클라우드 컴퓨팅
공격 및 방어 플랫폼 개발

부산대학교

정보컴퓨터 공학부

지도교수 김 태 운

팀 명 DDALPI

팀 원 201513137 이강빈
202155599 장진영
202055505 강수민

목차

1 자문의견서 반영사항

본 과제는 클라우드 사용이 확대됨에 따라 다양한 공격 유형, 특히 YoYo Attack 에 대비하기 위한 방어 메커니즘을 연구하는 것이다. 초기 착수 보고서에 공격과 방어 메커니즘에 대한 구체적인 설명이 부족하였다. 이에 보완하고자 공격자와 방어자가 모두에게 중요한 지표인 HTTP 응답 시간을 중심으로 공격자와 방어자의 메커니즘을 구체적인 설명 및 구현 계획을 추가한다. 그리고 YoYo Attack 공격에 대한 구체적인 사례가 없지만 이를 최대한 구체적으로 설명함으로써 대체하기로 한다. 또한 클라우드 서비스 환경에서 YoYo Attack 을 실험적으로 검증하고, 이에 따른 방어 메커니즘을 제시한다.

2 요구조건 및 제약사항

착수보고서에서 중간보고서 작성 시기까지 추가적으로 수정된 요구조건들은 없다. 하지만 실험 수행 과정에서 Network Layer 에 대한 공격과 방어가 효율적이지 않다는 것을 확인했다. 그래서 Network Layer 에 대한 공격과 방어는 제외하도록 한다.

또한 어떤 도구를 이용하여 공격을 할 것이며, 추후 공격 플랫폼이 될 RPi Cluster 를 어떻게 구성할지 등에 대한 명시가 요구되며, On Premise 방어자 환경 구성에 대한 구체적인 설명을 추가할 것이다.

클라우드 서비스에서 실제 공격을 실행하고 방어하는 것에 기술적 및 금전적 및 윤리적 제약이 존재하기에, 우선 On premise 환경을 조성하고, local 환경에서 공격과 방어 메커니즘을 실행할 것이다. 또한 구현한 것과 클라우드 서비스 정책 및 제약 사항들을 추가적으로 고려하여 작성 할 것이다.

3 설계 상세화 및 변경내역

1. 방어 메커니즘이 도입 된 Server 에서 정상 이용자가 어느 정도의 피해를 입을 수 있는지 구체화 할 것이다.
2. 방어자는 방어 시스템을 사용함으로써 리소스의 손익이 나는지 구체화 할 것이다.
3. 공격자 또한 YoYo Attack 메커니즘에 따라서 리소스의 손익을 구체화 할 것이다.

4 보고시점 과제수행 및 중간결과

4.1 공격자

- 공격자 주요 지표 : HTTP 요청에 대한 응답시간

➔ 공격자 측에서 공격 성공 여부 및 대상 시스템의 상태 유추에 사용 할 수 있는 지표 : **응답시간 증감**

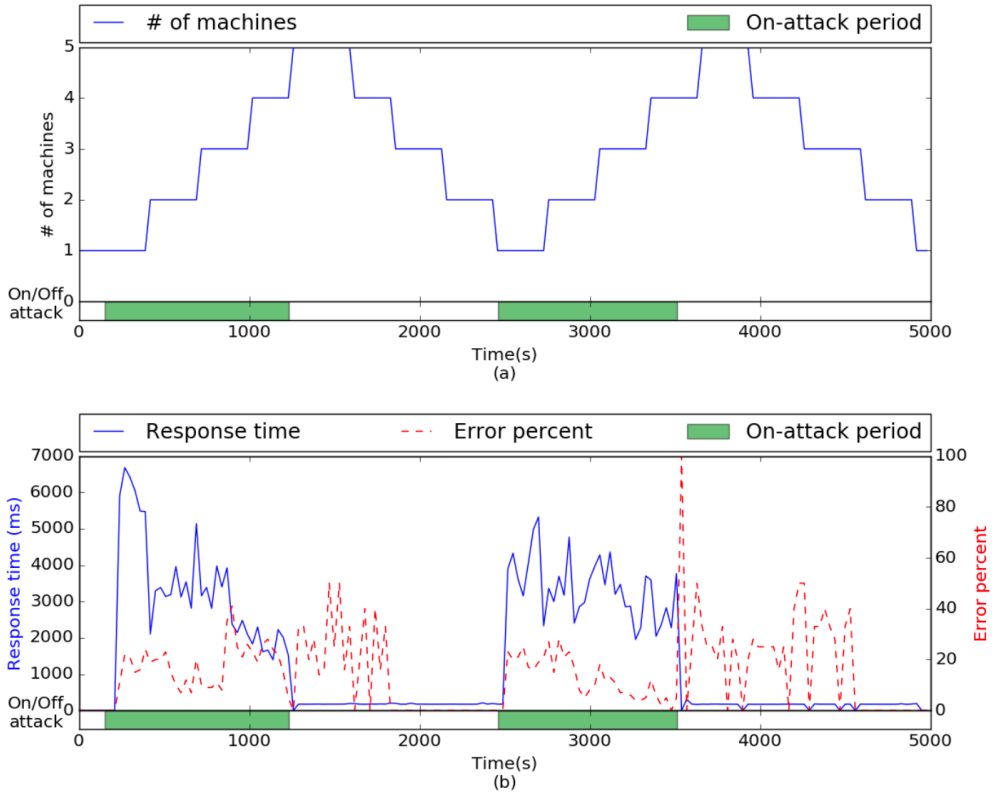


Fig.1 YoYo Attack on system with discrete scale policy[i]

- 공격자 입장에서의 응답시간 변화 요인

- Scale Out 이 일어나면 Pod 수 증가하며, 응답 시간 점점 감소한다.

- Scale Out 이 최대로 일어나면, 응답시간 변화 없거나 점점 증가한다.

- Scale In 이 일어나면 Pod 수 감소하며, 응답 시간이 점점 증가한다.

