응답 시간 기반의 EDoS 공격 대응 기법 연구

이강빈, 장진영O, 강수민, 김태운*
부산대학교 정보컴퓨터공학부
midasj0314@pusan.ac.kr, ddglackrp@pusan.ac.kr, ramiregi@pusan.ac.kr, taewoon@pusan.ac.kr

Research on the Response Time-based EDoS Attack Defense Technique

GangBeen Lee, JinYoung JangO, SuMin Kang, TaeWoon Kim*
School of Computer Science and Engineering, Pusan National University

요 약

이 논문은 클라우드 서비스 사용이 급증하면서 새롭게 대두된 클라우드 환경 공격인 EDoS, 특히 YoYo Attack을 효과적으로 방어하기 위한 매커니즘을 제안한다. YoYo Attack은 Application-Layer 요청을 통해 서버 자원을 과도하게 사용하여, 클라우드의 자동 확장 기능을 악용함으로써 경제적 피해를 유발하는 공격 방식이다. 본 연구에서는 공격자의 응답 시간을 교란하여 YoYo Attack을 방어하고, 단위 시간당 요청 횟수에 따라 방어 적용 강도를 동적으로 조절하는 매커니즘을 제안한다. 이러한 매커니즘은 공격자의 공격 성공 여부에 혼란을 야기하고, 클라우드 자원의 효율적 관리를 가능하게 하여 경제적 피해를 줄이는 것을 목표로 한다. 연구 결과, 제안된 방어 매커니즘이 YoYo Attack에 효과적으로 대응할 수 있음을 확인하였으며, 클라우드 자원의 남용을 방지하고 서버의 안정적인 운영을 도모 할 수 있을 것으로 기대된다.

1. 서 론

클라우드 서비스 사용이 급증하면서 많은 기업들이 온프레미스 환경에서 클라우드로 전환하고 있다 [1][2]. 이로 인해 해커들은 클라우드 환경을 겨냥한 새로운 공격 방식을 개발하고 있다. [3]

EDoS (Economic Denial of Sustainability)는 클라우드 환경을 대상으로 하는 공격 방식 중 하나로, 클라우드의 자동 확장 기능인 Auto Scaling을 악용하여 경제적 피해를 유발한다. 그 중 YoYo Attack은 Application-Layer 요청을 통해 서버 자원을 과도하게 사용하게 하여 사용량 기반 과금 시스템에서 막대한 비용을 발생시킨다.

관련 연구에 따르면 YoYo Attack은 기존 DDoS와 달리 공격자의 자원이 적게 들며, 장기간에 걸친 저강도 공격으로, 정상적인 트래픽과 구분하기 어려워 기존 방어 방식이 효과적이지 않을 수 있다고 지적한다. 하지만 방어 전략으로 자동 확장 기능 정책의 조정이나 자원 제한 방식이 제안되고 있으나, 이러한 방법은 서비스 비용 증가나 성능 저하라는 Trade-off를 동반하는 한계가 있다. [4]

본 연구는 클라우드 환경에서 이러한 위협에 효과적으로 대응하기 위해 YoYo Attack 방어를 위한 새로운 매커니즘을 개발하는 것을 목표로 한다. 이를 통해 클라우드 자원의 효율적 관리와 기업의 재정적 피해를 줄이는 데 기여하고자 한다.

2. 배경 지식

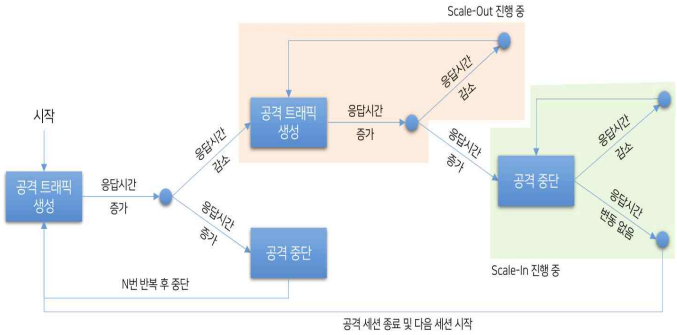
2.1 공격자 매커니즘

YoYo Attack은 다수의 Application-Layer 요청을 통해 서버에 부하를 가하여 Auto Scaling의 Scale Out을 유도하고, Pod 수를 증가시킨다. 이후 공격을 중단하여 Auto Scaling의 Scale In을 유도해 Pod 수를 감소시킨다. 이후 해당 과정을 반복한다. 이를 통해 서버에 경제적 타격을 가한다.

