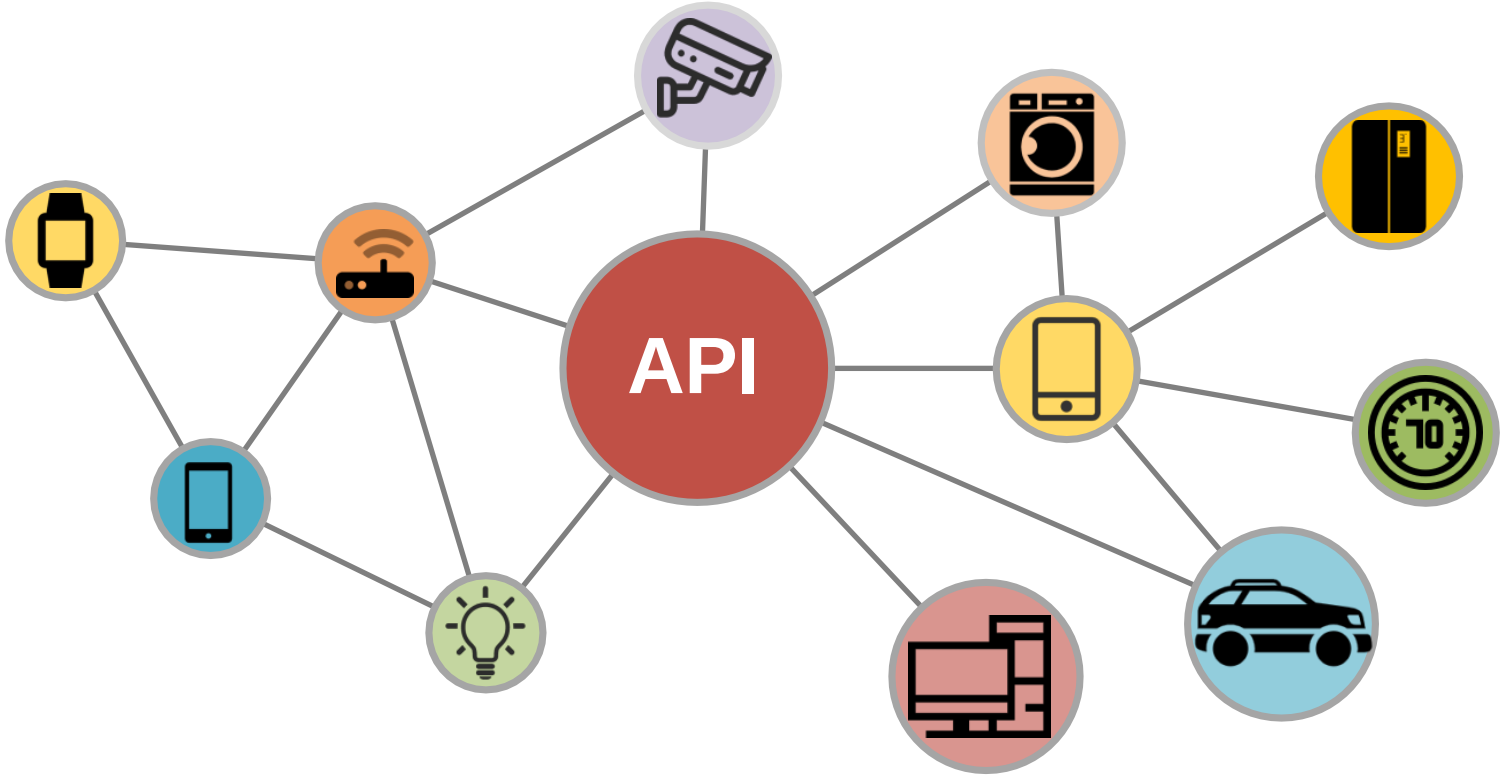
Vulnerability-oriented Testing for RESTful APIs

Wenlong Du*, *Jian Li**, Yanhao Wang, Libo Chen#

Ruijie Zhao, Junmin Zhu, Zhengguang Han, Yijun Wang, and Zhi Xue

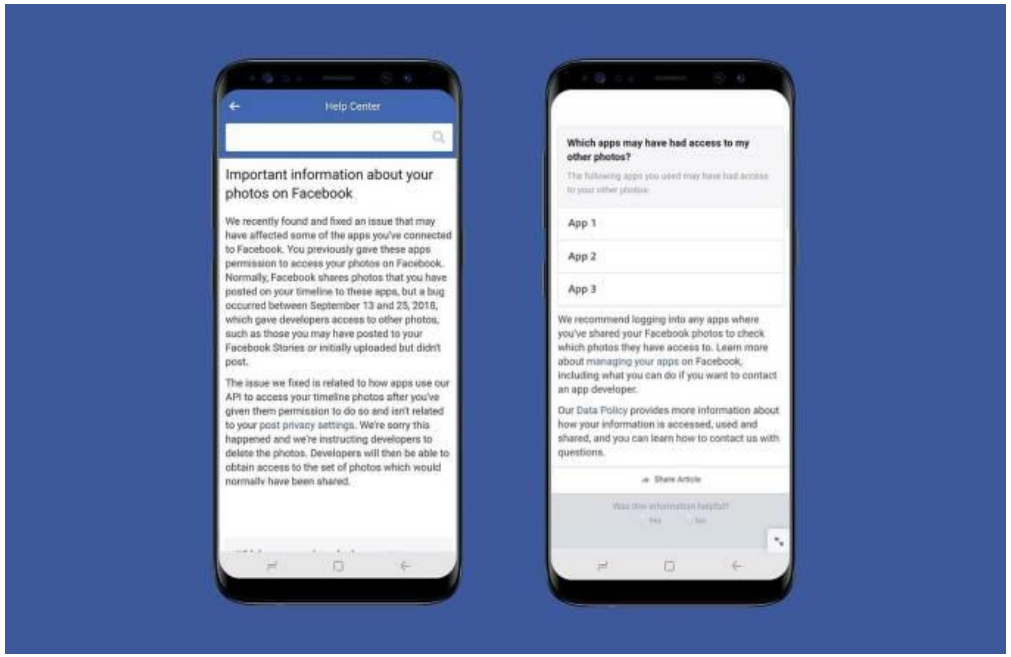
RESTful APIs

- RESTful API has been widely used and spanned across various scenarios
  - Examples: Cloud Services, Content Management Systems (CMS), Internet of Things (IoT)



API Security Issues

- More and more **security issues** appear on APIs
- Photo API bug in Facebook may have affected 6.8 million users



How to discover API security weaknesses to make your API more secure?

