Google Developer Student Clubs

Web Application Security

```
lookup.KeyValue
f.constant(['em
=tf.constant([0
.lookup.StaticV

_buckets=5)
```

Gianluca Piccirillo - U3 Software Engineering

Nabil Amimer - U3 Computer Engineering

Why it matters?

1. User information protection
2. Securing your web servers
3. Legal and regulatory compliance
4. Defend against threat actors
5. Peace of mind

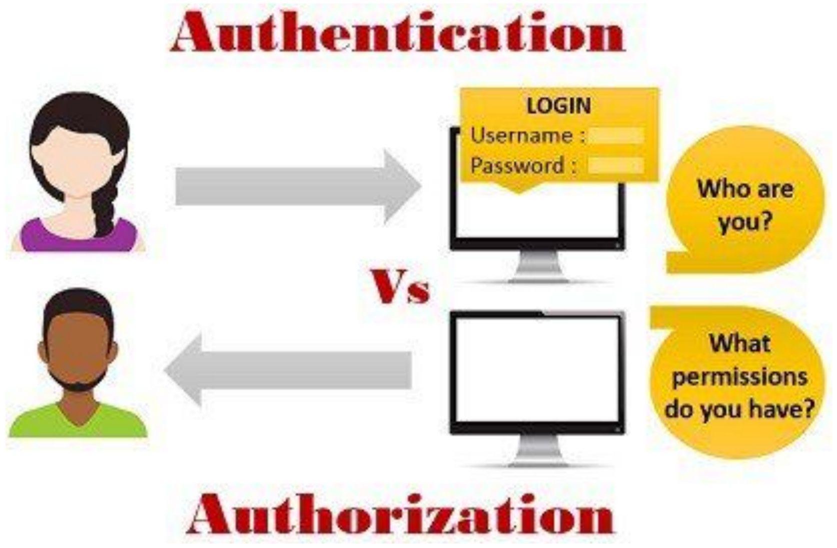
More importantly, it enables you to deploy your applications to production.

Authentication, Authorization and Access Control

Authentication is verifying an Identity (user, application, process, device…).

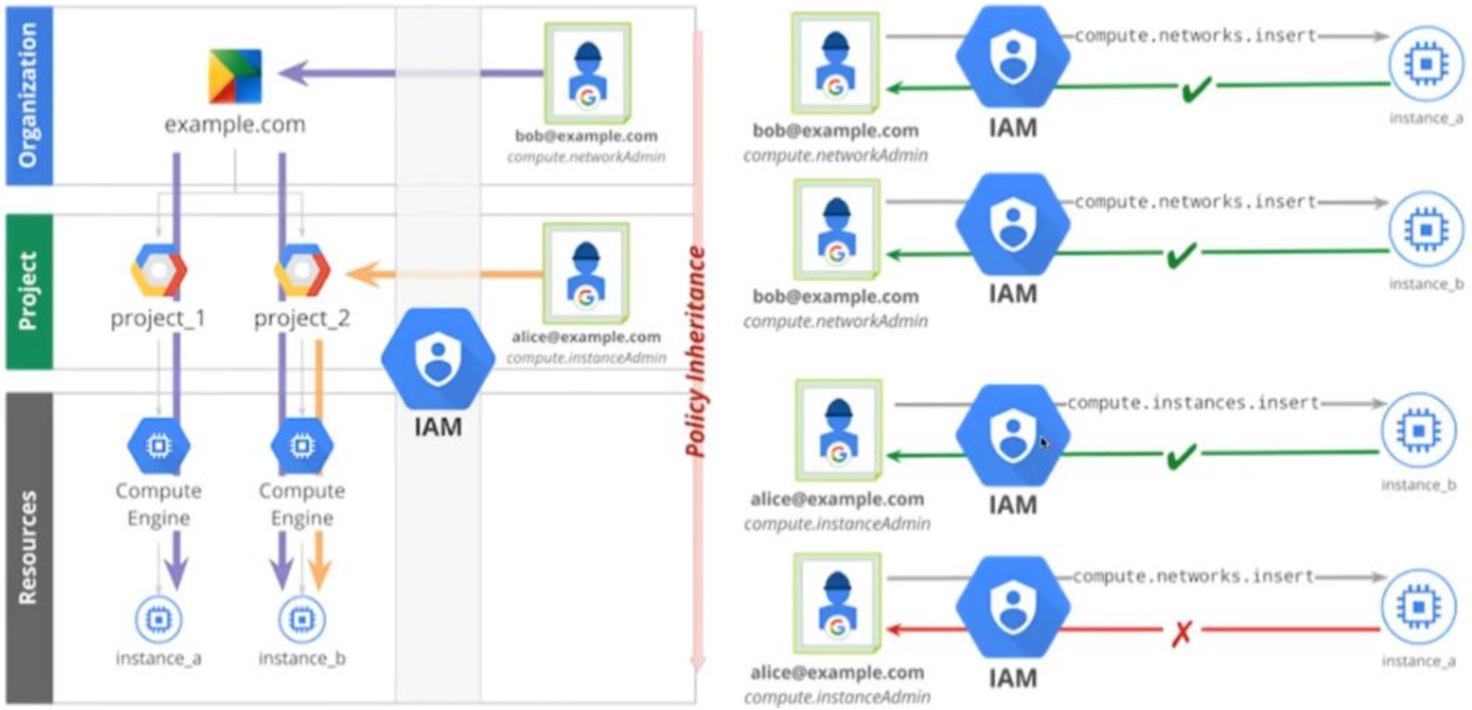
Authorization is determining permissions for an authenticated identity.