– 공격 메커니즘

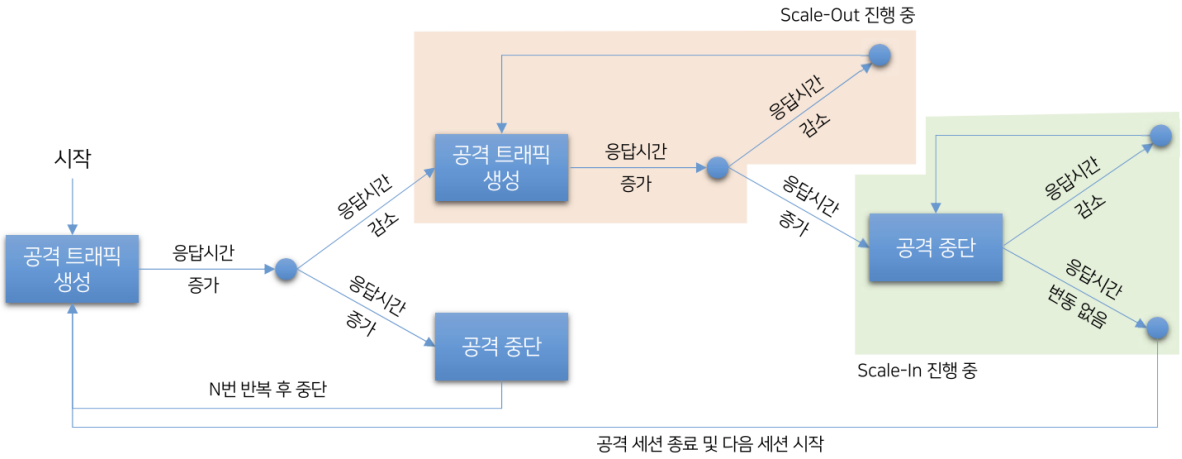


Fig 2. 공격자 입장에서 공격 메커니즘 개요

– 응답시간 증가, 감소 경향성 파악 방법

- HTTP 요청을 n 회 비동기로 전송한다.

- n 회 요청에 대한 평균 응답 시간을 기록한다.

- 위 과정을 m 회 반복한다. (n x m 회 HTTP 요청 전송)

- 평균 응답 시간의 경향성을 알기 위해 Polynomial Regression 을 활용하여 다항 함수를 구한다.

- 피공격자의 현재 네트워크 상황을 추정하기 위해, 위에서 구한 다항 함수를 미분하여 기울기 α 값을 구한다.

  – (α < 0) :

    음의 기울기 값을 갖는다는 것은 응답시간이 감소하는 경향성을을 나타낸다. 응답시간이 감소하므로 , Scale Out 이 진행 중이라 판단하여 공격을 지속한다.

  – (α >= 0) :

    기울기 값이 0 이라는 것은 응답시간이 변화가 없다는 것을 나타낸다. 양의 기울기 값을 갖는다는 것은 응답시간이 증가하는 경향성을 나타낸다. 응답시간의 변화가 없거나 증가하므로, Scale Out 이 완료되었다고 볼 수 있거나, Auto Scaling 이 일어나지 않는 Server 로 판단할 수 있다. 따라서 공격을 중단한다.

4.2 방어자

– 방어자의 방어수단 : HTTP 요청에 대한 응답시간 교란

➔ 방어자 측에서 공격 성공 여부를 교란하기 위해서 응답시간을 **고의적으로 조작함**

응답 시간에 혼동을 주는 다양한 방법들이 있으며 우리가 실험에서 채택한 방법들은 아래와 같으며, 추후 구체적으로 실험 결과와 같이 상술 하도록 한다.

3. CPU Sleep 을 활용한 응답 시간 증가
4. 특정 IP 의 네트워크 패킷 드랍
5. 리눅스 TC 를 활용한 응답 시간 증가

– 공격자 의심 IP 색출 방법

- Server 에 요청을 보내는 사용자 IP Log 기록을 활용한다.

- Log 기록을 활용하여 머신 러닝 Outlier Detection 활용한다. 이 방법을 사용하게 될 시 아래 실험 상황 에서와 같이 비정상 사용자 요청 횟수 기준을 정의 할 필요가 없어진다.

- 정상 사용자와 비정상 사용자의 HTTP 요청 횟수 기준을 정의한다.

- 위에서 정의된 횟수보다 많은 요청을 보낸다면 공격자 의심 IP 로 등록한다.

– 방어자 주의점

- HTTP 요청을 보내는 사용자가 공격자인지 정상 사용자인지 정확하게 구분 할 수 없다.

- 공격자로 의심이 되는 IP 를 색출 하면서도, 정상 사용자일 수 있는 가능성을 절대 배제하면 안된다.

