JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 19, No. 2, pp. 327-334, Feb. 2018

# 블룸필터를 이용한 아웃바운드 트래픽 모니터링 방안 연구

강 성 중 · 김 형 중*
고려대학교 빅데이터응용및보안학과

# Study on Outbound Traffic Monitoring with Bloom Filter

**Seong-Jung Kang · Hyoung-Joong Kim***
Department of Big Data Application and Security, Korea University

**[요 약]**

PC가 악성코드에 감염되면 C&C서버와 통신하며 공격자의 명령에 따라 내부 네트워크에 확산, 정보획득 등의 과정을 거쳐 최종적인 악성행위를 하게 된다. 기업은 외부로부터의 공격을 사전에 차단하는데 중점을 두고 있으나 APT공격을 목적으로 한 악성코드는 어떤 형대로든 내부로 유입된다. 이때 피해의 확산을 방지하기 위하여 악성코드에 감염되어 C&C서버와 통신을 시도하는 PC를 찾아내는 내부 모니터링이 필요하다. 본 논문에서 수많은 패킷들의 목적지IP가 블랙리스트 IP인지 여부를 빠르고 효과적으로 대조하기 위한 블룸필터를 이용한 목적지 IP 모니터링 방안을 제시한다.

**[Abstract]**

When a PC is infected with a malicious code, it communicates with the control and command (C&C) server and, by the attacker's instructions, spreads to the internal network and acquires information. The company focuses on preventing attacks from the outside in advance, but malicious codes aiming at APT attacks are infiltrated into the inside somehow. In order to prevent the spread of the damage, it is necessary to perform internal monitoring to detect a PC that is infected with malicious code and attempts to communicate with the C&C server. In this paper, a destination IP monitoring method is proposed in this paper using Bloom filter to quickly and effectively check whether the destination IP of many packets is in the blacklist.

**색인어 :** 내부관제, APT공격, 블룸필터, 블랙리스트, 패킷모니터링

**Key word :** Internal Monitoring, APT Threat, Bloom filter, Blacklist, Packet Monitoring

**http://dx.doi.org/10.9728/dcs.2018.19.2.327**



**Received** 21 November 2017;**Revised** 17 December 2017
**Accepted** 26 February 2018

***Corresponding Author; Hyoung-Joong Kim**

**Tel:** +82-2-4390-4895
**E-mail:** khj-@korea.ac.kr

## Ⅰ. 서 론

기업들은 중요 정보를 지키고 정보자산이 외부로부터의 각종 침해사고 시도에 대하여 효과적으로 방어하고자 각종 보안시스템을 도입하여 운영하고 있다. 그러나 최근 랜섬웨어(Ransomware) 등 보다 정교한 악성코드를 이용한 "지능형지속공격"(APT, Advanced Persistent Threat)이 빈번하게 발생하여 이를 탐지하고 차단하기 위한 보안시스템까지 등장하고 있다[1]. 그러나 내부 네트워크 인프라는 네트워크 관문, 서버, PC로 고정된 방역구조를 가지고 있고, 이러한 구조는 차단 Zone을 한정시킴으로서 이미 침입한 내부 네트워크에 유통된 악성코드를 차단하지 못한다[2].

공격자들의 기술이 발전하고 각종 어플리케이션의 취약점이 항상 존재함을 고려하면 악성코드는 언제든지 감염될 수 있다고 가정하고 이미 악성코드에 감염된 PC를 탐지하여 확산을 막는 활동을 추가적으로 실시해야 한다[3].

악성코드에 감염된 PC는 평소와 다른 프로세스가 구동되고[4] 악성코드를 제어하는 C&C(Command and Control) 서버를 목적지 IP로의 통신을 시도하거나, 이미 알려진 블랙리스트 IP로의 통신을 시도한다. 이때 사전에 확보한 블랙리스트, 화이트리스트 목록을 검색하여 차단하는 내부 관제를 수행해야 하지만, 검색해야하는 리스트의 크기가 커질수록 성능에 영향을 미친다. 이에 따라, 보다 빠른 리스트 검색을 위하여 블룸필터(Bloom filter)를 이용하여 검색하는 목적지 IP가 블랙리스트 및 화이트리스트에 속하는지 여부를 빠르게 판단할 수 있는 방법을 제시한다.

## Ⅱ. 선행연구 및 관련연구

### 2-1 악성코드 활동 메커니즘

APT란 과거 단순하게 하드디스크를 파괴한다거나, 특정 파일을 유출한다거나 하는 악성코드와 달리 외부 C&C 서버를 통하여 잠행적이고 지속적으로 특정 목표를 달성하기까지 시스템 내의 취약점을 공격하거나 네트워크 내의 다른 시스템을 경유로 목표시스템을 공격하는 등 고급화(advanced)되고 진화된 해킹 기법이다.