Access Control is the process of managing authorization within an application or in an organization. (Ex. Role based access control RBAC)



https://www.starwindsoftware.com/blog/identity-and-access-management-iam-in-a-nutshell

Access Control Models (RBAC example)



https://cloudinfrastructureservices.co.uk/designing-azure-subscription-vs-resource-groups-best-practices/

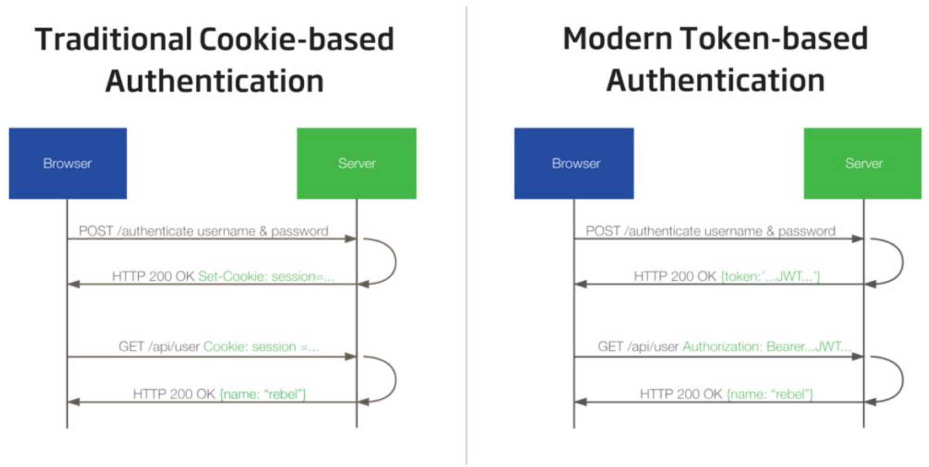
Authentication Design Patterns

Server side session authentication:

A session token is stored on the browser. The server holds session data related to the user (associated with token). Usually stored in browser cookies.

Token based authentication:

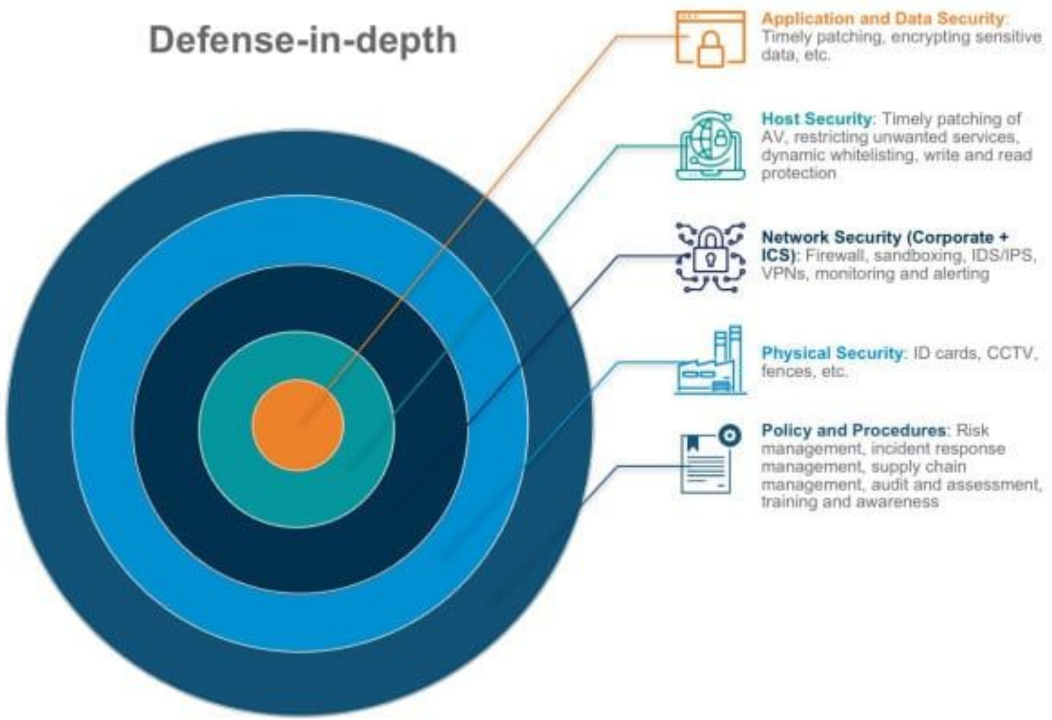
A token is issued upon login and stored on the browser. Token usually contains session data about the user itself. Usually stored in local storage and sent in the “Authorization” HTTP header.



Traditional Cookie-based Authentication
Browser
Server
POST /authenticate username & password
HTTP 200 OK Set-Cookie: session=...
GET /api/user Cookie: session =...
HTTP 200 OK {name: "rebel"}
Modern Token-based Authentication
Browser
Server
POST /authenticate username & password
HTTP 200 OK {token:'...JWT...'}
GET /api/user Authorization: Bearer...JWT...
HTTP 200 OK {name: "rebel"}

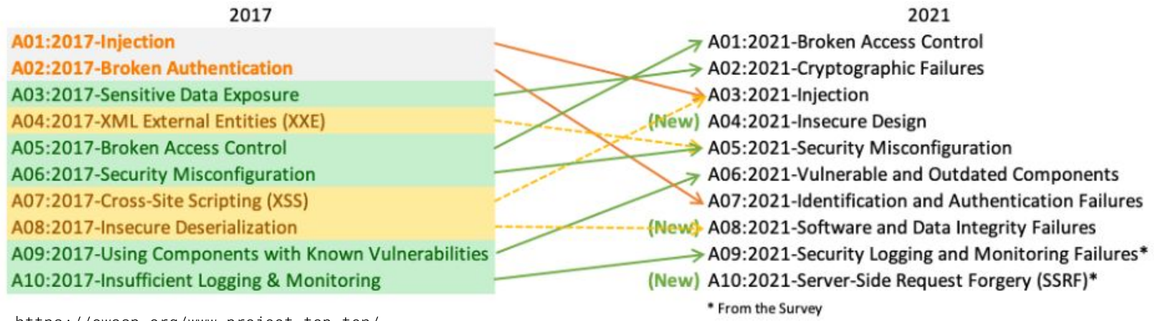
Where are we in the bigger picture?

