M10-02 · GIBALJA VISUAL MANUAL

정상·예외·권한·보안 시나리오 검수하기

한 문장 목표: 성공 경로만 눌러 보는 검수를 넘어, 실패해도 안전한지와 거절되어야 할 요청이 정확히 거절되는지를 20개 시나리오와 증거로 판정합니다.

그림 1. 검수의 네 방향. 정상 4개, 예외·복구 5개, 권한 6개, 보안 5개를 함께 보아야 검수 범위의 한쪽이 비지 않습니다.

검증은 세 층으로 나눠 봅니다

성공 경로 하나가 아니라 정상·예외·권한·보안을 함께 봅니다.

01

정상

허용된 사용자가
목표를 완수하는가

REV
PORT

03

권한

주체·행동·자원 관계를
매 요청 검사하는가

02

예외·복구

실패해도 상태와
작용이 안전한가

IEW
FOLIO

04

보안

변조·노출·우회·경합에도
통제가 유지되는가

0. 한눈에 보는 두 번째 지도

그림 2. 행복 경로는 출발점입니다. 허용된 절차가 작동해도 예외·거절·권한·경합의 빈칸은 그대로 남습니다.

성공 절차가 작동해도 거절·복구·권한의 빈칸은 남아 있습니다.

A

행복 경로만 확인

생성 → 제출 → 승인
허용된 역할만 사용
오류·재시도 없음
6/20 PASS

EXPAND

B

검수 포트폴리오

허용과 기대 거절
실패 뒤 상태 불변
객체·기능 권한
20/20 PASS + Gate

난이도	그림 먼저	개념·판정	실습	셀프 테스트	최종 산출물
Level 2	35분	65분	85분	15분	시나리오 목록·증거·검수 결과서

정상 시나리오는 “해야 할 일이 된다”를 확인합니다. 품질 검수는 거기서 멈추지 않습니다. 잘못된 상태, 의존성 실패, 만료 세션, 다른 사용자의 문서, 관리자 전용 기능, 변조된 속성, 잘못된 입력, 동시 요청에서도 상태와 권한이 지켜져야 합니다. 이 매뉴얼은 공격 기법을 실행하는 안내서가 아니라, 합성 시스템의 방어 계약을 안전하게 확인하고 증거로 남기는 학습서입니다.

0.1 첫 두 장에서 기억할 여덟 문장

```
행복 경로는 검수 포트폴리오의 한 부분이다.
시나리오는 행위자·자원 관계·시작 상태·행동·기대를 함께 가진다.
실패가 안전하려면 상태·수량·부작용·노출 불변식이 유지되어야 한다.
인증은 신원을, 권한은 특정 자원과 기능의 허용을 판정한다.
로그인한 사용자도 다른 사람의 객체와 관리자 기능에는 거절될 수 있다.
기대 거절은 실패가 아니라 통제가 작동한 PASS다.
통제·시나리오·실행 결과·결함은 양방향으로 추적한다.
Release Gate는 전체·Critical·DENY·범주·통제 기준을 모두 만족해야 열린다.
```

1. 이 PDF를 공부하는 방법

1.1 1회차 · 그림 16장만 읽기 · 35분

그림 제목과 캡션을 먼저 읽습니다. 다음 흐름을 말할 수 있으면 됩니다.

```
통제 → 행위자·행동·자원·맥락
→ 정상·예외·권한·보안 시나리오
→ 기대 허용·기대 거절·안전 실패
→ 실제 결과·불변식·증거
→ 결함·재검·회귀 → Release Gate
```

1.2 2회차 · 시나리오 검수 스튜디오 · 50분

실습 생성기를 실행합니다.

```
./02_Labs/G10_Test_Quality_Security/L10-02_create-scenario-review-practice.sh
```

세 버전을 순서대로 실행합니다.

```
happy-path-only-v1
  → partial-guards-v2
  → review-candidate-v3
```

첫 버전은 6/20만 통과합니다. 두 번째는 13/20으로 올라가지만 객체·기능 권한과 안전 처리 결함 7개가 남습니다. 세 번째는 20/20과 5/5 회귀 계약을 만족해 `release_ready`가 됩니다.

1.3 3회차 · 내 기능에 적용 · 115분

단계별 실습서를 따라 정상·예외·권한·보안 검수 결과서를 채웁니다. 낯선 표현은 시나리오·권한·보안 검수 용어집에서 찾습니다.

1.4 손의 순서를 고정합니다

먼저 하지 않을 것	먼저 할 것
화면 클릭 목록부터 작성	통제·행위자·자원·상태 목록 고정
정상 시나리오만 통과 확인	기대 거절·예외·복구·경합 추가
역할 이름만 비교	actor·action·resource·context 비교
오류 메시지만 확인	상태·수량·부작용·노출 불변식 비교
20개 중 19개 통과로 승인	Critical·DENY·통제·범주 Gate 판정
FAIL 화면만 결함으로 전달	version·scenario·mismatch·evidence 묶기

2. 학습 범위와 안전 경계를 고정합니다

2.1 이번 실습의 합성 시스템

```
service: synthetic_document_approval
controls: 10
actors: 6
access_rules: 6
scenarios: 20
categories:
  normal: 4
  exception-recovery: 5
  authorization: 6
  security: 5
variants:
  happy-path-only-v1: 6/20
  partial-guards-v2: 13/20
  review-candidate-v3: 20/20
regression_contracts: 5/5
```

외부 네트워크·API 키·실제 문서·고객 데이터·공격 실행·실비용이 없습니다. 서버는 loopback 주소에서만 열리고, trace에는 실행 ID·버전·개수·판정만 남습니다.

2.2 이전·다음 매뉴얼과의 경계

연결	여기서 가져오는 것	이번 매뉴얼이 더하는 것
M10-01 요구사항에서 테스트 케이스 만들기	사전 조건·행동·기대·oracle·증거	네 범주의 시나리오 포트폴리오와 기대 거절
M10-03 개인정보와 비밀정보를 안전하게 다루기	다음 단계	여기서는 합성 입력·오류·감사 증거의 기본 안전 계약만 확인
M10-04 AI 환각·프롬프트 주입·정보 유출 검수	다음 단계	여기서는 일반 웹·API 흐름의 권한과 안전 실패에 집중
M12 요구사항·설계	이후 연결	검수에서 발견한 정책 빈칸을 요구사항으로 되돌려 보냄

이번 매뉴얼은 침투 테스트, 취약점 인증, 법률 적합성, 개인정보 영향평가, 운영 인프라 보안 검토를 완료했다고 주장하지 않습니다. 합성 시나리오의 방어 계약과 릴리스 증거를 학습하는 범위입니다.

2.3 시나리오 테스트가 증명하는 것과 증명하지 않는 것

증명 가능한 주장	이 검수만으로 증명하지 못하는 주장
명시한 actor·resource·state에서 기대가 충족됨	모든 가능한 사용자·입력·운영 조건에서 결함이 없음
기대 거절과 안전 실패가 field 단위로 확인됨	알려지지 않은 공격과 취약점이 모두 차단됨
특정 contract version에서 재현됨	다른 배포·환경·인프라에서도 동일함
통제와 시나리오가 추적됨	조직 전체의 보안 거버넌스가 적합함

3. 검수 포트폴리오를 네 범주로 균형 잡습니다

3.1 정상 시나리오

정상 시나리오는 허용된 행위자가 올바른 상태에서 목표를 완수하는지 봅니다. “버튼이 눌린다”가 아니라 상태전이·결과 code·수량·감사 증거까지 확인합니다.

실습 ID	정상 흐름	핵심 기대
SC-01	소유자가 초안 생성	201·DRAFT·record 1·audit 1
SC-02	검토자가 있는 초안 제출	SUBMITTED·submission 1
SC-03	배정 검토자가 승인	APPROVED·approval 1
SC-04	소유자가 승인 결과 조회	visible true·APPROVED

3.2 예외·복구 시나리오

예외 시나리오는 잘못된 값과 상태를 거절하는 데서 끝나지 않습니다. 실패한 요청이 상태와 부작용을 만들지 않았는지, 재시도가 중복 반영되지 않는지 확인합니다.

실습 ID	예외·복구	안전 조건
SC-05	빈 제목 생성	record 0·audit 0
SC-06	검토자 없는 제출	DRAFT 유지·submission 0
SC-07	감사 의존성 timeout	SUBMITTED 유지·approval 0

실습 ID	예외·복구	안전 조건
SC-08	같은 제출 재시도	submission·audit 정확히 1
SC-09	DRAFT 직접 승인	INVALID_STATE·DRAFT 유지

3.3 권한 시나리오

권한 시나리오는 같은 기능의 허용과 거절을 쌍으로 봅니다. 소유자, 다른 소유자, 배정 검토자, 미배정 검토자의 차이를 자원 관계로 설명해야 합니다.

실습 ID	권한 질문	기대
SC-10	비로그인 사용자가 보호 문서를 보는가	401·visible false
SC-11	소유자가 자기 문서를 보는가	200·visible true
SC-12	다른 사용자가 object ID를 바꿔 보는가	404·visible false
SC-13	미배정 검토자가 승인하는가	403·approval 0
SC-14	배정 검토자가 승인하는가	200·approval 1
SC-15	소유자가 관리자 export를 호출하는가	403·export 0

3.4 보안 시나리오

이 장의 보안 시나리오는 안전한 합성 계약 검수입니다. 역할 속성 변조, 만료 세션, 표시 값 인코딩, 일반화 오류, 동시 승인 불변식을 확인합니다.

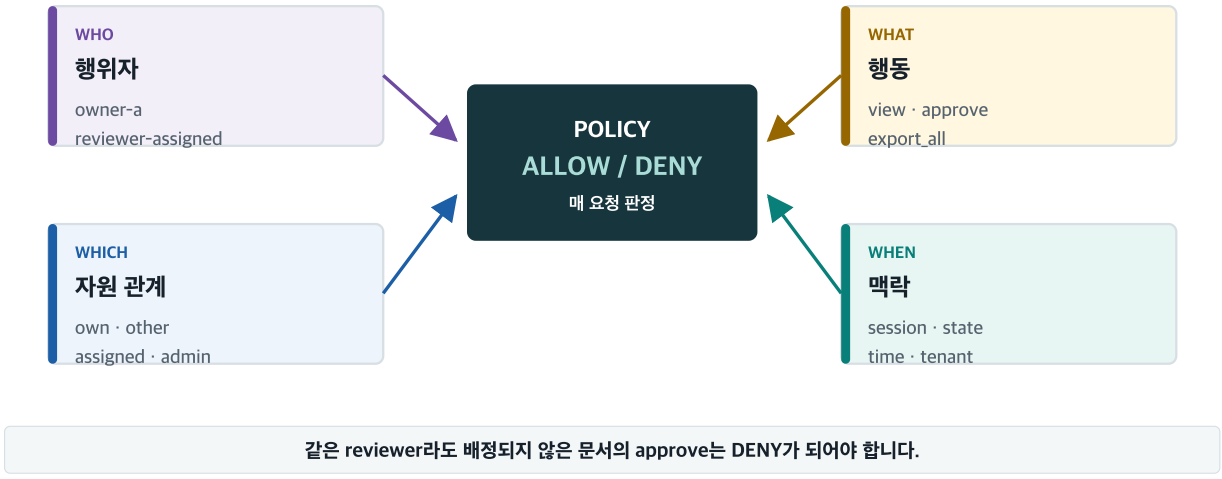
실습 ID	보안 질문	기대
SC-16	입력에 `role=admin`을 넣으면 승격되는가	무시·OWNER 유지
SC-17	만료 세션이 보호 자원에 도달하는가	401·visible false
SC-18	제목 markup이 실행되는가	인코딩된 문자열·executable false
SC-19	잘못된 요청이 내부 상세를 노출하는가	400·internal detail false
SC-20	동시 승인이 중복 부작용을 만드는가	approval·audit 정확히 1

4. 권한 판정은 네 요소의 관계입니다

M10-02 · 03

권한은 역할 이름 하나가 아니라 네 요소의 관계입니다

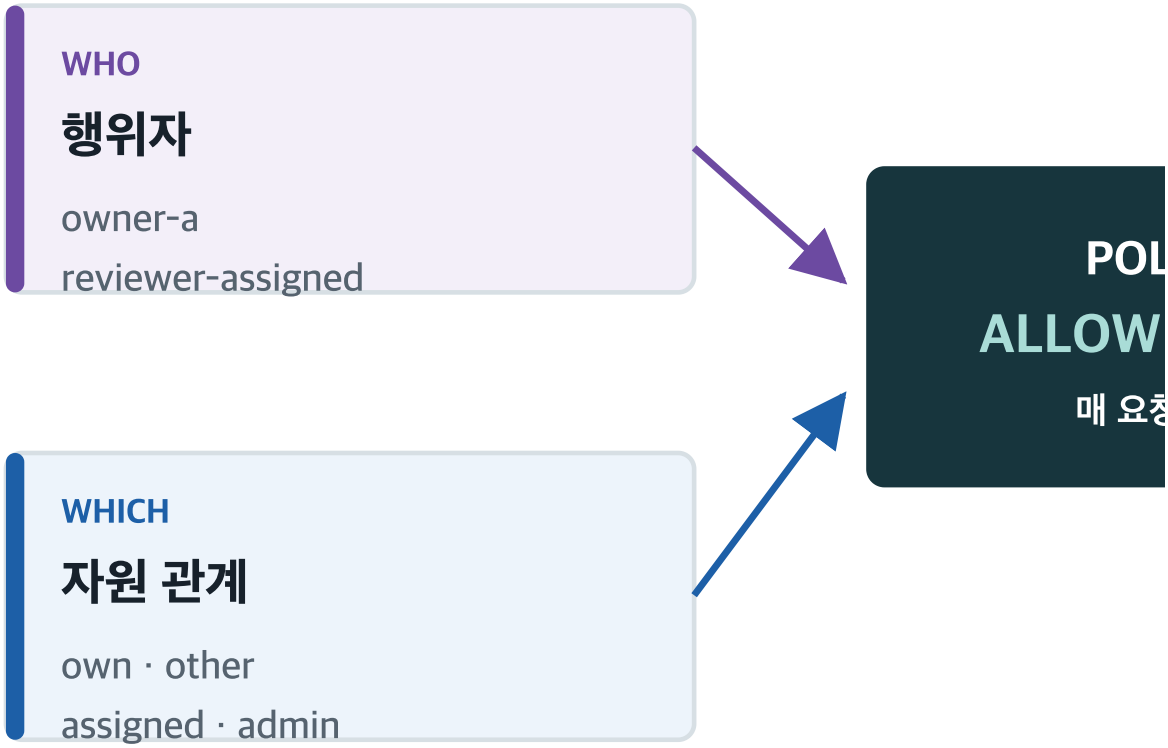
누가·무엇을·어떤 자원에·어떤 맥락에서 하는지 함께 판정합니다.

