두둥!

지난 발표 QnA

Q. “origin과 site 등이 왜 나누어져 있나요?”

웹 생태계에서 다른 출처로의 리소스 요청을 제한하는 것과 관련된 두 가지 정책
- SOP(Same-Origin Policy) : 같은 출처 리소스 공유 정책, 2011년 RFC 6454
- CORS(Cross-Origin Resource Sharing) Policy : 교차 출처 리소스 공유 정책, 2009년

이러한 정책들이 생긴 이유
클라이언트 어플리케이션, 특히나 웹에서 돌아가는 클라이언트 어플리케이션은 사용자의 공격에 취약하다. 개발자 도구만 열어도 DOM이 어떻게 작성되어 있는지, 어떤 서버와 통신하는지, 리소스의 출처는 어디인지와 같은 **각종 정보들이 제재없이 열람가능**하다.
이러한 상황 속에 다른 출처의 어플리케이션이 서로 통신하는 것에 대해 아무런 제약이 존재하지 않는다면, 악의를 가진 사용자가 소스 코드를 확인한 후 **CSRF, XSS**와 같은 방법을 사용하여 정보를 탈취할 수 있다.

지난 발표 QnA

Q. “.com과 .net의 차이는 무엇인가요?”

.com, co.kr : company, 회사의 도메인에 사용 (co.kr - company korea)

.net, ne.kr : network, 네트워크를 관리하는 기관의 도메인에 사용

.org, or.kr : organization, 비영리기구 도메인에 사용

.gov : government, 정부관련 기관 도메인에 사용

.rec : research, 연구소 도메인에 사용

CS Study 12주차

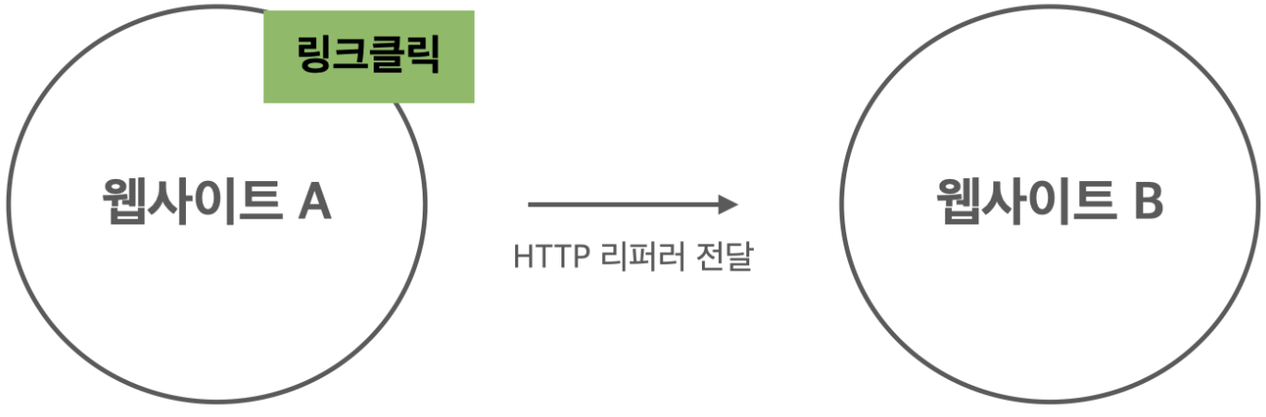
Referer & Referrer-Policy

김신아

Referer와 Referrer-Policy

HTTP 리퍼러를 정의한 RFC에서 ‘referrer’를 ‘referer’라고 잘못 친 것에서 기인하여
HTTP 리퍼러는 ‘HTTP referer’라고 불린다.
Referrer-Policy에서 Referer는 **HTTP referer**를 의미한다.