Existing Methods

- **RESTler:**
  - **Functionality:**
    - Generates Stateful Test Cases: Parses OpenAPI and infers **producer-consumer** dependencies
    - Detects Errors: Detects ***status code 500*** (“Internal Server Error”)
    - Bug Detection: Utilizes checkers for resource leaks and hierarchy violations
  - **Limitations:**
    - Mainly catches HTTP 500 errors and specific logic bugs
    - **Less** effective in testing for security-specific issues such as **SSRF** or **XSS**
    - Requires extensive time

Motivation: Thinking Like a Hacker

Motivation: Thinking Like a Hacker

- Imagining you are an experienced hacker… 

Motivation: Thinking Like a Hacker

- Imagining you are an experienced hacker… 
  - **Objective**: You aim to uncover SSRF vulnerabilities in Jellyfin. 

Motivation: Thinking Like a Hacker

- Imagining you are an experienced hacker… 🕵️
  - **Objective**: You aim to uncover SSRF vulnerabilities in Jellyfin.🔷
  - **Challenge**: Jellyfin boasts hundreds of API endpoints... 🤯

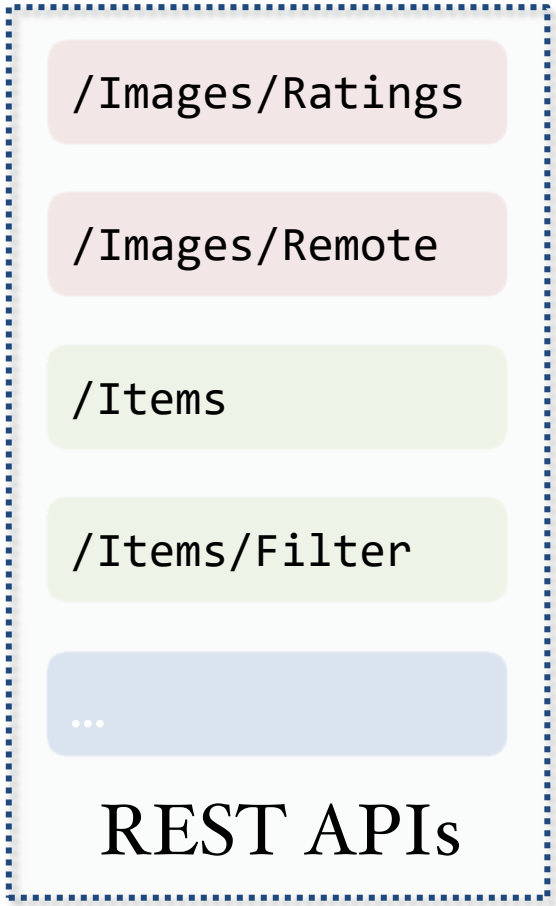
Motivation: Thinking Like a Hacker

- Imagining you are an experienced hacker… 🕵️
  - **Objective**: You aim to uncover SSRF vulnerabilities in Jellyfin.🔷
  - **Challenge**: Jellyfin boasts hundreds of API endpoints... 🤯
  - **Goal**: You need to conduct your tests quickly and efficiently. ⚡

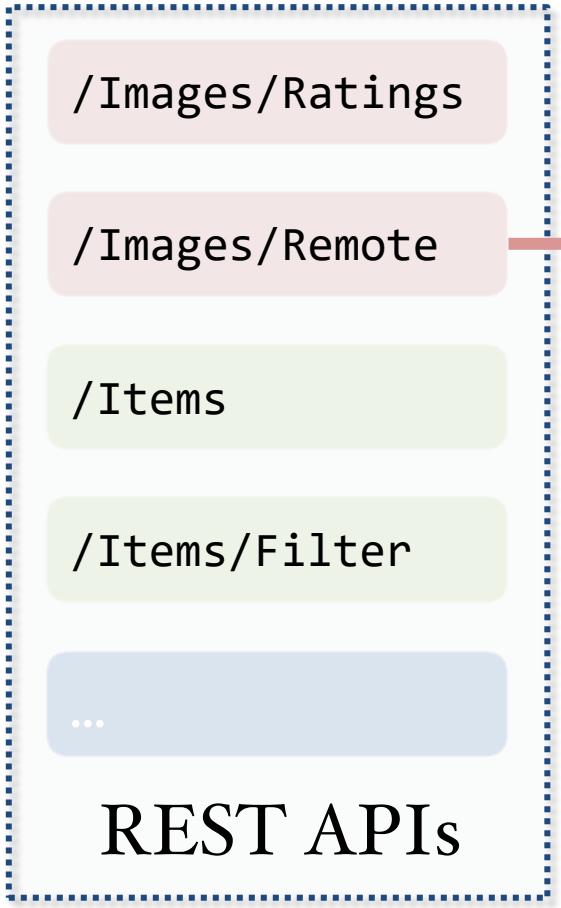
Motivation: Thinking Like a Hacker

- Imagining you are an experienced hacker… 🥷
  - **Objective**: You aim to uncover SSRF vulnerabilities in Jellyfin.🔷
  - **Challenge**: Jellyfin boasts hundreds of API endpoints... 🤯
  - **Goal**: You need to conduct your tests quickly and efficiently. ⚡
  - **Question**: Which API endpoint should be your primary target? 🤔

Motivation: What to Test?

