M04-03 · GIBALJA VISUAL MANUAL

로그인·인증·권한 구분하기

> **한 문장 목표:** 사용자가 로그인했다는 사실만 보지 않고, 무엇으로 신원을 증명했는지, 그 결과를 어떻게 이어 쓰는지, 지금 이 행동을 이 객체에 허용할지, 언제 만료·취소·재인증할지를 하나의 정책 계약으로 작성합니다.

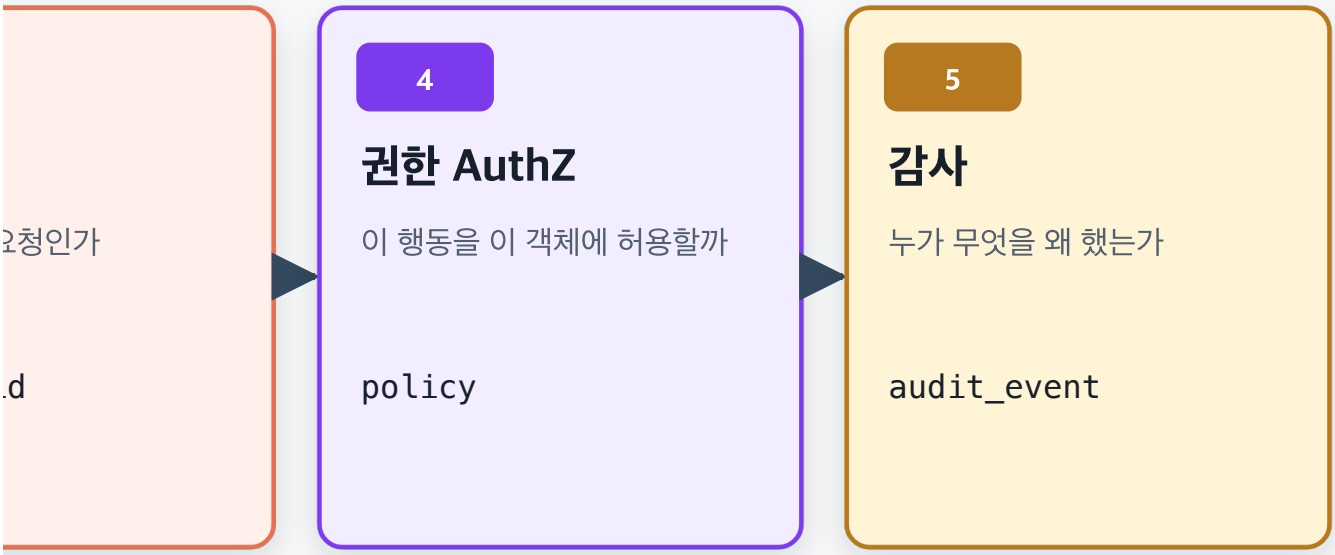
난이도	개념	실습	셀프 테스트	최종 산출물
Level 1	60분	60분	15분	인증 흐름도, 권한 매트릭스, 수명주기·실패 시나리오

"회원만 볼 수 있습니다"는 화면 문구입니다. 계약은 "유효한 session에서 `subject=usr_17`을 만들고, `GET /orders/{orderId}`마다 `orders:read` scope와 `order.ownerId == subject.id`를 server에서 판정합니다. 인증 정보가 없거나 만료되면 `401`, 인증됐지만 이 객체가 아니면 노출 정책에 따라 `403` 또는 `404`, 관리자 삭제는 `AAL2`이면서 인증 후 5분 이내여야 합니다"까지 닫아야 합니다.

그림 1. 로그인 화면 뒤에서 인증·세션·권한·감사가 차례로 이어집니다. 각 층의 입력·결과·만료·실패를 분리해야 설계가 검수 가능합니다.

같지 않은 것	로그인 화면 ≠ 인증 결과 ≠ 세션 secret ≠
완료 기준	각 층의 입력·결과·만료·오류·기록이 명시됨

좋은 기획은 비밀번호 칸보다 로그인 이후



권한 정책

모든 요청의 신뢰와 허용 경계를 설계한다

1. 이 PDF를 공부하는 방법

1회차 · 다섯 층 구분 · 20분

`로그인 → 인증(AuthN) → 세션 → 권한(AuthZ) → 감사`를 소리 내어 설명합니다. “로그인됐으니 허용”이라는 문장이 나오면 어느 층이 생략됐는지 표시합니다.

2회차 · 운반 수단 비교 · 25분

server session cookie와 bearer access token을 비교하고, ID token·access token·refresh token의 소비자를 연결합니다. `cookie`, `JWT`, `session`, `OAuth`를 같은 축의 대안처럼 말하지 않는 것이 핵심입니다.

3회차 · 아홉 상황 실행 · 35분

인증·권한 정책 스튜디오에서 인증 없음·만료·취소·scope 부족·소유권 불일치·한도 초과·step-up·공개 자원을 판정합니다.

4회차 · 권한 계약 작성 · 40분

인증·권한 정책 템플릿에 실제 기능 하나를 옮깁니다. 허용 칸만 쓰지 말고 거부·만료·취소·재인증·감사 증거도 씁니다.

5회차 · 셀프 테스트 · 15분

정답을 가리고 12문제를 풉니다. 틀린 문제는 `증명`, `연속성`, `소비자`, `정책 입력`, `실패`, `수명` 중 하나로 분류합니다.

학습 완료 기준

“관리자만 가능합니다”를 “`DELETE /articles/{id}`는 `role=admin`, `articles:delete` scope, 같은 tenant, `AAL2`, `auth_time` 5분 이내를 모두 만족해야 하며, 아니면 기본 거부합니다. 약한·오래된 인증이면 OAuth 환경에서는 `401` step-up challenge, role·scope 부족이면 `403`, 모든 판정은 policy ID와 request ID를 남깁니다”처럼 설명할 수 있습니다.

2. 먼저 비슷해 보이는 말을 분리합니다

말	이 교재의 뜻	대표 질문	결과
식별 identification	주체가 자신을 어떤 ID라고 주장	누구라고 말하는가	username·subject 후보
신원 확인 identity proofing	계정이 실제 사람·조직과 어떻게 연결되는지 확인	이 계정을 누구에게 발급했는가	검증 수준·가입 증거
로그인 login	인증을 시작하고 완료하는 사용자 흐름	어디서 어떤 수단으로 들어오는가	성공·실패·복구 UX
인증 authentication, AuthN	주체가 등록된 authenticator를 통제함을 검증	주장이 유효한 증거로 뒷받침되는가	인증 사건·assurance
세션 session	인증된 상호작용의 연속성	다음 요청도 같은 주체인가	session secret·상태
권한 판정 authorization, AuthZ	특정 요청을 허용·거부	이 주체가 이 행동을 이 객체에 해도 되는가	allow·deny·challenge
접근 제어 access control	권한 정책을 실제로 집행하는 체계	어디서 모든 요청을 막거나 통과시키는가	정책 집행·테스트
연합 federation	외부 Identity Provider(IdP)의 assertion에 의존	어느 발급자를 신뢰하고 어떻게 검증하는가	RP session·claims

현장에서는 authorization을 “인가”라고도 합니다. 이 교재는 인허가 업무와 혼동을 줄이기 위해 **권한 판정**이라는 말을 주로 쓰고 `AuthZ`를 함께 표기합니다.

2.1. 이메일 확인이 항상 인증은 아닙니다

가입 중 이메일로 보낸 코드를 입력하는 행위는 이메일 주소 통제 확인일 수 있습니다. 그것이 계정의 장기 로그인 authenticator인지, 연락처 검증인지, 복구 수단인지 목적을 구분합니다.

2.2. 생체 정보는 보통 기기 안의 authenticator를 활성화합니다

서비스가 지문 원본을 받는다고 가정하지 않습니다. passkey·WebAuthn 흐름에서는 기기가 사용자 확인 후 공개키 자격 증명으로 서명하고, relying party가 그 결과를 검증할 수 있습니다.

2.3. 인증 성공은 무제한 권한이 아닙니다

인증은 `subject`와 인증 맥락을 만듭니다. 권한 판정은 그 값을 입력으로 사용하지만 role·scope·소유권·tenant·객체 상태·시간·위험 조건을 더 봅니다.

3. 로그인 뒤의 다섯 층을 계약으로 만듭니다

층	입력	핵심 처리	출력	대표 실패
로그인	식별자·인증 수단 선택	흐름·복구·오류 안내	인증 요청	잠금·사용자 취소
인증	authenticator output	서명·secret·origin·rate limit 검증	subject·AAL·auth_time	증명 실패
세션	인증 사건	session secret 발급·회전·만료	연속 요청의 주체	만료·취소·탈취
권한	subject·action·resource·context	policy 평가	allow·deny·step-up	role·scope·객체 조건 실패
감사	요청·판정·정책	최소 필요한 증거 기록	audit event·alert	기록 누락·민감정보 과다

다섯 층은 특정 framework의 코드 계층 표준이 아니라 **YEONCORE 학습용 책임 지도**입니다. 실제 시스템은 IdP·gateway·backend·policy engine·감사 시스템으로 나뉠 수 있습니다.

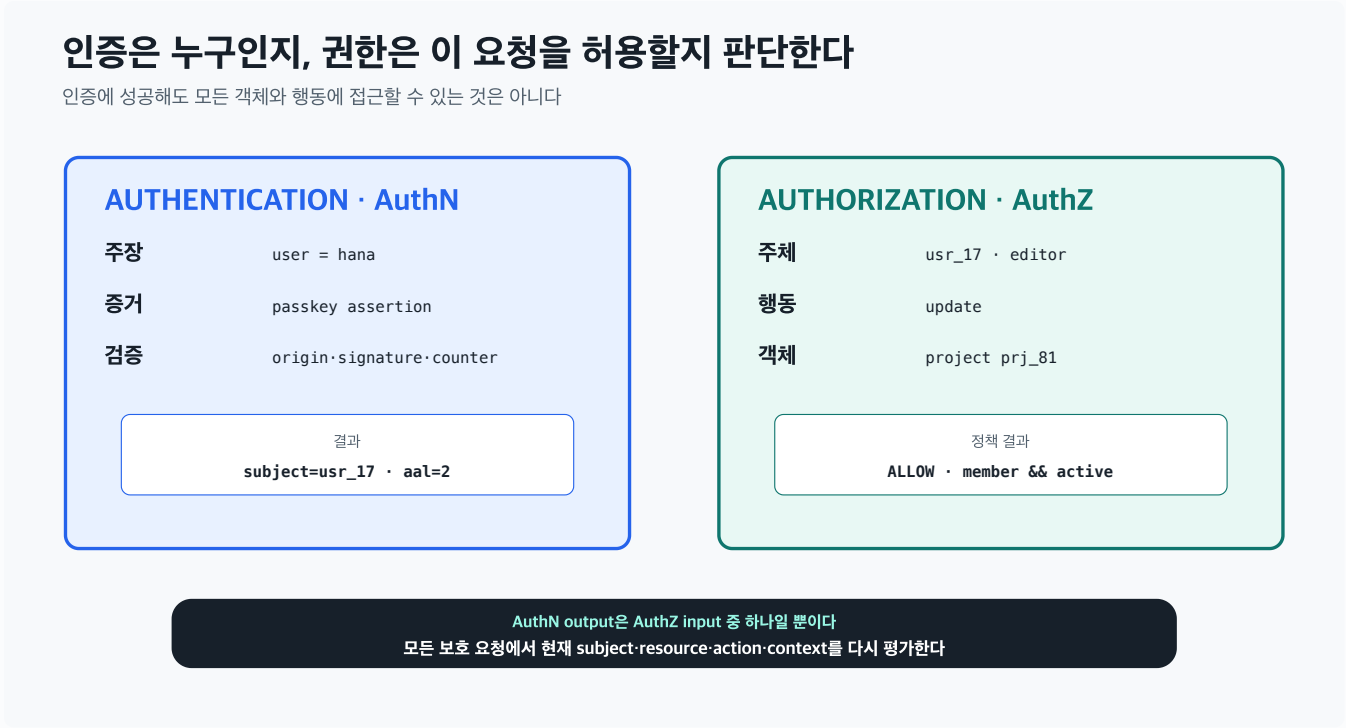


그림 2. 인증은 주장의 증거를 검증해 subject 맥락을 만들고, 권한은 그 subject가 특정 action을 특정 resource에 수행할 수 있는지 정책으로 판단합니다.

AUTHENTICATION · AuthN

주장	`user = hana`
증거	`passkey assertion`
검증	`origin·signature·counter`

결과

`subject=usr_17 · aal=2`

AuthN output은 AuthZ
모든 보호 요청에서 현재 subject·reso

AUTHORIZATION · AuthZ

주체	`usr_17 · editor`
행동	`update`
객체	`project prj_81`

정책 결과

`ALLOW · member && active`

Z input 중 하나일 뿐이다
urce·action·context를 다시 평가한다

30초 확인

```
AuthN 질문: 이 요청의 주체를 신뢰할 만하게 식별했는가?
AuthZ 질문: 그 주체가 지금 이 action을 이 resource에 해도 되는가?
```

4. 로그인 흐름은 성공 화면보다 실패·복구 경로가 더 중요합니다

로그인 화면 명세에는 input field만 있으면 부족합니다.

항목	설계 질문
identifier	email·username·phone 중 무엇이며 정규화 규칙은 무엇인가
authenticator	password·passkey·OTP·외부 IdP 중 무엇을 지원하는가
enumeration	존재하는 계정과 없는 계정이 응답·시간으로 구분되지 않는가
rate limit	계정·IP·device·위험 신호를 어떻게 제한하고 해제하는가
failure	잘못된 증명·잠김·비활성·IdP 장애를 사용자에게 어떻게 안전하게 알리는가
recovery	분실·기기 변경·계정 복구가 본 인증보다 약해지지 않는가
consent	외부 로그인에서 어떤 claim·scope를 왜 요청하는가
redirect	완료 후 허용된 목적지로만 돌아가는가
audit	성공·실패·복구·authenticator 변경을 어떻게 기록·통지하는가

4.1. 로그인 실패 문구와 내부 원인은 분리합니다

```
사용자 화면: 로그인 정보를 확인해 주세요.
내부 원인: account_not_found | invalid_password | locked | disabled
```

내부 원인을 그대로 노출하면 계정 존재 여부를 추측하게 만들 수 있습니다. 반대로 모든 상황을 같은 내부 코드로 뭉치면 운영·탐지·고객 지원이 어려워집니다. **외부 메시지와 내부 진단 코드를 별도로 설계**합니다.

