바이러스 쓰로틀링의 웜 탐지 효율 향상 알고리즘
(An Algorithm for Increasing Worm Detection Effetiveness in Virus Throttling)

김 장 복 [†] 김 상 중 [††] 최 선 정 [†††]
(Jang-Bok Kim) (Sang-Joong Kim) (Sun-Jung Choi)
심 재 홍 [††††] 정 기 현 [†††††] 최 경 희 [††††††]
(Jae-Hong Shim) (Gi-Hyun Chung) (Kyung-Hee Choi)

요 약 인터넷 웜 조기 탐지 기법의 대표적인 기술 중 하나인 바이러스 쓰로틀링[5, 6]은 호스트에서 생성되는 접속 요청 패킷을 지연시킴으로써 인터넷 웜의 전파를 줄이는 방법이다. 그러나 기존 바이러스 쓰로틀링은 웜의 발생 시기를 판단하는데 있어서 지연된 접속 요청 패킷의 개수만을 이용한다. 이 때문에 낮은 비율로 웜 패킷을 생성시키는 인터넷웜의 경우에는 웜 탐지 시간이 느린 단점을 가지고 있다. 본 논문에서는 이러한 단점을 해결하기 위해서 지연 큐 길이의 가중치 평균(Weighted Average Queue Length)를 구하고, 그것의 성향을 반영하여 웜 탐지 시간을 단축하고자 한다. 뿐만 아니라 본 논문에서 제안한 알고리즘은 지연큐 변화 성향 반영으로 생길 수 있는 웜 탐지의 오판 가능성을 낮추도록 설계되었다. 그리고 실제 실험을 통해서 본 논문에서 제안한 알고리즘의 성능을 평가한다.

키워드 : 바이러스 쓰로틀링, 인터넷 웜, 웜 조기 탐지

Abstract The virus throttling technique[5,6] is the one of well-known worm early technique. Virus throttling reduce the worm propagration by delaying connection packets artificially. However the worm detection time is not sufficiently fast as expected when the worm generated worm packets at a low rate. This is because the virus throttling technique use only delay queue length. In this paper we use the trend of weighted average delay queue length (*TWAQL*). By using *TWAQL*, the worm detection time is not only shorten at a low rate Internet worm, but also the false alarm does not largely increase. By experiment, we also proved our proposed algorithm had better performance.

Key words : Virus Throttling, Internet Worm, Worm Early Detection

1. 서 론

스스로 전파되는 기능을 가진 인터넷 웜(Internet worm)들은 인터넷에 연결된 호스트들의 취약성을 이용하여 전파된다[1-4]. 웜의 감염된 호스트들은 웜을 다른 호스트에 전파시키기 위해서 스캐닝을 하고 웜 코드를 전파하게 된다. 호스트 스캐닝 과정에서 대상 호스트의 IP 주소를 무작위로 생성하여 패킷을 생성하므로 다량의 트래픽이 발생하게 된다. 따라서 전파되는 웜이 유해한 데이타를 전달하지 않더라도 웜이 전파되는 과정에 생성되는 다량의 네트워크 트래픽은 네트워크의 성능을 현저하게 저하시키는 요인이 된다. 따라서 새로운 웜이 발생하였을 때 웜의 발생 여부를 빠르게 탐지하여 웜 전파를 막는 것은 아주 중요하다.

웜 조기 탐지(Worm early detection) 기술은 웜이 발생하였을 때 빠르게 웜을 탐지하여 웜의 확산을 막기 위한 기술이다. 이러한 기술로는 바이러스 쓰로틀링(virus throttling)[5,6], 송-수신 상호관계(DSC: desti-

· 본 연구는 정보통신부 및 정보통신연구진흥원의 해외교수요원초빙사업의 연구결과로 수행되었음
† 학생회원 : 아주대학교 정보통신전문대학원
bbok@ajou.ac.kr
†† 비 회 원 : 계명문화대학 컴퓨터 인터넷학부 교수
sjkim@km-c.ac.kr
††† 비 회 원 : 경문대학 정보통신과 교수
sjchoi@kmc.ac.kr
†††† 정 회 원 : 조선대학교 인터넷소프트웨어공학부
jhshim@chosun.ac.kr
††††† 정 회 원 : 아주대학교 전자공학부 교수
khchung@ajou.ac.kr
†††††† 종신회원 : 아주대학교 정보통신전문대학원 교수
khchoi@ajou.ac.kr
논문접수 : 2006년 3월 22일
심사완료 : 2007년 3월 16일

nation-source correlation)[7]를 이용한 시스템, 순차적 가설 시험(sequential hypothesis testing)[8,9]을 이용한 알고리즘 등이 있다. 최근에는 접속 실패 회수를 이용한 웜 탐지 기법[13] 및 DNS를 이용한 웜 탐지 기법[14]등이 제안되었다.

