code intelligence

REST API Testing Guide

How to Improve the Security and Quality of Your REST APIs Through Testing

TL;DR: In this guide, contains best practices for REST API testing, that will help you improve the security and stability of your web applications.

> “The only thing growing faster than API integration is API attack traffic.”
>
> Niklas Henrich
>
> CTO, Code Intelligence

Content

What Is a REST API?

REST (Representational State Transfer) is a highly popular web API type because it offers flexible, fast, and simple communication between RESTful web applications. Compared to other API formats, REST is by far the most used, as over 80% of public web APIs are RESTful. Although stateful REST APIs are theoretically compatible with any protocol or data format, they mostly communicate through HTTP, using JSON, XLT, HTML, XML, or simple text. Out of these data formats, JSON is the most common as it is compatible with most languages.

Their adaptability makes REST APIs especially useful for services that are growing in complexity. Thanks to their ability to process commands from multiple users and different data formats, REST APIs are highly popular in various industries, such as ecommerce or IoT.

REST APIs use five HTTP methods to request a command:

GET: Retrieve a resource
POST: Create a new resource
PUT: Update an existing resource
PATCH: Modify an existing resource
DELETE: Delete an existing resource

Below, you can see an example of a POST request:

```
POST/WebGoat/register.mvc HTTP1.1
Connection:keep-alive
Content-Length:75
Cache-Control:max-age=0
Upgrade-Insecure-Requests:1
Content-Type:application/x-www-form-urlencoded
User-Agent:Mozilla/5.0(X11; Linux x86_64)AppleWebKit/537.36(KHTML, like
Gecko)Chrome/92.0.4515.159 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.0,image/avif,image/webp,image/
apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding:gzip, deflate
Accept-Language:en-US,en;q=0.9

username=cifuzz&password=nnMjJa721x>&mathingPassword=nnMjJa72lx&agree=agree
```

REST vs SOAP

The main difference between REST and SOAP (Simple Object Access Protocol) is that to be RESTful, an API has to simply meet a specific set of characteristics. Meanwhile, SOAP is an actual protocol, built to enable applications to communicate across languages and platforms. REST APIs are generally seen as more flexible and faster than SOAP protocols. Although SOAP protocols slightly decrease the speed of web services, they provide several features such as improved security, atomicity, consistency isolation, and durability (ACID). SOAP interfaces can process multiple protocol types (HTTP, SMTP TCP, etc.). However, SOAP return messages are always sent in XML. Thus, while REST APIs enable flexible high-speed communication, SOAP web services are slightly slower, but offer more built-in functionality.

REST vs gRPC

gRPC (Remote Procedure Call) is a Google-developed open-source data interchange mechanism that uses the HTTP/2 protocol. gRPC APIs exchange data using the Protocol Buffers binary format (Protobuf), which imposes standards that developers must follow when creating or using gRPC web APIs. While REST APIs are mainly useful for microservice architectures and third-party apps, gRPC is often applied in IoT systems, browserless mobile apps and applications with multiplexed streams.

Types of REST API Tests

API tests are not only done for security, but also for other reasons such as performance, functionality, and stability. Which testing approach is the right one for your REST APIs, strongly depends on what you are trying to achieve. However, most modern API testing tools can be used for more than one form of testing. Generally, REST API testing approaches, include:

Unit Testing: Testing the functionality of individual operations

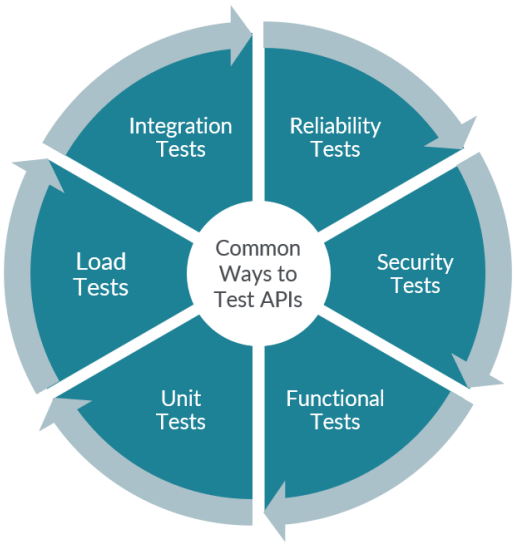
Integration Testing: Testing the interaction between multiple software modules

Functional Testing: Ensuring that REST APIs behave exactly as it should

Load Testing: Measuring how many calls REST APIs can handle

Reliability Testing: Ensuring that REST APIs produce consistent results and connections

Security Testing: Validating REST API encryption methods and access control



Common Ways to Test APIs

The Challenges of REST API Testing

Securing REST APIs is a challenging task, as they are highly complex: They are difficult to reach, produce countless parameter combinations, and constantly communicate with a vast number of other systems. Looking for security vulnerabilities in REST APIs manually is like looking for a needle in a haystack. To deal with the complexity of REST APIs, many dev teams test APIs with automated methods. Below, you can find an overview of the top 6 challenges of REST API testing.

