# Special Report

## 웹 취약점과 해킹 매커니즘 #1 개요

### ■ 개요

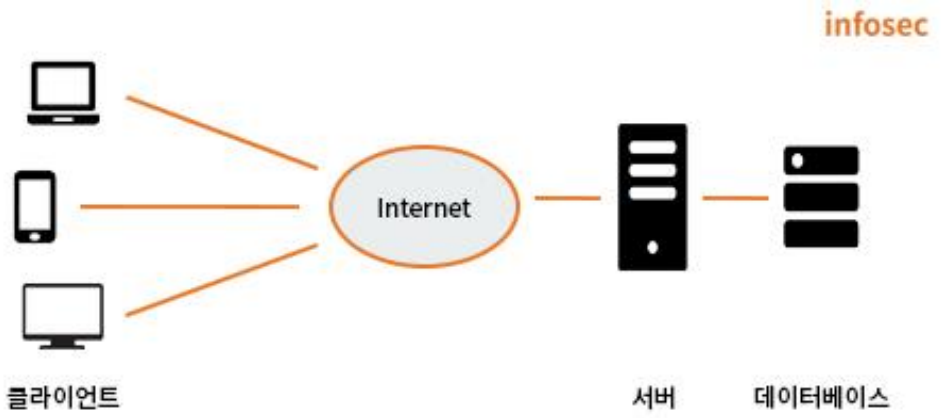
‘웹 취약점과 해킹 매커니즘’ 이라는 주제로 새롭게 시작하는 Special Report는 주요 웹 취약점인 SQL Injection, 크로스 사이트 스크립팅, 파일 다운로드 등 취약점이 발생하는 원리와 이러한 취약점에 대한 공격 방법 및 공격을 방어할 수 있는 시큐어 코딩에 대해 알아본다.

웹에서 발생할 수 있는 취약점이 다양하기 때문에 매월 시리즈물로 발간될 예정이며, 이를 통해 EQST Insight 구독자들은 웹 취약점에 대한 이해와 함께 취약점 조치 방안을 알 수 있다.

이번 3월호 Special Report에서는 웹 해킹에 필요한 기본 지식들에 대해 설명하고 웹에서 사용되는 기본 용어 설명, 웹의 동작 방식, 웹 해킹에 사용되는 도구 등에 대해 알아보고자 한다.

## ■ 웹(World Wide Web, WWW)과 HTTP(HyperText Transfer Protocol)

웹은 인터넷을 통해 연결된 사용자들이 정보를 공유할 수 있는 정보 공간이며 클라이언트/서버 구조[1]로 동작한다. 클라이언트는 서비스를 사용하는 사용자 혹은 사용자의 단말기를 가리킨다. 인터넷 익스플로러, 크롬 등의 브라우저는 웹 서버로 접속해 화면 출력을 위해 웹 페이지를 요청하는 대표적인 클라이언트이다. 이때 클라이언트의 요청에 서비스를 응답해 주는 것이 바로 서버이다.
웹 서버와 클라이언트는 서로 통신을 하기 위해 HTTP 프로토콜을 사용한다. HTTP는 웹 서버와 클라이언트 사이의 통신을 위해 사용하는 통신 프로토콜로, 인터넷상의 하이퍼텍스트 문서 교환을 위해 사용되며 암호화된 HTTP를 HTTPS라고 한다.




[클라이언트/서버 구조]

HTTP 프로토콜은 인터넷상 불특정 다수의 클라이언트와 통신을 하는데, 이때 연결을 계속 유지한다면 많은 리소스가 발생하게 된다. 이러한 문제를 해결하기 위해 클라이언트의 요청에 대한 서버에 응답이 끝나면 연결을 끊어버리는 비연결성(Connectionless)이라는 특징을 갖는다. 이로 인해 서버는 클라이언트를 식별할 수 없어 사용자의 상태 정보를 기억하지 못하는 무상태(Stateless)의 특징도 나타난다.

하지만 웹 서비스 운영 시 로그인 유지 등과 같이 사용자의 상태 정보를 기억해야 하는 경우가 많아졌으며, HTTP는 이러한 문제점을 해결하기 위한 기술인 '쿠키(Cookie)[2]', '세션(Session)[3]'을 사용하게 되었다.