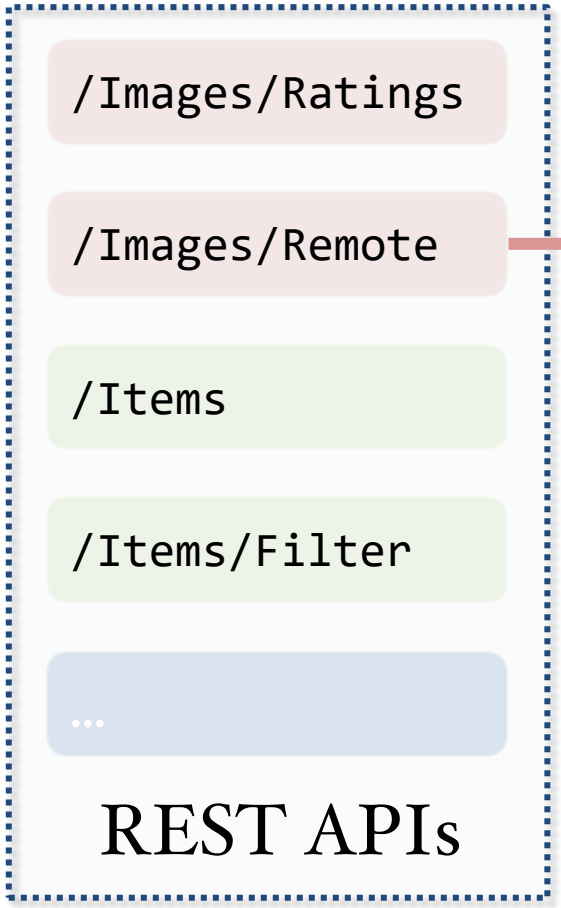


Motivation: What to Test?




```
/Images/Remote:
    get:
      operationId: GetRemoteImage
      parameters:
        - description: The image url.
          in: query
          name: imageUrl
          required: true
          schema:
            description: The image url.
            format: uri
            type: string
    responses:
       description: Remote image returned.
    summary: Gets a remote image.
```

Motivation: What to Test?




```
/Images/Remote:
    get:
      operationId: GetRemoteImage
      parameters:
        - description: The image url.
          in: query
          name: imageUrl
          required: true
          schema:
            description: The image url.
            format: uri
            type: string
    responses:
       description: Remote image returned.
    summary: Gets a remote image.
```

Motivation: What to Test?

/Images/Ratings

/Images/Remote

/Items

/Items/Filter

...

REST APIs

```
/Images/Remote:
    get:
      operationId: GetRemoteImage
      parameters:
        - description: The image url.
          in: query
          name: imageUrl
          required: true
          schema:
            description: The image url.
            format: uri
            type: string
    responses:
       description: Remote image returned.
    summary: Gets a remote image.
```

What if the **imageUrl** points to an **internal** resource? 🤔

Motivation: How to Test?

Attacker

/Images/Ratings

/Images/Remote

/Items

/Items/Filter

...

REST APIs

Internal Service

Motivation: How to Test?

Attacker

/Images/Ratings

/Images/Remote

/Items

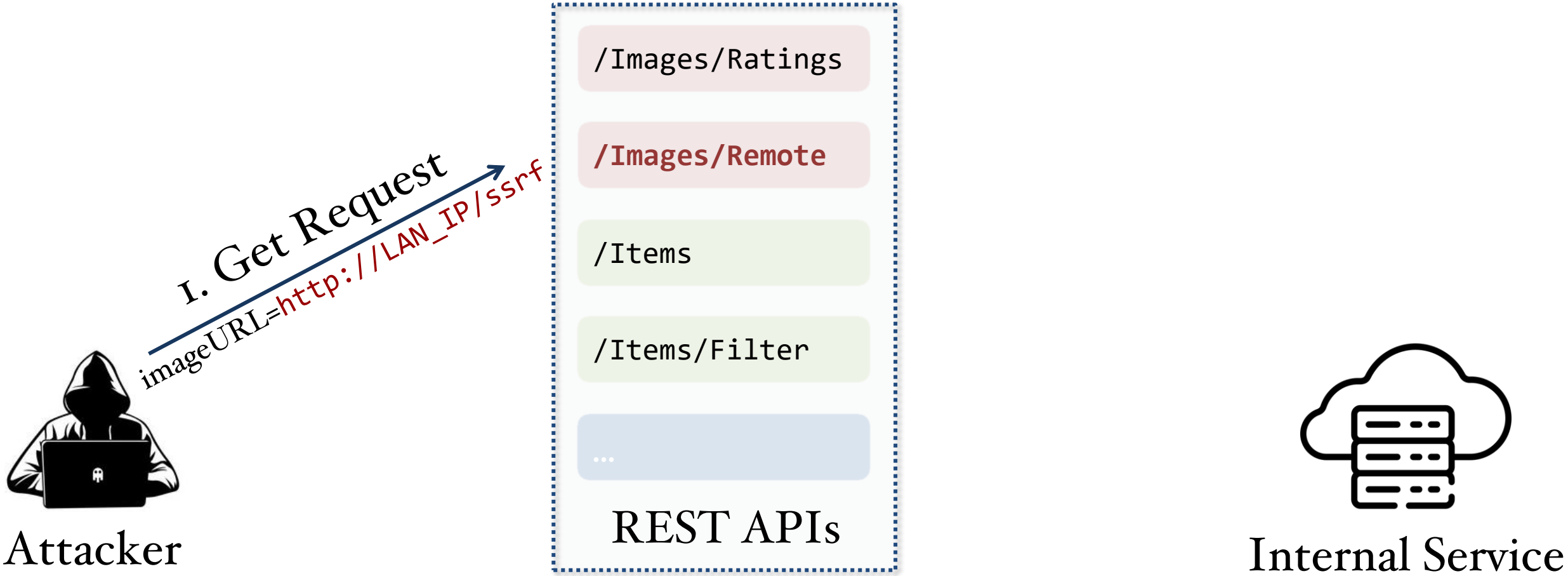
/Items/Filter

...