그러나 위에서 소개한 방법들은 경계값-기반(threshold-based) 기술로 탐지의 오판(false alarm) 가능성이 높다는 단점을 가지고 있다. 보통 경계값(threshold)을 어떻게 주는가에 따라 성능상의 차이가 발생하는데, 오판율을 낮추기 위해서는 경계값을 좀더 높게 설정하여야 한다. 그러나 이는 결국 웜 탐지 성능을 저하시키기 때문에 동일한 경계값을 가지고도 오판율을 낮추는 것은 아주 중요한 문제이다. C. Zou[10]는 경계값-기반 기술의 경우 오판 가능성이 높다는 단점을 지적하고, 동적 검역 방어 (dynamic quarantine defense)라는 기법을 사용하여 오판율을 낮추는 방법을 제안하였다. 또한 경계값-기반 변이 탐지(threshold-based anomaly detection) 시스템에 트래픽 추세(trend)라는 개념을 칼만-필터(Kalman-filter)를 통해 추가하여 웜 탐지 오판율을 줄이는 방법에 대해 발표하였다[11].

바이러스 쓰로틀링[5,6]은 새로운 세션의 연결(connection) 비율을 제한하여 웜의 전파 속도를 늦추고 차단하는 기법이다. 웜에 감염되지 않은 정상적인 호스트에서 이루어지는 새로운 연결들은 보통 낮은 비율로 이루어지나, 웜은 상대적으로 높은 비율로 연결요청을 시도한다. 바이러스 쓰로틀링은 정상 트래픽과 웜 트래픽의 이러한 차이점을 이용하여 세션의 연결요청 패킷들을 지연시키면서 단위시간당 전송되는 연결요청 개수를 인위적으로 제한함으로써, 웜의 전파 속도를 지연시키는 것은 물론 웜 탐지 시 더 이상의 전파를 차단한다.

그림 1은 바이러스 쓰로틀링 기법에 의해 제어되는 전송 패킷들의 흐름을 나타낸 것이다. 새로운 연결요청 패킷(P)의 전송요청이 들어오면 워킹 셋(working set)에서 P와 동일한 수신 IP 주소가 존재하는지 확인하고, 만약 존재한다면 P를 정상 트래픽으로 간주하여 지연없이 즉시 전송한다. 그렇지 않은 경우 P를 지연 큐(delay queue)에 저장한다. 즉, 상대적으로 최근에 접속이 이루어졌던 호스트에 대해서는 지연 없이 바로 연결요청 패킷을 전송하고 그렇지 않은 호스트에 대해서는 패킷을 지연 큐에 보관한 후 적당한 시기에 비율 제한기(rate limiter)에 의해 전송되게 한다. 비율 제한기는 일정 시간 간격을 두고 주기적으로 지연 큐에서 제일 오래된 패킷을 꺼내어 이를 전송한다. 이때 이 패킷과 동일한 수신 IP 주소를 가지는 지연 큐 내의 다른 패킷들도 동시에 전송한다. 비율 제한기는 매 패킷을 처리할 때마다 해당 패킷의 수신 IP 주소를 워킹 셋에 추가한다. 마지막으로 지연 큐 길이 감시자(queue length detector)는 패킷이 지연 큐에 저장될 때마다 지연 큐의 길이를 검사하여 사전에 정의된 경계값(threshold: 일반적으로 큐 크기임)을 초과하면, 웜이 발생하였다고 판단한다. 일단 웜이 탐지 되면 시스템은 모든 패킷을 차단하여 더 이상의 웜 전파를 차단한다.

그러나 기존 바이러스 쓰로틀링 시스템에서 지연 큐 감시자는 현재 지연 큐의 길이를 경계값과 비교하여 이를 초과했을 때만 웜이 발생했다고 판단하는 단순 판단 기법을 사용하였기 때문에 웜에 의해 발생되는 패킷의 개수가 작을 경우 웜 탐지 시간이 길다는 단점을 가진다. 이를 해결 하기 위해서 [12]에서 현재 큐가 변화하는 성향을 반영한 가중치 평균 큐 길이와 현재 큐 길이를 합하여 그것이 경계값을 초과하였을 때 웜이 발생하였다고 판단하는 알고리즘을 제안하였다. 그러나 이 알고리즘은 웜 탐지 조건을 더욱 민감하게 하기 때문에 오판이 많이 발생하는 단점을 가지고 있다. 특히 P2P와 같이 갑자기 발생한 다량의 정상적인 트래픽에 의해서