- 방어 메커니즘이 적용될 경우, 예상되는 공격자의 응답시간 그래프

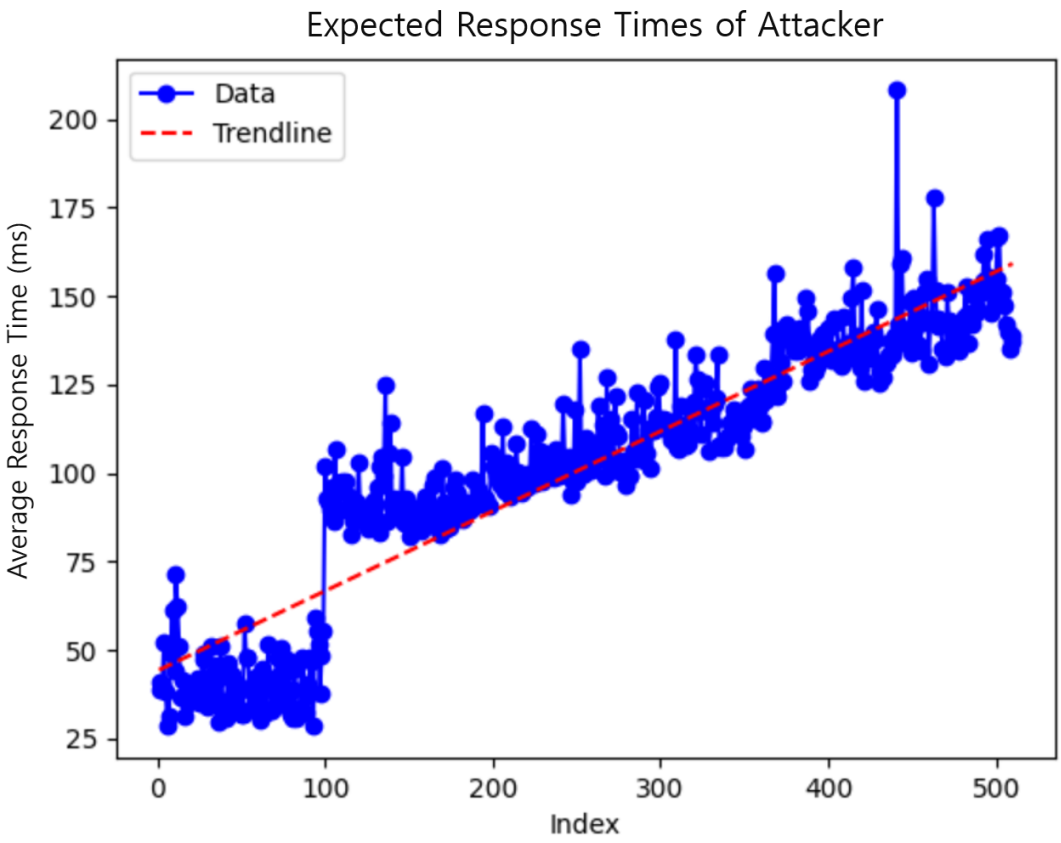


Fig 3. 10 회 단위의 비동기 HTTP 요청에 대한 평균 응답시간 그래프

지정한 요청 횟수가 넘어가게 되면 응답시간을 고의적으로 증가시켜서 공격자 입장에서 응답시간을 분석할 시 위와 같은 그래프가 나타나게 한다.

➔ 공격자는 응답시간 증가로 인해 Scale Out 이 더이상 안된다고 판단하고 공격을 중단하게 된다.

➔ 방어자는 공격자의 **공격 종료를 유도할 수 있다.**

- 방어자가 사용할 수 있는 메커니즘

1 단순 IP 차단

특정 IP 주소나 IP 범위에서 오는 트래픽을 방화벽, 네트워크 장비 혹은 서버 내에서 설정하여 차단하는 방법이다.

2 CPU Sleep 을 활용한 방어

2.1 특정 조건에 따른 Server CPU Sleep

서버의 CPU 를 특정 조건에 따라 일시적으로 Sleep 상태로 전환하여 시스템 자원을 보호 하는 방법이다.

2.2 공격자 의심 IP 지정 CPU Sleep

특정 IP 주소나 공격자 의심 사용자가 서버에 접속할 때, 그 요청에 대해 서버 CPU 를 Sleep 상태로 전환한다. 이로 인해 공격자 의심 사용자가 실제 응답을 받지 못하거나 매우 느리게 응답을 받게 되어 응답시간을 효과적으로 교란할 수 있다. 공격 효과를 감소시킬 수 있다.

3 Reverse Proxy Load Balancing 을 활용한 네트워크 패킷 드랍 응용

Load Balancing 와 Dummy Server 를 활용한 방어 기법이다.

4 VM 방화벽 설정을 활용한 네트워크 패킷 드랍

가상 머신(VM)의 방화벽 설정을 통해 특정 조건에 맞는 네트워크 패킷을 필터링 할 수 있다.

5 OpenWRT 를 통한 중개 방어

OpenWRT 를 라우터에 설치하여 네트워크 트래픽을 중계하는 과정에서 방어 기능을 추가한다. 예를 들어, 패킷 필터링, QoS 조정, 특정 IP 차단 등을 통해 악성 트래픽을 효과적으로 관리하고 네트워크 보안을 강화할 수 있다.

4.3 실험결과

- 실험수행

가상 머신 환경

- 가상화 소프트웨어: Oracle VM VirtualBox 7.0.18

  가상머신(VM) 구성

  ○ VM 개수: 1 개

  ○ VM 사양

  CPU: 4 개

  메모리: 11237MB

  디스크 크기: 25GB

  설치된 OS: Ubuntu 24.04 LTS

  ○ 네트워크 설정: 어뎁터에 브릿지

  호스트 머신의 네트워크 어댑터로 Intel(R) Wi-Fi 6 AX201 160MHz 를 사용

- 쿠버네티스 : Minikube 사용

  쿠버네티스 버전: v1.30.0

- HPA(Horizontal pod Auto Scaling) 설정

  스케일링 대상 리소스 종류: Deployment

  스케일링 기준

  최소 파드 수: 1

  최대 파드 수: 10

  목표 메트릭 값:

  메트릭 종류: CPU 사용률

  타입: Utilization

  평균 사용률 목표치: 30%

- 실험 상황 가정

- 서비스 모니터링 상 하나의 IP 로 1000 회 이상의 요청이 들어오는 경우는 거의 없다.
- 특정 IP 에서 1000 회 이상의 요청이 들어오면 해당 IP 는 공격자로 의심된다.
- 응답 시간이 가장 긴 HTTP 요청이 Auto Scaling 일으키기 가장 좋은 요청으로 가정한다.

- 공격자

- 공격 방법

- YoYo Attack 은 Application Layer 에 HTTP 요청을 통해 진행한다.
- 실행 할 수 있는 HTTP 요청을 파악한다.
  - HTML 버튼, HTML Form 등
- 실행 할 수 있는 HTTP 요청들을 각각 n 회 반복하여 평균 응답 시간을 측정한다.
- 평균 응답 시간이 가장 긴 API 에 다량의 HTTP 요청을 실행한다.

실험 환경에서 Auto Scaling 에 따른 응답 시간 확인

- Scale Out 이 최대로 일어날 때 까지 응답 시간이 점차 감소한다.
- 40~50 사이에 최대 Pod 수 도달 이후 응답시간이 증가 or 변화 없다.

