Web Sockets Attacks

⋮ 25/08/2022

Aug 24, 2022

5 min read
WS Web-Notes

Websockets

Refrences

What is WebSocket in Arabic

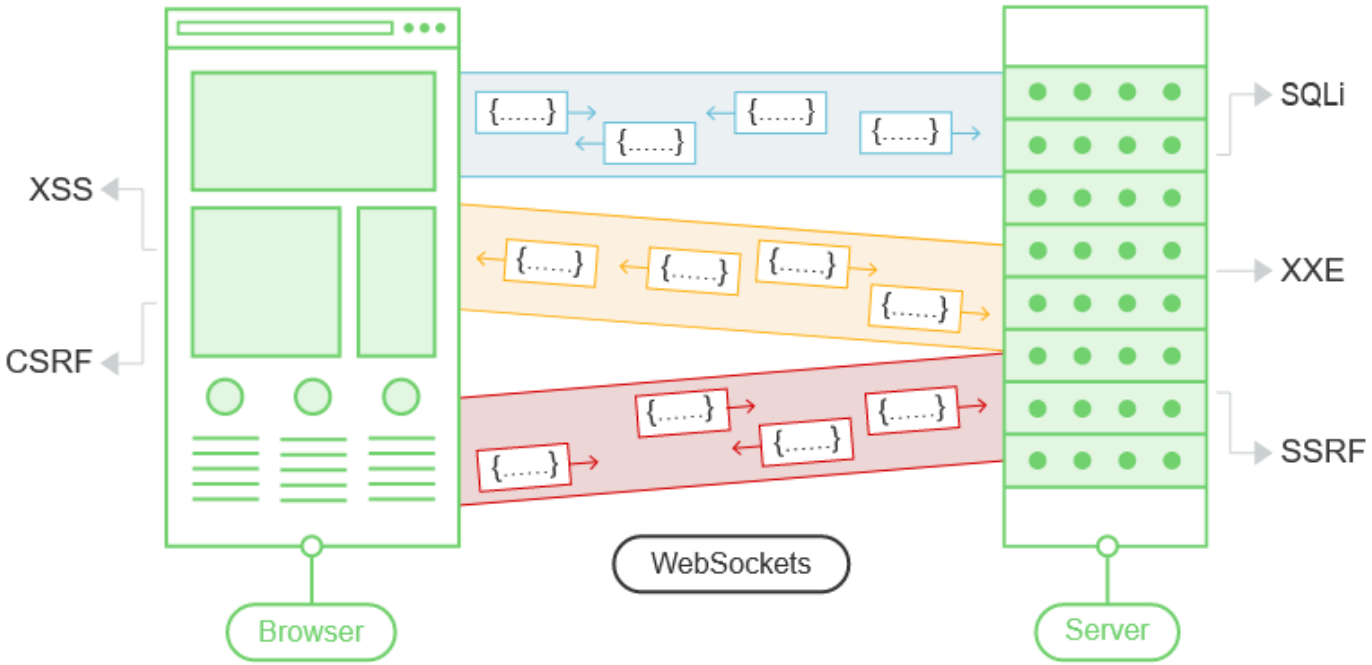
What are WebSockets? | Web Security Academy

How to Exploit WebSocket | WebSocket for Beginners

WebSockets are widely used in modern web applications. They are initiated over HTTP and provide long-lived connections with asynchronous communication in both directions.

WebSockets are used for various purposes, including user actions and transmitting sensitive information.

Virtually any web security vulnerability that arises with regular HTTP can also arise in relation to WebSockets communications.



Manipulating WebSocket traffic

Finding WebSockets security vulnerabilities generally involves manipulating them in ways that the application doesn't expect. You can do this using Burp Suite.

Intercepting and modifying WebSocket messages

How to test WebSockets with Burp Suite

Replaying and generating new WebSocket messages

- In Burp Proxy, select a message in the WebSockets history, or in the Inte or b, and choose "Send to Repeater" from the context menu.
- In Burp Repeater, you can now edit the message that was selected, and send it over and over.
- You can enter a new message and send it in either direction, to the client or server.
- In the "History" panel within Burp Repeater, you can view the history of messages that have been transmitted over the WebSocket connection. This includes messages that you have generated in Burp Repeater, and also any t and anye browser or server via the same connection.
- If you want to edit and resend any message in the history panel, you can do this by selecting the message and choosing "Edit and resend" from the context menu.