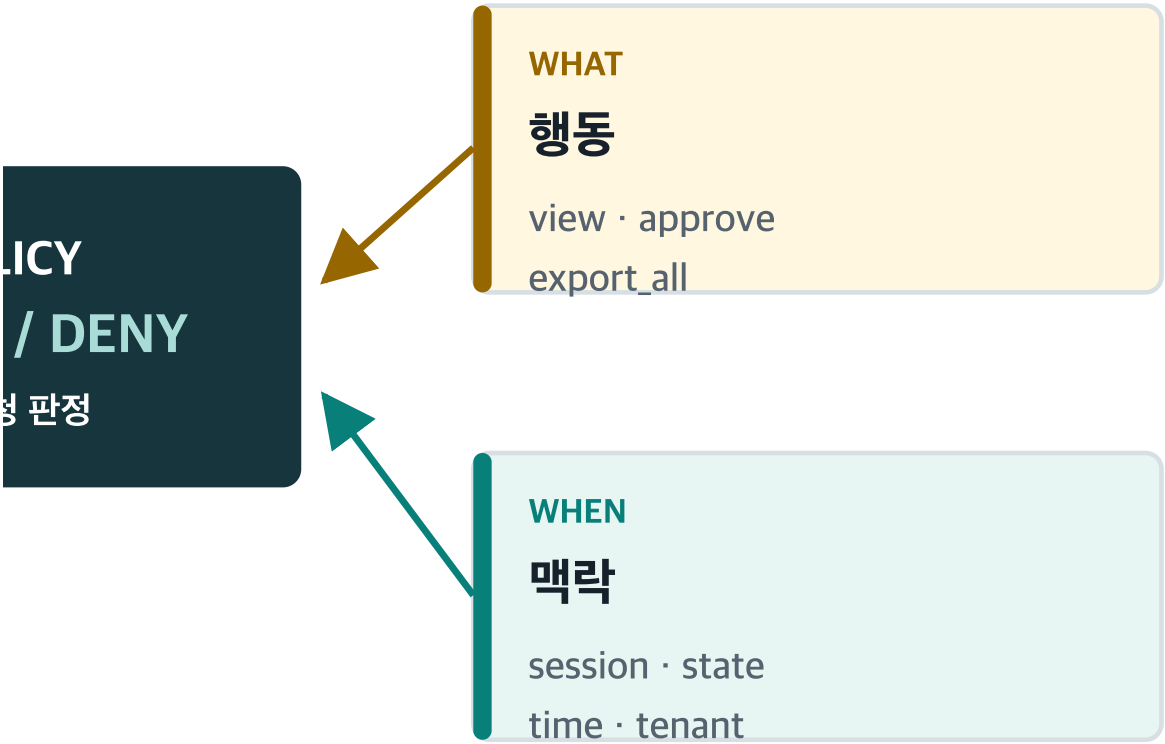


같은 reviewer라도 배정되지 않은 문서의 approve는 DENY가 되어야 합니다.

그림 3. 권한은 역할 이름 하나가 아닙니다. 같은 검토자도 배정 관계와 문서 상태가 다르면 판정이 달라집니다.

누가·무엇을·어떤 자원에·어떤 맥락에서 하는지 함께 판정합니다.




LICY
/ DENY
형 판정
WHAT
행동
view · approve
export_all
WHEN
맥락
session · state
time · tenant

4.1 네 요소

요소	질문	실습 예
Actor · 행위자	누가 요청하는가	owner-a·reviewer-unassigned
Action · 행동	무엇을 하려는가	view·approve·export_all
Resource · 자원	어느 객체·기능인가	own-document·admin-export
Context · 맥락	어떤 세션·상태·배정인가	session valid·SUBMITTED·assigned false

역할 기반 접근 제어(Role-Based Access Control, RBAC)는 유용하지만 역할만으로 자원 소유·배정·tenant·상태를 모두 표현하기 어렵습니다. 시나리오에는 역할과 함께 관계·속성·맥락을 적습니다.

4.2 정책 문장을 판정식으로 바꿉니다

```
allow(actor, action, resource, context)
  = authenticated(actor)
  AND action_is_allowed(actor.role, action)
  AND relation_is_valid(actor, resource)
  AND state_allows(resource.state, action)
  AND context_is_valid(context)
```

하나라도 거짓이면 기본 판정은 DENY입니다. 허용되지 않은 조합이 우연히 통과하는 것을 막기 위해 deny by default를 사용합니다.

4.3 “기대 거절”은 성공한 테스트입니다

요청	제품 동작	테스트 판정
소유자가 자기 문서 조회	허용	기대와 같으면 PASS
다른 사용자가 문서 ID 변조	거절	기대와 같으면 PASS
미배정 검토자가 승인	거절	기대와 같으면 PASS
만료 세션이 조회	거절	기대와 같으면 PASS

HTTP 401·403·404가 나왔다고 테스트 실패가 아닙니다. 시나리오의 expected와 actual이 같으면 통제가 작동한 PASS입니다.

5. 재현 가능한 시나리오 카드를 만듭니다

그림 4. 클릭 순서보다 행위자·자원 관계·시작 상태·관찰 가능한 기대·판정 증거가 완전한지가 중요합니다.

행동 절차만 적지 않고 자원 관계·상태·oracle을 함께 고정합니다.

WHO

행위자·자원 관계

owner-b · other-document

GIVEN

시작 상태

session valid · SUBM

THEN

기대

404 · NOT_FOUND

ORACLE

판정 기준

visible false · audit


MITTED
1
WHEN
행동
view doc-owner-a
EVIDENCE
실행 증거
before/after · actual

5.1 최소 필드

필드	목적	질문
ID·version	변경과 실행 추적	어떤 계약 버전의 시나리오인가
Control links	근거 연결	어떤 제품·보안·신뢰성 통제를 확인하는가
Actor·resource relation	권한 조건	누가 어느 자원과 어떤 관계인가
Preconditions	재현 시작점	세션·상태·배정·의존성은 무엇인가
Action	한 사건	어떤 요청을 몇 번 보내는가
Expected	관찰 결과	HTTP·code·state·visibility·count는 무엇인가
Oracle	판정 기준	어떤 field를 비교하는가
Evidence	재현 근거	요청 요약·응답·전후 상태·audit 수는 무엇인가
Risk·priority·owner	처리 순서	실패 영향과 담당자는 누구인가

5.2 SC-12를 카드로 읽습니다

```
ID: SC-12
controls: CTRL-06 객체 수준 권한
actor: owner-b
resource_relation: other-document
given:
  session: valid
  state: SUBMITTED
  owned: false
when:
  type: view
  document_id: doc-owner-a
then:
  http: 404
  code: NOT_FOUND
  visible: false
  audit_count: 1
```

“다른 문서를 못 본다”보다 위 카드가 강합니다. 어떤 관계와 상태에서 어떤 ID를 사용했고, 무엇을 보지 못해야 하며, 감사 증거가 몇 건이어야 하는지 재현할 수 있기 때문입니다.

5.3 한 시나리오에는 한 핵심 위험을 둡니다

여러 위험을 한 시나리오에 몰아넣으면 FAIL 원인이 흐려집니다. 예를 들어 만료 세션, 다른 사용자 문서, 잘못된 상태를 동시에 넣지 않습니다. 각각을 분리한 뒤 필요한 경우 조합 시나리오를 별도로 추가합니다.

6. 정상 흐름도 상태와 증거로 검수합니다

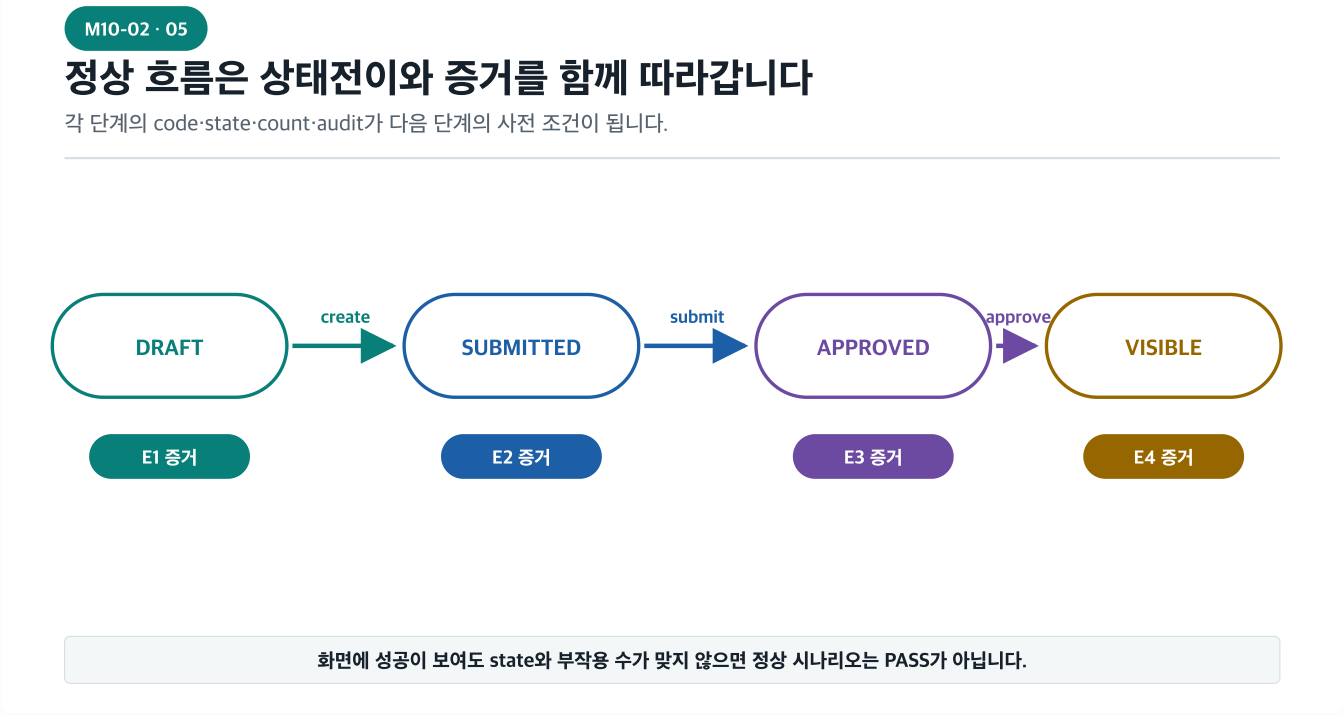
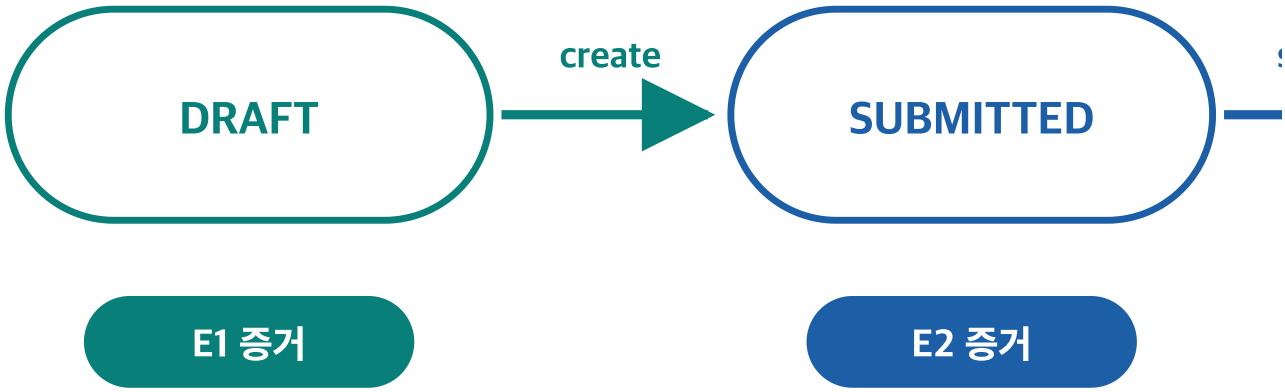
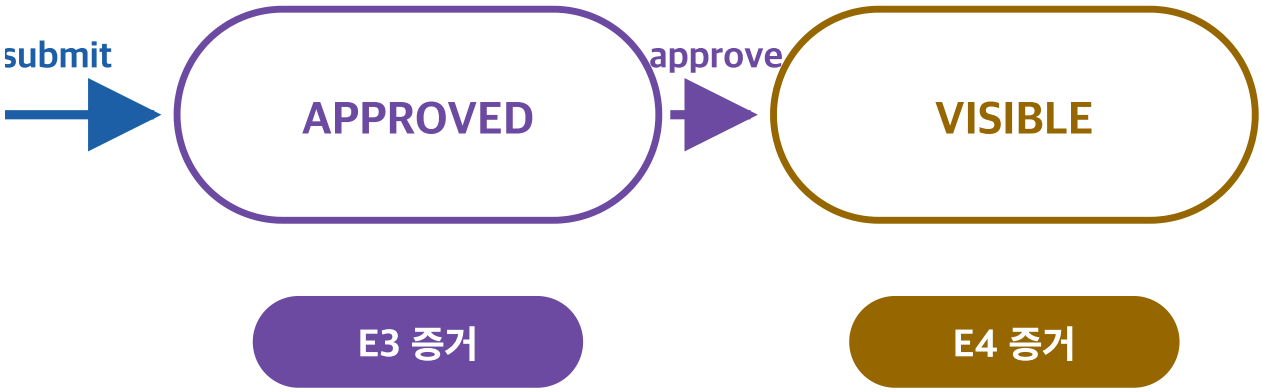


그림 5. 정상 시나리오는 화면 성공만 확인하지 않습니다. 상태와 부작용 수가 다음 단계의 사전 조건으로 맞아야 합니다.

