# Automated REST API Endpoint Identification for Security Testing at Scale

Lei Ding, Azzedine Benameur, Jeffrey Jacob, Jay Chen

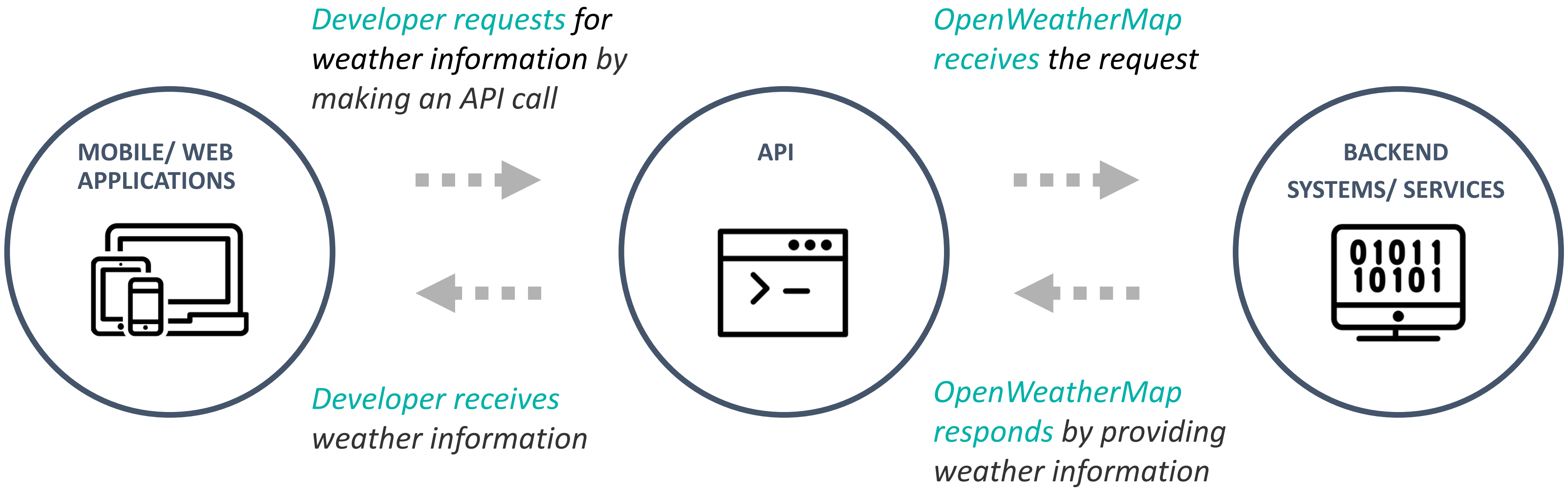
Accenture Labs, Security R&D

**Steve Pham**

Accenture Digital

# REST API Background

**API Use case: A developer wants to retrieve real-time weather information