29주차 발표_민서연

CORS

01 CORS

CORS

Cross-origin-resource-sharing

Access to XMLHttpRequest at 'https://localhost:44300/api/route/' from origin 'http://localhost:3000' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource.

CORS = 교차 출처 리소스 공유 정책

서버와 클라이언트가 정해진 헤더를 통해 서로 요청이나 응답에 반응 할 지 결정하는 방식

01 CORS

왜 발생할까요

최신 브라우저에서 구현된 동일 출처 정책 때문에 발생한 것!

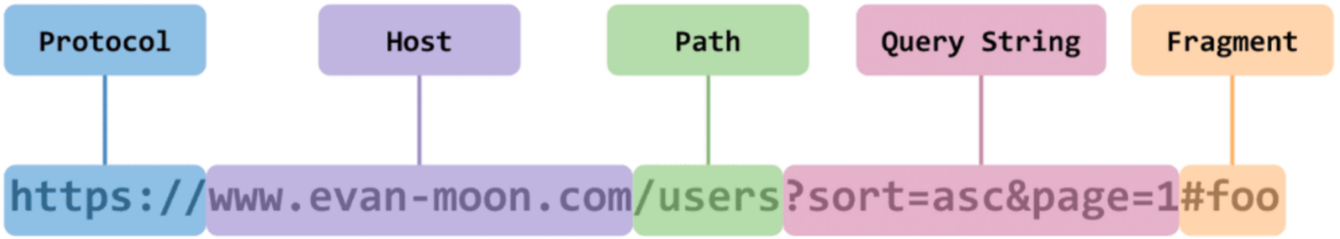
동일 출처 정책이란?!

동일한 출처의 리소스에만 접근하도록 제한하는 정책

브라우저에서 자동으로 쿠키가 첨부된다는 특징에서 해당 부분이 보안상의 문제를 발생 시킬 수 있겠다는 이슈로 생겨난 정책

01 CORS

URL 구조



출처가 다르다면 CORS가 발생한다.

출처: Protocal, Host, 포트번호

예) Http://localhost:8080
(프로토콜) (Host) (포트번호)

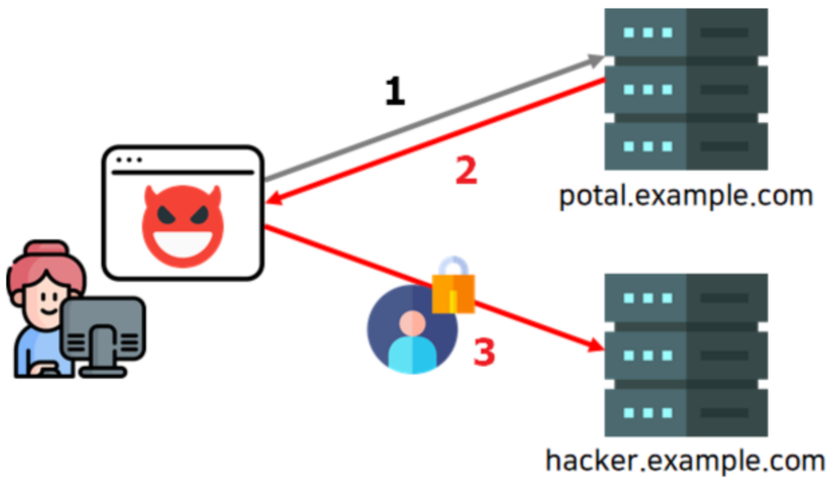
같은 출처 일 경우: Protocal, Host, 포트 번호가 모두 동일