각 단계의 code·state·count·audit가 다음 단계의 사전 조건이 됩니다.




submit
APPROVED
approve
VISIBLE
E3 증거
E4 증거

6.1 정상 흐름의 관찰 층

층	예	놓치기 쉬운 결함
Transport	HTTP 200·201	잘못된 상태인데 200 반환
Business code	SUBMITTED·APPROVED	화면 문구와 실제 code 불일치
State	DRAFT→SUBMITTED→APPROVED	금지 전이를 건너뜀
Data count	record·submission·approval	중복 저장·중복 승인
Visibility	visible true·false	다른 사용자 데이터 노출
Audit	audit_count·event	누락·중복 감사 증거

6.2 상태전이는 허용 경로와 금지 경로를 짝지어 봅니다

시작 상태	행동	기대 상태	시나리오
NONE	create	DRAFT	SC-01
DRAFT	submit	SUBMITTED	SC-02
SUBMITTED	approve	APPROVED	SC-03·SC-14
DRAFT	approve	DRAFT 유지·409	SC-09

정상 전이만 있으면 DRAFT에서 직접 APPROVED로 가는 결함을 놓칩니다. 허용 전이마다 대표 금지 전이를 최소 한 개 연결합니다.

6.3 정상 시나리오 리뷰 질문

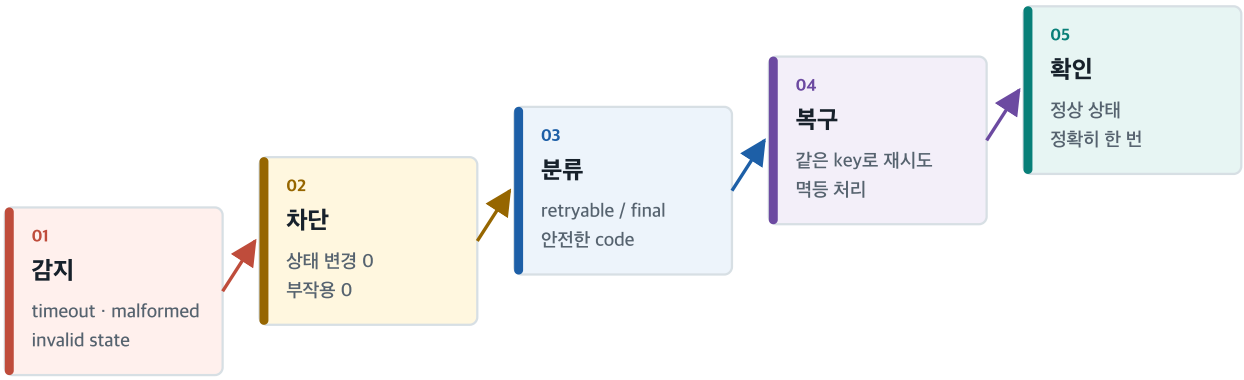
- 시작 상태를 외부 증거로 확인할 수 있는가.
- 한 행동이 한 상태전이를 일으키는가.
- 성공 code와 저장된 state가 함께 맞는가.
- 성공 부작용 수가 정확히 한 개인가.
- 다음 시나리오가 이전 결과에 숨게 의존하지 않는가.

7. 예외는 감지에서 복구 확인까지 검수합니다

M10-02 · 06

예외 검수는 실패에서 복구까지 사다리를 오릅니다

오류를 잡는 것만으로 끝내지 않고 안전한 상태와 재시도 결과를 확인합니다.

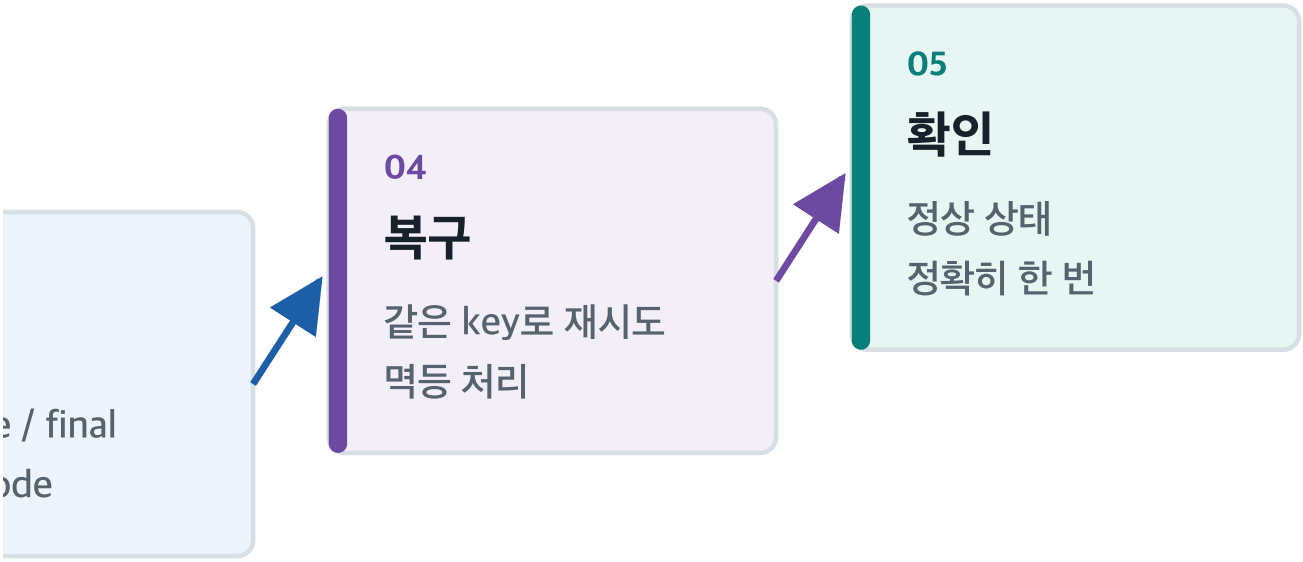


SC-07은 timeout 뒤 SUBMITTED를 유지하고, SC-08은 재시도에도 제출을 한 번만 반영합니다.

그림 6. 오류를 잡았다는 사실만으로는 부족합니다. 상태를 지키고, 재시도 가능성을 분류하고, 복구 뒤 정확히 한 번 반영됐는지 확인합니다.

[illegible]

오류를 잡는 것만으로 끝내지 않고 안전한 상태와 재시도 결과를 확인합


e / final
ode
04
복구
같은 key로 재시도
멱등 처리
05
확인
정상 상태
정확히 한 번

7.1 예외 검수의 다섯 단계

1. **감지:** validation·invalid state·timeout·malformed request를 식별합니다.
2. **차단:** 저장·전이·부작용이 일어나지 않게 합니다.
3. **분류:** 다시 시도할 수 있는 실패와 최종 실패를 구분합니다.
4. **복구:** 같은 요청을 안전하게 다시 시도하거나 사용자가 수정합니다.
5. **확인:** 정상 상태와 정확히 한 번의 부작용을 확인합니다.

7.2 오류 code만 비교하지 않습니다

```
expected:
  http: 503
  code: RETRYABLE_DEPENDENCY
  state: SUBMITTED
  approval_count: 0
  audit_count: 0
  internal_detail: false
```

위 계약에서 503만 맞고 state가 APPROVED라면 FAIL입니다. 오류를 반환하면서 작업을 반영하는 부분 실패가 더 위험할 수 있습니다.

7.3 재시도 시나리오의 핵심 질문

질문	확인 값
같은 요청임을 무엇으로 식별하는가	idempotency key·operation ID
첫 시도가 반영됐는가	state·count·event
재시도 결과는 무엇인가	같은 성공 또는 ALREADY_*
부작용은 몇 번인가	submission·approval·audit 정확히 1
다른 payload에 같은 key를 쓰면	충돌 거절·기록

7.4 안전 실패 불변식

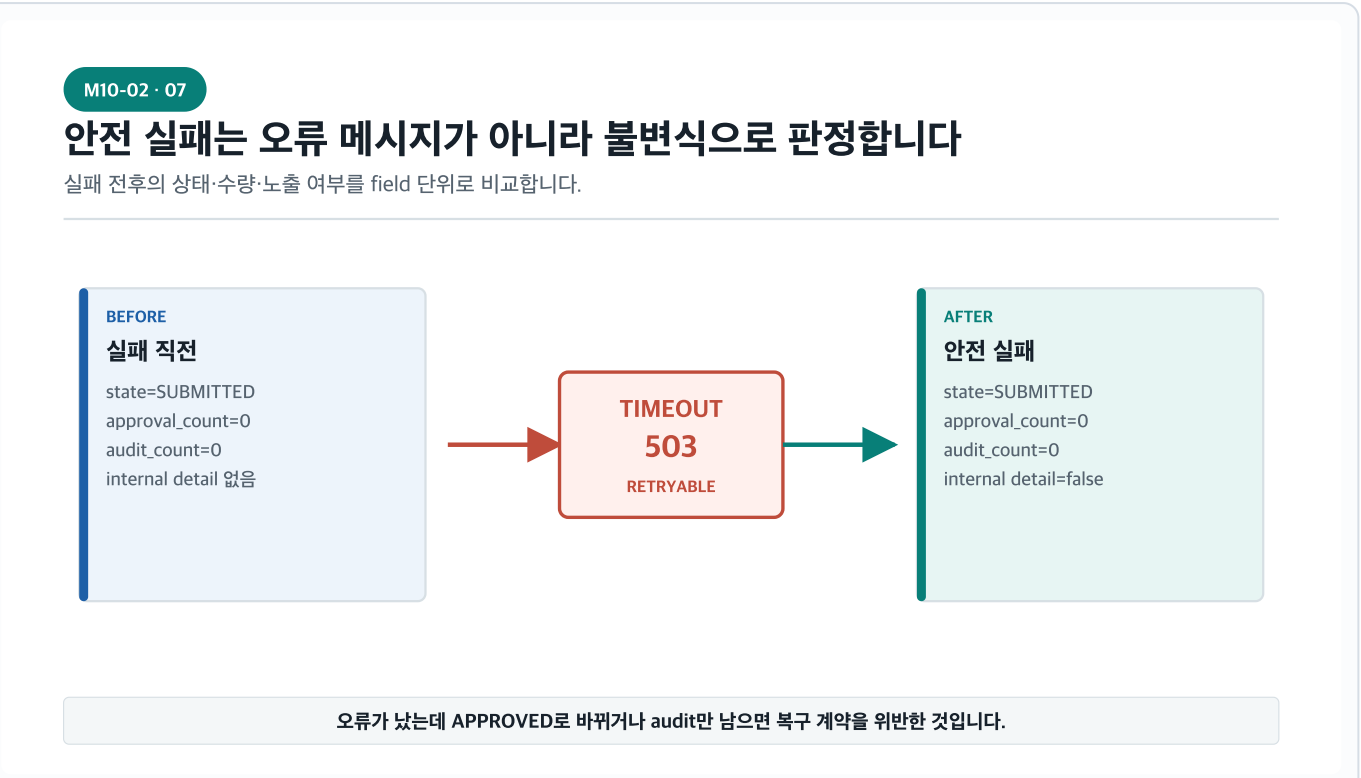


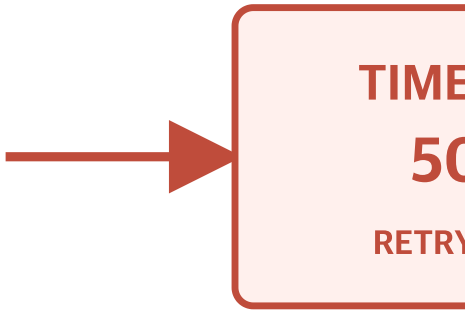
그림 7. 안전 실패는 메시지의 친절함이 아니라 전후 불변식으로 판정합니다.

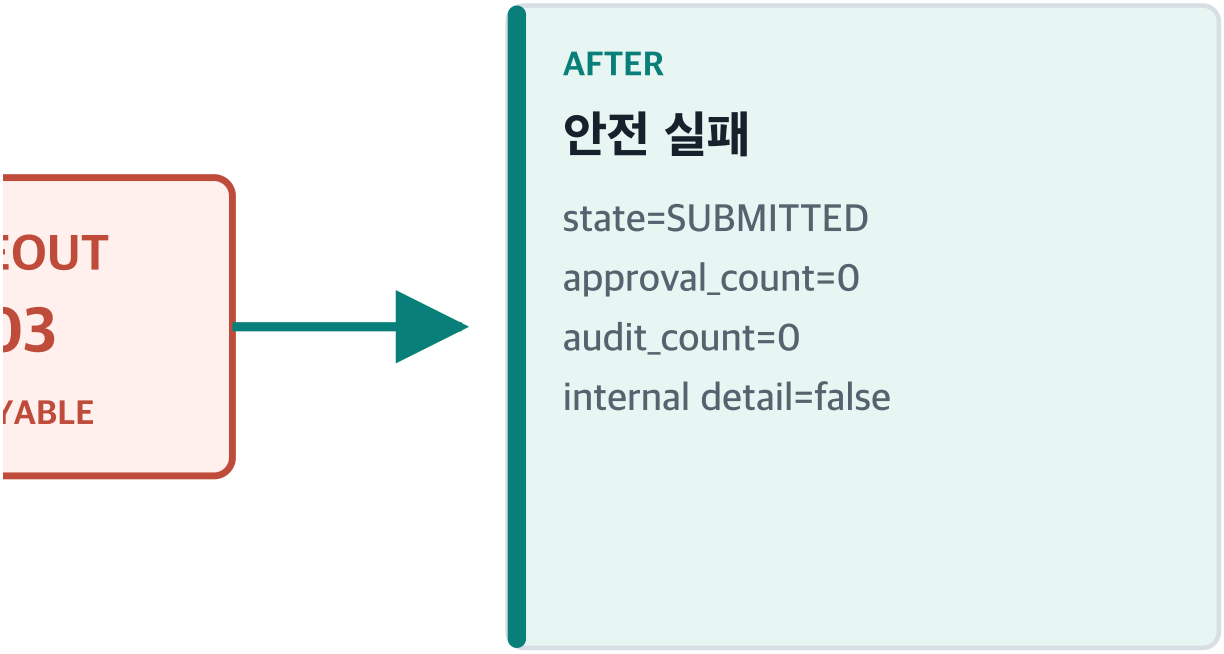
실패 전후의 상태·수량·노출 여부를 field 단위로 비교합니다.

