Django 7: Security

IN608: Intermediate Application Development Concepts

Last Session’s Content

- Django authentication
  - AbstractUser
  - Auth views
- Django crispy forms
- Session-based authentication

Today’s Content

- Security in Django
    - XSS protection
    - CSRF protection
    - SQL injection protection
    - Clickjacking protection
    - Secret key

Security in Django

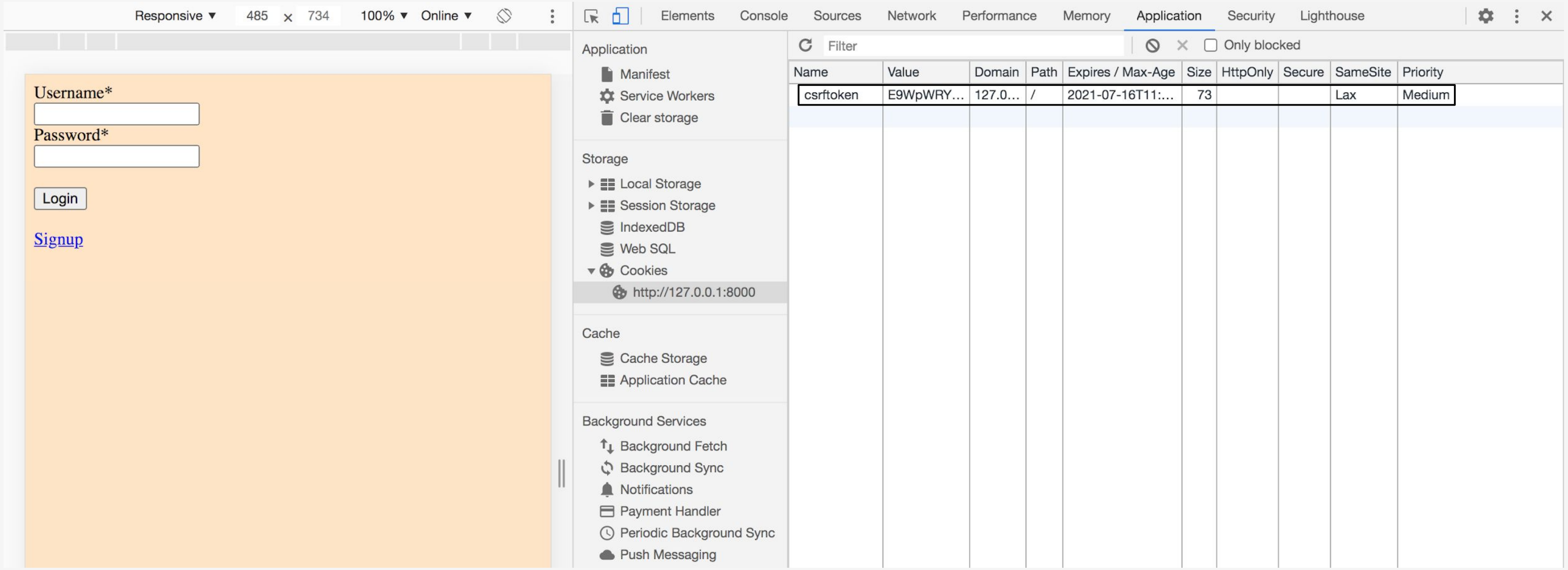
XSS Protection

- XSS (cross-site scripting) attacks are a type of injection
- An attacker can use XSS to send & execute a malicious script to an unsuspecting user
- The user’s browser has no way to know that the script should not be trusted
- The malicious script can access cookies, session tokens & other sensitive data
- These scripts can rewrite the content of an HTML page
- Django templates protect against majority of XSS attacks
- Resource: https://owasp.org/www-community/attacks/xss