다른 출처 일 경우: Protocal, Host, 포트 번호가 하나라도 다름

01 CORS

동일 정책의 필요성

동일 출처가 아닌 경우의 접근을 차단하지 않는다면
CSRF, XSS와 같은 방법으로 해커가 공격 해서 개인정보를 가로챌 수 있다



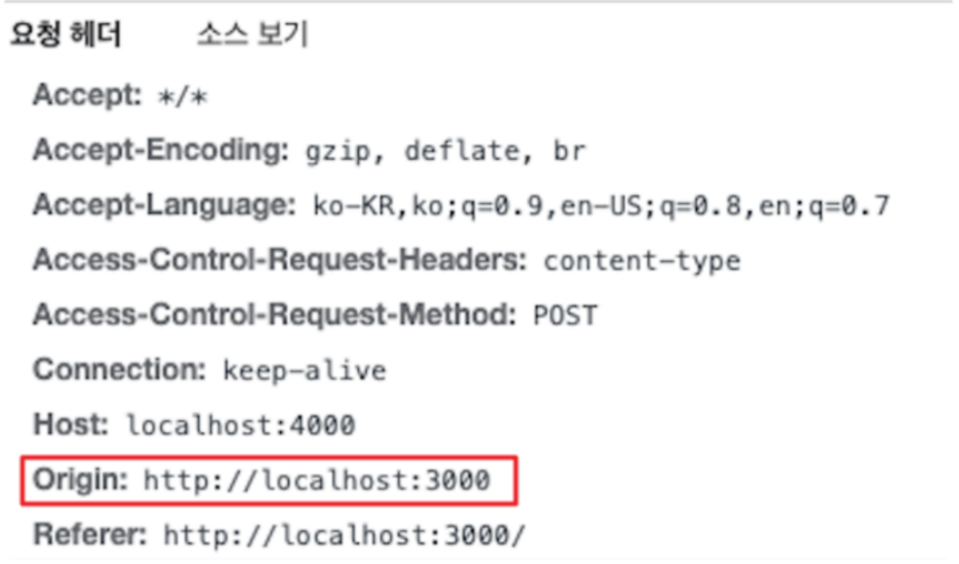
해킹을 이렇게 당한답니다

1. 사용자가 악성 사이트에 접속
2. 해커가 심어둔 악성 파일이 실행되어 어느 포털 사이트에 요청 보냄
3. 해당 브라우저의 쿠키를 이용해서 개인정보의 응답을 받은뒤 해커의 서버에게 전달

01 CORS

CORS 기본 동작

- 클라이언트에서 HTTP 요청의 헤더에 Origin을 담아서 전달하기

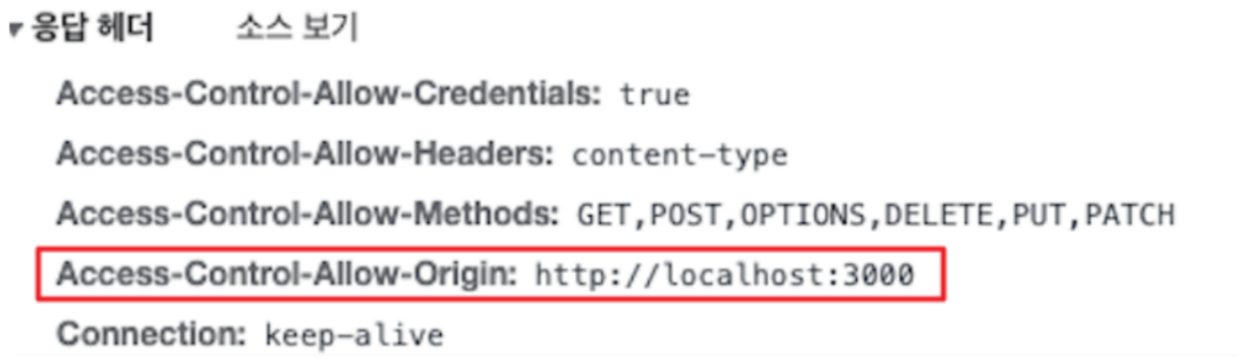


웹은 HTTP 프로토콜을 이용해서 서버에 요청을 보내는데
브라우저는 요청 헤더에 origin이라는 필드에 출처를 담아 보냄

01 CORS

CORS 기본 동작

- 서버는 응답 헤더에 Access-Control-Allow-Origin을 담아 클라이언트로 전달



서버가 요청에 대한 응답을 할 때 Access-Control-Allow-Origin이라는 필드를 추가해서
값으로 해당 리소스에 접근이 허용된 출처 url을 담아서 보낸다