Manipulating WebSocket connections

What are WebSockets? | Web Security Academy

There are various situations in which manipulating the WebSocket handshake might be necessary:

- It can enable you to reach more attack surface.

- Some attacks might cause your connection to drop so you need to establish a new one.
- Tokens or other data in the original handshake request might be stale and need updating.

You can manipulate the WebSocket handshake using Burp Repeater:

- Send a WebSocket message to Burp Repeater.
- In Burp Repeater, click on the pencil icon next to the WebSocket URL. This opens a wizard that lets you attach to an existing connected WebSocket, clone a connected WebSocket, or reconnect to a disconnected WebSocket.
- If you choose to clone a connected WebSocket or reconnect to a disconnected WebSocket, then the wizard will show full details of the WebSocket handshake request, which you can edit as required before the handshake is performed.
- When you click “Connect”, Burp will attempt to carry out the configured handshake and display the result. If a new WebSocket the connection was successfully established, you can then use this to send new messages in Burp Repeater.

WebSockets security vulnerabilities

In principle, practically any web security vulnerability might arise in relation to WebSockets:

- User-supplied input transmitted to the server might be processed in unsafe ways, leading to vulnerabilities such as SQL injection or XML external entity injection.
- Some blind vulnerabilities reached via WebSockets might only be detectable using out-of-band (OAST) techniques.
- If attacker-controlled data is transmitted via WebSockets to other application users, then it might lead to XSS or other client-side vulnerabilities.

Manipulating WebSocket messages to exploit vulnerabilities

The majority of`input-based` of inequalities affecting WebSockets can be found and exploited by tampering with the contents of WebSocket messages

For example, suppose a chat application uses WebSockets to send chat messages between the browser and the server. When a user types a chat message, a WebSocket message like the following is sent to the server:

```
{"message":"Hello Carlos"}
```

The contents of the message are transmitted (again via WebSockets) to another chat user, and rendered in the user’s browser as follows:

```
<td>Hello Carlos</td>
```

In this situation, provided no other input processing or defenses are in play, an attacker can perform a proof-of-concept XSS attack by submitting the following WebSocket message:

```
{"message":"<img src=1 onerror='alert(1)'>"}
```

Lab: Manipulating WebSocket messages to exploit vulnerabilities | Web Security Academy

---

Manipulating the WebSocket handshake to exploit vulnerabilities

Some WebSockets vulnerabilities can only be found and exploited by manipulating the WebSocket handshake. These vulnerabilities tend to involve design flaws, such as:

- Misplaced trust in HTTP headers to perform security decisions, such as the `X-Forwarded-For` header.
- Flaws in session handling mechanisms, since the session context in which WebSocket messages are processed, is generally determined by the session context of the handshake message.
- Attack surface is introduced by custom HTTP headers used by the application.

Lab: Manipulating the WebSocket handshake to exploit vulnerabilities | Web Security Academy

---

Using cross-site WebSockets to exploit vulnerabilities

Some WebSockets security vulnerabilities arise when an attacker makes a cross-domain WebSocket connection from a website that the attacker controls.

This is known as a cross-site WebSocket hijacking attack, and it involves exploiting a cross-site request forgery (CSRF) vulnerability on a WebSocket handshake. The attack often has a serious impact, allowing an attacker to perform privileged actions on behalf of the victim user or capture sensitive data to which the victim user has access.

Cross-site WebSocket hijacking | Web Security Academy

How to secure a WebSocket connection

To minimize the risk of security vulnerabilities arising with WebSockets, use the following guidelines:

- Use the `wss://` protocol (WebSockets over TLS).
- Hard code the URL of the WebSockets endpoint, and certainly don't incorporate user-controllable data into this URL.
- Protect the WebSocket handshake message against CSRF, to avoid cross-site WebSockets hijacking vulnerabilities.
- Treat data received via the WebSocket as untrusted in both directions. Handle data safely on both the server and client ends, to prevent input-based vulnerabilities such as SQL injection and cross-site scripting.