4.2. password 정책은 길이 한 줄로 끝나지 않습니다

비밀번호 허용 길이, 유출 비밀번호 차단, rate limiting, password manager·붙여넣기 허용, 안전한 hashing, 변경·복구·알림을 함께 봅니다. 임의의 주기적 변경을 무조건 요구하는 방식보다 compromise 징후와 위험 사건에 대응하는 정책이 필요합니다.

5. 인증 강도는 factor 수와 프로토콜을 함께 봅니다

그림 3. 지식·소유·생체는 factor 분류입니다. 독립성·피싱 저항·사용자 의도·복구 경로가 실제 assurance를 좌우합니다.

KNOW

아는 것

password·PIN

HAVE

가진 것

device·security key

MFA 판정 질문 독립 factor인가 · 같은 채널이 함께 탈취되지 않는기

복구 판정 질문 복구가 본 인증보다 약하지 않은가 · 분실 시 즉시 inv

passkey는 공개키 자격 증명을 RP에 묶고 사용자 확인을

ARE

신체 특성

biometric activation

CRYPTO

피싱 저항

passkey·WebAuthn

ㅏ · 사용자 의도가 확인되는가

validate 가능한가 · 이벤트가 기록되는가

결합할 수 있지만 복구·기기 변경 설계가 여전히 필요하다

5.1. 세 가지 factor 분류

factor	뜻	예	주의
knowledge	사용자가 아는 것	password·PIN	탈취·재사용·phishing
possession	사용자가 가진 것	security key·등록 기기	분실·복제·channel 공격
inherence	사용자 신체 특성	fingerprint·face	기기 내 activation과 원격 검증 구분

두 개의 비밀번호는 knowledge factor 두 번이지 독립적인 다중 factor 인증(MFA)이라고 보기 어렵습니다. 같은 기기·같은 channel에 함께 의존하는 수단도 위협 모델에서 독립성을 검토합니다.

5.2. NIST의 Authentication Assurance Level

NIST SP 800-63B-4는 Authentication Assurance Level(AAL)을 사용해 인증 사건의 assurance를 표현합니다. 이 숫자를 제품 badge처럼 붙이기 전에 해당 system이 실제 요구사항과 authenticator 관리·session 수명·재인증 조건을 만족하는지 검토해야 합니다.

```
현재 session AAL은 최초 인증 사건보다 높을 수 없습니다.
더 높은 assurance가 필요하면 step-up authentication을 수행합니다.
```

5.3. passkey를 “비밀번호가 없는 버튼”으로만 설명하지 않습니다

passkey는 WebAuthn 공개키 자격 증명으로 구현될 수 있습니다. relying party 범위와 origin 검증, 사용자 존재·확인, credential 등록·동기화·분실·삭제·복구·기기 이전을 함께 설계합니다.

5.4. 복구는 우회 로그인입니다

복구 경로가 약하면 강한 MFA도 우회됩니다. 복구 신청, 대기 시간, 기존 session 취소, authenticator 재등록, 사용자 통지, 지원 담당자 권한, 감사 증거를 시나리오로 작성합니다.

6. 세션은 인증 사건의 연속성을 이어 줍니다

HTTP 요청은 기본적으로 서로 독립적입니다. 로그인할 때마다 password·passkey를 다시 내지 않도록 서비스는 인증 사건 뒤에 session을 만들 수 있습니다.

```
인증 사건 → session secret 발급 → client가 다음 요청에 secret 제시
→ server가 session 상태 확인 → subject 맥락 복원 → 권한 판정
```

NIST SP 800-63B-4는 session host와 subscriber software 사이에 session secret을 공유하거나 그 소유를 암호학적으로 증명해 연속성을 묶는다고 설명합니다.

6.1. session cookie 예

```
Set-Cookie: __Host-session=s_opaque_7f...; Path=/; Secure; HttpOnly; SameSite=Lax
```

속성	줄이는 위험	한계
`Secure`	HTTPS가 아닌 전송에 cookie를 붙이지 않음	HTTPS 설정·인증서 검증은 별도
`HttpOnly`	script API에서 cookie 직접 읽기 제한	XSS가 사용자의 요청을 대신 보내는 위험까지 제거하지 않음
`SameSite`	일부 cross-site 요청의 자동 cookie 첨부 제한	유일한 Cross-Site Request Forgery(CSRF) 방어로 쓰지 않음
`Path·Domain`	cookie 전송 범위 제한	보안 경계를 잘못 넓히면 다른 app과 충돌
host prefix	host-only·Secure 등 강한 제약 조합	browser 지원과 naming 규칙 확인

session ID에는 예측 가능한 user ID나 role을 직접 넣지 않는 opaque reference가 흔합니다. server는 session store에서 subject·AAL·발급·마지막 활동·취소 상태를 찾습니다.

6.2. session fixation을 막습니다

로그인 전 session ID를 로그인 뒤에도 그대로 쓰면 공격자가 미리 아는 ID에 인증 상태가 붙을 수 있습니다. 인증 성공과 권한 상승 시 session ID를 새로 발급하고 이전 ID를 무효화합니다.

6.3. client가 보낸 role을 믿지 않습니다

```
{"userId":"usr_17","role":"admin"}
```

요청 body·query에 있는 role은 권한의 근거가 아닙니다. 검증된 server session이나 token claims를 기반으로 subject를 만들고, 필요하면 최신 계정·조직·권한 상태를 조회합니다.

7. server session과 bearer token을 비교합니다

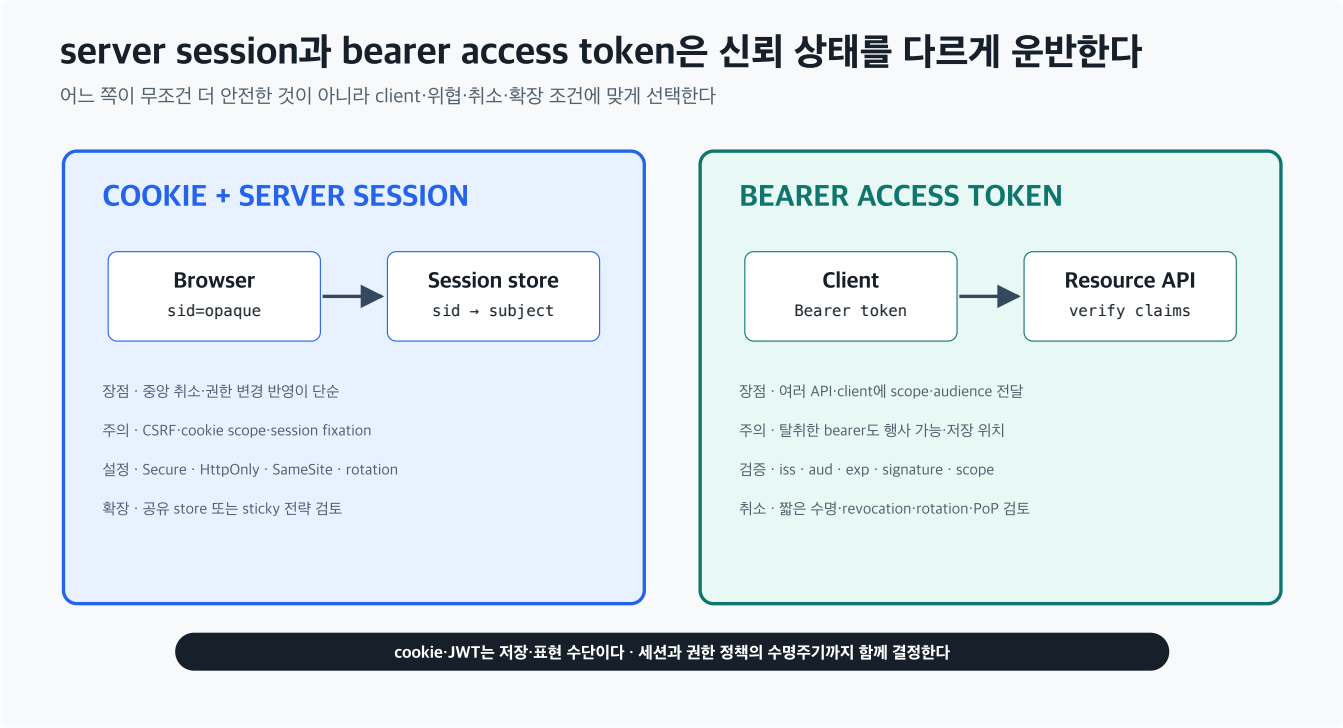
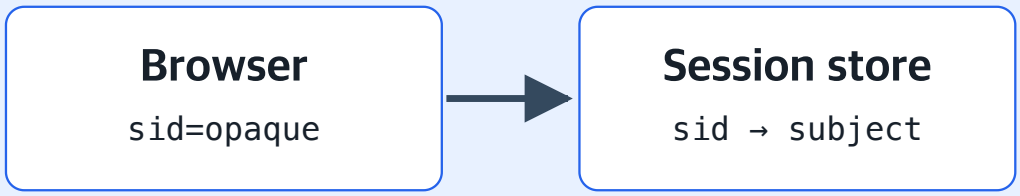


그림 4. server session은 opaque ID를 server 상태와 연결하고, bearer access token은 보호 API가 검증할 권한 정보를 운반할 수 있습니다. 선택 축을 섞지 않습니다.

COOKIE + SERVER SESSION



장점 · 중앙 취소·권한 변경 반영이 단순

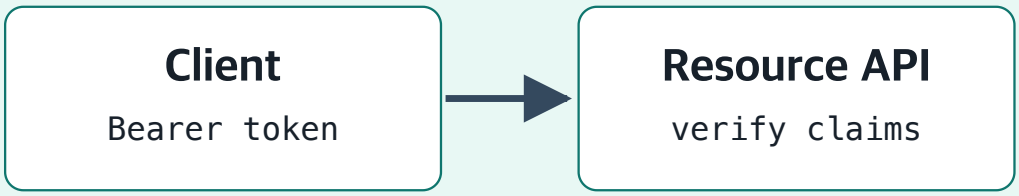
주의 · CSRF·cookie scope·session fixation

설정 · Secure · HttpOnly · SameSite · rotation

확장 · 공유 store 또는 sticky 전략 검토

cookie·JWT는 저장·표현 수단이다 · 세션

BEARER ACCESS TOKEN



장점 · 여러 API·client에 scope·audience 전달

주의 · 탈취한 bearer도 행사 가능·저장 위치

검증 · iss · aud · exp · signature · scope

취소 · 짧은 수명·revocation·rotation·PoP 검토

과 권한 정책의 수명주기까지 함께 결정한다

비교 축	cookie + server session	bearer access token
주 소비자	web app backend·session host	resource server API
client가 운반	session ID·secret	access token
권한 정보 위치	server store·정책 서비스	token claims 또는 introspection 결과
중앙 취소	store 상태 변경이 직접적	짧은 수명·revocation·introspection 등 설계 필요
대표 위험	fixation·hijacking·CSRF·cookie scope	token 탈취·leak·replay·audience 혼동
확장 고려	공유 session store·replication	issuer·key rotation·audience·scope·clock

7.1. 비교 축을 바로잡습니다

아래 문장은 서로 다른 층을 섞습니다.

> “세션 대신 JWT를 씁니다.”

더 정확한 질문은 다음과 같습니다.

1. 인증 연속성을 server state로 관리합니까, token으로 운반합니까?
2. client는 cookie 자동 전송입니까, Authorization header입니까?
3. token은 opaque입니까, JWT입니까?
4. 누가 issuer·audience·signature·expiry·scope를 검증합니까?
5. 만료 전 권한 변경·계정 정지·로그아웃을 어떻게 반영합니까?

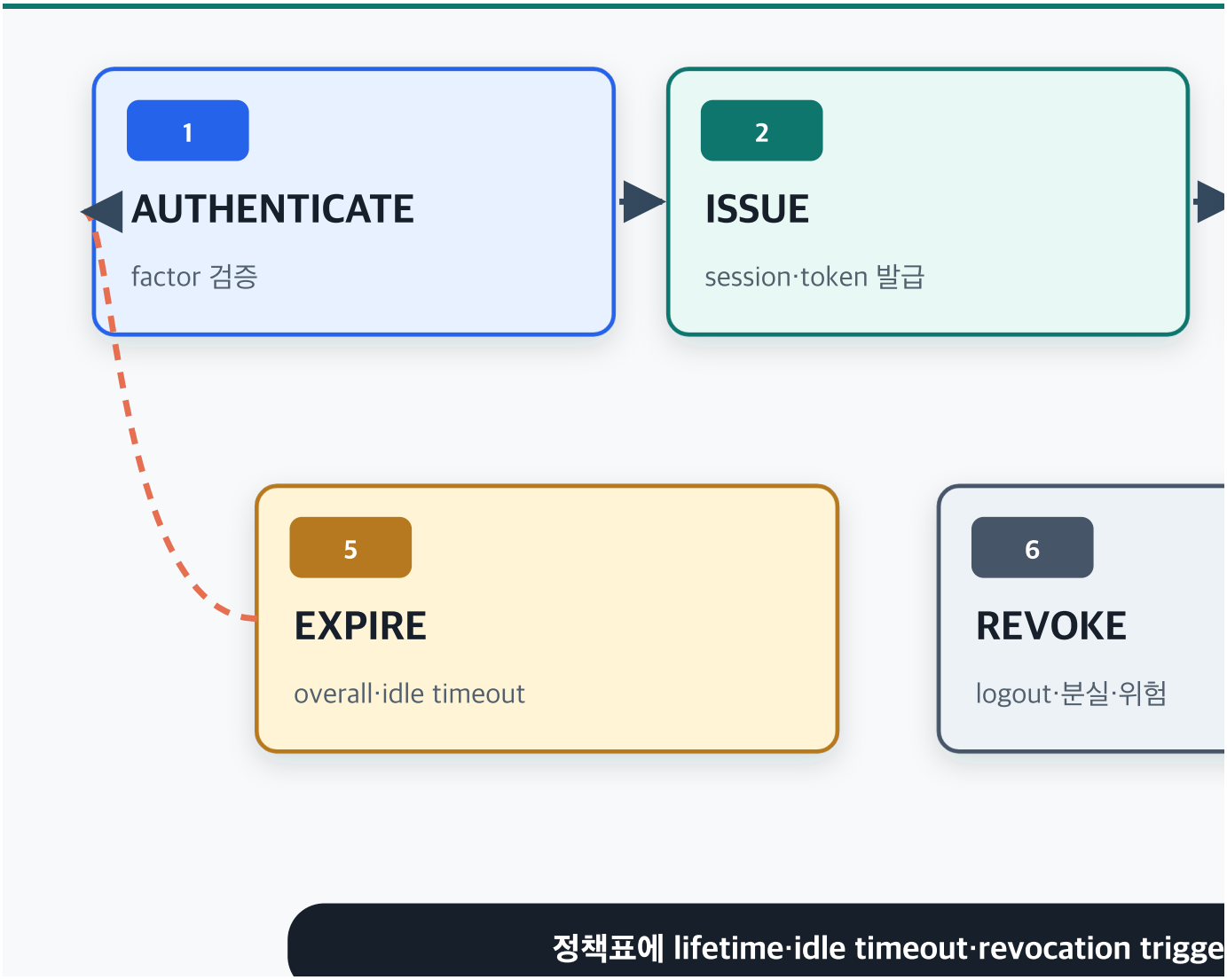
cookie 안에 서명 token을 넣을 수도 있고, OAuth access token이 opaque일 수도 있습니다. `cookie`, `JWT`, `session`, `OAuth`는 같은 차원의 선택지가 아닙니다.