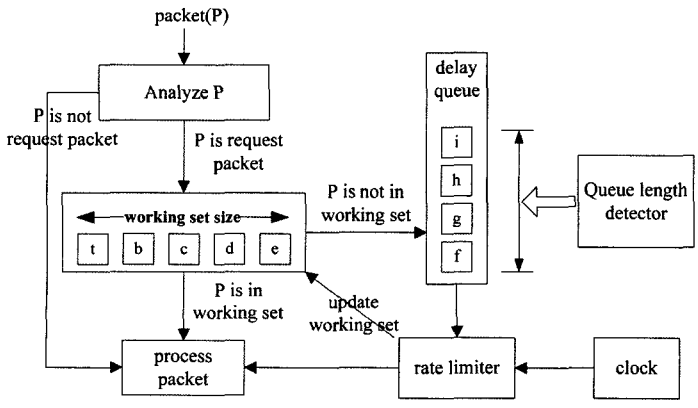

그림 1 바이러스 쓰로틀링 시스템에서의 패킷 흐름도

그것이 웜이라고 오판할 가능성이 높아진다. 뿐만 아니라 웜이 아주 낮은 비율로 발생할 경우 웜 탐지 속도가 여전히 빠르지 못한 단점을 지니고 있다. 본 논문에서는 현재 큐가 변화하는 성향을 반영한 가중치 평균을 그대로 사용하는 것이 아니라 그것의 변화 방향과 지속 시간을 반영하여 위에서 언급한 단점을 해결하고자 한다.

본 논문의 2장에서는 기존 바이러스 쓰로틀링을 개선한 알고리즘에 대해서 논의할 것이다. 그리고 3장에서는 제안된 해결방안의 효율성을 검증하기 위한 실험결과를 제시하고 이를 분석하며, 4장에서 본 논문의 결론을 논의할 것이다.

2. 개선된 웜 탐지 알고리즘

기존 바이러스 쓰로틀링의 큐 길이 감시자는 패킷이 지연큐에 쌓일 때마다 현재 지연 큐 길이를 비교하여 그것이 정해진 경계값(threshold)을 넘어가면 웜이 발생하였다고 탐지하는 방법을 사용하였다. 그러나 웜 패킷의 발생 비율이 낮은 경우에는 지연 큐 길이를 이용하여 웜을 탐지하는 방법은 탐지 속도가 빠르지 못하다. 이에 [12]에서는 현재 큐 길이의 변화 추이를 웜 탐지에 반영하여 빠르게 웜을 탐지하려 하였다. 큐 길이의 추세를 반영하기 위해서 최근 지연 큐 길이의 가중치 평균 큐 길이(Weighted Average Queue Length: WAQL)를 사용하였다[12]. 그리고 만약 현재 지연 큐 길이와 가중치 평균 큐 길이의 합이 정해진 경계값을 넘으면 웜을 탐지함으로써 낮은 비율로 웜 패킷을 생성하는 웜에 대해서도 기존 바이러스 쓰로틀링보다 빠르게 탐지할 수 있었다.

if (*WAQL* + *DQL*) > *threshold* **then**
  Worm is detected
else
  Worm is not detected
end if

그러나 이 방법은 낮은 비율의 웜에 대한 반응 속도가 여전히 높지 못하다는 단점과 갑자기 발생한 정상적인 트래픽에 웜의 오판 탐지 가능성이 존재하는 단점을 가지고 있다. 그림 2와 그림 3은 이러한 단점을 표현한 그림이다. 먼저 그림 2는 [12]의 알고리즘에서 초당 2개의 패킷을 생성하는 웜을 시뮬레이션하였을때 *WAQL*, *DQL*, *DQL+WAQL*의 변화 모습을 표현한 것이다. *DQL*은 현재 지연 큐 길이를 나타낸다. 그림에서 보는 바와 같이 *DQL*과 *WAQL*은 *DQL*의 증가 시간과 관계없이 *DQL*의 값을 부드럽게 필터링한 효과만을 가진다. 따라서 그림 2와 같은 상황에서는 여전히 웜 탐지 시간

