



REPORT

API SECURITY RISK ANALYSIS

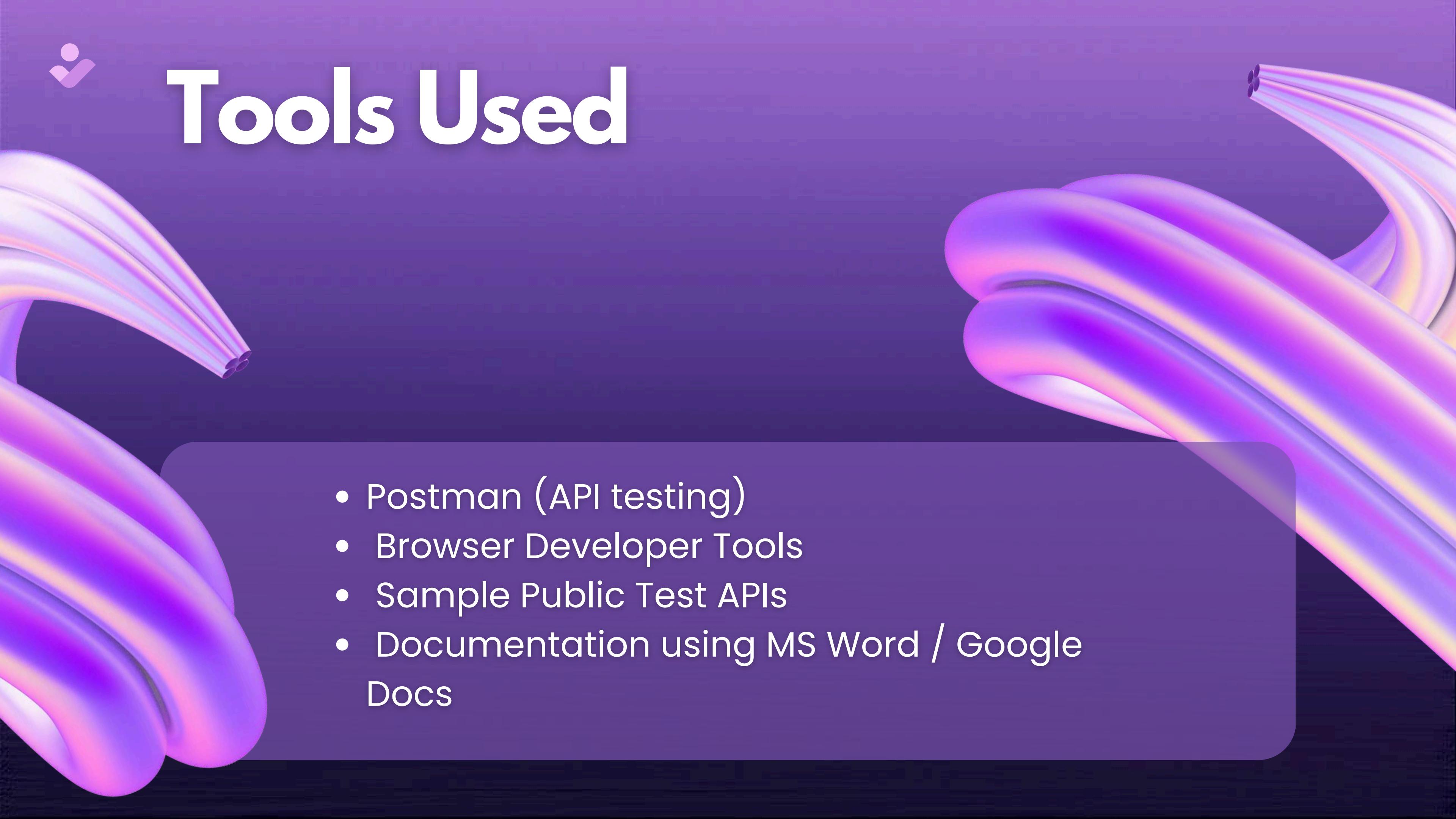
**TITLE:API SECURITY RISK ANALYSIS
SUBMITTED BY:RENUGA P
COURSE:CYBER SECURITY LAB**



Introduction

APIs (Application Programming Interfaces) are essential for modern applications because they enable communication between systems, mobile apps, and web services. However, insecure APIs can expose sensitive data, allow unauthorized access, and lead to cyberattacks. This report analyzes common API security risks, their impact, and recommended mitigation strategies.





