CONFIDENTIAL

Vulnerability Assessment & Penetration Testing

Technical Report

Information Systems — [CLIENT]

Client	[CLIENT]
Target	[TARGET]
Asset type	Web Application and API
Testing approach	Grey-box
Standards	OWASP Top 10 2025, OWASP API Security Top 10 2023
Overall risk	**Medium**
Report version	1.0
Prepared by	Security Assessment Team

Table of Contents

List of Figures **3**

List of Tables **5**

1 Scope of Work **6**

1.1 Methodology 6

1.2 Standards and Frameworks 7

1.2.1 Web Application (OWASP Top 10 2025) 7

1.2.2 API (OWASP API Security Top 10 2023) 7

1.3 List of Assets 8

2 Summary of Vulnerabilities **9**

3 Findings on Web Application and API **12**

3.1 Other 12

3.1.1 Broken Object Level Authorization (API1) – F-01 - Broken Object Level Authorization Across Project-Scoped Resources 12

3.1.2 Broken Function Level Authorization (API5) – F-02 - Broken Function Level Authorization Across Restricted Features 53

3.1.3 Unrestricted Access to Sensitive Business Flows (API6) – F-04 - Arbitrary Historical Date Accepted for Submitted Daily Reports 68

3.1.4 Unrestricted Access to Sensitive Business Flows (API6) – F-05 - Future-Dated Delivery Accepted as Completed Progress 71

3.1.5 Broken Authentication (API2) – F-03 - Insufficient Session Invalidation After Logout 74

3.1.6 Security Misconfiguration (API8) – F-06 - Improper Input Validation in Worker Profile Fields 80

3.1.7 Broken Object Level Authorization (API1) – P-01 - Authorization and Object Access Controls 86

3.1.8 Broken Authentication (API2) – P-02 - Authentication and Session Cookie Controls 88

3.1.9 Broken Object Property Level Authorization (API3) – P-03 - Object Property and Project Binding Controls 90

3.1.10 Unrestricted Access to Sensitive Business Flows (API6) – P-04 - Business Workflow Controls 92

3.1.11 Injection (A05) – P-05 - Injection and Output-Encoding Controls 93

3.1.12 Security Misconfiguration (API8) – P-06 - Active Content Upload Controls 95

3.1.13 Security Misconfiguration (API8) – P-07 - Cross-Origin Resource Sharing Controls 96

4 Summary of Recommendations **97**

5 Conclusion **99**

6 Attachments **100**

List of Figures

Figure 1 - Vulnerability Assessment and Penetration Testing Methodology 6
Figure 2 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 1 step 1 13
Figure 3 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 1 step 2 14
Figure 4 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 1 14
Figure 5 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 2 15
Figure 6 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 3 15
Figure 7 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 4 16
Figure 8 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 5 16
Figure 9 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 6 17
Figure 10 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 1 ... 17
Figure 11 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 2 ... 18
Figure 12 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 3 ... 18
Figure 13 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 4 ... 19
Figure 14 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 5 ... 19
Figure 15 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 6 ... 20
Figure 16 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 7 ... 20
Figure 17 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 8 ... 21
Figure 18 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 9 ... 21
Figure 19 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 10 . 22
Figure 20 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 4 step 1 ... 22
Figure 21 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 4 step 2 ... 23
Figure 22 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 4 step 3 ... 23
Figure 23 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 4 step 4 ... 24
Figure 24 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 5 step 1 ... 24
Figure 25 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 5 step 2 ... 25
Figure 26 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 5 step 3 ... 25
Figure 27 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 5 step 4 ... 26
Figure 28 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 6 step 1 ... 26
Figure 29 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 6 step 2 ... 27
Figure 30 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 6 step 3 ... 27
Figure 31 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 6 step 4 ... 28
Figure 32 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 7 step 1 ... 28
Figure 33 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 7 step 2 ... 29
Figure 34 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 7 step 3 ... 29
Figure 35 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 7 step 4 ... 30
Figure 36 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 8 step 1 ... 31
Figure 37 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 8 step 2 ... 31
Figure 38 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 8 step 3 ... 32
Figure 39 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 8 step 4 ... 32
Figure 40 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 9 step 1 ... 33
Figure 41 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 9 step 2 ... 33
Figure 42 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 9 step 3 ... 34
Figure 43 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 9 step 4 ... 34
Figure 44 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 10 step 1 . 35
Figure 45 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 10 step 2 . 35
Figure 46 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 10 step 3 . 36
Figure 47 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 10 step 4 . 36
Figure 48 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 11 step 1 . 37

Figure 49 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 11 step 2 . 37
Figure 50 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 11 step 3 . 38
Figure 51 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 11 step 4 . 38
Figure 52 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 12 step 1 . 39
Figure 53 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 12 step 2 . 39
Figure 54 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 12 step 3 . 40
Figure 55 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 12 step 4 . 40
Figure 56 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 13 step 1 . 41
Figure 57 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 13 step 2 . 41
Figure 58 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 13 step 3 . 42
Figure 59 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 13 step 4 . 42
Figure 60 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 14 step 1 . 43
Figure 61 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 14 step 2 . 43
Figure 62 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 14 step 3 . 44
Figure 63 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 14 step 4 . 44
Figure 64 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 15 step 1 . 45
Figure 65 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 15 step 2 . 45
Figure 66 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 15 step 3 . 46
Figure 67 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 15 step 4 . 46
Figure 68 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 16 step 1 . 47
Figure 69 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 16 step 2 . 47
Figure 70 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 16 step 3 . 48
Figure 71 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 16 step 4 . 48
Figure 72 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 17 step 1 . 49
Figure 73 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 17 step 2 . 49
Figure 74 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 17 step 3 . 50
Figure 75 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 17 step 4 . 50
Figure 76 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 18 step 1 . 51
Figure 77 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 18 step 2 . 51
Figure 78 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 18 step 3 . 52
Figure 79 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 18 step 4 . 52
Figure 80 - Broken Function Level Authorization Across Restricted Features - scenario 1 step 1 54
Figure 81 - Broken Function Level Authorization Across Restricted Features - scenario 1 step 2 54
Figure 82 - Broken Function Level Authorization Across Restricted Features - scenario 1 step 3 55
Figure 83 - Broken Function Level Authorization Across Restricted Features - scenario 1 step 4 55
Figure 84 - Broken Function Level Authorization Across Restricted Features - scenario 2 step 1 56
Figure 85 - Broken Function Level Authorization Across Restricted Features - scenario 2 step 2 56
Figure 86 - Broken Function Level Authorization Across Restricted Features - scenario 2 step 3 57
Figure 87 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 1 57
Figure 88 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 2 58
Figure 89 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 3 58
Figure 90 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 4 59
Figure 91 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 5 59
Figure 92 - Broken Function Level Authorization Across Restricted Features - scenario 4 step 1 60
Figure 93 - Broken Function Level Authorization Across Restricted Features - scenario 4 step 2 60
Figure 94 - Broken Function Level Authorization Across Restricted Features - scenario 4 step 3 61
Figure 95 - Broken Function Level Authorization Across Restricted Features - scenario 5 step 1 61
Figure 96 - Broken Function Level Authorization Across Restricted Features - scenario 5 step 2 62
Figure 97 - Broken Function Level Authorization Across Restricted Features - scenario 5 step 3 62
Figure 98 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 1 63
Figure 99 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 2 63

Figure 100 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 3 64
Figure 101 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 4 64
Figure 102 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 5 65
Figure 103 - Broken Function Level Authorization Across Restricted Features - scenario 7 step 1 65
Figure 104 - Broken Function Level Authorization Across Restricted Features - scenario 7 step 2 66
Figure 105 - Broken Function Level Authorization Across Restricted Features - scenario 7 step 3 66
Figure 106 - Broken Function Level Authorization Across Restricted Features - scenario 7 step 4 67
Figure 107 - Arbitrary Historical Date Accepted for Submitted Daily Reports - scenario 1 step 1 69
Figure 108 - Arbitrary Historical Date Accepted for Submitted Daily Reports - scenario 1 step 2 69
Figure 109 - Arbitrary Historical Date Accepted for Submitted Daily Reports - scenario 1 step 3 70
Figure 110 - Arbitrary Historical Date Accepted for Submitted Daily Reports - scenario 1 step 4 70
Figure 111 - Future-Dated Delivery Accepted as Completed Progress - scenario 1 step 1 72
Figure 112 - Future-Dated Delivery Accepted as Completed Progress - scenario 1 step 2 72
Figure 113 - Future-Dated Delivery Accepted as Completed Progress - scenario 1 step 3 73
Figure 114 - Insufficient Session Invalidation After Logout - scenario 1 step 1 75
Figure 115 - Insufficient Session Invalidation After Logout - scenario 1 step 2 76
Figure 116 - Insufficient Session Invalidation After Logout - scenario 1 step 3 76
Figure 117 - Insufficient Session Invalidation After Logout - scenario 1 step 4 77
Figure 118 - Insufficient Session Invalidation After Logout - scenario 2 step 1 77
Figure 119 - Insufficient Session Invalidation After Logout - scenario 2 step 2 78
Figure 120 - Insufficient Session Invalidation After Logout - scenario 2 step 3 78
Figure 121 - Insufficient Session Invalidation After Logout - scenario 2 step 4 79
Figure 122 - Improper Input Validation in Worker Profile Fields - scenario 1 step 1 81
Figure 123 - Improper Input Validation in Worker Profile Fields - scenario 1 step 2 81
Figure 124 - Improper Input Validation in Worker Profile Fields - scenario 1 step 3 82
Figure 125 - Improper Input Validation in Worker Profile Fields - scenario 1 step 4 82
Figure 126 - Improper Input Validation in Worker Profile Fields - scenario 2 step 1 83
Figure 127 - Improper Input Validation in Worker Profile Fields - scenario 2 step 2 83
Figure 128 - Improper Input Validation in Worker Profile Fields - scenario 2 step 3 84
Figure 129 - Improper Input Validation in Worker Profile Fields - scenario 2 step 4 84
Figure 130 - Improper Input Validation in Worker Profile Fields - scenario 2 step 5 85

List of Tables

Table 1 - Penetration Testing Approaches 6
Table 2 - OWASP Top 10 2025 Categories 7
Table 3 - OWASP API Security Top 10 2023 Categories 7
Table 4 - List of Assets 8
Table 5 - Vulnerability Summary – Other 9

1 Scope of Work

1.1 Methodology

The methodology used in this vulnerability assessment and penetration test is described below.

