Cross Site Request Forgery (CSRF)

Earlier in the course …

Earlier in the course …

Step 3: Create custom login form

Best Practice

- **Best practice** is to use the Spring MVC Form tag `<form:form>`
  - Provides automatic support for security defenses *(more on this later)*

```
<form:form action="…" method="…" >
   …
</form:form>
```

Earlier in the course …

Step 3: Create custom login form

- **Best practice** is to use the Spring MVC Form tag `<form:form>`
  - Provides automatic support for security defenses *(more on this later)*

```
<form:form action="…" method="…" >
   …
</form:form>
```

Spring Security

Spring Security

- Spring Security protects against **Cross-Site Request Forgery (CSRF)**

What is CSRF?

What is CSRF?

A security attack where an evil website
tricks you into executing an action
on a web application
that you are currently logged in.

CSRF Examples

CSRF Examples

- You are logged into your banking app

CSRF Examples

- You are logged into your banking app
  - tricked into sending money to another person

CSRF Examples

- You are logged into your banking app
  - tricked into sending money to another person

- You are logged into an e-commerce app

CSRF Examples

- You are logged into your banking app
  - tricked into sending money to another person

- You are logged into an e-commerce app
  - tricked into purchasing unwanted items

CSRF Protection

CSRF Protection

- To protect against CSRF attacks

CSRF Protection

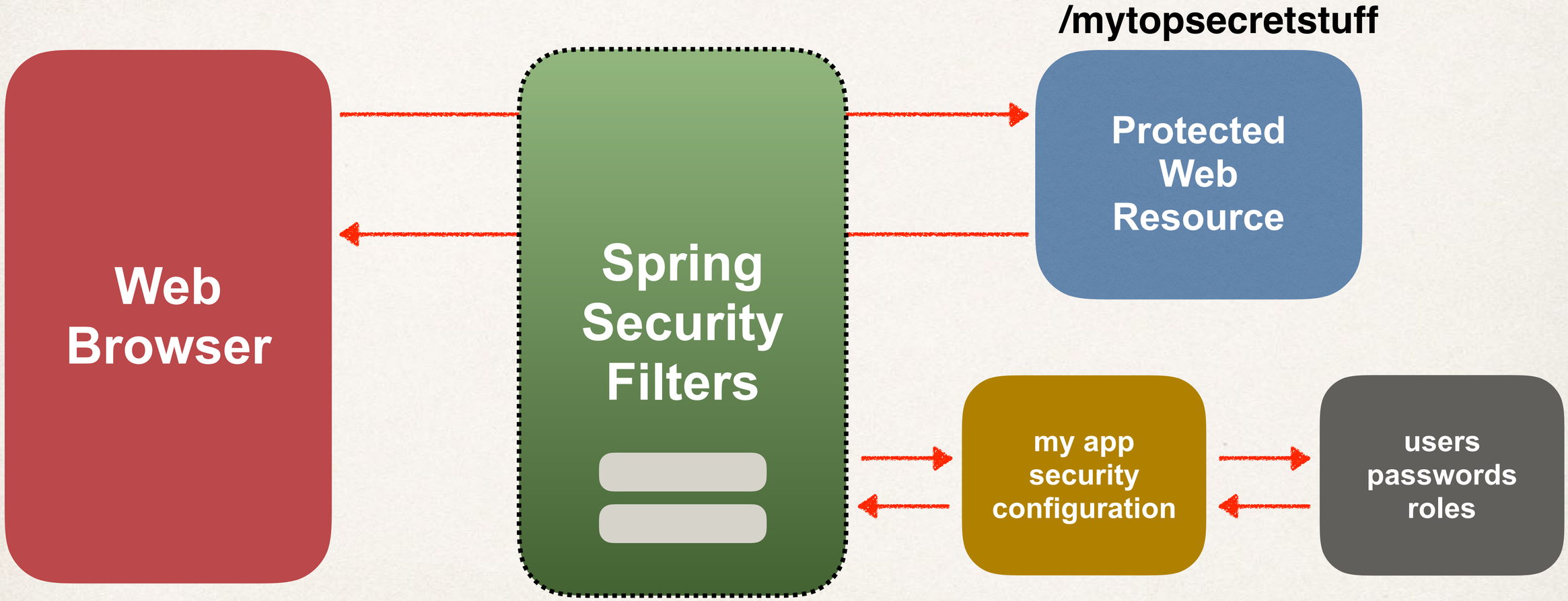
- To protect against CSRF attacks

- Embed additional authentication data / token into all HTML forms

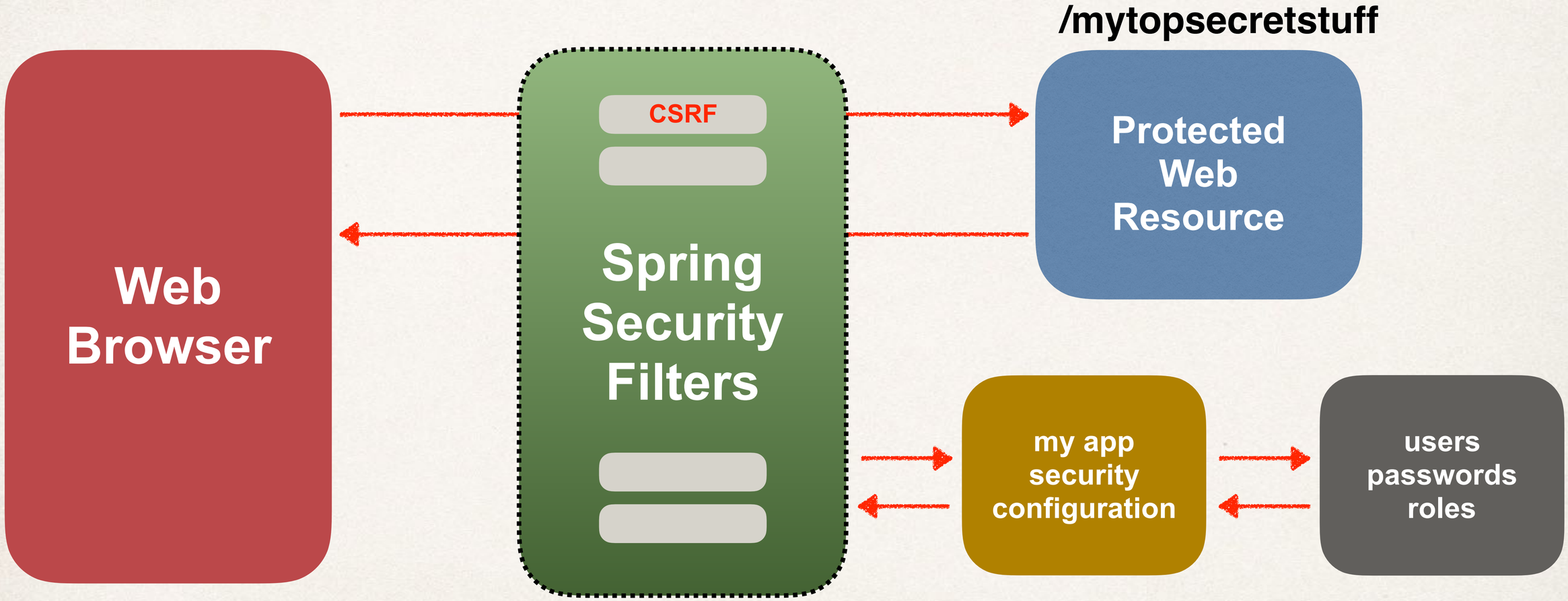
CSRF Protection

- To protect against CSRF attacks
- Embed additional authentication data/token into all HTML forms
- On subsequent requests, web app will verify token before processing