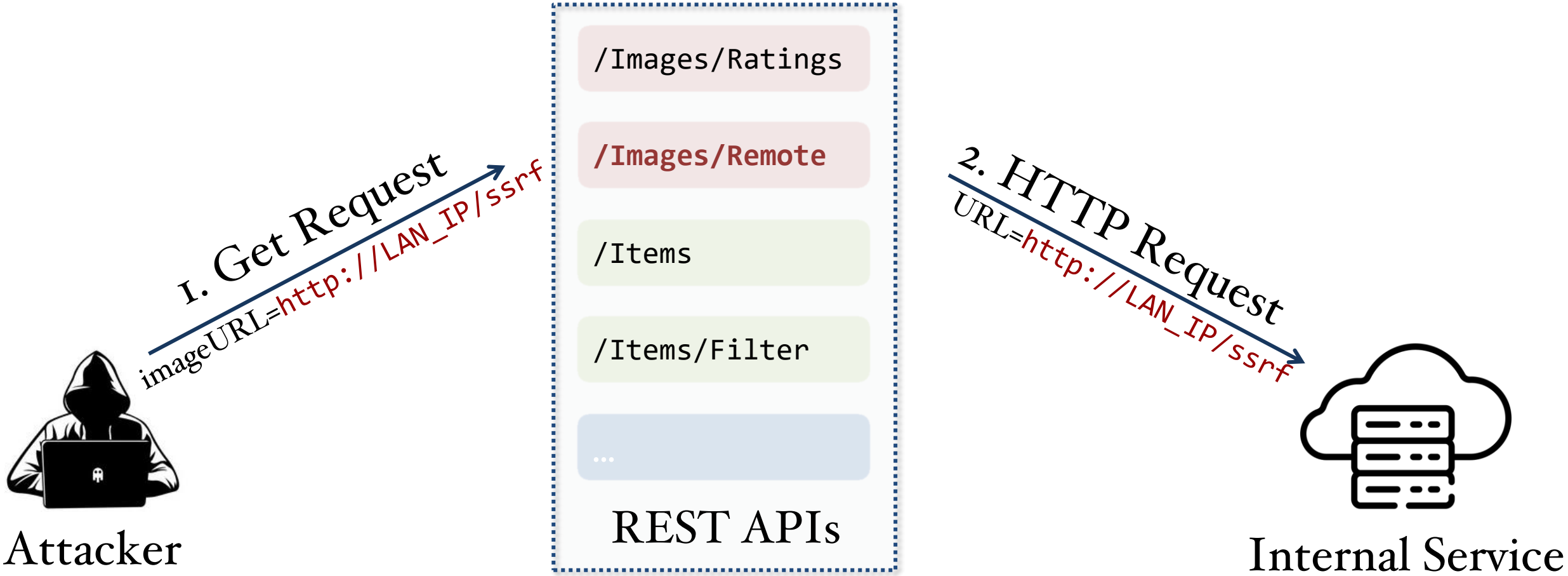
REST APIs

Internal Service

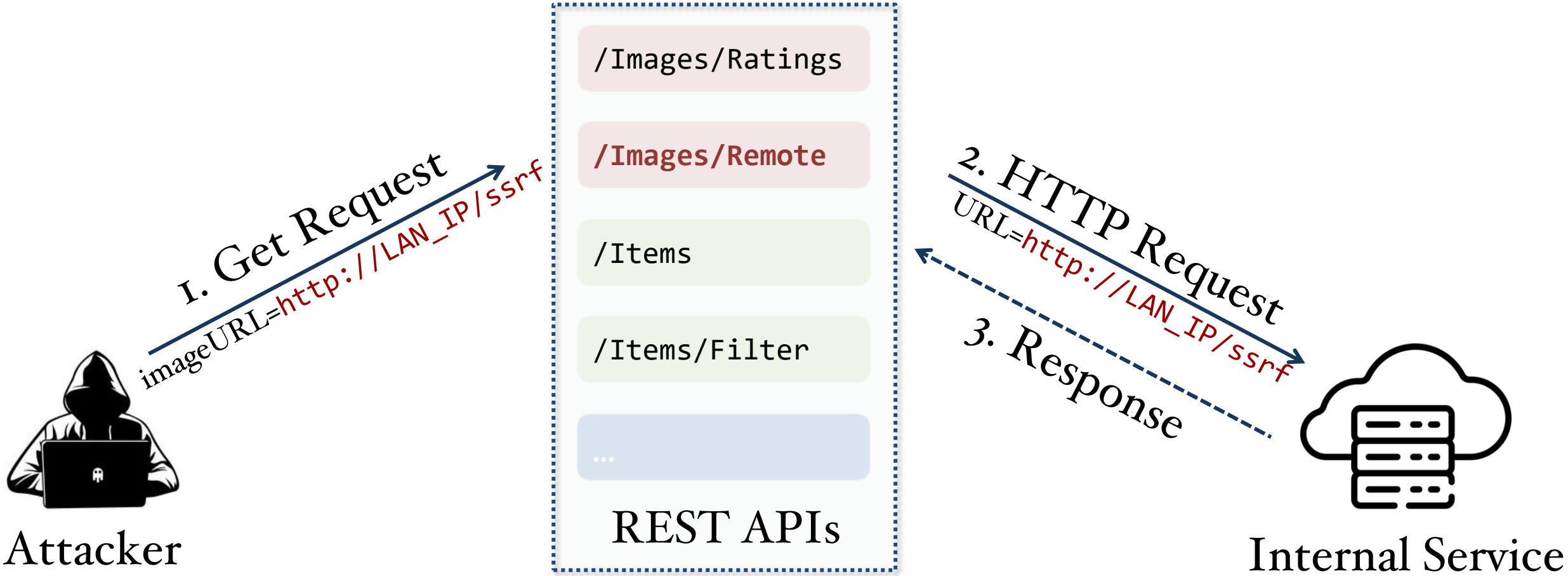
Motivation: How to Test?