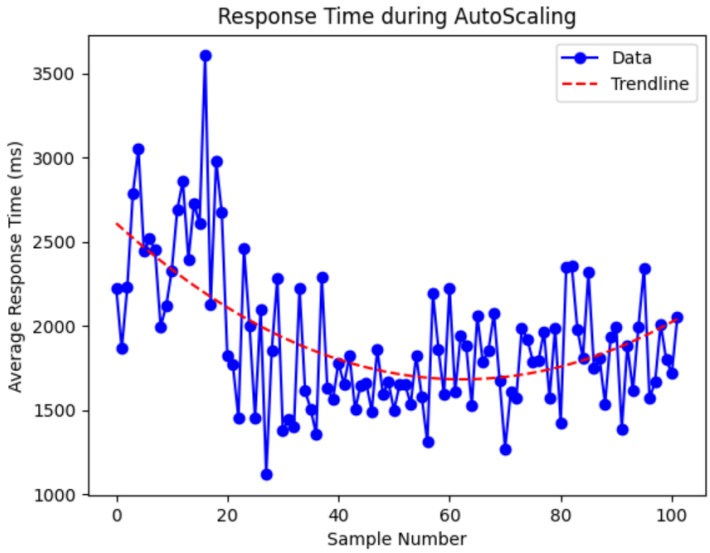


Fig 4. Auto Scaling 을 사용하는 서버의 HTTP 요청에 따른 응답시간 그래프

- 실험 환경에서 특정 Pod 개수에 따른 응답시간 확인

아래의 표는 Pod 의 개수 별 실제 얻어진 응답 시간 데이터로

회 당 HTTP 요청 10 회 비동기 통신 응답 시간 측정 (30 회 반복)

# of Pod : 1

(ms)	시도 1	시도 2	시도 3
1회	1972.0	2642.7	2177.7
2회	2104.8	2072.4	1954.6
3회	1790.8	1967.8	2169.7
4회	1840.0	1907.3	2264.5
5회	1708.3	2090.5	1982.2
6회	1689.5	1983.1	2101.9
7회	2005.4	2534.8	2126.9
8회	1706.7	2108.4	1874.4
9회	3050.4	1927.0	2008.7
10회	1709.1	2074.0	2012.6
11회	2021.8	1788.3	2074.7
12회	1894.1	2073.4	2005.0
13회	2015.1	2036.4	2071.9
14회	2519.1	1993.3	2192.7
15회	1877.3	1839.9	2053.1
16회	1857.0	2178.1	2147.0
17회	1769.8	1779.6	1863.3
18회	1986.1	2046.2	2393.9
19회	2802.2	1881.9	1835.4
20회	2540.2	1942.3	1720.6
21회	1817.7	1955.7	2208.1
22회	1774.2	2370.6	1830.8
23회	1878.0	1807.1	2024.4
24회	1966.8	2147.5	1751.4
25회	2265.9	2080.5	2114.2
26회	1745.1	1971.6	1952.0
27회	1801.6	1996.6	1811.9
28회	1822.5	2041.7	1774.7
29회	2062.0	2054.1	1853.2
30회	1773.9	1896.0	2351.3
평균	1992	2039	2023

# of Pod : 2

(ms)	시도 1	시도 2	시도 3
1회	3763.0	1334.9	1484.4
2회	1493.4	1633.9	1537.2
3회	1001.5	1214.4	1601.0
4회	1313.3	1556.5	1462.8
5회	1734.6	1551.7	1569.0
6회	1668.4	1583.4	1337.3
7회	1302.0	1813.2	2227.2
8회	1395.8	1795.3	1276.7
9회	1258.7	1372.5	3249.1
10회	1534.0	1719.2	1549.5
11회	1064.1	1531.2	1500.0
12회	2097.2	2219.5	1609.9
13회	1211.9	1349.3	1295.8
14회	1270.6	1834.2	1927.4
15회	1385.3	1522.4	1452.6
16회	1249.1	1379.4	1593.5
17회	1238.5	1574.9	1328.5
18회	1324.4	1575.6	1319.7
19회	1487.3	1262.2	1800.0
20회	1408.4	1285.3	1349.3
21회	1679.8	1572.9	1608.9
22회	1568.9	1315.8	1448.2
23회	1740.5	2039.0	1351.2
24회	1499.7	1323.6	1226.1
25회	1408.0	1495.6	1698.0
26회	1626.3	1283.0	1328.9
27회	1195.2	1416.4	1411.7
28회	1286.8	1717.8	1609.7
29회	1557.0	1520.7	1445.0
30회	1309.3	1475.8	1549.4
평균	1502	1542	1571

# of Pod : 3

(ms)	시도 1	시도 2	시도 3
1회	2607.3	2034.7	1577.4
2회	999.2	1655.8	1072.1
3회	930.4	1228.7	1529.2
4회	1650.8	1138.1	1135.8
5회	928.1	1335.2	1303.7
6회	1060.3	1368.7	1634.1
7회	1350.6	1281.2	1481.8
8회	972.9	1358.3	1277.1
9회	1155.0	1300.8	1417.9
10회	1383.2	1358.1	1117.1
11회	1131.5	1341.8	1911.7
12회	1351.1	1145.0	1090.3
13회	1042.0	1622.0	1282.7
14회	1357.6	1502.3	1369.7
15회	1263.3	1345.5	1514.9
16회	1573.5	1520.4	1202.7
17회	928.7	1888.1	1245.2
18회	1306.6	1070.3	1804.8
19회	2325.5	1403.6	1292.1
20회	1211.3	1396.0	1481.7
21회	1191.5	1214.5	1285.3
22회	1494.8	1878.5	1573.2
23회	1284.1	1076.5	1158.4
24회	1398.5	1799.9	1234.0
25회	1295.0	1847.2	1712.0
26회	1498.7	1170.1	1262.1
27회	1589.0	1757.4	1498.0
28회	1427.3	1189.9	1335.8
29회	1076.7	1709.4	1824.1
30회	1732.9	926.1	1194.5
평균	1350	1428	1393

➔ 결국 Pod 의 개수가 증가 할 수록 평균 응답 시간이 짧아지는 것을 공격자 입장에서 확인할 수 있다.

- 방어자

- 방어 메커니즘

1 공격자 의심 IP 차단

개념

- 공격자로 의심 되는 IP의 네트워크 패킷은 Server 에서 처리하지 않는다.