---

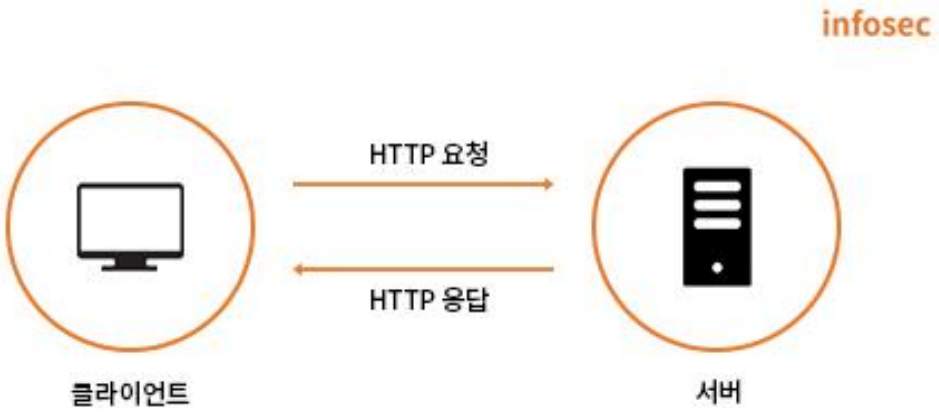
[1] 서비스를 요청하는 클라이언트와 클라이언트의 요청을 처리하는 서버의 협동작업을 통해 사용자가 원하는 결과를 얻는 처리 방식이다.

[2] 쿠키는 클라이언트 측에 사용자 정보를 저장하기 때문에 공격자로부터 위변조의 가능성이 높아 보안에 취약하다.

[3] 세션은 서버 측에 사용자 정보를 저장하기 때문에 쿠키보다는 안전하지만, 세션 정보도 탈취당할 수 있다.

## ■ HTTP 요청과 응답

HTTP는 서버/클라이언트 모델을 따르며 요청(Request)와 응답(Response) 형태로 구성되어 있다. 클라이언트는 웹 브라우저를 통해 웹 서버로 요청을 전송하고 웹 서버는 이에 대한 응답을 클라이언트에게 전송한다.




[HTTP 요청과 응답]

### HTTP 요청

HTTP 요청은 클라이언트가 서버에게 특정 동작을 요청하기 위해 전송하는 메시지이며 요청 페이지와 함께 서버에 전달하는 클라이언트의 정보를 포함하고 있다.

1) HTTP 요청 헤더 구성

```
1  GET /business/expert/eqst.do HTTP/1.1
2  Host: infosec.adtcaps.co.kr
3  Cookie: JSESSIONID=0E02DE7F0B79B7EBD6DEE9338CE17EE7; _ga=GA1.3.306907063.1646632323; _gid=GA1.3.3
4  Cache-Control: max-age=0
5  Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="98"
6  Sec-Ch-Ua-Mobile: ?0
7  Sec-Ch-Ua-Platform: "Windows"
8  Upgrade-Insecure-Requests: 1
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*
11 Sec-Fetch-Site: cross-site
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Referer: https://www.google.com/
16 Accept-Encoding: gzip, deflate
17 Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
```

[HTTP 요청 헤더]

· GET /business/expert/eqst.do HTTP/1.1 : 요청 URL 정보 및 HTTP 버전

· Host : 요청 도메인

· Cookie : 클라이언트 측에 저장된 사용자 상태 정보

· User-Agent : 사용자의 웹 브라우저 종류

· Accept : 요청 데이터 타입

· Referer : 요청을 보낸 페이지의 URL

2) HTTP 요청 메소드

HTTP 요청 헤더 중 요청 메소드를 통해 클라이언트가 웹 서버에게 요청의 목적과 종류를 알린다. 주로 GET, POST 방식으로 자원을 요청한다. TRACE, PUT, DELETE와 같은 메소드는 사용자가 웹 서비스를 이용할 때 필요하지 않기 때문에 설정되어 있을 경우 취약점이 되기도 한다.

infosec

| Method | 의미 |
|---|---|
| GET | 서버 측에 자원 요청 |
| POST | 서버로 자원 전송 |
| HEAD | HTTP 헤더 정보만 수신 |
| TRACE | 원격지 서버에 루프백 테스트 |
| PUT | 요청된 자원 갱신 |
| DELETE | 요청된 자원 삭제 |
| OPTIONS | 응답 가능한 HTTP 메소드 요청 |

[HTTP 요청 메소드]

## HTTP 응답

서버는 클라이언트로부터 요청이 오면 응답 헤더의 정보와 바디의 데이터를 포함하여 요청에 대한 응답을 한다.