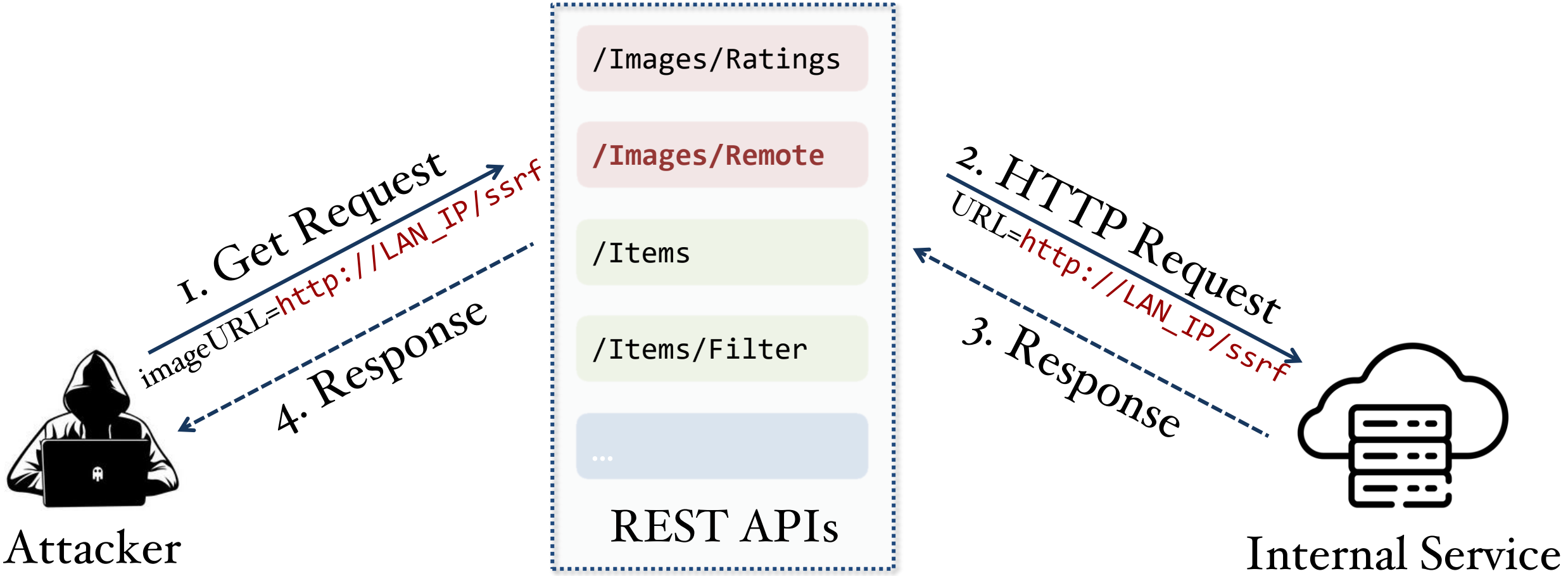
Motivation: How to Test?



Motivation: How to Test?



Motivation: How to Test?



Why Did We Choose This API?

- There is a distinct correlation between specific API security **vulnerabilities** and their **functionalities**
- Example
  - SSRF: Involves requesting remote resources
  - Unrestricted File Upload: Pertains to file operations
  - …

Intuition Verification

Objective

- To demonstrate the correlation between API **functionalities** and **vulnerabilities**

Approach

- Analyzed **six specific CWE types** selected from the CVE database
- Manually summarized the functionalities of APIs corresponding to each vulnerability, if detailed API information was available
- Derived insights from **API specifications**, **source code**, and **vulnerability descriptions**

Example: CVE-2022-43776

CVE-2022-43776 Detail

Description

The url parameter of the /api/geojson endpoint in Metabase versions <44.5 can be used to perform Server Side Request Forgery attacks. Previously implemented blacklists could be circumvented by leveraging 301 and 302 redirects.

Example: CVE-2022-43776

`GET /api/geojson/`

Load a custom GeoJSON file based on a URL or file path provided as a query parameter. This behaves similarly to /api/geojson/:key but doesn't require the custom map to be saved to the DB first.

Load a file based on URL

PARAMS:

- `url` value must be a non-blank string.
- `respond`
- `raise`

Intuition Verification

- **Understanding API Vulnerabilities:**
  - CWE-918 Server-Side Request Forgery (SSRF):
    - Prevalence in APIs that **request remote sources**, such as proxy interfaces.
    - **Example Keywords**: "remote", "proxy", "URL".
    - **Hit Rate:** 81% (17/21)
  - CWE-434: Unrestricted File Upload:
    - Common in APIs focused on **file operations** like uploading
    - **Example Keywords**: "upload", "submit", "import".
    - **Hit Rate**: 83% (35/42)
  - CWE-22 Path Traversal:
    - Associated with **handling files through a "Path" variable**.
    - **Example Keywords**: "path", "dir", "file".
    - Hit Rate: 52% (16/31)

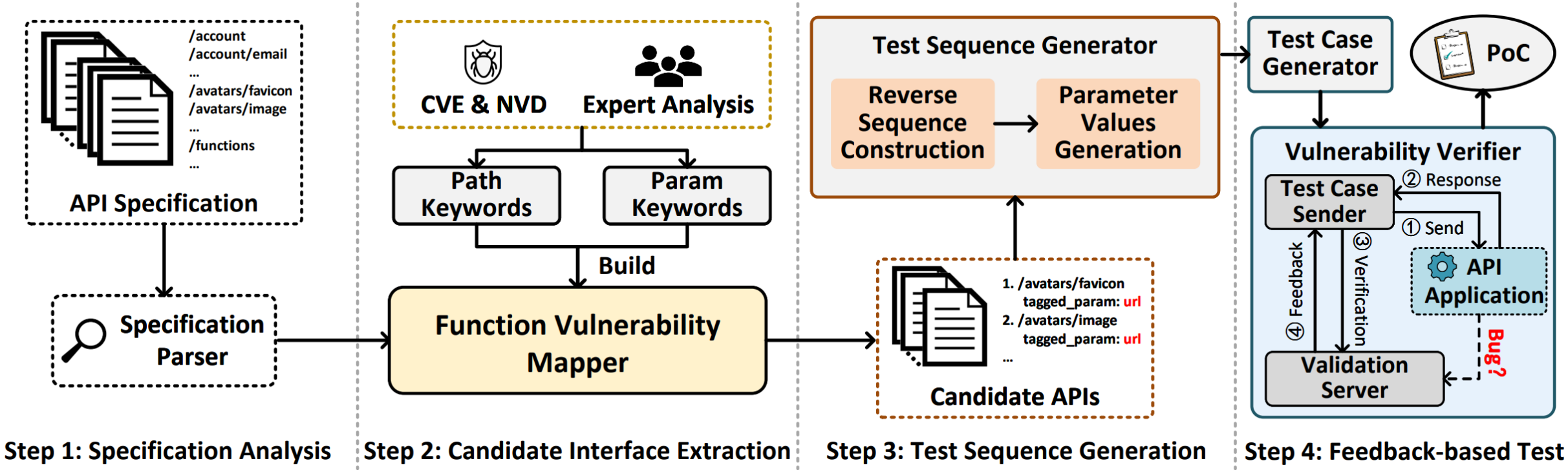
Intuition Verification

- **Understanding API Vulnerabilities:**
  - CWE-78 OS Command Injection:
    - Utilized for setting configurations via **commands** in OS shell.
    - **Example Keywords**: "CMD", "command", "system".
    - **Hit Rate**: 40% (22/55)
  - CWE-89 SQL Injection:
    - Responsible for handling SQL **database operations**.
    - **Example Keywords**: "SQL", "database", "select".
    - **Hit Rate**: 53% (37/70)
  - CWE-79 Cross-site Scripting (XSS):
    - Present in APIs that **display front-end pages** showcasing text.
    - **Example Keywords**: "display", "content", "view".
    - **Hit Rate**: 35% (19/55)
- On average, **57%** of vulnerable API interfaces affected by a specific type of bug belonged to the same functionality.

