HTTP vs HTTPS

HTTP (HyperText Transfer Protocol)

- **기본 프로토콜**: 웹 브라우저와 서버 간의 데이터를 전송하는데 사용되는 기본 프로토콜입니다.
- **보안 취약성**: HTTP는 데이터를 암호화하지 않고 전송하기 때문에 중간에 누군가가 데이터를 가로채서 내용을 볼 수 있습니다. 이는 보안 취약성을 초래합니다.

HTTPS (HyperText Transfer Protocol Secure)

- **보안 강화**: HTTPS는 HTTP의 보안 버전으로, 데이터를 SSL/TLS 프로토콜을 통해 암호화하여 전송합니다. 이를 통해 중간에서 데이터가 가로채지더라도 내용을 볼 수 없게 됩니다.

  ▼ SSL/TLS

  ### SSL/TLS 프로토콜

  SSL과 TLS는 모두 웹 통신의 보안을 강화하기 위해 사용되는 암호화 프로토콜입니다.

  SSL/TLS 프로토콜은 데이터를 안전하게 전송하기 위한 규칙과 절차를 정의합니다.

  ▼ SSL (Secure Sockets Layer) 프로토콜

  1. **보안 수준**:
     - SSL 2.0은 보안 결함이 있어 SSL 3.0으로 빠르게 대체되었습니다.
     - 그러나 SSL 3.0 역시 여러 가지 보안 취약점이 발견되어 더 이상 안전한 것으로 간주되지 않습니다.
  2. **현재 상태**:
     - SSL 프로토콜은 더 이상 사용되지 않으며, 대부분의 현대 웹사이트와 애플리케이션은 TLS로 전환되었습니다.

  ▼ **TLS (Transport Layer Security) 프로토콜**

  1. **보안 수준**:

- TLS는 SSL의 보안 취약점을 보완하며, 향상된 암호화 기술과 더 강력한 보안 기능을 제공합니다.
- 각 버전(TLS 1.0, 1.1, 1.2, 1.3)은 이전 버전보다 더 강력한 보안을 제공합니다.
- TLS는 SSL보다 더 강력하고 최신의 암호화 알고리즘을 지원합니다.

2. **현재 상태**:
    - TLS는 현재 가장 널리 사용되는 암호화 프로토콜입니다.
    - 대부분의 현대 웹 브라우저와 서버는 TLS 1.2와 TLS 1.3을 지원합니다.
    - TLS 1.3은 더 간단하고 효율적인 핸드셰이크 과정을 사용하여 보안을 강화하고 성능을 향상시켰습니다.

    ▼ 동작방식

    1. **핸드셰이크 과정**:
        - 클라이언트와 서버가 서로 인사를 하고 암호화 방법을 협상합니다.
        - 서버는 클라이언트에게 TLS 인증서를 보내어 자신의 신원을 증명합니다.
        - 클라이언트는 인증서의 유효성을 확인하고, 서버의 공개 키를 사용하여 세션 키를 암호화하여 서버에 보냅니다.
        - 서버는 자신의 비공개 키를 사용하여 세션 키를 복호화합니다.
    2. **데이터 전송**:
        - 세션 키를 사용하여 데이터가 암호화되고, 클라이언트와 서버 간에 안전하게 전송됩니다.

TLS 인증서

TLS 인증서는 웹 서버의 신원을 증명하고 데이터를 암호화하기 위해 필요한 디지털 문서입니다.

- `역할`

  TLS 인증서는 웹 서버와 클라이언트 간의 안전한 통신을 위해 사용됩니다. 인증서는 서버의 신원을 확인하고 데이터를 암호화하는 데 중요한 역할을 합니다.

  - **서버 신원 확인**: 클라이언트가 접속하는 서버가 신뢰할 수 있는 서버인지 확인합니다.
  - **데이터 암호화**: 서버와 클라이언트 간의 데이터를 암호화하여 전송합니다. 이를 통해 중간에 데이터가 도청되거나 변조되는 것을 방지합니다.

- `구성 요소`

  인증서는 서버의 공개 키와 디지털 서명 등 여러 정보를 포함합니다. 이를 통해 클라이언트는 서버가 신뢰할 수 있는지 확인할 수 있습니다. 클라이언트는 이 공개 키를 사용하여 데이터를 암호화하고 서버는 해당 데이터의 복호화를 위해 비공개 키를 사용합니다.