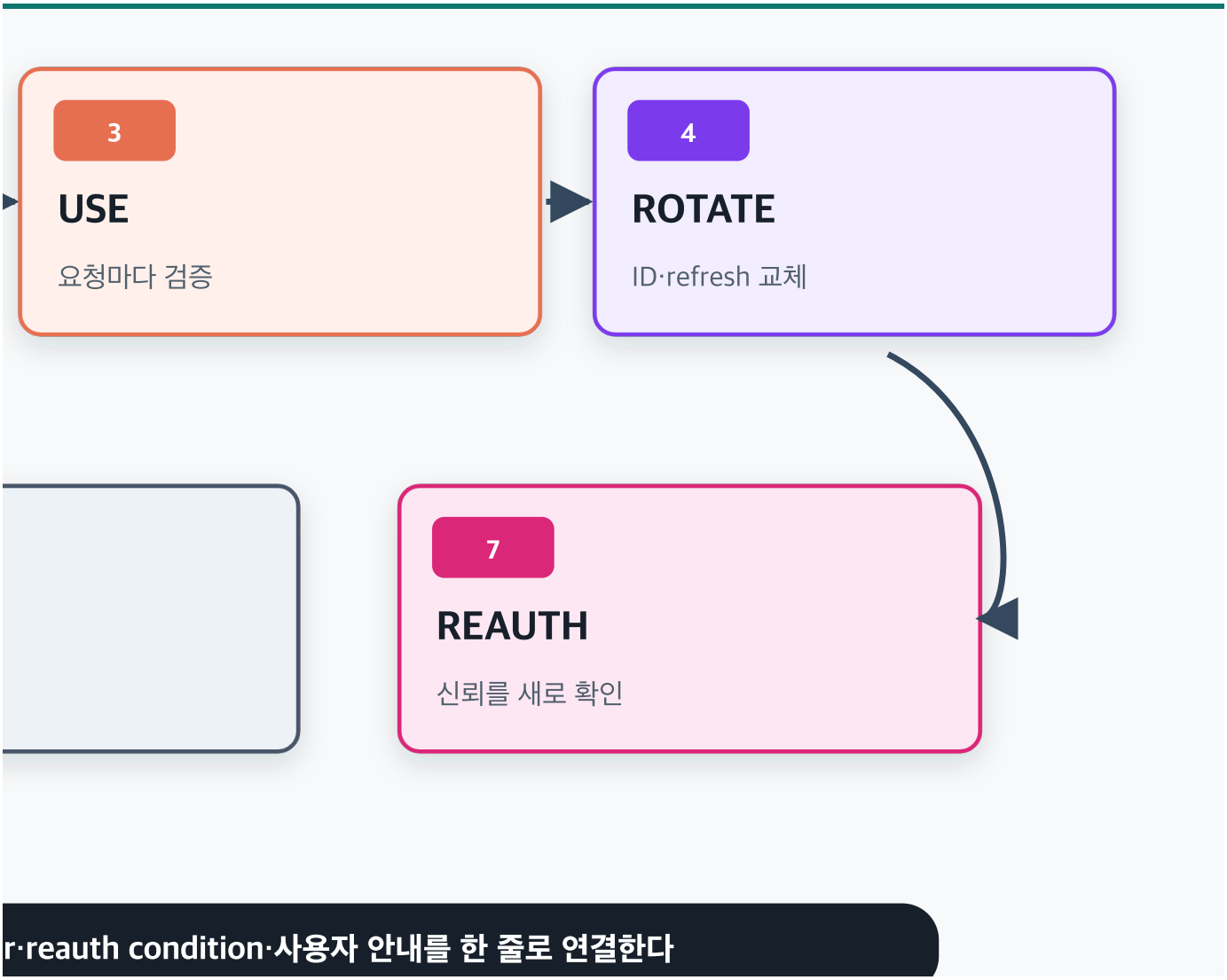
7.2. bearer는 “가진 사람이 행사”합니다

RFC 6750의 bearer token은 별도 key 소유 증명 없이 token을 가진 party가 관련 resource에 접근할 수 있는 방식입니다. 따라서 저장·전송·로그·URL·오류·분석 도구로 새지 않게 보호해야 합니다. 필요하면 mutual TLS나 Demonstrating Proof of Possession(DPoP) 같은 sender-constrained 방식도 위협 모델에 따라 검토합니다.

8. session·token 수명주기를 먼저 표로 닫습니다

그림 5. 인증 상태는 발급 이후 요청마다 검증되고, 회전·만료·취소·재인증을 거쳐 끝납니다. 로그아웃 버튼만으로 수명주기가 완성되지 않습니다.


1
AUTHENTICATE
factor 검증
2
ISSUE
session·token 발급
5
EXPIRE
overall·idle timeout
6
REVOKE
logout·분실·위험
정책표에 lifetime·idle timeout·revocation trigge


3
USE
요청마다 검증
4
ROTATE
ID·refresh 교체
7
REAUTH
신뢰를 새로 확인
r·reauth condition·사용자 안내를 한 줄로 연결한다

사건	정책 질문	검증 증거
issue	어떤 인증 사건 뒤 무엇을 발급하는가	`session_id`, `iat`, `auth_time`, `aal`
use	요청마다 무엇을 검증하는가	active·issuer·audience·expiry·scope·object policy
rotate	ID·refresh token·key를 언제 교체하는가	이전 값 무효화·reuse detection
idle timeout	활동이 없을 때 언제 끝나는가	`last_activity_at`
overall timeout	인증 후 최대 얼마인가	`authenticated_at`·absolute expiry
revoke	logout·분실·정지·권한 변경을 어떻게 반영하는가	revocation record·session version
reauth	어떤 중요 행동에서 신뢰를 새로 확인하는가	`acr`, `amr`, `auth_time`, max age

NIST는 overall timeout과 inactivity timeout을 구분합니다. 활동은 inactivity timeout을 다시 시작할 수 있지만 overall timeout을 무한히 늘려서는 안 됩니다. 성공적인 재인증은 정책에 따라 둘을 새로 설정할 수 있습니다.

8.1. logout의 범위를 명시합니다

logout 종류	종료 범위
현재 tab·app	client의 local state만 지우는가
현재 session	해당 session secret을 server에서 무효화하는가
모든 기기	account의 모든 session·refresh token을 취소하는가
연합 logout	RP session과 IdP session 중 어디까지 종료하는가

“로그아웃했습니다”라는 한 문장만으로는 부족합니다. 이미 발급된 access token, refresh token, server session, 다른 기기, IdP session의 상태를 각각 정의합니다.

8.2. 계정 정지와 권한 변경의 전파 시간을 정합니다

token 수명이 60분이면 방금 제거한 관리자 role이 최대 60분 남을 수 있습니다. 짧은 access token, token version, introspection, 중앙 policy 조회, revocation event 등으로 허용 가능한 지연을 설계합니다.

9. ID token·access token·refresh token의 소비자를 구분합니다

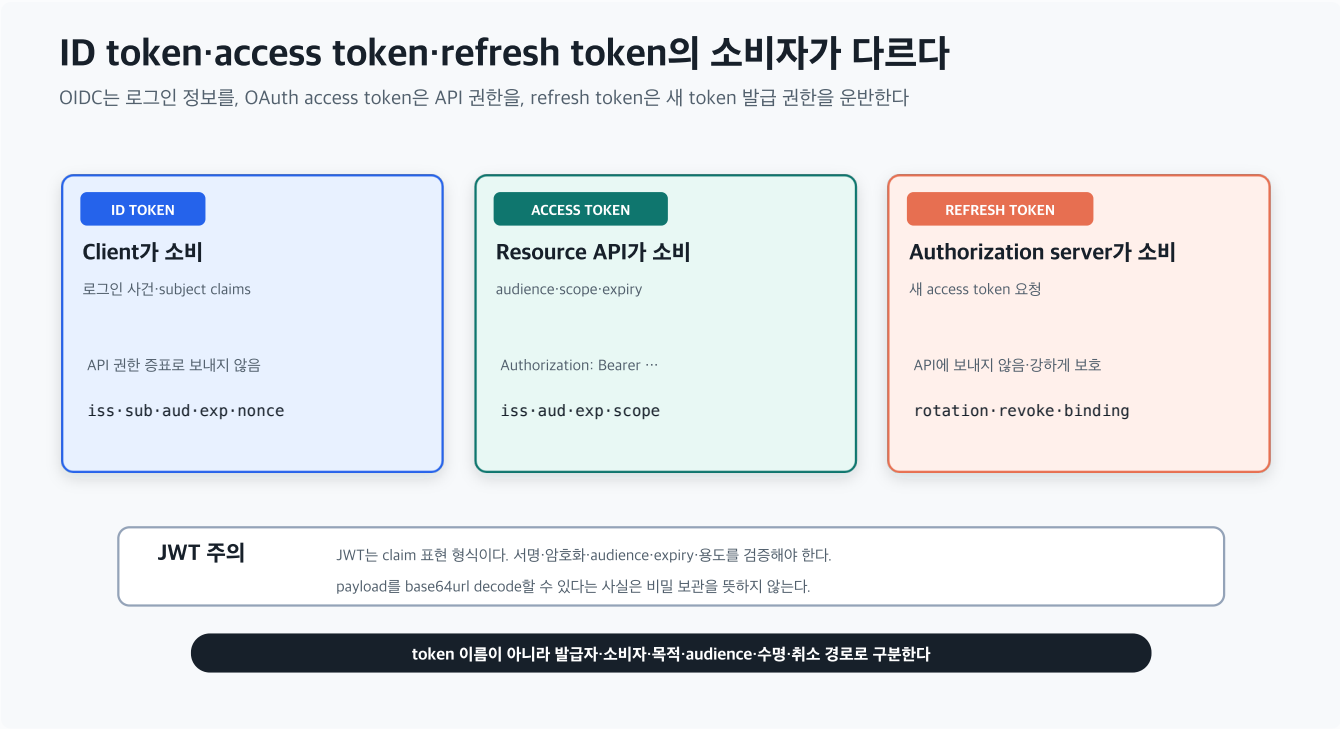


그림 6. 세 token은 이름보다 발급자·소비자·목적·audience·수명·취소 경로로 구분합니다. ID token을 API 권한 증표로 보내지 않습니다.

ID TOKEN

Client가 소비

로그인 사건·subject claims

API 권한 증표로 보내지 않음

`iss·sub·aud·exp·nonce`

ACCESS TOKEN

Resource API가 소

audience·scope·expiry

Authorization: Bearer …

`iss·aud·exp·scope`

JWT 주의

JWT는 claim 표현 형식이다. 서명·암호화·audience·exp

payload를 base64url decode할 수 있다는 사실은 비밀

token 이름이 아니라 발급자·소비자·목

비

REFRESH TOKEN

Authorization server가 소비

새 access token 요청

API에 보내지 않음·강하게 보호

`rotation·revoke·binding`

piry·용도를 검증해야 한다.

| 보관을 뜻하지 않는다.

적·audience·수명·취소 경로로 구분한다

token	주 소비자	목적	대표 검증
ID token	OpenID Connect client·relying party	사용자의 인증 사건과 claims 전달	`iss·sub·aud·exp·iat·nonce` 등
access token	resource server	보호 resource 접근 권한 전달	issuer·audience·expiry·signature 또는 introspection·scope
refresh token	authorization server token endpoint	새 access token 요청	client binding·rotation·reuse·revocation

9.1. OAuth와 OpenID Connect를 구분합니다

```
OAuth 2.0: client가 resource에 접근하도록 권한을 위임하는 framework
OpenID Connect: OAuth 2.0 위의 identity layer로 로그인·인증 정보를 전달
```

“OAuth 로그인”이라고 부르는 기능은 보통 OpenID Connect를 함께 사용합니다. OAuth access token만 받아 임의의 profile API를 호출해 로그인으로 해석하는 임시 설계를 피합니다.

9.2. JWT는 session 정책이 아니라 claim 형식입니다

RFC 7519의 JSON Web Token(JWT)은 claims를 JSON으로 표현해 JWS로 서명·MAC하거나 JWE로 암호화할 수 있는 형식입니다.

```
base64url decode 가능 ≠ 암호화됨
서명 유효 ≠ 내 API용 audience임
expiry 남음 ≠ account·grant가 아직 허용됨
JWT임 ≠ access token임
```

검증 library에 허용 algorithm·issuer·audience·clock tolerance를 명시하고, token 종류를 혼동하지 않습니다. client는 access token 내부 형식에 의존하지 않는 것이 안전한 상호운용 원칙입니다.

9.3. 외부 로그인은 검증된 구현을 사용합니다

Authorization Code flow와 Proof Key for Code Exchange(PKCE), 정확한 redirect URI, `state·nonce`, issuer·ID token 검증, key rotation은 직접 간이 구현하지 않습니다. RFC 9700의 최신 OAuth 보안 권고를 따르는 검증된 OpenID Connect library와 provider metadata를 사용합니다.

