

인증과 인가

2차 기술 세미나

2024. 02 . 20

Cloud Service

박선주

목차

인증과 인가의 개념

- 인증과 인가의 정의
- 인증과 인가의 활용

인증의 구현

- 인증 상태 저장 방식
 1. 쿠키방식
 2. Session 방식
- HTTP 인증 프레임워크
 3. Basic 방식
 4. Bearer 방식 - JWT
 5. Bearer 방식 - OAuth

+ 추가 인증

인증

Authentication

인증하다

authenticate

‘서비스에게 자신이 무엇이라고 스스로를 증명하는 것’

인증의 결과에 따라
‘현재 가지는 권한’이 정해진다.

認 알 인, 적을 잉

부수 言 총획 14획

1. (알 인)
2. 알다
3. 인식하다(認識--)

證 증거 증

부수 言 총획 19획

1. 증거(證據)
2. 증명서(證明書)
3. 병상(病狀), 병세(病勢)

동사

verify >

validate >

prove to be
genuine >

certify >

substantiate >

validate >

ratify >

confirm >

seal >

sanction >

인가

Authorization

(인증이 된 사용자라도)
'특정 서비스에 접근할 권한을
가지고 있는지 확인하는 것'

인가의 결과에 따라
'무엇을 할 수 있는지' 정해진다.

認 알 인, 적을 잉

부수 言 총획 14획

1. (알 인)
2. 알다
3. 인식하다(認識--)

可 옳을 가, 오락케 임금 이름 극

부수 可 총획 5획

1. (옳을 가)
2. 옳다
3. 허락하다(許諾--)

인가하다

authorize

동사

give
permission for
>

permit >

sanction >

allow >

agree to >

⋮

give someone
the authority >

give someone
permission >

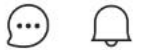
mandate >

commission >

empower >

⋮

인증과 인가의 활용



네이버를 더 안전하고 편리하게 이용하세요

NAVER 로그인

[아이디 찾기](#) | [비밀번호 찾기](#) | [회원가입](#)

뉴스스탠드 · 언론사편집 / 엔터 / 스포츠 / 경제

AD | X

한국어

NAVER

PC방 등 공용PC라면 QR코드 로그인에 더 안전해요 x

ID 로그인

일회용 번호

QR코드

id12345

.....

☒ 로그인 상태 유지

IP보안 ☐

로그인

[비밀번호 찾기](#) | [아이디 찾기](#) | [회원가입](#)



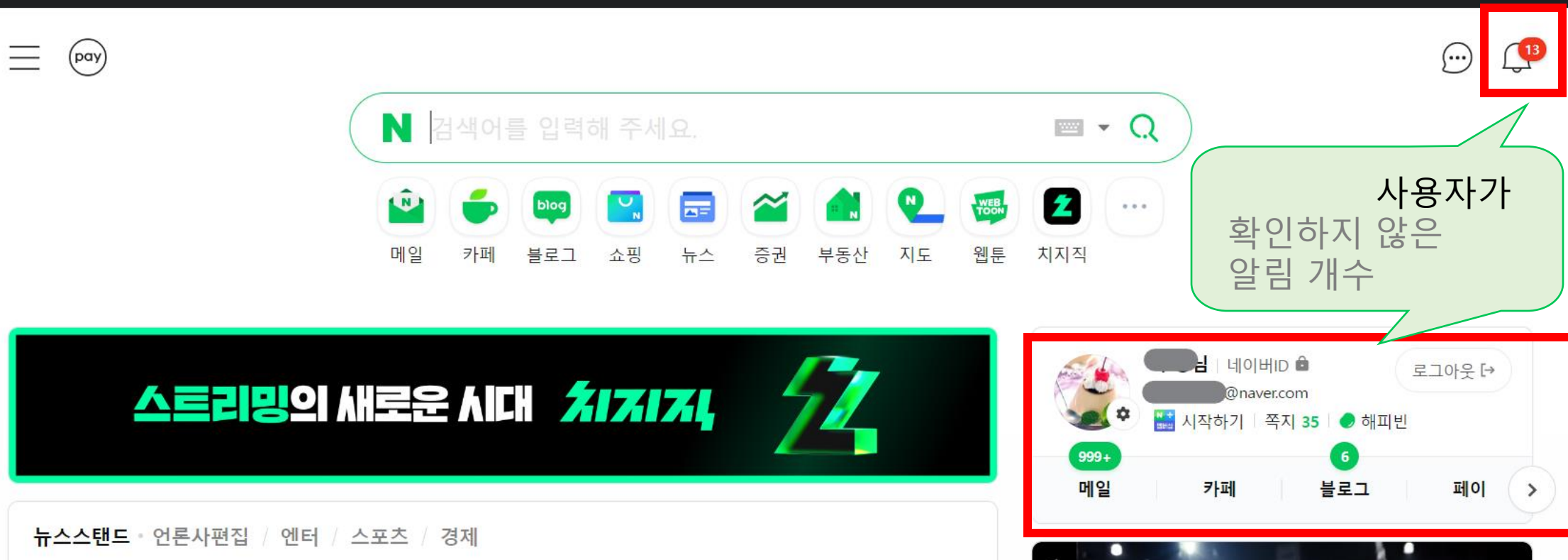
안전하게 내 정보 지키는 법

실명인증하고 보안 강화

NAVER

인증(로그인)을 통해 사용자에게
개인화된 서비스를 제공할 수 있다.

현재 로그인된 사용자는
자신의 메일 읽기, 블로그 글쓰기...
해당하는 권한이 **인가**되어 있다.
(다른 사람의 메일 읽기, 블로그 글쓰기... 는 권한이 없다.)