  - **공개 키**: 서버의 공개 키.
  - **인증 기관의 서명**: 신뢰할 수 있는 인증 기관(CA)이 인증서를 발급할 때 서명합니다.
  - **서버 정보**: 서버의 도메인 이름, 회사 정보 등.

- `인증서 적용 과정`

  1. **인증서 발급**:
     - 인증 기관(CA, Certificate Authority)이 웹 서버의 신원을 확인하고 인증서를 발급합니다.
     - 발급된 인증서는 서버에 설치됩니다.
  2. **프로토콜 사용**:
     - 웹 서버가 클라이언트와 통신할 때 SSL 또는 TLS 프로토콜을 사용하여 데이터를 암호화합니다.
     - 이 과정에서 서버는 클라이언트에게 SSL/TLS 인증서를 제시하여 자신의 신원을 증명합니다.
  3. **서버 인증서 설치**:
     - SSL/TLS 인증서를 서버에 설치합니다.
     - 서버 설정을 변경하여 HTTPS를 사용하도록 합니다.
  4. **클라이언트와의 핸드셰이크**:

- 클라이언트가 서버에 접속할 때, 서버는 인증서를 클라이언트에게 제시합니다.
- 클라이언트는 인증서의 유효성을 확인하고 서버의 신원을 검증합니다.
- 검증이 완료되면 SSL/TLS 프로토콜을 통해 데이터를 암호화하여 안전하게 통신합니다.

따라서, SSL/TLS 인증서는 프로토콜을 사용하여 안전한 통신을 설정하는 데 필수적인 요소입니다. 프로토콜은 데이터를 어떻게 암호화하고 전송할지를 정의하며, 인증서는 이러한 과정에서 서버의 신원을 보장하고 암호화된 통신을 가능하게 합니다.

- **인증서 사용**: HTTPS는 SSL/TLS 인증서를 사용하여 웹사이트의 신원을 확인합니다. 이를 통해 사용자는 자신이 접속한 웹사이트가 신뢰할 수 있는지 확인할 수 있습니다.
- **SEO 및 신뢰성**: HTTPS를 사용하면 검색 엔진 최적화(SEO)에 긍정적인 영향을 미치며, 사용자들에게 신뢰성을 제공합니다.

▼ HTTPS 사용이 검색 엔진 최적화에 긍정적인 영향인 이유

1. **검색 엔진의 우선순위**:
   - 구글과 같은 주요 검색 엔진은 HTTPS를 사용하는 웹사이트에 대해 더 높은 순위를 부여하는 경향이 있습니다. 2014년에 구글은 HTTPS를 사용하는 웹사이트를 우선적으로 고려할 것이라고 발표했습니다. 이는 사용자 데이터의 보안을 중요시하는 구글의 정책과 일치합니다.
2. **사용자 신뢰성 증가**:
   - HTTPS를 사용하면 사용자의 신뢰도가 증가합니다. 웹사이트 방문자가 주소창의 자물쇠 아이콘을 보면 해당 사이트가 안전하다는 인식을 갖게 되며, 이는 사용자 경험(UX) 향상으로 이어집니다. 좋은 사용자 경험은 자연스럽게 더 많은 방문과 낮은 이탈률로 이어져 SEO에 긍정적인 영향을 줍니다.
3. **데이터 무결성**:
   - HTTPS는 데이터 전송 중에 데이터를 암호화하여 중간에 변조되는 것을 방지합니다. 이는 검색 엔진이 웹사이트의 콘텐츠가 안전하게 사용자에게

전달될 것이라고 신뢰하게 만듭니다. 데이터 무결성을 보장하는 웹사이트는 검색 엔진으로부터 더 높은 평가를 받습니다.

4. **참조 데이터 보존**:
    - HTTPS 사이트로의 트래픽은 참조 헤더를 유지합니다. 이는 구글 애널리틱스와 같은 도구가 HTTPS 웹사이트로의 참조 트래픽을 제대로 추적할 수 있게 하며, 이는 사이트 성능을 정확히 측정하고 최적화 전략을 세우는 데 도움이 됩니다. 반면, HTTP에서 HTTPS로의 트래픽은 참조 데이터를 잃어버리게 됩니다.
5. **미래를 대비한 준비**:
    - 인터넷의 보안 표준이 점점 더 엄격해짐에 따라, HTTPS는 기본 요구 사항이 되어가고 있습니다. HTTPS를 미리 도입하면 향후 검색 엔진 알고리즘의 변화에 미리 대비할 수 있어 장기적인 SEO 전략에 유리합니다.