Referer는 웹 브라우저로 월드 와이드 웹을 서핑할 때,
하이퍼링크를 통해서 각각의 사이트로 **방문 시 남는 흔적**을 말한다.



cf. HTTP 리퍼러를 정의한 RFC -> https://datatracker.ietf.org/doc/html/rfc7231

Referer와 Referrer-Policy

Referrer-Policy
브라우저는 전달되는 주소의 노출 정도를 정책으로 정할 수 있다.

e.g. I
<**meta** name = “referrer” content = “no-referrer” />
-> referer를 남기지 않는 정책

<**meta** name = “referrer” content = “unsafe-url” />
-> 조건 없이 주소를 남기는 정책, 주소 전체를 남김

<**meta** name = “referrer” content = “origin” />
-> 도메인 정보만 남기는 정책, 예를 들어 https://example.com/test/foo 라는 주소가 있으면 https://example.com 까지만 전송

<**meta** name = “referrer” content = “strict-origin” />
-> 이동하는 웹사이트의 주소가 https일때만 도메인 주소를 남기는 정책

Referer와 Referrer-Policy

e.g. II

<**meta** name = “referrer” content = “no-referrer-when-downgrade” />

-> 이동하는 웹사이트 주소가 https일 때 주소를 남기는 정책

<**meta** name = “referrer” content = “same-origin” />

-> 웹사이트가 같을 때 주소를 남기는 정책

<**meta** name = “referrer” content = “origin-when-cross-origin” />

-> 웹사이트가 같을 때는 전체주소, 다를 때는 도메인 주소만 남기는 정책

<**meta** name = “referrer” content = “strict-origin-when-cross-origin” />

-> https를 사용하며 웹사이트가 같을 때는 전체주소, 다를 때는 도메인 주소만 남기는 정책

http를 사용하는 웹사이트에는 주소를 남기지 않음

Referer와 Referrer-Policy

	No Data	Origin Only	Full URL
no-referrer	✓		
origin		✓	
unsafe-url			✓
strict-origin	HTTPS -> HTTP	HTTPS -> HTTPS, HTTP -> HTTP	
no-referrer-when-downgrade	HTTPS -> HTTP		HTTPS -> HTTPS, HTTP -> HTTP
origin-when-cross-origin		cross-origin	same-origin
same-origin	cross-origin		same-origin
strict-origin-when-cross-origin	HTTPS -> HTTP	cross-origin, HTTPS -> HTTPS, HTTP -> HTTP	

브라우저별 표준

브라우저	기본 정책
Chrome	85 버전부터 no-referrer-when-downgrade에서 strict-origin-when-cross-origin 으로 변경
Firefox	no-referrer-when-downgrade, 시크릿 모드에서는 strict-origin-when-cross-origin
Edge	no-referrer-when-downgrade
Safari	strict-origin-when-cross-origin와 비슷하게 동작

Referrer-Policy 설정하는 올바른 방법

사이트에 referrer policy를 설정하는 방법은 여러가지가 있다.