인증의 구현

인증 상태 저장 방식

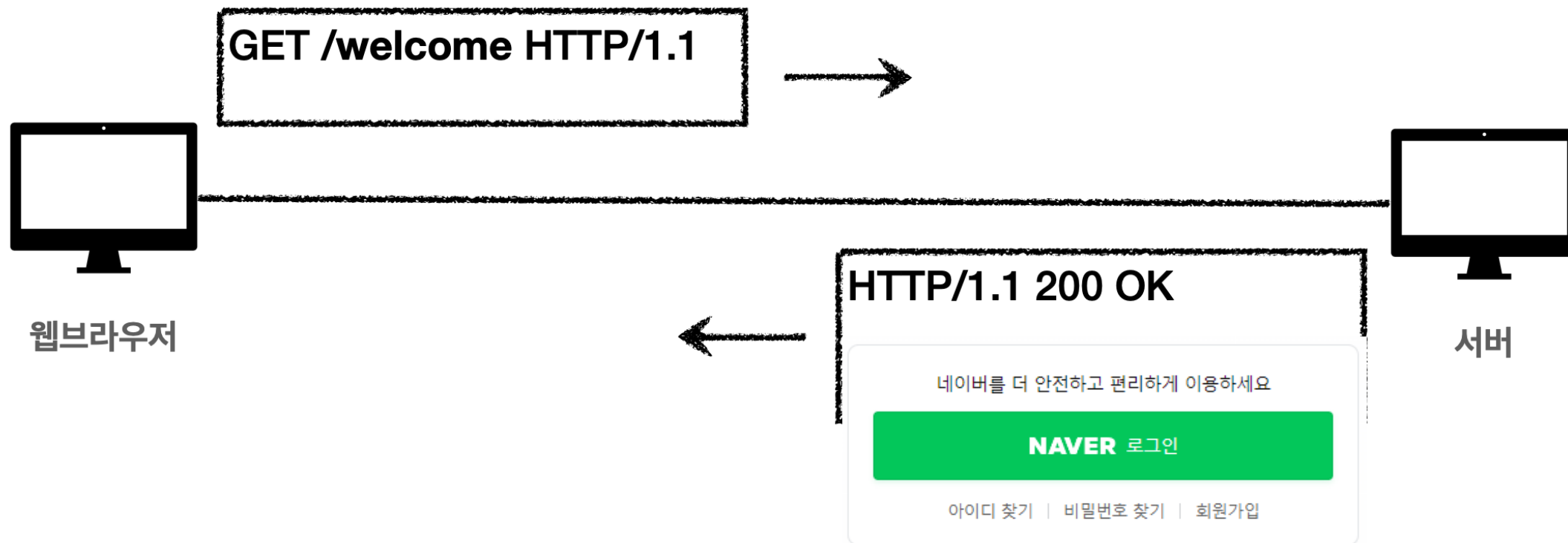
1. 쿠키방식
2. Session 방식

HTTP 인증 프레임워크

1. Basic 방식
2. Bearer 토큰 방식 (JWT, OAuth)

로그인 구현

처음 welcome 페이지 접근



로그인 구현

로그인

POST /login HTTP/1.1
user=홍길동

NAVER

로그인

아이디

비밀번호

로그인 상태 유지

로그인



웹브라우저



서버

HTTP/1.1 200 OK

로그아웃

네이버ID

@naver.com

시작하기 | 쪽지 35 | 해피빈

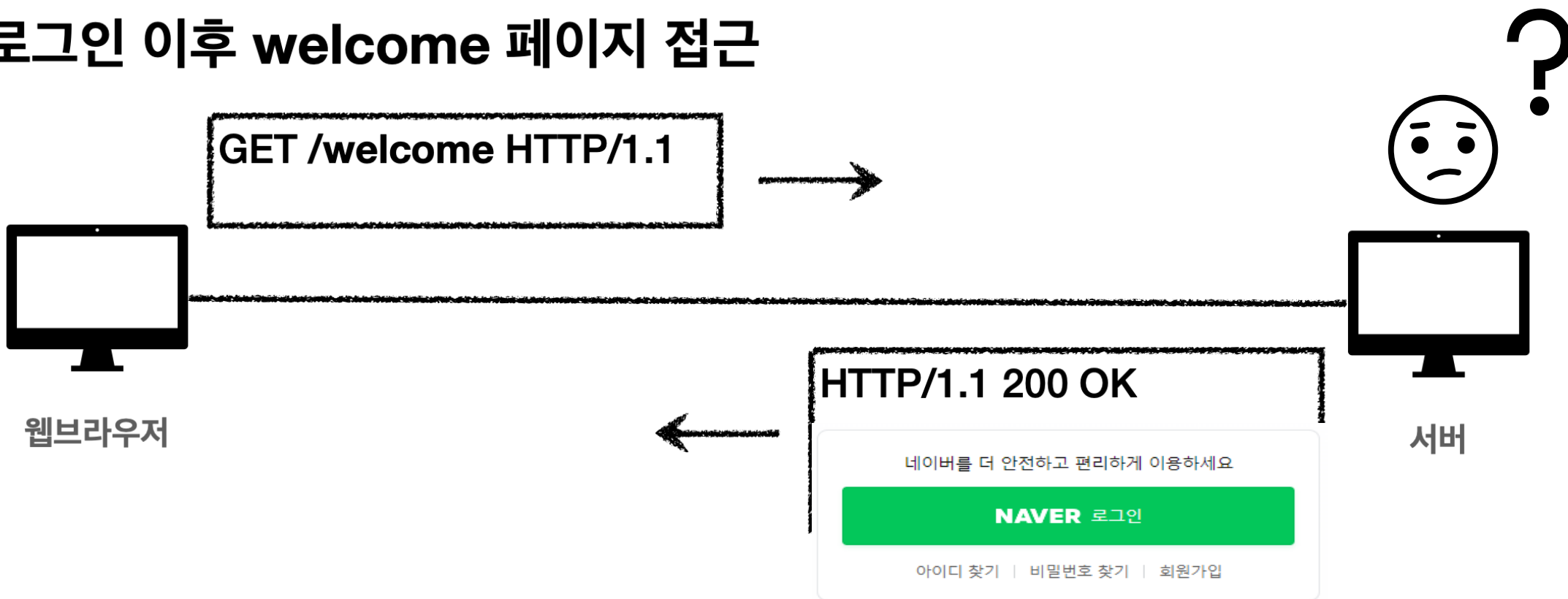
999+

메일 | 카페 | 블로그 | 페이지

HTTP – 무상태 프로토콜

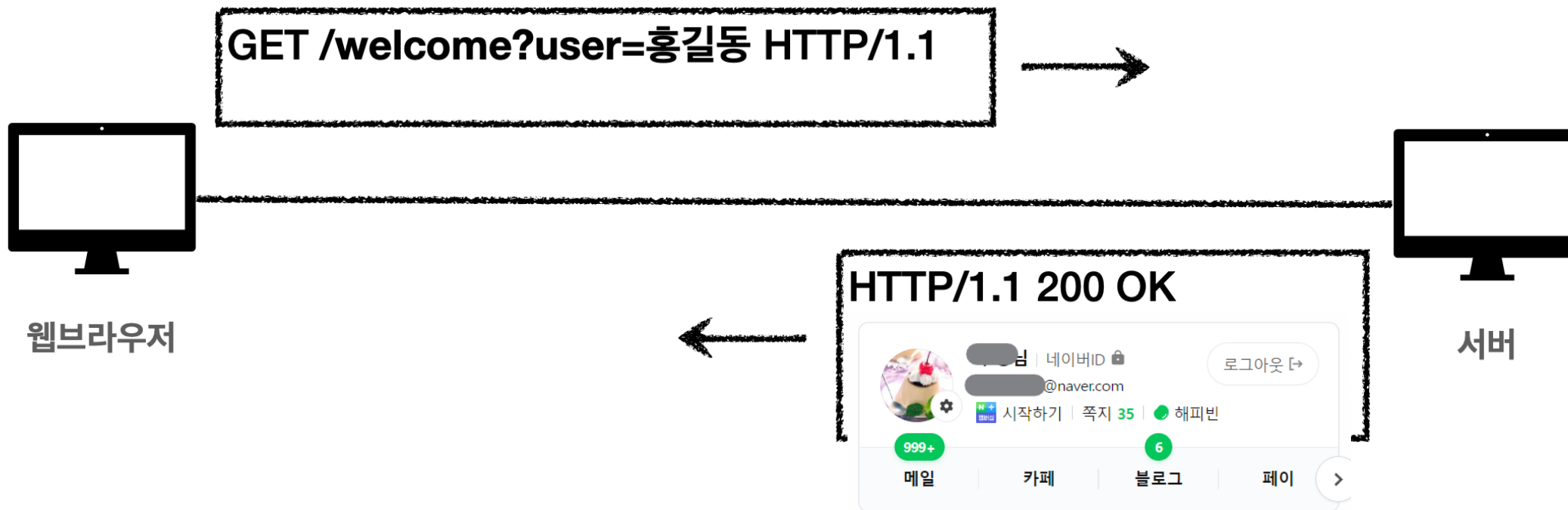
무상태 (Stateless) : 클라이언트와 서버가 이전 요청과 응답을 기억하지 않는다.

로그인 이후 welcome 페이지 접근



HTTP - 무상태 프로토콜

대안 - 모든 요청에 사용자 정보 포함

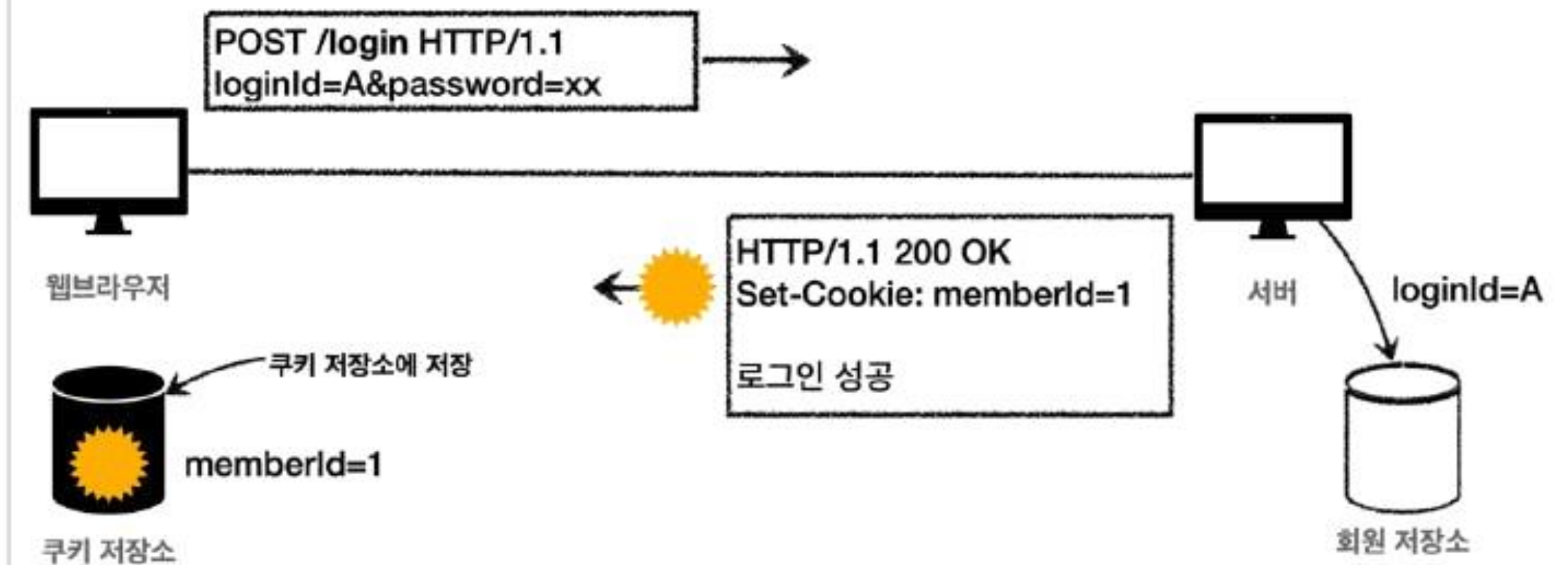


1.Cookie 방식

클라이언트는 요청을 보낼 때마다, 저장 되어있는 쿠키를 요청 헤더의 Cookie에 담아 보낸다.
서버는 쿠키에 담긴 정보를 바탕으로 해당 요청의 클라이언트가 누구인지 식별한다.

쿠키 생성

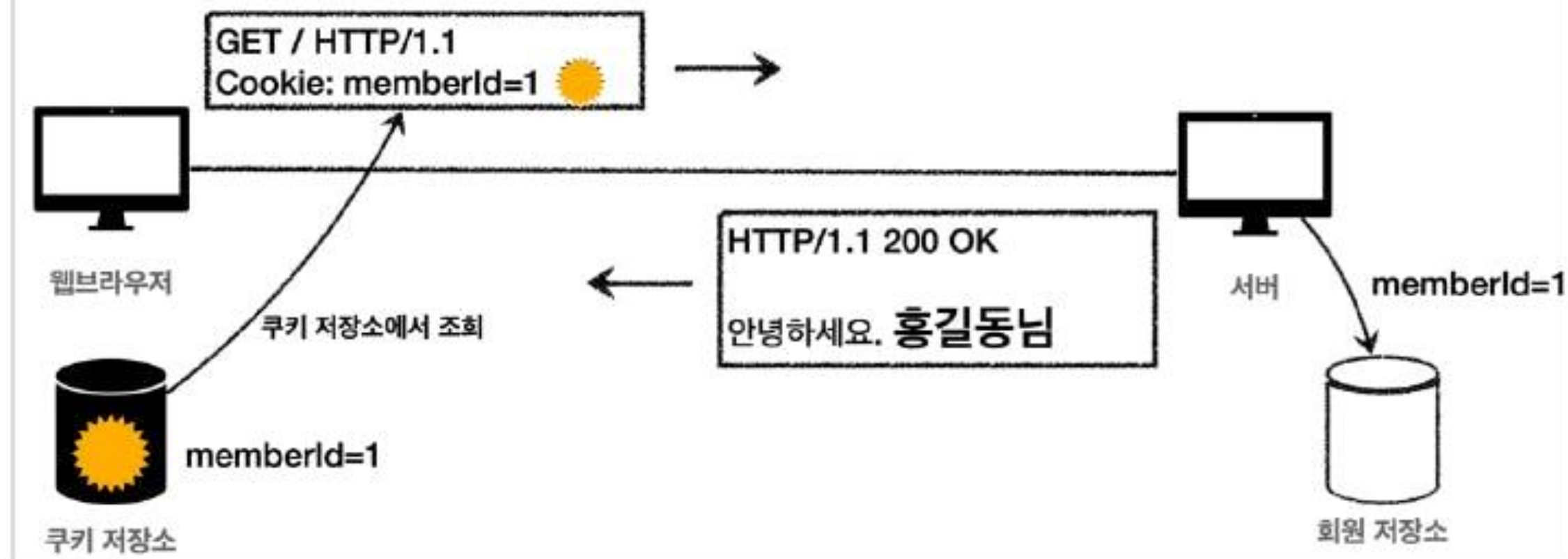
쿠키 로그인



클라이언트 쿠키 전달1

쿠키

로그인 이후 **welcome** 페이지 접근



Cookie 방식의 단점

쿠키의 단점을 그대로 가진다.