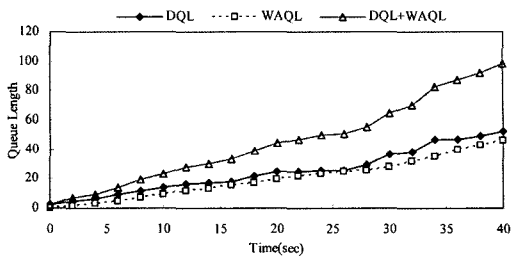

그림 2 낮은 비율의 패킷을 생성하는 웜에서의 *WAQL*의 변화 모습

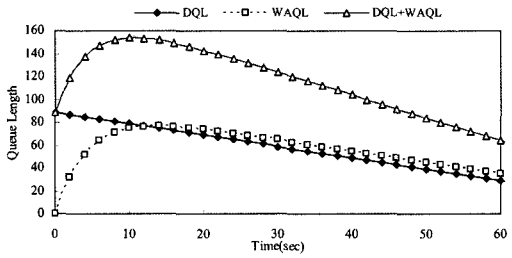

그림 3 갑작스러운 패킷 생성시의 *WAQL*의 변화 모습

은 길 수 밖에 없다. 그리고 그림 3은 0초에 90개의 패킷을 갑자기 발생 시켰을때의 *DQL*, *WAQL*, *DQL+WAQL*의 변화 모습을 그래프로 도식화한 것이다. 웜 패킷이 발생한 이후 더이상의 패킷이 나타나지 않음에도 *WAQL*은 계속해서 증가하여 *DQL+WAQL*값이 160가깝게 증가되는 현상을 볼 수 있다. 만약 [5,6]에서와 동일하게 경계값을 100으로 설정하였다면, [12]의 알고리즘을 사용하면 갑작스러운 패킷 증가로 인한 웜이 발생하였다고 오판하게 된다.

위에서 언급한 단점을 해결하기 위해서 본 논문에서는 지연 큐 길이가 변화하는 방향과 지속 시간을 이용한 지연 큐의 가중치 평균의 추세값(*TWAQL*)을 사용할 것이다. 만약 현재 지연 큐 길이와 *TWAQL*의 값의 합이 지정된 경계값 보다 커지게 된다면 제안된 알고리즘은 웜이 발생하였다고 판단하게 된다.

if (*TWAQL* + *DQL*) > *threshold* **then**
  Worm is detected
else
  Worm is not detected
end if

먼저 *TWAQL*을 구하기 위해서 지연 큐 길이의 가중치 평균(*WAQL*)을 사용한다. 지연 큐 길이의 가중치 평균 값은 다음과 같이 구한다. 지연 큐의 길이는 *TWAQL* 계산에 바로 사용하지 않는 이유는 지연 큐의

길이를 그대로 사용할 경우 지연 큐의 증감이 급속하게 바뀔 수 있기 때문이다.

$$WAQL_n = a * avgDQL_n + (1-a) * WAQL_{n-1}$$

$avgDQL_n$은 구간 $n-1$과 n 사이의 평균 지연 큐의 길이를 나타내며, $WAQL_{n-1}$은 현 시점을 기준으로 바로 전 구간에서의 가중치 평균값을 나타낸다. 마지막으로 a는 $WAQL_n$을 계산할 때 현주기의 $avgDQL_n$과 $WAQL_{n-1}$의 가중치를 결정하기 위해서 사용되어지는 가중치 상수 이다. 이렇게 구한 지연 큐 길이의 가중치 평균을 이용하여 지연 큐 길이의 변화 방향을 결정하며, 방향이 지속되는 시간에 따라서 지연 큐의 가중치 평균값의 추세를 나타내는 값($TWAQL$)을 계산하게 된다.

```
WAQL_{n+1} = α * avgDQL_n + (1- α)*WAQL_n
t_{n+1} = Current Time

if WAQL_{n-1} < 1 then
        Direction = STAY
        w_0 = t_n
else if WAQL_{n+1} - WAQL_n >= ε && Direction != UP then
        Direction = UP
        t_0 = t_n
        w_0 = TWAQL_n
else if WAQL_{n+1} - WAQL_n <= - ε && Direction != DOWN then
        Direction = DOWN
        t_0 = t_n
        w_0 = TWAQL_n
else if abs(WAQL_{n+1} - WAQL_n) < ε   && Direction != STAY then
        Direction = STAY
        w_0 = TWAQL_n
end if

if Direction == UP then
        TWAQL_{n+1} = β * (t_{n+1} - t_0)^2 + w_0
else if Direction == DOWN then
        TWAQL_{n+1} = -β * (t_{n+1} - t_0)^2 + w_0
else
        TWAQL_{n+1} = w_0
end if
```

그림 4 $TWAQL$을 계산하는 알고리즘

그림 4는 $TWAQL$을 구하기 위한 알고리즘을 표현한 것이다. $TWAQL$값은 알고리즘에서도 볼 수 있듯이 $WAQL$의 증감 방향에 따라 증가 또는 감소하게 되며, 그리고 방향이 지속되는 시간의 거듭 제곱으로 변화한다. 만약 $WAQL$이 증가한다면, $TWAQL$은 시간 t에 대해서 $TWAQL_{n+1} = \beta * (t-t_0)^2+w_0$를 이용하여 계산한다. β는 가중치 값이며, t_0는 $WAQL$ 값이 증가하기 시작한 시간이다. 반대로 $WAQL$의 값이 감소하게 되면 $TWAQL$은 $TWAQL_{n+1} = -\beta * (t-t_0)^2+w_0$와 같다. 그리고 $WAQL$의 값이 증가하거나 감소하는 판단 기준은 ε 값을 사용하며, $WAQL_{n+1}$과 $WAQL_n$ 차이의 절대값이 ε 보다 작다면 $WAQL$값은 변화하지 않는 것으로 간주한다. $WAQL$값이 증가하지도 감소하지도 않으면, $TWAQL$값도 변화하지 않는다. 만약 $TWAQL$값이 0보다 작아지게 되면 $TWAQL$값은 0으로 설정된다.