from OpenWeatherMap for his/her website**



*Developer requests for weather information by making an API call*

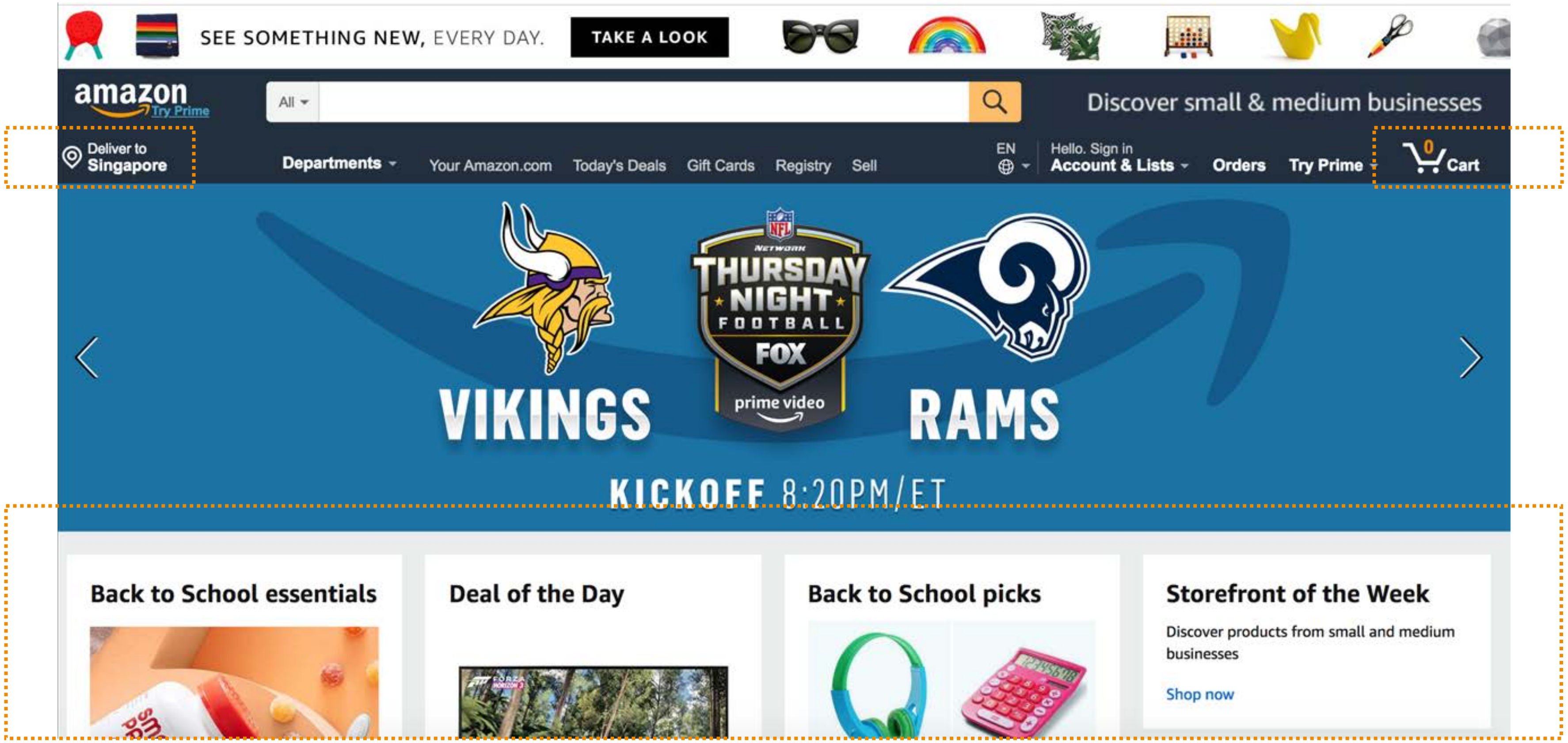
*OpenWeatherMap receives the request*

*OpenWeatherMap responds by providing weather information*

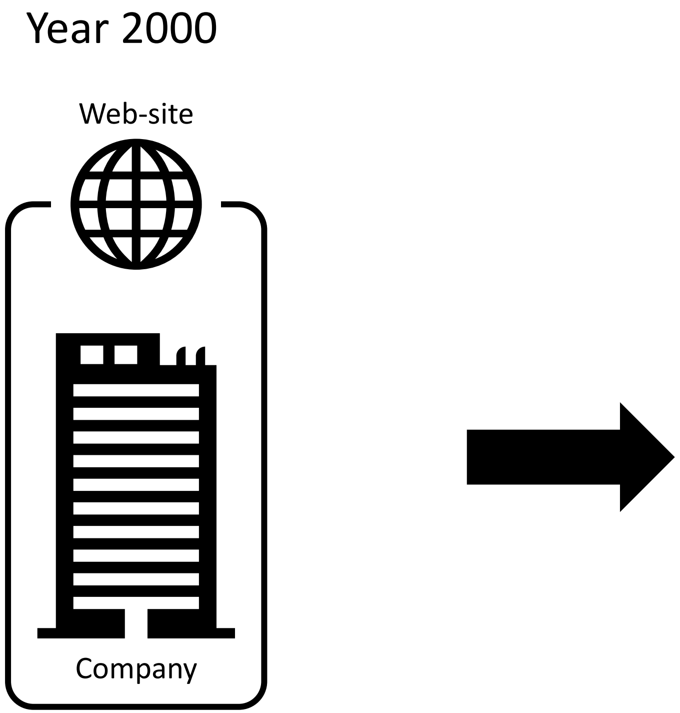
*Developer receives weather information*