구현 방법

- VM 방화벽 설정을 활용하여 해당 IP 를 자체적으로 차단한다.
- Application Server 미들웨어에서 IP 블랙리스트를 만들어 요청을 처리하지 않는다.

한계

- 공격자로 의심한 IP 가 정상 사용자일 수 있는 가능성이 있어 **적용이 어렵다.**

2 CPU Sleep 을 활용한 방어(Application Server Middle ware CPU sleep)

2.1 IP 관계없이 조건에 따른 CPU Sleep

구현 방법

현재 Server 상황이 특정 조건에 부합하면 요청 처리 전 CPU sleep 을 한다.

사용될 수 있는 조건

1. 현재 Pod 수
2. 특정 IP 요청 수

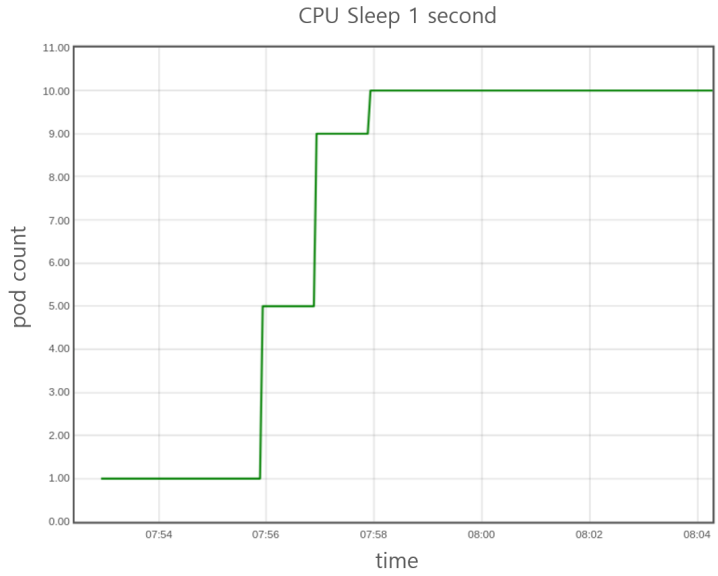
기대되는 효과

➔ CPU Sleep 에 의해 CPU 사용률이 감소하고 Scale In 을 유도해 Pod 수를 감소한다.

실험 결과

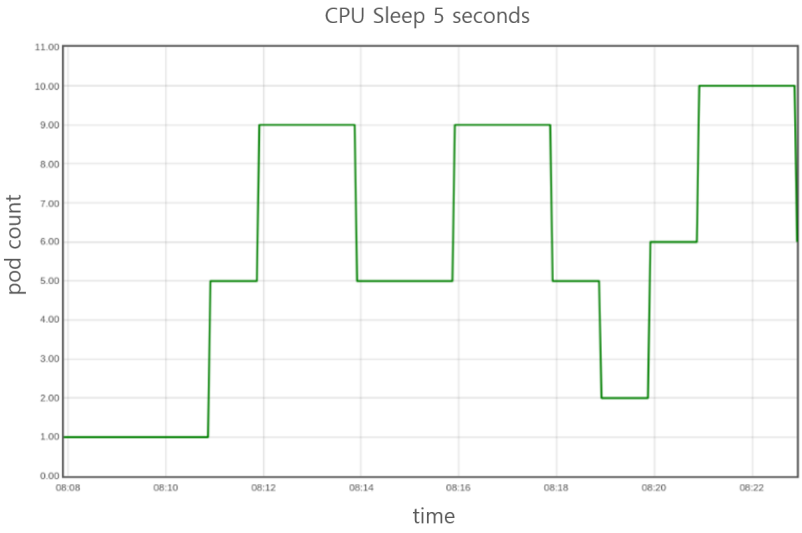
CPU Sleep 조건 : 현재 가용 가능한 Pod 수가 최대 Pod 수의 절반일때 CPU Sleep 한다.

- 1 초 CPU Sleep



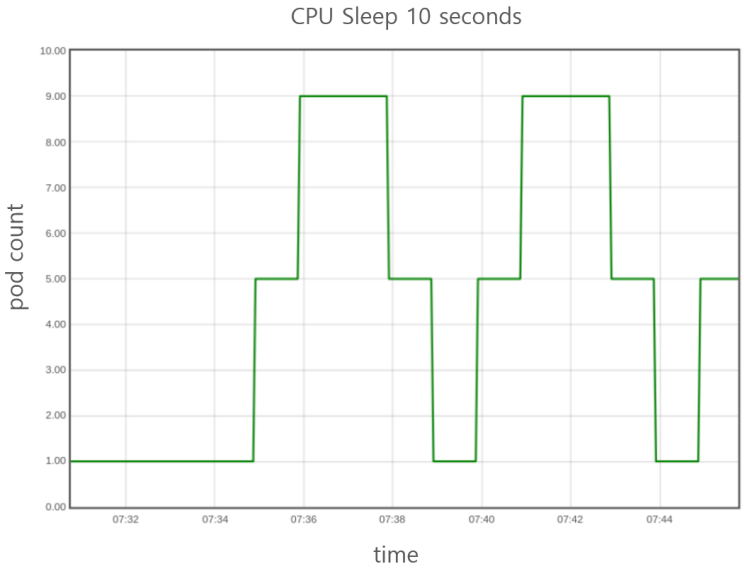
결과 : Scale In 이 일어나지 않으므로 Pod 수가 줄 지 않는다.

- 5 초 CPU Sleep



결과 : Scale In 이 일어나지만 곧 바로 Scale Out 이 일어난다.

- 10 초 CPU Sleep



결과 : 5 초 CPU Sleep 의 결과와 유사하다.

Scale In 이 일어나지만 곧 바로 Scale Out 이 일어난다.