본 논문에서 제안한 알고리즘을 사용하게 되면 웜 탐지 시간은 적어도 $\sqrt{Threshold/\beta}$ 안에 탐지되게 된다. 왜냐하면 웜 전파가 탐지되는 조건이 $TWAQL_{n+1} + DQL = threshold$ 이고, $TWAQL_{n+1} = \beta*(t-t_0)^2 + w_0$ 이기 때문에 최소 웜 탐지 시간은 $\sqrt{(Threshold - DQL - w_0)/\beta}$ 가 된다. DQL의 최소값이 0이기 때문에, 결국 웜 탐지는 적어도 $\sqrt{Threshold/\beta}$ 안에 이루어지게 된다.

그리고 본 논문에서 제안한 알고리즘은 기존 바이러스 쓰로틀링 시스템과 비교하여 전체 시스템의 처리 능력에는 크게 영향을 주지 않는다. 이는 제안한 알고리즘이 반복 등이 포함되지 않은 사칙 연산 및 조건으로만 이루어져 알고리즘 복잡성을 변화시키지 않기 때문이다.

3. 성능 평가

제안한 알고리즘의 성능을 평가하기 위해서 먼저 리눅스(Linux) 시스템의 커널에 기존 바이러스 쓰로틀링과 개선된 바이러스 쓰로틀링을 구현하였다. 그리고 리눅스 양단에 패킷 생성기와 패킷 수신기를 설치하였다. 패킷 생성기는 인터넷 웜 등의 패킷을 생성하고, 패킷 수신기는 바이러스 쓰로틀링이 구현된 리눅스를 통과한 패킷을 수신하는 역할을 한다. 이때 리눅스 시스템은 패킷 수신기와 연결된 인터페이스에만 바이러스 쓰로틀링이 동작하도록 하여 패킷 수신기로 전달되는 패킷 전달을 조절하도록 하였다. 실험을 위한 환경은 그림 5와 같다.

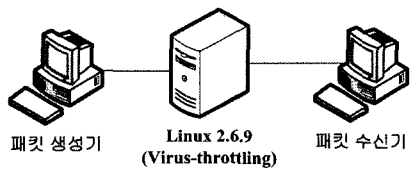

그림 5 성능 평가를 위한 실험 환경

먼저 그림 2와 그림 3의 환경과 동일한 조건에서 본 논문에서 제안한 알고리즘을 이용하였을 때의 $TWAQL$과 $CQL+TWAQL$의 변화를 살펴보았다. 가중치 평균을 구하기 위한 샘플링 시간은 2초로 하였으며, $a=0.4$, $\beta=0.1$, $\varepsilon=0.1$, 웜탐지 경계값은 100으로 설정하였다. 그리고 바이러스 쓰로틀링의 워킹 셋의 크기는 5로 설정하였다. 그림 6은 초당 2개의 패킷을 생성하는 웜을 생성하였을 경우 DQL, $WAQL$, $TWAQL$과 $CQL+TWAQL$의 값의 변화를 도식화한 것이다. 그림 2에서는 웜 탐지 시간이 40초 걸린 반면 본 논문에서 제안한 알고리즘은 28초만에 웜을 탐지하였음을 알 수 있다. 그 이유는 웜이 생성하는 패킷의 양이 작더라고, $WAQL$이 지속적으로 증가하게 되면, $TWAQL$이 $WAQL$의 크기와 상관없

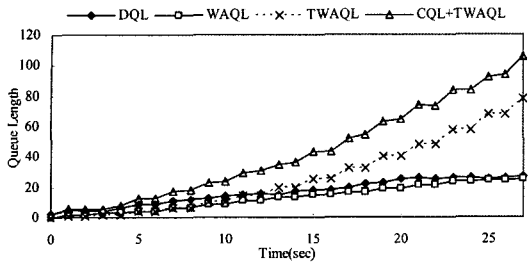

그림 6 낮은 비율의 패킷을 생성하는 웜에서의 *TWAQL*의 모습

이 상승 곡선의 지속시간을 기준으로 거듭제곱으로 커지기 때문이다.