- Securing on more levels means better security
- Today, we're going to be talking about App Security


Defense-in-depth
Application and Data Security: Timely patching, encrypting sensitive data, etc.
Host Security: Timely patching of AV, restricting unwanted services, dynamic whitelisting, write and read protection
Network Security (Corporate + ICS): Firewall, sandboxing, IDS/IPS, VPNs, monitoring and alerting
Physical Security: ID cards, CCTV, fences, etc.
Policy and Procedures: Risk management, incident response management, supply chain management, audit and assessment, training and awareness

Common Web App Attack Vectors

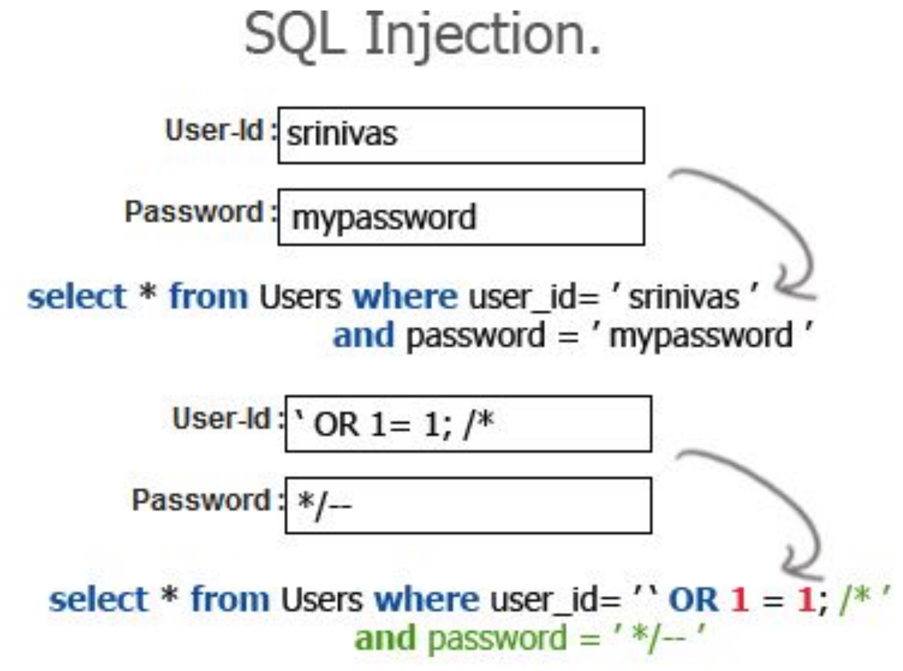
- Code injections
  - SQL injection (SQLi)
  - Cross-site scripting (XSS)
  - Remote code execution (RCE)
- Request forgeries
  - Cross-site request forgery (CSRF)
  - Server side request forgery (SSRF)
- File Inclusions
  - Local file inclusions (LFI)
  - Remote file inclusions (RFI)



2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

https://owasp.org/www-project-top-ten/

SQL Injection Simple Example



SQL Injection.

User-Id : srinivas

Password : mypassword

```
select * from Users where user_id= ' srinivas '
                      and password = ' mypassword '
```

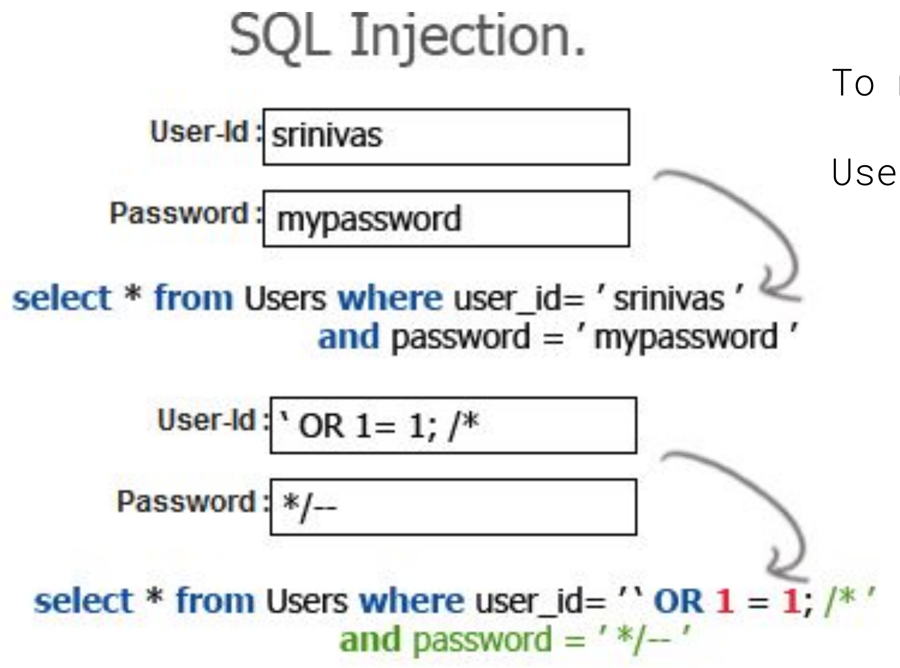
User-Id : ` OR 1= 1; /*

Password : */--

```
select * from Users where user_id= '` OR 1 = 1; /* '
                      and password = ' */-- '
```