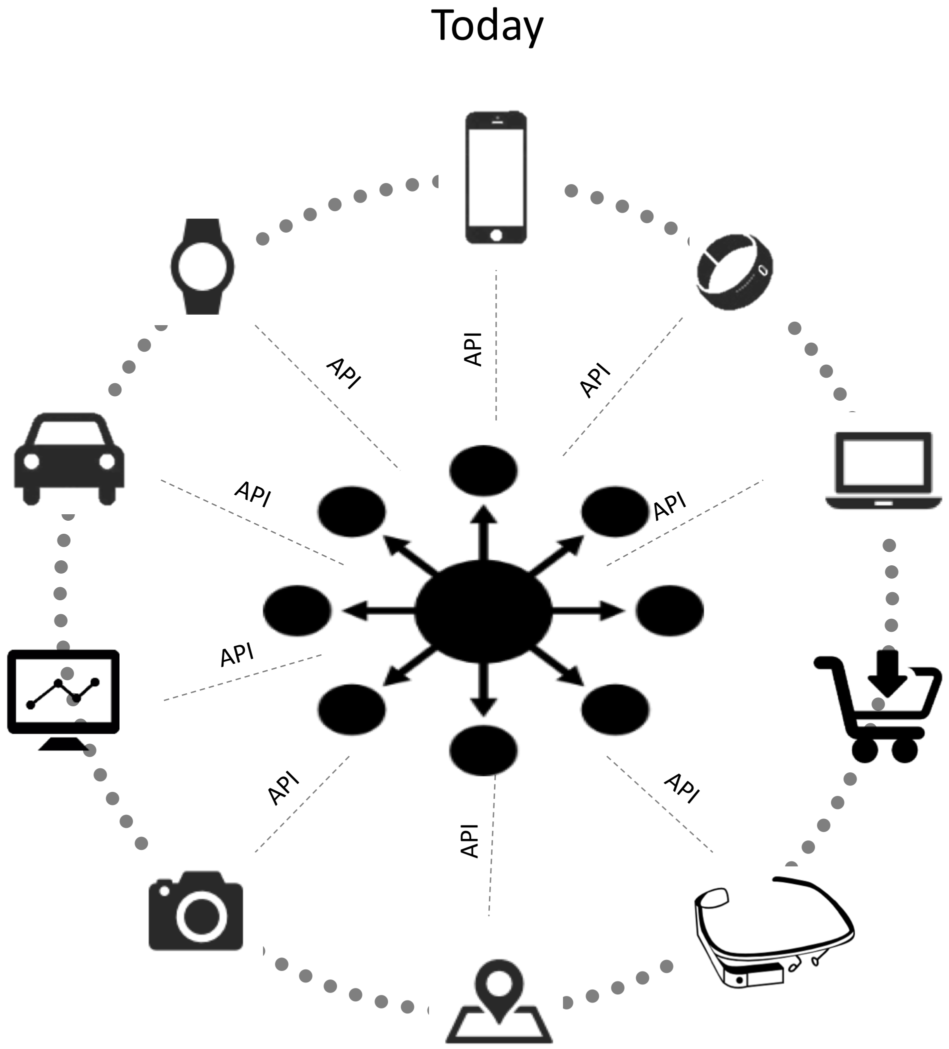
# REST API Background

**API Use case: Amazon website**



# Why do we need API?





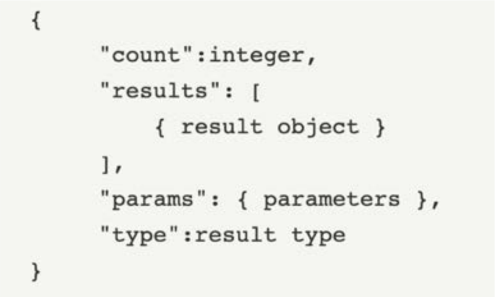
# What is an API Endpoint?

