Tandem Attack: DDoS Attack on Microservices Auto-scaling Mechanisms

Prof. Anat Bremler-Barr (Tel-Aviv University)  Michael Czeizler (Reichman University)

Abstract

In this work we demonstrate how an attacker can exploit the tandem behavior of microservices with different auto-scaling mechanisms to create an attack we denote as the Tandem Attack. We demonstrate the attack on a typical Serverless architecture and show that using managed infrastructure does not liberate from strict planning of scaling definitions.

Introduction

Auto-scaling

- A primary ability of cloud computing which allows systems to adapt to fluctuating traffic loads.
- Dynamically increasing (scale-up) and decreasing (scale-down) the number of resources used.

Microservices Architecture

- Software systems are implemented by loosely-coupled services
- Each service is responsible for specific task and defined with separate scaling properties.

Attack Methodology

- Microservices with separate auto-scaling mechanisms work in tandem to process traffic, they can overload each other.
- Overload results:
  - Throttling (Denial of service - DoS)
  - Over-provisioning of resources (Economic Denial of Sustainability - EDoS).

Experiment Architecture

![Diagram of Experiment Architecture]

Evaluation

The Attacker's Perspective

- Very easy to overload web components that do not require auth/login
- Automated solutions for handling captcha and login exist
- Real time systems are more prone (financial, security, medical)
- In async. systems likely to see performance degradation
- Can create Tandem attack only by following 2 metrics:
  - Success/Fail ratio of requests
  - RTT
- Prior knowledge on the system can assist in optimizing the attack

Mitigations

- Rate limiting - limit traffic as close as possible to the origin:
  - Easy & Effective
  - In our use case: Limit the Lambda to match the DB scaling capabilities
- Decoy by inserting noise to response time (when under attack)
- Randomizing the RTT reduces the attacker’s ability to evaluate the system's state
- Attacker using same resources to attack multiple targets might be exausted
- Retry comes with costs
  - Can compensate when services are not synced
  - Significant increase in latency (sometimes not acceptable) & costs
  - If attack is relitively strong might prolong the effect of the attack
- Developing better service control planes that can backpressure - relevant especially for cloud services

Conclusions

- Trade-off between cost & performance – No complete solution !
  - Over provisioning/reserve pools can compensate for DDOS up to a certain extent but with extra costs
  - In large systems the micro-service connectivity and dependency can become complex and hard to analyze
  - Serverless is not a solution for Tandem attack

Future Work

- Detection - collect real-time data from systems with complex micro-services dependency and implement Algorithm that identifies issues
- Mitigation - Algorithm that identifies tandem issues and resolves

Experiment Results

YoYo traffic pattern – Synthetic Example

AWS Experiment – Economic Damage

AWS Experiment – Performance Damage

<table>
<thead>
<tr>
<th>Experiment</th>
<th>DDOS</th>
<th>DDOS</th>
<th>DDOS</th>
<th>DDOS</th>
<th>DDOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDOS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DDOS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DDOS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DDOS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DDOS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Economic Damage verses Time

Performance Damage versus Time