CSRF Protection

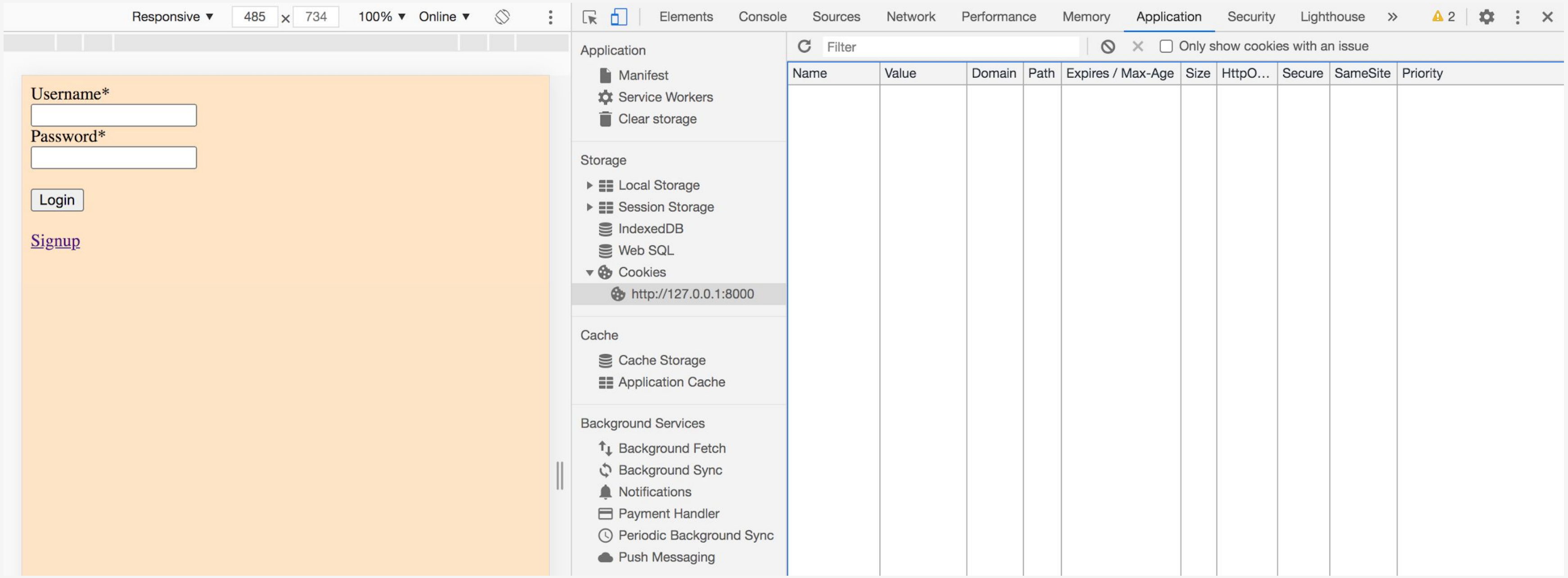
- CSRF (cross-site request forgery) is an attack that allows a malicious user to execute actions using the credentials of an unsuspecting user
- Django has in-built protection against most types of CSRF attacks - `Csrf View` middleware
- CSRF protection works by checking for a token in each POST request
- It ensures that a malicious user cannot replay a form POST to your site & have another logged in user submit that form
- `{% csrf_token %}`
- Resources:
    - https://docs.djangoproject.com/en/3.0/ref/csrf
    - https://owasp.org/www-community/attacks/csrf