1) HTTP 응답 헤더 구성


[HTTP 응답 헤더]

· HTTP/1.1 200 OK : HTTP 버전과 응답 코드

· Server : 웹 서버 정보

· Content-Length : 응답 패킷의 길이

· Content-Type : MIME 타입[4]

[4] MIME(Multipurpose Internet Mail Extensions)타입은 파일 변환을 뜻하며 'text/css', 'text/xml', 'Application/javascript' 등의 표현으로 응답하는 자원의 컨텐츠 타입을 나타낸다.

2) HTTP 응답 코드(상태 코드)

서버는 클라이언트가 보낸 HTTP 요청에 대한 응답 코드를 보내는데 이를 보고 요청의 성공과 실패 여부와 같은 서버의 상태를 판단할 수 있다. 응답 코드는 100번대부터 500번대까지의 세 자리 숫자로 구성되며 이 중 클라이언트 오류를 나타내는 400번대 코드와 서버 오류를 나타내는 500번대 코드를 주의 깊게 봐야 한다. 아래의 표는 자주 볼 수 있는 HTTP 응답 코드의 몇 가지 예시이다.

infosec

| 구분 | 응답 코드 | 응답 메시지 | 의미 |
|---|---|---|---|
| 1xx: 정보 | 100 | Continue | 진행 중 |
| 2xx: 성공 | 200 | OK | 요청에 대한 성공 |
| | 201 | Created | 요청 자원이 정상적으로 생성이 됨 |
| 3xx: 리다이렉션 | 301 | Moved Permanently | 요청한 자원이 새 URL에 존재 |
| | 302 | Found | 임시적으로 주소가 바뀌었을 경우 |
| 4xx: 클라이언트 오류 | 400 | Bad Request | 잘못된 요청 |
| | 401 | Unauthorized | 권한 없는 요청 |
| | 403 | Forbidden | 서버에서 해당 자원에 대한 접근 금지 |
| | 404 | Not Found | 요청 자원이 서버에 존재하지 않음 |
| 5xx: 서버 오류 | 500 | Internal Server Error | 내부 서버 오류 |
| | 502 | Bad Gateway | 게이트웨이로부터 잘못된 응답 수신 |
| | 503 | Service Unavailable | 현재 서버를 사용할 수 없음 |
| | 504 | Gateway Timeout | 게이트웨이가 응답을 받지 못함 |

[HTTP 응답 코드]

## ■ 사용 툴 소개

일반적인 사용자가 보는 화면은 웹 브라우저 하나지만 실제 요청 시 많은 데이터가 전달되는데, 이를 변조하고 결과를 확인하기 위해서 공개된 툴을 사용하는 것이 좋다. 웹 취약점 진단 시 사용하는 툴의 종류는 다양하게 존재하지만, Special Report에서는 주로 웹 프록시[5] 툴인 '버프 스위트(Burp Suite)'와 브라우저의 확장 기능 중 하나인 '개발자 도구'가 사용된다.

