되며 메모리가 일정 하한선(sysctl_ipfrag_low_thresh)에 이르게 될 때까지 이 과정을 반복한다. 아래는 ip_evictor 함수의 간단한 코드 구조를 보여주고 있다.

```
ip_evictor() {
  do {
    if(atomic_read(ip_frag_mem) <= sysctl_ipfrag_low_thresh)
      return; // return if ip_frag_mem is less than low threshold
    ipq_kill() ; // find the oldest queue and kill it
  } while(progress);
}
```

리눅스에서는 지금까지 설명된 두 가지 메커니즘을 통해 호스트에서 IP Fragment를 위한 버퍼 사용량을 조절하고 있다. 그러나 짧은 시간에 다량으로 유입되는 IP Fragment 공격에 대해서는 빈번한 ip_evictor 함수만이 실행되기 때문에 다른 IP Fragment 패킷이 재결합 되는 과정을 방해하게 되어 공격에 효과적으로 대응할 수 없다.

## 3. 동적 패킷필터링 기법

기존 패킷필터링[3][4]에서는 패킷 헤더에 기반한 정적인 규칙을 제공하면 경유되는 모든 패킷에 대해서 검사를 하고 패킷의 통과 여부를 결정한다. 앞에서 설명한 Tiny Fragment 공격과 Overlapping Fragment 공격은 이러한 패킷필터를 통과하기 위한 공격들로 이미 많은 상용 패킷필터에서 RFC1858에서 제안한 알고리즘을 적용하여 대응하고 있다. 또한 IP 헤더의 Fragment More 비트를 이용한 정적인 규칙을 설정하면 모든 Fragment 패킷을 통과시키지 않고 거를 수 있다. 그러나 이러한 정적인 규칙을 이용해서 공격 IP Fragment 패킷을 막는 것은 정상적인 비공격성 IP Fragment 패킷까지도 통과하지 못하기 때문에 적용하기 어렵다.

본 논문에서 제안하고자 하는 패킷필터링 기법은 정적인 규칙에 따라 필터링 검사를 수행하지 않고 처음 IP Fragment 패킷이 통과할 때 해당 IP에 대한 목록을 필터링 규칙에 동적으로 추가하고 타이머를 설정한다. 이후에 새로 추가된 규칙에 해당하는 IP Fragment 패킷이 통과하면 패킷은 그대로 통과되고 지금까지 통과된 트래픽 양을 계산하게 된다. 만약 타이머가 종료되기 전까지 이상 트래픽이 발견되지 않으면 해당 규칙은 자동으로 삭제된다. 그러나 추가된 규칙에 해당하는 IP Fragment 패킷의 트래픽이 기준치 이상으로 증가하면 필터링 규칙은 해당 IP 주소로 패킷이 더 이상 통과하지 못하도록 차단 규칙으로 변경된다.

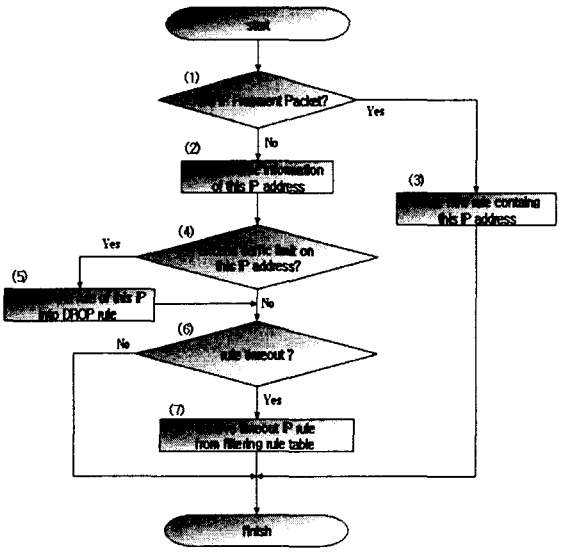


그림 2: 동적 패킷필터링 알고리즘

그림 2는 본 논문에서 제안하는 동적 필터링 기법 과정을 나타내며 IP Fragment 패킷에 대해서만 적용하고 있다. IP Fragment 패킷이 수신되면 단계 (1)에서는 수신된 IP Fragment 패킷의 IP 주소 정보를 이용하여 이전의 규칙들 중에 해당 주소에 대한 항목이 있는지 확인한다. 만약 처음으