1. Securing REST API Parameter Combinations

As presented below, REST APIs consist of various parameters such as request method, request URI and query parameter - just to name a few. These parameters can take up countless combinations that have to be tested, as specific parameter combinations can lead to erroneous program states.

2. Validating REST API Parameters

Validating REST API parameters is highly challenging. If they are not validated properly, issues such as wrong string/data types and parameter data outside the predefined value range can come up.

3. Maintaining the Data Formatting Schema

The data formatting schema specifies how REST APIs handle responses and requests. The challenge in maintaining data formatting is that whenever new parameters are added, they have to be included in the schema.

4. Testing REST API Call Sequences

Testers need to ensure that REST API calls are called in the correct order to prevent errors. In REST APIs this is especially important since they are generally multithreaded.

5. REST API Testing Set-Up

Setting up automated testing cycles is the part of REST API testing that requires the most manual effort. Especially for large projects enterprise testing platforms will help you speed up the initial set-up dramatically.

6. Error Reporting for REST APIs

Especially with black-box testing tools, error reporting for REST APIs is tricky, as the amount of tested parameter combinations is unknown. The best way to monitor and report REST API tests is with coverage-guided testing approaches, as they can provide meaningful coverage and error reports.

Find out more about common REST API testing challenges.

Enhancing Security Through Automated REST API Test Tools

Security testing is a particularly important part of REST API testing, as the implications of an exploited security vulnerability are usually not limited to the functionality and usability of a program. Due to the complexity and connectivity of REST APIs, finding an API tester that can detect API endpoints and cover all relevant parameter combinations is a tough nut. Manual testing is often too time-consuming and tends to neglect edge cases and vulnerabilities that stem from the communication between services.

To test microservice architectures with all their dependencies, it is considered a best practice to automate your testing efforts as much as possible. The main reasons why automated testing tools are so beneficial for REST APIs are:

System Complexity: REST APIs and backend services are often integrated into a layered architecture, which makes it difficult to cover all relevant test cases. Automated API test tools enable developers to deal with this complexity by identifying endpoints and testing relevant parameter combinations more efficiently than it would be possible with manual testing methods.

Missing GUI: Since REST APIs do not have a GUI, all REST API tests must be performed at the message level, making it even more difficult for developers to conduct manual tests. In many cases, it is easier for an API tester to write a script that automates tests than it would be to write them manually.

Structured Inputs/Outputs: REST APIs generally underlie highly standardized protocols that mainly process HTTP, JSON, and XML files. Therefore, they provide a fairly stable and uniform interface to the tested program. Since the structure of the inputs and outputs are partly predefined, automating REST API tests is usually a viable option.

Automated API testing tools will save you time and increase the functionality, reliability, and security of your application. So, automate your testing if you can! But, don't avoid manual testing completely. Your team should always be able to run manual tests, to validate if the automated tests are still working, as they are supposed to. As always, you need to find the mix that fits your use case best.

How to Automate Security Testing for REST APIs

Due to the complex structure of REST APIs, automated testing is one of the most effective ways to ensure their security and stability. But not all automated testing approaches are equally effective. The fastest way to implement software test automation would be with black-box API testing tools, such as Burp or OWASP ZAP, potentially enhanced with some additional system tests.

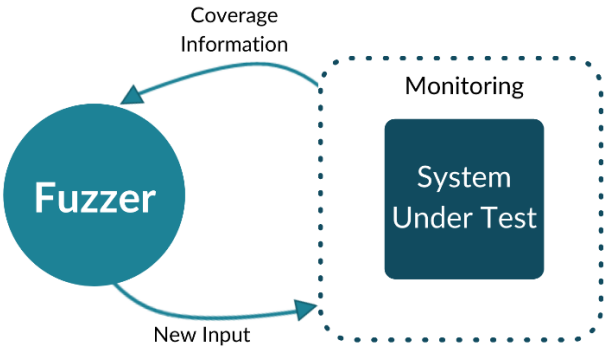
Although these black-box approaches are somewhat automated, they leave plenty of room for improvement, as they still require testers to have prior knowledge about the system under test in order to be effective. Black-box tests are great to test APIs from an attacker's perspective. They generate test inputs randomly, from static corpora, from OpenAPI imports, or based on heuristics.