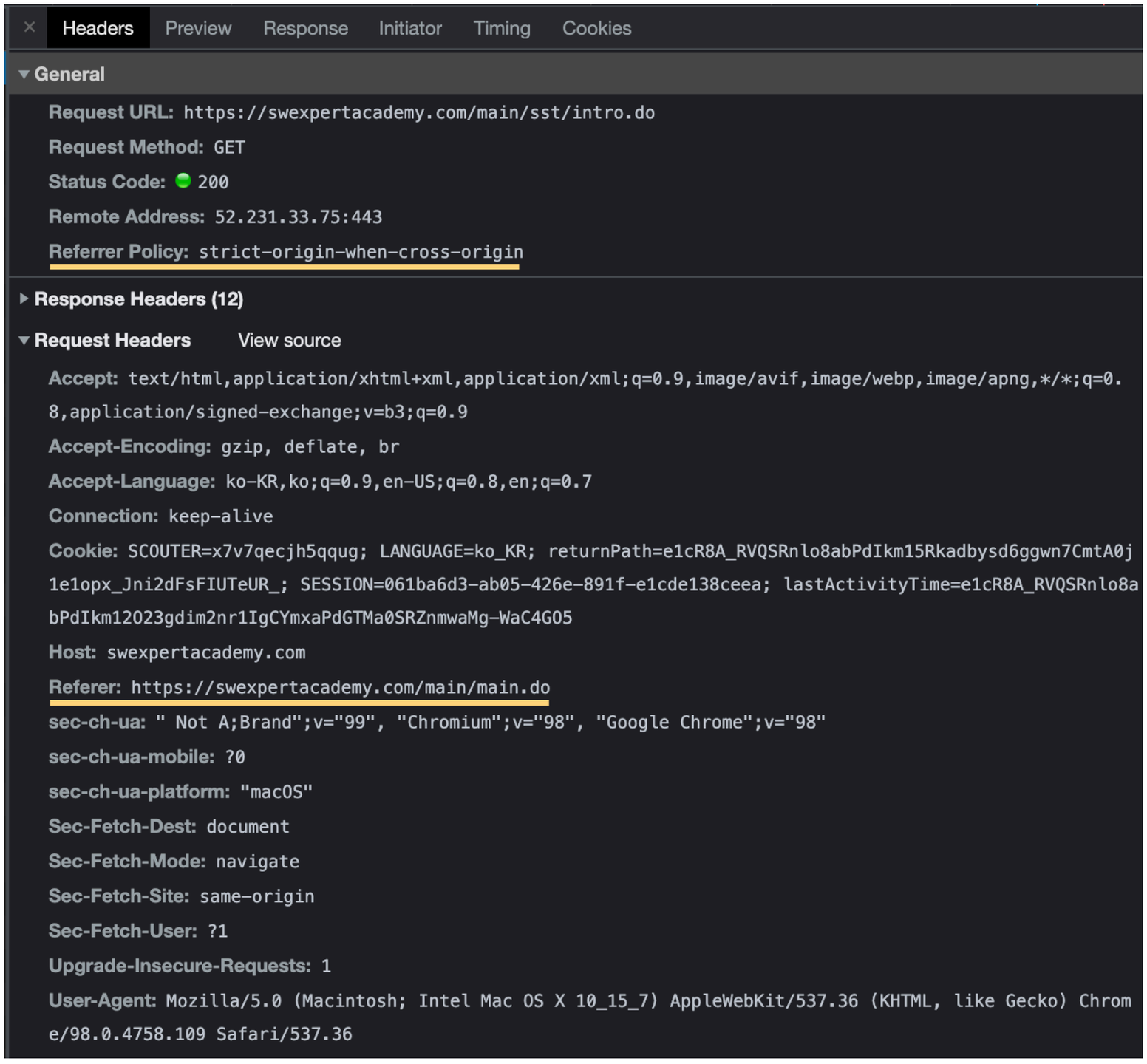
- HTTP Header
- HTML
- Javascript

페이지마다, 요청마다 다른 정책을 쓸수 있다는 것을 의미한다. HTTP Header와 meta 엘리먼트는 모두 페이지 레벨에서 동작한다. 유효 정책을 정하는 순위는 아래와 같다.

- element 레벨
- page 레벨
- 브라우저 기본값

```
<meta name="referrer" content="strict-origin-when-cross-origin" />
<img src="..." referrerpolicy="no-referrer-when-downgrade" />
```

Referrer-Policy를 보는 방법



어떤 정책이 좋을까?

명시적으로 보안이 강화된 strict-origin-when-cross-origin을 사용하는 것이 좋다.

<**meta** name = “referrer” content = “strict-origin-when-cross-origin” />
-> https를 사용하며 웹사이트가 같을 때는 전체주소, 다를 때는 도메인 주소만 남기는 정책
http를 사용하는 웹사이트에는 주소를 남기지 않음

왜 명시적으로 써야 할까?

referrer policy가 제공되지 않는다면, 브라우저 기본 정책이 사용된다.