로 들어온 IP Fragment 패킷이면 단계 (3)에서 해당 주소에 대한 항목으로 새로운 규칙을 만들고 이후에 들어오는 IP Fragment 패킷에 대한 감시 상태로 들어가게 된다. 단계 (2)에서는 현재 IP Fragment 패킷에 해당하는 항목이 규칙 내에 있으면 현재 IP Fragment 패킷의 크기(byte)를 더해서 트래픽 정보를 갱신한다. 단계 (4)에서는 단계 (3)에서 갱신된 트래픽 정보를 이용하여 이미 설정된 기준치 값과 비교하여 이상 트래픽 유무를 판단하게 된다. 만약 이상 트래픽으로 판단되면 단계 (5)에서 해당 IP Fragment 패킷의 주소에 해당하는 패킷을 모두 차단하도록 필터링 규칙을 변경한다. 변경된 이후에는 목적지 주소가 해당 IP 주소인 IP Fragment 패킷은 모두 차단되어 이상 트래픽이 해당 목적지로 전달되는 것을 막게 된다. 단계 (6)에서는 IP Fragment 패킷에 대한 규칙 중에서 일정 시간 동안 이상 트래픽을 발생하지 않는 항목이 있는지 판단하고 만약 시간이 경과된 항목에 대해서는 단계 (7)에서 해당 항목을 자동으로 삭제하게 된다.

위의 알고리즘에서 트래픽 제한값(traffic limit)은 시간당 패킷 양으로 표시되며 패킷 수가 아닌 실제 통과된 패킷의 총 바이트로 계산된다. 그러므로 이러한 트래픽 제한값은 호스트의 IP Fragment 패킷 처리를 위한 버퍼 크기와 관련이 있기 때문에 유동적으로 변경될 수 있다. 이러한 패킷 양에 의한 동적 패킷필터링 기법은 공격성으로 판단되는 IP Fragment 패킷의 급격한 유입을 미리 차단하여 호스트에서 IP Fragment 패킷을 정상적으로 처리하도록 도와준다.

## III. 결론 및 향후계획

지금까지 IP Fragment 패킷을 이용한 공격기법과 이에 대한 대응기술에 대해서 살펴보았다. 특히 IP Fragment 패킷을 이용하여 호스트 시스템의 자원을 고갈시키는 공격은 기존 패킷 필터링 방법으로는 효과적인 대응이 되지 못하기 때문에 본 논문에서는 IP Fragment 패킷의 트래픽 정보를 이용한 동적 패킷필터링 기법을 제안하였다. 향후 계획은 지금까지 제안한 알고리즘을 바탕으로 실제로 시스템을 구현하고 알고리즘에 대한 성능 분석을 통해 기존 패킷필터링 시스템과 비교하는 것이다. 또한 분석을 통해 현재 알고리즘의 성능을 보다 최적화하기 위한 연구가 필요하다.

## 참고문헌

[1] RFC 1858, "Security Considerations for IP Fragment Filtering".
[2] RFC 791, "Internet Protocol".
[3] Linux 2.4 Packet Filtering HOWTO.
[4] Linux Netfilter Hacking HOWTO.
[5] Steven McCanne, Van Jacobson, "A New Architecture for User-level Packet Capture", 1992.
[6] Bruce Corbridge, Robert Henig, Charles Slater, "Packet Filtering in an IP Router"
[7] Guido van Rooij, "Real Stateful TCP Packet Filtering in IP Filtering"