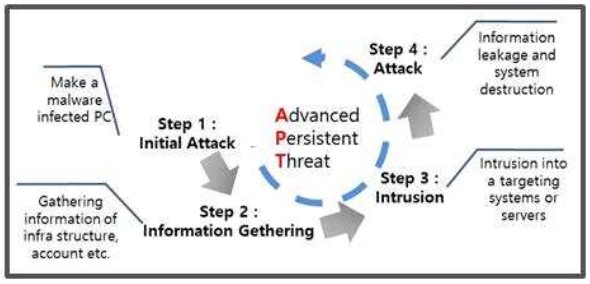


**그림 1.** APT공격 단계
**Fig. 1.** APT attack process

안티바이러스 보안시스템들이 시그니처 탐지를 기반으로 작동하는 기본적인 동작 방식을 우회하여 안티바이러스에 탐지되지 않도록 몇 비트만 바꾸어 유포하여 정상적인 프로그램처럼 보이게 한다.

APT공격은 다음과 같은 4단계로 감염되고 활동한다[5].

- 1단계: APT공격은 메일, 웹사이트, 외부의 업데이트 서버 등을 이용하여 악성코드 감염 PC를 만든다.
- 2단계: 감염된 PC를 이용하여 조직 내부 정보를 수집하거나, 악성코드를 내부 네트워크로 확산한다.
- 3단계: 계정정보, 접근권한 등을 탈취하여 목표로 하는 시스템에 침입한다.
- 4단계: 공격자의 명령에 따라 내부 중요 정보 유출 및 시스템 파괴 또는 자신의 흔적을 지운다.

APT공격은 사전에 1단계에서 예방하거나 탐지하기 힘들다고 가정하고, 2단계부터 4단계 사이에 C&C서버가 명령을 내리고 감염된 PC는 그 명령을 수행 하면서 발생하는 C&C서버와 감염 PC의 통신 여부를 모니터링하면 감염 PC를 찾아낼 수 있다.

### 2-2 APT공격의 주요 사례

2011년 농협 해킹 사건은 APT에 감염된 PC를 통해 목표 대상 시스템을 점령하고 악성행동을 하여 서비스 마비를 야기한 대표적인 사례이다. 외주업체 직원의 서버관리용 노트북이 APT에 이용되는 악성코드에 감염되고, 외주 직원은 해당 노트북을 통해 서버에 접속하여 업무를 수행하였다. 악성코드는 서버에 확산되고, 이로 인해 서버 550여 대의 하드디스크가 손상을 입었으며 재해복구용 서버도 파괴되었으며 최소 80억 원 이상의 피해가 발생하였다[6].

2016년 인터파크의 개인정보 유출 사건의 경우 한 직원이 지인으로 위장한 공격자의 메일을 받아 첨부파일을 실행시켜 본인의 PC가 감염되고, 해커는 악성코드를 이용하여 DB에 접근 가능한 PC를 찾아 개인정보 파일을 유출하였다. 인터파크는 ISMS인증 등 각종 정보보호 관련 인증을 득하였고, 망분리 등 내부적으로 보안 각종 보안시스템을 구축하였으나 이와 같은 사고가 발생하였다[7].

두 사고사례에서 얻을 수 있는 교훈은 조직 내부와 외부는 어떤 경로로든 접점이 존재하며, 이를 통하여 APT 악성코드는 조직 내로 유입될 수밖에 없다. 그러나 두 사고사례 모두 감염된 PC가 평소와 다른 행동을 보이는 것을 모니터링 하고 그 경로를 차단하였다면 농협의 경우는 공격자의 명령이 악성코드에 전달되지 않았을 것이며, 인터파크의 경우 탈취한 개인정보파일을 공격자에게 유출되지 않았을 것이다. APT가 성공하려면 공격자가 의도한 최종 단계까지 실행되어야 하는데 그 최종 목적을 위한 최종명령이 전

달되지 않는다면 공격자의 공격은 결국 실패했을 것이다.

### 2-3 블룸필터

블룸필터[8]란 어떤 원소가 집합에 속하는지 여부를 검사하는데 사용하는 확률적 자료구조로 1970년 Burton Howard Bloom에 의해 고안되었다. 블룸필터는 사전에 특정 그룹의 원소들의 해시 값을 계산한 후 각 해시 값에 대응하는 비트를 1로 설정해 놓고, 검색하는 원소의 해시 값과 비교하여 모든 비트가 1인 경우 검색하는 값은 그룹에 속한다고 판단하는 통계적 특성의 자료구조이다.