- 용량 제한
- 브라우저 간 쿠키 지원 형태가 다름

요청 시 쿠키의 값을 그대로 보낸다.

- 쿠키의 크기 인한 네트워크 부하
- 요청 때마다 개인정보가 노출 될 수 있다.

클라이언트가 쿠키를 조작할 수 있다.

Cookie: memberId=1 -> Cookie: memberId=2
(다른 사용자의 이름이 보임)

모든 요청에 쿠키 정보 자동 포함



웹브라우저



쿠키 저장소

GET /welcome HTTP/1.1

Cookie: memberId=1

GET /board HTTP/1.1

Cookie: memberId=1

GET /order HTTP/1.1

Cookie: memberId=1

GET /xxx... HTTP/1.1

Cookie: memberId=1

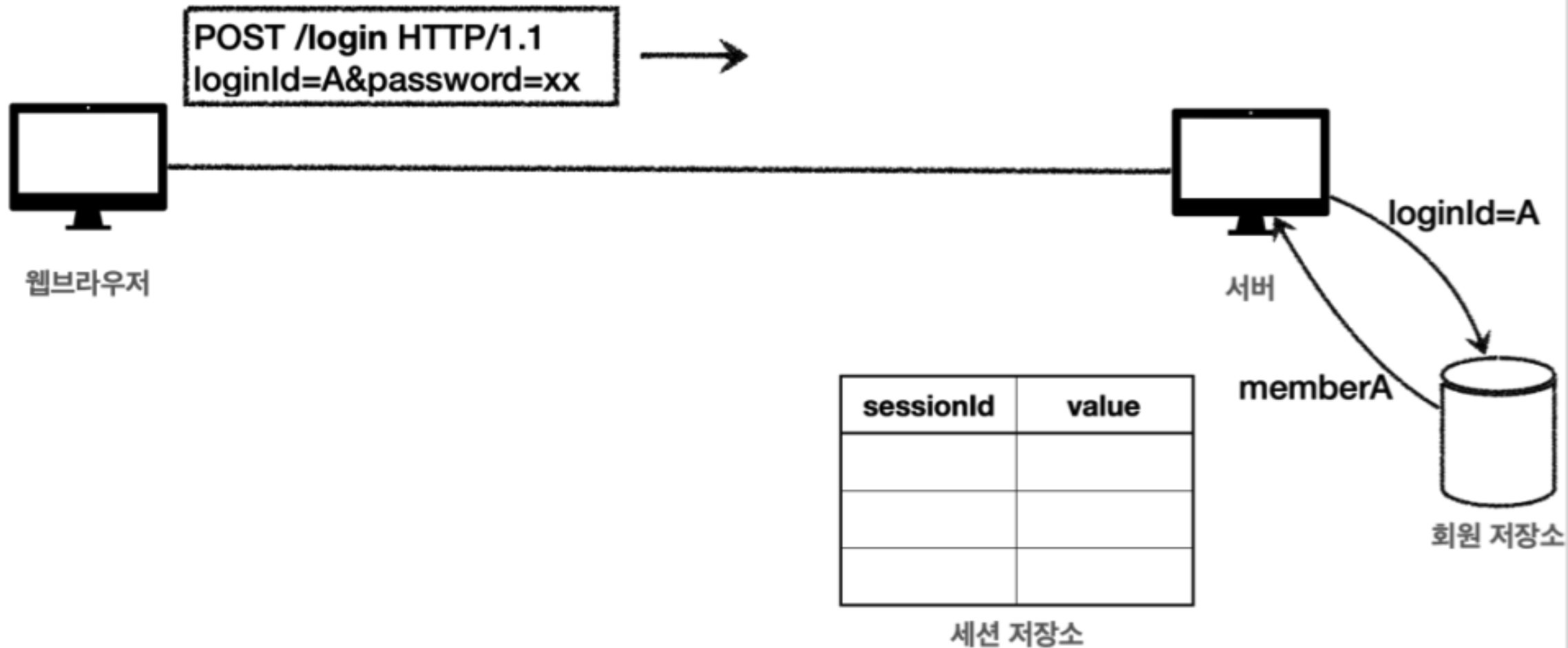
memberId=1

2. Session 방식

요청 때마다 개인정보가 노출 될 수 있다는 쿠키의 위험을 보완하기 위한 방법 중 하나
비밀번호 등의 개인정보는 서버측에만 저장하고 관리한다.

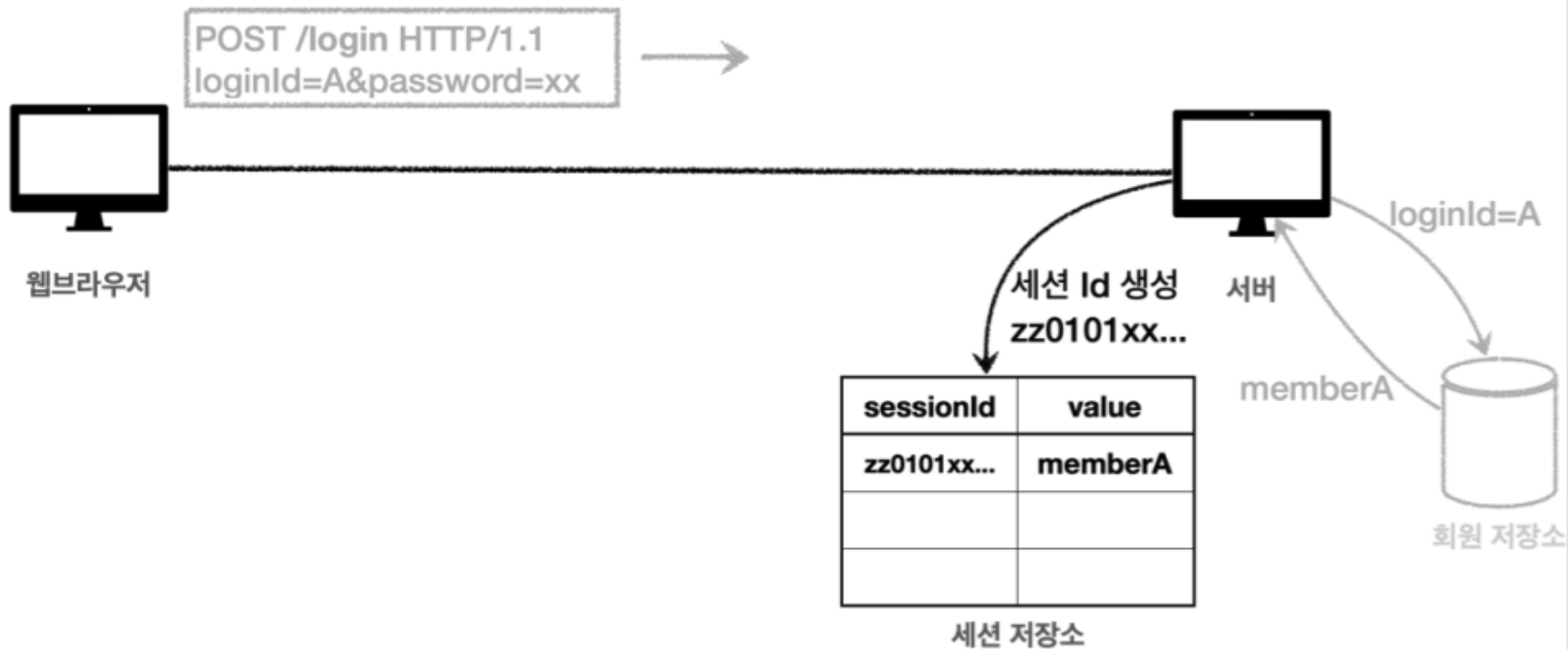
서버측에서 클라이언트를 인증했을 때 클라이언트에게 세션ID를 만들어주어
다음 번 요청에 이미 인증이 되었음을 알릴 수 있도록 해준다.