### 버프 스위트(Burp Suite)

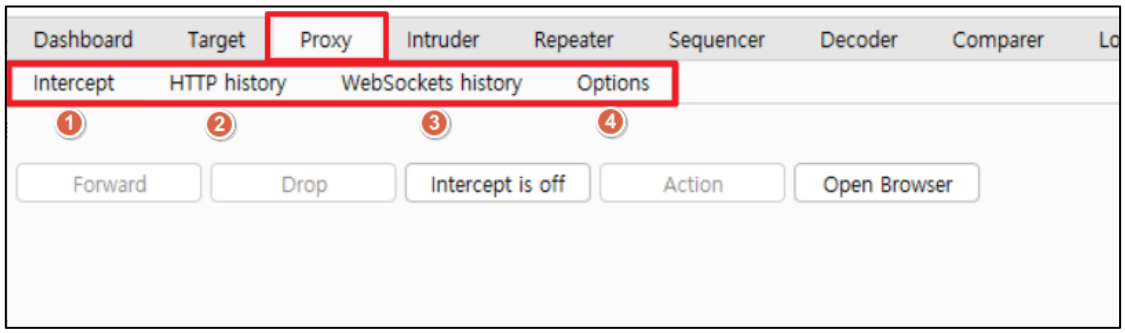
조작할 수 있게 해주는 웹 프록시 툴이다. 공격자는 버스 스위트를 통해 브라우저에서 서버로 전송되는 요청 패킷을 확인하고 변조할 수 있으며, 서버에서 브라우저로 전송되는 응답 패킷을 가로채 보안 로직을 삭제하거나 변경하는 것이 가능하다.


[버프 스위트 동작 원리]

또한 버프 스위트는 애플리케이션 단에서 패킷을 받고 응답을 전송하기 때문에 SSL 적용이 되지 않은 상태이므로 평문으로 나타난다.

[5] 프록시 서버는 클라이언트와 서버의 중간에서 통신을 매개하는 역할을 한다.

버프 스위트의 Proxy 기능은 다음과 같다.




[버프 스위트 - Proxy]

