APPENDIX B

한국어판 부록

B.1 보안 체크리스트

책에서 설명하는 취약성 관련 대책 체크리스트이며 웹 애플리케이션을 만들 때 참고하자. 체크 내용만 표로 나타냈고 각 체크 항목에 관한 설명은 책에서 상세히 설명한다.

목적	설명 위치	체크 내용	
통신 암호화 (HTTPS)	3.3.1	리소스를 HTTPS로 전송	
	3.3.1	신뢰할 수 있는 CA가 발행한 증명서 사용	
	3.3.5	Mixed Content 발생하지 않는지 확인	
	3.3.6	HTTP로 접속 시 HTTPS로 리다이렉트 설정	
	3.3.6	HSTS 유효화 시 Strict-Transport-Security 헤더를 페이지 응답에 추가	
	3.3.6	서브 도메인 HSTS 유효화 시 Strict-Transport-Security 헤더에 includeSubdomain 추가	
	3.3.6	HSTS Preload 사용 시 Strict-Transport-Security 헤더에 preload 추가	
	3.3.6	HSTS Preload 사용 시 Strict-Transport-Security 헤더에 includeSubdomain 추가	
	3.3.6	HSTS Preload 사용 시 Preload List에 등록	
	3.3.6	HSTS 유효 기간 설정 시 Strict-Transport-Security 헤더에 적절한 max-age 값 추가	

목적	설명 위치	체크 내용	
교차 출처 통신	4.4.2	Access-Control-Allow-Origin에 허가된 출처만 설정됐는지 확인	
	4.4.2	모든 출처의 접속이 문제없을 때만 Access-Control-Allow-Origin을 *(와일드카드)로 설정	
	4.4.3	Access-Control-Allow-Method는 불필요한 HTTP 메서드를 포함하지 않음	
	4.4.3	Access-Control-Allow-Headers는 불필요한 HTTP 헤더를 포함하지 않음	
	4.4.3	Access-Control-Max-Age는 리소스 특성을 고려해 적절히 설정	
	4.4.4	fetch 함수를 사용해 교차 출처로 cookie를 포함하는 요청을 전송할 때 적절한 credentials 옵션 설정	
	4.4.4	XMLHttpRequest를 사용해 교차 출처로 cookie를 포함하는 요청을 전송할 때 withCredentials=true 설정	
	4.4.4	교차 출처에서 Cookie를 포함하는 요청을 허가할 때 응답에 Access-Control-Allow-Credentials:true 설정	
	4.4.4	Access-Control-Allow-Credentials 헤더를 추가할 때 Access-Control-Allow-Origin 값을 *(와일드카드)로 지정하지 않음	
	4.4.5	fetch 함수를 사용한 요청에 대해 동일 출처 정책으로 제한하고 싶을 때 mode:same-origin 또는 mode:no-cors를 fetch 함수의 옵션에 지정	
	4.4.6	<canvas> 요소에 교차 출처에서 가져온 이미지를 불러올 때 <img> 요소에 crossorigin 속성을 설정	
	4.4.6	cookie 전송 제어를 위해 HTML 요소의 crossorigin 속성에 적정값을 설정	
	4.6	iframe을 통해 데이터를 주고 받을 때 데이터 수신 측의 iframe에서 데이터 송신자가 접속을 허가하는 출처인지 확인	
	4.7.2	SharedArrayBuffer의 사용과 Performance 종류의 API 정확도를 떨어뜨리고 싶지 않을 때 Cross Origin Isolation(COOP + COEP) 유효화	
	4.7.3	self.crossOriginIsolated의 값이 true일 때만 SharedArrayBuffer 사용	
XSS 대책	5.2.5	사용자가 입력한 문자열을 HTML에 삽입할 때 문자열을 이스케이프 처리	
	5.2.5	사용자가 입력한 문자열을 HTML 요소의 속성에 삽입할 때 문자열을 따옴표로 감싸기	
	5.2.5	사용자가 입력한 문자열을 <a> 요소의 href 속성에 삽입할 때 문자열이 http: 또는 https:로 시작하는 URL 문자열인지 확인	
	5.2.5	사용자가 입력한 문자열을 HTML에 삽입할 때 innerHTML 등의 싱크 대신 appendChild 함수 등을 사용	
	5.2.5	민감한 정보를 포함하는 cookie에는 HttpOnly 속성 추가	
	5.2.5	사용자가 입력한 문자열을 HTML에 삽입할 때 DOMPurify 등 XSS 대책을 위한 라이브러리와 Sanitizer API를 사용해 XSS 원인이 되는 문자열 제거	
	5.4.1	Content-Security-Policy(CSP) 적용	
	5.4.1	CSP 값에 unsafe-inline를 지정하지 않음	