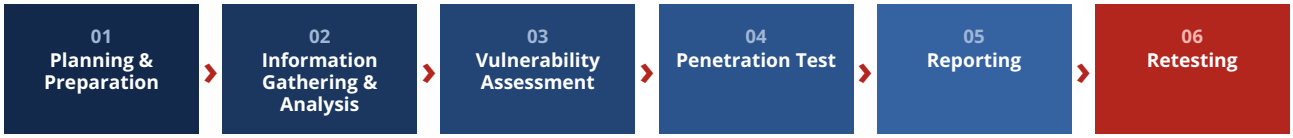


Figure 1 - Vulnerability Assessment and Penetration Testing Methodology

1. **Planning and understanding the target system and supporting applications.** Understanding the target system, supporting applications, public and internal IP addresses, operating systems (OS), databases, and application topology within the relevant work unit, through interviews or the documentation provided.
2. **Conducting the vulnerability assessment.** Assessing vulnerabilities and identifying security weaknesses within the scope of the system and its supporting network infrastructure, including web applications, operating systems, databases, and networks.
3. **Performing penetration testing based on the assessment results.** Exploitation aims to prove that the security gaps or vulnerabilities found exist, and to assess their level of risk and impact on the company's applications, information, and business reputation.
4. **Reporting and discussing results with Team [CLIENT].** Preparing the report and discussing the penetration test results and recommendations for improvement, and reaching a mutual agreement on reducing the existing security gaps. Recommendations are based on the standards and frameworks used in the test.
5. **Retesting the implemented improvements.** Retesting the improvements implemented to ensure that security gaps and risks have been effectively reduced.

The testing approaches used in penetration testing are black-box, grey-box and white-box:

Table 1 - Penetration Testing Approaches

Approach	Description	This engagement
Black-box	Conducted without any prior information or knowledge about the target. Testing uses only a list of IP addresses or domains, without detailed knowledge of the system profile or application topology.	—
Grey-box	Conducted with some information and knowledge about the target, including user access to web applications or servers but without full access rights. Testing is still limited by firewall filtering or a WAF (Web Application Firewall).	✔ Applied
White-box	Conducted with comprehensive information about the target, such as the application topology, test user IDs, internal network access, and the IP addresses of the target and users, without any firewall or WAF filtering.	—

1.2 Standards and Frameworks

Findings in this report are classified against the following standards and frameworks: **OWASP Top 10 2025** and **OWASP API Security Top 10 2023**.

1.2.1 Web Application (OWASP Top 10 2025)

The OWASP Top 10 (2025) is the standard awareness document for the most critical security risks to web applications. Reference: https://owasp.org/Top10/2025/

Table 2 - OWASP Top 10 2025 Categories

ID	Category
A01:2025	Broken Access Control
A02:2025	Security Misconfiguration
A03:2025	Software Supply Chain Failures
A04:2025	Cryptographic Failures
A05:2025	Injection
A06:2025	Insecure Design
A07:2025	Authentication Failures
A08:2025	Software or Data Integrity Failures
A09:2025	Security Logging and Alerting Failures
A10:2025	Mishandling of Exceptional Conditions

1.2.2 API (OWASP API Security Top 10 2023)

The OWASP API Security Top 10 (2023) lists the most critical security risks specific to APIs. Reference: https://owasp.org/API-Security/editions/2023/en/0x11-t10/

Table 3 - OWASP API Security Top 10 2023 Categories

ID	Category
API1:2023	Broken Object Level Authorization
API2:2023	Broken Authentication
API3:2023	Broken Object Property Level Authorization
API4:2023	Unrestricted Resource Consumption
API5:2023	Broken Function Level Authorization
API6:2023	Unrestricted Access to Sensitive Business Flows
API7:2023	Server Side Request Forgery
API8:2023	Security Misconfiguration
API9:2023	Improper Inventory Management

ID	Category
API10:2023	Unsafe Consumption of APIs

1.3 List of Assets

The scope of security testing and penetration testing for the [CLIENT] web application is as follows:

Table 4 - List of Assets

No	Name	URL	Type	Status
1	**[APPLICATION]**	`[TARGET]`	Web Application	FAIL
2	**[APPLICATION] API**	`[TARGET]`	API	FAIL

Status legend: **PASS** — no vulnerability found **FAIL** — vulnerability found

2 Summary of Vulnerabilities

Testing of the [CLIENT] web application identified **6 vulnerabilities**: **5 Medium** and **1 Low**; in addition, **7 tested scenarios** did not result in a vulnerability (PASS).

0	0	5	1	0	7
CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL	PASS

Other

Web Application and API · 6 vulnerabilities, 7 PASS

Table 5 - Vulnerability Summary – Other

No	Vulnerability	Severity	CVSS	Exploited	Recommendation
1	**Broken Object Level Authorization (API1) – F-01 - Broken Object Level Authorization Across Project-Scoped Resources** API1:2023 Broken Object Level Authorization	Medium	6.5	Yes	Resolve the authenticated identity to explicit project membership before loading project resources.
2	**Broken Function Level Authorization (API5) – F-02 - Broken Function Level Authorization Across Restricted Features** API5:2023 Broken Function Level Authorization	Medium	6.5	Yes	Enforce the approved role-to-function matrix before loading business data.
3	**Unrestricted Access to Sensitive Business Flows (API6) – F-04 - Arbitrary Historical Date Accepted for Submitted Daily Reports** API6:2023 Unrestricted Access to Sensitive Business Flows	Medium	4.3	Yes	Derive report dates from the server-side active work date and enforce an allowed reporting window.
4	**Unrestricted Access to Sensitive Business Flows (API6) – F-05 - Future-Dated Delivery Accepted as Completed Progress** API6:2023 Unrestricted Access to Sensitive Business Flows	Medium	4.3	Yes	Reject delivery dates after the authoritative server date.
5	**Broken Authentication (API2) – F-03 - Insufficient Session Invalidation After Logout** API2:2023 Broken Authentication	Medium	4.2	Yes	Revoke or invalidate the server-side session when logout succeeds.

No	Vulnerability	Severity	CVSS	Exploited	Recommendation
6	**Security Misconfiguration (API8) – F-06 - Improper Input Validation in Worker Profile Fields** API8:2023 Security Misconfiguration	Low	2.7	Yes	Enforce server-side format and length rules for phone and national identity number fields.
7	**Broken Object Level Authorization (API1) – P-01 - Authorization and Object Access Controls** API1:2023 Broken Object Level Authorization API5:2023 Broken Function Level Authorization	PASS	—	—	Retain the tested controls and add regression tests for the exact role, object and action combinations.
8	**Broken Authentication (API2) – P-02 - Authentication and Session Cookie Controls** API2:2023 Broken Authentication A07:2025 Authentication Failures A04:2025 Cryptographic Failures	PASS	—	—	Retain the tested token, CSRF and cookie controls; separately invalidate sessions at logout.
9	**Broken Object Property Level Authorization (API3) – P-03 - Object Property and Project Binding Controls** API3:2023 Broken Object Property Level Authorization	PASS	—	—	Retain tested property allowlists, role checks, candidate isolation and parent-project binding.
10	**Unrestricted Access to Sensitive Business Flows (API6) – P-04 - Business Workflow Controls** API6:2023 Unrestricted Access to Sensitive Business Flows	PASS	—	—	Retain server-side state-transition and quantity invariants, with regression tests.
11	**Injection (A05) – P-05 - Injection and Output-Encoding Controls** A05:2025 Injection API8:2023 Security Misconfiguration	PASS	—	—	Retain strict parsing and context-aware output encoding; extend coverage to other input/rendering contexts.
12	**Security Misconfiguration (API8) – P-06 - Active Content Upload Controls** API8:2023 Security Misconfiguration A08:2025 Software or Data Integrity Failures	PASS	—	—	Retain server-side extension, MIME, content-signature and storage controls on every upload function.
13	**Security Misconfiguration (API8) – P-07 - Cross-Origin Resource Sharing Controls** API8:2023 Security Misconfiguration	PASS	—	—	Retain explicit trusted-origin handling and credential restrictions on every API response.