Challenge

- C1: How can we efficiently distinguish between API functional interfaces and pinpoint potentially vulnerable functions?
- C2: How can we generate test case sequences that match protocol states and target vulnerable interfaces?
- C3: How can we generate valid test cases for different functions based on their security vulnerabilities?

Architecture of VoAPI2



Semantic Keyword Collection

- Build datasets based on **CVEs** and **NVDs**
- Conduct **word frequency** analysis and leverage **expert experience** to get keywords

Vulnerability Type	#Keywords API path	#Keywords API parameter	API Type
SSRF	10	22	Resource Request APIs
Unrestricted Upload	12	8	File Upload APIs
Path Traversal	12	3	Path Processing APIs
Command Injection	12	7	System Configuration APIs
SQL Injection	11	4	Database Operation APIs
XSS	15	12	Text Display APIs

Candidate Interface Extraction

- Analyzes the **API specification** and generate a grammar file
- Utilizes the semantic keywords to check for their presence in the **paths** and **parameters** of a given API
- Categorizes APIs into corresponding feature categories based on keywords and map potential vulnerability types

Reverse Sequence Construction

```
GET /database/collections/{collectionId}/documents/{documentId}
```

Reverse Sequence Construction

```
GET /database/collections/{collectionId}/documents/{documentId}
```

Reverse Sequence Construction

```
GET /database/collections/{collectionId}/documents/{documentId}
```

```
GET /database/collections
POST /database/collections
GET /database/collections/{collectionId}
```

```
POST /database/collections/{collectionId}/documents
```

Reverse Sequence Construction

```
GET /database/collections/{collectionId}/documents/{documentId}
```

```
GET /database/collections ❌
POST /database/collections
GET /database/collections/{collectionId}
```

```
POST /database/collections/{collectionId}/documents
```

Reverse Sequence Construction

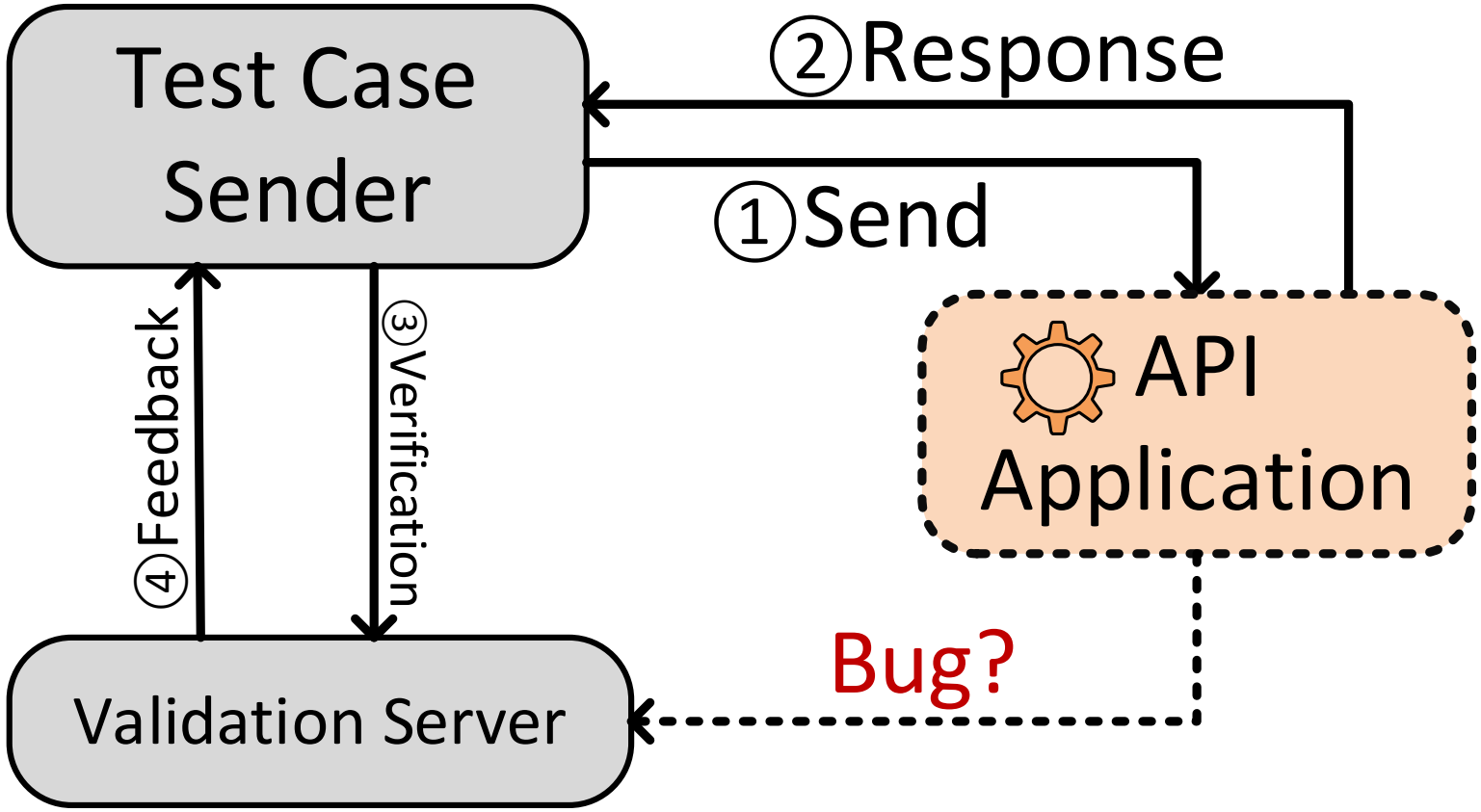
```
GET /database/collections/{collectionId}/documents/{documentId}
```

```
GET /database/collections ❌
POST /database/collections
GET /database/collections/{collectionId}
```

```
POST /database/collections/{collectionId}/documents
```

Use Producer-Consumer Relationship, CRUD Semantics and Resource Hierarchy to get producer

Feedback-based Testing and Verification



Samples in Testing Corpus

Sample Testing Corpus for Different API Types.