결론 및 한계

- 적어도 5 초 이상의 CPU Sleep 을 활용 해야지 Scale In 이 일어난다.
- 하지만 5 초 동안 쌓인 HTTP 요청을 처리하기 위해 CPU 사용률이 증가해 곧 바로 Scale Out 이 일어난다.
- 특정 상황이 만족하면 모든 사용자의 요청을 처리하기 전에 CPU Sleep 이 발생하므로 사용자의 요청을 효율적으로 처리하지 못하며, 사용자의 서비스 품질이 낮아진다.
- 5 초 이상의 CPU Sleep 은 멀티 프로세싱 처리에 부적합하다.
- 결과적으로 **YoYo Attack 방어에 효율적이지 않다.**

2.2 공격자 의심 IP 요청 한정 CPU Sleep

구현 방법

- 특정 IP 요청 수가 1000 회 이상이 되면 공격자 IP 로 규정한다.
- 해당 IP 요청을 미들웨어에서 CPU Sleep 을 활용하여 응답시간을 교란한다.
- 요청 횟수에 비례하여 CPU Sleep 시간을 증가한다.

기대되는 효과

- 정상 사용자의 요청은 바로 처리되어 정상 사용자의 서비스 품질의 저하를 막을 수 있다.
- 요청 횟수에 비례하여 CPU Sleep 시간을 증가 시킴으로써, 공격자가 Scale Out 이 끝났다고 생각하게 하여,공격 중단을 판단하게 한다.
- CPU Sleep 을 second 단위로 건다면 Scale In 을 유도하여 적절한 Pod 수를 유지할 수 있다.

실험 결과

1. CPU Sleep 시간을 milli Second 단위로 설정

- 특정 IP 가 1000 회 이상의 요청 시 기본 CPU Sleep 시간 : 1000ms
- 추가적인 1000 회 요청 당 500ms 를 추가적으로 CPU Sleep

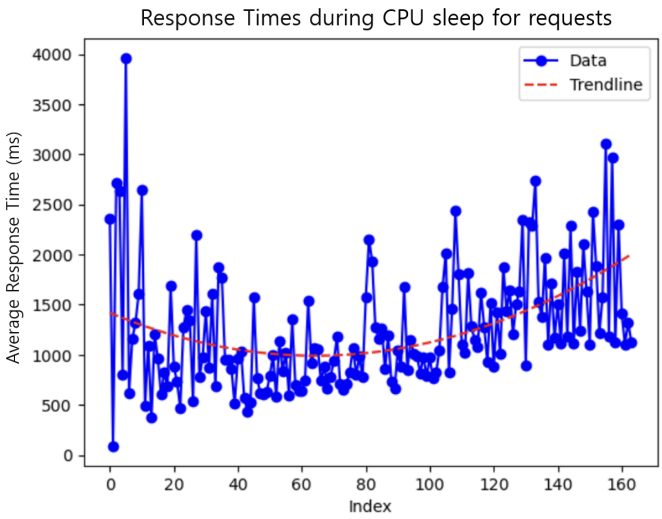


Fig 5. 공격자 입장 비동기 HTTP 요청에 따른 평균 응답시간

특정 횟수(정의된 요청 횟수)가 넘어가면 응답시간이 교란된다.

결론

- 공격자 요청에 비례해 응답시간을 증가 시켜 공격자가 Scale Out 이 전부 일어났다고 판단하게 하여, 자연스럽게 공격을 중단하도록 유도한다.
- IP 별로 CPU Sleep 시간을 다르게 하여 유동적으로 대응할 수 있다.

2. CPU Sleep 시간을 Second 단위로 설정

- 공격자 요청에 대한 응답에 초 단위의 CPU Sleep 을 통해 CPU 사용률을 과하게 높이지 않음으로써 Scale Out 이 일어나지 않게 한다.
- CPU Sleep 을 second 단위로 적용하는 것은 멀티프로세싱 처리에 **부적합**하다. 정상 사용자가 방어 프로그램에 적용되지 않아도 공격자의 요청에 의한 CPU Sleep 으로 **서비스 품질이 낮아질 가능성이 있다.**
- 정상 사용자를 공격자로 의심한다면 서비스 품질이 매우 저하된다.

3 Reverse Proxy Load Balaning 을 이용한 패킷드랍 응용

사전 정의

- 정상 Server : 일반 사용자가 사용하는 정상 Server
- Fake Server : 응답 시간 교란을 위해 사용하는 Dummy Server

방법

- Reverse Proxy 를 이용해 Load Balancing 을 적용한다.
- 정상 Server 와 Fake Server 를 확률적으로 Load Balacncing 한다.
  (10% 확률로 Load Balancing 을 하게 되면 100 개의 요청 중 10 개가 Fake Server 에 가게 된다.)
- Fake Server 에서는 CPU Sleep 과 HTTP Error 응답 등을 통해 네트워크 패킷이 드랍된 것 처럼 하여 공격자에게 교란을 준다.
- 이때 요청 횟수에 비례하여 CPU Sleep 시간을 증가한다.

기대 효과

- 정상 사용자는 Server 에 요청을 많이 보내지 않으므로 Fake Server 로 갈 확률이 낮다.
- 만약 정상 사용자가 Fake Server 로 가더라 하더라도 새로고침 등을 활용한 재요청 시 정상 Server 로 갈 확률이 매우 높다.
- 공격자는 정상 사용자에 비해 많은 요청을 보내므로 Fake Server 에 갈 확률이 높아진다.
- 정상 사용자의 피해를 적게 하면서, 공격자의 응답 시간 교란 할 수 있다.