References used for every finding: OWASP Top 10 2025, OWASP API Security Top 10 2023, Bugcrowd VRT, CWE 4.0, CVSS v3.1.

3 Findings on Web Application and API

3.1 Other

3.1.1 Broken Object Level Authorization (API1) – F-01 - Broken Object Level Authorization Across Project-Scoped Resources

SEVERITY	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS
Medium	6.5 / 10	Yes	CWE-639

Standard mapping	**API1:2023** Broken Object Level Authorization (OWASP API Security Top 10 2023)
Impact rating	Medium
CVSS vector	`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N`
Affected feature / URL	• Project list and project details (read) • Personnel records (list and direct record, read) • Inventory items and deliveries (read) • Cost categories, budget/financial lines, and cost records (read) • Documents: metadata, direct file download, and cost receipts (read/download) • Purchase orders, organization entries, floors, project mail settings, and daily reports (read)
Testing approach	Grey-box

DESCRIPTION

Object-level authorization did not consistently bind the requested project, child object, collection, or file to the caller's project membership and object ownership. Authenticated users assigned to a different project retrieved foreign project-scoped data in 17 documented scenarios, in both directions between the two test projects.

Attack prerequisites: The demonstrated access used authenticated test accounts assigned to a different project than the data owner; no victim interaction was required.

Expected: Requests for projects, child objects, collections, and files outside the caller's project membership are denied or filtered out.

Actual: Authenticated users assigned to a different project retrieved foreign project-scoped data in 17 documented scenarios, in both directions between the two test projects.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

Authenticated Roles A, B or C read unrelated project data without victim interaction. Recorded disclosures included personnel identity/contact fields, project/member details, inventory, financial data, document metadata and file references, purchase orders, deliveries, organization/floor metadata, and selected document and receipt bodies. No unauthorized write impact is claimed.

RECOMMENDATION

1. Resolve the authenticated identity to explicit project membership before loading project resources.

2. Enforce deny-by-default authorization using identity, membership, object ownership and the parent project.
3. Bind parent and child resources, including files, to the same authorized project; filter collections by membership.
4. Add bidirectional cross-project access regression tests for every resource.

EVIDENCE

Public-edition evidence: Each recorded step is followed by its corresponding selectively redacted screenshot. Client identity, origins, credentials, opaque identifiers, test-account identifiers, and personal data are removed at the pixel level.

UI context — Cross-project Project List visibility

1. LEADER test account displayed projects outside its isolated assignment, including both A/B fixtures.

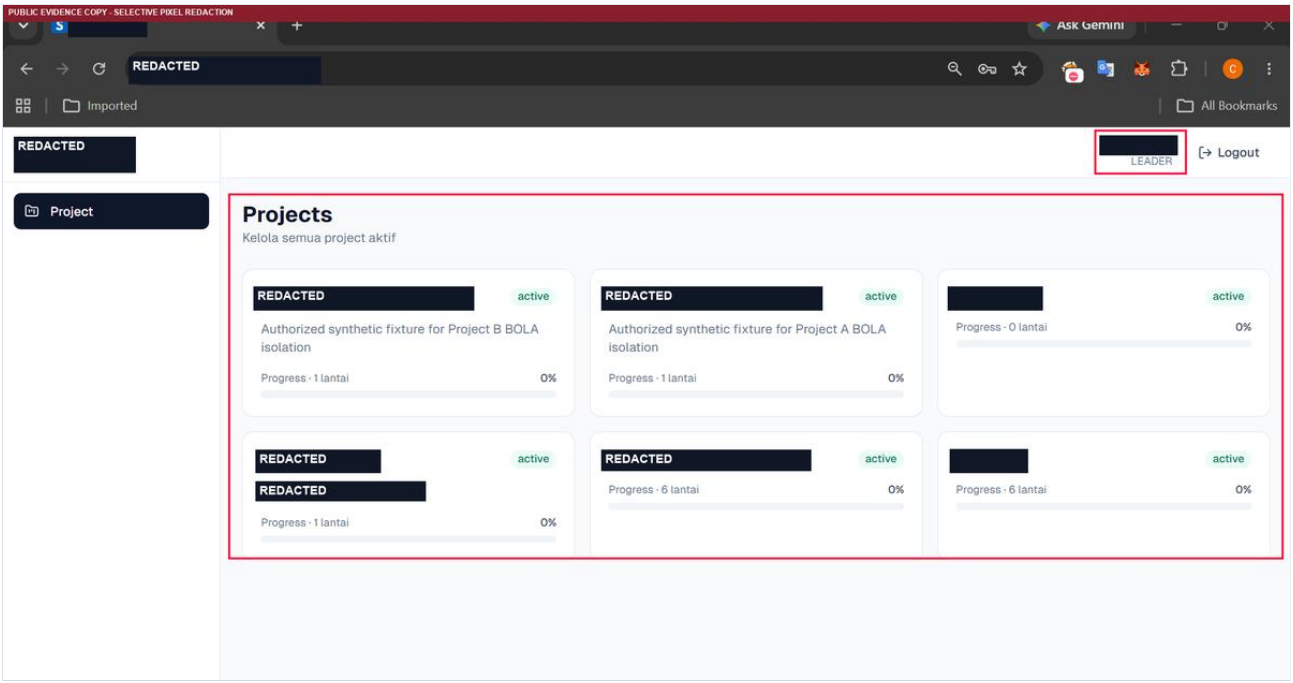


Figure 2 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 1 step 1

2. The second Leader identity displayed the same cross-project collection, providing real UI context for the API authorization failure.

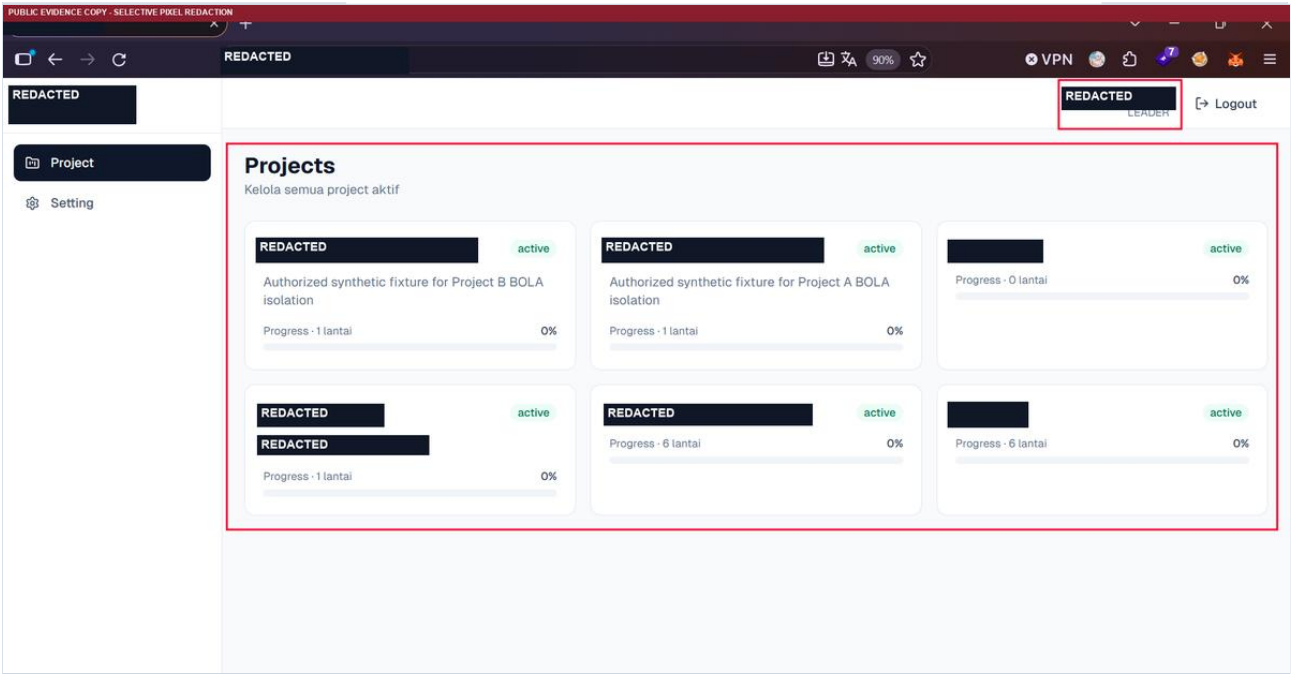


Figure 3 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 1 step 2

Affected scenario 1 — Project - Project List

1. Log in with the authorized test account (CLIENT) test account and open Project at /project. The screenshot shows exactly one project card ([TEST FIXTURE]), proving collection filtering at the UI layer for CLIENT.