- API endpoint is the location from which APIs can access the server side resources
  - https://api.example.com/v1/media/recent?access_token=ACCESS_TOKEN
  - Above URL may support one of various different HTTP methods
    - GET: fetch information
    - POST: create objects
    - PUT: update objects
    - …
- We define an API endpoint as a combination of a path and a HTTP method

# API Endpoint Request and Response

- A sample request:
  - GET https://openapi.example.com/v1/listing/active?api_key=API_KEY

- Each API response is wrapped in a standard structure that holds the results
  - A sample response



```
{
      "count":integer,
      "results": [
          { result object }
      ],
      "params": { parameters },
      "type":result type
}
```

# What is the API Specification?

- Different formal descriptions
  - Open API Specification
  - RAML
  - WADL
  - API Blueprints
  - …
- Different API versions

# API Security Testing Challenges

- Identification of API endpoints largely depends on documentation

- Blind spots can be caused by
  - Lack of API specification
  - API specification discrepancy
  - Undocumented API endpoints
  - Complicated routing/orchestration logic

- Can we automatically identify API endpoint?

# Use ML to Accelerate Automation Process

- Reduce probing and testing time
  - Minimize API endpoint search space

- Learning to find URLs that may lead to different API endpoints
  - Analyze lexical and host-based features because they contain information about the URL that is straightforward to collect using automated crawling tools
  - The list of features is extensive, but not necessarily exhaustive
  - Character level comparison is slow

# Fuzz Algorithms

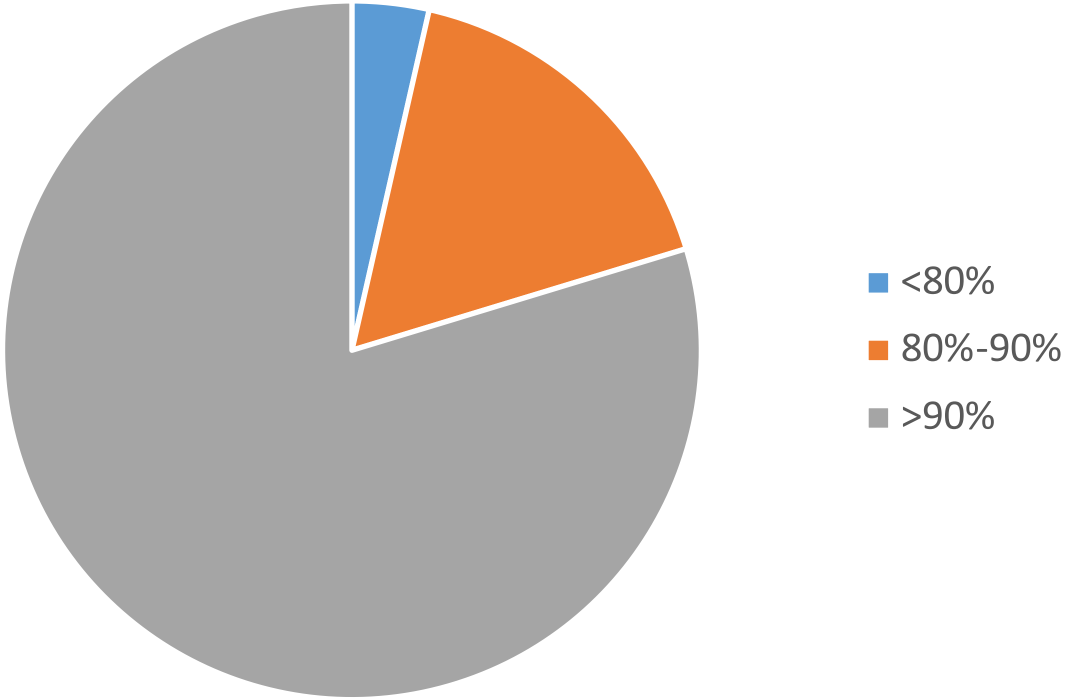
- Fuzz algorithm
  - Example:
    - "YANKEES", "NEW YORK YANKEES" : 60
    - "NEW YORK METS", "NEW YORK YANKEES”: 75