그림 7은 그림 3과 비교를 위해서, 그림 3 실험과 마찬가지로 0초에 90개의 패킷을 생성하였을때 *DQL, WAQL, TWAQL, CQL+TWAQL*의 변화 모습을 그래프로 표현한 것이다. 본 논문에서 제안한 알고리즘에서는 *WAQL*의 값이 웜 탐지 기준에 직접적으로 반영되지 않고, *WAQL*의 변화 성향이 *TWAQL*이라는 값으로 반영되어 웜 탐지에 사용되어진다. 따라서 *DQL*이 0초에서 90개로 늘어난 뒤 점차 감소하므로, *WAQL* 역시 약간의 상승후에 감소 추세를 가져오게 되므로 *TWAQL* 값은 크게 증가하지 않는다. 따라서 그림 3과는 다르게 *CQL+TWAQL*값이 100을 넘지 못한다. 결국 경계값을 100으로 설정하였을 때 그림 3은 웜이 발생하였다고 오판하게 되나, 본 논문에서 제안한 알고리즘은 웜이 발생하였다고 판단하지 않는다. 이는 인터넷 트래픽의 특성인 갑작스러운 트래픽(Burst traffic)에도 유연하게 동작이 가능하다는 의미로써 웜 탐지 속도를 늘이면서도 웜 탐지 오판을 줄일 수 있다는 것으로 해석할 수 있다. 예를 들어 P2P를 사용한다고 가정하였을 때, P2P 프로그램에 의해서 생길 수 있는 동시 접속 요청 개수가 90개까지는 웜으로 판단하지 않는다는 것을 의미한다. 그러나 P2P를 사용한다고 하더라도 동시 접속 요청 개수가 90개까지 올라가는 경우는 상당히 드물기 때문에 P2P 환경에서 웜이 탐지되었다고 오판할 가능성은 낮아지게 된다.

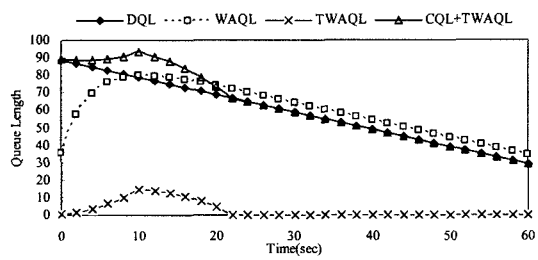

그림 7 갑작스러운 패킷 생성시의 TWAQL의 변화 모습

바이러스 쓰로틀링의 비율 제한기는 접속 요청 패킷의 지연시간을 결정시켜주는 주요 인자중 하나로써, 비율 제한기의 주기가 짧아질수록 사용자의 접속 요청 패킷의 지연시간이 줄어든다. 물론 비율 제한기의 주기가 짧아지면 짧아질수록 웜 탐지 시간이 늘어나고 낮은 비율의 패킷을 생성하는 웜은 탐지하지 못하게 된다. 이때 비율 제한기의 주기가 변화할 때 [5,6]에서 제안한 기존 바이러스 쓰로틀링과, [12]에서 제안한 *WAQL*을 이용한 바이러스 쓰로틀링, 그리고 본 논문에서 제안한 알고리즘을 이용한 바이러스 쓰로틀링에 대해서 웜 탐지 시간이 어떻게 변화하는지 알아보았다. 이를 위해서 비율 제한기의 주기를 1, 0.75, 0.5, 0.25초로 변화시켰으며, 초당 발생되는 웜 패킷의 개수는 5개로 하였다. 다른 파라미터들은 이전 실험과 동일하게 설정하였다.

그림 8은 실험 결과를 그래프로 도식화한 것이다. 'Typical'은 [5,6]에서 제안한 바이러스 쓰로틀링의 결과값이며, 'use *WAQL*'은 [12]에서 제안한 알고리즘을 사용하였을 때의 결과값이다. 그리고 'use *TWAQL*'은 본 논문에서 제안한 알고리즘을 사용하였을 때의 결과값을 나타낸다. 예상했던대로 개선된 2개의 알고리즘들은 모든 구간에서 기존 알고리즘보다 빠른 탐지 시간을 가졌다. 그러나 비율 제한기 주기의 변화에 따라 각 알고리즘의 웜 탐지 시간의 변화 모습은 다르게 나타났다. 기존 바이러스 쓰로틀링은 비율제한기가 1초에서 0.25초로 줄어들면 거의 5배 가깝게 웜 탐지 시간이 느려졌고, *WAQL*을 직접 사용한 알고리즘의 경우에는 2배정도 탐지 시간이 느려졌다. 그러나 본 논문에서 제안한 알고리즘을 사용하였을 경우에는 거의 시간변화가 없다는 것을 알 수 있다. 즉, 본 논문에서 제안한 알고리즘은 모든 경우에 대해서 비슷한 성능으로 인터넷 웜의 발생을 탐지 해낼 수 있다는 것을 알 수 있다. 그 이유는 앞서 이야기 했듯이 본 논문에서 제안한 알고리즘은 시간이 지연큐의 길이가 증가하는 구간이 늘어날수록 *TWAQL*값이 커지기 때문에 비율 제한기의 주기가 낮은 경우에도 좋은 성능을 보여주게 된다. 물론 비율 제