Tools Used

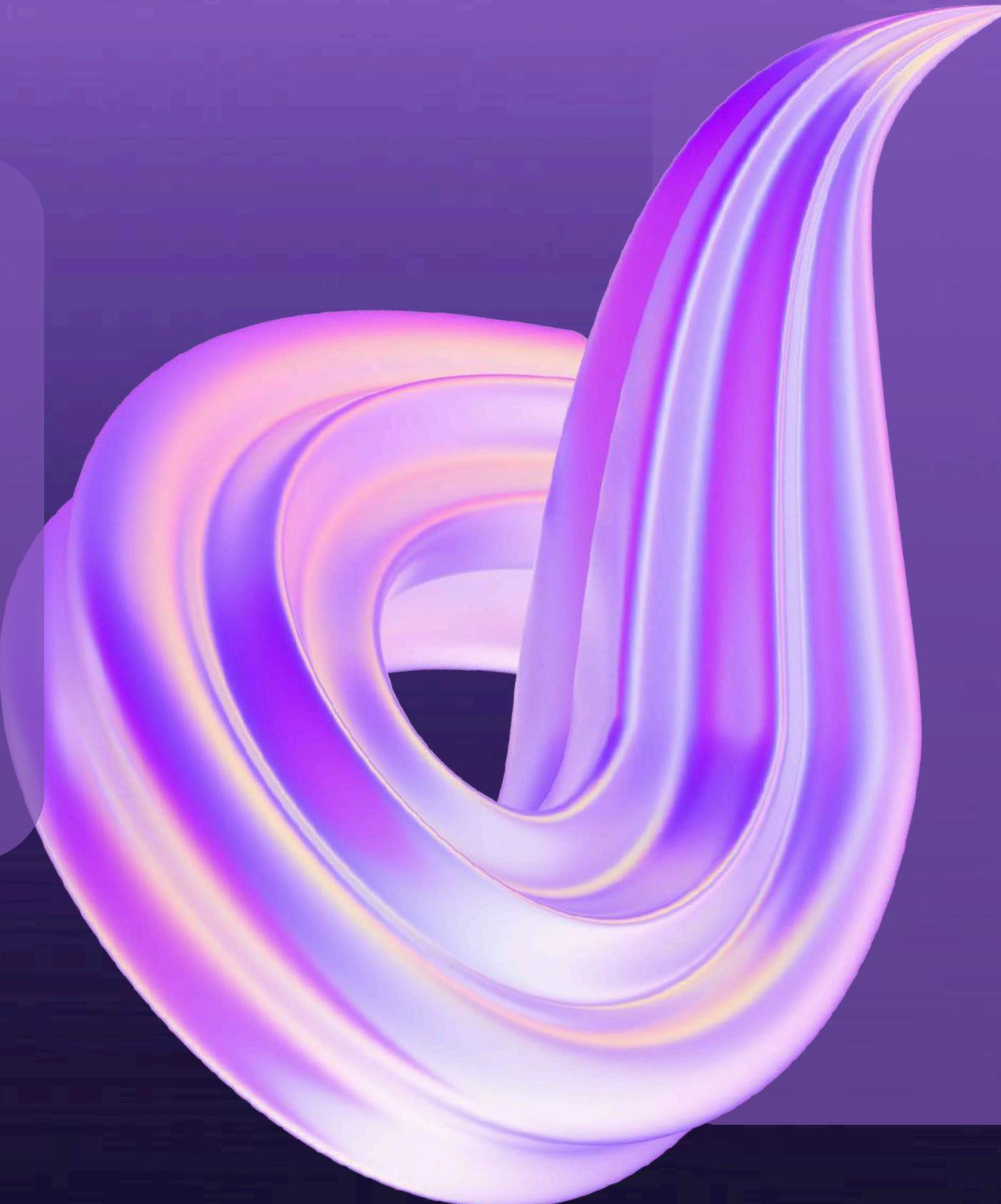
- Postman (API testing)
- Browser Developer Tools
- Sample Public Test APIs
- Documentation using MS Word / Google Docs