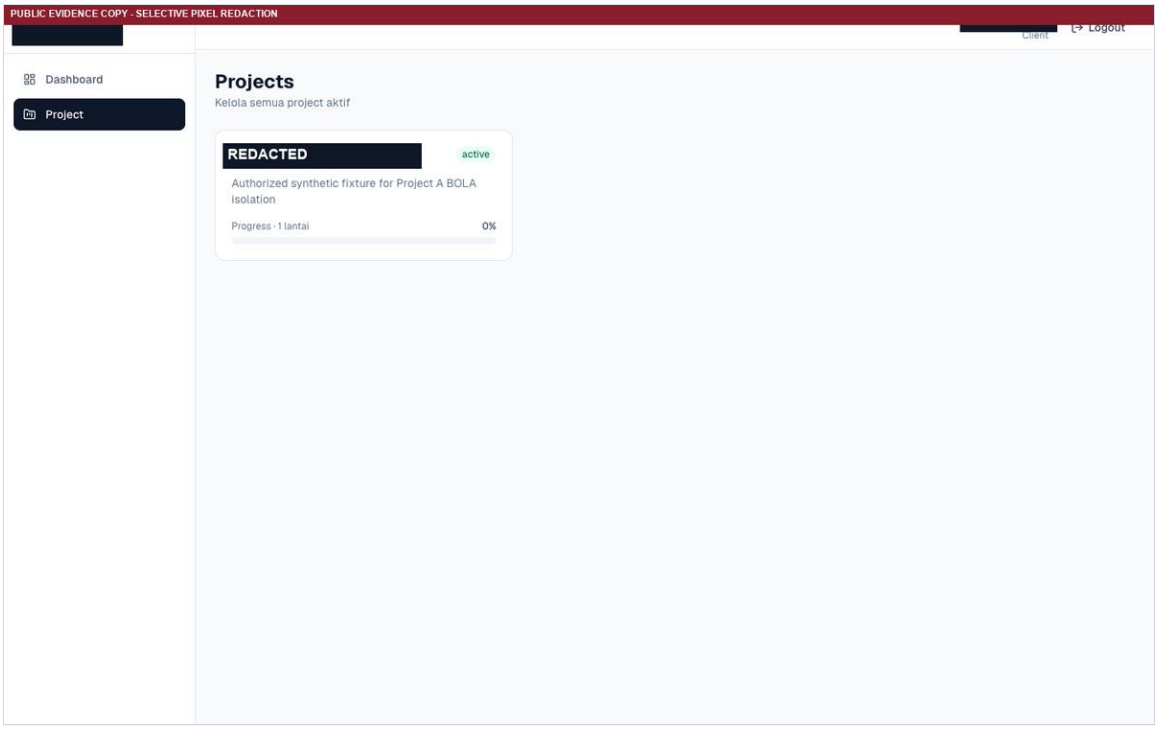


Figure 4 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 1

2. Log in with test account (PM) and open Project at /project. The screenshot shows the project collection as visible to PM, providing the list context for the enumeration checks below.

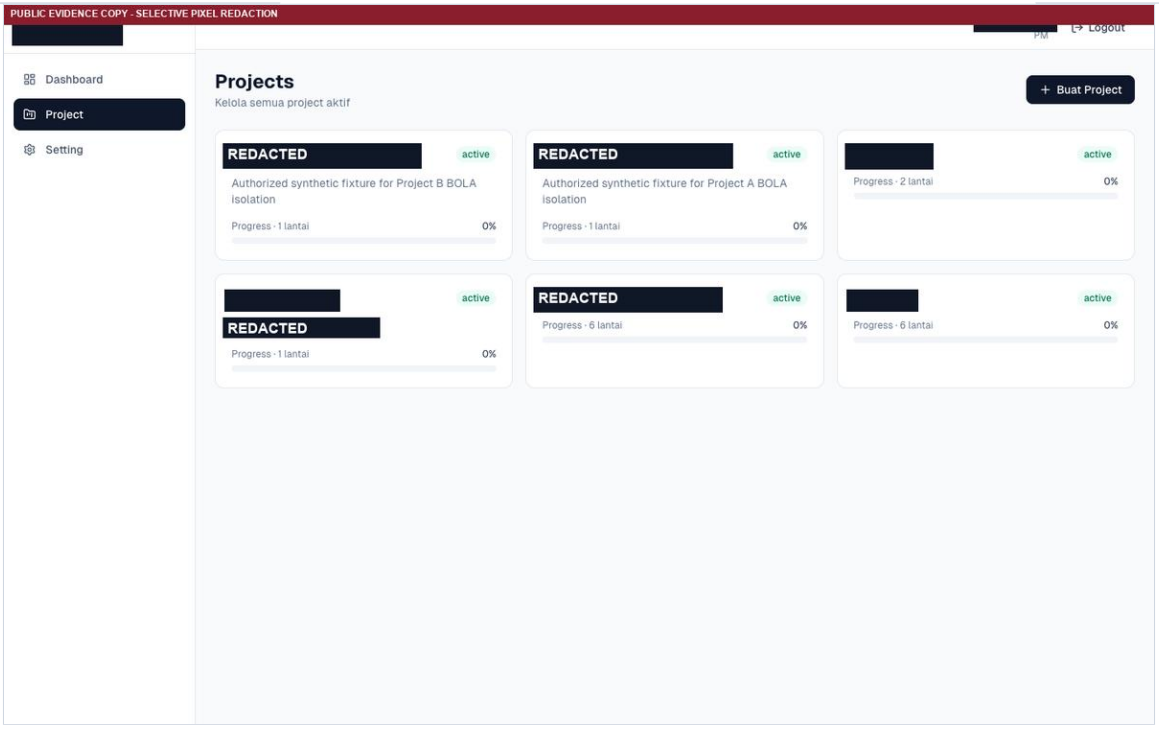


Figure 5 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 2

3. CLIENT-A requests the collection. The screenshot shows the red-boxed request and the red-boxed own project id [OBJECT ID] as the only entry, proving filtering holds.

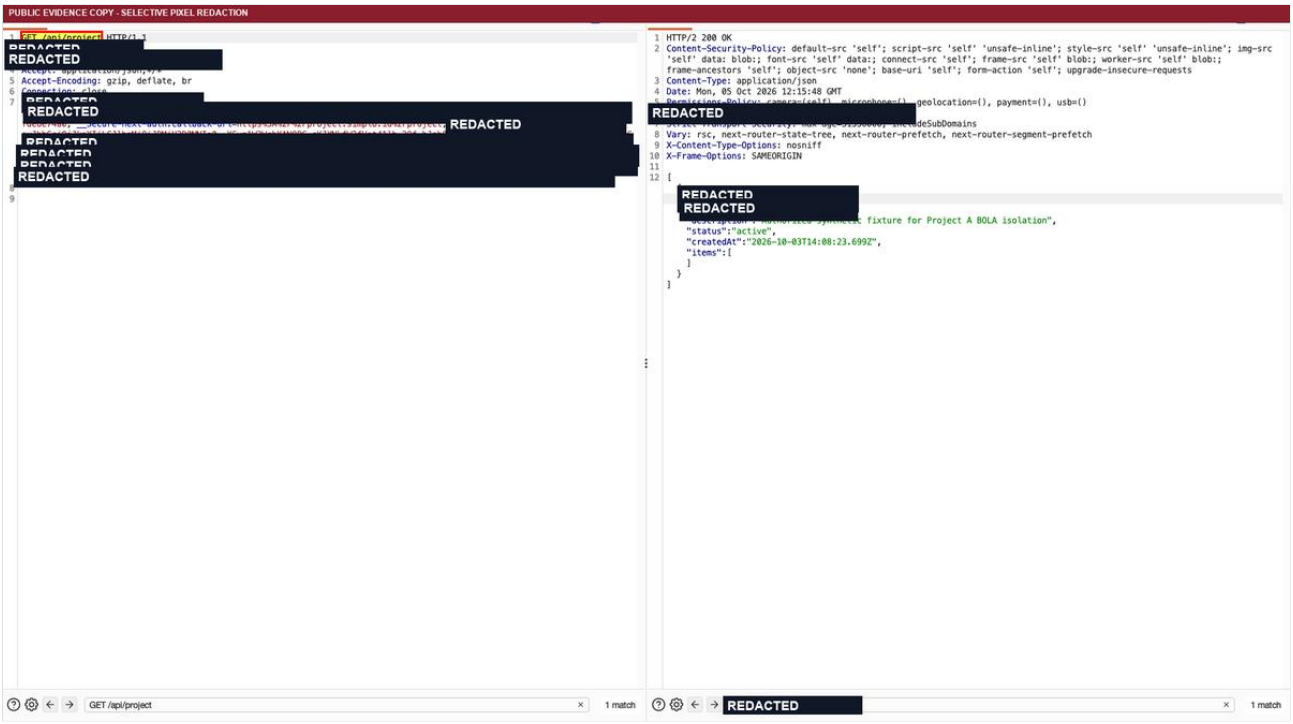


Figure 6 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 3

4. CLIENT-B requests the collection. The screenshot shows the red-boxed own project id [OBJECT ID] as the only entry, proving filtering holds.