10. 권한을 네 입력의 함수로 씁니다

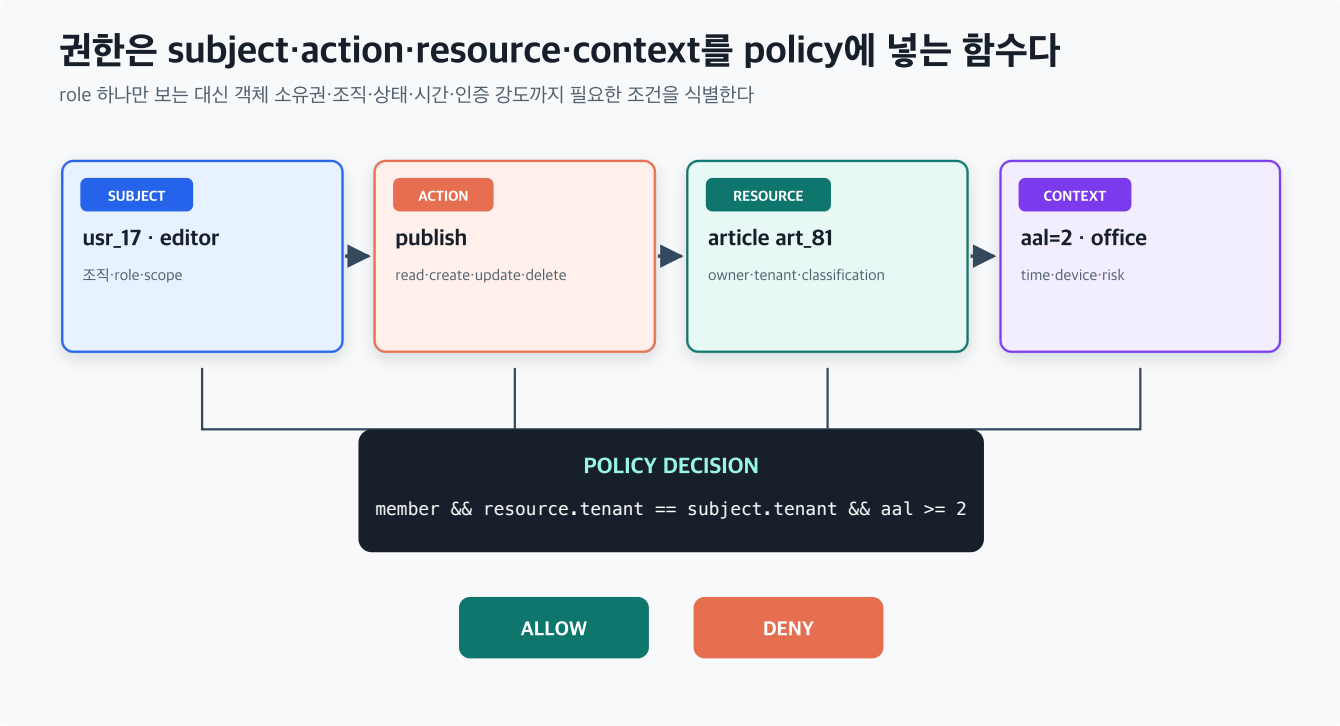
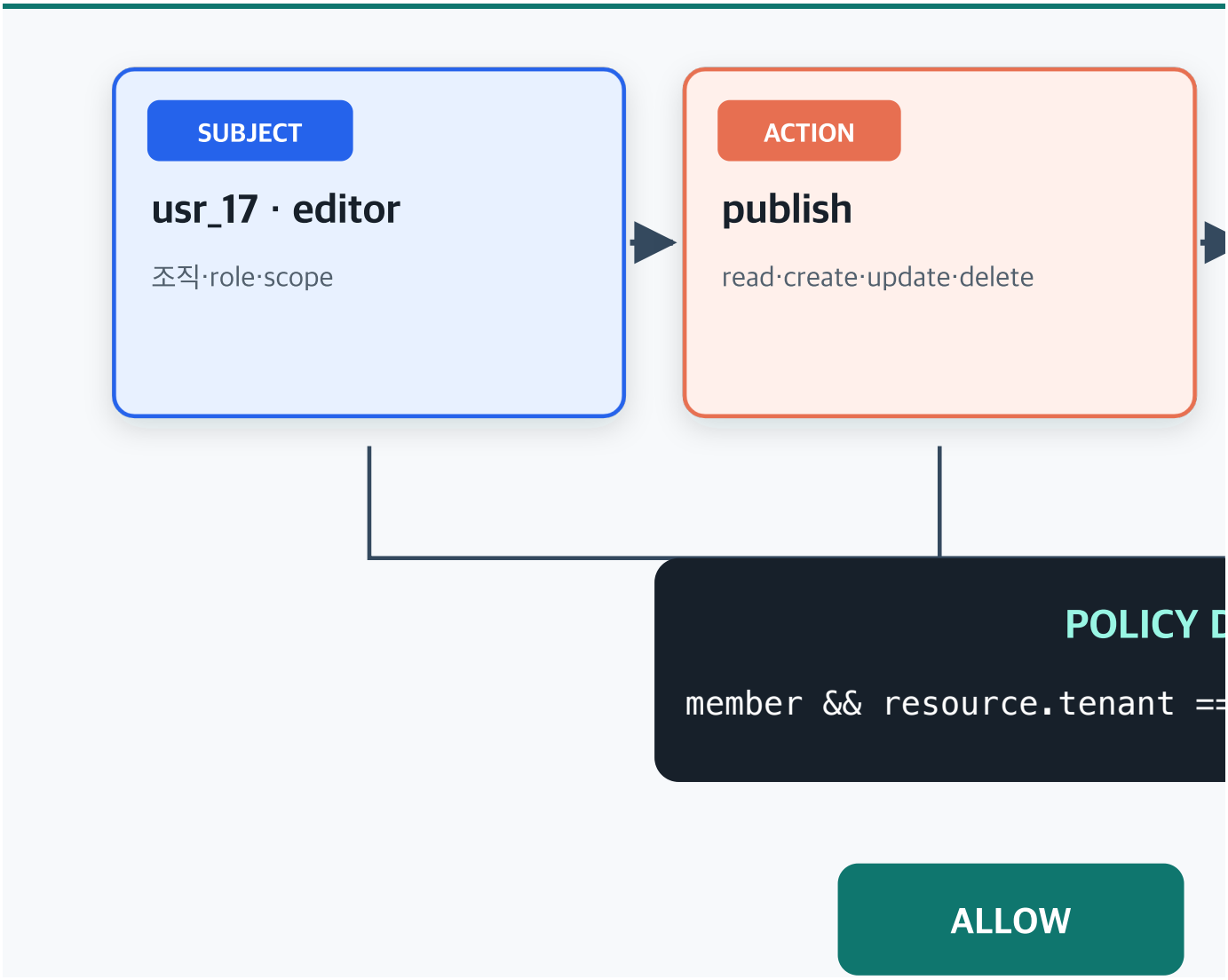
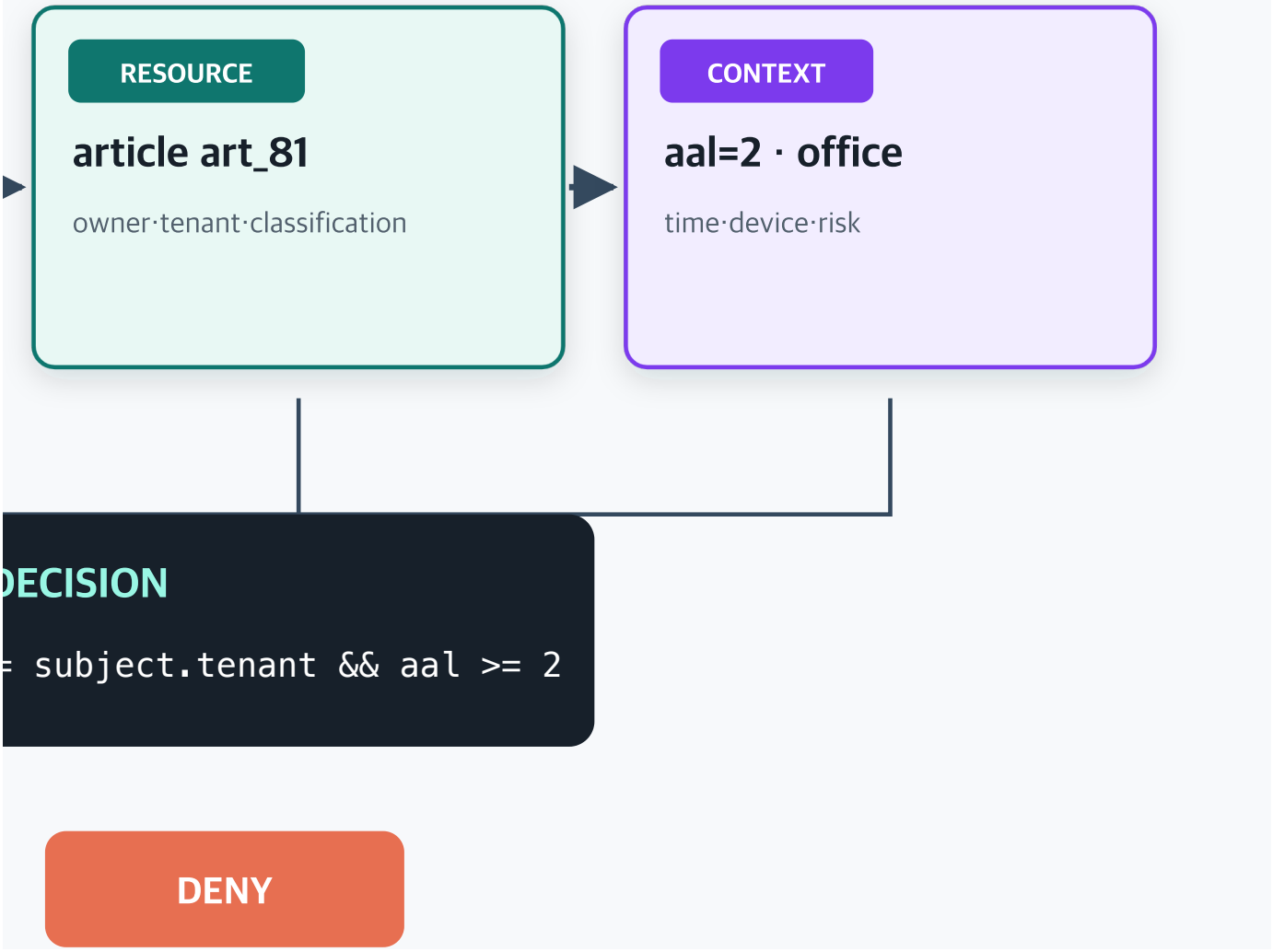


그림 7. 권한 판정은 subject·action·resource·context를 정책에 넣어 ALLOW·DENY·CHALLENGE를 결정하는 함수로 볼 수 있습니다.


SUBJECT
usr_17 · editor
조직·role·scope
ACTION
publish
read·create·update·delete
POLICY D
member && resource.tenant ==
ALLOW


RESOURCE
article art_81
owner·tenant·classification
CONTEXT
aal=2 · office
time·device·risk
ECISION
= subject.tenant && aal >= 2
DENY

```
decision = policy(subject, action, resource, context)
```

입력	예	출처
subject	`id=usr_17`, `role=editor`, `tenant=org_1`	검증된 session·token·directory
action	`read`, `refund`, `publish`, `delete`	method·operation·업무 command
resource	`order=ord_81`, `owner=usr_17`, `state=paid`	server가 조회한 실제 객체
context	`aal=2`, `auth_age=90`, `device=managed`, `risk=low`	인증·환경·위험 engine
policy	`same tenant AND scope AND amount <= limit`	승인된 중앙 정책

10.1. policy input은 신뢰 경계를 표시합니다

```
신뢰 가능: server가 token signature와 audience를 검증해 만든 subject
검증 필요: DB에서 orderId로 찾은 owner·tenant·state
신뢰 불가: client가 body에 넣은 role·owner·price·isAdmin
```

10.2. allow·deny·challenge를 구분합니다

decision	뜻	예
ALLOW	현재 조건으로 요청 실행	owner가 자신의 주문 조회
DENY	다른 인증을 해도 현재 정책상 불가할 수 있음	다른 사용자의 주문 조회
CHALLENGE	더 강하거나 최근의 인증이 있으면 재평가	고액 환불 전 step-up

정책 engine이 boolean만 반환해도 application은 `policy_id`, `reason_code`, `obligations`, `audit fields`를 함께 관리할 수 있습니다. 민감한 내부 정책 전체를 client에 노출하지는 않습니다.