공격자는 서버의 응답 시간을 관찰하면서 공격을 조절한다. 서버가 다량의 요청으로 인해 부하가 증가하고, 이를 효과적으로 처리하기 위해 Pod를 추가로 생성하면, 전체적인 응답 속도가 빨라지게 된다. 이 경우 공격자는 응답 시간이 감소한 것을 보고, 공격이 원활하게 진행되고 있다고 판단한다.

반면, Pod를 더 이상 추가하지 못하고 처리 능력이 한계에 도달하면, 요청에 대한 응답 시간이 증가하거나 일정하게 유지된다. 이때 공격자는 응답 시간이 길어진 것을 통해 공격이 더 이상 진행되지 않는다고 판단하여 공격을 중단한다.

결론적으로 YoYo Attack의 공격자는 서버의 응답 시간 변화를 이용하여 공격의 성공여부를 판단한다. 공격 매커니즘의 흐름도는 (그림 1)과 같다.



(그림 1) 연구에 사용한 공격자 매커니즘

2.2 방어 매커니즘 핵심 Metric : 요청에 대한 응답시간

공격자는 요청에 대한 응답 시간을 기반으로 공격 지속 여부를 결정한다. 따라서 서버는 YoYo Attack을 효과적으로 방어하기 위해 응답시간을 교란하여 공격자가 공격이 원활하게 진행되지 않는다고 판단하게 한다. 이를 통해 공격자의 공격을 중단 시키고, 서버의 부하를 줄임으로써 Pod 수를 적절하게 유지하도록 한다.

2.3. 단위 시간 당 요청 횟수에 따른 방어 매커니즘 동적 적용

본 연구에서 제안하는 YoYo Attack 방어 매커니즘은 정상 사용자의 피해를 최소화 하기 위해, 단위 시간 당 요청 횟수에 따라 방어 매커니즘 적용 강도를 동적으로 조절한다.

2.3.1 단위 시간 기준

서버는 Scale Out과 Scale In의 소요 시간을 기준으로 단위 시간을 설정한다. 즉 Pod가 최대 개수로 확장된 이후 최소 개수로 축소되는 시간을 단위 시간으로 본다. 이 시간 동안의 요청 횟수를 계속 갱신해서 방어 매커니즘을 동적으로 적용한다.

2.3.2 단위 요청 횟수 기준

Pod를 하나 추가할 수 있는 요청 횟수를 기준으로 단위 요청 횟수를 설정한다. Pod를 최대 개수로 확장 시키려면 Pod 하나 추가하는 데 필요한 요청 횟수보다 더 많은 요청을 보내야 한다. 따라서, 이 요청 횟수를 기준으로 방어 매커니즘을 적용한다.

2.3.3 Sigmoid 함수 활용

요청을 많이 보내지 않는 정상 사용자는 방어 매커니즘의 영향을 적게 받고 서비스를 원활하게 사용할 수 있어야 한다. 이를 위해 요청 횟수가 적을 때 방어 매커니즘의 적용 강도를 약하게 하고, 요청 횟수가 증가할수록 방어 매커니즘 적용 강도가 강해지도록 Sigmoid 함수를 적용한다. Sigmoid 함수의 파라미터는 서비스 환경에 맞추어 조정할 수 있다.

3. 제안하는 방어 매커니즘

본 연구에서 제안하는 방어 매커니즘은 다음의 세 가지 방법으로 구성된다.

1. Thread Sleep 활용
2. VM 방화벽 설정을 활용한 패킷 드랍
3. 더미 서버로의 확률적 리다이렉트

세 가지 방어 매커니즘은 단위 시간당 요청 횟수에 기반하여 공격자의 요청에 대한 응답을 효과적으로 지연시키고, 서버 부하를 줄여 안정적인 서비스를 유지한다.

3.1 Thread Sleep 활용

단위 시간당 요청 횟수를 Sigmoid 함수에 적용하여 Thread Sleep 시간을 결정하고, 해당 IP의 요청을 처리하는 Thread를 일정 시간 동안 Sleep을 활용하여 멈춘다. 요청 횟수가 많은 IP에 대해서는 긴 Sleep 시간을 적용하여, 의도적으로 응답을 지연시킨다. 이를 통해 공격자의 요청에 대해 효과적으로 응답 시간을 교란할 수 있으며, 즉각적인 요청 처리를 줄임으로써 방어 매커니즘 적용 전에 비해 동일 시간에 들어오는 요청 횟수를 감소시키고, 서버 부하를 줄인다. 이를 통해 Pod 수를 적절하게 유지한다.