세션 로그인



세션

세션 관리

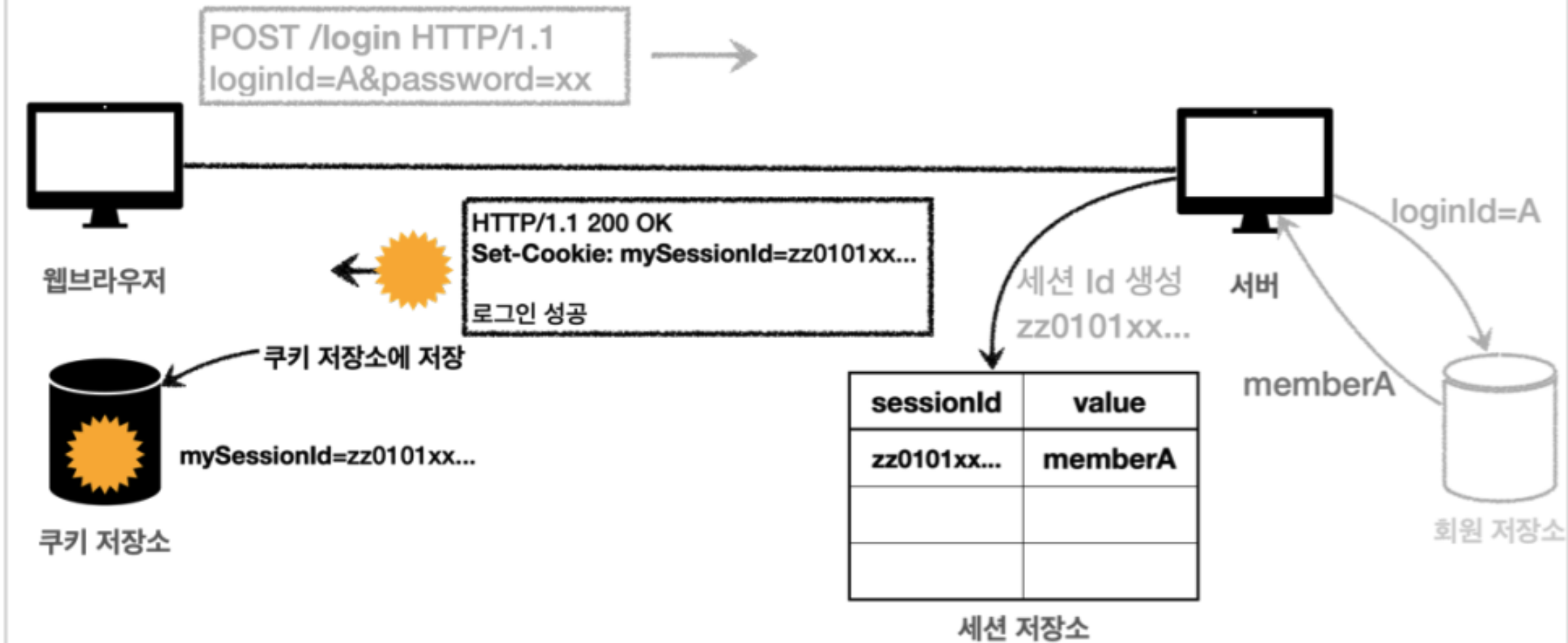


세션

세션id를 쿠키로 전달

세션ID 만으로는 값의 추정이 불가능하다.

Cookie:mySessionId=zz0101xx-b ... == 홍길동 ??



Session 방식의 단점

서버에서 세션 저장소를 사용하므로 요청이 많아지면 서버에 부하가 심해진다.

쿠키 방식과 같이 자체로 개인정보를 담고 있지는 않지만,
세션 ID 자체를 탈취당해 공격자가 클라이언트인 척 위장할 수 있다.

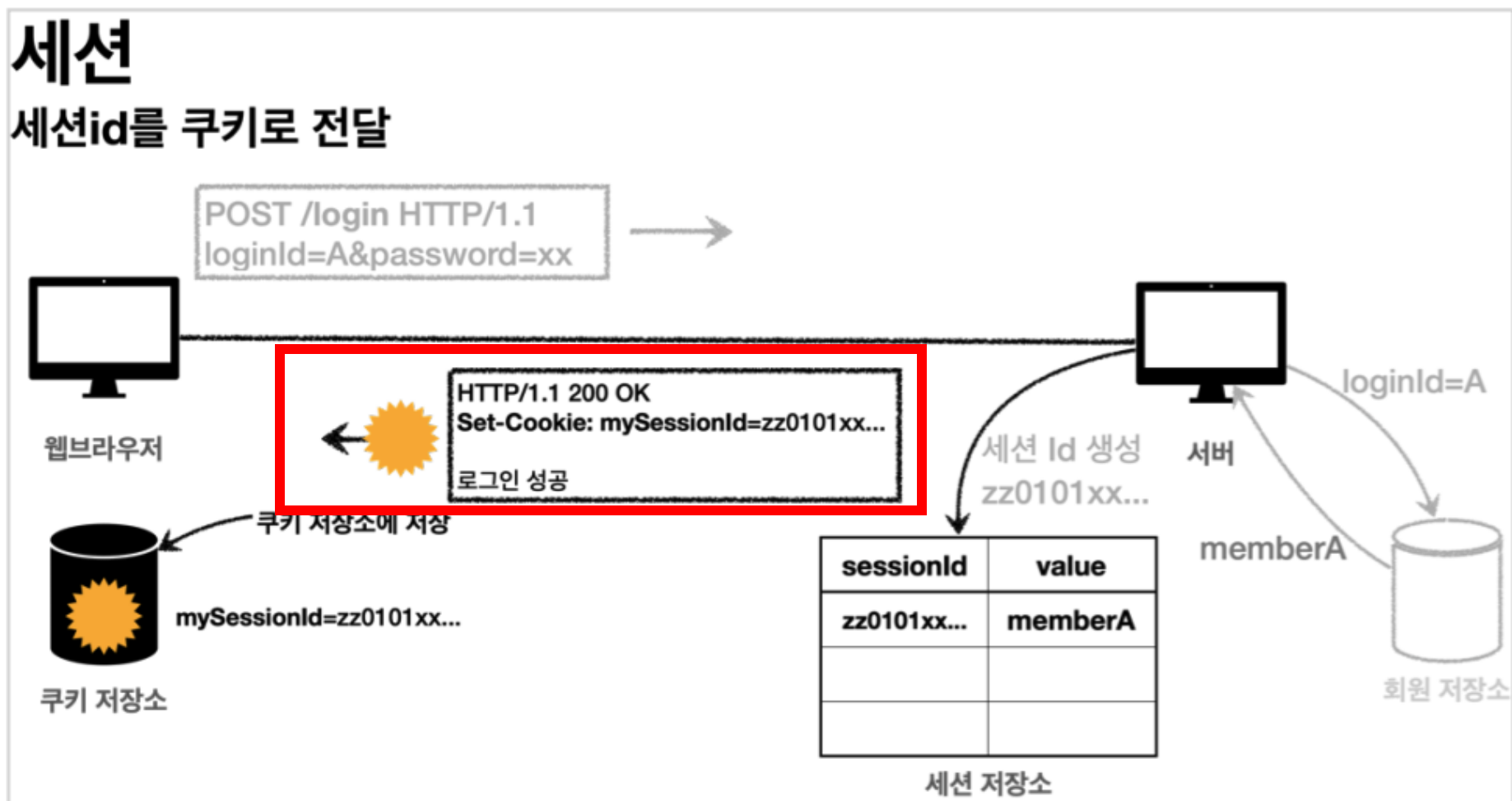
세션 하이재킹 (Session Hijacking)

공격자가 유효한 세션ID를 가로채는 것

- 쿠키 도난 : 브라우저의 쿠키를 훔쳐가는것
- 세션 고정 : 충분히 많은 세션 ID를 사용하지 않아 재사용 가능한 것
- 취약한 세션 ID 활용 : 추측하기 쉬운 세션ID

쿠키 도난 방지하기

쿠키를 생성할 때 키워드를 통해서 쿠키 도난 방지의 위험을 보완 할 수 있다.



XSS

1. 사용자의 브라우저에 Javascript를 주입해 사용자의 쿠키를 읽고 공격자의 웹 서버로 보내게 한다.
2. 수집한 쿠키 값을 스크립트에 추가해 사용자의 세션을 이용한다.

쿠키를 생성할 때 HttpOnly 키워드를 사용한다.

자바스크립트가 쿠키에 접근하지 못하게 한다.
(JS Document API의 접근을 제한)

```
Set-Cookie: session_id= 34217815; HttpOnly
```

중간자 공격

사용자와 서버가 요청과 응답을 사이에서 엿듣거나
중간에 끼어들어 데이터 전송을 가로채 쿠키를 훔친다.

HTTPS 암호화를 사용한다.

쿠키를 생성할 때 Secure 키워드를 사용한다.
HTTPS 프로토콜 상의 encrypted 요청만 쿠키가 전송된다.

```
Set-Cookie: session_id= 34217815; Secure
```


CSRF 공격

사용자가 특정 링크로 접속하게 되었을 때
HTTP 요청과 함께 세션 쿠키를 보내게 된다.

쿠키를 생성할 때 SameSite 키워드를 사용한다.

Same-Site=Strict: 모든 외부 사이트에
요청을 보낼 때 쿠키를 제거한다.

Set-Cookie: session_id= 34217815; **SameSite=Strict**

Same-Site=Lax: 모든 외부 사이트에 GET
요청을 보낼 때 빼고 쿠키를 제거한다.