- 브라우저 모드(비공개/시크릿 모드)에 따라서 no-referrer-when-downgrade 이거나, strict-origin-when-cross-origin, 또는 이보다 엄격하다. 이는 웹사이트가 모든 브라우저에서 예상대로 작동하는 것은 아니다.
- 브라우저는 출처 간 요청에 대해 strict-origin-when-cross-origin과 같은 더 엄격한 기본값 및 리퍼러 트리밍과 같은 매커니즘을 채택하고 있다. 브라우저 기본값이 변경되기 전에 개인정보 보호 강화 정책을 명시적으로 선택하면 제어력을 확보할 수 있고 테스트를 실행할 수 있다.

왜 strict-origin-when-cross-origin을 추천하는가?

- **안전성**
  : 웹사이트가 https일 경우, https가 아닌 요청에 대해서 웹사이트 주소를 노출하고 싶지 않을 것이다. policy를 설정해놓지 않으면 유저의 정보가 중간자 공격의 위험에 노출될 수 있다.
  no-referrer-when-downgrade, strict-origin-when-cross-origin,
  no-referrer, strict-origin으로 막을 수 있다.

- **개인정보 보안**
  : cross-origin 요청의 경우, no-refferer-when-downgrade는 모든 주소를 노출시킨다.
  strict-origin-when-cross-origin와 strict-origin은 origin만 공유하고, no-referrer의 경우에는 아무정보도 안나타나게 된다.

- **용이성**
  : no-referrer와 strict-origin은 전체 URL을 공유하지 않는다.
  이러한 문제는 same-origin일때도 공유를 안하다는 것이다.
  이를 피하기 위해서는 strict-origin-when-cross-origin을 쓰면 된다.

왜 strict-origin-when-cross-origin을 추천하는가?

- 클라이언트 사이드에서

```
<meta name="referrer" content="strict-origin-when-cross-origin" />
```

- 혹은 서버 사이드에서

```
const helmet = require('helmet')
app.use(helmet.referrerPolicy({ policy : 'trict-origin-when-cross-origin' }))
```

만약 예외가 필요하다면

별도로 element나 요청별로 예외를 두는 것이 좋다.
unsafe-url과 같은건 사용하지 않는 게 좋다.

```
<meta name="referrer" content="strict-origin-when-cross-origin" />
<img src="..." referrerpolicy="no-referrer-when-downgrade" />
```

```
fetch(url, { referrerPolicy : 'no-referrer-when-downgrade' })
```

cf. element 별로 정책을 주는 것도 모든 브라우저에서 되는 것은 아니다.

Thank You

별첨

별첨

	No Data	Origin Only	Full URL
no-referrer	✓		
origin		✓	
unsafe-url			✓
strict-origin	HTTPS -> HTTP	HTTPS -> HTTPS, HTTP -> HTTP	
no-referrer-when-downgrade	HTTPS -> HTTP		HTTPS -> HTTPS, HTTP -> HTTP
origin-when-cross-origin		cross-origin	same-origin
same-origin	cross-origin		same-origin
strict-origin-when-cross-origin	HTTPS -> HTTP	cross-origin, HTTPS -> HTTPS, HTTP -> HTTP	

별첨

브라우저	기본 정책
Chrome	85 버전부터 no-referrer-when-downgrade에서 strict-origin-when-cross-origin 으로 변경
Firefox	no-referrer-when-downgrade, 시크릿 모드에서는 strict-origin-when-cross-origin
Edge	no-referrer-when-downgrade
Safari	strict-origin-when-cross-origin와 비슷하게 동작

별첨

https://yceffort.kr/2020/09/referer-and-referrer-policy

https://datatracker.ietf.org/doc/html/rfc7231

https://scshim.tistory.com/entry/Referer-Policy%EC%9D%B4%EB%9E%80

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy

https://mystarlight.tistory.com/113

https://evan-moon.github.io/2020/05/21/about-cors/