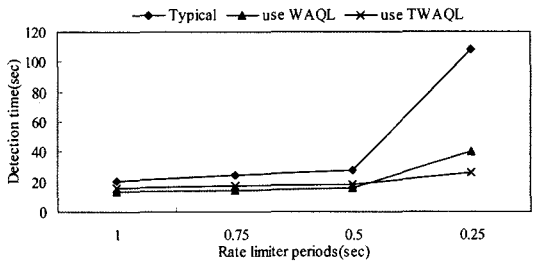

그림 8 비율 제한기(Rate limiter)의 주기에 따른 웜 탐지 시간의 비교

한기의 주기가 1, 0.75, 0.5초일 때 *WAQL*을 직접 사용한 알고리즘을 사용하였을 때보다 웜 탐지 시간이 느리지만 무시할 정도의 시간이기 때문에 본 논문에서 제안한 알고리즘이 더욱 우수함을 알 수 있다.

그림 9는 웜이 발생시키는 초당 웜 패킷수의 변화에 따른 3가지 알고리즘에 대해서 웜 탐지 시간의 변화를 나타낸 것이다. 초당 웜 패킷수는 1, 2, 4, 8, 16 개로 변화시켰으며, 비율 제한기의 주기는 1초로 고정하였다. 그림 9의 결과 역시 예상했듯이, 초당 웜 패킷수가 작은 경우 본 논문에서 제안한 알고리즘이 월등히 높은 성능을 보여주었으며, 초당 웜 패킷수가 늘어날 경우에는 3가지 알고리즘 모두 비슷한 성능을 보여주었다. 그 이유는 앞선 실험에서 설명한 것과 동일하다.

마지막으로 그림 10은 본 논문에서 제안한 알고리즘의 β 값의 변화에 따라서 웜의 탐지 시간 변화모습을 나타낸 것이다. 비율 제한기의 주기는 1초로 하였으며, 초당 2개의 패킷을 생성하였을 때 β 값을 0.01부터 1까지 변화시키면서 웜 탐지 시간을 측정하였다. 그림 10의 직선은 실험을 통해서 얻은 값이며, 점선은 본 논문의 알고리즘의 최소 탐지 시간 $\sqrt{Threshold / \beta}$ 을 이용하여 계산하여 표현한 것이다. 먼저 β 값이 증가할수록 웜 탐지 시간은 짧아진다는 것을 볼 수 있다. 이는 β 값이 커지면 *TWAQL*이 상승하는 속도가 빨라지기 때문이다. 그리고 실험으로 측정한 웜탐지 시간은 항상 최소 탐지 시간보다 빠르게 웜을 탐지 하고 있음을 위 그래프를 통해서 확인 할 수 있다.

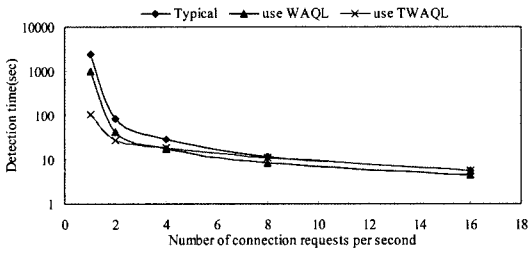

그림 9 초당 웜 패킷수에 따른 웜 탐지 시간의 비교

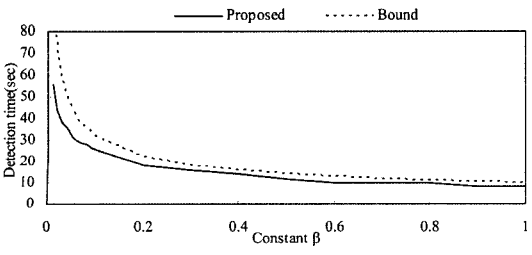

그림 10 β 값의 변화에 따른 웜 탐지 시간

4. 결 론

본 논문에서 바이러스 쓰로틀링의 지연큐 길이의 가중치 평균(*WAQL*)을 구하고 가중치 평균의 변화 성향을 반영하여 기존 바이러스 쓰로틀링보다 빠르게 웜을 탐지할 수 알고리즘을 제안하였다. 이 알고리즘은 이전 연구[12]에서 지연큐 길이의 가중치 평균을 바로 웜 탐지에 반영하였던 것에 비해 웜 탐지 오판이 줄일 수 있으며, 또한 더욱 빠르게 웜을 탐지 할 수 있다. 본 논문에서는 이러한 장점들을 확인하기 위해서 바이러스 쓰로틀링을 리눅스에 구현하고 실험을 하여 본 논문에서 제안한 알고리즘이 우수함을 증명하였다.