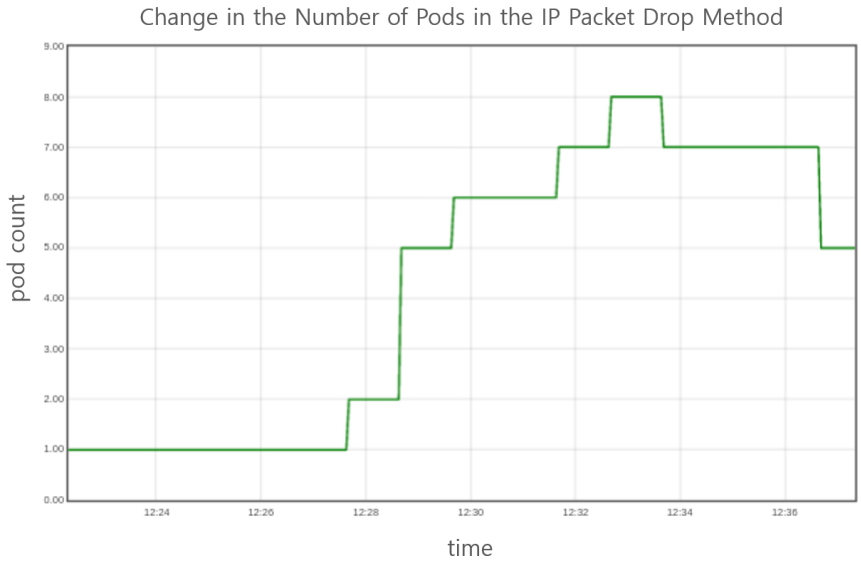
실제 실험 결과

실험 환경

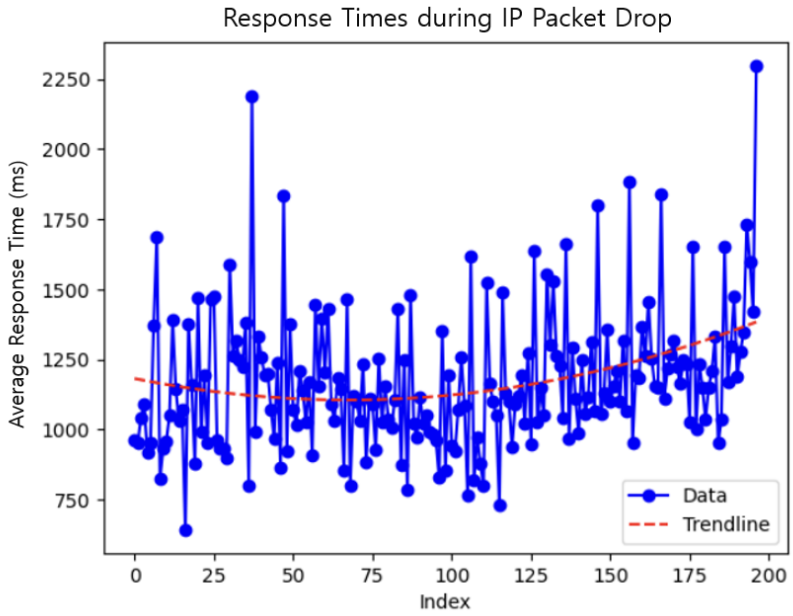
- 70% 확률로 정상 Server, 30% 확률로 Fake Server 로 요청 전달한다.
- Fake Server 에는 특정 IP 가 300 회 이상 요청 시 CPU Sleep 적용 기본 1000ms 적용한다.
- 100 회 당 500ms 추가 적용한다. (예를 들어 400 회 요청 시 1500 ms CPU Sleep)
    - 300 회가 조건의 이유
      30%의 확률로 요청이 Fake Server 로 전달된다. 정상 Server 에 1000 회 이상 요청을 보낼 때, 300 회 정도의 요청이 Fake Server 로 갈 것으로 예상된다.

결과

- 모든 요청이 CPU 부하를 주는 API 로 전달되지 않아 Scale Out 이 최대로 일어나지 않는다.



결과적으로 공격 측에서 응답시간이 점차 증가기에 공격을 중단한다.



결론

- Pod 수를 적절히 유지하면서 공격자의 공격 중단을 유도할 수 있는 두 가지 측면에서 **효율적인 방어 수단이다.**

4 VM 방화벽 설정을 통한 패킷 드랍

구현 방법

- 특정 IP 의 요청 수가 1000 회 이상이 되면 공격자 IP 로 의심한다.

1. 일정 확률로 공격자 의심 IP 를 VM 으로 전달한다. 또한 추가적으로 요청 횟수에 비례해서 VM 으로 전달될 확률을 증가시킨다.

2. VM 이 공격자 의심 IP 를 전달받으면 iptables 를 활용해 방화벽 설정을 바꿔 해당 IP 의 네트워크 패킷을 일정시간 드랍한다. 이때 공격자 의심 IP 의 전달 횟수에 비례하여 패킷 드랍 시간을 증가시킬 수 있다.

    - 위 두가지 방법을 통해서 결과적으로 패킷이 드랍되어 응답시간을 늘릴 수 있다.

기대 효과

- 요청 횟수에 따라 네트워크 패킷 드랍 시간이 증가하므로 정상 사용자는 해당 방어 시스템에 적용되지 않게 할 수 있다.

- VM 의 방화벽 설정을 통한 패킷 드랍이므로 Server 에 불필요한 요청이 전달되지 않도록 할 수 있다.

- 정상 사용자의 피해를 적게 하면서, 공격자의 응답 시간 교란 할 수 있다.

실제 실험 결과

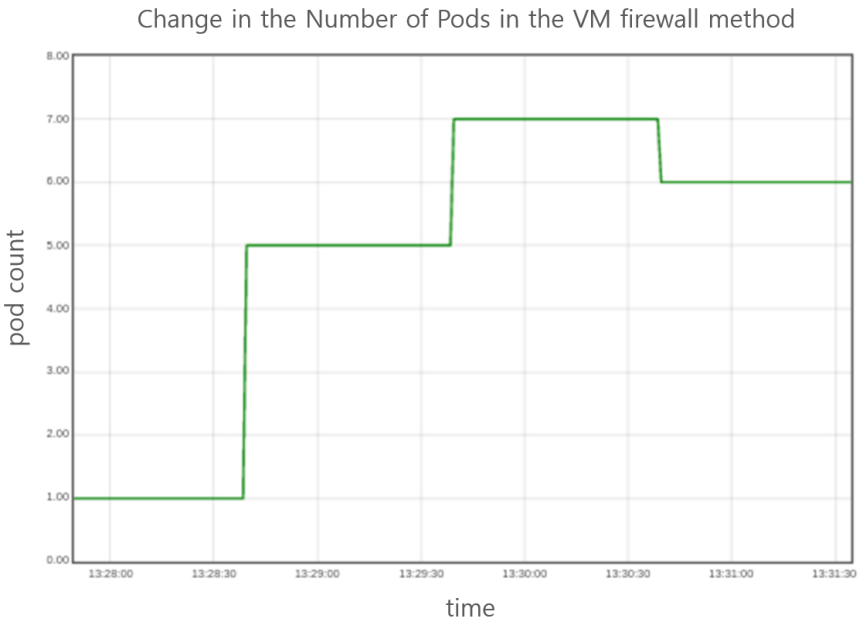
실험 환경

- 특정 조건 없이 30% 확률로 VM 에게 현재 요청 IP 전달한다.