이와 같은 이유들로 인해 HTTPS를 도입하는 것이 검색엔진 최적화에 긍정적인 영향을 미치며, 이는 더 나은 검색 순위와 더 많은 웹 트래픽으로 이어질 수 있습니다.

HTTPS 적용 방법

SSL/TLS 인증서를 발급받아 웹 서버에 설치하고, 서버 설정을 변경하여 HTTPS를 활성화하는 단계로 구성됩니다.

1. **SSL/TLS 인증서 구매 및 설치**
    - 인증서를 제공하는 CA(Certificate Authority)에서 SSL/TLS 인증서를 구매합니다. 예를 들어, Let's Encrypt는 무료 인증서 제공합니다.
    - 인증서를 서버에 설치합니다. 이 과정은 사용하는 서버 소프트웨어(Apache, Nginx 등)에 따라 다릅니다.

        ▼ 인증서 설치 예 (Apache, Nginx)

        Apache 서버에 설치

        1. **인증서 파일 준비**: 발급받은 인증서 파일(CRT 파일)과 CA 번들 파일(또는 체인 파일)을 준비합니다.
        2. **Apache 설정 파일 수정**: Apache 설정 파일(httpd.conf 또는 apache2.conf)을 열어 다음과 같이 수정합니다.

```
<VirtualHost *:443>
    ServerName www.example.com
    DocumentRoot /var/www/html
    SSLEngine on
    SSLCertificateFile /path/to/your_domain_nam
e.crt
    SSLCertificateKeyFile /path/to/your_privat
e.key
    SSLCertificateChainFile /path/to/CA_bundle.
crt
</VirtualHost>
```

3. **Apache 재시작**: 설정을 적용하려면 Apache 서버를 재시작합니다.

```
sudo service apache2 restart
```

Nginx 서버에 설치

1. **인증서 파일 준비**: 발급받은 인증서 파일(CRT 파일)과 CA 번들 파일(또는 체인 파일)을 준비합니다.
2. **Nginx 설정 파일 수정**: Nginx 설정 파일(/etc/nginx/nginx.conf 또는 /etc/nginx/sites-available/default)을 열어 다음과 같이 수정합니다.

```
server {
    listen 443 ssl;
    server_name www.example.com;

    ssl_certificate /path/to/your_domain_name.c
rt;
    ssl_certificate_key /path/to/your_private.k
ey;
    ssl_trusted_certificate /path/to/CA_bundle.
crt;
```

```
    location / {
        root /var/www/html;
        index index.html index.htm;
    }
}
```

3. **Nginx 재시작**: 설정을 적용하려면 Nginx 서버를 재시작합니다.

```
sudo service nginx restart
```

2. **서버 설정 변경**
   - 서버 소프트웨어의 설정 파일을 수정하여 HTTPS를 사용하도록 설정합니다. 예를 들어, Apache의 경우 httpd.conf 또는 apache2.conf 파일을 수정해야 합니다.
   - HTTP 트래픽을 HTTPS로 리디렉션하도록 설정합니다. 이를 통해 사용자가 HTTP URL로 접속하더라도 자동으로 HTTPS로 전환됩니다.

     ▼ HTTP에서 HTTPS로 리디렉션 (Apache, Nginx)

     Apache에서 리디렉션 설정

     - .htaccess 파일을 사용하여 HTTP에서 HTTPS로 리디렉션합니다.

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUES
T_URI} [L,R=301]
```

     Nginx에서 리디렉션 설정

     - Nginx 설정 파일을 수정하여 HTTP에서 HTTPS로 리디렉션합니다.