3.2 VM 방화벽 설정을 활용한 패킷 드랍

방화벽 설정을 통해 특정 IP의 패킷을 일정 시간 동안 확률적으로 드랍하고 다시 해제한다. 패킷 드랍을 해제하는 이유는 해당 IP가 공격자가 아닌 정상 사용자일 가능성도 있기 때문이다.

확률은 단위 시간당 요청 횟수를 Sigmoid 함수에 적용하여 설정한다. 요청 횟수가 많은 IP일수록 패킷이 더 자주 드랍된다. 이를 통해 공격자의 요청을 주기적으로 드랍하여 응답 시간을 교란하고, 즉각적인 요청 처리를 줄임으로써 동일 시간 내의 요청 횟수를 감소시킨다. 결과적으로 Pod 수를 적절히 유지하면서 공격자의 공격 중단을 유도할 수 있다.

3.3 더미 서버로의 확률적 리다이렉트

더미 서버는 기능은 같지만 Auto Scaling이 적용되지 않고, 응답을 의도적으로 지연시키는 서버이다.

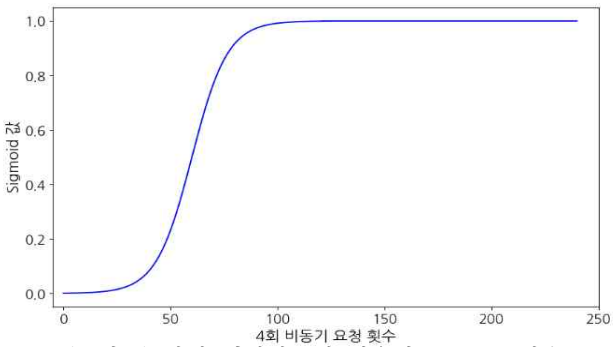
단위 시간당 요청 횟수를 Sigmoid 함수에 적용해 리다이렉트 확률을 결정하고, 해당 확률에 따라 요청을 더미 서버로 보낸다. 요청 횟수가 많아질수록 더미 서버로 리다이렉트될 확률이 증가하며, 이를 통해 공격자의 요청에 대한 응답 시간을 지연시켜 교란한다. 또한, 다량의 요청이 본 서버 대신 더미 서버로 리다이렉트되므로 본 서버의 부하를 줄이고, Pod 수를 적절하게 유지한다.

4. 실험 및 분석

제안된 방어 매커니즘을 검증하기 위해, AWS T3 medium 정도의 성능을 갖춘 VM 환경에서 실험을 진행했다. 서버는 minikube를 활용하여 HPA(Horizontal Pod Auto Scaling)를 적용했다. 공격자는 두 대의 인스턴스를 사용해 공격을 시도했다. 이 실험에서는 공격자가 서버를 AWS T3 인스턴스 중 성능이 가장 높은 t3.2xlarge로 가정하여 공격을 진행했다. 공격자는 서버가 이상적으로 처리할 수 있는 최대한의 동시 요청 8개(vCPU 8개)를 보내는 방식으로 실험을 진행했다. [5]

4.1 방어 매커니즘에 적용한 Sigmoid 함수

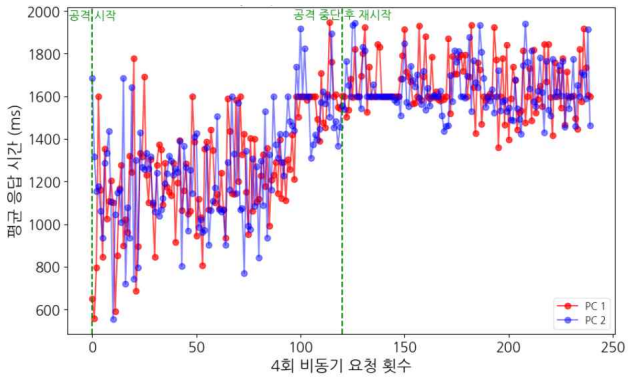
(그림 2)는 실험 환경에서 방어 매커니즘에 적용한 Sigmoid 함수이다. 실험에 사용한 서버는 단위 시간 내 240회 정도의 요청이 들어오면 Pod 수가 하나 증가했다. 해당 정보를 기반으로 4회 비동기 요청 횟수를 기준으로 60회에 Sigmoid 함수의 결과가 0.5가 되도록 하였다. Sigmoid 함수의 결과값을 통해 방어 매커니즘의 강도를 동적으로 조절했다.