BEFORE

실패 직전

state=SUBMITTED
approval_count=0
audit_count=0
internal detail 없음




EOUT
03
YABLE
AFTER
안전 실패
state=SUBMITTED
approval_count=0
audit_count=0
internal detail=false

실패 전후에 다음을 비교합니다.

```
state_after == safe_expected_state
record_delta == 0
approval_delta == 0
duplicate_effect == 0
unauthorized_visibility == false
internal_detail_exposed == false
```

이 불변식을 시나리오 expected에 적지 않으면, 테스트는 오류 문구만 보고 데이터 손상을 통과시킬 수 있습니다.

8. 접근 제어표로 기대 허용과 기대 거절을 만듭니다

M10-02 · 08

접근 제어표는 허용보다 기대 거절을 먼저 보이게 합니다

행위자·자원 관계·행동의 조합마다 ALLOW 또는 DENY를 명시합니다.

행위자 / 자원·행동	own view	other view	assigned approve	unassigned approve	admin export
owner-a	ALLOW	DENY	DENY	DENY	DENY
owner-b	ALLOW	DENY	DENY	DENY	DENY
reviewer-assigned	DENY	DENY	ALLOW	DENY	DENY
reviewer-unassigned	DENY	DENY	DENY	DENY	DENY
admin	ALLOW	ALLOW	ALLOW	ALLOW	ALLOW

표의 빈칸은 구현이 아니라 정책의 빈칸이며, 모든 DENY 셀은 검수 시나리오 후보입니다.

그림 8. 표의 빈칸은 구현의 빈칸이 아니라 정책의 빈칸입니다. 모든 DENY 셀은 검수 시나리오 후보입니다.

행위자·자원 관계·행동의 조합마다 ALLOW 또는 DENY를 명시합니다.

행위자 / 자원·행동	own view	other view
owner-a	ALLOW	DENY
owner-b	ALLOW	DENY
reviewer-assigned	DENY	DENY
reviewer-unassigned	DENY	DENY
admin	ALLOW	ALLOW

	assigned approve	unassigned approve	admin export
	DENY	DENY	DENY
	DENY	DENY	DENY
	ALLOW	DENY	DENY
	DENY	DENY	DENY
	ALLOW	ALLOW	ALLOW

8.1 표를 만드는 순서

1. 행에 실제 행위자 유형을 적습니다.
2. 열에 자원 관계와 행동을 함께 적습니다.
3. 각 셀을 ALLOW·DENY·NOT APPLICABLE 중 하나로 결정합니다.
4. 결정 근거와 정책 owner를 연결합니다.
5. ALLOW 대표와 DENY 대표를 시나리오로 만듭니다.
6. 고위험 기능과 객체는 모든 관계 조합을 우선 검수합니다.

8.2 `404`와 `403`을 구분하는 이유

권한이 없는 객체의 존재 자체를 숨기기 위해 `404 NOT_FOUND`를 사용할 수 있습니다. 기능 접근은 `403 FORBIDDEN`으로 명확히 거절할 수 있습니다. 어느 code가 정답인지는 제품·보안 정책이 결정합니다. 중요한 것은 같은 상황에 일관된 정책을 적용하고, 응답 body와 시간 차이로 존재 여부를 새지 않게 하는 것입니다.

8.3 접근 제어표 리뷰 질문

- anonymous를 포함했는가.
- 소유자와 다른 소유자를 분리했는가.
- 배정된 역할과 미배정 역할을 분리했는가.
- 관리자 전용 기능을 별도 열로 두었는가.
- 상태·tenant·시간처럼 판정을 바꾸는 맥락을 기록했는가.
- DENY 셀이 테스트 없이 남아 있지 않은가.

9. 인증과 권한을 두 개의 Gate로 분리합니다

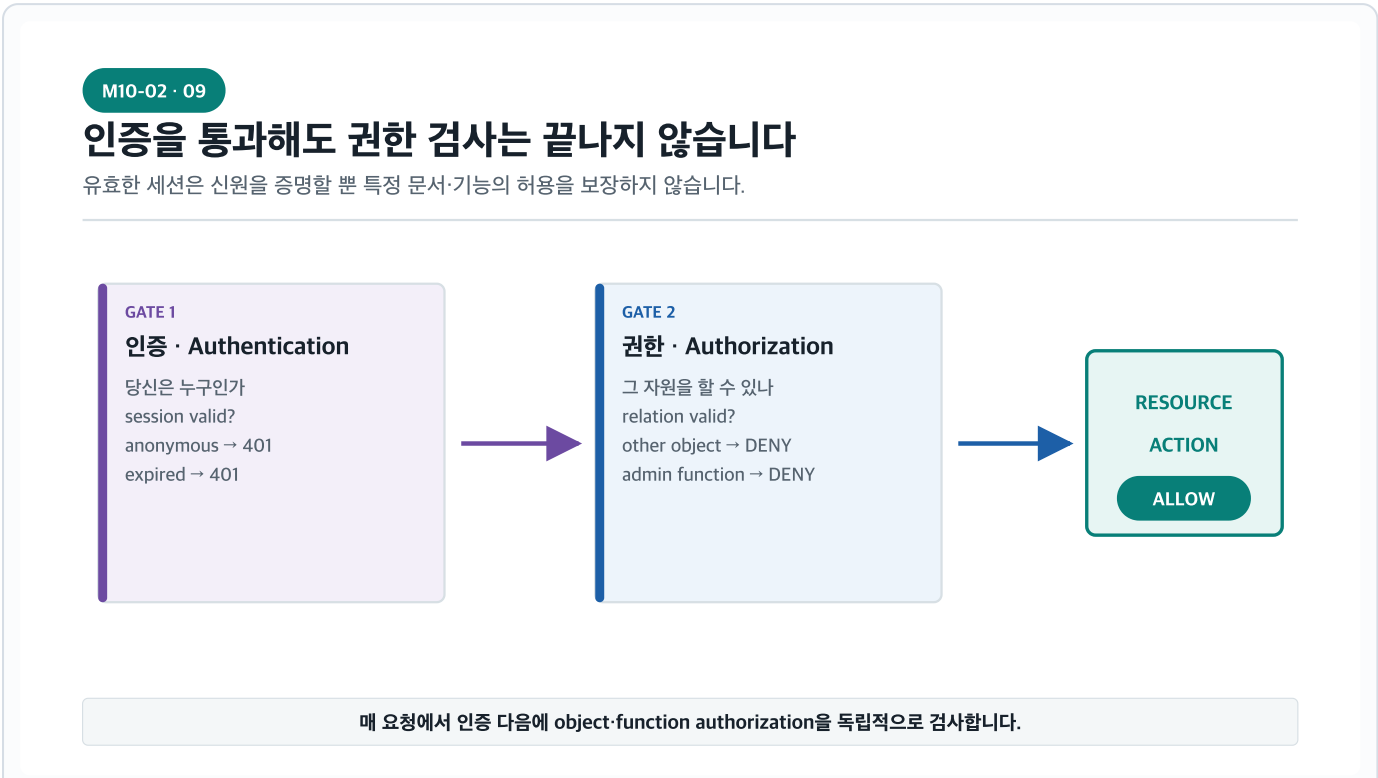


그림 9. 유효한 세션은 신원을 증명할 뿐, 특정 문서나 관리자 기능의 허용을 보장하지 않습니다.

유효한 세션은 신원을 증명할 뿐 특정 문서·기능의 허용을 보장하지 않

GATE 1

인증 · Authentication

당신은 누구인가
session valid?
anonymous → 401
expired → 401

GATE 2

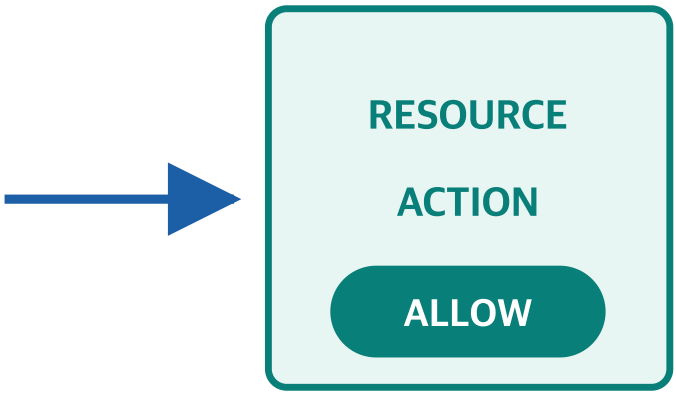
권한 ·

그 자원을
relation
other ob
admin fu

습니다.

Authorization

할 수 있나
valid?
ject → DENY
unction → DENY



9.1 인증 시나리오

상태	기대	실습
session missing	보호 자원 도달 전 401	SC-10
session expired	401·visible false	SC-17
session valid	권한 검사로 진행	SC-11~SC-16

인증 시나리오는 로그인 화면만 보지 않습니다. 세션 만료·취소·갱신·동시 사용·보호 자원 직접 요청을 포함합니다.

9.2 권한 시나리오

권한은 인증 뒤 매 요청에서 다시 검사합니다. 프런트엔드에서 버튼을 숨겼더라도 서버 endpoint가 요청을 거절해야 합니다. 클라이언트가 전송한 role·owner·tenant 값을 권한 근거로 믿지 않습니다.

9.3 최소 권한과 deny by default

원칙	시나리오로 바꾸는 질문
최소 권한	목표 수행에 필요하지 않은 기능도 허용되는가
Deny by default	정책에 없는 새 기능이 기본 허용되는가
매 요청 검사	처음 조회 후 다른 객체로 ID를 바꾸면 다시 검사하는가
서버 권위	숨은 field에 role을 넣으면 권한이 바뀌는가
안전 종료	세션 만료·로그아웃 뒤 토큰이 계속 작동하는가