- Resource Request API: http://IP:PORT/ssrf/{0}
- File Upload API: evil files (.jsp,.asp,.php, etc.)
- Path Processing API: /etc/passwd; C://Windows//win.ini
- System Configuration API: curl http://IP:PORT/command/{0}
- Database Operation API: 1" or "1"="1; SQLMap
- Text Display API: <img src='http://IP:PORT/xss/{0}'>

Evaluation

- Q1: Can VoAPI2 discover vulnerabilities in real-world APIs?
- Q2: Can VoAPI2 efficiently generate test cases?
- Q3: How does the vulnerability-oriented strategy of VoAPI2 affect the testing results?

Evaluation

- Dataset
  - Seven real-world RESTful API applications
  - Three scales
  - Code Repository and CMS and Web Service

Applications	Endpoint	Type	#Download
Gitlab	358	Code Repository	100M+
Jellyfin	405	CMS	100M+
Appwrite	95	CMS	5M+
Microcks	44	CMS	600K+
Casdoor	121	CMS	20K+
Gitea	325	Code Repository	20K+
Rbaskets	22	WebService	10K+

Compared Tools

- Vulnerability Scanners
  - ZAP
  - Astra
- RESTful API Testing Tools
  - Restler
  - RestTestGen
  - MINER

Q1: Real-world Vulnerabilities

Application	Version	Path	Parameter	Type	Producer	0-day	Bug-IDs	Vulnerability Identification Tools					
								VoAPI[2]	RESTler	RestTestGen	MINER	ZAP	Astra
Appwrite	0.9.3	/avatars/favicon	url	SSRF	✗	✓	CVE-2023-27159	✓	✗	✗	✗	✓	✓
	0.9.3	/avatars/image	url	SSRF	✗	✓	CVE-2023-27159	✓	✗	✓	✗	✓	✓
	0.9.3	/teams	name	XSS	✗	✗	CVE-2022-2925	✓	✗	✗	✗	✗	✓
	0.9.3	/teams/{teamId}/memberships	name	XSS	✓	✗	CVE-2022-2925	✓	✗	✗	✗	✗	✗
	0.9.3	/database/collections	name	XSS	✗	✗	CVE-2022-2925	✓	✗	✗	✗	✗	✗
	0.9.3	/functions	name	XSS	✗	✗	CVE-2022-2925	✓	✗	✗	✗	✗	✗
	0.9.3	/usrs	name	XSS	✗	✗	CVE-2022-2925	✓	✗	✗	✗	✗	✗
Rbaskets	1.2.3	/api/baskets/{name}	forward_url	SSRF	✗	✓	CVE-2023-27163	✓	✗	✗	✗	✗	✗
Jellyfin	10.7.1	/Images/Remote	imageUrl	SSRF	✗	✗	CVE-2021-29490	✓	✓	✗	✓	✓	✓
	10.7.1	/Items/RemoteSearch/Image	imageUrl	SSRF	✗	✗	CVE-2021-29490	✓	✗	✗	✗	✓	✓
	10.7.1	/Items/{itemId}/RemoteImages/Download	imageUrl	SSRF	✓	✗	CVE-2021-29490	✓	✗	✗	✗	✗	✗
	10.7.1	/Repositories	Url	SSRF	✗	✓	CVE-2023-27161	✓	✗	✗	✗	✗	✗
	10.7.1	/Playlists	name	XSS	✗	✗	CVE-2023-23636	✓	✗	✗	✗	✗	✗
	10.7.1	/Repositories	name	XSS	✗	✗	CVE-2022-35910	✓	✗	✗	✗	✗	✗
	10.7.1	/Collections	name	XSS	✗	✗	CVE-2023-23635	✓	✓	✗	✗	✗	✗
	10.7.1	/Startup/User	Name	XSS	✗	✓	1 unassigned	✓	✗	✗	✗	✗	✗
Casdoor	1.13.0	/api/get-organizations	field	SQL Injection	✗	✗	CVE-2022-24124	✓	✗	✗	✗	✓	✓
	1.13.0	/api/upload-resource	fullFilePath	Unrestricted Upload	✗	✗	CVE-2022-38638	✓	✗	✗	✗	✗	✗
Microcks	1.17.1	/jobs	repositoryUrl	SSRF	✗	✓	1 unassigned	✓	✓	✓	✓	✗	✗
	1.17.1	/artifact/download	url	SSRF	✗	✓	1 unassigned	✓	✓	✗	✓	✓	✓
Gitea	1.16.7	/repos/{owner}/{repo}/contents/{filepath}	content	Unrestricted Upload	✓	✗	CVE-2022-1928	✓	✗	✗	✗	✗	✗
	1.16.7	/repos/{owner}/{repo}/hooks	url	SSRF	✓	✗	CVE-2018-15192	✓	✗	✗	✗	✗	✗
GitLab	8.17.0	/v3/hooks	url	SSRF	✗	✗	CVE-2018-8801	✓	✓	✗	✗	✗	✗
	8.17.0	/v3/projects	import_url	SSRF	✗	✗	CVE-2022-0249	✓	✓	✗	✗	✗	✗
	8.17.0	/v3/projects/{id}/deploy_keys	title	XSS	✓	✗	CVE-2022-2230	✓	✗	✗	✗	✗	✗
	8.17.0	/v3/projects/{id}/milestone	title	XSS	✓	✗	CVE-2022-1190	✓	✗	✓	✗	✗	✗

Q1: Real-world Vulnerabilities

- **Vulnerability**
  - Identified 26 vulnerabilities
  - 7 previously unknown
  - 23 of them are assigned CVE numbers
  - Other tools can only detect few of them