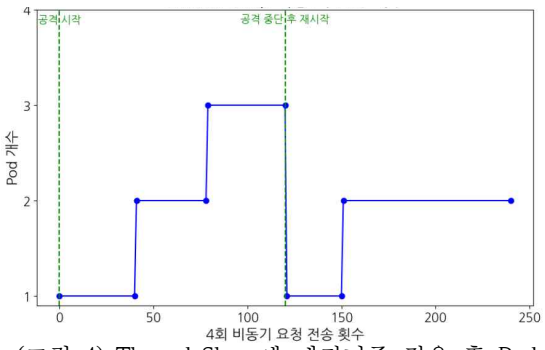
(그림 2) 방어 매커니즘에 적용된 Sigmoid 함수

4.2 Thread Sleep 실험 및 분석

Thread Sleep 방어 매커니즘이 공격자의 응답시간과 Pod 수에 어떤 영향을 주는지 분석했다.



(그림 3) Thread Sleep에 적용된 공격자들의 평균 응답 시간



(그림 4) Thread Sleep에 매커니즘 적용 후 Pod 개수

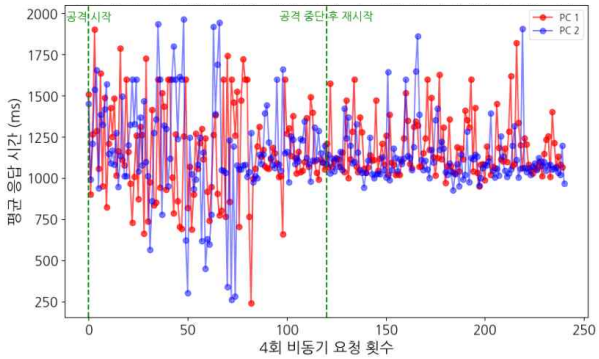
(그림 3)은 공격자가 공격을 진행 할수록 응답 시간이 감소하지 않고 방어 매커니즘 적용에 의해 응답 시간이 증가하는 결과를 보여준다. 이에 따라 서버는 공격자의 공격 중단을 유도할 수 있다.

(그림 4)는 Pod 수가 방어 매커니즘의 적용으로 최대 개수로 증가하지 않고, 낮은 개수로 적절하게 유지하는 것을 보여준다. 이는 공격자의 모든 요청을 즉시 처리하지 않아 서버 부하를 줄이게 되기 때문이다.

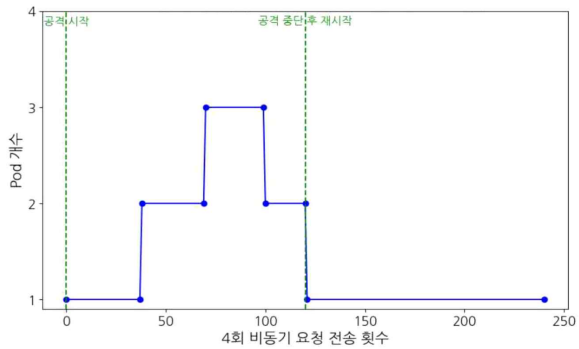
4.3 VM 방화벽 설정을 활용한 패킷 드랍

VM 방화벽 설정을 활용한 패킷 드랍 방어 매커니즘이 공격자의 응답시간과 Pod 수에 어떤 영향을 주는지 분석했다.

네트워크 패킷을 드랍하기 위해 VM 운영체제 리눅스의 iptables 명령어를 활용했다.



(그림 5) 패킷 드랍 방어가 적용된 공격자들의 평균 응답 시간



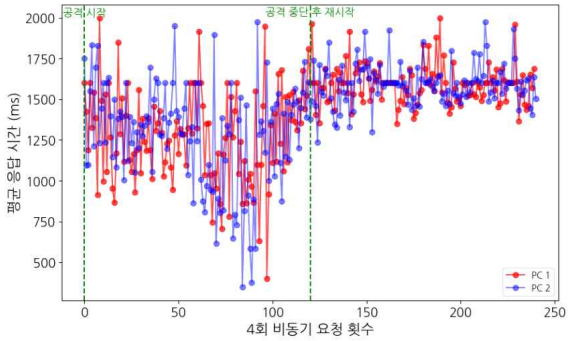
(그림 6) 패킷 드랍 매커니즘 적용 후 Pod 개수

(그림 5)은 공격자가 공격을 지속할 때 방어 매커니즘 적용으로 인해 응답 시간이 감소하지 않고, 증가한 후 큰 변화가 나타나지 않는 것을 보여준다. 이를 통해 서버는 공격자의 공격을 중단하도록 유도할 수 있다.