https://hararei.com/pages/sql-injection-protection.html

SQL Injection Simple Example



srinivas

mypassword

```
select * from Users where user_id= ' srinivas '
        and password = ' mypassword '
```

` OR 1= 1; /*

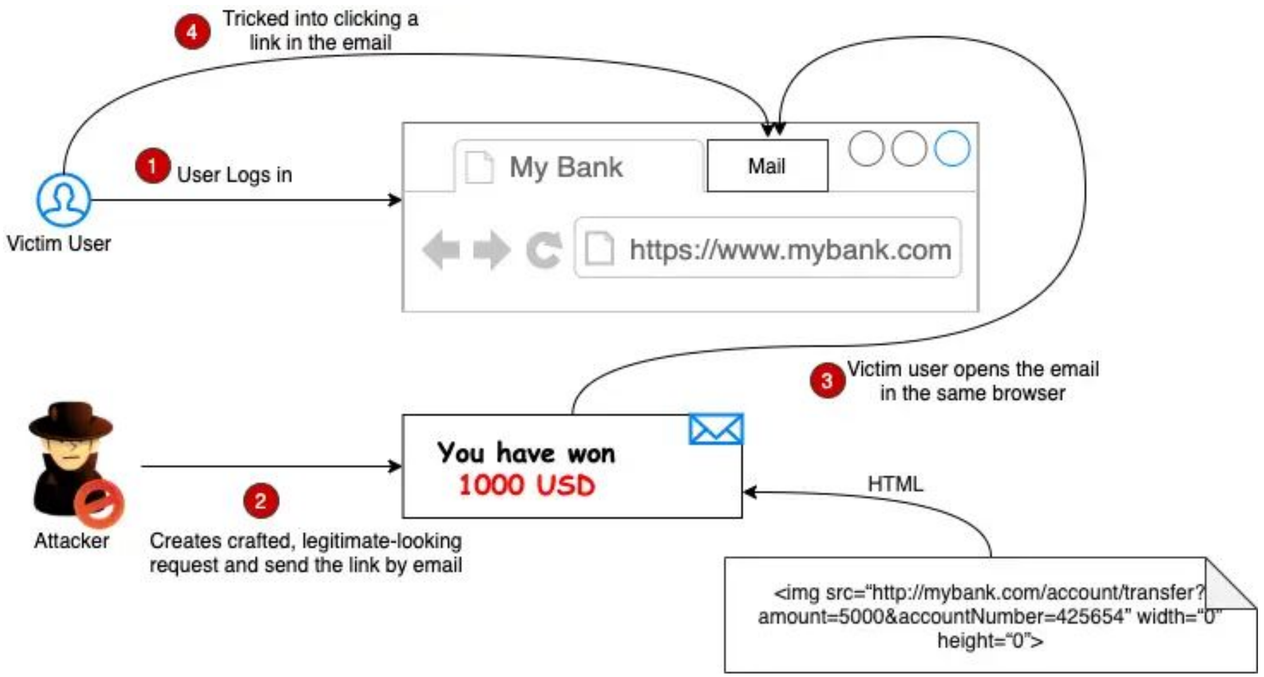
*/--

```
select * from Users where user_id= '` OR 1 = 1; /* '
        and password = ' */-- '
```

To mitigate this:

Use prepared statements!