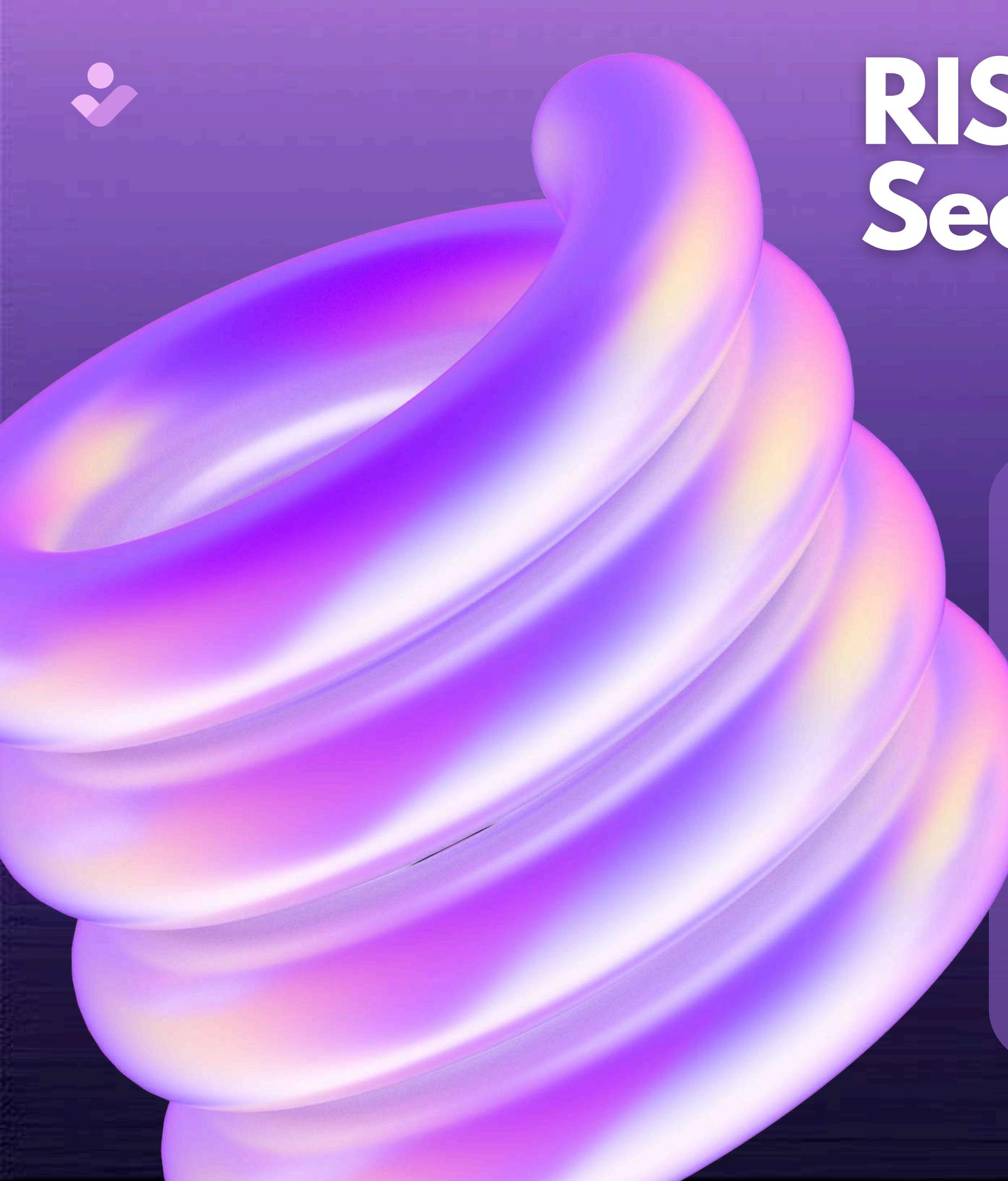


Methodology

The following steps were used to perform the API security assessment:

1. Identified available API endpoints.
2. Tested authentication and authorization mechanisms.
3. Checked for sensitive data exposure in responses.
4. Verified input validation and error handling.
5. Tested for rate limiting and access control.





RISK 1:identified Security Risks

Risk 1: Broken Authentication

Description: Weak or missing authentication allows attackers to access APIs without proper login.

Impact: Unauthorized users may access private data.

Example: API accepts requests without tokens.