CSRF Protection



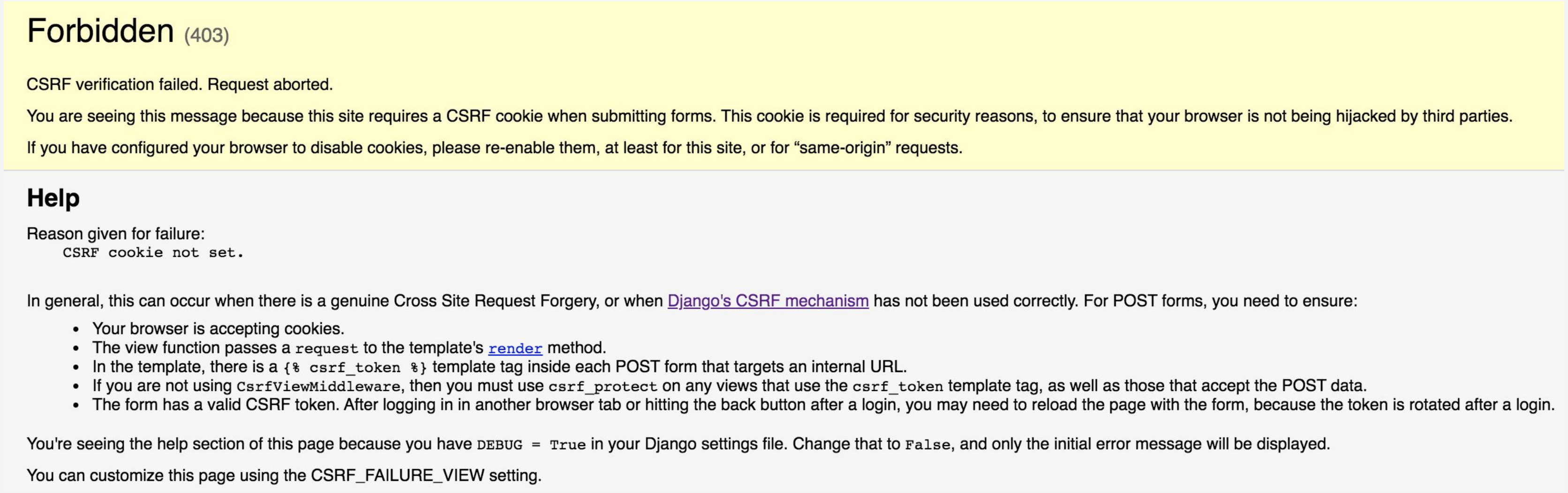
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Priority
csrftoken	E9WpWRY…	127.0…	/	2021-07-16T11:…	73			Lax	Medium

CSRF Protection



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpO…	Secure	SameSite	Priority

CSRF Protection



Forbidden (403)

CSRF verification failed. Request aborted.

You are seeing this message because this site requires a CSRF cookie when submitting forms. This cookie is required for security reasons, to ensure that your browser is not being hijacked by third parties.

If you have configured your browser to disable cookies, please re-enable them, at least for this site, or for “same-origin” requests.

Help

Reason given for failure:

```
    CSRF cookie not set.
```

In general, this can occur when there is a genuine Cross Site Request Forgery, or when Django's CSRF mechanism has not been used correctly. For POST forms, you need to ensure:

- Your browser is accepting cookies.
- The view function passes a `request` to the template's `render` method.
- In the template, there is a `{% csrf_token %}` template tag inside each POST form that targets an internal URL.
- If you are not using `CsrfViewMiddleware`, then you must use `csrf_protect` on any views that use the `csrf_token` template tag, as well as those that accept the POST data.
- The form has a valid CSRF token. After logging in in another browser tab or hitting the back button after a login, you may need to reload the page with the form, because the token is rotated after a login.

You're seeing the help section of this page because you have `DEBUG = True` in your Django settings file. Change that to `False`, and only the initial error message will be displayed.

You can customize this page using the CSRF_FAILURE_VIEW setting.

SQL Injection Protection

- SQL injection allows a malicious user to execute arbitrary SQL code against a database
- Results in access to sensitive data, data modification & execution of admin database operations
- Django querysets are protected from SQL injection
- Queries are constructed using query parameterization
- Resource: https://owasp.org/www-community/attacks/SQL_Injection

Clickjacking Protection

- Clickjacking is a type of attack where a malicious site wraps another site in a frame
- Results in an unsuspecting user being tricked into performing unintended actions
- In Django, `X-Frame Options` middleware protects against clickjacking
- Resources:
  - https://docs.djangoproject.com/en/3.0/ref/clickjacking/#clickjacking-prevention
  - https://owasp.org/www-community/attacks/Clickjacking

Secret Key

- `mvt/settings.py`
- `SECRET_KEY`
- Used to provide cryptographic signing
- Should be set to a unique & unpredictable value
- `django-admin startproject` automatically adds a randomly generated `SECRET_KEY` to each new project
- Resource: https://docs.djangoproject.com/en/3.0/ref/settings/#std:setting-SECRET_KEY

Additional Reading

- OWASP top 10 web app security risks - https://owasp.org/www-project-top-ten