어떤 원소가 특정 그룹에 속하는지 검색하기 위하여 통상 그룹 내의 모든 원소를 검색하여 검색하는 내용이 포함되어 있는지를 판별한다. 그러나 검색 대상 그룹이 매우 크고 검색해야 할 원소가 다수 존재할 경우 탐색해야할 자료의 범위가 넓어지므로 검색 속도가 매우 느리게 된다. 이런 검색 방법은 빠른 응답속도를 요구하는 시스템에서 사용하기에는 적합하지 않다. 10,000개의 원소를 가진 DB에서 10개의 원소 중 실제 DB에 존재하는 값이 5개라고 가정하면, 일반적인 검색은 10개의 원소 각각 10,000번의 대조작업을 수행 한 후 5개의 결과 값을 찾아내지만, 블룸필터를 이용하면 검색하는 10개의 원소에 대하여 블룸필터에서 5개의 원소를 먼저 거른 후 실제 존재하는 5개의 원소에 대해서만 DB에서 검색할 수 있다. 즉, 데이터가 실제 존재하느냐의 여부를 먼저 판단 한 후 검색을 진행함으로써 불필요한 검색에 자원을 소비하지 않는 장점이 있다.

가령 블룸필터를 이용하여 x라는 원소가 그룹{a, b, c}에 속하는지를 확인하기 위해서, 먼저 그룹의 원소 a, b, c에 대하여 각각 3개의 해시함수를 이용하여 얻은 출력 값에 해당하는 배열의 인덱스를 1로 수정한다. 이후 x에 대하여 동일한 3개의 해시함수를 이용하여 얻은 출력 값을 기존 {a, b, c}에 대한 해시 결과 인덱스와 비교하여 모두 1이면 x는 {a, b, c}에 포함되고, 그렇지 않으면 x는 {a, b, c}에 속하지 않는다고 판단한다. 이때 그룹에 속하지 않는다고 판단한 원소에 대하여는 검색작업을 수행하지 않아도 되므로 처음부터 모든 원소에 대하여 전수 검색을 하는 것보다 검색의 범위를 줄일 수 있는 장점이 있다.

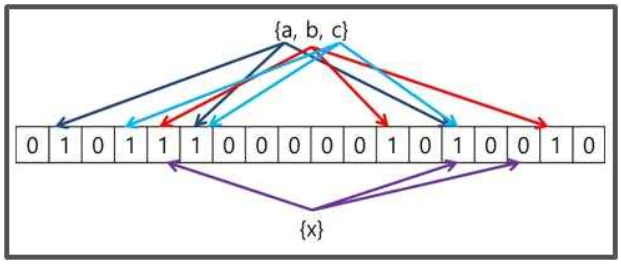


**그림 2.** 블룸필터의 구조
**Fig. 2.** Structure of Bloom filter

그러나 블룸필터는 실제 DB에 존재하지 않지만 검색하는 값이 DB에 존재하는 값이라는 결과를 줄 수 있는 오탐(false positive)가능성이 있다. 즉, 블룸필터의 결과는 "확실하게 해당 원소가 그룹에 속한다."가 아닌 "아마도 원소가 그룹에 속할 것이다."를 의미한다. 블룸필터가 이런 false positive 결과를 만들 확률은 블룸필터의 원소의 개수 $n$, 블룸필터의 비트 배열의 크기 $m$, 블룸필터의 해시함수의 개수 $k$라 할 때 대략적으로 다음과 같다[9].

$$(1-e^{-\frac{kn}{m}})^k \quad (1)$$

1보다 작은 수의 $k$의 제곱으로 표현되는 확률로 해시함수의 개수 $k$가 증가하면 오탐이 일어날 확률이 줄어들기는 하나 다수의 해시함수를 수행하는 연산 비용까지 고려하면 적절한 $k$ 값을 선택하는 것이 중요하다.

### 2-4 블룸필터를 이용한 패킷 모니터링

블룸필터를 이용한 패킷 모니터링에 관한 연구는 주로 외부에서 내부로의 통신에 대한 DDoS(분산서비스거부)공격이나 스캐닝을 탐지하는 것에 초점을 맞춘다.

출발지 IP, 도착지 IP, 도착지 포트번호를 가지고 각각의 해시테이블을 만들어 놓고, 만약 입력받는 패킷이 DDoS공격 패킷이라면 해시테이블은 계속해서 '1'이 나타나게 된다. 이때 해시테이블의 생명주기를 주어 특정 기간 동안 '1'이 입력되지 않으면 '0'으로 리셋하고, 동일한 해시테이블에 계속해서 '1'이라면 외부로부터 공격이 일어나고 있을 가능성이 있다고 판단한다[10][11].