However, such inputs often fail to reach complex vulnerabilities and edge cases, since they do not take code coverage into account. For example, a black-box testing tool would take the API request from above and try out countless different parameter settings in hopes of identifying a request that breaks something.

Automated white-box testing is far more effective at finding buggy REST API requests: Since they use information about the source code, white-box approaches can automatically exclude irrelevant parameter settings from the corpus. Through information about code coverage, they can find crashing REST API requests much faster and much more accurately. White-box automation also enables better reporting by providing code-coverage visibility. The advantages of this approach are especially useful to secure vast microservice environments that are connected through APIs, and projects that are expanding in size.

Fuzz Testing for REST APIs

The most efficient way to implement automated white-box testing is with feedback-based fuzzing. Feedback-based fuzz testing is a form of data-driven testing, during which code instrumentation is used to measure the test progress within individual microservices and APIs. Through this instrumentation, the fuzzer collects information about test inputs, which it then uses to create further inputs that traverse even more code paths. Modern fuzzers can be customized to scan for specific bug classes and to remove blockers. Since this technology can be integrated into any build system, it enables developers to continuously test APIs for security vulnerabilities and stability issues.



Feedback-based Fuzz Testing

Show Me Some Code

In this recorded live coding session, I will demonstrate how to use fuzzing to automate your security testing for REST APIs, on an intentionally insecure web application (WebGoat) that is usually used for educational purposes.

Use Case: How to Automate Your REST API Testing in 5 Easy Steps

Click below to watch the full live coding session on YouTube:

https://youtu.be/Pyf04VPO0GE

What Kind of Web Vulnerabilities Can You Find With Fuzzing

In the webinar linked above, I am using the Code Intelligence Testing Platform, to build automated REST API testing cycles. The Code Intelligence platform, enables runtime error detection, advanced REST and gRPC API scans, and OWASP vulnerability detection. Here, is an overview of some bug classes you can find with this fuzzing platform. Click here to see the full list.

Category	Reference	Severity
Broken Access	OWASP A01:2021	critical
Cryptographic Failures	OWASP A02:2021	critical
Injections (XXS, SQL, …)	OWASP A03:2021	critical
Insecure Design	OWASP A04:2021	critical
Security Misconfiguration	OWASP A05:2021	critical
Vulnerable and Outdated Components	OWASP A06:2021	critical
Identification and Authentication Failures	OWASP A07:2021	critical
Software and Data Integrity Failures	OWASP A08:2021	critical
Identification Logging and Monitoring Failures	OWASP A09:2021	critical
Server-Side Request Forgery	OWASP A10:2021	critical
Improper Neutralization of Input During Web Page Generation	CWE-79	high
Sensitive Cookie with Improper SameSite Attribute	CWE-1275	medium
Sensitive Cookie Without ‘HttpOnly’ Flag	CWE-1004	medium
Sensitive Cookie in HTTPS Session Without ‘Secure’ Attribute	CWE-614	medium
Insufficient Logging	CWE-778	medium
Logging of Excessive Data	CWE-779	medium
Exposure of Sensitive Information to an Unauthorized Actor	CWE-200	high
Generation of Error Message Containing Sensitive Information	CWE-209	high
Failed Assertion	CWE-617	low
Uncaught Exceptions	CWE-248, CWE-216	low
Logic Issues	CWE-840	low
Infinite Loops	CWE-835	low
Denial of Service (DoS)	CWE-400	low

How to Get Started With REST API Testing

I hope you now have a broad overview of REST API testing and the different approaches to it. If you are curious about how fuzz testing can help you build more secure web apps, you can always reach out to me via oss-security@code-intelligence.com, or book a demo with one of my colleagues. We will help you find the right API Testing approach, that fits your purposes.

→ Book a Demo

About Me

Hi, I'm Daniel Teuchert, a Senior Security Engineer at Code Intelligence. As part member of our customer success team, I support dev teams in automating their security testing, and implementing modern fuzzing. I hold a master's diploma in IT Security in addition to OSWE and OSCP offensive security certifications.

Imprint

Code Intelligence
Rheinwerkallee 6
53227 Bonn, Germany

Get in Touch
info@code-intelligence.com

Follow Us