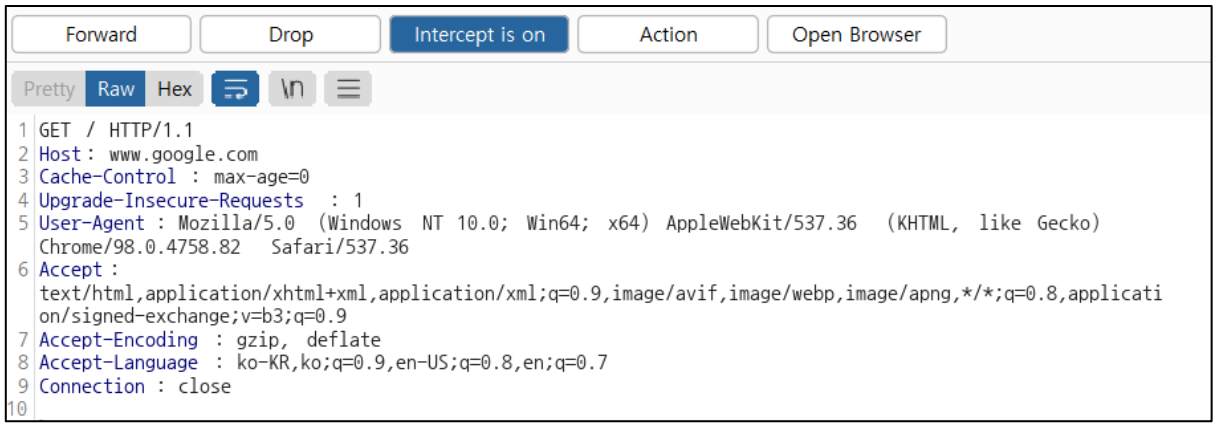
①Intercept : 패킷을 가로채서 변조하고 응답을 확인하는 곳

②HTTP history : 웹으로 주고받은 데이터들이 쌓이는 곳으로 통신 로직 분석 가능

③WebSockets history : 웹 소켓을 통해 주고받은 데이터가 쌓임

④Options : 웹 프록시 기능을 하기 위해 설정 변경을 하는 곳

Proxy > Intercept > Intercept is on 을 통해 웹 서버로 요청되는 패킷을 중간에서 확인할 수 있으며 조작이 가능하다



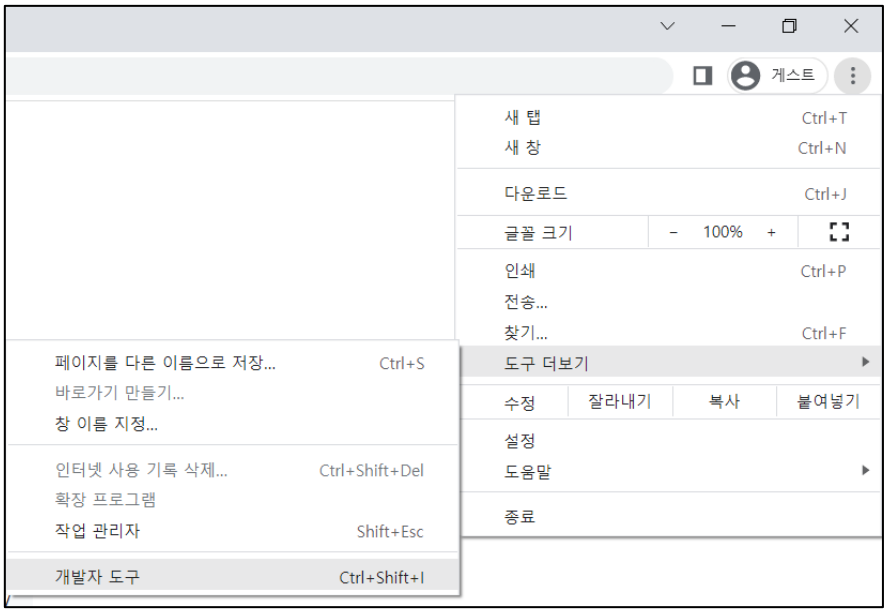

[Proxy - Intercept is on]

이외에도 사용자가 정의한 자동화 공격을 수행할 수 있는 Intruder 기능, HTTP 요청을 편집하고 재전송해서 응답을 볼 수 있는 기능인 Repeater 등이 주로 사용된다.

## 개발자 도구

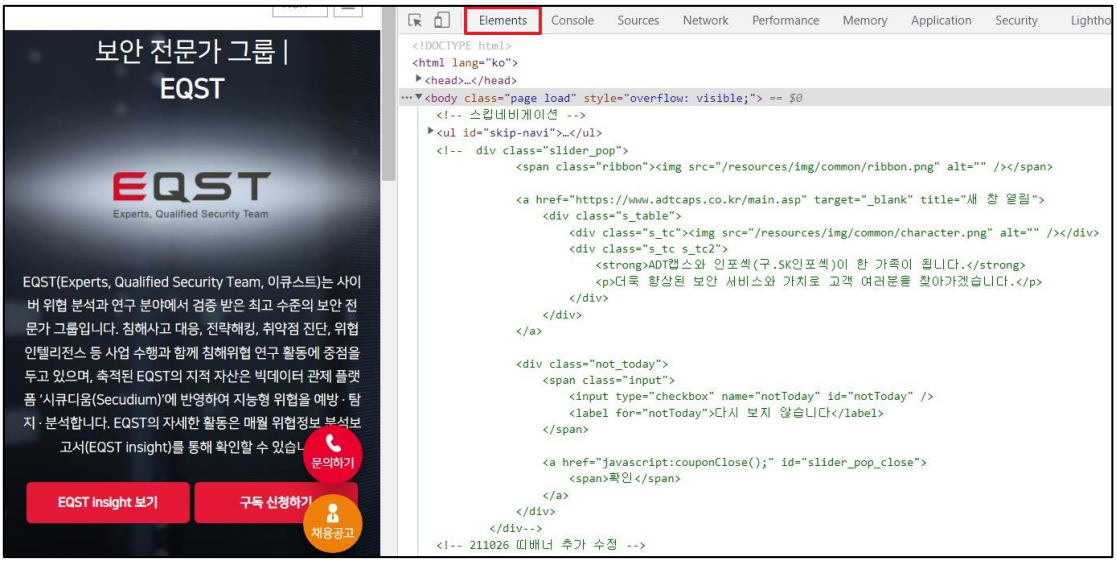
주요 브라우저들은 개발자 도구를 기본적으로 제공해 준다. 본 인사이트에서는 Chrome 브라우저를 사용하며, 개발자 도구를 실행하는 방법은 다음과 같다.