Set-Cookie: session_id= 34217815; **SameSite=Lax**

요소

콘솔

소스

네트워크

성능

메모리

애플리케이션

보안

Lighthouse

녹음기

로컬 스토리지

https://www.naver.com

https://shopsquare.na

https://siape.veta.na

세션 저장소

IndexedDB

웹 SQL

쿠키

https://www.naver.com

https://shopsquare.na

https://siape.veta.na

필터

문제가 있는 쿠키만 표시

이름	값	Domain	Path	Expire...	크	HttpOnly	Secure	SameSite
NID...	9wHKEYfYdL+L2...	.naver.com	/	세션	71	✓	✓	Lax
ASID	7a22408b000001...	.naver.com	/	2024-...	36		✓	None
NID...	AAABoz416SKq3...	.naver.com	/	세션	5...		✓	Lax
NID...	itlB27NXiAsX5nln...	.naver.com	/	세션	51		✓	
NNB	KDPRMVR76J2V6	.naver.com	/	2025-...	16		✓	None
nid_...	KDPRMVR76J2V6	.nid.naver....	/	2025-...	20		✓	
pcvi...	pcview	.naver.com	/	세션	12			
NV_...	"MTEyMzcxMDE="	.naver.com	/	2024-...	36			
NV ...	"MTEyMzcxMDE="	.naver.com	/	2024-...	39			

실제 Naver.com 에 접속했을 때 나타나는 쿠키 속성

인증의 구현

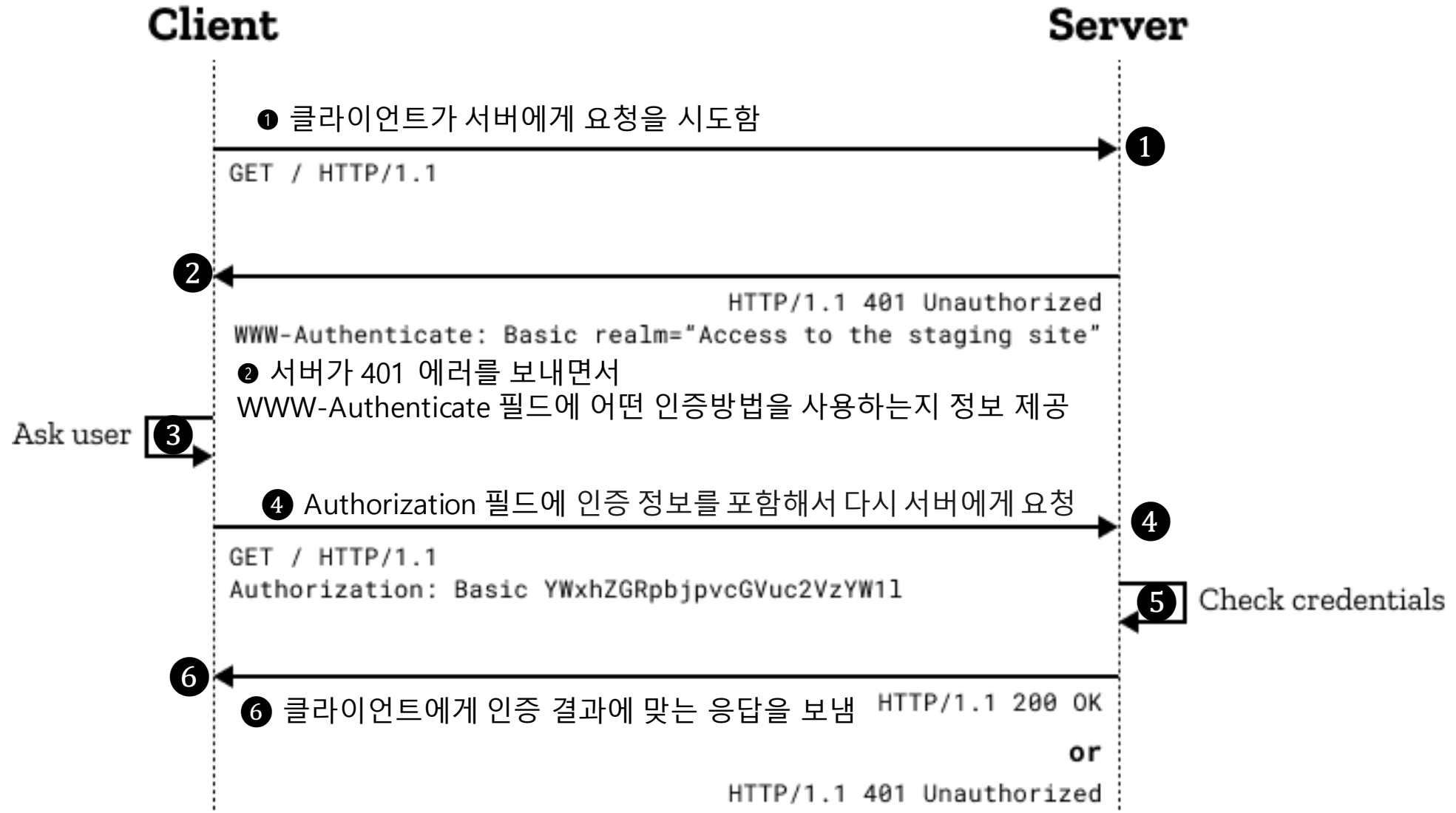
인증 상태 저장 방식

1. 쿠키방식
2. Session 방식

HTTP 인증 프레임워크

1. Basic 방식
2. Bearer 토큰 방식 (JWT, OAuth)

HTTP 인증 프레임워크



HTTP 헤더 – Authorization 필드

Authorization: <type> ...

Scheme	설명
Basic	사용자 아이디와 암호를 Base64로 인코딩한 값을 토큰으로 사용한다. (RFC 7617)
Bearer	JWT 혹은 OAuth2.0에 대한 토큰을 사용한다. (RFC 6750)
Digest	서버에서 난수 데이터 문자열을 클라이언트에 보낸다. 클라이언트는 사용자 정보와 nonce를 포함하는 해시값을 사용하여 응답한다 (RFC 7616)
HOBA	전자 서명 기반 인증 (RFC 7486)
Mutual	암호를 이용한 클라이언트-서버 상호 인증 (draft-ietf-httpauth-mutual)
AWS4-HMAC-SHA256Mutual	AWS 전자 서명 기반 인증

RFC(Request for Comments) :

IETF 에서 제공 관리하는 인터넷 개발에 있어 필요한 기술, 연구 결과, 절차 등을 기술해 놓은 메모



Internet Assigned Numbers Authority

[IANA \(인터넷할당번호관리기관\)](#)

Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry

Created

2014-02-17

Last Updated

2023-11-17

Available Formats



Registries included below

- [HTTP Authentication Schemes](#)
- [HTTP Mutual Authentication Algorithms](#)

HTTP Authentication Schemes

Registration Procedure(s)

[IETF Review](#)

Reference

[\[RFC9110, Section 16.4.1\]](#)

Available Formats



Authentication Scheme Name	Reference	Notes
Basic	[RFC7617]	
Bearer	[RFC6750]	
Digest	[RFC7616]	
DPoP	[RFC9449, Section 7.1]	
HOBA	[RFC7486, Section 3]	The HOBA scheme can be used with either HTTP servers or proxies. When used in response authentication header fields are used instead, as with any other HTTP authentication scheme.
Mutual	[RFC8120]	
Negotiate	[RFC4559, Section 3]	This authentication scheme violates both HTTP semantics (being connection-oriented) and header field syntax.
OAuth	[RFC5849, Section 3.5.1]	
PrivateToken	[RFC-ietf-privacypass-auth-scheme-15, Section 2]	
SCRAM-SHA-1	[RFC7804]	
SCRAM-SHA-256	[RFC7804]	
vapid	[RFC 8292, Section 3]	

3. HTTP Basic 방식

형식 : `Authorization: <type> <credentials>`

예 : `Authorization: Basic YWJjMTIzOnF3ZXJ0eQ==`

아이디와 비밀번호를 콜론으로 구분해 합친 문자열을 (abc123:qwerty)

Base64로 인코딩(YWJjMTIzOnF3ZXJ0eQ==)한다.

Base64 URL-safe Encode :
URL로 이용할 수 있는 문자열로 만들어
구분하도록 해주는 인코딩 방식



참고: Base64 인코딩은 암호화나 해싱을 의미하지 않습니다! 이 방법은 인증에 대해서 문자를 그대로 보내는 것과 동일하다고 할 수 있습니다 (base64인코딩은 복호화 가능). Basic 인증을 하는 경우 HTTPS로 접속하는 것을 권장합니다.

HTTP 헤더 – Authorization 필드

Authorization: <type> ...

Scheme	설명
Basic	사용자 아이디와 암호를 Base64로 인코딩한 값을 토큰으로 사용한다. (RFC 7617)
Bearer	JWT 혹은 OAuth2.0에 대한 토큰을 사용한다. (RFC 6750)
Digest	서버에서 난수 데이터 문자열을 클라이언트에 보낸다. 클라이언트는 사용자 정보와 nonce를 포함하는 해시값을 사용하여 응답한다 (RFC 7616)
HOBA	전자 서명 기반 인증 (RFC 7486)
Mutual	암호를 이용한 클라이언트-서버 상호 인증 (draft-ietf-httpauth-mutual)
AWS4-HMAC-SHA256Mutual	AWS 전자 서명 기반 인증

RFC(Request for Comments) :

IETF 에서 제공 관리하는 인터넷 개발에 있어 필요한 기술, 연구 결과, 절차 등을 기술해 놓은 메모



Internet Assigned Numbers Authority

[IANA \(인터넷할당번호관리기관\)](#)

Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry

Created

2014-02-17

Last Updated

2023-11-17

Available Formats



Registries included below

- [HTTP Authentication Schemes](#)
- [HTTP Mutual Authentication Algorithms](#)

HTTP Authentication Schemes

Registration Procedure(s)

IETF Review

Reference

[\[RFC9110, Section 16.4.1\]](#)

Available Formats



Authentication Scheme Name	Reference	Notes
Basic	[RFC7617]	
Bearer	[RFC6750]	
Digest	[RFC7616]	
DPoP	[RFC9449, Section 7.1]	
HOBA	[RFC7486, Section 3]	The HOBA scheme can be used with either HTTP servers or proxies. When used in response authentication header fields are used instead, as with any other HTTP authentication scheme.
Mutual	[RFC8120]	
Negotiate	[RFC4559, Section 3]	This authentication scheme violates both HTTP semantics (being connection-oriented) and header field syntax.
OAuth	[RFC5849, Section 3.5.1]	
PrivateToken	[RFC-ietf-privacypass-auth-scheme-15, Section 2]	
SCRAM-SHA-1	[RFC7804]	
SCRAM-SHA-256	[RFC7804]	
vapid	[RFC 8292, Section 3]	

Bearer 방식

“Bearer”은 소유자라는 뜻으로, “이 토큰의 소유자에게 권한을 부여한다”라는 의미로 이름을 붙임

토큰은 일종의 출입 카드 라고 볼 수 있음.