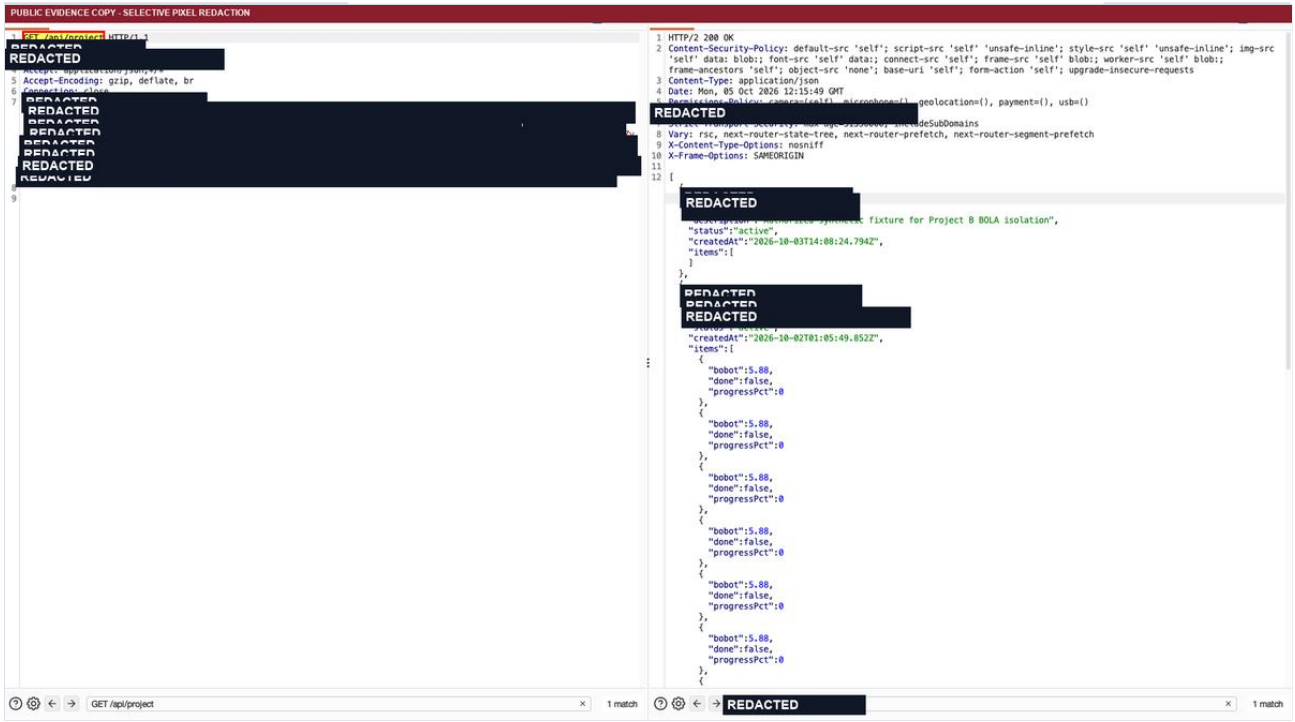


Figure 7 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 4

5. PM-A requests the collection. The screenshot shows the red-boxed foreign project id [OBJECT ID] enumerated in the HTTP 200 response despite no cross-project membership, proving tenant enumeration.

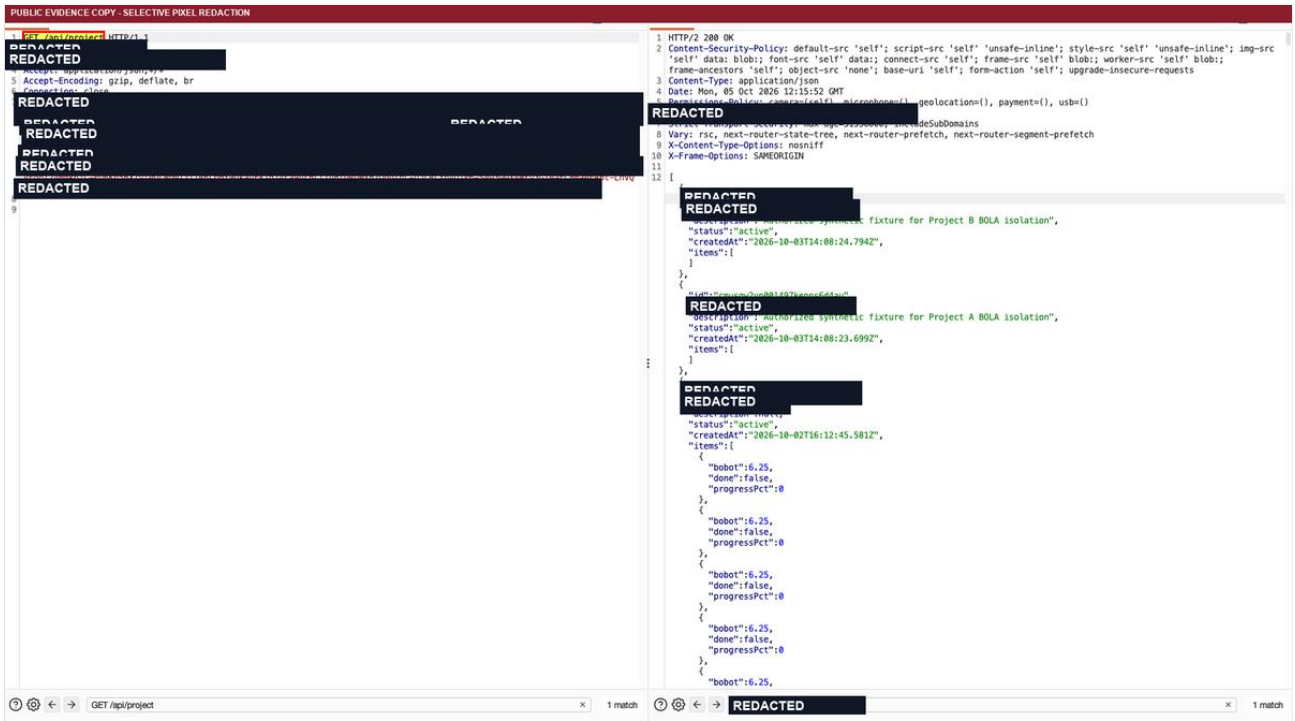


Figure 8 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 5

6. PM-B requests the collection. The screenshot shows the red-boxed foreign project id [OBJECT ID] enumerated, proving the bidirectional failure.

