Bezbednosni mehanizmi

Same-Origin Policy i CORS mehanizam

CORS

- Skraćeno od *Cross-Origin Resource Sharing*

- Jeste li sretali greške u vezi sa CORS mehanizmom?
- Čemu služi?
- Ko/šta sprovodi *Same-Origin Policy i CORS mehanizam*?

Deljenje resursa

- Resurs - adržaj kojeg klijent može zatražiti od web servera:
  - HTML
  - CSS
  - Fontovi
  - Mediji (slike, GIF, video, audio)
  - JSON, XML
  - ...

- Šta kada web stranica zatraži resurs?
  - Ako je poreklo stranice i resursa(*origin*) isto, *browser* će uvek dopustiti pristup!
  - Ako poreklo stranice i resursa nije isto, browser će se zapitati...

Problem i *Same-Origin Policy*

- Maliciozna web stranica može (pokušati) iskoristiti resurse drugih, benignih web stranica

- Primeri - kada pristup ne bi bio kontrolisan:
  - Krađa ličnih podataka (resursa) koji su možda i zaštićeni (browser sam pridoda *cookie* prilikom pristupa, ako je korisnik ulogovan)
  - Pozivanje API metoda (gde bi to, opet, mogli biti iskorišteni *cookies*)
  - *Port-scanning* – pretraga privatnih (lokalno-dostupnih). Npr. pristup localhost:4003 da bi se videlo je li aktivan

- Stoga je definisana politika (*policy*) pristupa koju sprovodi browser (*user-agent*) shodno poreklu resursa (*origin*). Principi rukovanja *origin*-om su opisani u IETF 6454.
- CORS mehanizam možete smatrati relaksacijom *Same-Origin* politike.

Kako definišemo poreklo (*origin*)?

- *Origin* je definisan trojkom: šema, celi hostname, port

- Šema: neretko se mapira na protokol, ali je širi koncept definisan RFC standardom za URI; Primeri: file://, data://, http://, https://
- Celi *hostname*; Dakle, example.com i sub.example.com *hosts* nisu isti!
- *Port* – neretko implicitno definisan protokolom – 80, 443… Ali! Niko vas ne sprečava da uradite ovakvo nešto: https://example.com:80

Primeri *origin*-a

- Stranica: https://example.com/loc/bar.html
  Da li će se za sledeće resurse smatrati da je origin isti?

- https://example.com/baz/foo.html ✓
- https://sub.example.com/baz/qux.html ✗
- http://example.com/baz/qux.html ✗
- https://example.com:8080/baz/foo.html ✗

Detalji CORS mehanizma

- CORS mehanizam prepozaje tri različite kategorije pristupa resursima:
  - **Cross-origin writes**; linkovi, redirect, forme. Podrazumevano *dozvoljeni* zahtevi. Međutim, iako su writes dozvoljeni, ne znači da će svi zahtevi u potpunosti proći. Takođe, moramo upoznati ***preflight*** zahteve.
  - **Cross-origin embeds**; Resursi koji se učitavaju pomoću tagova: *img, video, script, iframe*... Podrazumevano *dozvoljeni*. *Iframe* je, naravno, poseban, te se on konfiguriše pomoću *X-Frame-options header*-a.
  - **Cross-origin reads;** Resursi dobavljeni **AJAX (fetch)** zahtevima. Podrazumevano *zabranjeni*.

Primeri

Primer 1:

- Cross-origin writes - obični linkovi, forme...
- Cross-origin embeds – slike...

Primer 2 i Primer 3: Kako stvari stoje sa AJAX pozivima?

AJAX pozivi

- Razlikujemo:
  - *Jednostavne* AJAX pozive, kod kojih ***preflight request*** nije potreban
  - Kompleksne AJAX pozive, kod kojih će ***preflight*** biti izvršen

- Kada se zahtev smatra kompleksnim:
  - Koristi se HTTP metoda koja nije GET, POST, HEAD
  - Koristi se *header* koji nije *Accept, Accept-Language, Content-Language*
  - *Content-Type header* ima vrednost koja nije iz skupa sledećih: *application/x-www-form-urlencoded, multipart/form-data, or text/plain*

- Zašto pravimo distinkciju? *Backwards compatibility, CORS* je mlađi od HTML formi, te šablona rada sa njima

Pre-flight Requests

- Omogućavaju *browser*-u da proveri da li se neki zahtev uopšte sme izvršiti?
- Šalje se *pre-flight* zahtev ka istoj putanji, samo sa *OPTIONS* metodom

- Odgovor bi trebalo da odgovori na sledeća pitanja:
  - Koji *origin*-i su dozvoljeni?
  - Koje metode su dozvoljene?
  - Koji *header*-i su dozvoljeni?
  - Da li je dozvoljeno automatsko slanje kredencijala? (*cookies, client-side SSL certificates, basic authentication*)

Pre-flight Requests

- Stoga, postoje sledeći *header*-i:
  - Access-Control-Allow-Origin – npr. https://origin2.com (može samo **jedna** vrednost)
  - Access-Control-Allow-Methods – npr. POST (može **više** vrednosti, razdvojeni zarezom)
  - Access-Control-Allow-Headers – npr. Content-Type (može više, razdvojeni zarezom)
  - Access-Control-Allow-Credentials – vrednost može biti samo ***true*** (opasno)

- Za prva tri *header*-a je moguće iskoristiti *wildcard* karakter *

- AJAX i *Credentials* - podešava se kroz fetch config:

```
fetch('./', { credentials: 'include' }) // 'include', 'omit', 'same-origin'
```

Access-Control-Allow-Origin

- Kako omogućiti više vrednosti? **Dinamički!** Pročitate Origin header request-a, i odlučite je li ga podržavate, ili ne!

- Kada je u redu postaviti vrednost na *:
  - Autentikacija i autorizacija nisu potrebni
  - Resurs bi trebalo da bude dostupan širokom spektru korisnika bez restrikcija
  - Mnoge web stranice i klijenti će pristupati resursu, te ne znamo u napred i nije nas briga (ne može doći do štete)

- Ako je *Credentials header true* – *Origin ne može biti **

NULL *Origin*

- *Origin* je *null* ako se pristupi resursu dok *browser* renderuje lokalni fajl
- Drugim rečima, zahtevi koje upit JavaScript kod iz statičkog fajla sa loklane mašine će imati *origin header null*

- Kada u takvim slučajevima server ne dozvoljava pristup resursima *null origin-a,* otežan je sam razvoj i smanjena prodktivnost
- Dozvola se mora svesno dati, otvara ranjivosti

- Primer 4

Diskusija za kraj

- Od čega CORS mehanizam štiti?
- Ko sprovodi CORS mehanizam?
- Da li CORS mehanizam štiti server?

- Primer 5 – slanje istog zahteva kroz *Postman*

- Tu priča samo počinje :)
- U zavisnosti od browser-a i trenutne konfiguracije, može se dogoditi da se pošalje *preflight* zahtev i za poziv koji se smatra nekompleksnim

Linkovi

- https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
- https://datatracker.ietf.org/doc/html/rfc6454
- https://ieftimov.com/posts/deep-dive-cors-history-how-it-works-best-practices/