토큰의 장점:

- 한 번의 인증으로 받은 토큰을 사용해서 개인정보의 노출을 줄여 공격자가 가로챌 기회가 적다.
- 인증 정보에 대한 별도의 저장소가 필요없다 (쿠키/세션에 사용자 정보 저장 안함).
- 사용자 권한과 같은 세부 정보를 토큰에 저장할 수도 있다.
- 다른 플랫폼에서 가지고 있는 계정으로도 인증을 할 수 있다. (OAuth)

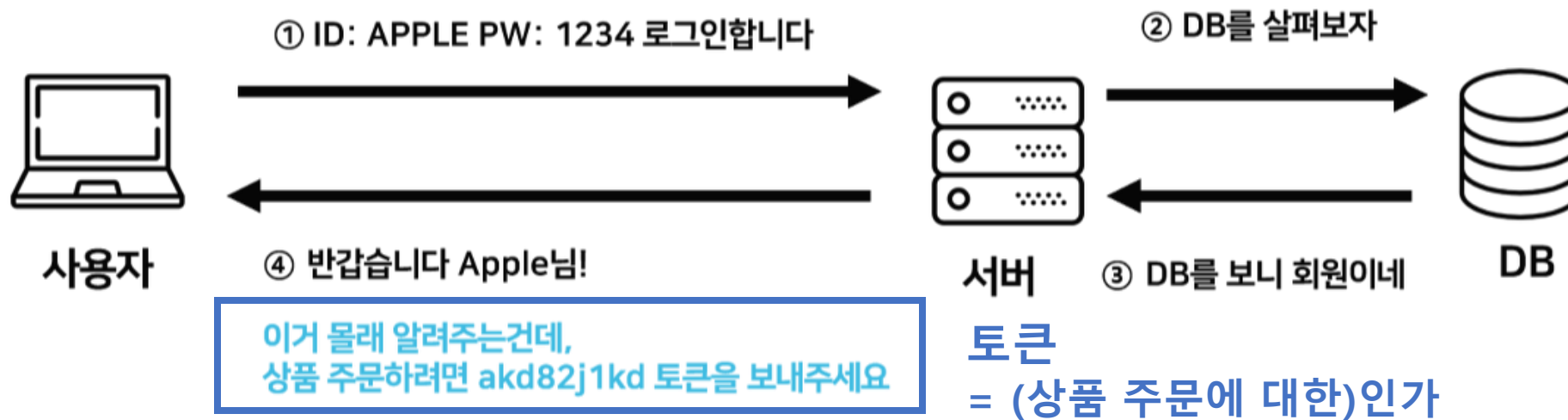
토큰의 수명:

공격자는 토큰을 탈취하더라도 토큰으로 할 수 있는 일이 무엇인지 찾아야 하기 때문에 그 사이 시스템은 토큰이 탈취당한 것을 알고 탈취된 토큰을 무효화해버리는 등 대처가 가능함.

4. JWT 인증

JWT (JSON Web Token)

인증에 필요한 정보들을 암호화시킨 JSON 토큰으로 클라이언트를 식별



이후 사용자가 JWT를 이용해 상품을 주문할 때 HTTP 헤더:

Authorization: Bearer akd82j1kd

JWT (JSON Web Token)

header, payload, signature 세가지를 (.) 으로 구분하는 Base64로 인코딩된 문자열

구성



실제
Encoded

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MzkwMjQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

Decoded

Header	Payload	Signature
<pre>{ "alg": "HS256", "typ": "JWT" }</pre>	<pre>{ "sub": "1234567890", "name": "John Doe", "iat": 1516239022 }</pre>	<pre>HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret)</pre>

JWT Header

토큰의 타입과 해시 암호화알고리즘 구성.

속성	설명
typ	토큰의 유형(JWT)
alg	HMAC, SHA256 or RSA와 같은 해싱 알고리즘

Decoded

Header	Payload	Signature
<pre>{ "alg": "HS256", "typ": "JWT" }</pre>	<pre>{ "sub": "1234567890", "name": "John Doe", "iat": 1516239022 }</pre>	<pre>HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret)</pre>

JWT Payload

토큰에 담을 name/value(클레임)으로 이루어진 정보들

속성	설명
iss	토큰의 발급자
sub	토큰 제목
aud	토큰 대상자
exp	토큰 만료 시간
nbf	토큰 활성 날짜
iat	토큰 발급 시간
jti	JWT 토큰 식별자

Decoded

Header	Payload	Signature
<pre>{ "alg": "HS256", "typ": "JWT" }</pre>	<pre>{ "sub": "1234567890", "name": "John Doe", "iat": 1516239022 }</pre>	<pre>HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret)</pre>

JWT Signature

헤더에서 정의한 해싱 알고리즘(alg)으로
(헤더 + 페이로드 + 서버가 갖고 있는 유일한 key값)을 암호화 한다.
서버가 비밀키로 새로 signature를 만들어 비교해보면 토큰이 변조되었는지 확인 할 수 있다.

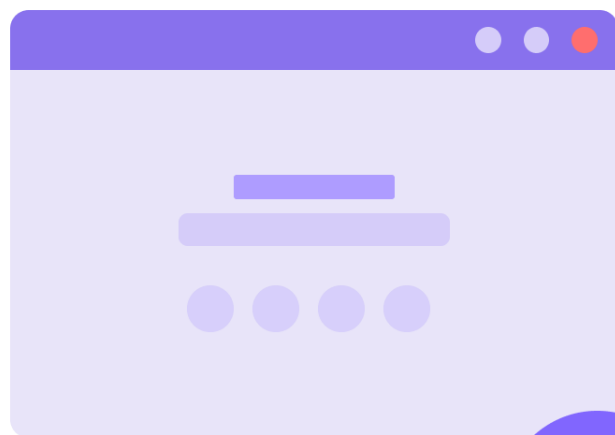
Signature = Base64Url(Header) + . + Base64Url(PayLoad) + server's key

HS256

Decoded

Header	Payload	Signature
<pre>{ "alg": "HS256", "typ": "JWT" }</pre>	<pre>{ "sub": "1234567890", "name": "John Doe", "iat": 1516239022 }</pre>	<pre>HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret)</pre>

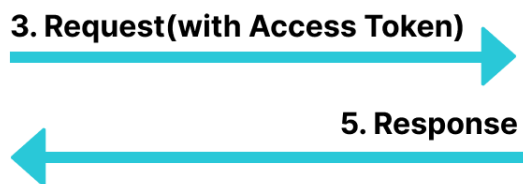
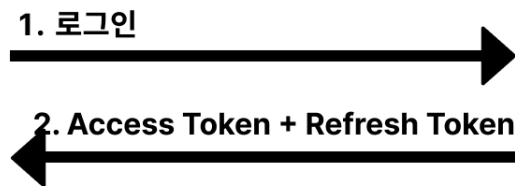
4. JWT 인증



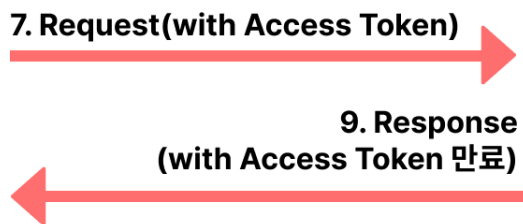
6.
Access
Token 만료

Refresh Token : Access Token보다 긴 유효기간을 가지고 Access Token이 만료됐을 때 새로 발급해주는 열쇠이다.

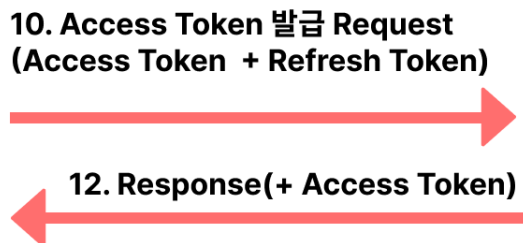
매번 요청으로 노출되는 Access Token과 달리, Refresh Token은 Access Token이 만료됐을 경우에만 서버로 보내기 때문에 탈취될 위험이 적다는 보안의 이점이 있다.



4.
Access
Token 검증



8.
Access Token
만료 확인



11.
Refresh Token 확인
+
Access Token 발급

5. OAuth 인증 방식

인터넷 사용자들이 비밀번호를 제공하지 않고
다른 웹사이트상의 자신들의 정보에 대해
웹사이트나 애플리케이션에게 접근 권한을 부여할 수 있는
공통적인 수단으로서 사용되는, 접근 위임을 위한 개방형 표준



OAuth 1.0 (2010) -> OAuth 2.0 (2012) -> OAuth2.1(초안 2020~)

소셜 로그인 :

이메일

이메일로 시작하기

or

페이스북으로 시작하기

Apple로 시작하기

Google로 시작하기

Cellonine

간편하게 로그인하고
다양한 서비스를 이용해보세요.