참 고 문 헌

[1] CERT, "CERT Advisory CA-2003-04 MS-SQL Server Worm," Jan. 2003. http://www.cert.org/advisories/CA-2003-04.html

[2] CERT, "CERT Advisory CA-2001-09 Code Red II Another Worm Exploiting Buffer Overflow in IIS Indexing Service DLL," Aug. 2001. http://www.cert.org/incident_notes/IN-2001-09.html

[3] CERT, "CERT Advisory CA-2001-08 Code Red Worm Exploiting Buffer Overflow in IIS Indexing Service DLL," July 2001. http://www.cert.org/incident_notes/IN-2001-08.html

[4] CERT, "CERT Advisory CA-2001-26 Nimda Worm," Sept. 2001. http://www.cert.org/advisories/CA-2001-26.html

[5] Matthew M. Williamson, "Throttling Viruses: Restricting propagation to defeat malicious mobile code," Proc. of the 18th Annual Computer Security Applications Conference, Dec. 2002.

[6] J. Twycross and M. M. Williamson, "Implementing and testing a virus throttle," Proc. of the 12th USENIX Security Symposium, pp. 285-294, Aug. 2003.

[7] X. Qin, D. Dagon, G. Gu, and W. Lee, "Worm detection using local networks," Technical report, College of Computing, Georgia Tech., Feb. 2004.

[8] J. Jung, S. E. Schechter, and A. W. Berger, "Fast Detection of Scanning Worm Infections," Proc. of 7th International Symposium on Recent Advances in Intrusion Detection (RAID), Sophia Antipolis, French Riviera, France, Sept. 2004.

[9] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," Proc. of the IEEE Symposium on Security and Privacy, May 2004.

[10] C. C. Zou, W. Gong, and D. Towsley, "Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense," ACM CCS Workshop on Rapid Malcode (WORM'03), Washington DC, Oct. 2003.

[11] C. Zou, L. Gao, W. Gong, D. Towsley, "Monitoring and early warning for Internet worms," ACM Conference on Computer and Communications Security, Washington, DC, Oct. 2003.
[12] Jangbok Kim, Jaehong Shim, Gihyun Jung, and Kyunghee Choi, "Reducing Worm Detection Time and False Alarm in Virus Throttling," LNAI 3802, p.297, December 2005.
[13] Stuart Staniford, "Containment of scanning worms in enterprise networks," Journal of Computer Security, 2004.
[14] David Whyte, Evangelos Kranakis, P.C. van Oorschot, "DNS-based Detection of Scanning Worms in an Enterprise Network," In Proc. of the 12th Annual Network and Distributed System Security Symposium, Feb. 2005.

김 장 복

2003년 아주대학교 정보및컴퓨터공학부 졸업(학사). 2005년 아주대학교 정보통신전문대학원 정보통신공학과 졸업(석사) 2005년~현재 아주대학교 정보통신전문대학원 컴퓨터공학과 박사과정. 관심분야는 네트워크 보안, 임베디드시스템, 운영체제, 임베디드 소프트웨어 테스팅

김 상 중

1977년 한양대학교 전자공학과 졸업(학사). 1980년 연세대학교 전자공학과 졸업(석사). 1997년 아주대학교 전자공학과 졸업(박사). 1977년~1998년 한국전자통신연구소 책임연구원. 1998년~현재 현재 계명문화대 컴퓨터인터넷학부 교수. 관심분야 정보통신시스템, 멀티미디어 컨텐츠, 멀티디어어 서비스 등

최 선 정

1995년 아주대학교 전자공학과 졸업(박사). 1985년~1997년 삼성SDI 연구소 수석연구원. 1997년~현재 국제대학 정보통신계열 부교수. 관심분야는 임베디드 시스템, 인터넷 보안, 실시간 시스템 등

심 재 홍

1987년 서울대학교 전산과학과 졸업(학사). 1989년 아주대학교 컴퓨터공학과 졸업(석사). 2001년 아주대학교 컴퓨터공학과 졸업(박사). 1989년~1994년 서울시스템(주) 공학연구소. 1999년~2000년 University of Arizona 객원연구원. 2001년~2001년 9월 아주대학교 정보통신전문대학원 BK21 전임연구원. 2001년 10월~현재 조선대학교 인터넷소프트웨어공학부 조교수. 관심분야 임베디드시스템, 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템

정 기 현

1984년 서강대학교 전자공학과 졸업(학사). 1988년 미국 Ilinois주립대 EECS 졸업(석사). 1990년 미국 Purdue대학 전기전자공학부 졸업(박사). 1991년~1992년 현대반도체 연구소. 1993년~현재 아주대학교 전자공학부 교수. 관심분야는 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등

최 경 희

1976년 서울대학교 수학교육과 졸업(학사). 1979년 프랑스 그랑데꼴 Enseeiht대학 졸업(석사). 1982년 프랑스 Paul Sabatier대학 정보공학부 졸업(박사). 1982년~현재 아주대학교 정보통신전문대학원 교수. 관심분야는 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템 등