이 경우 패킷 모니터링을 위한 블룸필터는 3개 이상의 factor에 대한 해시테이블이 필요하다. 또한 생명주기를 너무 길게 잡으면 정상적으로 서비스를 사용하는 사용자가 공격행위를 하는 것으로, 생명주기를 너무 짧게 잡으면 실제 공격행위를 정상행위로 오탐지할 가능성이 있다.

본 논문에서 제시하는 모델은 기본적으로 블룸필터를 이용하여 트래픽을 모니터링 하여 공격을 감지한다는 아이디어는 동일하지만 탐지하는 패킷의 방향성과 해시테이블의 크기에 있어 차이점을 둔다. 내부에서 발생하는 패킷의 도착지 IP가 블랙리스트에 포함된 IP인지 여부만 확인하기 때문에 도착지 IP 1개의 factor에 대한 해시테이블만 가지고 있으면 되므로 블룸필터의 해시테이블을 위한 공간도 보다 적게 필요하다. 또한 해시테이블의 생명주기도 필요하지 않으므로 알고리즘이 간단하고 구현이 쉽다.

## Ⅲ. 연구 모형 및 결과

### 3-1 연구모형

본 연구는 블룸필터를 이용하여 입력받는 목적지 IP를 사전에 정의된 블랙리스트와 화이트리스트 DB에서 찾아 블랙리스트로 통신하는 패킷은 차단하고 그렇지 않는 경우 정상적인 서비스를 제공하는 모형을 제안한다. 이때, 블룸필터가 가지고 있는 오탐지에 의한 차단의 경우를 최소화하기 위하여 블랙리스트 필터에서 블랙리스트로 판정된 IP 들을 다시 화이트리스트 필터로 걸러내는 방안을 제시하며 그 개념도는 <그림 3>과 같다.

먼저 블랙리스트 필터에서 블랙리스트가 아니라고 판정한 목적지 IP는 인터넷으로 포워딩 한다. 그리고 블랙리스트 필터가 참으로 판정된 IP는 즉시 차단하지 않는다. 위에서 언급한대로 블룸필터가 가지고 있는 오탐지 오류로 인해 실제 블랙리스트에 없지만 블랙리스트라고 판정하는 경우가 있기 때문에 블랙리스트라고 판정된 IP는 화이트리스트 필터로 보낸다. 화이트리스트 필터에서도 존재하지 않는 목적지 IP는 최종적으로 블랙리스트로 간주하고 통신을 차단한다. 블랙리스트만 검사하여 결과가 참인 경우 즉시 차단하여도 내부에서 블랙리스트로 접근하는 트래픽에 대한 모니터링이 충분히 가능하지만 블룸필터의 오탐지를 감안하여 최대한 정확한 판정을 위하여 화이트리스트까지 점검한다. 이때 내부 관제를 위하여 단순히 목적지 IP를 블랙리스트와 화이트리스트에서 검색하여 탐지하는 방법과 블룸필터를 이용하여 검색하는 방법의 연산시간을 비교하여 성능개선 효과를 확인한다.

### 3-2 데이터 수집

본 실험에 사용한 블랙리스트 데이터는 한국인터넷진흥원과 금융보안원에서 제공하는 블랙리스트 IP 24,269건을 이용하였으며. 화이트리스트는 보험연수원의 침임차단시스템(F/W)에서 운영하는 화이트리스트 IP 890건을 이용하였다. 화이트리스트는 업무상 접속이 필요한, 상대방이 확인된 호스트 IP를 대상으로 했으며, OS 업데이트, 안티바이러스의 업데이트를 위한 CDN서버들은 제외하였다.

PC가 접속 시도하는 목적지 IP에 대하여는 보험연수원 업무용 PC 50대에서 2017년 9월 1일부터 2017년 9월 30일까지 1개월 동안 수집한 패킷의 목적지 IP 49,589개의 정보를 이용한다. 해당 기간 아웃바운드 트래픽 수집대상 PC들에서 외부로 세션 요청된 전체 건수는 약 1,600만 건으로 본 연구의 목적은 아웃바운드 트래픽 중 블랙리스트를 목적지로 하는 패킷을 거르는데 있기 때문에 목적지 IP는 1,600만여 건 중 중복을 제거한 고유한 IP주소 49,589개만 이용한다.

### 3-3 실험 결과 및 분석

실험을 위한 프로그래밍은 Python의 bloomfilter library를 이용하였고 <그림 4>와 같이 구현하였다.