```
server {
```

```
    listen 80;
    server_name www.example.com;
    return 301 https://$host$request_uri;
}
```

3. **HTTPS 설정 테스트 및 검증**
    - 웹사이트가 제대로 HTTPS로 작동하는지 테스트합니다. 웹 브라우저에서 사이트를 열어 보고, 브라우저 주소창에 자물쇠 아이콘이 나타나는지 확인합니다.
    - SSL/TLS 검사 도구를 사용하여 인증서 설치 및 설정이 올바른지 확인합니다. 예를 들어, SSL Labs의 SSL 테스트를 사용할 수 있습니다.
4. **사이트 및 링크 업데이트**
    - 웹사이트 내 모든 URL을 HTTPS로 업데이트합니다. 내부 링크, 이미지, 스크립트 등 모든 리소스가 HTTPS를 사용하도록 수정합니다.
    - 외부 리소스(예: 서드파티 API, CDN 등) 역시 가능하다면 HTTPS로 제공되는지 확인하고 수정합니다.
5. **SEO 설정 및 알림**
    - 구글 서치 콘솔 및 기타 검색 엔진 도구에서 HTTPS 버전의 웹사이트를 등록합니다.
    - 사이트맵을 HTTPS URL로 업데이트하고 검색 엔진에 제출합니다.

        ▼ 사이트 맵

        사이트맵(Sitemap)은 웹사이트의 구조와 콘텐츠를 검색 엔진에 효과적으로 전달하여 크롤링과 인덱싱을 최적화하는 중요한 도구입니다. 이를 통해 웹사이트의 SEO 성능을 향상시킬 수 있습니다.

        사이트맵은 웹사이트의 페이지, 콘텐츠, 구조 등을 나열한 파일입니다. 이는 검색 엔진이 웹사이트를 보다 효과적으로 크롤링하고 인덱싱할 수 있도록 돕는 역할을 합니다. 사이트맵은 주로 XML 형식으로 작성되며, 검색 엔진 봇이 사이트 구조를 이해하는 데 유용합니다.

        사이트맵의 주요 기능

        1. **검색 엔진 크롤링 개선**:

- 사이트맵은 검색 엔진이 웹사이트의 모든 페이지를 찾아서 크롤링하도록 도와줍니다. 특히, 사이트 구조가 복잡하거나 동적 콘텐츠가 많은 사이트의 경우 유용합니다.

2. **콘텐츠 업데이트 알림**:
   - 사이트맵은 웹사이트의 페이지가 언제 업데이트되었는지를 검색 엔진에 알릴 수 있습니다. 이를 통해 새로운 콘텐츠나 수정된 콘텐츠가 빠르게 인덱싱될 수 있습니다.

3. **웹사이트 구조 제공**:
   - 사이트맵은 웹사이트의 계층 구조와 각 페이지의 중요도를 검색 엔진에 전달합니다.

사이트맵의 형식

사이트맵은 주로 XML 형식으로 작성되지만, HTML 형식으로도 제공될 수 있습니다. 아래는 XML 사이트맵의 예입니다:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
    <url>
        <loc>https://www.example.com/</loc>
        <lastmod>2023-06-01</lastmod>
        <changefreq>monthly</changefreq>
        <priority>1.0</priority>
    </url>
    <url>
        <loc>https://www.example.com/about</loc>
        <lastmod>2023-05-20</lastmod>
        <changefreq>monthly</changefreq>
        <priority>0.8</priority>
    </url>
    <!-- 추가 페이지들 -->
</urlset>
```

사이트맵 생성 및 제출 방법

1. **사이트맵 생성**:
    - 수동으로 생성: XML 파일을 직접 작성하여 서버에 업로드합니다.
    - 자동 생성 도구 사용: 다양한 온라인 도구와 CMS 플러그인을 사용하여 사이트맵을 자동으로 생성할 수 있습니다. 예를 들어, WordPress 사용자는 Yoast SEO 플러그인을 통해 사이트맵을 생성할 수 있습니다.
2. **사이트맵 파일 업로드**:
    - 생성한 사이트맵 파일을 웹 서버의 루트 디렉토리에 업로드합니다. 일반적으로 sitemap.xml이라는 이름으로 저장합니다.
3. **사이트맵 검색 엔진에 제출**:
    - 구글 서치 콘솔: 구글 서치 콘솔에 로그인하여 'Sitemaps' 섹션에서 새로운 사이트맵 URL을 제출합니다.
        - 예: https://www.example.com/sitemap.xml
    - Bing 웹마스터 도구: Bing 웹마스터 도구에서도 사이트맵을 제출할 수 있습니다.

- HTTPS로 변경된 사실을 사용자 및 관련 당사자에게 알립니다.