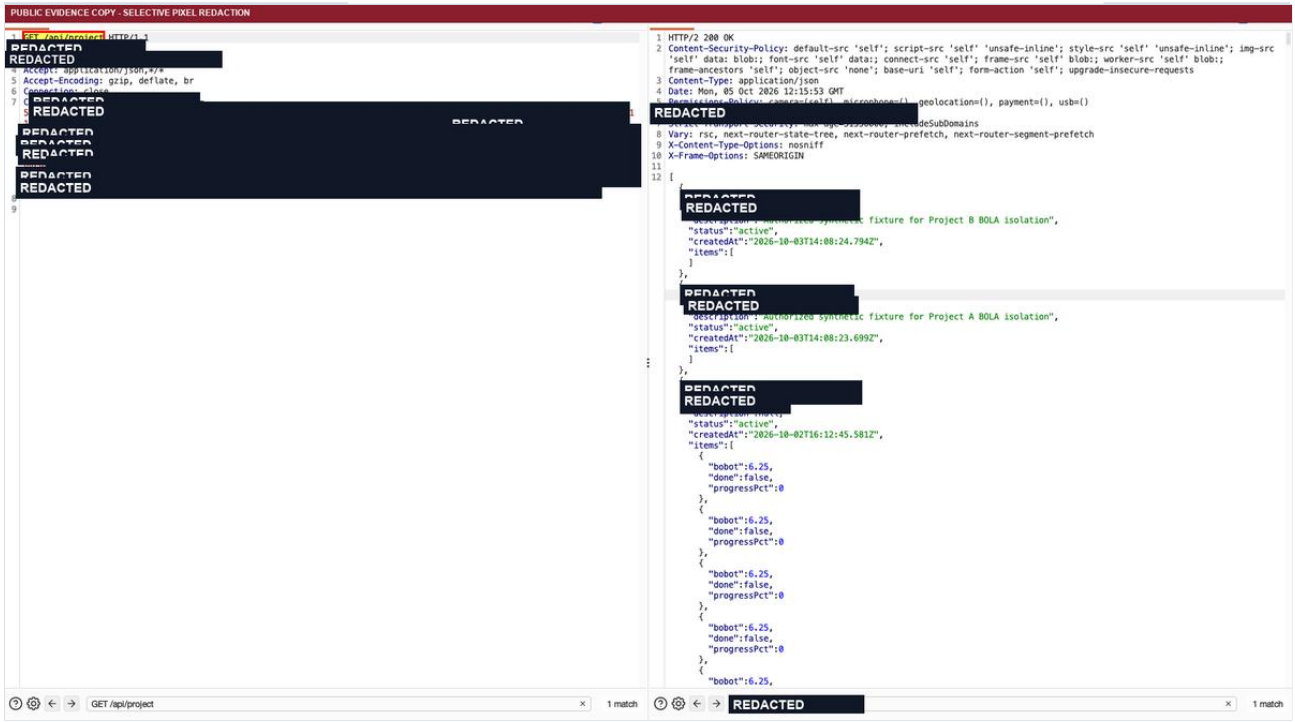


Figure 9 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 2 step 6

Affected scenario 2 — Project - Project Details

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

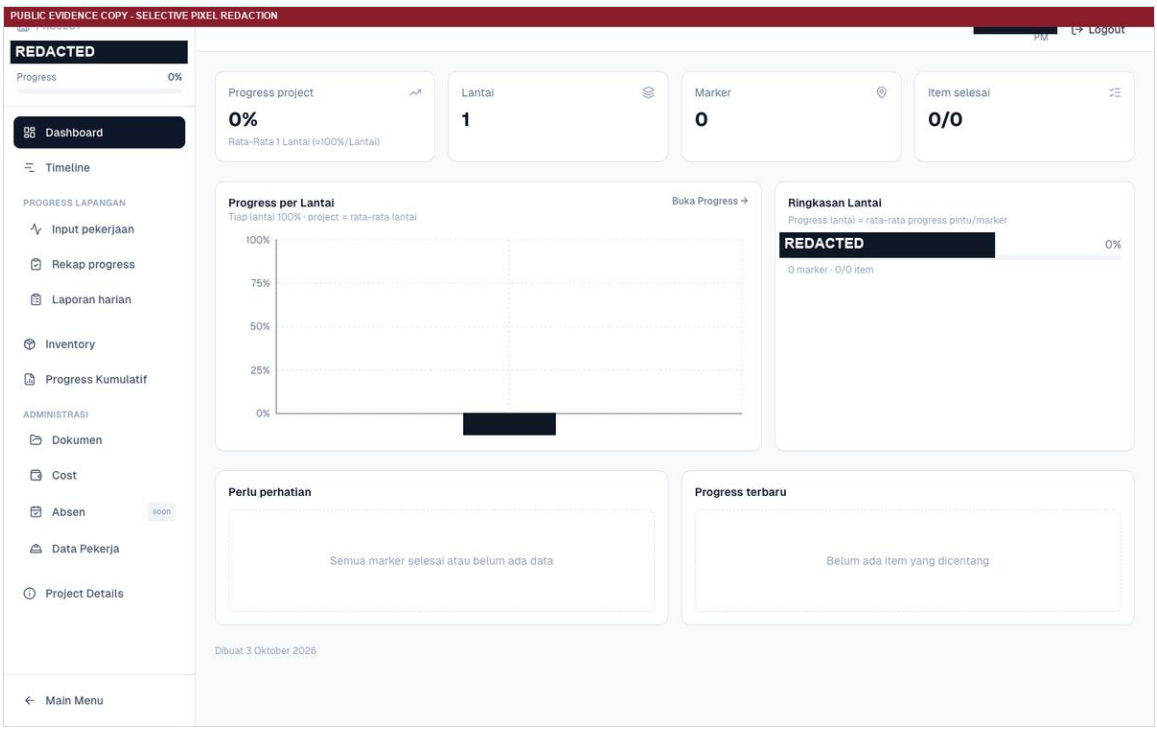


Figure 10 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 1

2. Click Project Details in the sidebar navigation. The screenshot shows the Project Details page with the project's own data visible (project identity, floor/marker counters, and Struktur Organisasi card),

establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

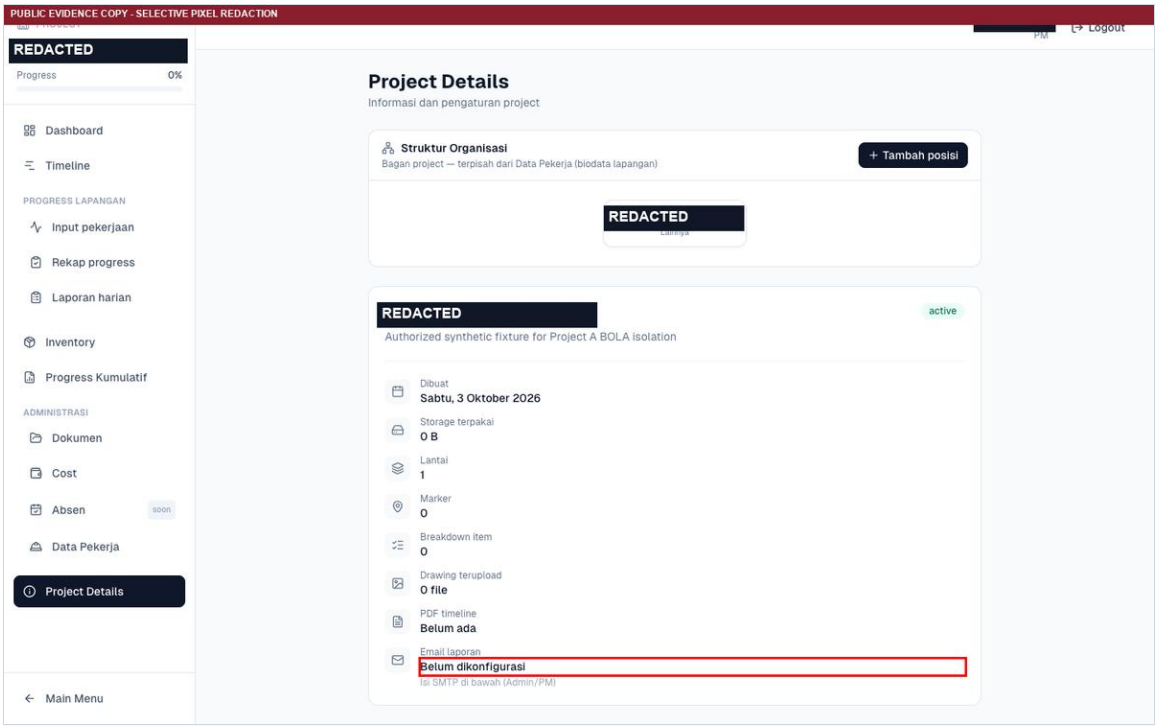


Figure 11 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 2

3. CLIENT-A requests Project B. The screenshot shows the red-boxed foreign project identifier and the red-boxed HTTP 403 Forbidden response, proving the control holds for CLIENT.

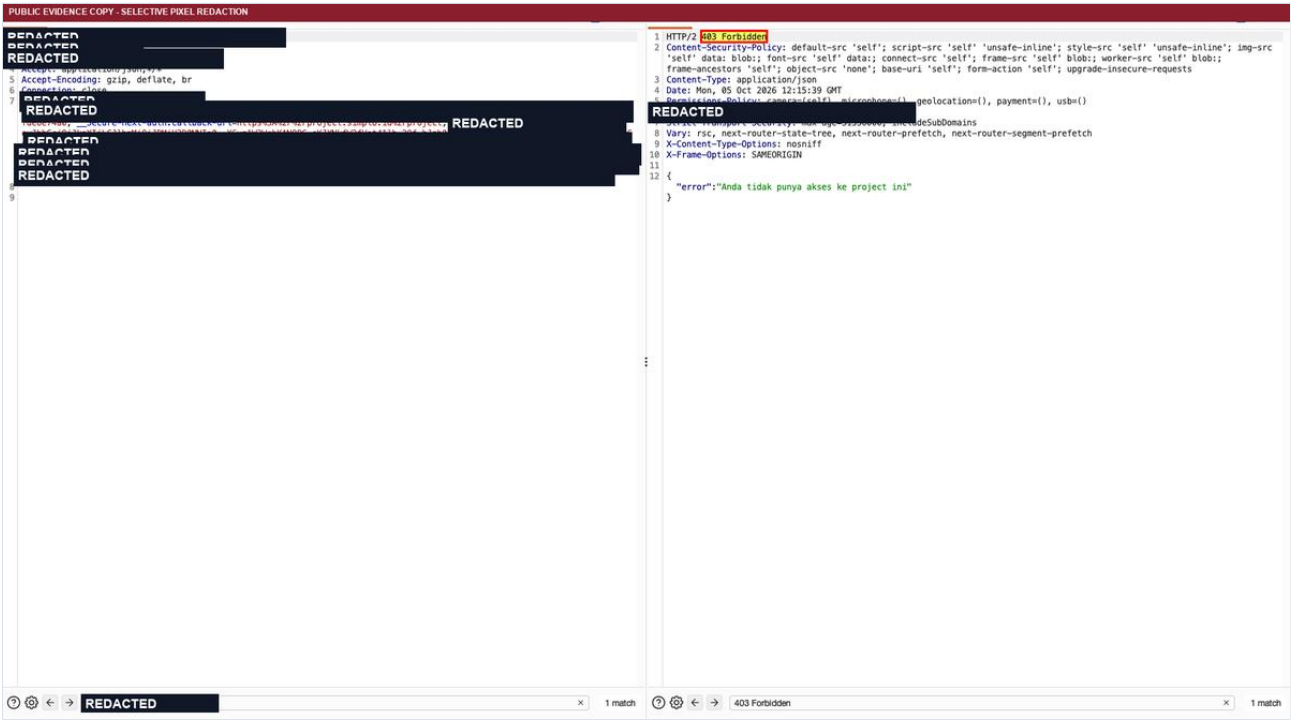


Figure 12 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 3

4. CLIENT-B requests Project A. The screenshot shows the mirrored red-boxed foreign identifier and 403 response, proving the control holds.