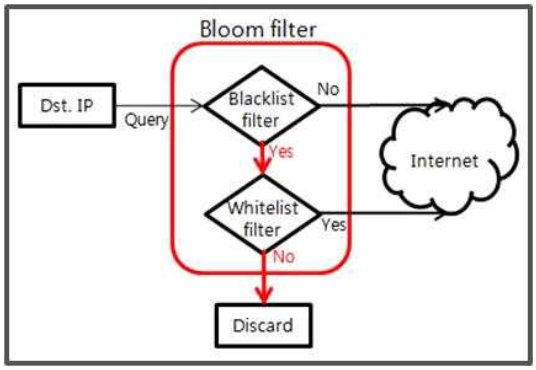


**그림 3.** 블룸필터를 이용한 목적지 IP 필터링
**Fig. 3.** Destination IP filtering with Bloom filter

블룸필터의 에러율은 0.0001로 설정하였고 이에 따른 비트 배열 m의 크기는 479,253개로 설정된다. 블룸필터를 이용한 경우와 이용하지 않는 경우 모두 위에서 제시한 모형에 바탕을 두어 입력받는 목적지 IP를 블랙리스트에 먼저 검사한 후 블랙리스트로 판정한 IP에 대하여 화이트리스트에서 검사하는 순서로 구동된다.

실험은 블랙리스트의 개수, 화이트리스트의 개수 및 검색하는 목적지 IP의 개수를 변화시키면서 각각의 경우에 대한 검색시간을 비교하는 것으로 성능을 측정한다. 블랙리스트와 화이트리스트 및 목적지 IP의 수는 랜덤으로 추출하여 실험한다. 또한 일반적인 검색과 블룸필터를 이용한 검색 결과를 대조하여 정확도를 측정한다.

```
def black_filter(black_search_ip):
    black = black_search_ip in black_IP
    if black:
        print("1st Decision:", black_search_ip.rstrip("\n"), "Black")
        white_filter(black_search_ip)
    else:
        print("1st Decision", black_search_ip.rstrip("\n"), "White")
        f = open('result_bloom_white1.csv', 'a')
        f.write(black_search_ip)


def white_filter(white_search_ip):
    white = white_search_ip in white_IP
    if white:
        print("Final Decision:", white_search_ip.rstrip("\n"), "White")
        f = open('result_bloom_white2.csv', 'a')
        f.write(white_search_ip)
    else:
        print("Final Decision", white_search_ip.rstrip("\n"), "Black")
        f = open('result_bloom_black.csv', 'a')
        f.write(white_search_ip)


start_time = time.clock()

for search_ip in search_List:
    black_filter(search_ip)

print("---%s seconds ---" % (time.clock() - start_time))
```

**그림 4.** 블룸필터를 이용한 블랙리스트 IP 탐색 프로그래밍
**Fig. 4.** Black list IP distinction program with Bloom filter

### 1) 블랙리스트, 화이트리스트, 검색대상 IP를 모두 변화시키는 실험 결과

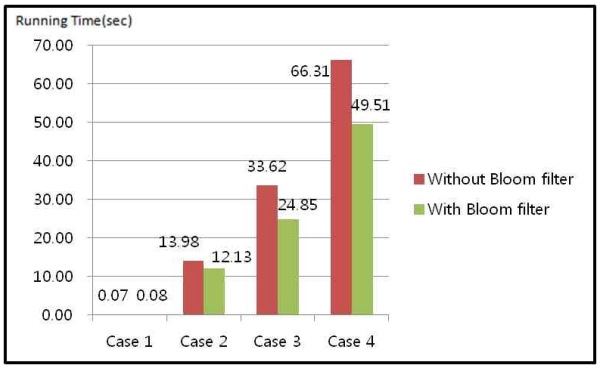


**그림 5.** 실행시간 비교
**Fig. 5.** Running time comparison

실험은 블랙리스트, 화이트리스트, 검색대상 IP 각각의 원소 집합 중 100개씩 뽑아내는 경우(Case 1), 약 1/4씩을 뽑아내는 경우(Case 2), 약 1/2씩을 뽑아내는 경우(Case 3), 전체를 대상으로 검색하는 경우(Case 4)를 테스트 한다. 시스템에서 처리하는 검색 리스트와 입력받는 목적지 IP의 크기에 따른 성능 변화를 확인하기 위한 실험이다.

<그림 5>의 결과에서 볼 수 있듯이 블룸필터를 사용하지 않은 경우와 사용한 경우 모두 블랙리스트 개수, 화이트리스트 개수, 검색해야 하는 목적지 IP의 수가 증가와 비례하게 검색 시간이 소요된다. 다만 블룸필터를 사용한 경우가 블룸필터를 사용하지 않은 경우보다 검색하는데 보다 짧은 시간이 소요됨을 알 수 있다. 최초 100개씩의 샘플을 가지고 실험한 결과의 차이가 없다고 볼 수 있지만 그 외의 샘플 단위 실험에서는 블룸필터를 사용하는 검색방법이 그렇지 않은 경우 대비 약 25%정도 실행시간이 적게 소요되었다.