카카오톡으로 시작하기

다른 이메일로 시작하기

Join

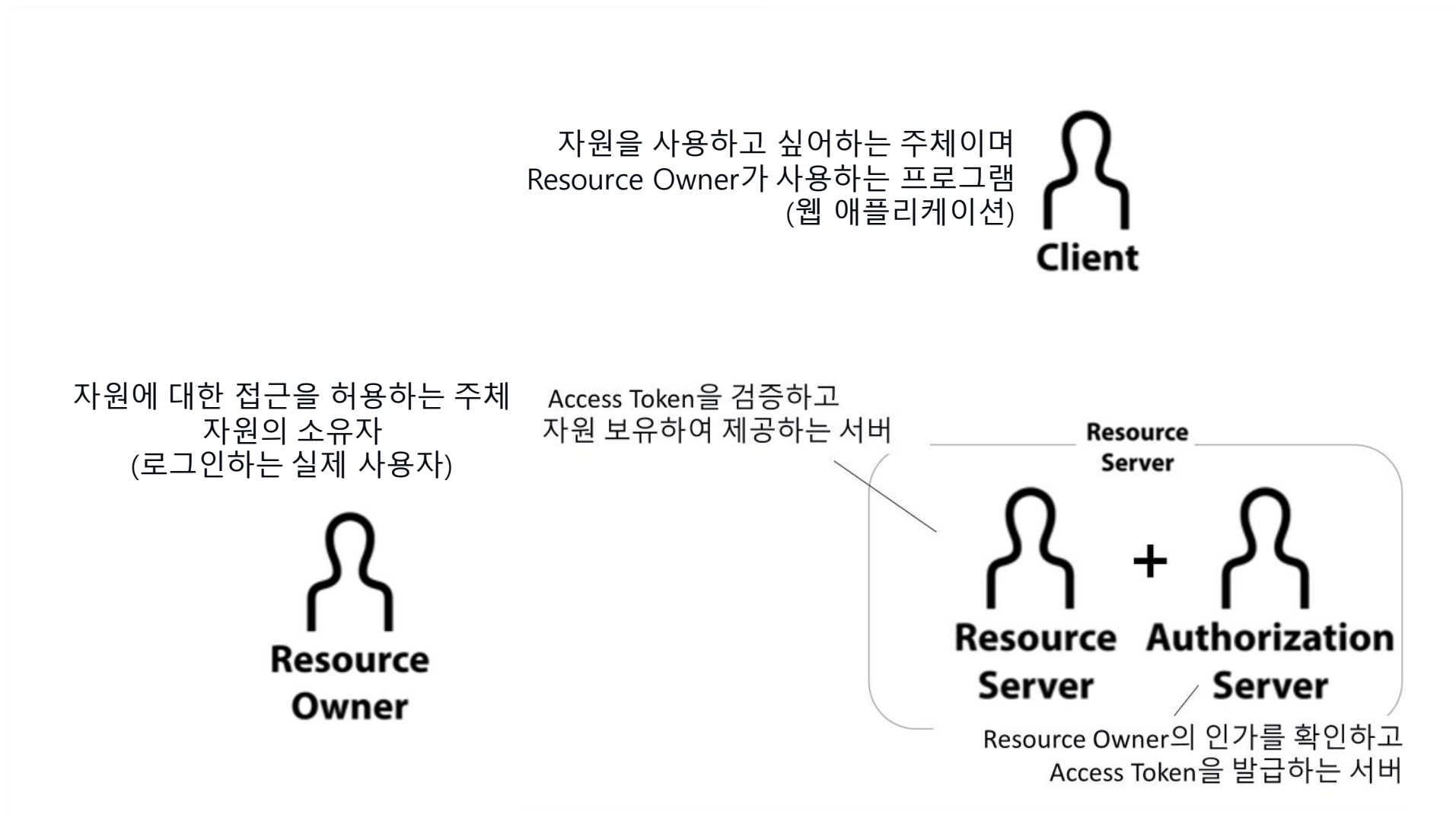
기존에 사용하시는 계정으로
간단하게 회원가입 하세요.

네이버 아이디로 가입

페이스북 아이디로 가입

구글 아이디로 가입

OAuth2.0 참여자



OAuth2.0 참여자

자원을 사용하고 싶어하는 주체이며
Resource Owner가 사용하는 프로그램
(웹 애플리케이션)



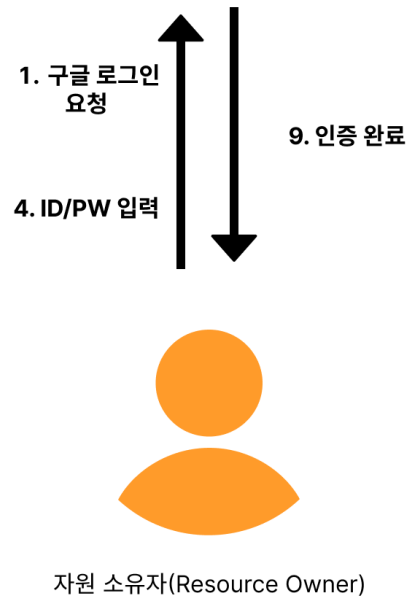
자원에 대한 접근을 허용하는 주체
자원의 소유자
(로그인하는 실제 사용자)



**Resource
Server**

자원을 보유하여
Access Token을 검증하고
Protected Resource를 제공하고
토큰을 발행해 주는 서버

① Resource Owner 인증 및 인가



2. 로그인 페이지 요청(with Client ID, Redirect URI)

3. 로그인 페이지 제공

5. Request(iD, PW)

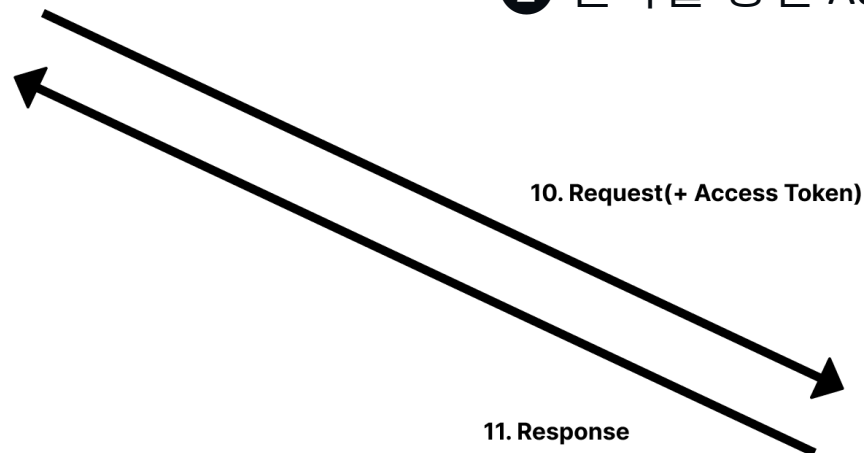
6. Authorization Code 발급

7. Authorization code로 Access Token 요청

8. Access Token 발급

인증 서버(Authorization Server)

② 인가를 통한 Access Token 획득



③ Access Token으로 Protected Resource 접근



Access Token: 자원 서버에 자원을 요청할 수 있는 토큰
Refresh Token : 권한 서버에 접근 토큰을 요청할 수 있는 토큰

Bearer 인증 방식의 한계

토큰은 인코딩된 문자열일 뿐이기 때문에,

공격자가 AccessToken과 RefreshToken을 탈취해 사용할 수 있음.

외부 공격자가 RefreshToken을 탈취한 상황이라면 얼마든지 새로운 AccessToken을 생성할 수 있음

결국 탈취를 막는 근본적인 해결은 없고 공격 기회를 줄이는 등의 보완만 가능하다고 함.

만약
아이디, 비밀번호
(자격증명)을
탈취 당하면?



TOP2: 계정 탈취

NAVER

PC방 등 공용PC라면 QR코드 로그인에 더 안전해요 x

ID 로그인 | [i] 일회용 번호 | QR코드

id12345

.....

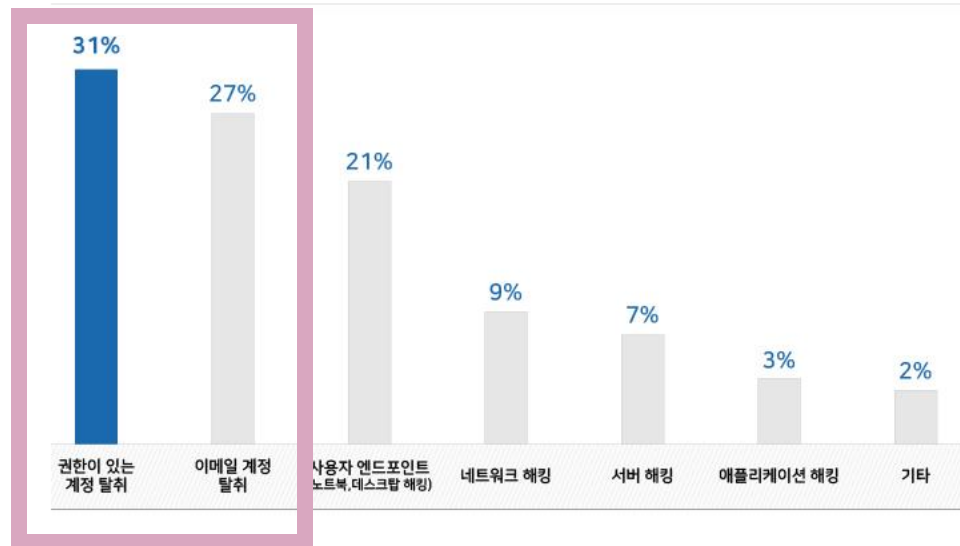
☒ 로그인 상태 유지 ☐ IP보안

로그인

비밀번호 찾기 | 아이디 찾기 | 회원가입

기업의 민감한 데이터로 접근하는 가장 쉽고 빠른 방법

(Which entry point gives you the easiest/fastest access to sensitive data?)