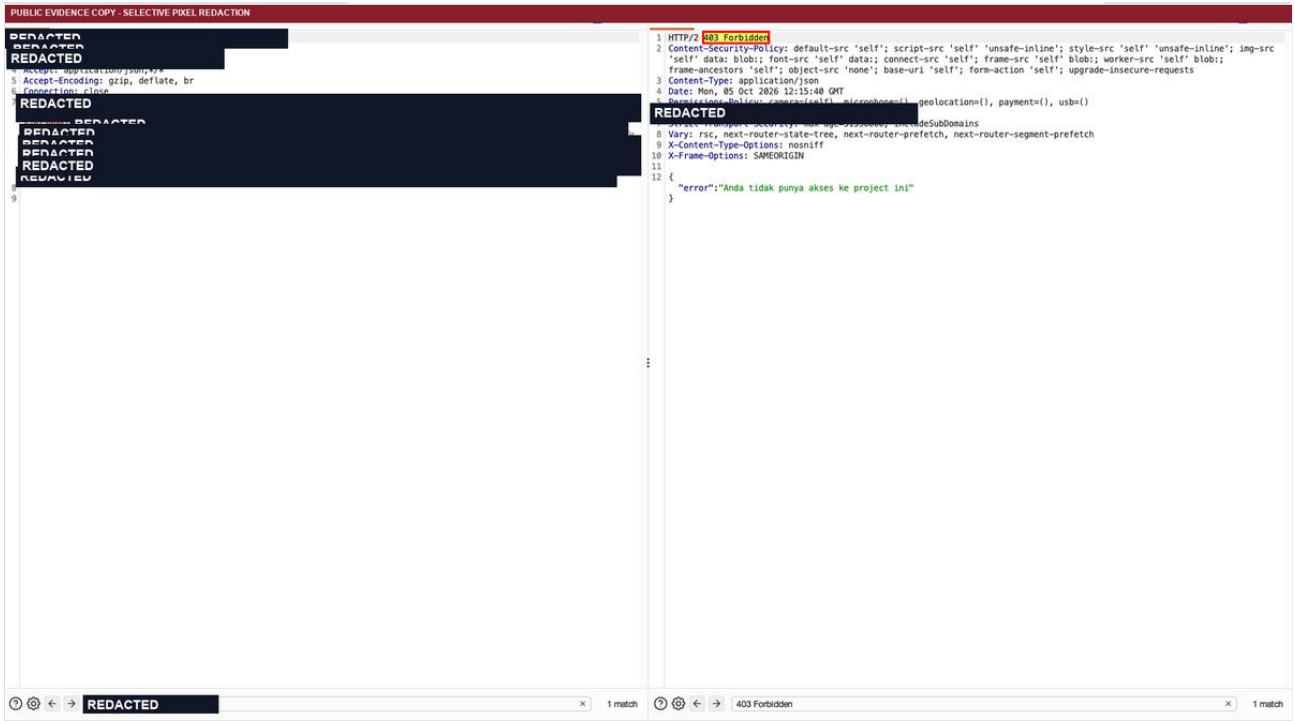


Figure 13 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 4

5. LEADER-A requests Project B. The screenshot shows the red-boxed foreign identifier and the red-boxed members roster in the HTTP 200 foreign project object, proving member PII disclosure.

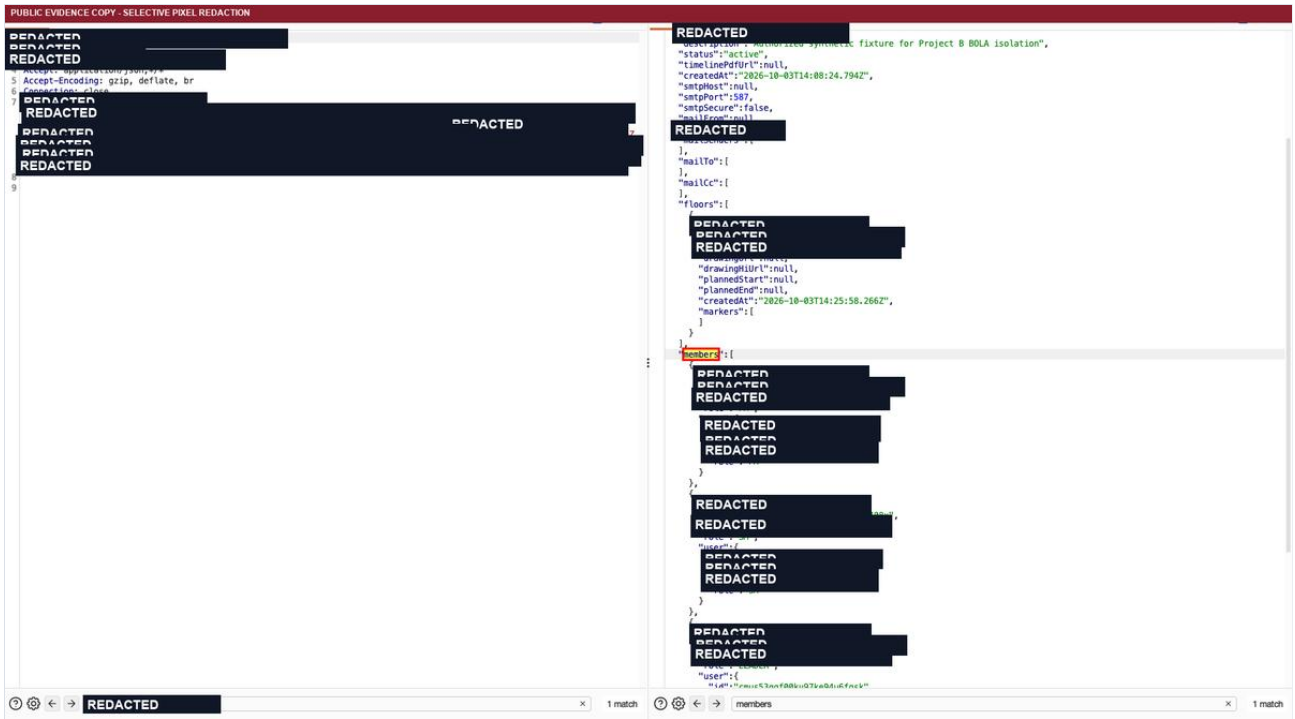


Figure 14 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 5

6. LEADER-B requests Project A. The screenshot shows the mirrored red-boxed foreign identifier and members roster, proving the bidirectional failure.

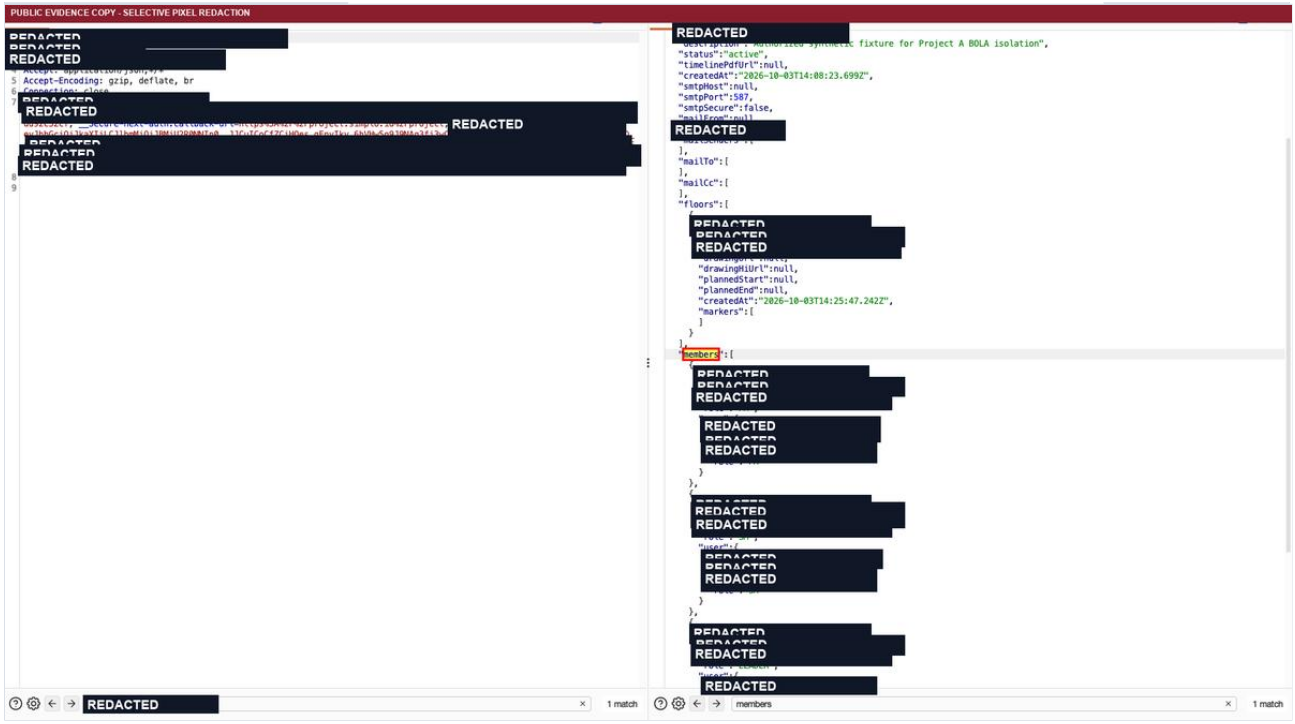


Figure 15 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 6

7. PM-A requests Project B. The screenshot shows the red-boxed foreign identifier and members roster in the HTTP 200 response, proving disclosure.