브라우저 더보기 > 도구 더보기 > 개발자 도구 (단축키 F12)



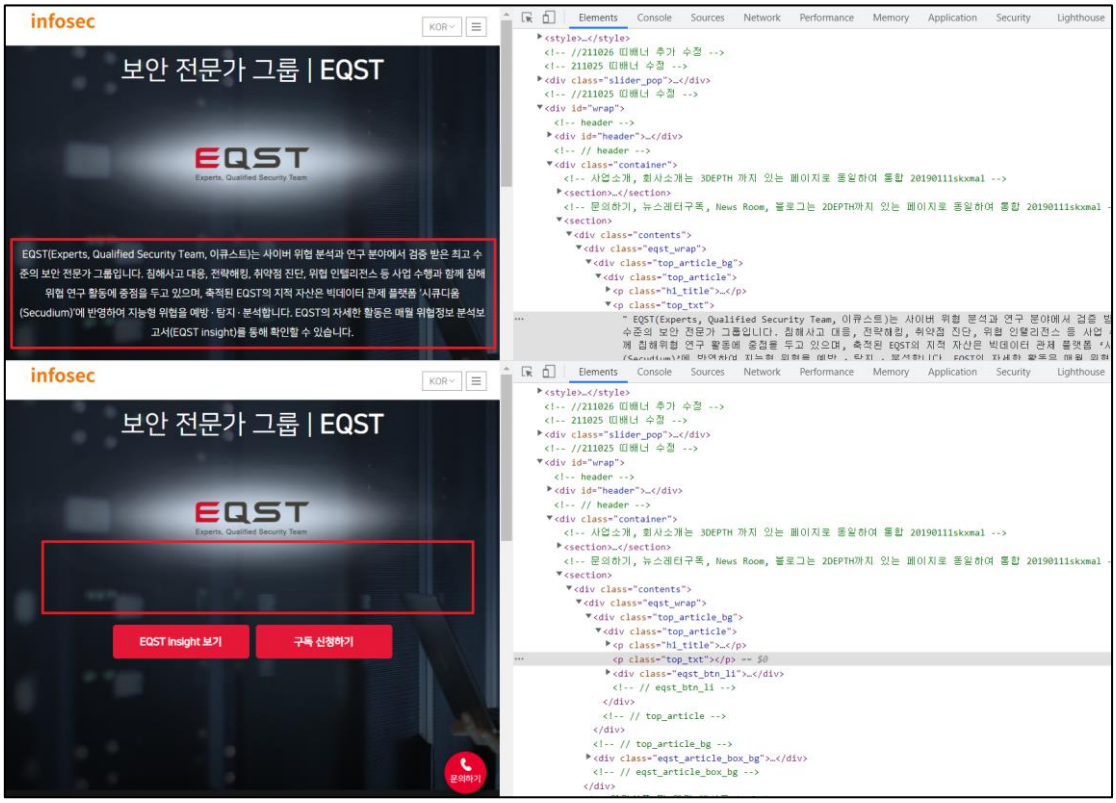

[개발자 도구 실행]

아래의 그림을 보면 왼쪽은 사용자가 보는 웹 페이지이고 오른쪽은 개발자 도구의 Elements 창이며 해당 페이지를 구성하고 있는 소스코드를 확인할 수 있다.



