API SECURITY

Websocket Vs REST API

Introduction

If you're someone dealing with APIs and involved in application/web development, WebSocket or REST API won't sound alien to you. However, only a few have the clarity on what are sets them apart. Well, with this post, we attempt to make things clear for everyone who is using these two technologies regularly.

Learning Objectives

Description of protocols

REST and WebSocket are two key technologies with which a developer deals regularly. Unless clarity on their differences and purposes isn't attained, effective utilization isn't possible. But before we delve deeper into the WebSocket API vs REST API, let's understand the basic meaning of these two.

REST

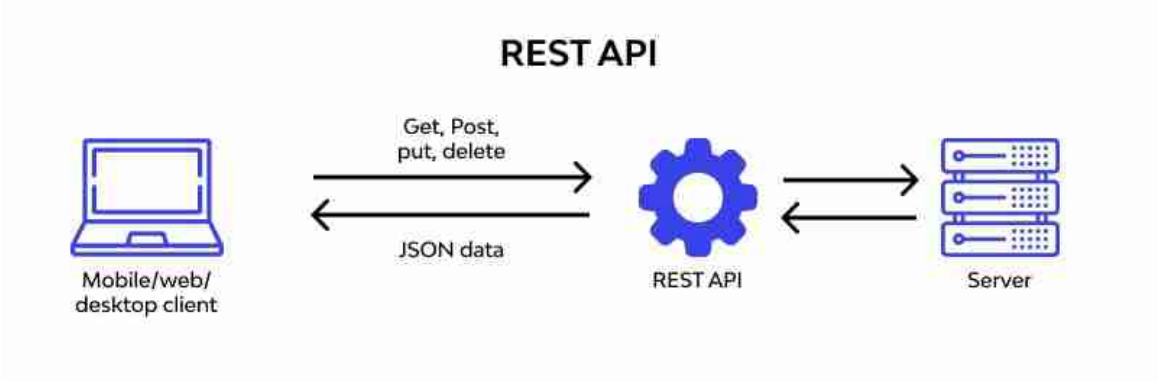
On a technical level, REST is nothing but a collection of design principles or policies referred to extensively during web application/service development. Only HTTP protocol-based web application development is assisted by REST.

As such HTTP-based communication is unidirectional mostly, the use of REST API makes data availability on-request possible. REST API will carry the information that you want to carry.

There are certain traits of REST-based operations.

For instance, they are stateless as well as standard. Every REST request will be created as per the HTTP verbs (GET, PUT, POST, and Delete).

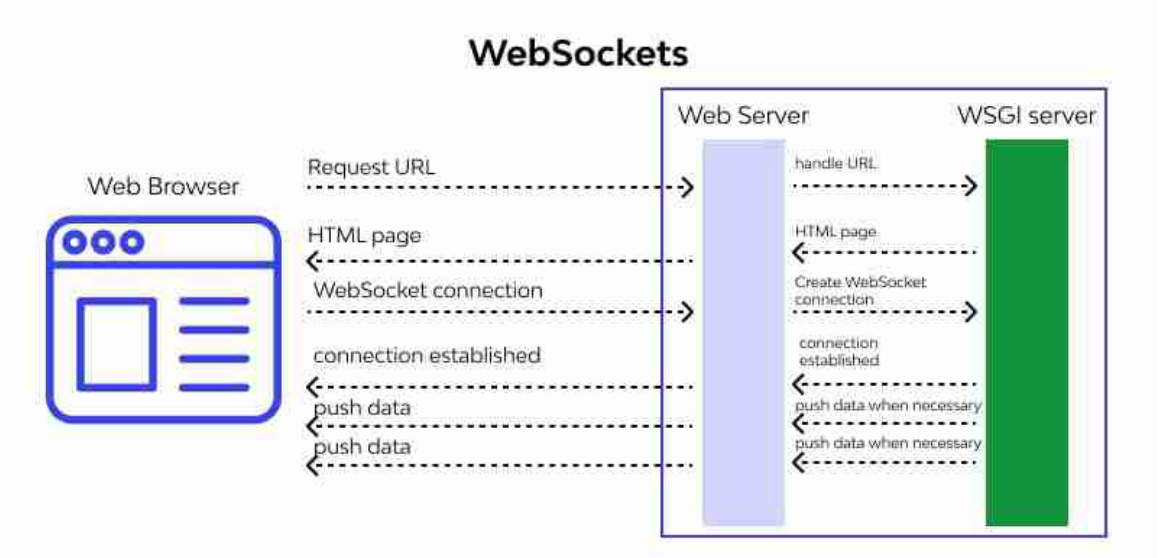
Depending upon the verb used in the request creation, the REST request will process the data. Say a REST API features Delete verb then the data it carries will be deleted as the verb is telling it to do so. REST architecture allows independent deployment of the client and the servers.



Rest API

Websocket

WebSocket protocol, contrary to REST, is stateful while helping two applications in seamless data transmission or information exchange. It's based on port & sockets and works only over a TCP connection. It supports bidirectional communication. By bidirectional, we mean that the same socket/port delivers and receive the data. Hence, a continuous connection is established.



WebSocket in action

Purposes of REST API usage

REST API exists to support the stateless communication. Such communication doesn't demand continuous data delivery. Data is requested only when it's required by the web application. You can compare the function of REST with the function of a vending machine. The vending machine won't provide you with the goods unless you place a request. It works only when it's given a command.