11. role·scope·소유권·속성·관계는 다른 질문입니다

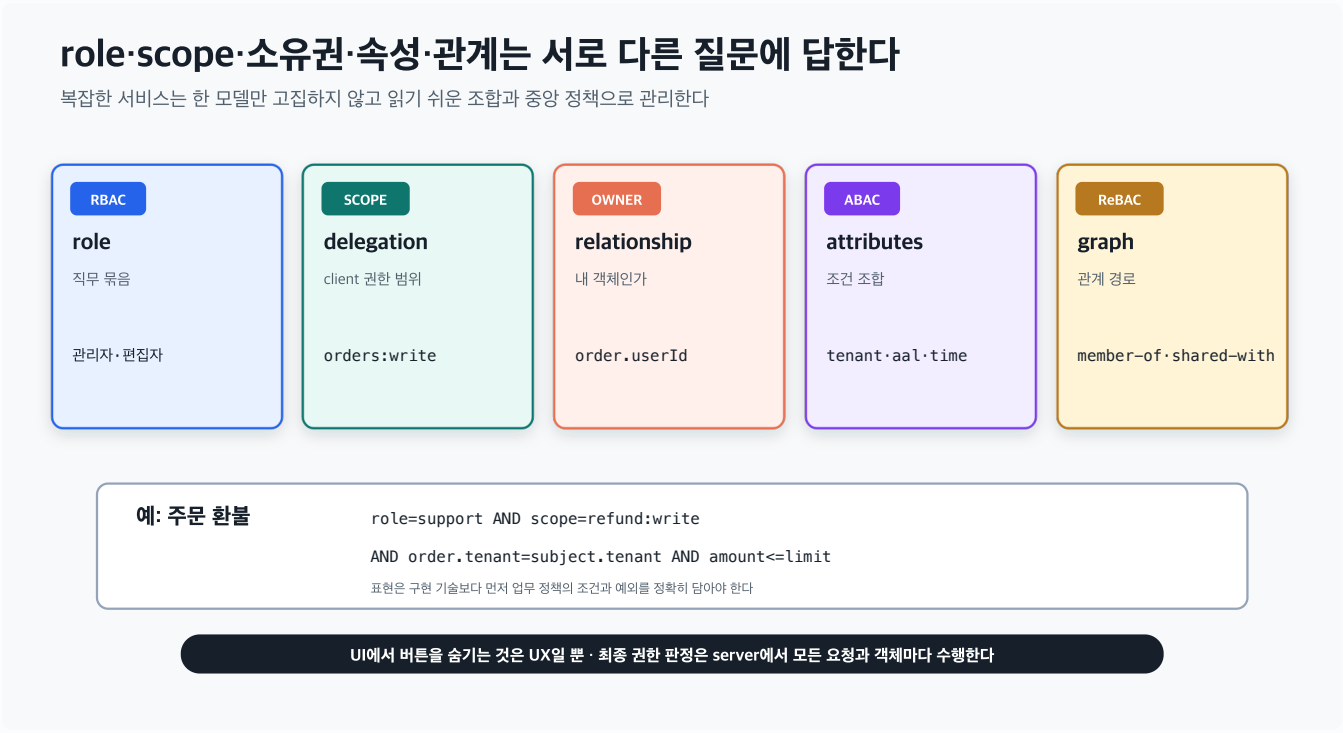
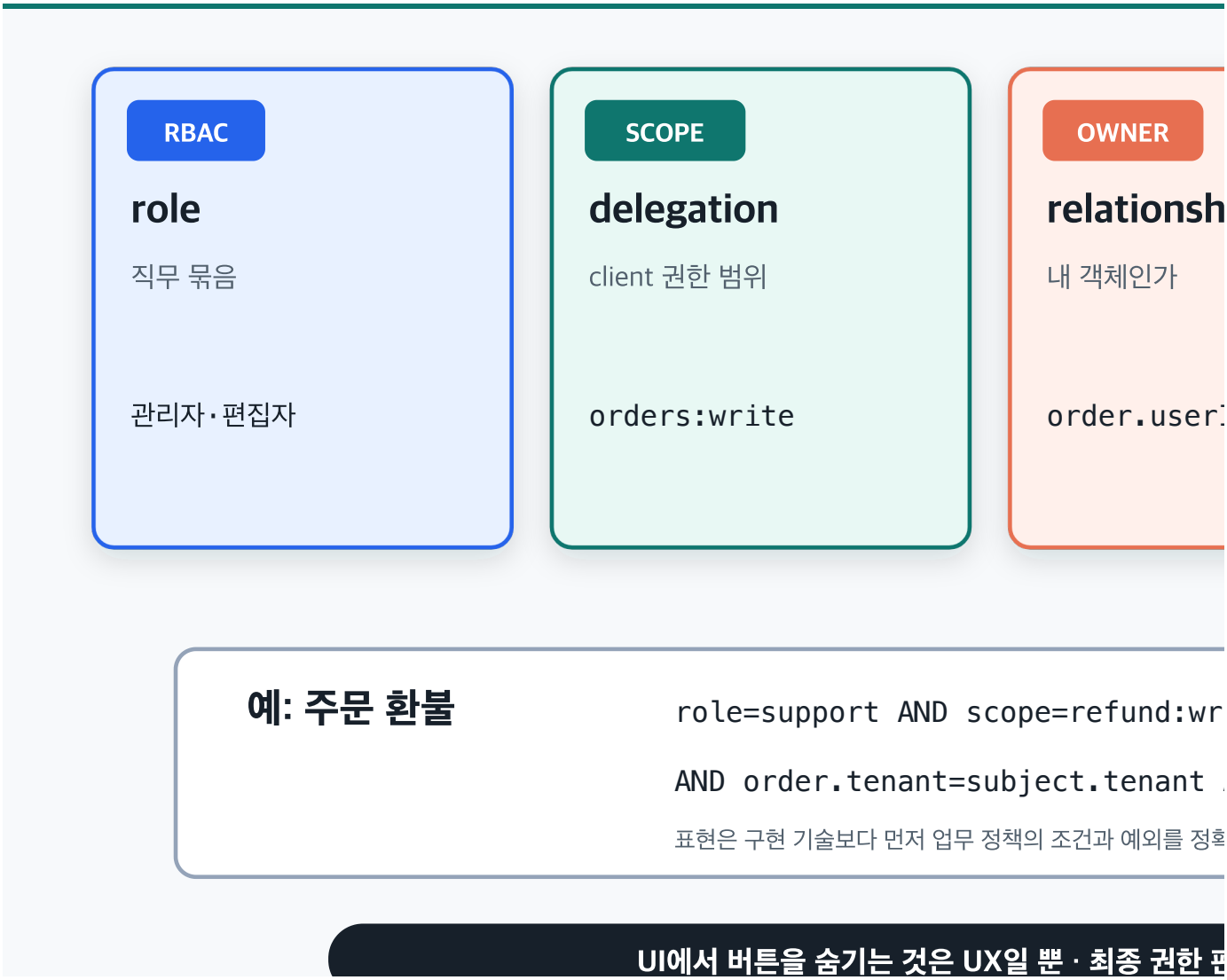
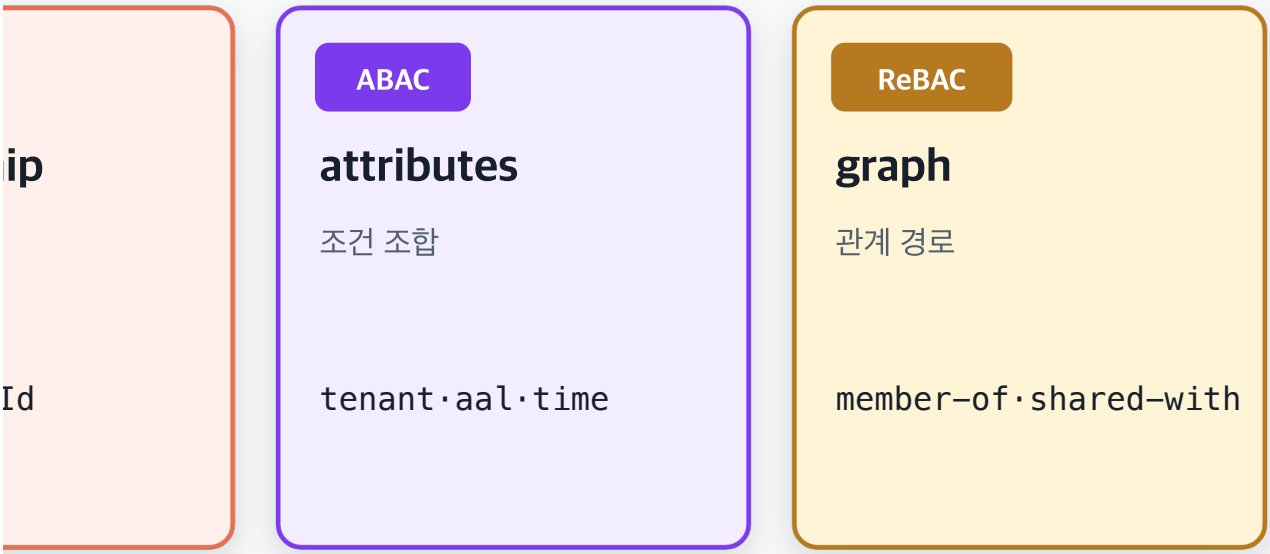


그림 8. 역할 기반, scope, 소유권, 속성 기반, 관계 기반 정책은 서로 대체되는 한 줄 용어가 아니라 다른 업무 질문을 표현합니다.


RBAC
role
직무 묶음
관리자·편집자
SCOPE
delegation
client 권한 범위
orders:write
OWNER
relationsh
내 객체인가
order.user
예: 주문 환불
role=support AND scope=refund:wr
AND order.tenant=subject.tenant
표현은 구현 기술보다 먼저 업무 정책의 조건과 예외를 정확
UI에서 버튼을 숨기는 것은 UX일 뿐 · 최종 권한 판



ite

AND amount<=limit

확히 담아야 한다

판정은 server에서 모든 요청과 객체마다 수행한다

모델·조건	답하는 질문	예
Role-Based Access Control(RBAC)	어떤 직무 묶음인가	`role=support`
scope	client에 어떤 위임 범위가 발급됐는가	`refund:write`
ownership	이 객체가 이 subject의 것인가	`order.userId == subject.id`
Attribute-Based Access Control(ABAC)	subject·object·environment 속성이 정책을 만족하는가	tenant·금액·시간·AAL
Relationship-Based Access Control(ReBAC)	graph 관계 경로가 있는가	member-of·shared-with

OWASP는 복잡한 application에서 RBAC만으로 모든 객체·관계를 표현하기보다 attribute·relationship 조건을 검토하라고 안내합니다. 그렇다고 무조건 복잡한 policy language를 도입하라는 뜻은 아닙니다. **업무 규칙을 가장 읽기 쉽게 표현하고 중앙에서 일관되게 집행할 조합**을 고릅니다.

11.1. scope와 role은 같지 않습니다

```
role=support: 조직 안에서 맡은 직무
scope=refund:write: 이 client·grant가 위임받은 API 범위
```

support role이어도 현재 access token에 환불 scope가 없을 수 있습니다. 반대로 scope 문자열이 있어도 실제 subject의 조직·객체·한도 정책을 통과하지 못할 수 있습니다.

11.2. tenant는 모든 객체 경로에서 확인합니다

multi-tenant 서비스에서 URL의 `tenantId`만 믿지 않습니다. subject tenant와 resource의 실제 tenant를 server에서 비교하고, nested resource와 batch·export·search 결과에도 같은 경계를 적용합니다.

12. 권한 매트릭스로 정책과 테스트를 한 장에 묶습니다

권한 매트릭스는 역할이 아니라 행동·객체·조건까지 쓴다

허용 칸에는 근거 조건을, 거부 칸에는 오류·노출 정책·감사 이벤트를 함께 정의한다

SUBJECT / RESOURCE	VIEW	UPDATE	DELETE	REFUND	PUBLISH
customer / own order	ALLOW	before paid	DENY	DENY	-
support / tenant order	ALLOW	note only	DENY	under limit	-
editor / own article	ALLOW	ALLOW	draft only	-	aal2
admin / any article	ALLOW	ALLOW	step-up	-	ALLOW
unknown / any	401	401	401	401	401

각 칸의 테스트 = 허용 1 + 다른 role 1 + 다른 owner 1 + 경계값 1 + 만료·취소 1
새 기능은 빈 칸을 ALLOW로 추정하지 않고 deny by default에서 출발한다

그림 9. 매트릭스 칸에는 ALLOW·DENY만 쓰지 않고 소유권·상태·한도·AAL 같은 조건을 적습니다. 각 칸이 테스트 묶음이 됩니다.

SUBJECT / RESOURCE	VIEW	UPDATE
customer / own order	ALLOW	before paid
support / tenant order	ALLOW	note only
editor / own article	ALLOW	ALLOW
admin / any article	ALLOW	ALLOW
unknown / any	401	401

각 칸의 테스트 = 허용 1 + 다른 role 1 +
새 기능은 빈 칸을 ALLOW로 추정하지

DELETE	REFUND	PUBLISH
DENY	DENY	-
DENY	under limit	-
draft only	-	aal2
step-up	-	ALLOW
401	401	401

다른 owner 1 + 경계값 1 + 만료·취소 1
않고 deny by default에서 출발한다

12.1. 매트릭스의 축

축	반드시 적을 것
subject	role·조직·tenant·외부 client·service account
resource	종류·소유자·tenant·분류·상태
action	read·create·update·delete보다 구체적인 publish·refund·export
condition	scope·amount·AAL·auth age·시간·device·approval
deny behavior	401·403·404·step-up·승인 필요
evidence	policy ID·request ID·subject·resource·reason

12.2. deny by default에서 시작합니다

명시된 ALLOW가 없는 조합은 거부합니다. 새 endpoint·새 action·새 resource type이 생길 때 framework default에 기대지 않고 정책과 테스트가 추가될 때만 열립니다.

12.3. 모든 요청·모든 객체에서 판정합니다

```
목록에서 숨김 → UX
버튼 disabled → UX
route guard → client navigation 보호
server policy check → 최종 권한 집행
```

client-side check는 사용자 경험을 개선하지만 최종 보안 경계가 아닙니다. URL 직접 입력, API 직접 호출, batch, export, background job, websocket message, static file에도 필요한 server-side 검사를 둡니다.

12.4. 허용 테스트만으로는 부족합니다

각 ALLOW 칸마다 최소한 다음 반례를 만듭니다.