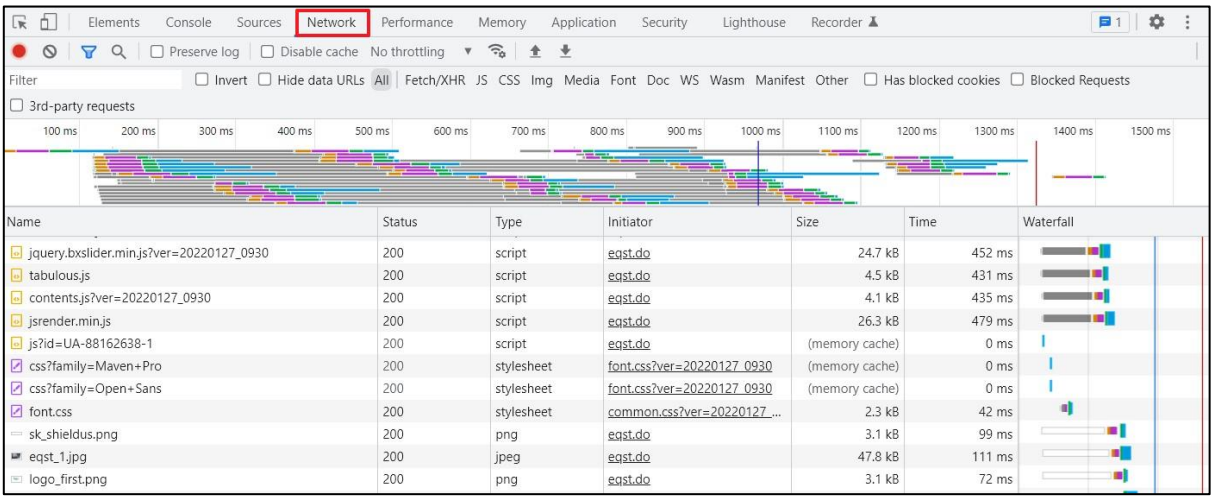

[개발자 도구 - Elements]

이곳에서 HTML, CSS, 자바스크립트 코드를 수정하여 일시적으로 화면을 조작하거나 패킷을 변경하여 결과를 확인할 수 있다.




[소스코드 변경을 통한 페이지 조작]

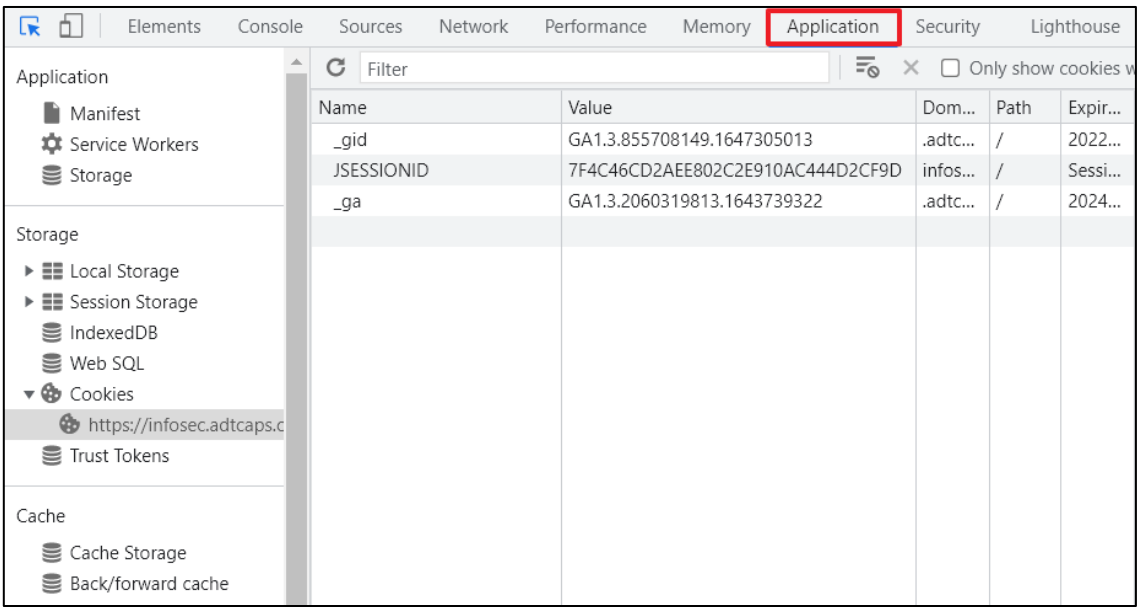
Network 창에서는 브라우저와 서버 사이의 HTTP 패킷의 흐름을 파악할 수 있다.




[개발자 도구 – Network]

이외에도 오류 메시지를 확인할 수 있으며 자바스크립트 코드를 직접 입력해서 실행할 수 있는 Console 창, 현재 페이지를 구성하는 스토리지 및 쿠키의 정보를 확인할 수 있는 Application 창 등 다양한 기능을 제공한다.


[개발자 도구 - Console]




[개발자 도구 - Application]

## ■ 맺음말

이번 달부터 새로 시작되는 EQST Insight의 Special Report – '웹 취약점과 해킹 메커니즘' 연재에 앞서 웹에 대한 기본적인 내용을 살펴보았다. 앞으로의 리포트에서는 웹에서 취약한 소스코드 사용 시 발생할 수 있는 공격과 해킹이 어떻게 이루어지는지, 어떻게 하면 안전한 소스코드를 구성할 수 있는지에 대한 내용을 다룰 것이다.