  - Challenge: the score of the “bad” match is higher than the “right” one

# Fuzz Algorithms

- Partial fuzz algorithm
  - Example:
    - “YANKEES", "NEW YORK YANKEES”: 100
    - "NEW YORK METS", "NEW YORK YANKEES”: 69
  - The score of the “good” match is higher than the “right” one
  - Use best “partial” when two strings are of noticeably different lengths
  - Key idea: if the shorter string is length m, and the longer string is length n, we’re basically interested in the score of the best matching length-m substring

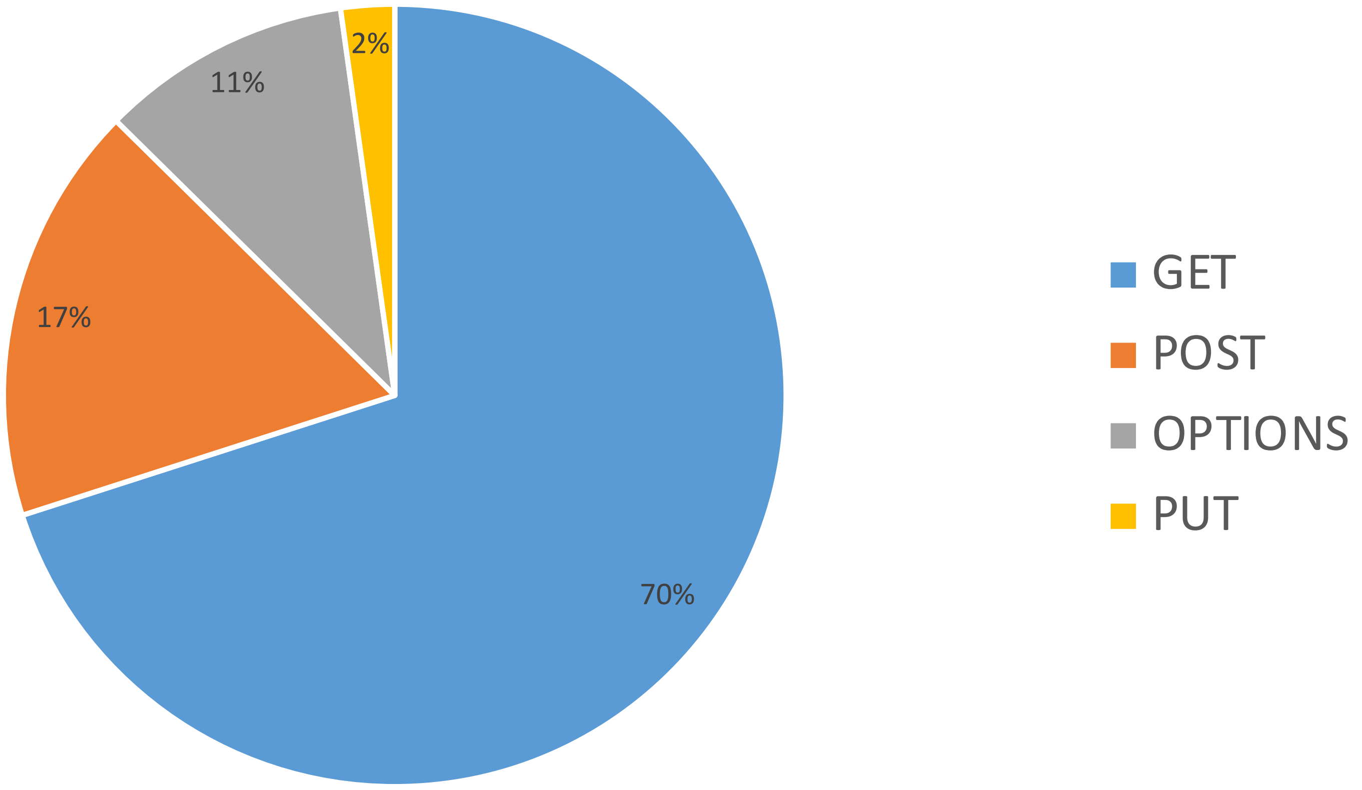