1. 인증 정보 없음·만료·취소
2. 다른 role
3. 같은 role이지만 다른 owner·tenant
4. scope 누락
5. 금액·시간·상태 경계값

6. AAL·auth age 부족

7. 식별자 변조·batch 일부 객체 혼합

13. 401·403·404와 다음 행동을 연결합니다

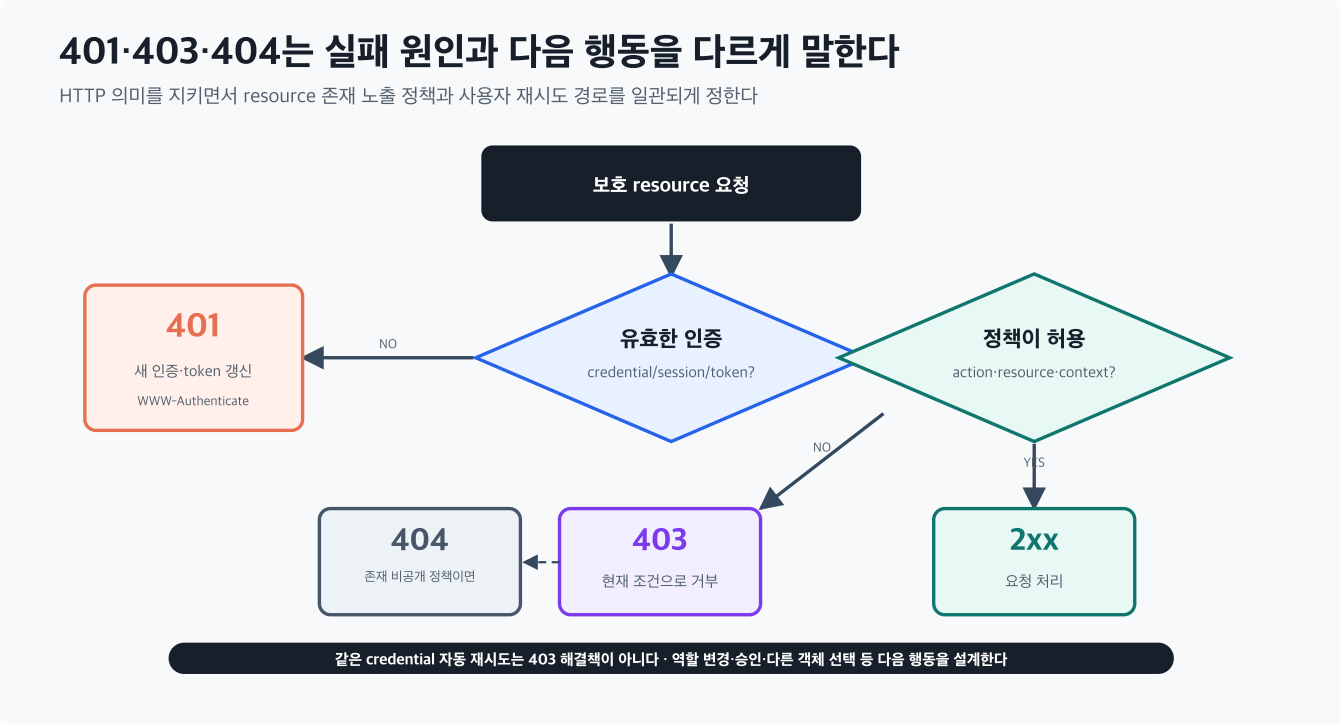
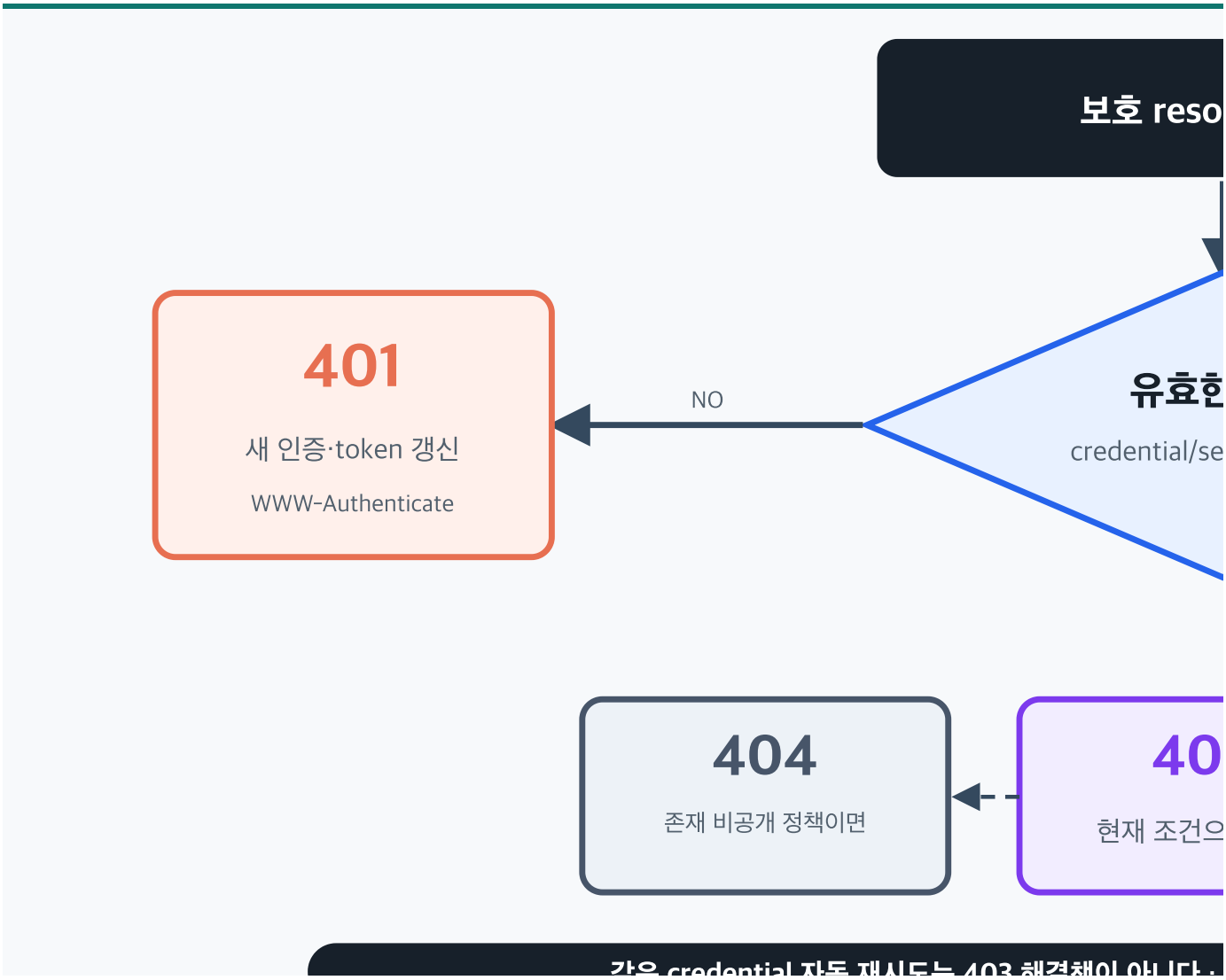
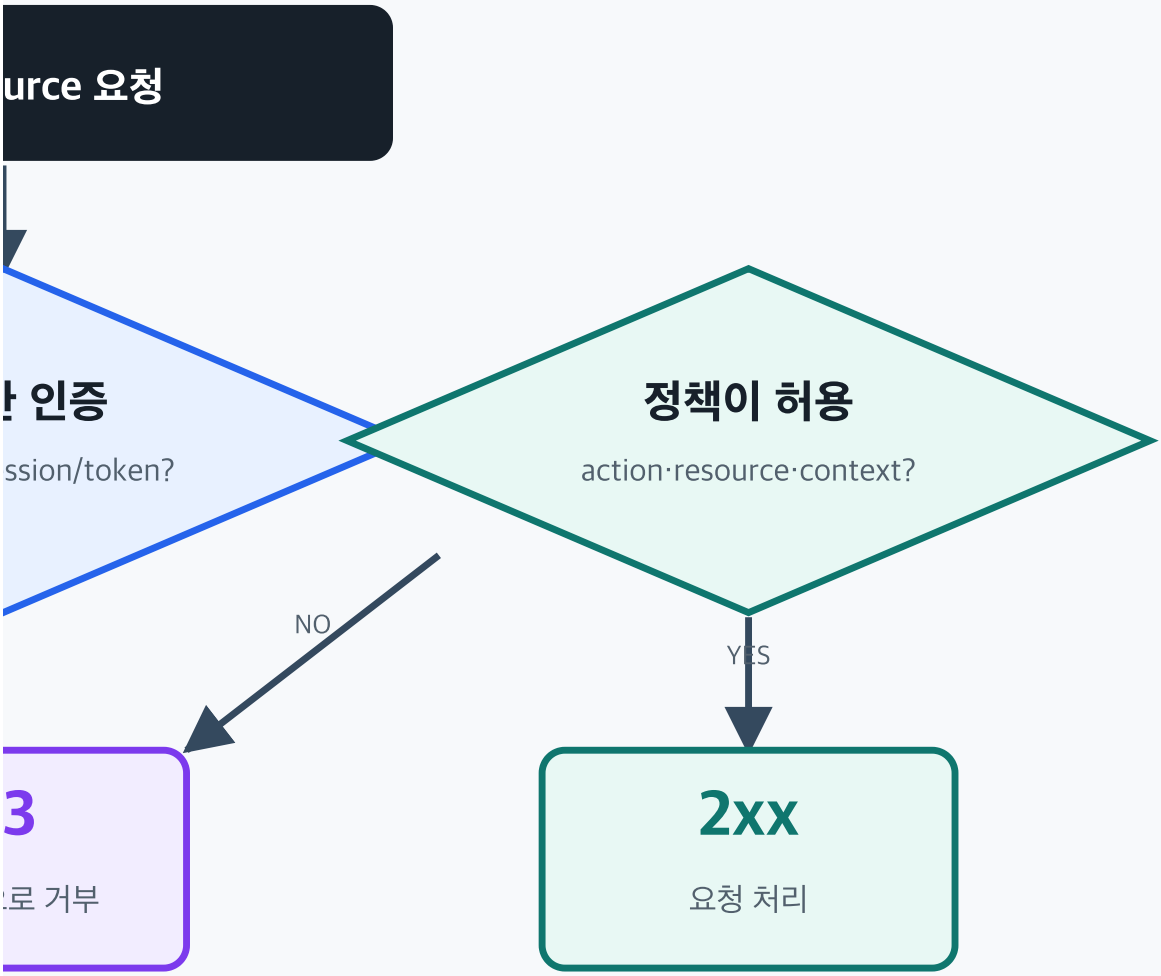


그림 10. 유효한 인증 정보가 없으면 401, 인증됐지만 정책이 거부하면 403, resource 존재를 숨기려면 일관된 404 정책을 사용할 수 있습니다.


보호 reso
401
새 인증·token 갱신
WWW-Authenticate
NO
유효한
credential/se
404
존재 비공개 정책이면
40
현재 조건으


urce 요청
인증
ssion/token?
정책이 허용
action·resource·context?
NO
YES
3
로 거부
2xx
요청 처리

응답 코드	HTTP 의미	대표 원인	client 다음 행동
401	target resource에 유효한 authentication credentials가 없음	미로그인·만료·invalid token·필요 assurance 부족	새 인증·token 갱신·step-up
403	server가 이해했지만 요청 수행을 거부	role·scope·소유권·상태·정책 불충족	다른 resource·승인·권한 요청
404	현재 representation이 없거나 존재 공개를 원하지 않음	실제 없음·비공개 존재 정책	접근 가능한 목록에서 다시 선택

RFC 9110은 401 response에 적용 가능한 `WWW-Authenticate` challenge를 요구합니다. RFC 6750은 bearer token의 `invalid_token`과 `insufficient_scope`를 정의하며 insufficient scope에는 403을 사용합니다.

13.1. 401의 이름에 속지 않습니다

status phrase는 `Unauthorized`지만 의미는 유효한 **인증 자격 증명(authentication credentials)** 부족에 가깝습니다. 그래서 "로그인 안 됨은 401, 로그인됨은 무조건 403"처럼 단순 암기하지 말고 credential 유효성과 정책 결과를 봅니다.

13.2. 403을 반복 로그인 화면으로 보내지 않습니다

같은 credential로 자동 재시도해도 403은 보통 해결되지 않습니다. 필요한 scope·승인·role 요청, 허용된 객체 선택, 상위 담당자 이관처럼 실제 해결 경로를 설계합니다.

13.3. 404 은폐 정책은 일관되어야 합니다

다른 사용자의 비공개 resource가 존재함을 숨기려 404를 사용할 수 있습니다. detail·response time·목록 count·검색·파일 URL·감사 log에서 존재가 새지 않는지 검토합니다. 모든 403을 무조건 404로 바꾸는 규칙은 아닙니다.