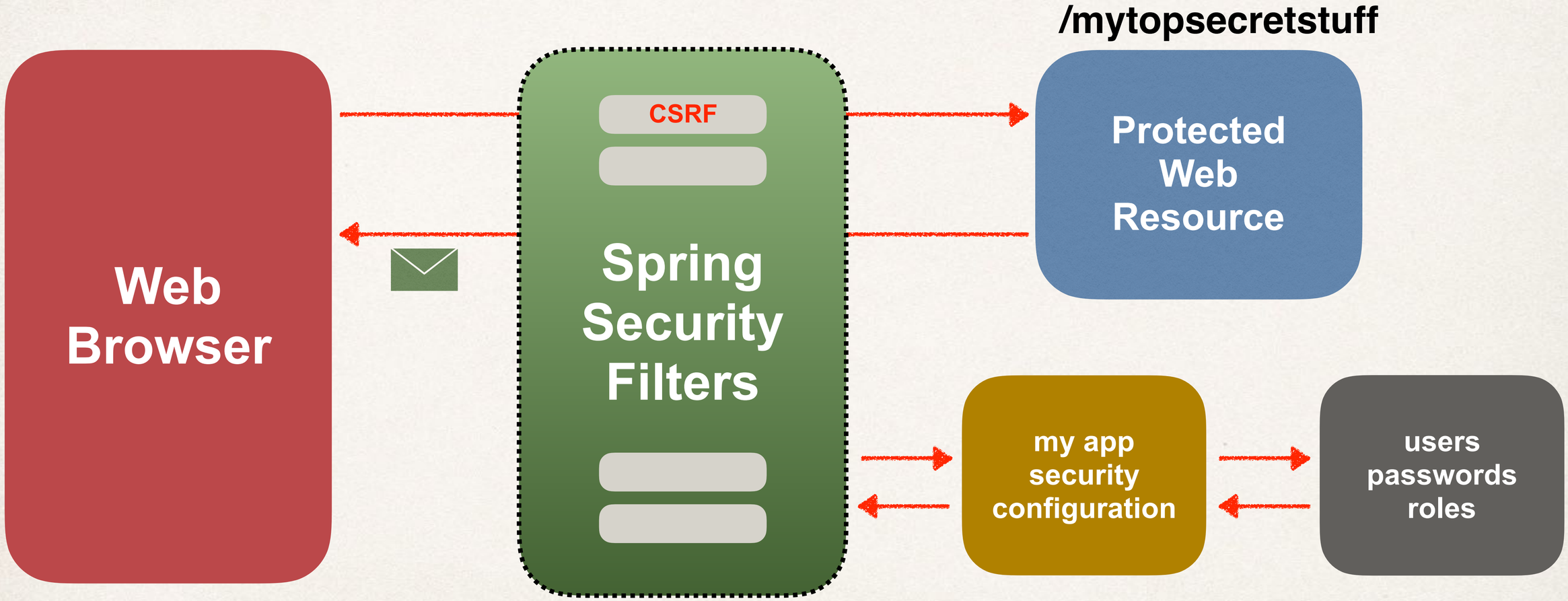
Spring Security Overview



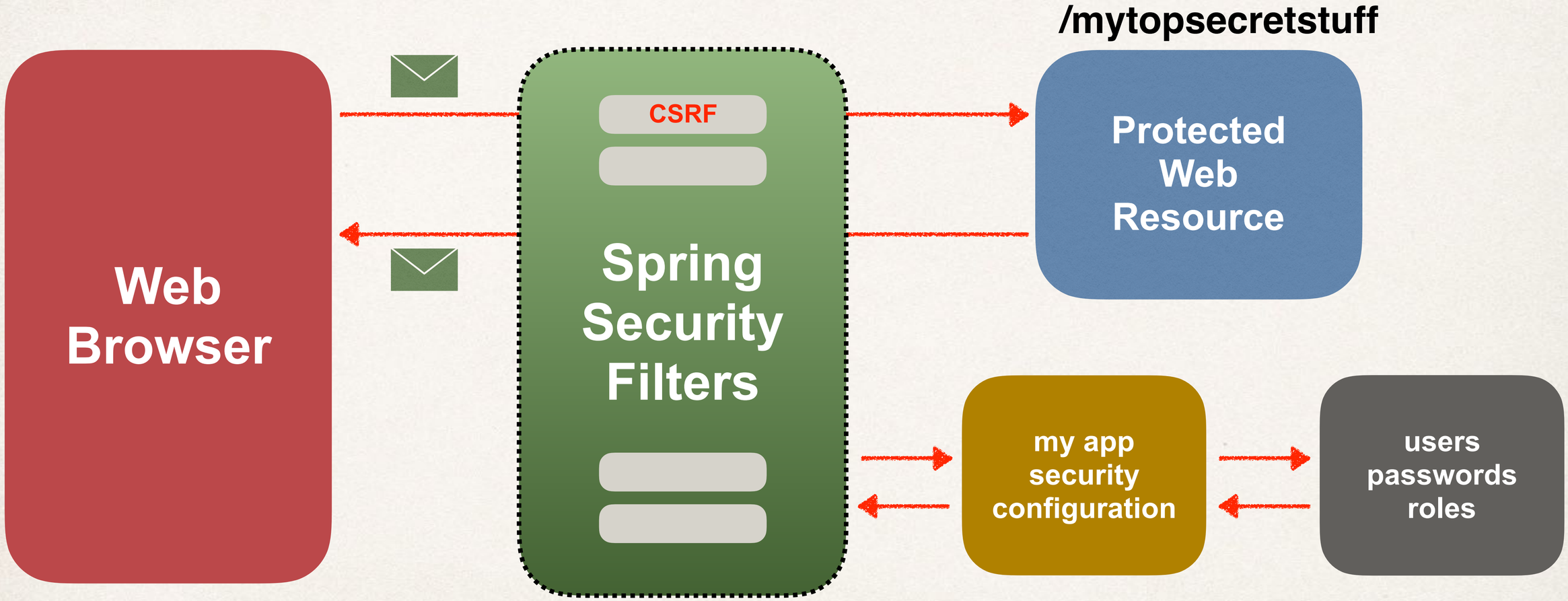
Spring Security Overview



Spring Security Overview



Spring Security Overview



Spring Security's CSRF Protection

Spring Security's CSRF Protection

- CSRF protection is enabled by default in Spring Security

Spring Security's CSRF Protection

- CSRF protection is enabled by default in Spring Security

- Spring Security uses the Synchronizer Token Pattern

Spring Security's CSRF Protection

- CSRF protection is enabled by default in Spring Security

- Spring Security uses the Synchronizer Token Pattern
  - Each request includes a session cookie and randomly generated token

Spring Security's CSRF Protection

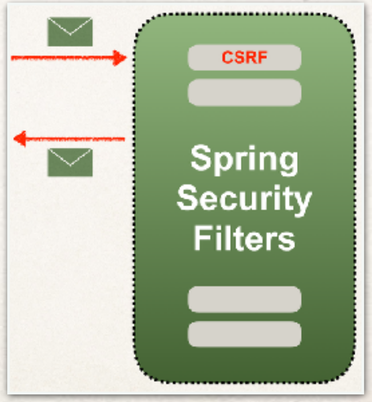
- CSRF protection is enabled by default in Spring Security
- Spring Security uses the Synchronizer Token Pattern
  - Each request includes a session cookie and randomly generated token
- For request processing, Spring Security verifies token before processing

Spring Security's CSRF Protection

- CSRF protection is enabled by default in Spring Security
- Spring Security uses the Synchronizer Token Pattern
  - Each request includes a session cookie and randomly generated token
- For request processing, Spring Security verifies token before processing
- All of this is handled by Spring Security Filters

Spring Security's CSRF Protection

- CSRF protection is enabled by default in Spring Security
- Spring Security uses the Synchronizer Token Pattern
  - Each request includes a session cookie and randomly generated token
- For request processing, Spring Security verifies token before processing
- All of this is handled by Spring Security Filters



When to use CSRF Protection?

When to use CSRF Protection?

- The Spring Security team recommends

When to use CSRF Protection?