10. 객체 수준 권한은 actor·action·object 관계를 봅니다

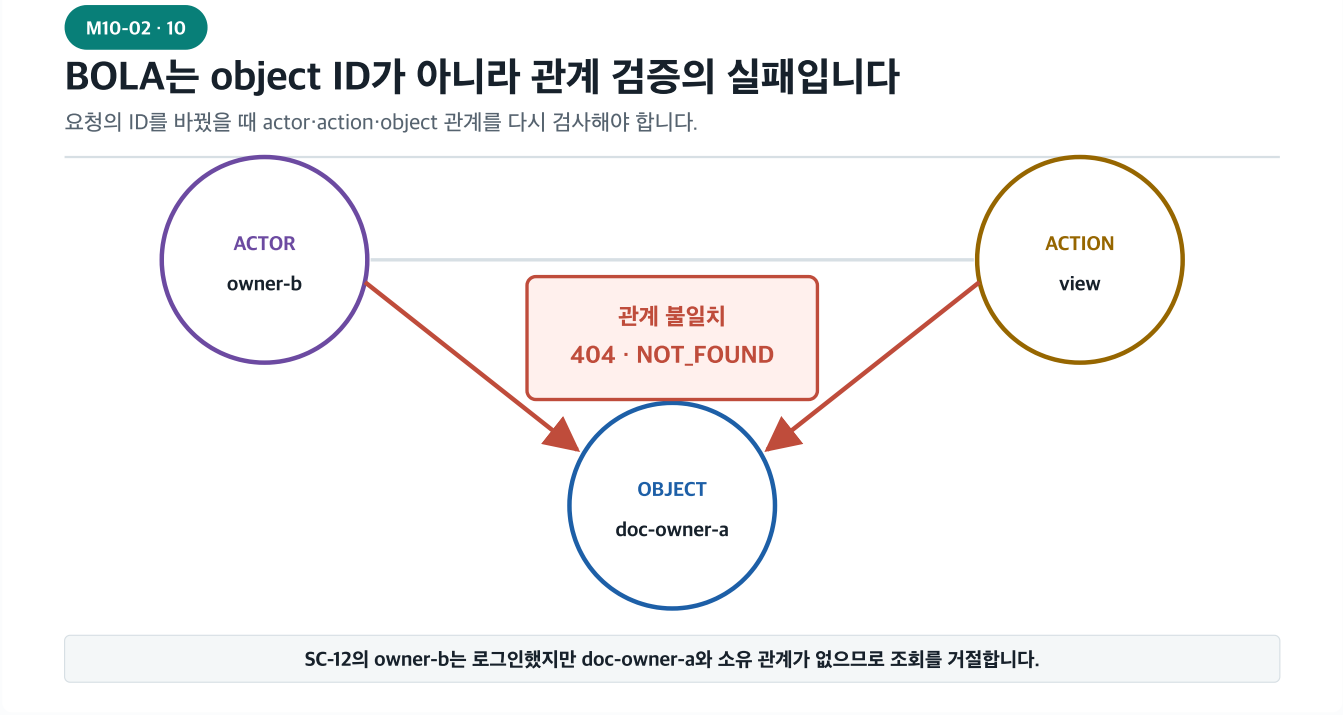
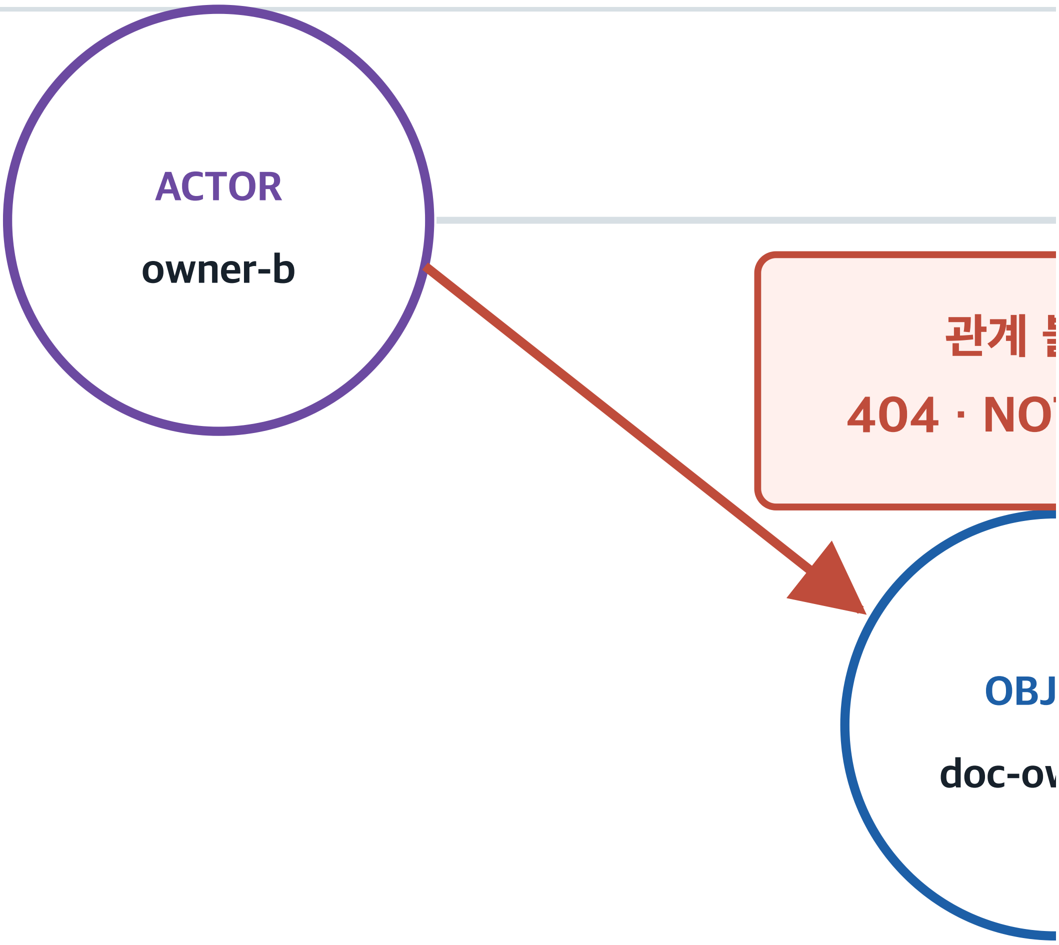
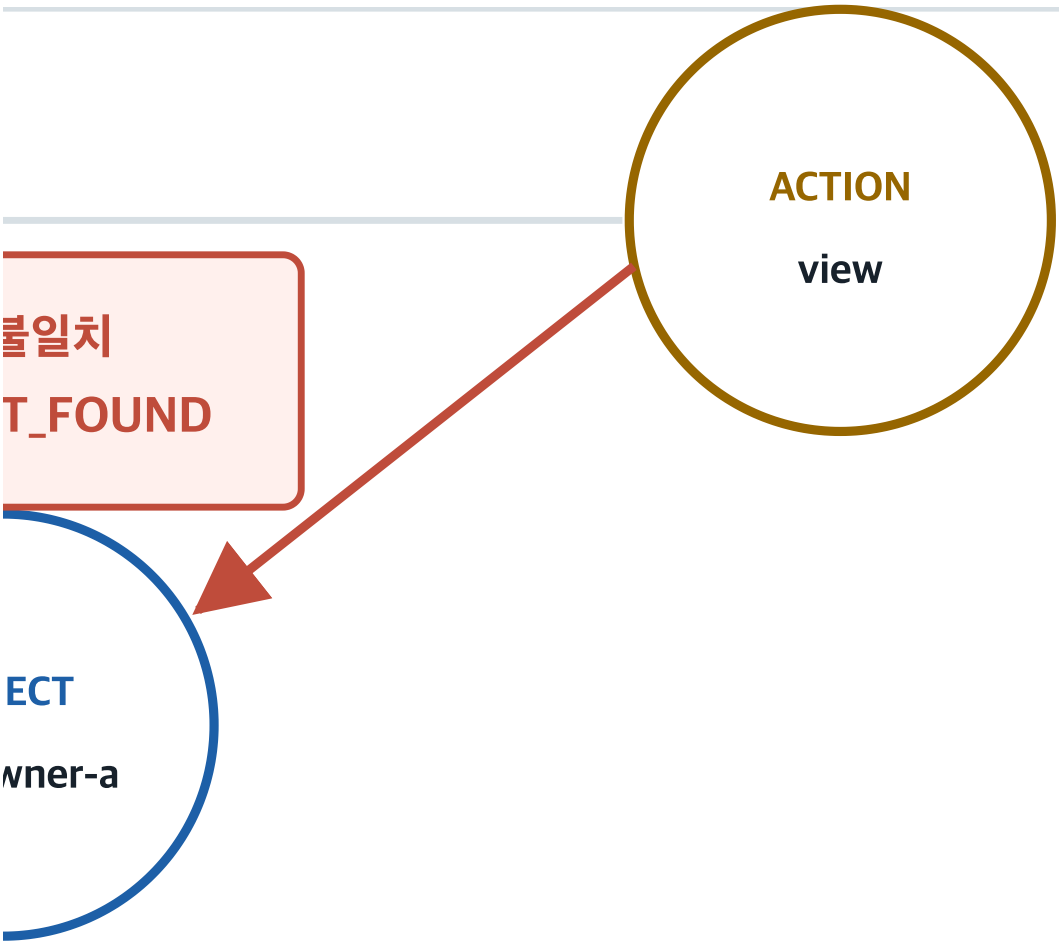


그림 10. 객체 수준 권한 결함은 ID 형식의 문제가 아니라 특정 행위자가 특정 객체를 할 수 있는지 관계 검사가 빠진 문제입니다.

요청의 ID를 바꿨을 때 actor·action·object 관계를 다시 검사해야 합니



다.



10.1 BOLA를 검수 카드로 바꿉니다

Broken Object Level Authorization(BOLA)은 API가 전달받은 object ID만으로 자원을 찾고, 요청자와 객체의 소유·배정·tenant 관계를 확인하지 않을 때 생깁니다.

```
기준 요청: owner-a → view → doc-owner-a → ALLOW
변형 요청: owner-b → view → doc-owner-a → DENY
```

두 요청은 action과 object가 같고 actor 관계만 다릅니다. 이처럼 한 축만 바꾸면 권한 판정의 원인을 분명하게 볼 수 있습니다.

10.2 객체 수준 권한 시나리오 후보

축	허용 대표	거절 대표
소유	own document	other document
배정	assigned document	unassigned document
Tenant	same tenant	other tenant
상태	active resource	archived·deleted resource
하위 자원	own attachment	attachment of other object
목록	filtered own list	hidden object inferred by paging

10.3 판정에서 볼 것

- 응답 status와 business code가 정책에 맞는가.
- body·count·header에 객체 정보가 남지 않는가.
- 검색·목록·내보내기에서도 같은 관계를 적용하는가.
- 하위 자원 endpoint도 상위 객체 관계를 재검사하는가.
- 거절 audit가 원문 비밀정보 없이 남는가.

11. 기능 수준 권한은 endpoint마다 확인합니다

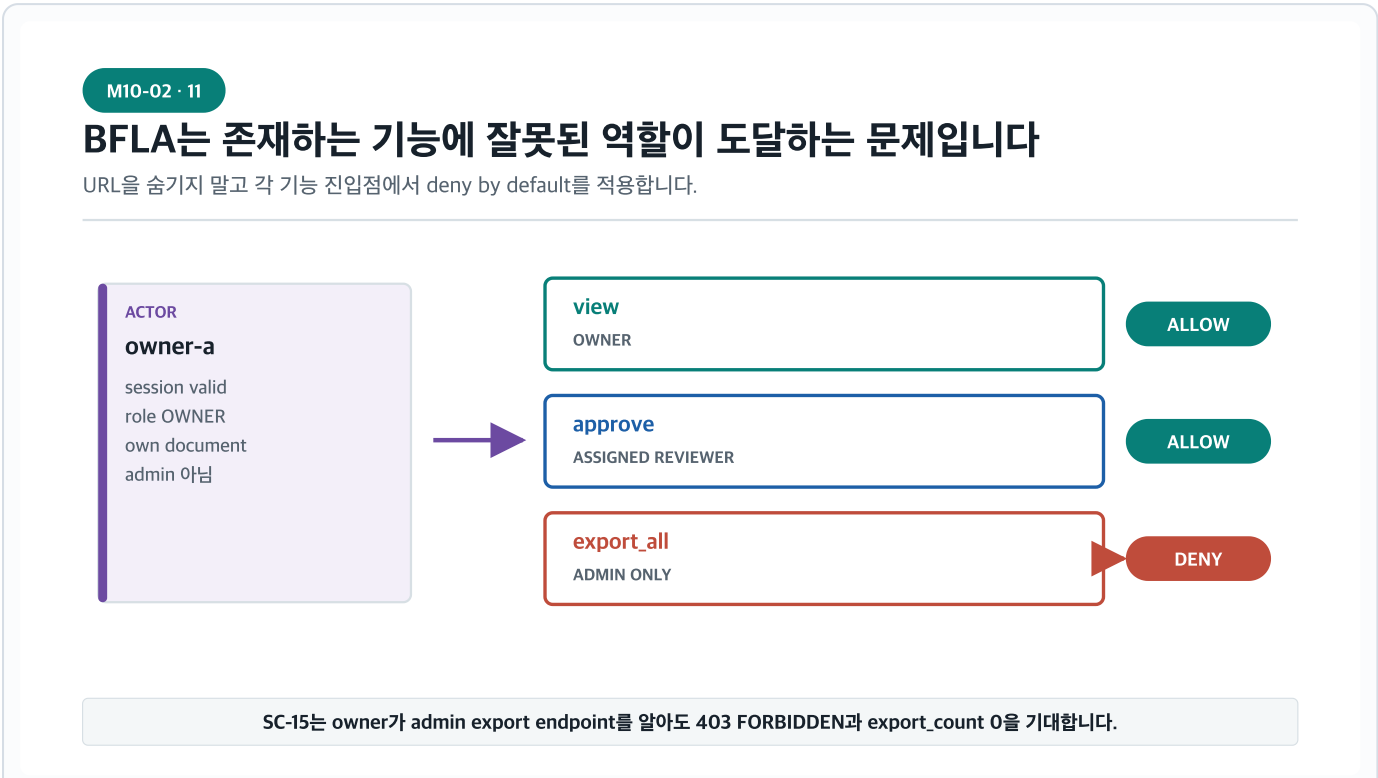
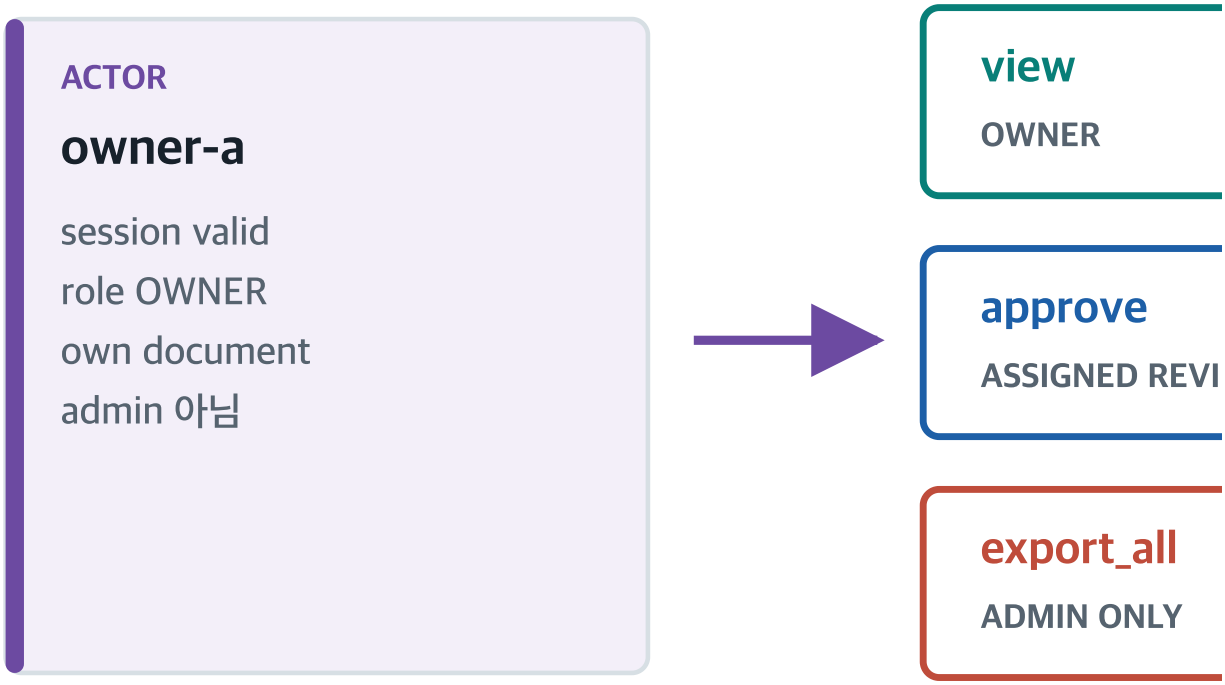
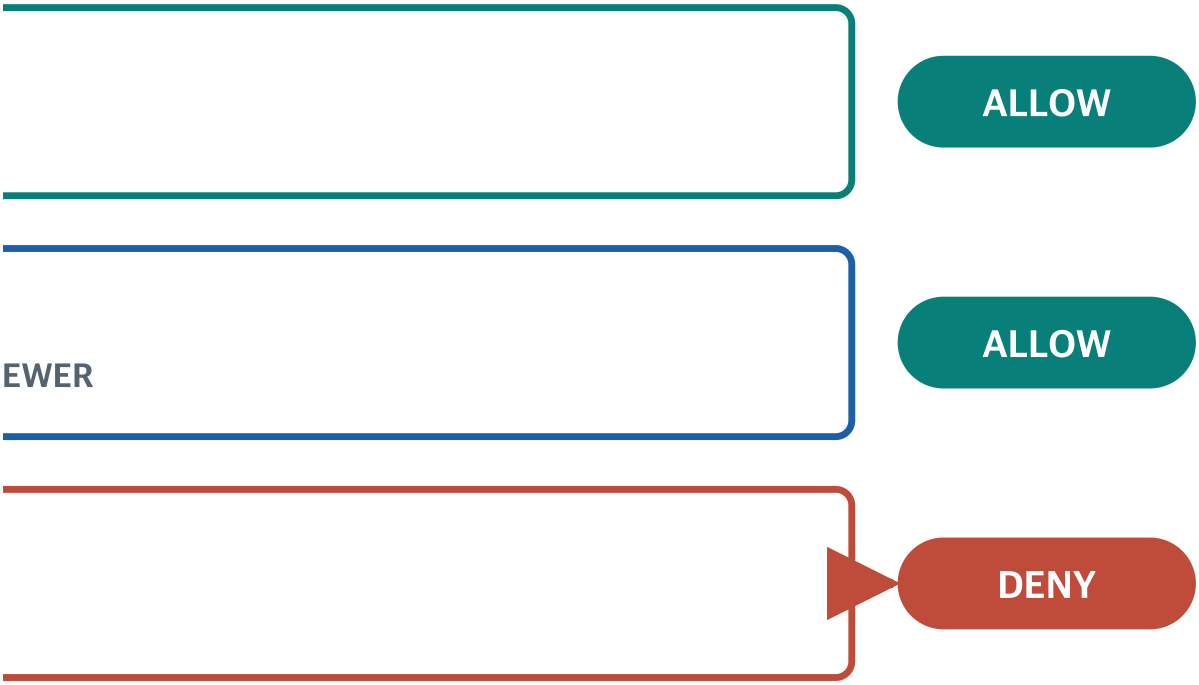


그림 11. 기능을 숨기는 것과 기능을 보호하는 것은 다릅니다. 각 endpoint에서 역할과 정책을 검사합니다.