# Reduced Probing Times



For example: If we crawled 10000 URLs from a domain, our algorithm is able to reduce the search space to around 1000 URLs for API endpoints identification

Accuracy: 98.6% → Trade off between reduction constraint and accuracy

# Method Distribution (Based on 719 domains scan)



# Vulnerability Assessment

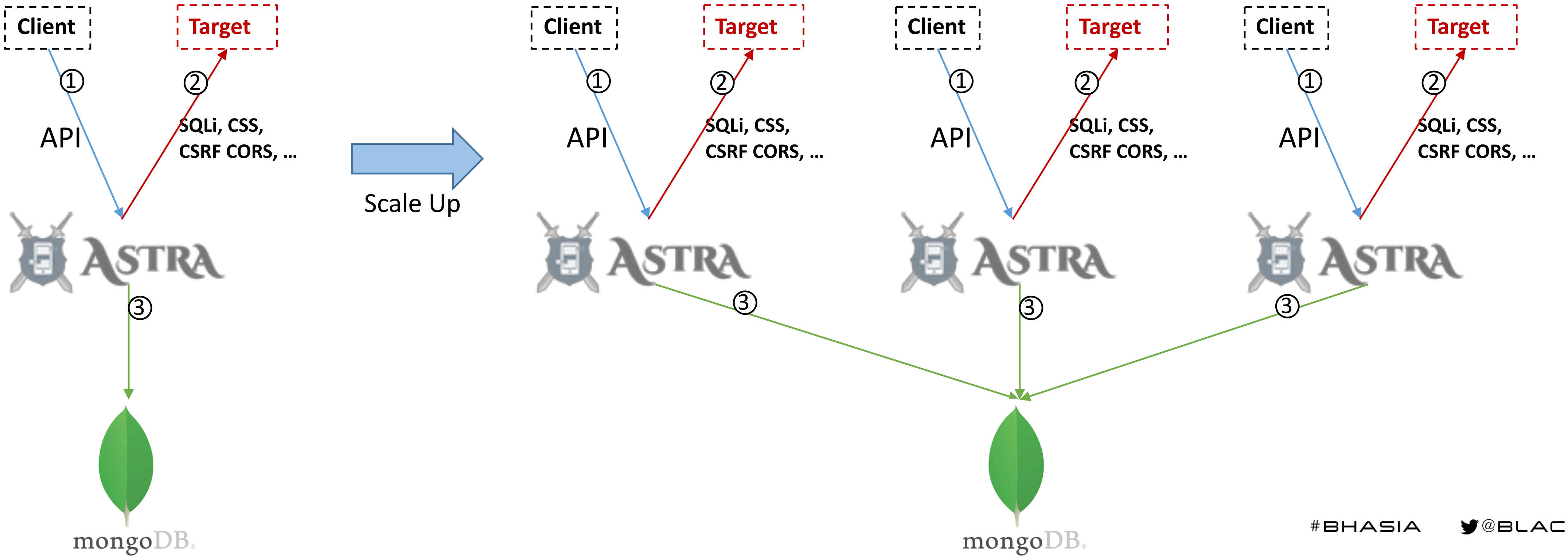
- Generate json file can be used as input for vulnerability assessment tools
  - json file contains: method, path, cookie information, payload data if any
- REST API vulnerability assessment tool example