목적	설명 위치	체크 내용	
XSS 대책	5.4.1	CSP 값에 unsafe-eval를 지정하지 않음	
	5.4.1	CSP 값에 unsafe-hashes를 지정하지 않고 HTML 요소에 연결하는 이벤트 핸들러는 자바스크립트의 addEventListener로 설정	
	5.4.1	실행과 불러오기 제어가 필요한 유형의 리소스는 필요한 CSP의 directive를 설정	
	5.4.1	CSP의 각 directive 값에 불필요한 호스트명을 지정하지 않음	
	5.4.2	CSP의 값에 인라인 스크립트를 허가할 때 unsafe-inline 대신 nonce-source와 hash-source 사용	
	5.4.2	CSP의 각 directive 값에 호스트명을 지정하지 않고 nonce-source와 hash-source를 사용	
	5.4.2	nonce-source를 CSP에 사용할 때 nonce의 값은 추측하기 어려운 값을 요청마다 변경하도록 설정	
	5.4.2	HTML을 요청마다 동적으로 생성할 수 없을 때는 hash-source를 사용하는 CSP 설정	
	5.4.2	동적으로 스크립트를 불러오고 싶을 때는 strict-dynamic을 사용해 <script> 요소를 동적으로 생성	
	5.4.2	CSP의 값에 object-src 'none'을 지정	
	5.4.2	CSP의 값에 base-url 'none'을 지정	
	5.4.3	스크립트의 삽입과 외부 스크립트 불러오기를 허가할 때 Trusted Types 사용	
	5.4.3	불필요한 Trusted Types의 Policy 함수를 작성하지 않고 CSP 헤더에 불필요한 Policy를 지정하지 않음	
	5.4.4	CSP와 Trusted Types를 사용하기 전 Report-Only 모드를 사용해 충분히 검사	
	5.4.4	CSP와 Trusted Types 사용 후에도 리포트를 계속해서 수집	
CSRF 대책	6.1.2	원타임 토큰을 사용한 CSRF 대책은 폼에 포함된 CSRF 토큰이 서버의 토큰과 일치하는지 확인	
	6.1.2	폼에 CSRF 토큰을 포함할 때 해당 토큰을 type=hidden 등으로 숨김 처리	
	6.1.2	CSRF 토큰은 추측하기 어려운 문자열로 하고 세션마다 변경	
	6.1.3	Double Submit Cookie를 통한 CSRF 대책은 CSRF 토큰을 값으로 갖는 cookie를 발행하고, 이 토큰을 요청 헤더(또는 바디)에 포함하여 cookie와 요청 양쪽의 토큰 일치 여부를 서버에서 확인	
	6.1.4	cookie의 SameSite 속성값을 Lax 또는 Strict로 지정(Lax가 기본값인 브라우저도 있음)	
	6.1.5	서버에서 출처를 확인할 수 있을 때 CSRF 대책을 위해 요청을 허가하는 출처 이외의 요청은 거부	

목적	설명 위치	체크 내용	
클릭재킹 대책	6.3.2	iframe 등을 사용해 다른 출처에서 포함되지 않을 페이지의 응답 헤더는 X-Frame-Options:SAMEORIGIN 또는 X-Frame-Options: DENY를 설정(CSP frame-ancestors를 사용하는 방법도 가능)	
	6.3.2	iframe 등을 사용해 다른 출처에서 포함되지 않을 페이지의 응답 헤더는 frame-ancestors 'none' 또는 frame-ancestors 'self'를 포함하는 CSP 설정(X-Frame-Options를 사용하는 방법도 가능)	
오픈 리다이렉트 대책	6.5.2	사용자가 입력한 URL 문자열을 사용해 리다이렉트할 때 리다이렉트 URL은 특정 URL 또는 출처로 제한	
인증 공격 대책	7.2.3	깃허브 등 개발에 사용하는 서비스는 이중 인증을 유효화	
	7.2.3	이중 인증을 구현	
	7.2.3	사용자가 비밀번호를 일정 횟수만큼 틀리면 계정 잠금	
	7.2.3	비밀번호 입력을 검증(validation)	
	7.3.4	비밀번호 입력 보조 기능 구현	
	7.3.4	비밀번호 입력 화면에서 웹 분석 서비스 등을 통해 데이터가 전송되지 않는지 확인	
	7.3.4	민감한 정보를 웹 스토리지에 저장해도 문제가 없을지 확인	
안전한 라이브러리 사용	8.3.1	사용 중인 라이브러리의 취약성 검사	
	8.3.2	유지보수가 종료된 라이브러리 사용 피하기	
	8.3.3	최신 라이브러리나 취약성 문제가 해결된 버전의 라이브러리 사용	
	8.3.4	CDN 등 외부에서 라이브러리를 불러올 때 SRI를 통한 변조 확인	
	8.3.5	CDN 등 외부에서 라이브러리를 불러올 때 취약성이 없는 버전 사용	