Remediation:

Implement strong authentication (JWT, OAuth)

Enforce token expiration and secure storage



Risk 2: Broken Authorization

Description: API does not properly verify user permissions.

Impact: Users can access or modify other users' data.

Remediation:

Apply role-based access control (RBAC).

Validate user identity on every request.

Risk 3: Sensitive Data Exposure

Description: API responses contain confidential data such as passwords or personal information.

Impact: Data leaks may lead to identity theft or privacy breaches.

Remediation:

Encrypt sensitive data.

Avoid sending unnecessary information in API responses.



Risk 4: Lack of Rate Limiting

Description: No restriction on request frequency.

Impact: Attackers can perform brute force or denial-of-service attacks.

Remediation:

Implement rate limiting.

Use throttling and request monitoring.

Risk 5: Improper Input Validation

Description: API accepts invalid or malicious input.

Impact: May lead to injection attacks or system crashes.

Remediation:

Validate all user inputs.

Use input filtering and sanitization.

The screenshot shows a REST client interface with the following details:

- Method: GET
- URL: https://jsonplaceholder.typicode.com
- Params: None
- Query Params: None
- Body: JSON (with an empty object {})
- Status: 200 OK
- Response Body (JSON):

```
1 {  
2   "id": 5,  
3   "name": "Chelsey Dietrich",  
4   "username": "Kamren",  
5   "email": "Lucio_Hettinger@annie.ca",  
6   "address": {  
7     "street": "Skiles Walks",  
8     "suite": "Apt. 502",  
9     "city": "Almond Town",  
10    "zipcode": "90593-1234",  
11    "geo": {  
12      "lat": -37.3379, "lon": 175.0522  
13    }  
14  },  
15  "phone": "1-770-736-8290 x4312",  
16  "website": "hildegard.org",  
17  "company": {  
18    "name": "Romaguera-Crona",  
19    "catchPhrase": "Multi-layered client-server neural-net",  
20    "bs": "syndicated content",  
21  }  
22}
```

Tool used:postman

GET https://jsonplaceholder.typicode.com/

Send

Params

Query Params

Key	Value	Description
Key	Value	Description

Body

HTML

```
</footer>
<script src="https://cdnjs.cloudflare.com/ajax/libs/prism/1.29.0/prism.min.js"></script>
```

200

What I'll do right after that (1)

1. Add/verify tests in the collection My Collection (My Collection) and make sure the collection is set up correctly.
2. Create an Environment (e.g., Dev/Production) so you can reuse the same test cases and auth variables so the same tests can be run in different environments.
3. Set up a Monitor to run the collection periodically and receive notifications on failure/recovery.

Also: if you haven't yet, you can take the quick tour of Postman (it's a quick walkthrough of the various features). If you have time, do that now, or should we proceed directly to the details?

GET Get data
Describe what you need. Press @ for code completion.

HTTP My Collection / Post data

Save Share

POST https://jsonplaceholder.typicode.com/users

Send

Docs Params Auth Headers (8) Body Scripts Tests Settings Cookies

raw JSON Schema Beautify

```
1 {  
2   "name": "<script>alert(1)</script>",  
3   "email": "test@test.com"  
4 }
```

Body 201 Created 367 ms 1.28 KB

{ } JSON Preview Visualize

```
1 {  
2   "name": "<script>alert(1)</script>".
```

Key Value Description

Body

200

{} JSON

```
26   "id": 2,  
27   "name": "Ervin Howell",  
28   "username": "Antonette",  
29   "email": "Shanna@melissa.tv",  
30   "address": {  
31     "street": "Victor Plains",  
32     "suite": "Suite 879",
```

GET Get data

Describe what you need. Press @ for code completion.