- The Spring Security team recommends

- Use CSRF protection for any normal browser web requests

When to use CSRF Protection?

- The Spring Security team recommends
- Use CSRF protection for any normal browser web requests
- If you are building a service for non-browser clients

When to use CSRF Protection?

- The Spring Security team recommends
- Use CSRF protection for any normal browser web requests
- If you are building a service for non-browser clients
  - you *may* want to disable CSRF protection (after careful review)

Use Spring Security CSRF Protection

Use Spring Security CSRF Protection

- For form submissions use POST instead of GET

Use Spring Security CSRF Protection

- For form submissions use POST instead of GET

- Include CSRF token in form submission

Use Spring Security CSRF Protection

- For form submissions use POST instead of GET
- Include CSRF token in form submission
- `<form:form>` automagically adds CSRF token

Use Spring Security CSRF Protection

- For form submissions use POST instead of GET
- Include CSRF token in form submission
- `<form:form>` automagically adds CSRF token

- If you don't use `<form:form>`, you must manually add CSRF token

Manually add CSRF token

```
<form action="..." method="POST">

        <input type="hidden"
               name="${_csrf.parameterName}"
              value="${_csrf.token}" />

</form>
```

What happens if you don't include CSRF token?

What happens if you don't include CSRF token?

What happens if you don't include CSRF token?

Let's Break It!

HTTP Status 403 – Forbidden

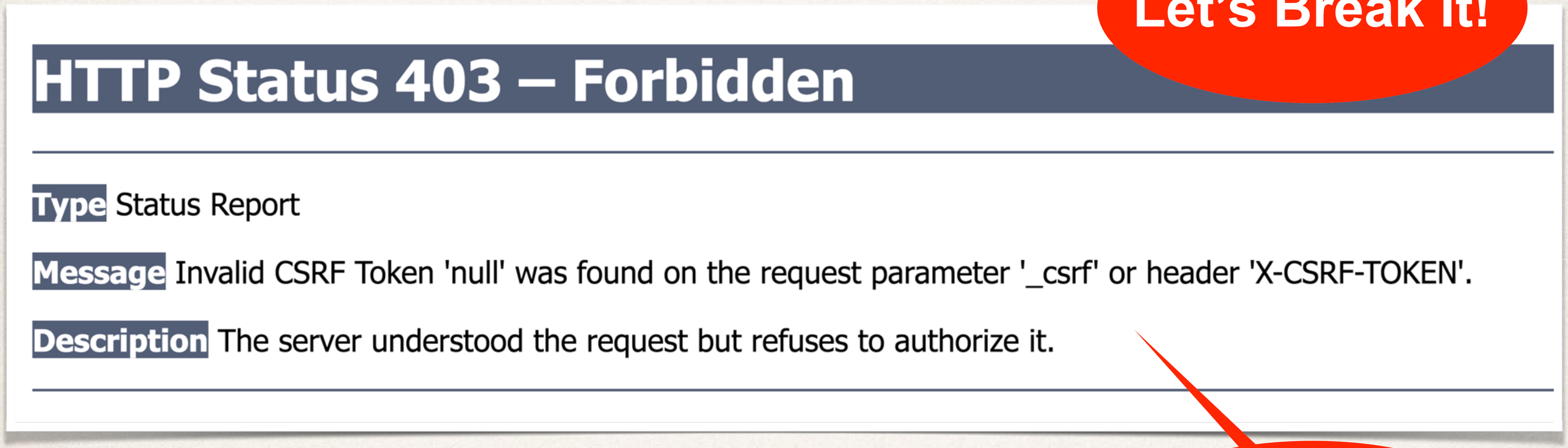
Type Status Report

Message Invalid CSRF Token 'null' was found on the request parameter '_csrf' or header 'X-CSRF-TOKEN'.

Description The server understood the request but refuses to authorize it.

What happens if you don't include CSRF token?

Let's Break It!



Token was not included

CSRF Resources

- CSRF Security Reference
  - `https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)`
- Spring Security CSRF Support
  - `https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/#csrf`