1. 기사 내용

제목	크롬 68, HTTPS가 아니면 '안전하지 않음' 딱지 부착
기사 날짜	2018년 7월 26일
스크랩 담당자	손우규
요약	크롬 보안 팀(Chrome Security Team)은 3년 반 전에 모든 HTTP 사이트에 안전하지 않다는 표시를 붙이자는 제안을 했다. 웹을 브라우징하는 사용자들이 최소한의 데이터 보호 장치가 마련된 곳을 안전하게 접속할 수 있도록 하기 위함이었다. HTTP를 기반으로 웹사이트가 로딩될 때, 암호화 기술은 동원되지 않는다. 네트워크 내에 있는 공격자가 전송되고 있는 정보에 접근할 수 있을뿐만 아니라, 내용물이 사용자에게 도달하기 전에 조작할 수 있다는 뜻이 된다. 반면 HTTPS는 연결에 암호화 기술을 적용하기 때문에 중간에 누군가 개입해서 전송되고 있는 정보를 가로채거나 조작할 수 없다. 사용자의 정보가 사용자의 의도와 달리 노출되는 일을 HTTPS는 막는다는 것이다. 구글의 투명성 보고서에 의하면 HTTPS 사용량은 전 세계적으로 크게 늘어났다고 한다. 전체 웹 페이지의 약 75%가 HTTPS 연결망을 통해 사용자들에게 전달된다. 크롬 OS, 맥OS, 안드로이드, 윈도우 등 모든 플랫폼에서 말이다. 리눅스의 경우 HTTPS가 적용된 건 약 66%였다.
출처	https://www.boannews.com/media/view.asp?idx=71759&kind=1&search=title&find=https

2. 용어

용어	설명
HTTP(HyperText Transfer Protocol)	HTTP는 클라이언트와 서버 사이에 이루어지는 요청/응답(request/response) 프로토콜이다. 예를 들면, 클라이언트인 웹 브라우저가 HTTP를 통하여 서버로부터 웹페이지나 그림 정보를 요청하면, 서버는 이 요청에 응답하여 필요한 정보를 해당 사용자에게 전달하게 된다. 이 정보가 모니터와 같은 출력 장치를 통해 사용자에게 나타나는 것이다.
HTTPS(HyperText Transfer Protocol over Secure Socket Layer)	HTTPS는 소켓 통신에서 일반 텍스트를 이용하는 대신에, SSL이나 TLS 프로토콜을 통해 세션 데이터를 암호화한다. 따라서 데이터의 적절한 보호를 보장한다. HTTPS의 기본 TCP/IP 포트는 443이다. 보호의 수준은 웹 브라우저에서의 구현 정확도와 서버 소프트웨어, 지원하는 암호화 알고리즘에 달려있다.

3. 관련 기술(Helmet)

3.1 개요

Helmet은 다양한 HTTP 헤더를 설정하여 Express 응용 프로그램의 보안을 유지하도록 도와준다.

3.2 방어 형태

3.2.1 Helmet

Helmet을 이용하면 HTTP 헤더를 적절히 설정하여 몇 가지 잘 알려진 웹 취약성으로부터 앱을 보호할 수 있다.

사실 Helmet은 보안 관련 HTTP 헤더를 설정하는 다음과 같은 더 작은 크기의 미들웨어 함수 9개의 모음이다.

- csp는 Content-Security-Policy 헤더를 설정하여 XSS(Cross-site scripting) 공격 및 기타 교차 사이트 인젝션을 예방한다.
- hidePoweredBy는 X-Powered-By 헤더를 제거한다.
- hpkp는 Public Key Pinning 헤더를 추가하여, 위조된 인증서를 이용한 중간자 공격을 방지한다.
- hsts는 서버에 대한 안전한(SSL/TLS를 통한 HTTP) 연결을 적용하는 Strict-Transport-Security 헤더를 설정한다.
- ieNoOpen은 IE8 이상에 대해 X-Download-Options를 설정한다.
- noCache는 Cache-Control 및 Pragma 헤더를 설정하여 클라이언트 측에서 캐싱을 사용하지 않도록 한다.
- noSniff는 X-Content-Type-Options 를 설정하여, 선언된 콘텐츠 유형으로부터 벗어난 응답에 대한 브라우저의 MIME 가로채기를 방지한다.
- frameguard는 X-Frame-Options 헤더를 설정하여 clickjacking에 대한 보호를 제공한다.