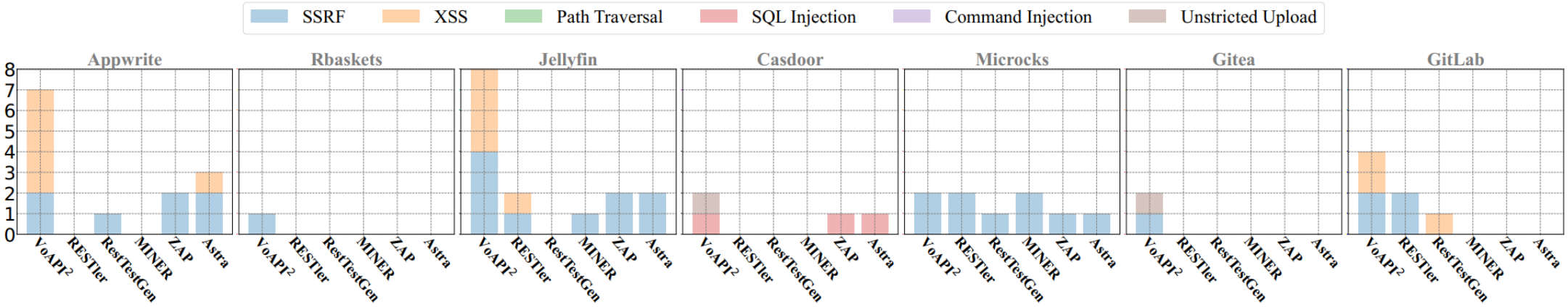
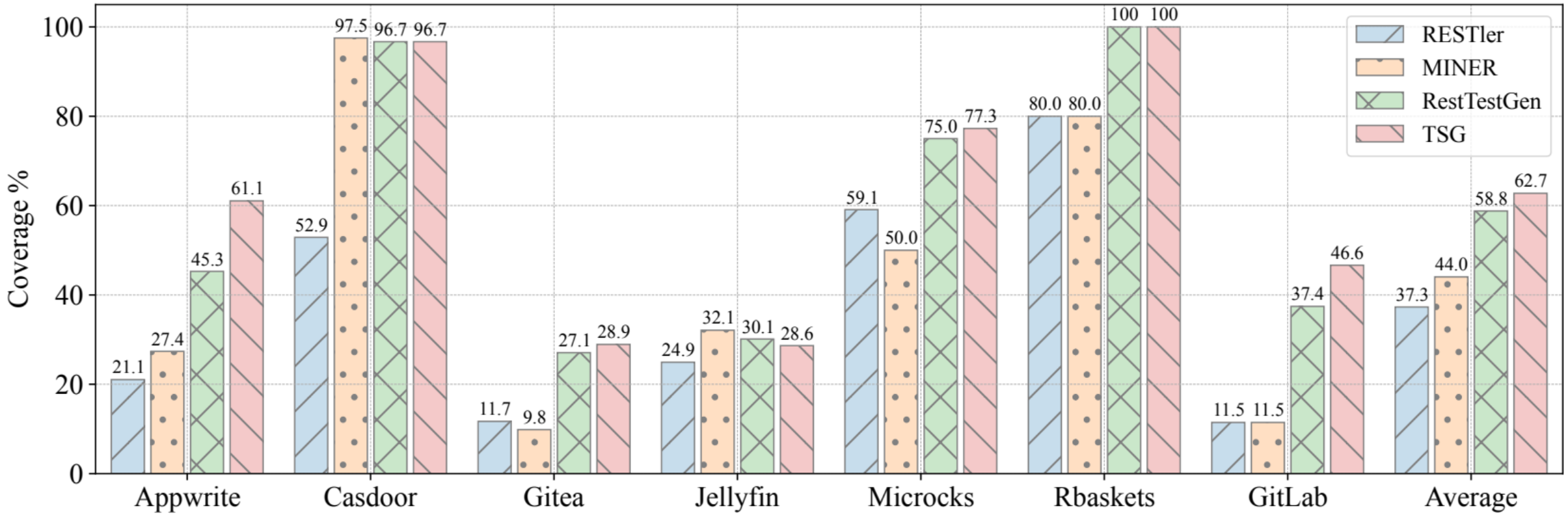


Figure 4: **The vulnerabilities and their types uncovered by different tools on evaluation benchmarks .**

Q2: Efficiency



VoAPI2 achieves **comparable results** in test sequence generation.

Q3: Ablation Study

- **Remove** vulnerability-oriented strategy in VoAPI2-V.
- **No new** vulnerabilities were found by VoAPI2-V.
- VoAPI2-V is much slower.

Application	Bug-IDs	Path	VoAPI2	VoAPI2-V
Appwrite	CVE-2023-27159	/avatars/favicon	2.81s	1min31s
	CVE-2023-27159	/avatars/image	2.90s	1min39s
	CVE-2022-2925	/teams	53.42s	26min30s
	CVE-2022-2925	.../memberships	1min03s	34min36s
	CVE-2022-2925	/database/collections	15.28s	17min18s
	CVE-2022-2925	/functions	1min39s	60min16s
	CVE-2022-2925	/users	1min30s	48min08s
Rbaskets	CVE-2023-27163	/api/baskets/{name}	2.08s	38s
Jellyfin	CVE-2021-29490	/Images/Remote	5.79s	225min46s
	CVE-2021-29490	/Items/.../Image	7.26s	246min29s
	CVE-2021-29490	/Items/.../Download	6.68s	228min13s
	CVE-2023-27161	/Repositories	10.24s	322min38s
	CVE-2023-23636	/Playlists	9min09s	368min04s
	CVE-2022-35910	/Repositories	8min57s	322min47s
	CVE-2023-23635	/Collections	8min25s	35min11s
	1 unassigned	/Startup/User	10min30s	495min28s
Casdoor	CVE-2022-24124	/api/get-organizations	5min04s	43min28s
	CVE-2022-38638	/api/upload-resource	2min33s	75min50s
Microcks	1 unassigned	/jobs	1.90s	11min19s
	1 unassigned	/artifact/download	2min06s	90min55s
Gitea	CVE-2022-1928	/repos/.../{filepath}	3.61s	94min41s
	CVE-2018-15192	/repos/.../hooks	2.49s	36min05s
GitLab	CVE-2018-8801	/v3/hooks	1min12s	236min05s
	CVE-2022-0249	/v3/projects	1min17s	20min17s
	CVE-2022-2230	/v3/.../deploy_keys	1min49s	53min33s
	CVE-2022-1190	/v3/.../milestone	2min36s	145min41s

Summary

- We propose VoAPI2, a novel inspection framework, to apply a vulnerability-oriented strategy to inspect bugs
- Based on the key insight that the type of vulnerability in an API interface is closely related to its functionality
- VoAPI2 discovered 7 zero-day and 19 disclosed bugs on seven real-world RESTful APIs

Our Code:

https://github.com/NSSL-SJTU/VoAPI2

Thank You

Questions?