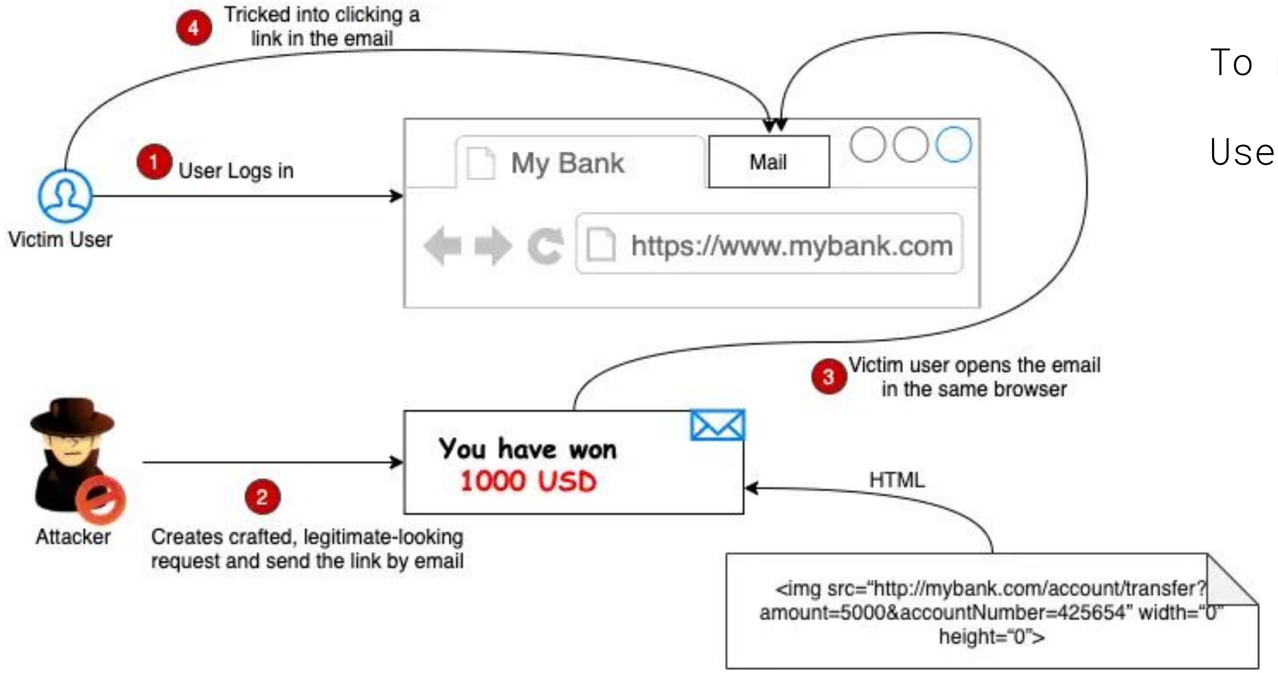
CSRF Simple Example


4 Tricked into clicking a link in the email
1 User Logs in
Victim User
My Bank
Mail
https://www.mybank.com
3 Victim user opens the email in the same browser
Attacker
You have won 1000 USD
HTML
2 Creates crafted, legitimate-looking request and send the link by email

```
<img src="http://mybank.com/account/transfer?amount=5000&accountNumber=425654" width="0" height="0">
```

https://reflectoring.io/complete-guide-to-csrf/

CSRF Simple Example



4 Tricked into clicking a link in the email
1 User Logs in
Victim User
My Bank
Mail
https://www.mybank.com
3 Victim user opens the email in the same browser
Attacker
You have won 1000 USD
HTML
2 Creates crafted, legitimate-looking request and send the link by email

```
<img src="http://mybank.com/account/transfer?amount=5000&accountNumber=425654" width="0" height="0">
```