URL을 숨기지 말고 각 기능 진입점에서 deny by default를 적용합니다




ALLOW
EWER
ALLOW
DENY

11.1 BFLA를 검수 카드로 바꿉니다

Broken Function Level Authorization(BFLA)은 존재하는 기능에 필요한 역할·그룹·정책이 적용되지 않아, 권한 없는 사용자가 관리자·운영자 기능에 도달하는 문제입니다.

```
actor: owner-a
session: valid
role: OWNER
action: export_all
expected: 403 FORBIDDEN, export_count=0, audit_count=1
```

11.2 기능 목록을 UI가 아니라 서버 기준으로 만듭니다

기능군	대표 행동	검수할 역할
조회	view·list·search	anonymous·owner·reviewer·admin
변경	create·edit·submit	owner·other user·admin
승인	approve·reject	assigned·unassigned·owner
관리	export_all·bulk_delete·role_change	owner·reviewer·admin
운영	retry_job·view_audit·feature_toggle	일반 사용자·운영자

UI 메뉴가 보이지 않아도 URL·API route·GraphQL operation·batch endpoint가 존재할 수 있습니다. 서버 route 목록과 권한 정책을 기준으로 기능 목록을 만듭니다.

11.3 BOLA와 BFLA를 구분합니다

구분	핵심 질문	실습
BOLA	이 사용자가 이 객체를 할 수 있는가	SC-12·SC-13
BFLA	이 사용자가 이 기능을 할 수 있는가	SC-15
둘 다	관리자 기능이 특정 객체를 처리할 수 있는가	별도 조합 시나리오

12. 입력과 오류의 안전 경계를 확인합니다

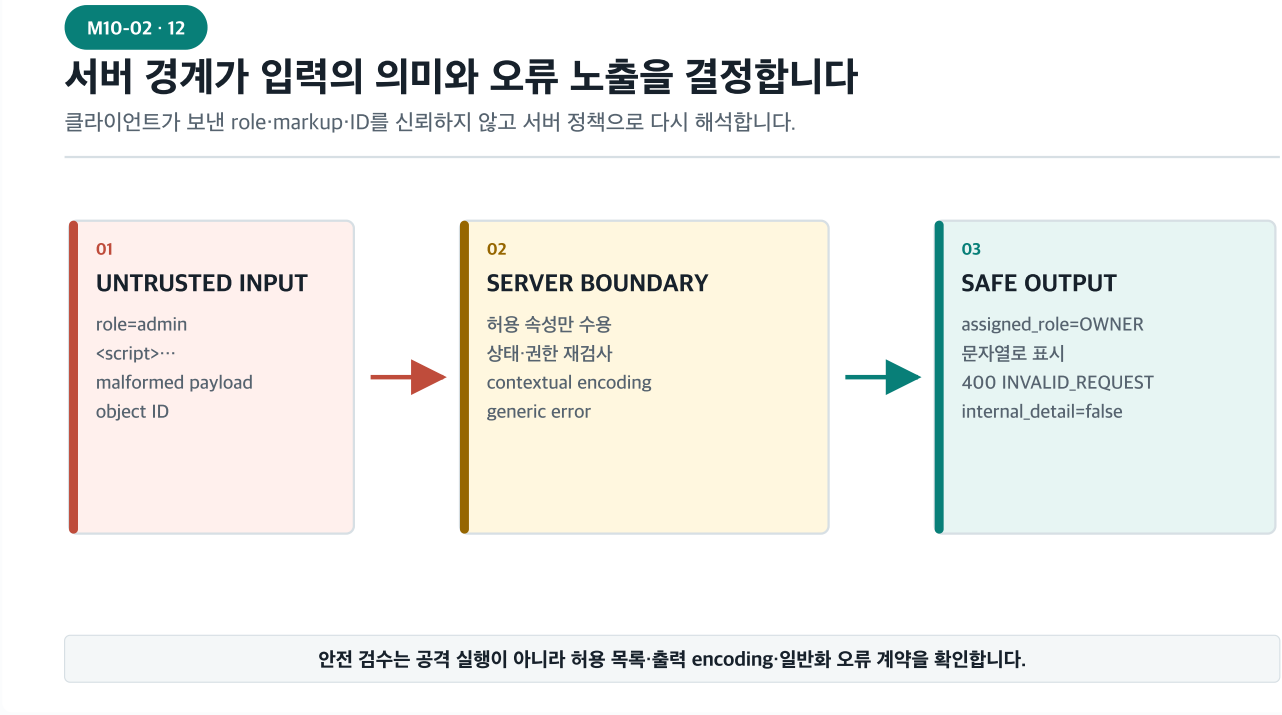


그림 12. 클라이언트가 보낸 role·markup·ID를 신뢰하지 않고, 서버 정책으로 수용·표시·오류 노출을 결정합니다.

클라이언트가 보낸 role·markup·ID를 신뢰하지 않고 서버 정책으로 다

01

UNTRUSTED INPUT

role=admin

<script>…

malformed payload

object ID

02

SERVER BOUNDA

허용 속성만 수용

상태·권한 재검사

contextual encoding

generic error

시 해석합니다.

ARY

03

SAFE OUTPUT

assigned_role=OWNER
문자열로 표시
400 INVALID_REQUEST
internal_detail=false

12.1 허용 속성만 수용합니다

SC-16은 문서 생성 요청에 `role=admin`을 넣습니다. 안전한 후보는 이 속성을 무시하고 `assigned_role=OWNER`를 유지합니다. 클라이언트가 보낸 권한·소유자·가격·승인 상태 같은 민감 속성을 그대로 바인딩하지 않습니다.

12.2 입력 검증과 출력 인코딩을 구분합니다

통제	목적	SC-18 예
입력 검증	허용 형식·길이·구조 결정	제목 문자열 정책 확인
정규화	비교 전 표현 통일	Unicode·공백·대소문자 정책
저장 정책	원문·정규형·표시형 결정	요구사항에 따라 안전하게 저장
출력 인코딩	표시 맥락에서 실행되지 않게 함	`<`를 `<`로 표시
콘텐츠 보안 정책	브라우저 실행 범위 제한	방어 심층화

입력 검증이 출력 인코딩을 대신하지 않습니다. 허용된 문자열도 HTML·URL·JavaScript·CSS 맥락에 맞게 표시해야 합니다.

12.3 일반화 오류 계약

SC-19는 잘못된 payload를 보냅니다. 기대는 `400 INVALID_REQUEST`와 `internal_detail=false`입니다. 파일 경로·stack trace·쿼리·내부 class 이름을 응답에 노출하지 않습니다. 내부에는 진단 가능한 event ID를 남기되, 사용자 응답과 감사 증거에는 필요한 최소 정보만 둡니다.

12.4 이 매뉴얼의 보안 경계

이 실습은 위험한 payload를 실제 시스템에 보내거나 방어를 우회하지 않습니다. 합성 문자열과 고정 actual을 비교해 다음 계약을 학습합니다.

- 서버가 권한 속성을 결정함.
- 사용자 제공 표시 값이 실행되지 않음.
- 잘못된 요청이 내부 상세를 노출하지 않음.
- 거절과 오류가 안전한 audit evidence를 남김.

13. 중복·경합과 비즈니스 규칙을 불변식으로 봅니다

13.1 동시 승인의 위험

두 요청이 모두 `SUBMITTED`를 읽은 뒤 각각 승인하면 approval과 audit가 두 번 생길 수 있습니다. SC-20은 동시 승인 두 번 뒤 다음을 기대합니다.

```
state: APPROVED
approval_count: 1
audit_count: 1
code: ALREADY_APPROVED
```

13.2 경합 시나리오 설계

단계	질문	증거
준비	같은 자원·같은 시작 상태인가	object ID·version·state
실행	요청이 겹치도록 했는가	operation ID·time window
판정	허용되는 성공 수는 몇 개인가	response codes
불변	최종 부작용 수는 몇 개인가	row·event·audit count
복구	재조회 결과가 일관되는가	final state·version

13.3 비즈니스 로직 오용 질문

- 순서를 건너뛸 수 있는가.
- 같은 혜택·승인·쿠폰을 반복할 수 있는가.
- 제한을 여러 계정·요청·채널로 나눠 우회할 수 있는가.
- 클라이언트가 계산한 값이나 상태를 서버가 신뢰하는가.
- 요청이 동시에 오면 한도를 초과하는가.
- 실패 뒤 사용자가 더 유리한 상태를 얻는가.

비즈니스 규칙은 입력 형식이 올바른 요청에서도 깨질 수 있습니다. 상태기계·소유 관계·정확히 한 번 불변식을 서버 권위로 확인합니다.

14. 검수 스튜디오에서 세 버전을 비교합니다

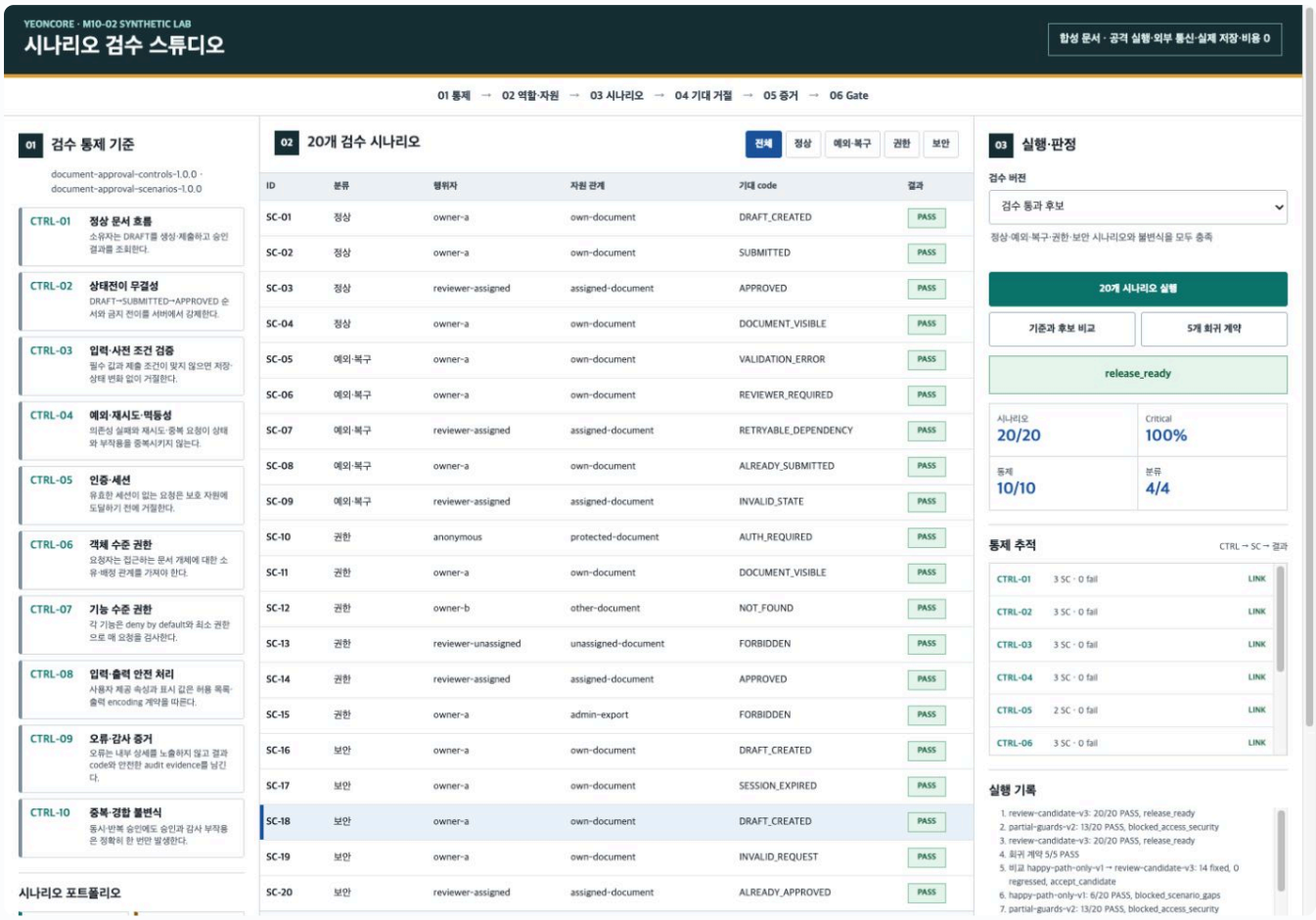


그림 13. 검수 통과 후보. 20/20 시나리오, Critical 100%, 통제 10/10, 범주 4/4가 한 화면에서 연결됩니다.