AI

Risk Impact Summary

Risk Type | Severity | Impact

Broken Authentication | High | Unauthorized access

Broken Authorization | High | Data manipulation

Data Exposure | High | Privacy breach

No Rate Limiting | Medium | System overload

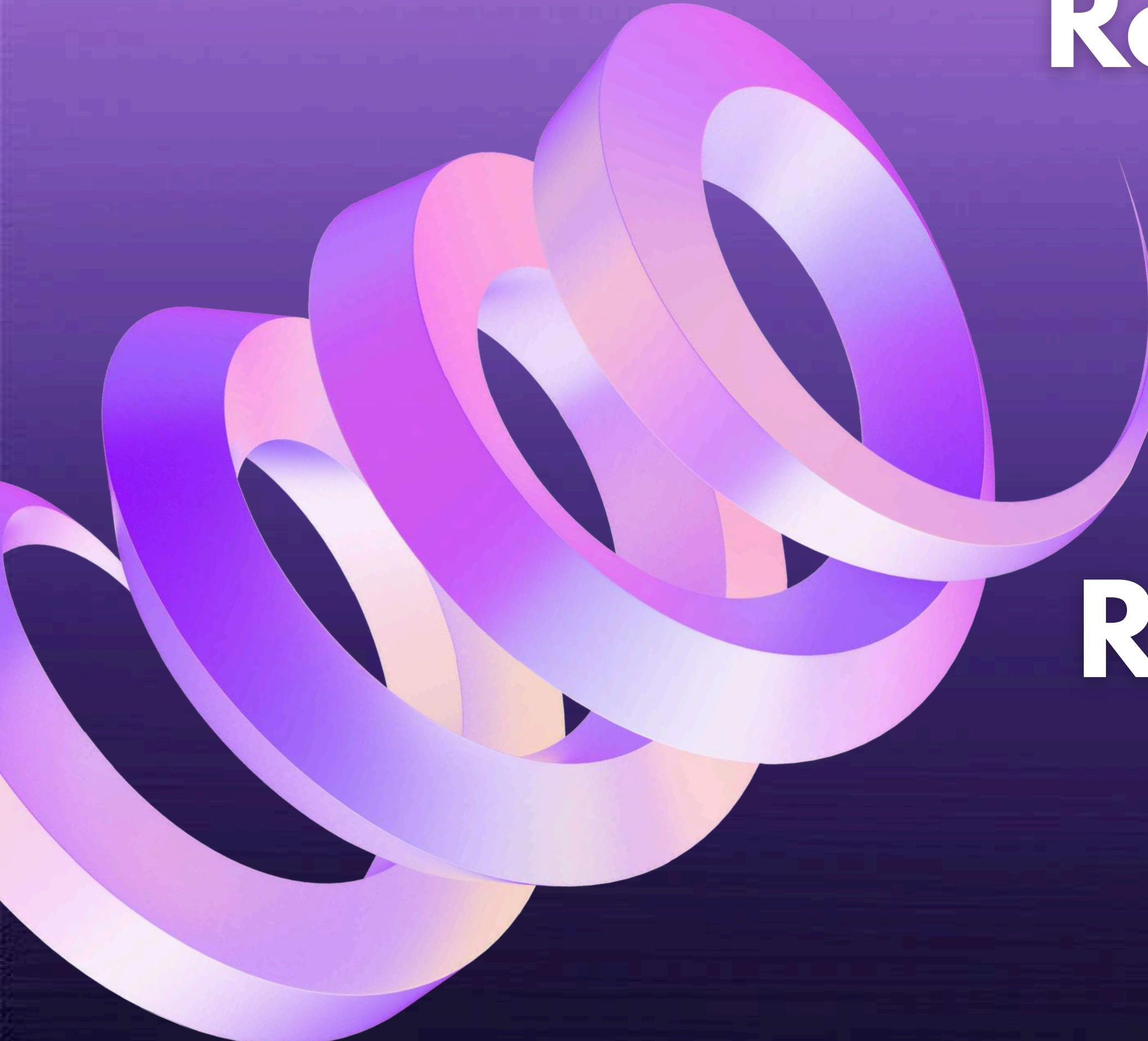
Poor Validation | Medium | Injection attacks



Conclusion

The API security analysis identified multiple risks related to authentication, authorization, data protection, and input validation. Implementing strong security controls, monitoring mechanisms, and proper validation techniques will significantly reduce the risk of cyberattacks and improve overall API security.

Recommendations

A large, abstract graphic on the left side of the slide features several thick, overlapping ribbons in shades of purple, pink, and white. These ribbons are twisted and looped, creating a complex, organic shape that spans most of the width of the slide.

Use secure authentication methods.
Encrypt sensitive data transmissions.
Implement access control policies.
Apply rate limiting and logging.
Conduct regular API security testing.

References

OWASP API Security Top 10
API Security Best Practices documentation
Cybersecurity Fundamentals Study Materials