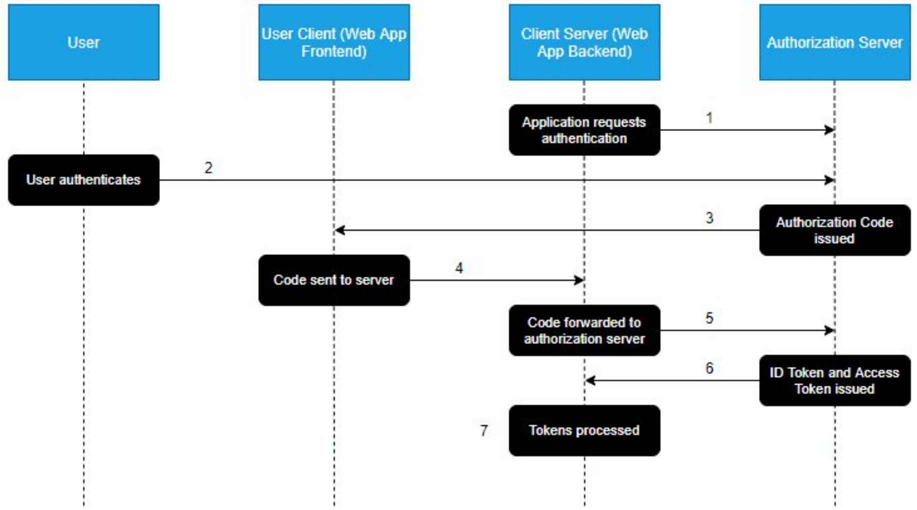
https://reflectoring.io/complete-guide-to-csrf/

To mitigate this:

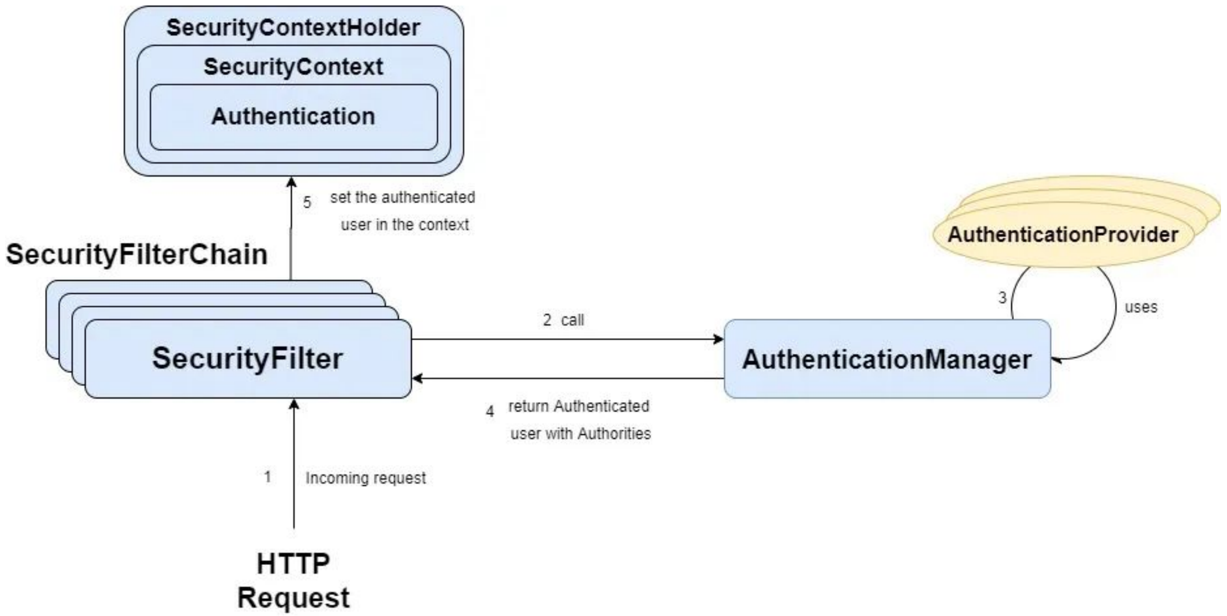
Use CSRF protection

OAuth 2.0 Authorization Framework

OAuth 2.0 (Open Authorization 2.0) is an industry-standard authorization framework that allows applications to access user data on behalf of the user without exposing their credentials. It is widely used for securing APIs, enabling Single Sign-On (SSO), and delegating access to third-party applications. OAuth 2.0 is designed to be flexible, extensible, and suitable for a wide range of use cases.


User
User Client (Web App Frontend)
Client Server (Web App Backend)
Authorization Server
Application requests authentication
1
User authenticates
2
3
Authorization Code issued
Code sent to server
4
Code forwarded to authorization server
5
6
ID Token and Access Token issued
7
Tokens processed

Spring Security Overview



https://medium.com/@haytambenayed/how-does-spring-security-work-internally-525d359d7af

Demo

Time for a demo!

Let's build and secure a REST API with Spring Security

Github: https://github.com