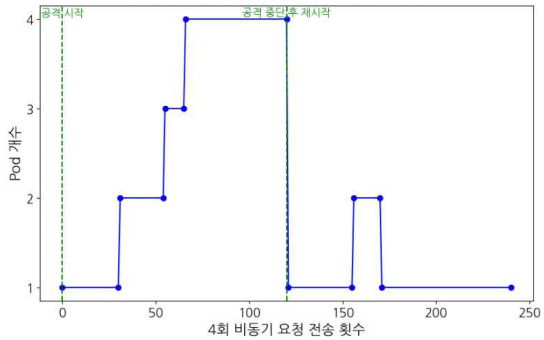
(그림 6)은 Pod 개수가 방어 매커니즘의 적용으로 최대치로 증가하지 않고, 낮은 수준으로 적절하게 유지하는 것을 보여준다. 이는 공격자의 패킷이 일정 시간 드랍되면서 요청이 바로 서버로 전달되지 않아 서버 부하가 줄어들기 때문이다.

4.4 더미 서버로의 확률적 리다이렉트

더미 서버로의 확률적 리다이렉트 방어 매커니즘이 공격자의 응답시간과 서버의 Pod 수에 어떤 영향을 주는지 분석했다.



(그림 7) 리다이렉트 방어가 적용된 공격자들의 평균 응답 시간



(그림 8) 리다이렉트 방어 적용 후 Pod 개수

(그림 7)은 공격자가 더미 서버로의 확률적 리다이렉트 방어 매커니즘에 적용되었을 때, 요청에 대한 응답시간 변화 그래프를 나타낸다. 공격자가 공격을 진행할수록 더미 서버로 리다이렉트 될 확률이 증가하게 된다. 이에 따라 더미 서버에 의해 공격자 요청에 대한 응답 시간이 증가하게 되고, 서버는 공격자의 공격 중단을 유도할 수 있다.

(그림 8)은 Pod 수가 방어 매커니즘의 적용으로 최대 개수로 증가하지 않고, 낮은 개수로 유지되는 것을 보여준다. 이는 공격자의 요청이 본 서버가 아닌 더미 서버에서 처리되기 때문에, 본 서버의 부하가 줄어들기 때문이다.

5. 결론

본 연구에서는 YoYo Attack 방어를 위한 방어 매커니즘을 제안한다. 방어 매커니즘으로는 단위 시간당 요청 횟수를 기반으로 Thread Sleep을 활용한 응답 시간 지연, VM 방화벽 설정을 활용한 패킷드랍, 그리고 더미 서버로의 확률적 리다이렉트를 활용한 응답 시간 지연 매커니즘이 있다. 제안한 매커니즘은 공격자의 응답 시간을 교란하여 응답 시간 감소 구간을 감지하지 못하게 함으로써 공격을 중단하도록 유도한다. 또한 방어 매커니즘 적용 후 동일 시간 내 요청 횟수가 감소하여 서버 부하가 줄어들어, Pod 개수를 방어 매커니즘 적용 전보다 적게 유지할 수 있다. 다만, 실험이 로컬 환경에서 진행되었기 때문에, 실제 서비스 환경에서의 추가 연구가 필요하다. 본 연구에서 제안한 방어 매커니즘은 YoYo Attack에 효과적으로 대응할 수 있음을 확인하였다. 이러한 방어 매커니즘을 통해 클라우드 환경에서의 자원 남용을 방지하고, 서버의 안정적인 운영을 지원할 수 있을 것으로 기대된다.

Acknowledgement

본 연구는 2024년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었으며 (2023-0-00041), 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터사업의 연구결과로 수행되었음 (IITP-2023-RS-2023-00259967).

6. 참고문헌

[1] M. Michalowski, “Cloud Computing Statistics,“ Spacelift, Mar. 27, 2024, https://spacelift.io/blog/cloud-computing-statistics

[2] D. Krook, “Cloud Migration Statistics,“ Auvik, Aug. 13, 2024, https://www.auvik.com/franklyit/blog/cloudmigration-statistics/

[3] 문가용, “클라우드 보안 현황,“ 보안뉴스, Aug. 8, 2024. https://m.boannews.com/html/detail.html?idx=131924

[4] H. Rohit, A. Kumar, and S. Singh, “Defense strategies for the Yo-Yo attack,“ in IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, Atlanta, GA, USA, May 2017, pp. 1-9. DOI: 10.1109/INFOCOM.2017.8057010.

[5] “Amazon EC2 T3 인스턴스,“ Amazon Web Services, https://aws.amazon.com/ko/ec2/instance-types/t3/