ASTRA

# Vulnerability Scanning at Scale

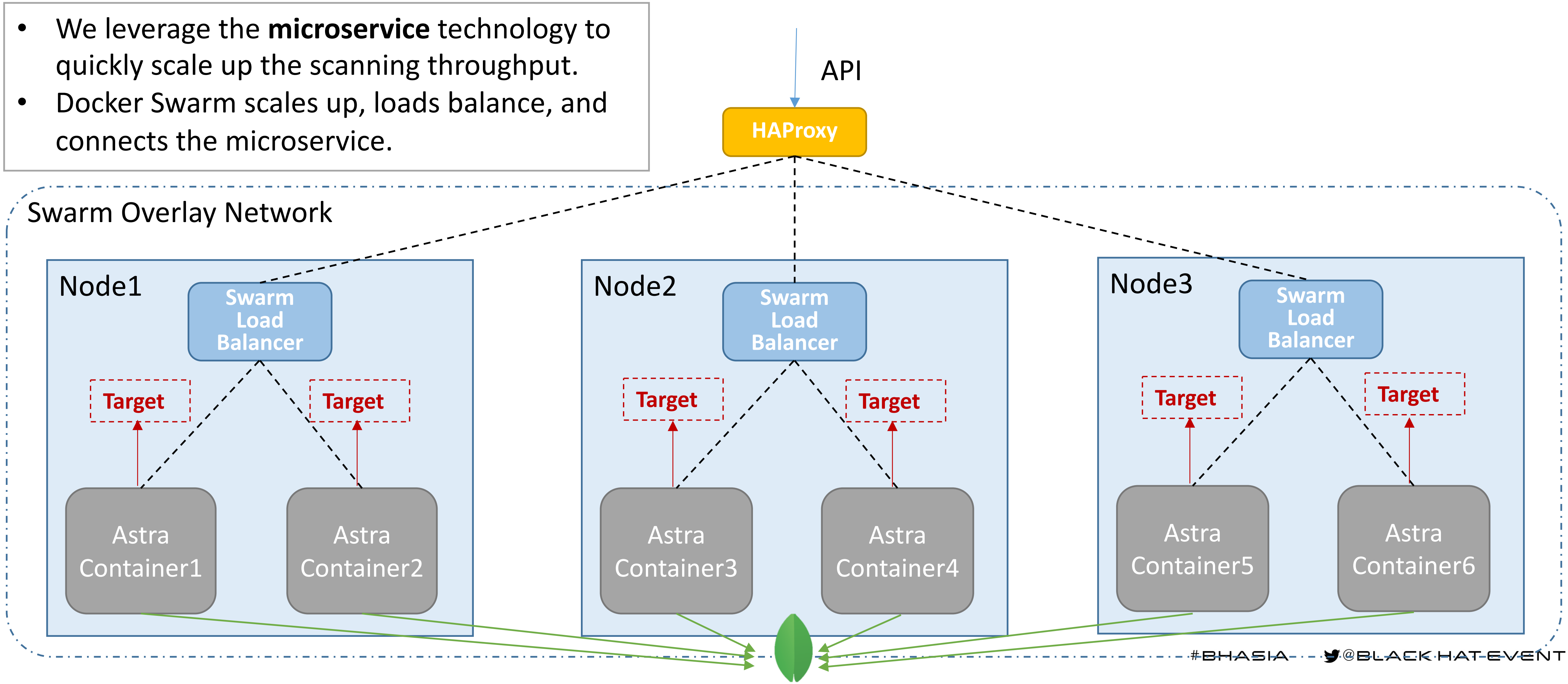
~ 20 secs per scan
< 4 concurrent scans
To slow!