### 2) 검색대상 IP만 변화시키는 실험 결과

다음은 시스템의 블랙리스트의 개수와 화이트리스트가 고정되어 있고 검색하는 입력 값만 계속 늘어나는 가정의 실험한다. 블랙리스트 개수와 화이트리스트 개수는 본 연구에 사용되는 데이터 전부인 블랙리스트 24,269개, 화이트리스트 890개 모두 사용하며, 검색대상 IP만 12,000개, 24,000개, 36,000개, 48,000개로 12,000단위로 변화시키며 실험하였고, 그 결과는 <그림 6>과 같다.

본 연구에서 쓰이는 샘플이 약 49,000개 정도이나 샘플의 수가 더 커질 경우 블룸필터를 이용했을 때와 이용하지 않았을 때 결과를 받을 수 있는 시간의 차이가 더 커진다.

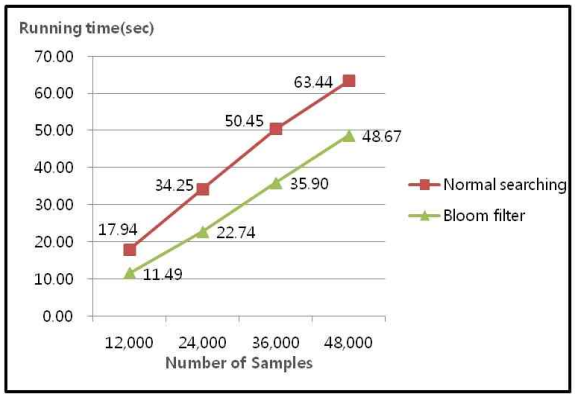


**그림 6.** 탐색 대상 샘플 개수에 따른 탐색시간 비교
**Fig. 6.** Running times depending on sample size

### 3) 탐색 결과의 정확성

블룸필터를 이용하여 검출한 블랙리스트 IP 검색결과와 1:1 매칭을 통하여 블랙리스트 IP를 검출한 경우의 검색 결과 IP 리스트를 비교하여 탐색 결과의 정확성을 비교한다.

우선 블룸필터를 이용하지 않은, 통상의 방법으로 검색하는 것은 블랙리스트 IP 24,269개와 검색 대상 IP를 1:1로 대조하므로 이 결과에는 오류가 없고 입력받는 목적지 IP 값의 블랙리스트 포함여부를 100% 필터링한다고 가정한다. 이때 결과는 총 49개의 블랙리스트를 검색해 찾아냈고, 그 리스트는 <표 1> 과 같다.

반면, 블룸필터를 이용하여 검출한 블랙리스트 IP는 <표 2>와 같이 실제 검색된 IP보다 6개가 많은 55개의 IP가 검색되었다. 이 가운데 실제보다 더 검색된 6개의 IP를 제외하고 나머지 49개의 IP는 모두 블랙리스트에 포함되어 있으며, 에러율은 전체 검색한 데이터 49,589개의 IP중 잘못 분류한 IP가 6개 이므로 0.012%가 된다.

**표 1.** 실제 Black IP로 판정된 IP 리스트
**Table 1.** Detected real Black IP destinations

| | | | |
|---|---|---|---|
| 72.246.103.25 | 104.70.122.56 | 162.220.223.28 | 180.234.30.120 |
| 185.188.32.3 | 192.33.14.30 | 195.81.195.55 | 37.252.227.51 |
| 91.198.22.70 | 216.146.43.70 | 46.105.95.113 | 150.70.188.165 |
| 185.188.32.6 | 217.146.26.212 | 37.252.230.28 | 52.27.2.86 |
| 150.70.188.166 | 150.70.188.172 | 150.70.188.175 | 150.70.188.177 |
| 159.8.209.218 | 185.188.32.4 | 199.19.57.1 | 37.252.248.78 |
| 80.82.77.139 | 195.22.28.198 | 123.133.65.58 | 150.70.188.168 |
| 176.126.252.12 | 77.247.181.165 | 195.81.195.53 | 146.0.32.144 |
| 150.70.173.7 | 150.70.188.167 | 150.70.188.169 | 150.70.188.170 |
| 150.70.188.176 | 150.70.188.178 | 150.70.188.179 | 37.252.248.72 |
| 185.188.32.2 | 72.246.103.26 | 185.188.32.5 | 150.70.173.48 |
| 155.94.88.58 | 64.233.188.27 | 173.239.198.45 | 80.83.239.28 |
| 150.70.188.171 | | | |