01 CORS

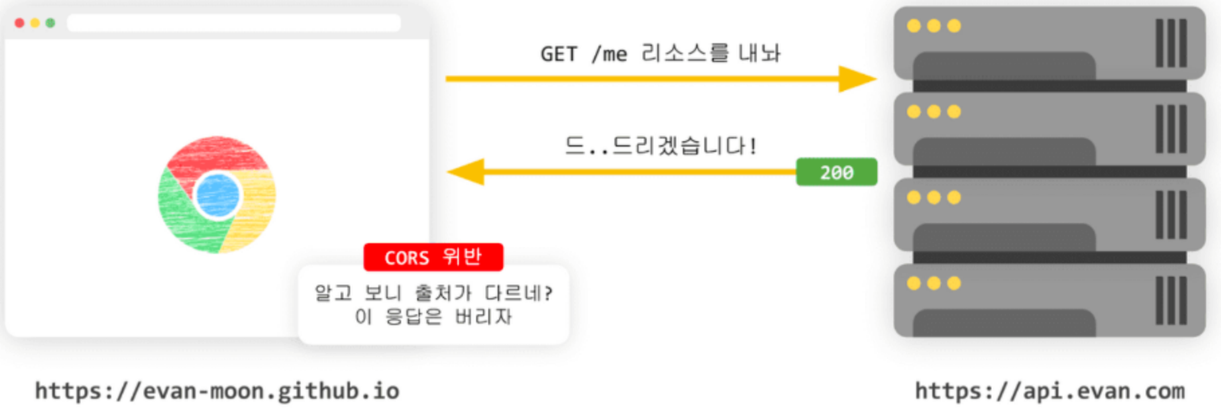
CORS 기본 동작

- 클라이언트에서 origin과 서버가 보내준 Access-Control-Allow-Origin을 비교

응답을 받은 브라우저는 자신이 보냈던 요청의 origin과 서버가 보내준 Access-Control-Allow-Origin을 비교해보고 차단 여부 판단

유효하지 않는다면 CORS에러

유효하다면 문제없이 리소스 가져옴



01 CORS

CORS 해결 방법

- Chrome 확장 프로그램

Chrome에서는 CORS문제를 해결하기 위한 확장 프로그램을 제공함

해당 확장 프로그램을 활성화 시키면 로컬 환경에서 api 테스트시 CORS 문제를 해결할 수 있음

01 CORS

CORS 해결 방법

- 프록시 사이트 이용

프록시는 클라이언트와 서버 사이의 중계 대리점

모든 출처를 허용한 서버 대리점인 프록시를 통해 리소스 요청이 가능하다.

무료 프록시 서버 대여 서비스들은 모두 악용 사례 때문에 api 요청 횟수 제한이 있어
실전에서는 사용하기 힘들다.

heroku 프록시 서버, cors proxy app 프록시 서버, cors.sh 프록시 서버

01 CORS

CORS 해결 방법

- 서버에서 Access-Control-Allow-Origin 헤더 세팅 노드 서버, 스프링 서버. 아파치 서버 등등

직접 서버에서 HTTP 헤더 설정을 통해 출처를 허용하게 설정하는 해결책

Access-Control-Allow-Origin: http://naver.com 헤더에 작성된 출처만 리소스 접근 허용

Access-Control-Request-Methods: GET, POST, PUT, DELETE 리소스 접근 허용 메서드 지정

Access-Control-Allow-Headers: Origin, Accept, X-Requested-With, Content-Type, Access-Control-Request-Method,.... 요청을 허용하는 헤더

Access-Control-Allow-Credentials: true 클라이언트의 요청이 쿠키를 통해서 자격 증명을 해야하는 경우 -> true

02 출처

https://inpa.tistory.com/entry/WEB-%F0%9F%93%9A-CORS-%F0%9F%92%AF-%EC%A0%95%EB%A6%AC-%ED%95%B4%EA%B2%B0-%EB%B0%A9%EB%B2%95-%F0%9F%91%8F
https://velog.io/@jh100m1/CORS-%EC%97%90%EB%9F%AC%EA%B0%80-%EB%AD%94%EB%8D%B0-%EC%96%B4%EB%96%BB%EA%B2%8C-%ED%95%B4%EA%B2%B0%ED%95%98%EB%8A%94%EA%B1%B4%EB%8D%B0