Purposes of Websocket usage

Extensively, WebSocket is used when an application demands ongoing or uninterrupted data delivery. For example, a chat application needs to receive the app all the time. Even if the end-user is not opening the app, the message should be delivered. Only WebSocket can enbale such continuous communication. In such unbroken data delivery, using REST will become resource-extensive whereas WebSocket simplifies the job.

What Sets WebSocket Apart from the REST?

Without much ado, let's talk about the most concerning aspects of REST API vs WebSocket API i.e. how are they different:

Characteristics of WebSocket

1. Stateful and Bidirectional
2. Affordable
3. Uses sockets and ports, so, is counted among low-level protocols.
4. Client and server are independently operational.
5. Widely used in the development of real-time applications;
6. Client-server data exchange or communication must take place on a unified TCP connection.
7. Connections can be scaled vertically.

Characteristics of REST

1. Stateless and Uni-directional
2. Costly
3. It is an upper-level protocol as CRUD operation is involved.
4. Only one communicating party (client/server) needs to be carrying out the communication.
5. Useful when there is heavy request traffic for an application/solution.
6. Requires a new TCP connection each time a request is made.
7. Connections can be horizontally.

Comparison table

Factor	REST	WebSocket
Cost	High (in comparison)	As ports and sockets are involved, it is cost-effective
Information storage	As REST is stateless, no logs, related to request, is saved or stored	Details like session and port details are used
Communication model	Uses Request-Response communication model	Follows Full Duplex communication model
TCP Connection	Each request needs new connection	Same connection is used throughout
Overheads	Essential for every request	There is no need of overheads

Protect your REST APIs:

When is it appropriate to use REST and when is Websocket?

This discussion is not over with talking about differences. It's crucial to understand that this comparison isn't about which is best. It's all about which is suitable. So, you need to find out the scenarios where REST is fitting and where WebSocket is suitable.

Well, we would suggest using REST when the data used is available on an ad-hoc basis. As REST is utterly diverse, every picked or used resource will be delivered. However, this isn't a great choice to make when data should be delivered at blazing speed.

On contrary, WebSocket is great when the application you're trying to develop should miss a single entry in the delivery. For instance, applications demanding tick data analysis. Wherever there is high load or request traffic, WebSocket is ideal.

Now, let's talk about what should you not use WebSockets for.

You shouldn't use WebSocket if you need both vertical and horizontal scaling as it only supports vertical scaling.

Conclusion

Whether you chose REST or WebSocket, one thing is certain your application will be able to communicate properly with other apps/software. You need to pick one based on your communication requirements and expectations. REST is far advanced and secured while WebSocket depends on lower-level components.

We hope that you know the differences between these two and can make a wise choice. Regardless of the choice, adopting stringent and robust API security is a must as it leads to secured data transmission and exchange. It ensures that only secured APIs are used and data transmitted isn't in the reach of threat actors.

FAQ

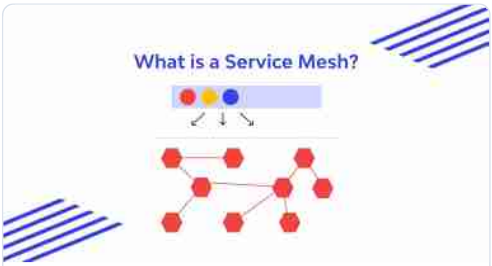
Related Topics

API SECURITY

What is identity and access management?

In this post, we are going to spill every bean related to Identity and Access Management (IAM) practices to use and adopt.

Learn more -->



API SECURITY

What is Service Mesh?

Not sure what Service mesh is how it works and how to use it for your benefit? All such questions are answered well next.

Learn more -->

API SECURITY,

How to start Se Startup?

Adopting sta solutions, their and continual m factors to look i resource perso

Learn more -->

Read next

From the docs

Modern Security Challenges For Financial Organizations

Our recent webinar with the industry overview and product demo.

Watch -->

Wallarm Security Platform. Datasheet

Solution brief on protecting apps and APIs with Wallarm.

Read -->

More Resources →

1. Wallarm Documentation.
2. Tipalti Case Study.

Full Documentation →

wallarm

Wallarm Security Platform	Why Wallarm · API Security Platform · Cloud WAF · Integrations
Solutions by Cloud	AWS · GCP · Azure · Kubernetes
Solutions by Industry	API Security for Healthcare · API Security for Fintech · API Security for Retail · API Security for Technology
Resources	Resource Library · Whitepapers · Datasheets · Case Studies · Webinars · Learning Center · Glossary · Support
Featured Resources	7 Steps to Kubernetes Security · Securing Apps on AWS with Wallarm · Top Five Challenges in Protecting APIs · Report: Path To DevOps Success

Evolution of Real Time Attack Detection What's Wrong with RASP?

Securing Apps on AWS with Wallarm A CISO's Guide To Cloud Application Security

Learn Wallarm

Documentation API specs Terraform Provider

Terms of Services Privacy Policy Cookies Policy Security Bug Bounty Program

Software License Agreement Service Level Agreement ->Cookies Settings

188 King St. Unit 508,
San Francisco, CA, 94107
(415) 940-7077

request@wallarm.com