**표 2.** 블룸필터를 이용하여 검출한 Black IP 리스트
**Table 2.** Detected Black IP destinations with Bloom filter

| | | | |
|---|---|---|---|
| 72.246.103.25 | 104.70.122.56 | 162.220.223.28 | 180.234.30.120 |
| 185.188.32.3 | 192.33.14.30 | 195.81.195.55 | 37.252.227.51 |
| 91.198.22.70 | 216.146.43.70 | 46.105.95.113 | 150.70.188.165 |
| 185.188.32.6 | 217.146.26.212 | 37.252.230.28 | 52.27.2.86 |
| **1.255.33.14** | 150.70.173.48 | 150.70.188.166 | 150.70.188.172 |
| 150.70.188.177 | 155.94.88.58 | 159.8.209.218 | 185.188.32.4 |
| 37.252.248.78 | 64.233.188.27 | 80.82.77.139 | 195.22.28.198 |
| 150.70.188.168 | 173.239.198.45 | 176.126.252.12 | **189.214.0.93** |
| 195.81.195.53 | 146.0.32.144 | 80.83.239.28 | **118.218.9.172** |
| 150.70.188.167 | 150.70.188.169 | 150.70.188.170 | 150.70.188.171 |
| 150.70.188.178 | 150.70.188.179 | **172.114.2.172** | **211.134.21.240** |
| 185.188.32.2 | 72.246.103.26 | 185.188.32.5 | **82.221.103.245** |
| 150.70.188.175 | 199.19.57.1 | 123.133.65.58 | 77.247.181.165 |
| 150.70.173.7 | 150.70.188.176 | 37.252.248.72 | |

#### 4) 블룸필터 에러율에 따른 정확도

블룸필터는 해시함수의 개수에 따라 정확도가 달라진다. 해시함수가 많을수록 블룸필터가 가지고 있는 오탐지율을 낮출 수 있다. 해시함수가 많다는 것은 블룸필터 내에서 특정 원소의 특징들이 세분화 되어있다는 의미와 같기 때문에 보다 정확하게 검색할 수 있다. 그러나 해시함수가 많다는 것은 연산 회수의 증가를 의미하고 이는 결국 성능 저하로 이어진다. 따라서 적절한 에러율을 선택하여 성능에 영향을 주지 않고 정확도를 올리는 방법이 필요하다.

블룸필터의 에러율을 변동시키면서 탐색 시간, 정확도 및 검출한 IP의 개수를 확인한 실험한 결과는 <그림 7> 및 <그림 8>과 같다.


**그림 7.** 블룸필터의 에러율에 따른 탐색 시간 및 정확도
**Fig. 7.** Running times and accuracy depending on error rate of Bloom filter

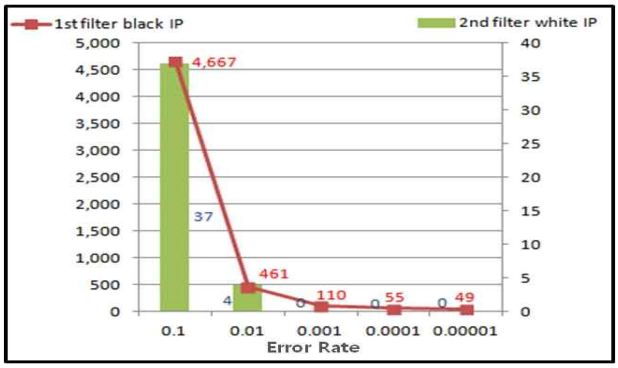


**그림 8.** 블룸필터의 에러율에 따른 탐지된 IP의 수
**Fig. 8.** Number of filtered IP depending on error rate of Bloom filter

에러율 0.1의 경우 블랙 IP로 탐지한 건수가 4,667건으로 올바른 탐지 결과가 49건임을 감안하면 이는 전혀 신뢰할 수 없는 탐지결과 이다. 또한 블랙 IP로 탐지하고 2차에서 화이트리스트 필터에서 화이트 IP로 판정된 건도 37건이나 된다. 에러율 0.01의 경우는 10%정도의 정확도로 블랙 IP를 검출해 내지만 이 역시 신뢰할 수 없는 결과이고, 2차 화이트리스트 필터에서도 4개의 오탐지된 IP를 추가로 걸러냈다. 에러율 0.001의 경우 2차 화이트리스트 필터에서 걸러지는 화이트 IP는 없지만 블랙 IP 역시 올바른 탐지 결과 대비 약 2배가 많은 IP를 검출해 냈다. 에러율 0.0001의 경우는 올바른 탐지 결과인 49개의 IP를 블랙으로 판정해 내 정확도 100%를 보였다. 여기서 특이한 점은 에러율 0.1 및 0.01의 경우 대비 에러율 0.001, 0.0001, 0.00001의 경우가 실행시간이 더 적게 걸렸다는 점이다. 해시함수가 추가되어 연산속도가 늘어나는 것보다, 1차 블랙리스트 필터에서 오탐지로 인해 2차 화이트리스트 필터를 가동하는 것이 오히려 실행시간을 늘렸다고 볼 수 있다. 결국 본 연구에서 최적의 에러율은 0.00001로 설정하면 성능과 정확도 모두 확보할 수 있다.