13.4. problem details 예

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json

{
  "type": "https://api.example/problems/insufficient-scope",
  "title": "요청한 작업을 수행할 권한이 없습니다",
  "status": 403,
  "requiredScope": "refund:write",
  "requestId": "req_81"
}
```

민감한 내부 role 구조·다른 사용자 정보·정책 식을 detail로 그대로 노출하지 않습니다.

14. 중요 행동은 재인증·step-up으로 신뢰를 새로 확인합니다

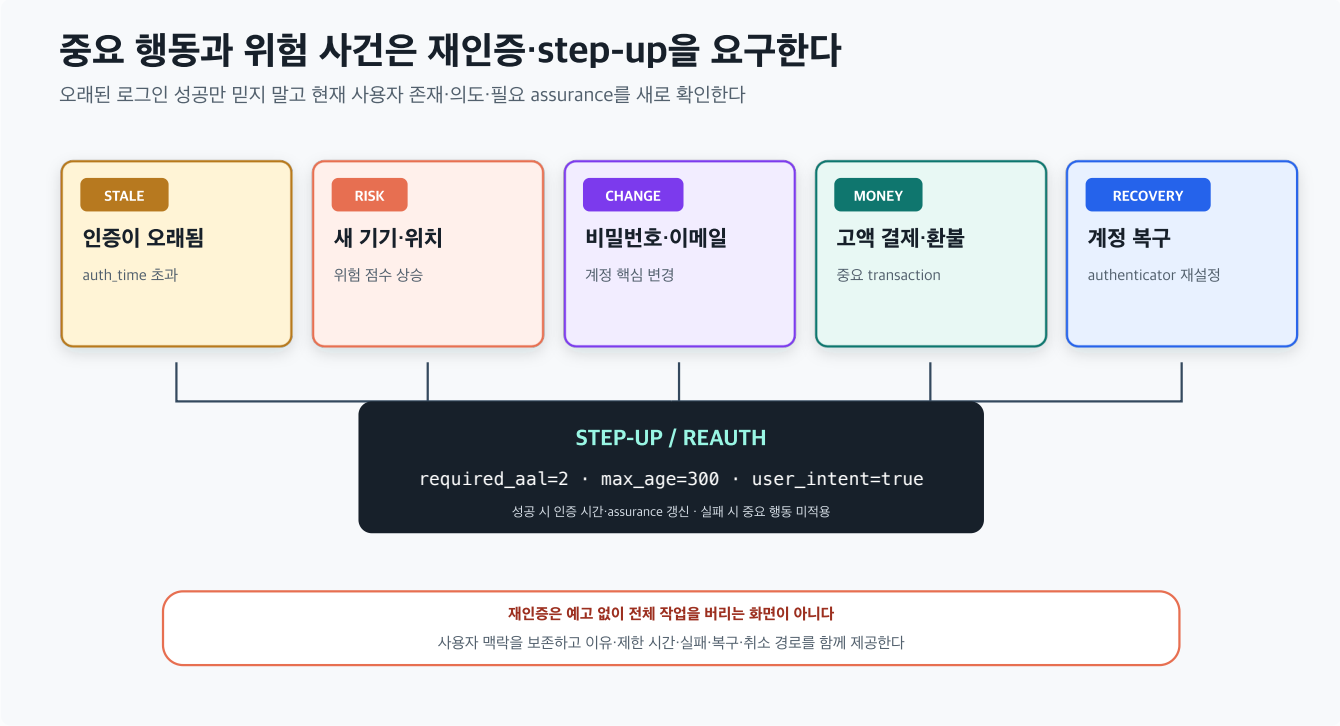
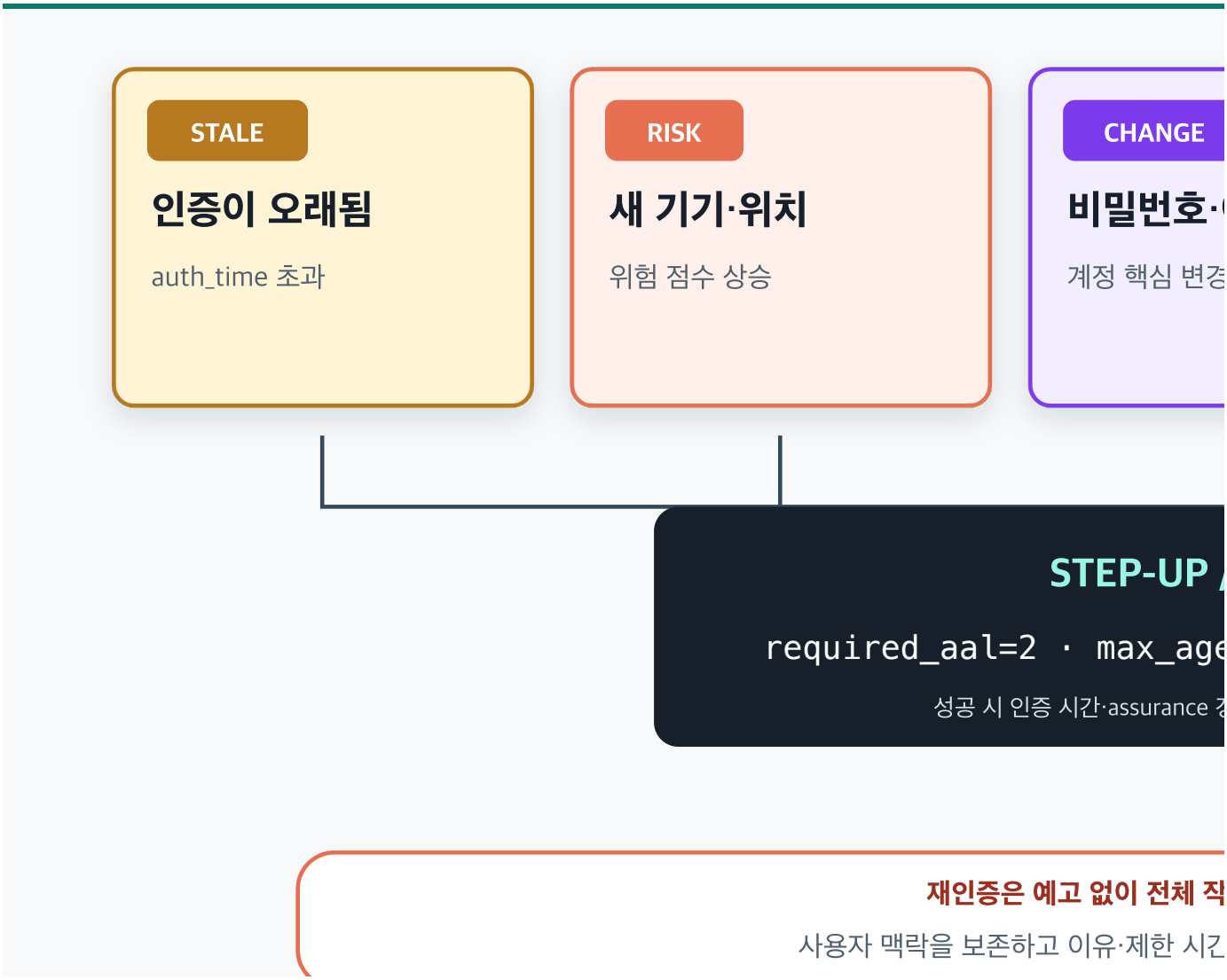
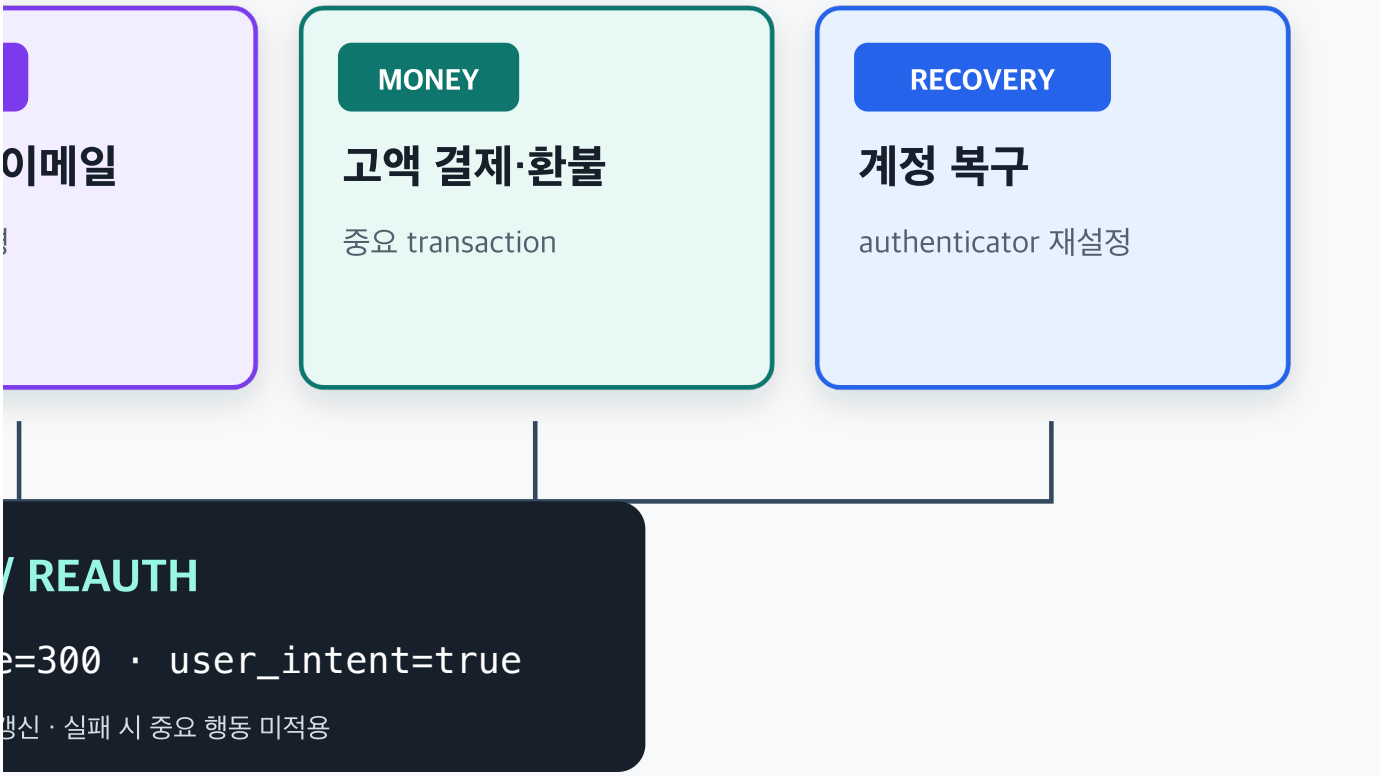


그림 11. 인증이 오래됐거나 위험 신호·계정 변경·고액 행동·복구가 있으면 필요한 강도와 최신성을 새로 만족시킵니다.


STALE
인증이 오래됨
auth_time 초과
RISK
새 기기·위치
위험 점수 상승
CHANGE
비밀번호·
계정 핵심 변경
STEP-UP
required_aal=2 · max_age
성공 시 인증 시간·assurance
재인증은 예고 없이 전체 직
사용자 맥락을 보존하고 이유·제한 시간



·업을 버리는 화면이 아니다

·실패·복구·취소 경로를 함께 제공한다

14.1. trigger 예

trigger	필요한 조치 예
password·email·MFA 변경	최근 인증·현재 authenticator 확인
고액 결제·환불·전자서명	transaction 문맥을 보여 주고 사용자 의도 확인
새 device·위치·위험 점수	추가 factor·알림·제한
관리자 삭제·대량 export	AAL2·auth age 5분 이내·승인
계정 복구·authenticator 재등록	강한 복구·기존 session 취소

14.2. OAuth resource server의 step-up

RFC 9470은 access token에 연결된 인증 강도·최신성이 resource server 요구보다 낮을 때 `insufficient_user_authentication` challenge와 `acr_values·max_age`를 사용해 더 적합한 인증을 요청하는 방식을 정의합니다.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="insufficient_user_authentication",
  acr_values="aal2", max_age="300"
```

이 profile을 쓰지 않는 일반 web session에서도 같은 개념을 policy로 설계할 수 있습니다. 다만 임의의 header를 표준처럼 만들지 않고 application 계약을 명시합니다.

14.3. 재인증 UX는 작업 맥락을 보존합니다

재인증 전에 어떤 행동 때문에 필요한지 알리고, 성공하면 원래 요청을 안전하게 다시 확인합니다. 중요 transaction은 재인증 뒤 대상·금액·수신자 같은 문맥이 바뀌지 않았는지 다시 검증합니다.

15. 권한 판정은 감사 가능한 증거를 남깁니다

감사 log는 password·session secret·access token 원문을 저장하는 곳이 아닙니다.

```
{
  "event": "authorization_decision",
  "requestId": "req_81",
  "subject": "usr_17",
  "action": "refund",
  "resource": "order:ord_81",
  "tenant": "shop_1",
  "policy": "order.support.refund.v3",
  "decision": "DENY",
  "reason": "amount_limit_exceeded",
  "httpStatus": 403
}
```

기록	목적	주의
request·trace ID	분산 요청 연결	외부 공개 ID와 내부 추적 ID 구분
subject·client	누가 요청했는지	최소 식별자·보존 기간
action·resource	무엇을 시도했는지	민감 resource 값 마스킹
policy ID·version	어떤 규칙을 적용했는지	정책 변경 이력 유지
decision·reason	왜 허용·거부했는지	내부 상세를 client에 그대로 노출하지 않음
auth context	AAL·auth age·IdP	authenticator secret은 기록 금지

권한 변경·관리자 role 부여·service account secret 발급·복구 성공·모든 session 취소는 별도 고위험 event로 기록하고 필요한 담당자·사용자에게 알립니다.

16. 대표 실패를 공격 이름보다 정책 결함으로 읽습니다

실패	빠진 계약	설계·테스트
credential stuffing	rate limit·유출 password·위험 탐지	다수 계정·IP·device 분산 시도
account enumeration	외부 메시지·timing·recovery 응답	존재·비존재 계정 비교
session fixation	인증·권한 상승 시 ID 회전	로그인 전 ID 재사용 시도
session hijacking	secret 보호·만료·취소·재인증	탈취 cookie·token replay
CSRF	cookie 자동 전송·origin·anti-CSRF	cross-site state-changing request

실패	빠진 계약	설계·테스트
XSS token theft	저장 위치·CSP·출력 encoding	script가 token·session으로 행동
Broken Object Level Authorization	객체별 owner·tenant 판정	ID만 다른 요청·batch 혼합
stale privilege	권한 변경 전파 시간·token lifetime	role 제거 직후 기존 token 사용
recovery bypass	복구 assurance·기존 session 취소	약한 지원 절차·기기 변경
confused deputy	audience·client·resource binding	다른 API용 token 재사용

보안 기능을 직접 새 암호·token 형식으로 만들지 않습니다. 검증된 identity provider, framework middleware, policy library를 사용하되 application의 실제 객체·업무 조건을 별도 정책과 테스트로 채웁니다.

17. 실습 · 요청 하나를 여덟 gate로 판정합니다

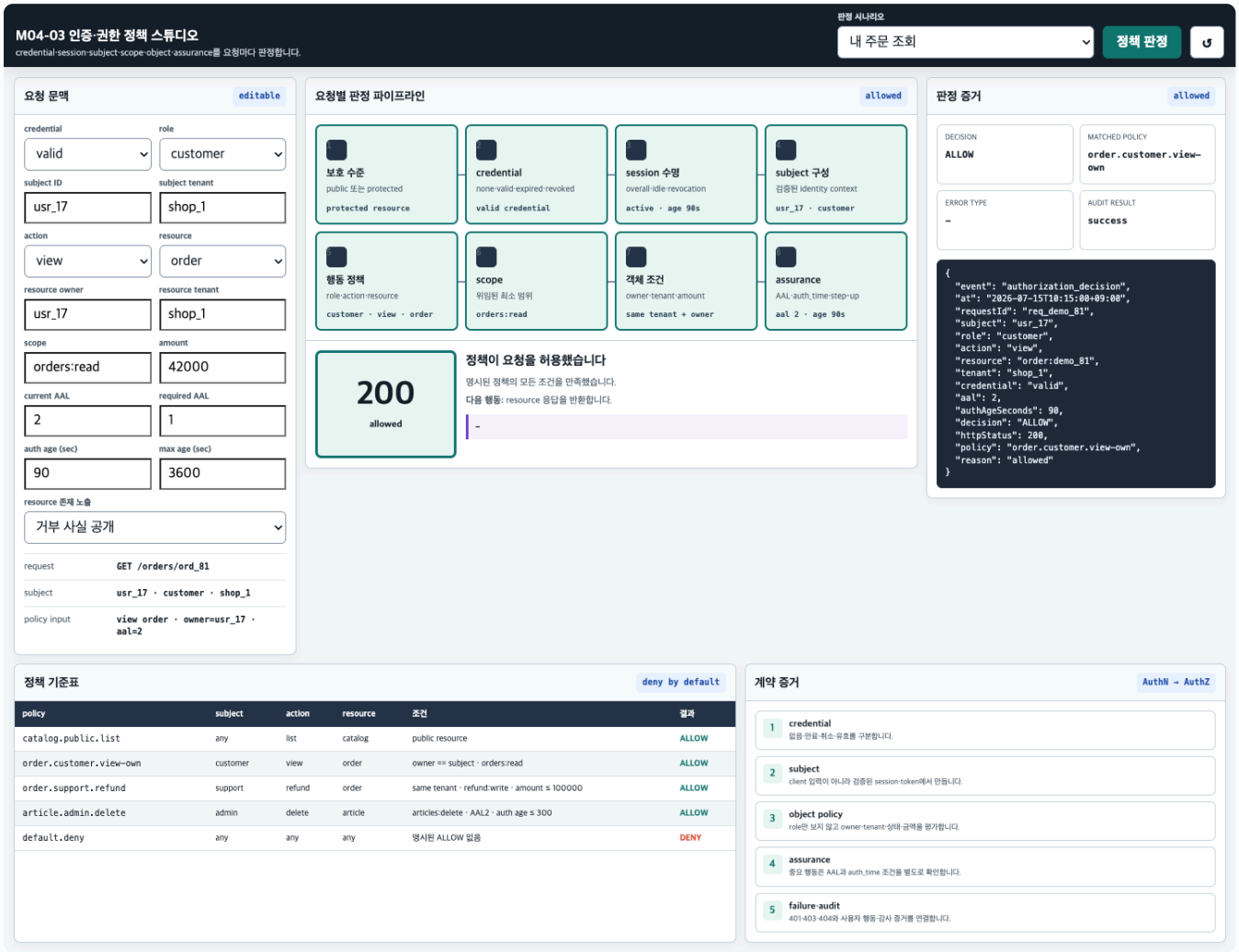


그림 12. 정책 스튜디오는 credential·session·subject·행동·scope·객체·assurance를 순서대로 판정하고 status·다음 행동·policy ID·감사 event를 함께 보여 줍니다.