14.1 세 버전의 학습 의미

버전	PASS	남은 결함	판정
happy-path-only-v1	6/20	14	blocked_scenario_gaps
partial-guards-v2	13/20	7	blocked_access_security
review-candidate-v3	20/20	0	release_ready

첫 버전이 6개를 통과한다는 사실은 시스템이 전혀 작동하지 않는다는 뜻이 아닙니다. 정상 흐름과 일부 허용 시나리오는 작동하지만 방어 계약 14개가 비어 있다는 뜻입니다.

그림 14. 부분 방어 버전. 통제와 범주는 연결됐지만 Critical과 기대 거절이 실패하므로 `blocked_access_security`입니다.

14.2 Coverage와 pass rate를 혼동하지 않습니다

부분 방어 버전은 통제 10/10과 범주 4/4가 모두 연결됐습니다. 그러나 연결된 시나리오가 실패합니다. Coverage는 “검사했다”를, pass rate는 “기대를 만족했다”를 말합니다. 둘 다 필요합니다.

14.3 실습 증거 묶음

생성기는 다음 6개 파일을 만듭니다.

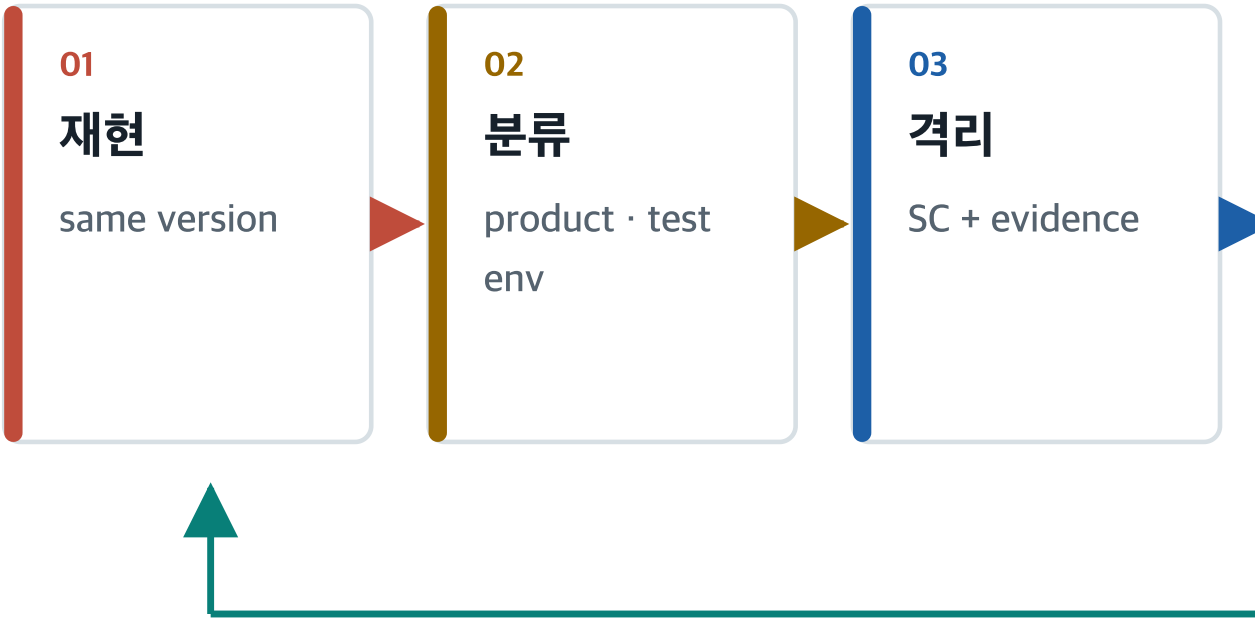
파일	의미
01-reset.txt	실행 전 trace 초기화
02-tests.txt	302개 자동 테스트 결과

파일	의미
03-scenario-review-audit.txt	52개 계약 감사
04-contract-probe.json	세 버전·비교·회귀 결과
05-review-contract.json	version·개수·Gate·안전 경계
06-versions.txt	실행 Python·의존성 정보

15. FAIL을 결함·재검·회귀로 닫습니다

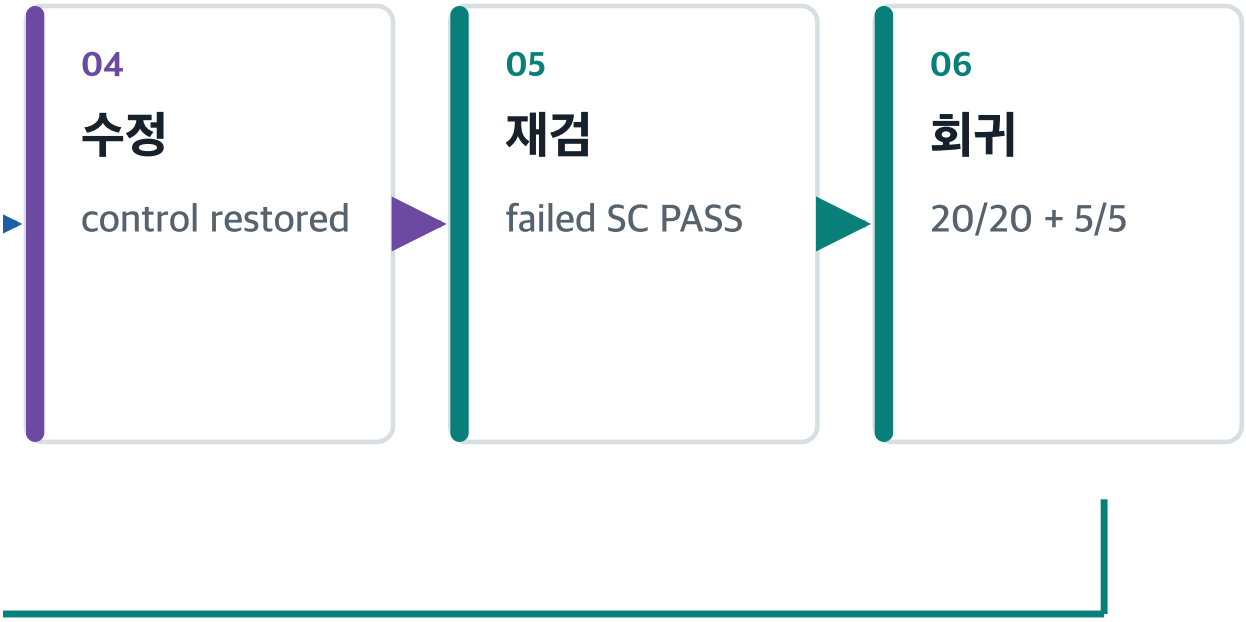
그림 15. 실패 장면만 전달하지 않고 scenario ID와 field mismatch를 결함 증거로 묶습니다.

실패 장면만 전달하지 않고 scenario ID와 field mismatch를 결함 증거



새 실패가 생기면 같은 층

로 묶습니다.



증거 묶음으로 다시 진단

15.1 FAIL이 곧 제품 결함은 아닙니다

후보 원인	확인
요구·정책	기대 허용·거절과 상태 불변식이 승인됐는가
시나리오	actor·relation·precondition·data가 맞는가
환경	contract·build·seed·설정 version이 같은가
Oracle	expected field와 비교 방식이 맞는가
Evidence	실제 요청·응답·전후 상태를 충분히 관찰했는가
제품	위 항목이 맞는데 actual이 expected와 다른가

15.2 좋은 결함 기록

```
defect_id: DEF-012
scenario_id: SC-12
control_id: CTRL-06
version: partial-guards-v2
expected:
  http: 404
  visible: false
  audit_count: 1
actual:
  http: 200
  visible: true
  audit_count: 0
impact: 다른 사용자의 문서 노출 가능
evidence: request summary + response fields + before/after + audit count
retest: SC-12
regression: CTRL-06 linked scenarios + full 20
```

15.3 수정 뒤 세 단계

1. 실패한 시나리오를 같은 version chain에서 재검합니다.
2. 같은 통제를 공유하는 관련 시나리오를 회귀합니다.
3. 전체 20개와 5개 고정 회귀 계약에서 새 실패가 없는지 확인합니다.

16. 릴리스 Gate를 증거 계약으로 운영합니다

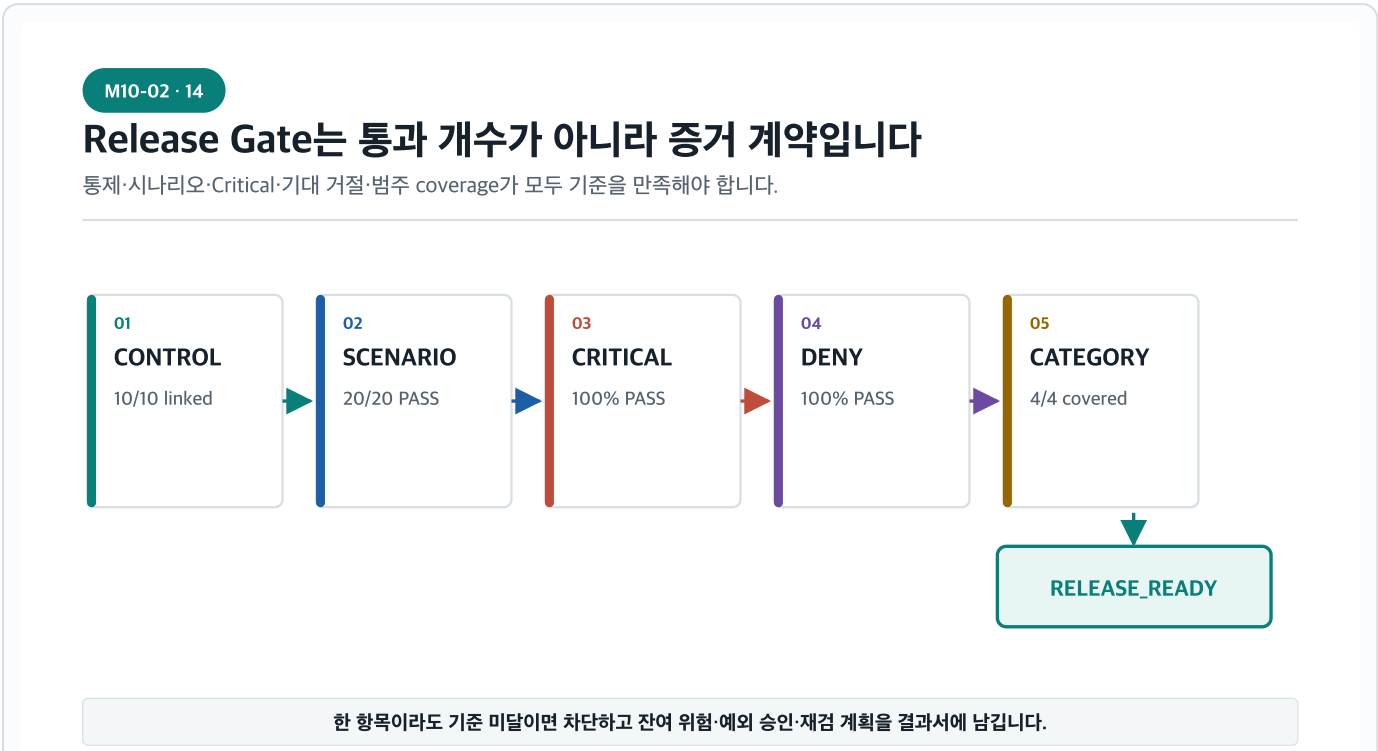
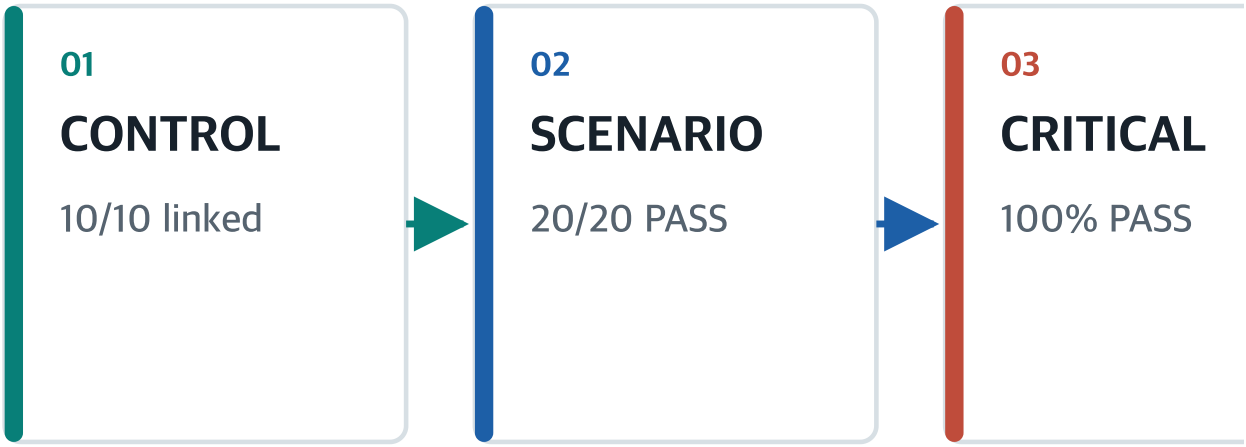
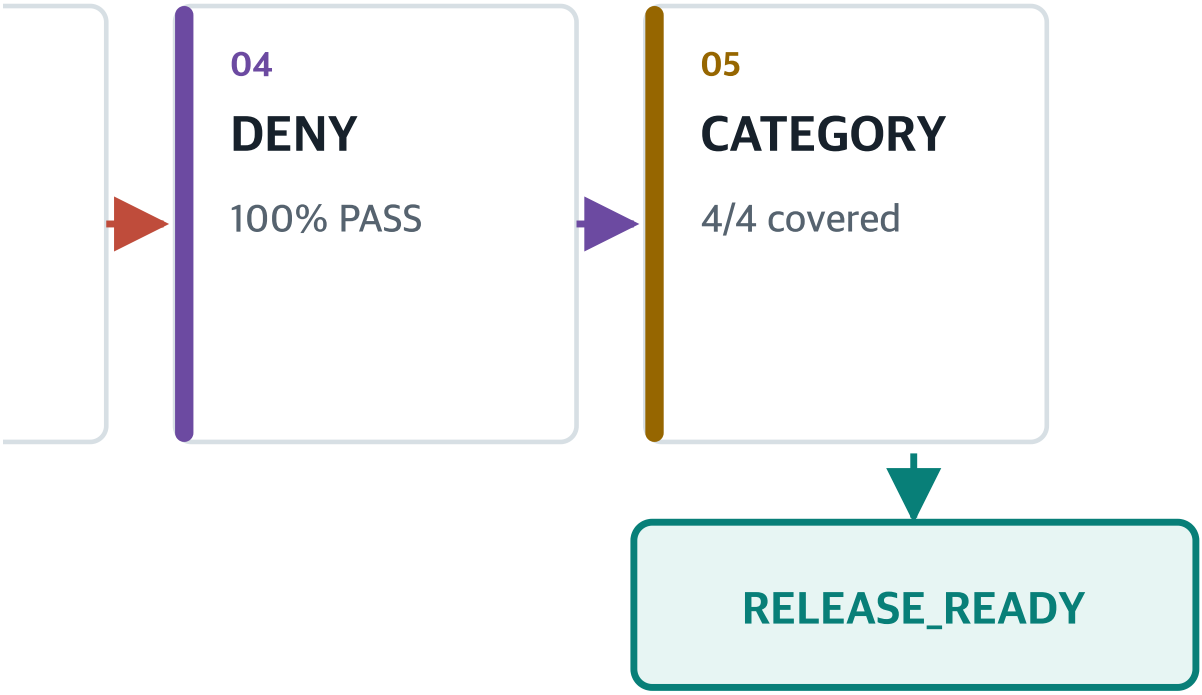


그림 16. 한 항목이라도 기준 미달이면 차단하고, 잔여 위험·예외 승인·재검 계획을 결과서에 남깁니다.