## Ⅳ. 연구의 한계 및 향후 연구 방향

본 연구에서 사용한 데이터는 이미 발생한 패킷의 데이터를 이용하여 실험한 것으로 본 연구에서 최종적으로 제안하는 모델은 네트워크 보안시스템 단에서 가동하여 제안하는 모델의 효과성 검증이 추가적으로 필요하다.

또한 블랙리스트라는 것 역시 악성코드 유포지 또는 악성 공격을 하는 소스 IP라고 판별된 IP들의 리스트로 만약 조직이 어떤 공격에 새로운 소스로부터 최초로 공격을 받는 경우라면 역시 탐지되지 않을 것이다. 이때 단순한 IP만이 아닌 PC의 프로세스, DNS쿼리, SNMP를 통한 시스템의 상태[12] 등 PC에서 발생하는 각종 징후를 종합적으로 모니터링하여 탐지의 정확성을 확보하는 연구도 진행되어야 한다.

## 참고문헌


[1] S.B. Han and S.K. Hong, “A countermeasure against the APT attack in the financial sector,” *Review of KIISC,* vol. 23, no. 1, pp. 44-53, 2013.

[2] S.C. Noh and K.C. Bang, “A Study on Methodology for Protection of Malicious Traffic in groupware Network System,” *Journal of Digital Contents Society,* vol. 8, no. 1, pp. 69-76, 2007.

[3] W.G. Kim and S.G. Lee, “An improvement of server diffusion prevention of APT attack through the end-point detection and blocking,” in *Proceedings of the Korean Society of Computer Information Conference*, vol. 25, no. 1, pp. 133-134, 2017.

[4] D.S. Moon, H.S. Lee, and I.K. Kim. “Host based feature description method for detecting APT attack,” *Journal of the Korea Institute of Information Security & Cryptology,* vol. 24, no. 5, pp. 839-850, 2014.

[5] K.H. Son, T.J. Lee, and D.H. Won. “Design for zombie PCs and APT attack detection based on traffic analysis,” *Journal of the Korea Institute of Information Security & Cryptology*, vol. 24, no. 3, pp. 491-498, 2014.

[6] M.G. Lee and C.S. Bae, “A study for the principle cases of advanced persistent threat attacks,” *in Proceeding of The Institute of Electronics Engineers of Korea*, pp. 939-942, 2013.

[7] Korea Communications Commission, Report of Investigation for Privacy Leakage at Interpark, Available : http://www.kcc.go.kr/user.do?mode=view&page=A05030000&dc=K00000001&boardId=1113&boardSeq=42740

[8] Burton Bloom, “Space/time tradeoffs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422-426. 1970.

[9] S.Y. Kim and J.H. Kim, “An analysis on the error probability of a Bloom filter,” *Journal of the Korea Institute of Information Security & Cryptology*, vol. 24, no. 5, pp. 809-815, 2014.

[10] E.S. Jung, S.W. Yoo, C.H. Han, and E.J. Park. “Effective detecting DoS attack and scanning at Internet backbone using Bloom filter,” in *Proceedings of Symposium of the Korean Institute of Communications and Information Sciences*, pp. 1298-1301, 2003.

[11] B.J. Choi, M.H. Jeong, S.W. Yoo, B.H. Roh, and K.H. Kim, “Optimized web-server defense against DDoS attack by Bloom filter,” *Journal of Korean Society for Internet Information*, vol. 6, no. 1, pp. 33-36, 2005.

[12] S.J. Moon, “Server Management Prediction System based on Network Log and SNMP,” *Journal of Digital Contents Society,* vol. 18, no. 4, pp. 747-751, 2017.

**강성중**(Seong-Jung Kang)

2008년 : 중앙대학교 컴퓨터공학부 학사
2016년~현 재 : 고려대학교 빅데이터응용 및 보안학과(석사과정)

2008년～2011년: KDB생명보험
2011년～2014년: 아주캐피탈
2014년～현 재: 보험연수원
※관심분야 : 정보보호(Information Security), 네트워크보안, 빅데이터 분석 등

**김형중**(Hyoung-Joong Kim)

1978년 : 서울대학교 전기공학과 학사
1986년 : 서울대학교 제어계측공학과(공학석사)
1989년 : 서울대학교 제어계측공학과(공학박사)

1989년～2006년: 강원대학교 교수
2006년～현 재: 고려대학교 정보보호대학원 교수
※관심분야 : 컴퓨터보안, 패턴인식, 가역정보은닉, 머신러닝, 빅데이터 분석 등