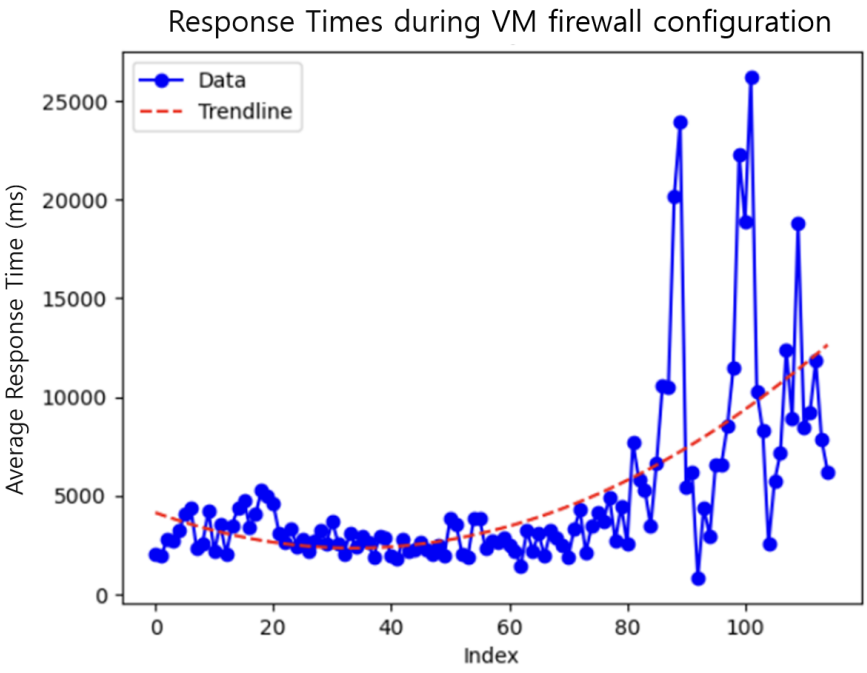
- 300 회 이상 요청 시, 1000ms 동안 패킷을 처리 하지 않는다.

- 100 회 당 500ms 추가 적용한다.(예시로 400 회 요청 시 1500ms 패킷 드랍)

    - 300 회가 조건의 이유
      30%의 확률로 VM 에 IP 를 전달한다.
      정상 Server 에 1000 회 이상 요청을 보낼 때, 300 회 정도 VM 에 IP 가 전달된다.



모든 요청이 CPU 부하를 주는 API 로 전달되지 않아 Scale Out 이 최대로 일어나지 않는다.



공격 측 입장에서 응답 시간이 점차 증가하여 공격을 중단한다.

결론

- Pod 수를 적절히 유지하면서, 공격자의 공격 중단을 유도할 수 있는 두 가지 측면에서 효율적인 방어 수단이다.

5 갱신된 과제 추진 계획 및 추가 계획

- 갱신된 과제 추진 계획

구분	작업일정																					
	5 월			6 월				7 월				8 월					9 월				10 월		
	3	4	5	1	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3
YoYo attack 스터디	■																						
VM 환경 YoYo attack 실습		■	■	■																			
VM 실습 환경 구축					■	■	■	■															
공격 알고리즘 구현								■	■	■													
방어 알고리즘 구현									■	■	■												
중간 보고서 작성											■	■	■	■									
OpenWRT 중개 방어													■	■	■								
RPi Clustering													■	■	■								
실 서비스 환경 구축														■	■	■							
UI/UX 개발														■	■	■							
모니터링 시스템 개발															■	■	■	■					
최종 시스템 구축															■	■	■	■					
디버깅 및 테스트																	■	■	■				
배포 테스트 및 배포																			■	■	■		
최종 보고서																					■	■	■

– 추가 계획

- OpenWRT 를 활용한 중개 방어
- Server 측 UI/UX 개발
- 공격자와 방어자
  - ■ 제어 및 모니터링 시스템
  - ■ 자원 사용량 측정 및 분석도구
- 공격자가 공격을 위해 얼마나 자원을 투입해야 하는지 또한 얼마나 투입했을 때 어떤 결과를 얻어내는지 측정
- 방어자가 공격을 막기 위해 얼마나 자원을 투입해야 하는지 또는 공격을 얼마나 효율적으로 방어하는지 측정
- 정상 사용자가 얼마나 피해 입을 수 있는지 측정

6 구성원별 진척도

이름	분류	세부 역할 분담
이강빈	Full Stack 개발	- Management & Monitoring Web Application 개발 - RPi Clustering (Attacker Platform) - OpenWRT (Defender Platform) - Front-End 개발 도구 선정

장진영	매커니즘 및 알고리즘 개발	- 공격 메커니즘 개발
		  - Application Layer 공격 방법
		  - 피공격자 상태 확인 메커니즘
		- 방어 메커니즘 개발
		  - 공격자 의심 IP 탐색 매커니즘 개발
		  - 응답시간 교란을 이용한 방어 매커니즘 개발
		- Back-End 개발 도구 선정

강수민	인프라	- 실험 가상환경 구성 및 결과 가공
		  - Docker, Minikube 를 통한 실험환경 구성 및 관리
		  - Prometheus, Grafana 을 통해 상태확인
		- Back-End 개발 도구 선정

공통	자료 조사	- YoYo Attack 공격 방식, 공격 탐지 방법 조사
	보고서 작성	- 중간 보고서, 최종 보고서
	발표 및 시연	- 발표자, 시연자, Q&A 담당자

7 참고문헌

[i] Fig 7. Bremler-Barr, A., Brosh, E. and Sides, M. (2017). DDoS attack on cloud auto-scaling mechanisms. [online] IEEE Xplore. doi:https://doi.org/10.1109/INFOCOM.2017.8057010.