- xssFilter는 X-XSS-Protection을 설정하여 대부분의 최신 웹 브라우저에서 XSS(Cross-site scripting) 필터를 사용하도록 한다.

다른 모든 npm 모듈처럼 Helmet은 다음과 같이 설치할 수 있습니다.
$ npm install --save helmet

이후 코드에서 Helmet을 사용하는 방법은 다음과 같다.

```
...
var helmet = require('helmet');
app.use(helmet());
...
```

적어도 X-Powered-By 헤더는 사용하지 않도록 설정

Helmet의 사용을 원치 않는 경우에는 적어도 X-Powered-By 헤더를 사용하면 안된다. 공격자는 이 헤더(기본적으로 사용하도록 설정되어 있음)를 이용해 Express를 실행하는 앱을 발견한 후 특정한 대상에 대한 공격을 실행할 수 있다.

따라서 우수 사례는 다음과 같이 app.disable() 메소드를 이용해 이 헤더를 끄는 것이다.

```
...
app.disable('x-powered-by');
...
```

3.3 문제점
모듈을 사용한다고 하더라도, 다른 모든 웹 앱과 마찬가지로 Express 앱은 다양한 웹 기반 공격에 취약할 수 있다. 알려져 있는 웹 취약성을 숙지한 후 이러한 취약성을 피하기 위한 예방 조치가 최선이다.

3.4 해결방안
HTTP 헤더를 설정하여 최소한의 취약성으로부터 앱을 보호한다. 때문에 HTTPS와 같은 안전한 프로토콜을 쓰는 것이 하나의 방법이다.

3.4.1 HTTPS
HTTPS에서 마지막의 S는 Over Secure Socket Layer의 약자로 Secure라는 말을 통해서 알 수 있듯이 보안이 강화된 HTTP라는 것을 짐작할 수 있다. HTTP는 암호화되지 않은 방법으로 데이터를 전송하기 때문에 서버와 클라이언트가 주고 받는 메시지를 감청하는 것이 매우 쉽다. 예를들어 로그인을 위해서 서버로 비밀번호를 전송하거나, 또는 중요한 기밀 문서를 열람하는 과정에서 악의적인 감청이나 데이터의 변조등이 일어날 수 있다는 것이다. 이를 보안한 것이 HTTPS다. HTTPS는 소켓 통신에서 일반 텍스트를 이용하는 대신에, SSL이나 TLS 프로토콜을 통해 세션 데이터를 암호화한다. 따라서 데이터의 적절한 보호를 보장한다. HTTPS의 기본 TCP/IP 포트는 443이다.

3.4.2 추가적인 고려사항
- 속도 제한(rate-limiting)을 구현하여 인증에 대한 무차별 대입 공격을 방지해야 한다. 이를 실행하는 한 가지 방법은 StrongLoop API Gateway를 이용하여 속도 제한 정책을 적용하는 것이다. 대안

적으로, express-limiter와 같은 미들웨어를 사용할 수 있지만, 그러한 경우에는 코드를 어느 정도 수정하여야 한다.

- csurf 미들웨어를 이용하여 교차 사이트 요청 위조(CSRF)로부터 보호하여야 한다.
- 항상 사용자 입력을 필터링하고 사용자 입력에서 민감한 데이터를 제거하여 XSS(Cross-site scripting) 및 명령 인젝션 공격으로부터 보호하여야 한다.
- 매개변수화된 조회 또는 준비된 명령문을 이용하여 SQL 인젝션 공격으로부터 방어하여야 한다.
- 오픈 소스 방식의 sqlmap 도구를 이용하여 앱 내의 SQL 인젝션 취약성을 발견하여야 한다.
- nmap 및 sslyze 도구를 이용하여 SSL 암호, 키 및 재협상의 구성, 그리고 인증서의 유효성을 테스트하여야 한다.
- safe-regex를 이용하여 정규식이 정규식 서비스 거부 공격을 쉽게 받지 않도록 하여야 한다.

4. 참고 자료

- http://expressjs.com/ko/advanced/best-practice-security.html