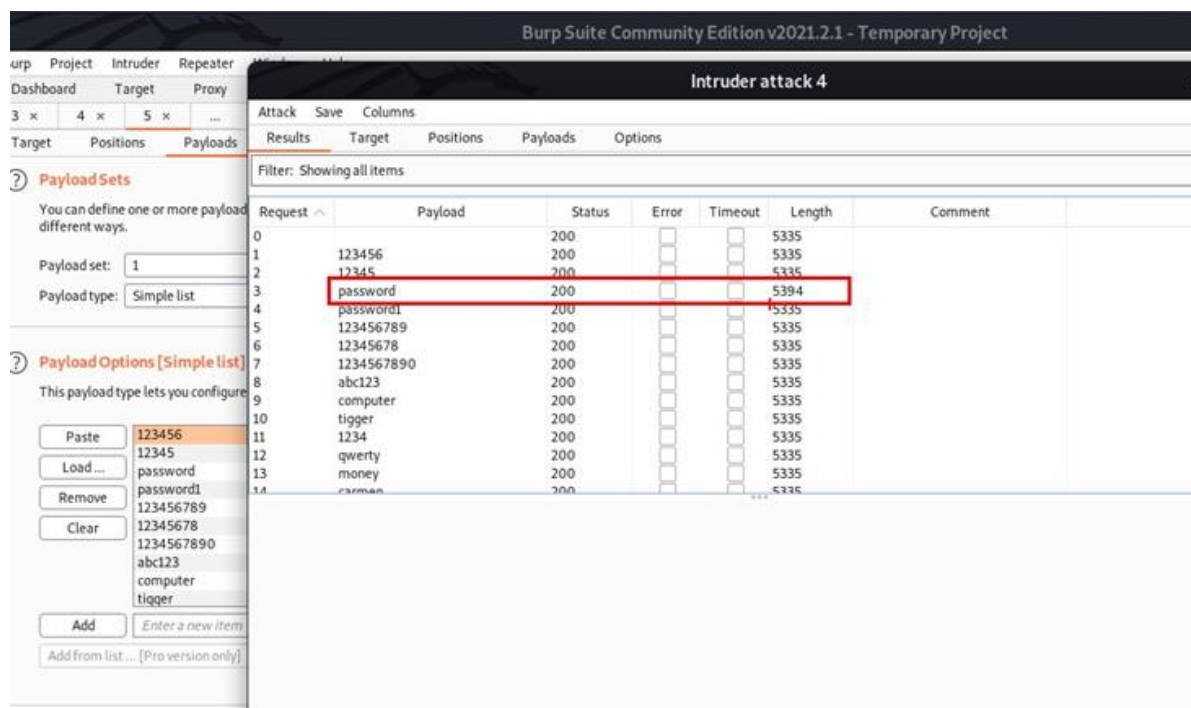


시스템은
정상 사용자와 구분하지 못하고
악의적인 행동을 막을 수 없다.

전수조사 공격(Brute Force Attack)

한국어

NAVER



응답의 길이가 달라지면 로그인이 성공했다고 판단

PC방 등 공용PC라면 QR코드 로그인 더 안전해요. x

ID 로그인 | 일회용 번호 | QR코드

you yourid yourid yourid
123456 12345 yourid
password password

로그인 상태 유지

로그인

비밀번호 찾기 | 아이디 찾기 | 회원가입



모호한 로깅 처리 하기

한국어 ▼

NAVER

PC방 등 공용PC라면 QR코드 로그인 더 안전해요 x

ID 로그인

일회용 번호

QR코드


yourid

Notpassword

아이디(로그인 전용 아이디) 또는 비밀번호를 잘못 입력했습니다.
입력하신 내용을 다시 확인해주세요.

로그인

[비밀번호 찾기](#) | [아이디 찾기](#) | [회원가입](#)



NAVER
안전하게 내 정보 지키는 법
실명인증하고 보안 강화

만약 로그인에 실패했을 때
가장 안전한 로그는?

1. 올바른 비밀번호를 입력해주세요.
2. 그런 아이디는 존재하지 않습니다.

정답 :
어라? 왜 안 되는 건지 저도 모르겠어요.

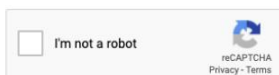
자동입력 방지하기

CAPTCHA

'Completely Automated Public Turing test to tell Computers and Humans Apart' 의 약어로
컴퓨터와 사람을 구분하기 위한 완전히 자동화된
튜링테스트를 의미

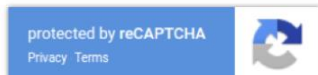
reCAPTCHA v2

("I'm not a robot" **Checkbox**)



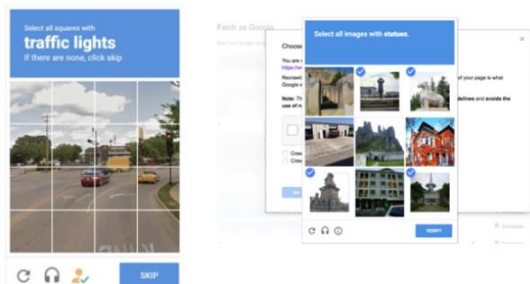
reCAPTCHA v2

(Invisible reCAPTCHA badge)



2b827

이미지 reCAPTCHA



일정 횟수가 초과하면 자동입력 방지 로직이 실행된다.

PC방 등 공용PC라엔 QR코드 로그인도 더 안전해요

ID 로그인일회용 번호QR코드

아이디(로그인 전용 아이디), 비밀번호 또는
자동입력 방지 문자를 잘못 입력했습니다.
입력하신 내용을 다시 확인해주세요.

비밀번호

세션서명구분일부정보
304-0275
제품명
생닭 (중량 1000g)
파우치형 요구르트
광양 매실
잡곡 녹두

가 격 개 수 총 합
300 2 600
800 1 800
300 6 1800
700 4 2800

해당 영수증은 가상으로 제작된 것으로 실제 영수증 사진이 아닙니다.
가장 많이 구매한 물건의 이름은 무엇입니까?

정답을 입력해주세요

☒ 로그인 상태 유지

IP보안 ☒

로그인

주요 웹 취약점

인증 취약성

사용자가 인증 없이
다른 사람의 기능이나 데이터에 접근할 수 있다.

Injection

시스템의 원래 목표와 다르게 데이터 변경과
데이터의 탐색이 가능하도록 스크립트를 주입시킬 수 있다.

SQL Injection,
Xpath Injection,
OS Command Injection,
LDAP Injection

XSS

페이지에 스크립트를 주입해 해당 페이지에
방문한 사람 모두가 스크립트를 실행 시키게 할 수 있다.

메서드 접근제어 부족

메서드 A의 접근을 막아도 접근을 막지 않은 다른
메서드를 통해 같은 기능을 실행 할 수 있다.

세션 고정

웹 애플리케이션이 인증에 고유한 세션ID를 할당하지 않아
기존 세션 ID가 재사용될 가능성이 있다.

중간자 공격

클라이언트와 서버 사이에 요청과 응답을 엿보거나 탈취할 수 있다.

CSRF

인증된 사용자가 실행 시켰다고 판단되도록 위조된 코드가 있는
페이지에 사용자가 방문하면 코드를 실행시키게 할 수 있다.

기밀 데이터 노출

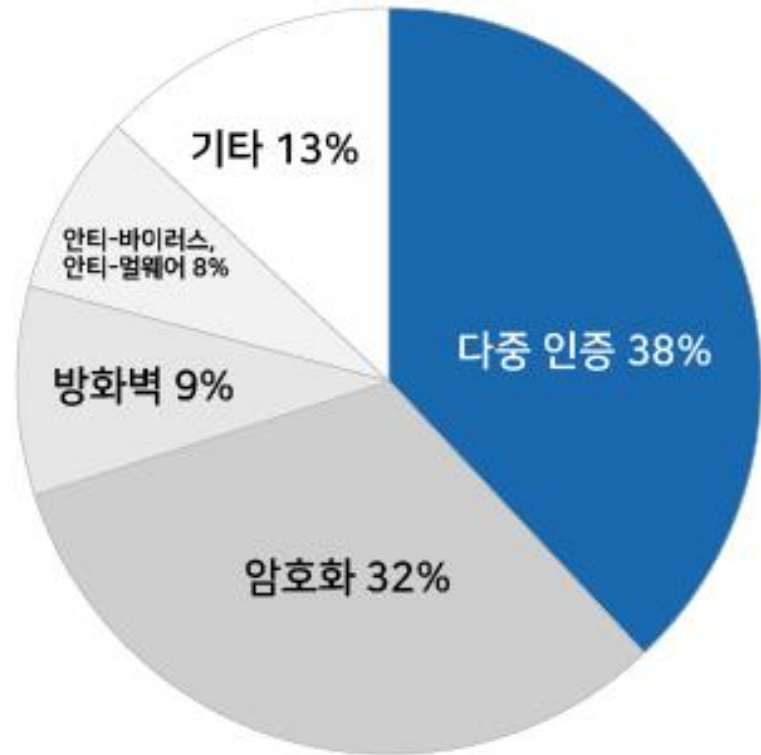
DB 계정 비밀번호, 요청에 사용되는 키와 토큰,
사용자의 개인정보 등이 노출되어 있다.

취약성이 있는 종속성 사용

라이브러리 등 직접 작성하지 않았지만
종속성이 있는 코드에 취약점이 있다.

가장 해킹이 힘든 보안 요소

(Which type of security is the hardest to get past?)



1위 : 다중 인증 (38%)

2위 : 암호화 (32%)

3위 : 방화벽 (9%)

+ 다중 인증 (Multi Factor Authentication)

최소 두 가지 이상의 인증을 거친 사용자에게만 접근을 허용하는 보안 기법.
아이디와 비밀번호를 입력하는 1차 인증 외에 지문 인식 등 추가적인 본인 인증을 요구한다.

지식 / 소유 / 속성 / 행위 / 장소 등의 인증 종류가 있다.



참고문서

 [프론트에서 안전하게 로그인 처리하기 \(ft. React\)](#)

 [JWT 토큰 인증 이란? \(쿠키 vs 세션 vs 토큰\)](#)

[\[Gitbook\] doden- 쿠키](#)

[\[Gitbook\] doden- 세션](#)

[RFC의-역사-RFC-종류-RFC-표준화-절차](#)

[캡차\(CAPTCHA\)를 알아보자 \(로봇이 아닙니다! 🤖\)](#)

[토큰 기반 인증에서 bearer는 무엇일까?](#)

[\[IT 이슈\] 진짜 해커들이 말하는 기업의 데이터에 접근하는 방법](#)

[Basic 인증과 Bearer 인증의 모든 것](#)

['다중 인증\(MFA\)'이란? 비밀번호 믿지 마세요!](#)

[\[Web\] 로그인 인증 방식](#)

[Spring 기반 OAuth 2.1 Authorization Server 개발 짭먹해보기](#)

[로그인에 사용하는 OAuth: 과거, 현재 그리고 미래](#)

[자바 웹 개발 워크북 - 구멍가게 코딩단 / 프리렉](#)

[웹 개발자를 위한 웹 보안 - 말콤 맥도널드 / No Starch Press](#)

[스프링 시큐리티 인 액션 - 로렌티우 스피카 / 위키북스](#)