통제·시나리오·Critical·기대 거절·범주 coverage가 모두 기준을 만족해



야 합니다.



16.1 실습 Gate

Gate	기준	후보 결과
전체 시나리오 pass rate	100%	20/20
Critical pass rate	100%	100%
기대 거절 pass rate	100%	100%
통제 coverage	100%	10/10
범주 coverage	100%	4/4
Orphan controls	0	0
Orphan scenarios	0	0
중복 scenario ID	0	0

실제 프로젝트의 threshold는 위험과 배포 유형에 맞게 승인합니다. 다만 Critical 권한·데이터 노출 시나리오를 단순 평균으로 희석하지 않습니다.

16.2 결과서의 최소 결론

항목	기록
Scope	포함 기능·actor·resource·환경·version
Basis	요구사항·정책·통제·공식 기준
Result	범주별·위험별·통제별 실행 결과
Defects	open·fixed·deferred와 영향
Evidence	실행 ID·요약·전후 상태·audit
Residual risk	검수하지 못한 조건과 이유
Decision	release_ready·blocked·exception
Approval	판정자·승인자·날짜·재검 조건

16.3 예외 승인을 숨기지 않습니다

기준 미달 상태로 배포해야 한다면 PASS로 바꾸지 않습니다. `blocked_with_exception`처럼 별도 판정을 사용하고, 다음을 기록합니다.

- 미충족 Gate와 실패 시나리오.
- 예상 영향과 노출 범위.
- 임시 완화 통제.
- 승인 권한자와 만료일.
- 수정 owner와 재검 일정.
- 관찰 지표와 중단 조건.

17. 공식 기준을 실무 질문으로 바꿉니다

17.1 OWASP ASVS 5.0.0

OWASP Application Security Verification Standard(ASVS)는 웹 애플리케이션 보안 통제를 검증하기 위한 요구사항 기준을 제공합니다. 버전이 바뀌면 requirement identifier가 달라질 수 있으므로 결과서에는 ASVS version과 identifier를 함께 적습니다.

공식 자료: OWASP ASVS

17.2 OWASP Web Security Testing Guide

OWASP Web Security Testing Guide(WSTG)는 인증·권한·세션·입력 검증·오류 처리·비즈니스 로직·API 등 테스트 영역을 제공합니다. `latest` 문서는 바뀔 수 있으므로 실제 증거에는 확인 날짜와 사용한 version을 고정합니다.

공식 자료: OWASP WSTG Application Testing

17.3 OWASP Authorization Cheat Sheet

최소 권한, deny by default, 매 요청 권한 검사, 객체·기능 수준 권한, 안전한 종료와 기록, 접근 로직 테스트 원칙을 검수 질문으로 바꿉니다.

공식 자료: OWASP Authorization Cheat Sheet

17.4 OWASP API Security Top 10 2023

API1 Broken Object Level Authorization은 객체 관계 검사를, API5 Broken Function Level Authorization은 기능 진입점의 역할·정책 검사를 강조합니다.

공식 자료: API1 BOLA, API5 BFLA

17.5 오류 처리와 비즈니스 로직

오류 응답에서 내부 상세를 노출하지 않는지, 상태 순서·소유 관계·중복·경합·재시도 규칙을 서버가 권위 있게 강제하는지 확인합니다.

공식 자료: WSTG Improper Error Handling, OWASP Business Logic Security Cheat Sheet

17.6 NIST SSDF v1.1

NIST Secure Software Development Framework(SSDF)은 보안 실천을 소프트웨어 개발 수명주기에 통합해 취약점 발생·영향·재발을 줄이도록 안내합니다. 검수 결과를 일회성 문서가 아니라 요구사항·개발·테스트·릴리스·개선으로 되돌리는 근거로 사용합니다.

공식 자료: NIST SP 800-218 SSDF v1.1

18. 현업 적용 체크리스트

18.1 검수 전

- [ ] 대상 build·contract·data·환경 version을 고정했습니다.
- [ ] 실제 actor와 resource 관계를 목록으로 만들었습니다.
- [ ] 정상·예외·권한·보안 범주가 모두 있습니다.
- [ ] 접근 제어표의 ALLOW·DENY가 승인됐습니다.
- [ ] Critical과 기대 거절 시나리오를 표시했습니다.
- [ ] 원문 비밀정보를 남기지 않는 증거 정책을 정했습니다.

18.2 실행 중

- [ ] 한 시나리오에서 한 핵심 위험을 확인합니다.
- [ ] expected와 actual을 field 단위로 비교합니다.
- [ ] 실패 전후 state·count·visibility·audit를 확인합니다.
- [ ] 거절이 정책에 맞으면 PASS로 기록합니다.
- [ ] FAIL은 version·scenario·mismatch·evidence로 묶습니다.
- [ ] 실제 운영 시스템에 위험한 입력을 시도하지 않습니다.

18.3 판정 후

- [ ] 통제·범주·Critical·DENY coverage와 pass rate를 나눠 봅니다.
- [ ] 결함 원인을 requirement·scenario·environment·oracle·product로 분류합니다.
- [ ] 수정 시나리오와 관련 통제를 재검합니다.
- [ ] 전체 회귀에서 새 실패가 0인지 확인합니다.
- [ ] 잔여 위험과 범위 제한을 결과서에 적습니다.
- [ ] 승인자·날짜·예외 만료·재검 조건을 남깁니다.

19. 30문항 셀프 테스트

19.1 문제

1. 행복 경로가 통과해도 Release Gate를 열 수 없는 이유를 한 문장으로 쓰십시오.
2. 정상·예외·권한·보안 네 범주의 핵심 질문을 각각 쓰십시오.
3. 기대 거절 시나리오에서 HTTP 403이 나온 경우 언제 PASS입니까.
4. actor·action·resource·context 네 요소를 SC-13에 적용하십시오.
5. 인증과 권한의 차이를 설명하십시오.
6. deny by default가 필요한 이유를 쓰십시오.
7. 최소 권한 원칙을 시나리오 질문으로 바꾸십시오.
8. 소유자와 다른 소유자의 객체 조회를 쌍으로 검수해야 하는 이유는 무엇입니까.
9. BOLA의 핵심 누락 검사는 무엇입니까.
10. BFLA의 핵심 누락 검사는 무엇입니까.
11. SC-12가 404를 기대하는 이유를 쓰십시오.
12. SC-15가 확인하는 자원과 기능은 무엇입니까.
13. 실패 응답 code만 확인하면 안 되는 이유를 쓰십시오.
14. 안전 실패 불변식 네 가지를 쓰십시오.
15. timeout 뒤 state가 APPROVED라면 왜 FAIL입니까.
16. 멱등성 시나리오에서 count를 확인해야 하는 이유는 무엇입니까.
17. 동시 승인 시나리오의 최종 approval_count는 몇이어야 합니까.
18. 클라이언트가 보낸 `role=admin`을 서버가 그대로 적용하면 어떤 통제가 깨집니까.
19. 입력 검증과 출력 인코딩의 목적 차이를 쓰십시오.
20. 일반화 오류 응답이 숨겨야 할 정보 세 가지를 쓰십시오.

21. Coverage 100%와 pass rate 100%의 차이를 설명하십시오.
22. Critical 실패를 전체 평균으로 희석하면 안 되는 이유는 무엇입니까.
23. Orphan control은 무엇입니까.
24. 좋은 결함 기록의 최소 필드 다섯 가지를 쓰십시오.
25. FAIL이 곧 제품 결함이 아닌 이유를 쓰십시오.
26. 수정 뒤 재검·관련 회귀·전체 회귀를 나누는 이유는 무엇입니까.
27. 잔여 위험에 반드시 적어야 할 내용 세 가지를 쓰십시오.
28. 예외 승인을 PASS로 기록하면 안 되는 이유는 무엇입니까.
29. 이 실습이 침투 테스트 완료를 증명하지 않는 이유를 쓰십시오.
30. 자신의 기능 하나를 고르고 정상·예외·권한·보안 시나리오를 한 개씩 제목으로 쓰십시오.

19.2 모범 답안

1. 정상 흐름 밖의 예외·기대 거절·권한·경합 통제가 실패할 수 있기 때문입니다.
2. 정상은 목표 완수, 예외는 안전 실패와 복구, 권한은 관계별 허용·거절, 보안은 변조·노출·우회·경합에도 통제 유지입니다.
3. 시나리오 expected가 403이고 state·count·visibility·audit도 기대와 같을 때 PASS입니다.
4. reviewer-unassigned·approve·unassigned-document·valid session와 SUBMITTED 상태입니다.
5. 인증은 요청자의 신원을, 권한은 그 신원이 특정 자원·기능을 할 수 있는지를 판정합니다.
6. 명시적으로 허용하지 않은 새 조합과 새 기능이 우연히 허용되는 것을 막기 위해서입니다.
7. 목표 수행에 필요하지 않은 기능과 자원까지 접근할 수 있는지 질문합니다.
8. action과 상태를 같게 두고 소유 관계만 바꾸면 객체 권한 검사의 효과를 분리해 볼 수 있기 때문입니다.
9. 요청자·행동·특정 객체의 소유·배정·tenant 관계 검사입니다.
10. 각 endpoint·operation에 필요한 역할·그룹·정책 검사입니다.
11. 권한 없는 사용자에게 객체 존재 자체를 노출하지 않는 정책을 검수하기 위해서입니다.
12. 관리자 내보내기 자원과 `export_all` 기능입니다.
13. 오류를 반환하면서 state나 부작용을 반영하거나 내부 정보를 노출할 수 있기 때문입니다.
14. 안전 상태 유지, record·approval delta 0, 중복 부작용 0, 비인가 visibility false, 내부 상세 비노출 중 네 가지입니다.
15. 승인 의무가 원자적으로 끝나지 않았는데 상태만 바뀌어 부분 실패와 무결성 위반이 생겼기 때문입니다.
16. 같은 요청의 재시도가 중복 저장·제출·감사 event를 만들지 않았는지 확인하기 위해서입니다.
17. 1입니다.
18. 서버 권위, 허용 속성 목록, 기능 수준 권한, 최소 권한 통제가 깨집니다.
19. 입력 검증은 수용할 값과 구조를 결정하고, 출력 인코딩은 표시 맥락에서 값이 실행되지 않게 합니다.
20. stack trace, 파일 경로, 쿼리, 내부 class, 비밀정보 중 세 가지입니다.
21. Coverage는 통제·범주를 검사 대상으로 연결했는지, pass rate는 연결된 시나리오가 기대를 만족했는지를 뜻합니다.

22. 권한 우회나 데이터 노출 한 건이 다수의 저위험 통과보다 영향이 클 수 있기 때문입니다.
23. 검증할 통제에 연결된 시나리오가 하나도 없는 상태입니다.
24. defect ID, scenario ID, control ID, version, expected·actual mismatch, impact, evidence, owner 중 다섯 가지입니다.
25. 요구·정책, 시나리오, 환경, oracle, 증거가 잘못되어도 FAIL이 생길 수 있기 때문입니다.
26. 수정 효과, 같은 통제의 영향, 예상 밖 퇴행을 서로 다른 범위에서 확인하기 위해서입니다.
27. 검수하지 못한 조건, 예상 영향, 이유, 완화 통제, owner, 재검 일정 중 세 가지입니다.
28. 기준 미달과 승인 책임·만료·재검 조건을 숨겨 잘못된 품질 신호를 만들기 때문입니다.
29. 합성 계약의 고정 시나리오만 확인하며 실제 공격 표면·인프라·알려지지 않은 취약점을 평가하지 않기 때문입니다.
30. 답은 기능에 따라 다르며 네 범주가 서로 다른 위험을 확인하고 observable expected를 가지면 됩니다.

20. 마지막 한 장 요약

```
1. 통제와 정책 version을 고정한다.
2. actor·action·resource·context를 목록으로 만든다.
3. 정상·예외·권한·보안 네 범주를 채운다.
4. ALLOW와 DENY를 접근 제어표로 승인한다.
5. 시나리오에 시작 상태·행동·기대·oracle·evidence를 적는다.
6. 실패 전후 state·count·visibility·audit 불변식을 비교한다.
7. 객체와 기능 권한을 매 요청 독립적으로 확인한다.
8. FAIL을 mismatch와 증거로 분류하고 재검·회귀한다.
9. 전체·Critical·DENY·통제·범주 Gate를 따로 판정한다.
10. 잔여 위험·예외 승인·재검 조건을 결과서에 남긴다.
```

완료 기준: 자신의 기능에 대해 네 범주의 시나리오, 접근 제어표, field 단위 expected·actual, 결함·재검 기록, 잔여 위험, 릴리스 판정을 한 검수 결과서로 설명할 수 있으면 이 매뉴얼을 마친 것입니다.