17.1. 준비 파일

- 실습 안내
- 인증·권한 정책 스튜디오
- 인증·권한 정책 템플릿
- 로그인·인증·권한 용어집

17.2. 아홉 시나리오의 기대 결과

시나리오	status	outcome	핵심 gate
내 주문 조회	200	allowed	scope·owner 통과
인증 정보 없음	401	authentication_required	credential 없음
session 만료	401	reauthentication_required	expiry
session 취소	401	reauthentication_required	revocation
환불 scope 없음	403	insufficient_scope	scope 실패
다른 사용자의 비공개 주문	404	not_found_hidden	ownership·visibility
상담원 환불 한도 초과	403	object_forbidden	amount condition
관리자 삭제·오래된 인증	401	step_up_required	AAL·auth age
공개 catalog	200	allowed	public route

17.3. 실습 완료 증거

다음 네 장면을 기록합니다.

1. credential이 없는 요청의 401 challenge
2. 같은 role이지만 다른 owner인 요청의 403 또는 404
3. scope는 있지만 금액 한도를 넘은 요청의 403
4. role·scope가 모두 있지만 AAL·auth age가 부족한 요청의 step-up

17.4. 직접 바꿔 볼 값

`role`, `scope`, `resource owner`, `tenant`, `amount`, `current AAL`, `required AAL`, `auth age`, `max age`, 존재 노출 정책을 하나씩 바꿉니다. 여러 값을 한꺼번에 바꾸면 어떤 조건이 결과를 바꿨는지 알기 어렵습니다.

실습기의 한계
이 도구는 학습용 정책 판정기입니다. 실제 password·passkey·OIDC·token signature를 검증하지 않으며, 완전한 policy engine·OAuth authorization server·보안 시험 도구가 아닙니다. 실제 구현은 검증된 제품·library와 통합 테스트·침투 테스트로 확인합니다.

18. 실무 산출물 · 한 기능의 권한 계약을 완성합니다

대상 기능을 하나만 고릅니다.

```
추천 1 · 주문 환불
추천 2 · 게시물 공개·삭제
추천 3 · 조직 문서 export
추천 4 · 사용자 권한 변경
```

18.1. 한 문장 정책

```
support subject는 같은 tenant의 paid order에 대해 refund:write scope가 있고,
환불 금액이 100,000원 이하이며 AAL2 인증 후 10분 이내일 때만 refund할 수 있다.
```

18.2. 권한 매트릭스 한 줄

subject	action	resource	scope	object·context 조건	실패
support	refund	order	`refund:write`	same tenant·paid·amount≤100000·AAL2·age≤600	401 step-up / 403 deny

18.3. session·token 정책 한 줄

발급	idle	overall	rotate	revoke	reauth
AAL2 성공 후	30분	8시간	권한 상승·refresh 사용	logout·정지·분실	환불·계정 변경

18.4. 실패와 다음 행동

조건	status·type	사용자 행동	내부 증거
access token 만료	401 authentication-required	안전한 갱신·재로그인	issuer·token ID hash·reason
scope 부족	403 insufficient-scope	승인·동의 요청	required scope·client
다른 tenant	403 또는 404	접근 가능한 목록	policy ID·tenant mismatch

조건	status·type	사용자 행동	내부 증거
인증 오래됨	401 step-up-required	최근 AAL2 인증	required AAL·max age
한도 초과	403 amount-limit	상위 승인 이관	amount·limit·policy version

19. 한 장 요약

그림 13. 로그인부터 인증·세션·token·권한·실패·수명까지 한 장으로 복습합니다. 마지막 줄은 구현·검수의 공통 완료 기준입니다.

1

로그인

증명 시작 UX

2

인증

factor·assurance

5

권한

subject·action·object

6

조건

role·scope·owner·context

요청 판정 · valid credential/session → subject

완료 기준 = deny by default · server-side · ever

3

세션

continuity·secret

4

토큰

purpose·audience

7

실패

401·403·404

8

수명

expire·revoke·reauth

→ policy(subject, action, resource, context)

y request · object-level · lifecycle · audit · test

기억할 여덟 칸

1. 로그인은 인증을 시작하는 사용자 흐름입니다.
2. 인증은 authenticator 통제를 검증해 subject와 assurance를 만듭니다.
3. 세션은 인증 사건의 연속성을 session secret으로 이어 줍니다.
4. token은 목적·소비자·audience·수명·취소 경로로 구분합니다.
5. 권한은 subject·action·resource·context의 함수입니다.
6. role·scope·owner·tenant·attribute·relationship을 필요한 만큼 조합합니다.
7. 401·403·404·step-up은 원인과 다음 행동이 다릅니다.
8. 발급·사용·회전·만료·취소·재인증·감사를 수명주기로 닫습니다.

20. 셀프 테스트 · 정답을 가리고 풉니다

문제 1

로그인과 인증은 무엇이 다릅니까?

문제 2

identity proofing과 authentication은 무엇이 다릅니까?

문제 3

인증에 성공한 뒤 session이 필요한 이유와 session secret의 역할을 씁니다.

문제 4

server session cookie와 bearer access token을 “state 위치, 주 소비자, 취소” 세 축으로 비교합니다.

문제 5

ID token·access token·refresh token의 주 소비자를 각각 씁니다.

문제 6

“JWT payload를 decode할 수 있으므로 token이 위조됐다”와 “JWT이므로 안전한 session이다”가 왜 틀렸습니까?

문제 7

권한 함수의 네 입력을 쓰고 주문 환불 예를 한 줄로 만듭니다.

문제 8

role, scope, ownership이 각각 답하는 질문을 씁니다.

문제 9

버튼을 숨기고 route guard를 두었으면 server 권한 검사가 없어도 됩니까?

문제 10

401·403·404를 “원인과 다음 행동”으로 구분합니다.

문제 11

idle timeout, overall timeout, revocation은 무엇이 다릅니까?

문제 12

관리자 삭제에 step-up이 필요한 조건과 남겨야 할 감사 증거를 씁니다.

답안 메모

구분	내 답의 핵심어
증명	
연속성	
token 소비자	
정책 입력	
실패·다음 행동	
만료·취소·재인증	

21. 셀프 테스트 정답과 해설

정답 1

로그인은 identifier와 authenticator를 받아 인증을 시작·완료하는 사용자 흐름입니다. 인증은 제출된 authenticator output이 등록된 계정과 유효하게 연결되는지 검증하는 보안 사건입니다.

정답 2

identity proofing은 계정이 실제 사람·조직과 어떻게 연결되는지 확인하는 가입·발급 과정입니다. authentication은 이후 접속자가 등록된 authenticator를 통제하는지 검증합니다. 실명 확인이 없는 pseudonymous account도 인증할 수 있습니다.

정답 3

매 요청마다 password·passkey를 다시 내지 않고 인증된 상호작용을 이어 가기 위해 session을 사용합니다. session secret은 subscriber software와 session host를 묶어 다음 요청이 같은 session에 속함을 증명하거나 나타냅니다.

정답 4

server session은 권한·session 상태를 server store에 두고 web backend가 opaque ID로 조회하기 쉬우며 중앙 취소가 직접적입니다. bearer access token은 resource server가 token 또는 introspection을 검증하고 여러 API에 audience·scope를 전달할 수 있지만 탈취·수명·취소 전략이 중요합니다.

정답 5

ID token은 OpenID Connect client, access token은 resource server, refresh token은 authorization server의 token endpoint가 주 소비자입니다.

정답 6

JWT의 base64url payload는 암호화되지 않았다면 누구나 decode할 수 있지만 그것만으로 위조는 아닙니다. 서명·issuer·audience·expiry·용도를 검증해야 합니다. JWT는 claim 형식이지 session 수명·저장·취소·권한 정책을 자동으로 안전하게 만드는 방식이 아닙니다.

정답 7

`subject`, `action`, `resource`, `context` 입니다. 예: `support subject가 refund:write scope로 같은 tenant의 paid order를 100000원 이하, AAL2 인증 후 10분 이내에 refund` 합니다.

정답 8

role은 조직·서비스에서 어떤 직무 묶음인지, scope는 현재 client·grant에 어떤 API 권한 범위가 위임됐는지, ownership은 요청한 resource가 현재 subject의 것인지 답합니다.

정답 9

안 됩니다. 버튼 숨김과 route guard는 UX이며 우회 가능합니다. server·gateway·policy enforcement point에서 모든 보호 요청과 실제 객체에 대해 권한을 판정해야 합니다.

정답 10

401은 유효한 인증 정보가 없거나 필요한 인증 강도·최신성이 부족해 새 인증·갱신·step-up이 필요한 경우입니다. 403은 인증됐지만 role·scope·객체 정책이 거부해 다른 객체·승인·권한 요청이 필요합니다. 404는 실제 resource 없음 또는 존재 비공개 정책이며 일관되게 적용해야 합니다.

정답 11

idle timeout은 일정 시간 활동이 없어 종료되는 제한, overall timeout은 인증·재인증 뒤 session이 유지될 수 있는 절대 최대 시간, revocation은 만료 전이라도 logout·분실·계정 정지·위험 사건으로 즉시 무효화하는 사건입니다.

정답 12

예: `role=admin`, `articles:delete`, 같은 tenant를 만족해도 `AAL2`와 `auth_time≤300초`가 아니면 step-up합니다. request ID, subject, action, resource, policy ID·version, 현재·요구 AAL, auth age, decision, reason, status를 기록하되 secret·token 원문은 남기지 않습니다.

22. 최종 제출 체크리스트와 공식 근거

- [ ] 로그인·인증·세션·권한·감사를 별도 책임으로 정의했습니다.
- [ ] identity proofing과 반복 authentication을 구분했습니다.
- [ ] authenticator 등록·분실·복구·삭제 수명주기가 있습니다.
- [ ] MFA factor 독립성·피싱 저항·사용자 의도를 검토했습니다.
- [ ] session secret 발급·회전·idle·overall·logout·revocation을 정의했습니다.
- [ ] cookie의 Secure·HttpOnly·SameSite·scope와 CSRF 방어를 검토했습니다.
- [ ] ID·access·refresh token의 소비자와 목적을 구분했습니다.
- [ ] access token의 issuer·audience·expiry·scope·signature 또는 introspection을 검증합니다.
- [ ] JWT payload에 secret·불필요한 개인정보를 넣지 않습니다.

- [ ] subject·action·resource·context의 신뢰 출처가 분명합니다.
- [ ] role·scope·owner·tenant·상태·한도 조건을 권한 매트릭스에 적었습니다.
- [ ] 명시된 ALLOW가 없으면 deny by default입니다.
- [ ] client가 아니라 server에서 모든 요청·객체의 권한을 검사합니다.
- [ ] 401·403·404·step-up의 problem type과 다음 행동을 정의했습니다.
- [ ] 권한 변경·계정 정지의 전파 지연을 검토했습니다.
- [ ] 중요한 행동의 AAL·auth age·reauth 조건이 있습니다.
- [ ] 허용·다른 role·다른 owner·경계값·만료·취소 테스트가 있습니다.
- [ ] audit event에 policy ID·decision·reason이 있고 secret 원문은 없습니다.

이 교재는 다음 공식·권위 자료를 2026-07-15에 확인해 학습자용으로 재구성했습니다.

- NIST SP 800-63B-4 · Authentication and Authenticator Management: authenticator·AAL·session·reauthentication
- NIST SP 800-162 · Attribute Based Access Control: subject·object·operation·environment attribute 정책
- RFC 9110 · HTTP Semantics: 401·403·404와 WWW-Authenticate 의미
- RFC 6750 · Bearer Token Usage: bearer token·invalid_token·insufficient_scope
- RFC 9700 · OAuth 2.0 Security Best Current Practice: redirect·PKCE·token·client 보안 최신 권고
- OpenID Connect Core 1.0: identity layer·ID token·claims·nonce
- RFC 7519 · JSON Web Token: JWT·claim·JWS·JWE
- RFC 7009 · OAuth Token Revocation: access·refresh token 취소
- RFC 9470 · OAuth Step Up Authentication: insufficient_user_authentication·acr_values·max_age
- OWASP Authentication Cheat Sheet: 로그인·재인증·일반 인증 설계
- OWASP Session Management Cheat Sheet: cookie·session ID·fixation·renewal·risk event
- OWASP Authorization Cheat Sheet: least privilege·deny by default·every request·object policy
- OWASP ASVS 5.0.0: application authentication·session·access control 검증 기준

M04-03 완료
이제 로그인 화면을 보안 설계 전체로 착각하지 않습니다. 인증 사건이 session·token으로 어떻게 이어지고, 모든 보호 요청에서 subject·action·resource·context가 어떻게 판정되며, 만료·취소·401·403·404·재인증·감사가 어떤 사용자 행동과 테스트로 연결되는지 한 장의 권한 계약으로 설명할 수 있습니다.