~ 20 secs per scan
< 12 concurrent scans



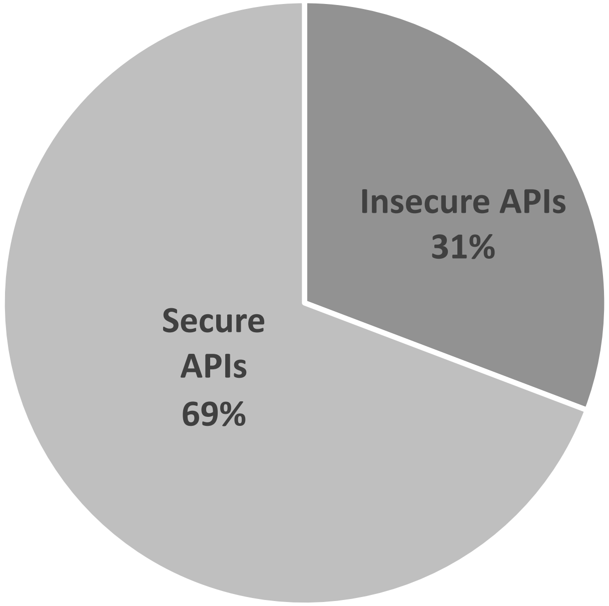
# Vulnerability Scanning at Scale

- We leverage the **microservice** technology to quickly scale up the scanning throughput.
- Docker Swarm scales up, loads balance, and connects the microservice.

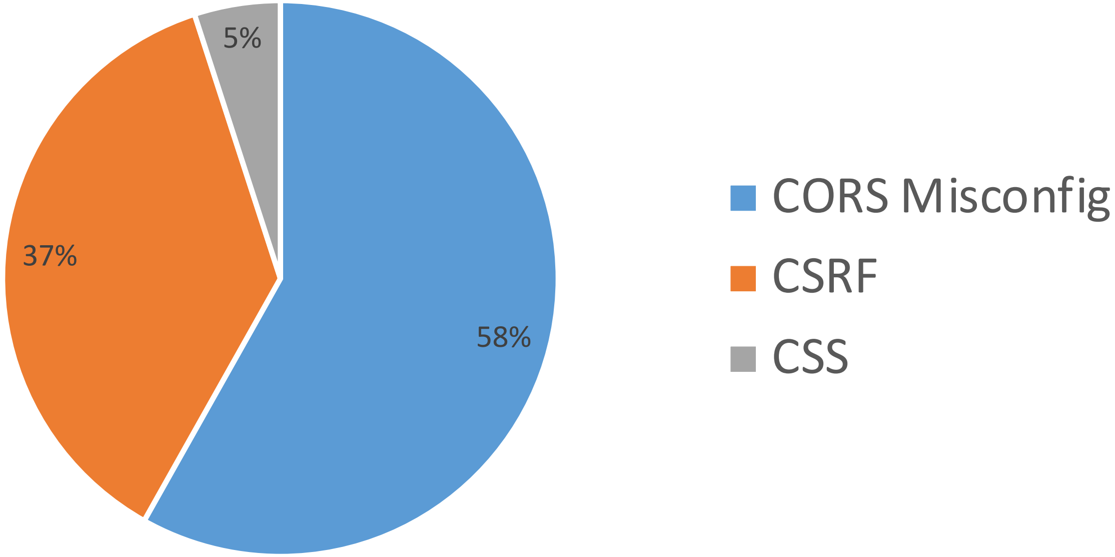


# Vulnerability Scanning Result

## Secure vs. Insecure APIs



## Assessed Vulnerability Distribution



CORS Misconfig: Cross-Origin Resource Sharing Misconfiguration
CSRF: Cross-Site Request Forgery
CSS: Cross-Site Scripting

# Lessons Learned

- It is even more powerful to extend the scan with proper authentication/authorization to gain the complete API Security landscape

- Spider traps may cause the crawler to download an infinite number of URLs from a website

# Contact us

Lei Ding
lei.a.ding@accenture.com

Jay Chen
jay.chen@accenture.com

Azzedine Benameur
azzedine.benameur@accenture.com

Steve Pham
pham.tan.hung@accenture.com