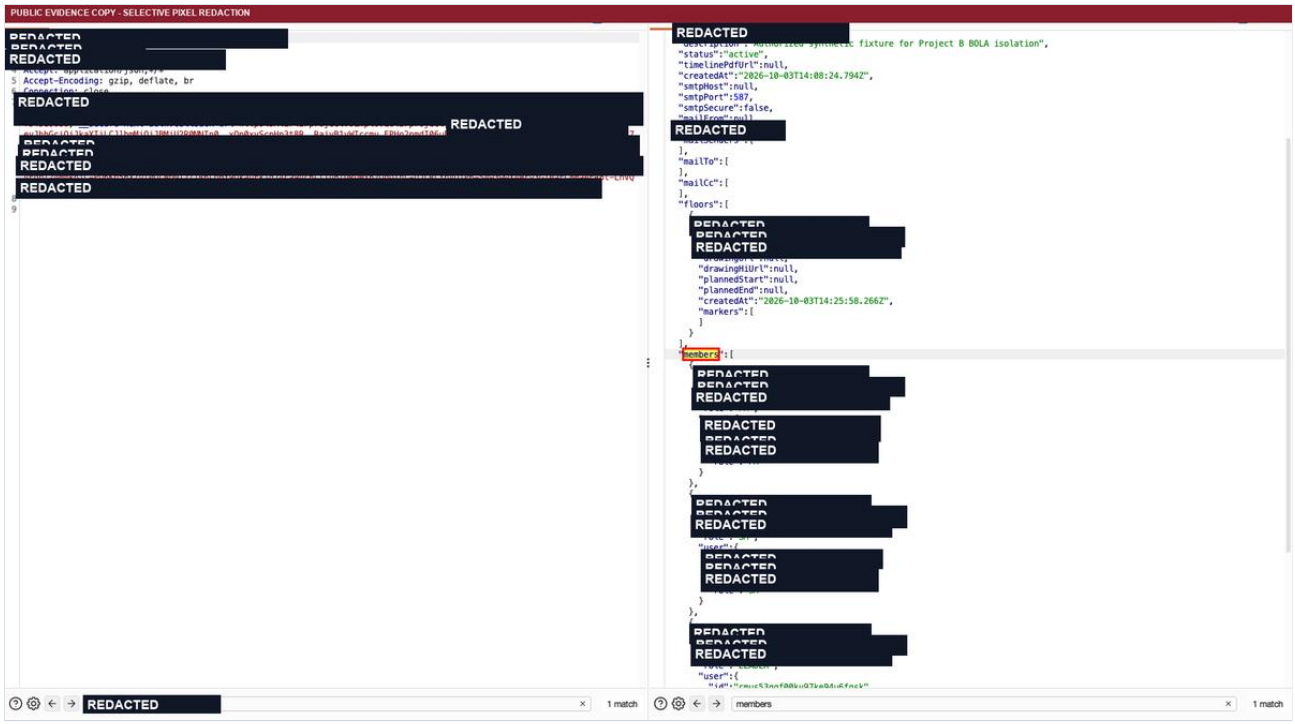


Figure 16 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 7

8. PM-B requests Project A. The screenshot shows the mirrored evidence, proving the bidirectional failure.

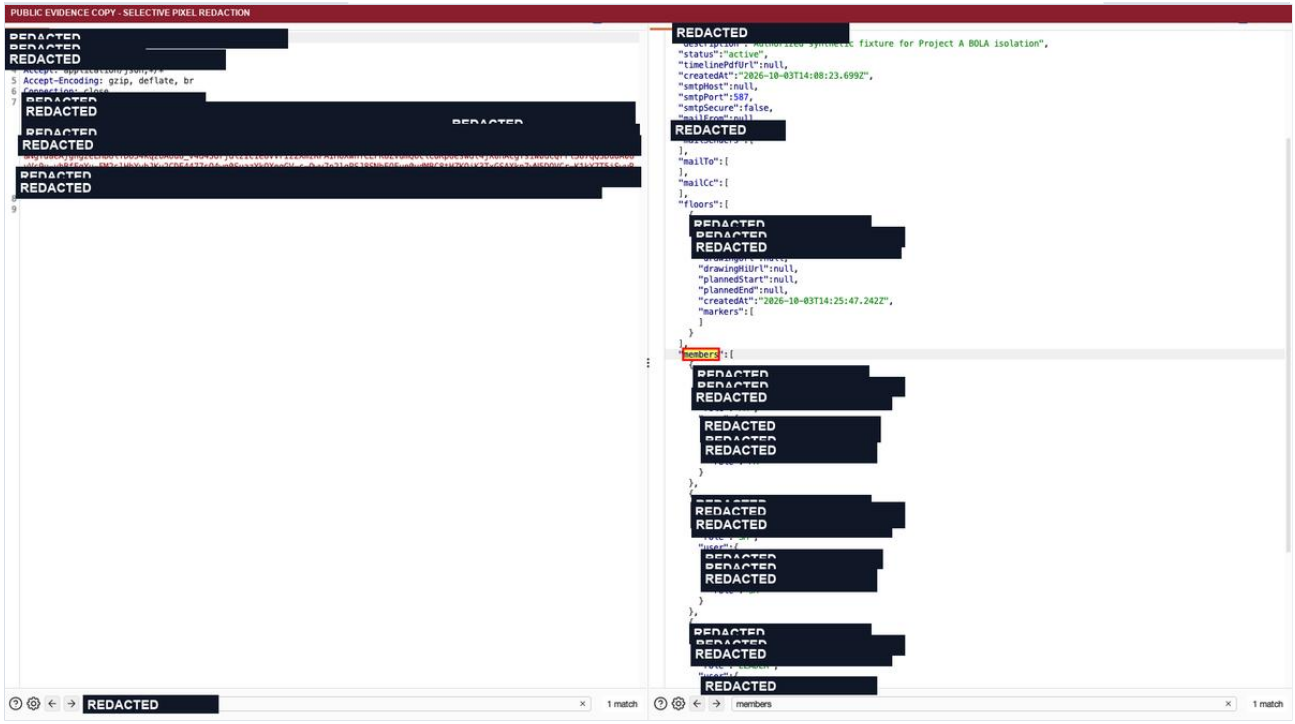


Figure 17 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 8

9. SM-A requests Project B. The screenshot shows the red-boxed foreign identifier and members roster in the HTTP 200 response, proving disclosure.

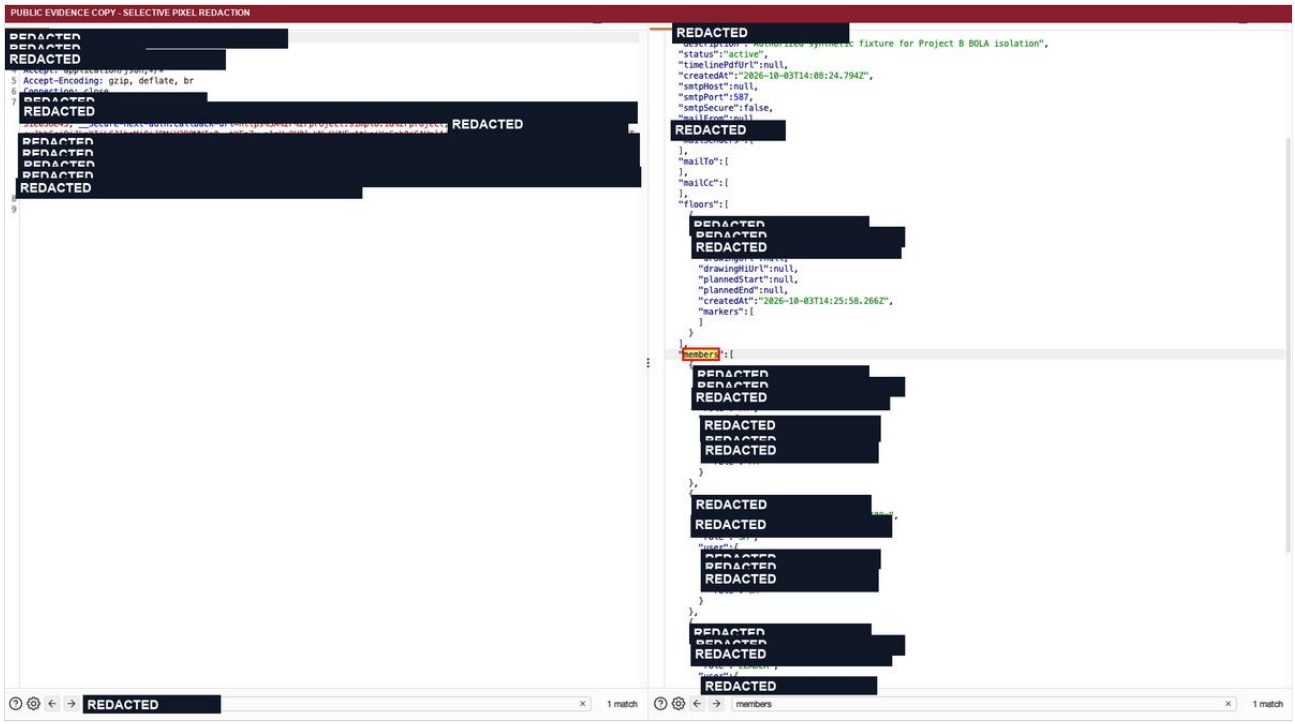


Figure 18 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 9

10. SM-B requests Project A. The screenshot shows the mirrored evidence, proving the bidirectional failure.

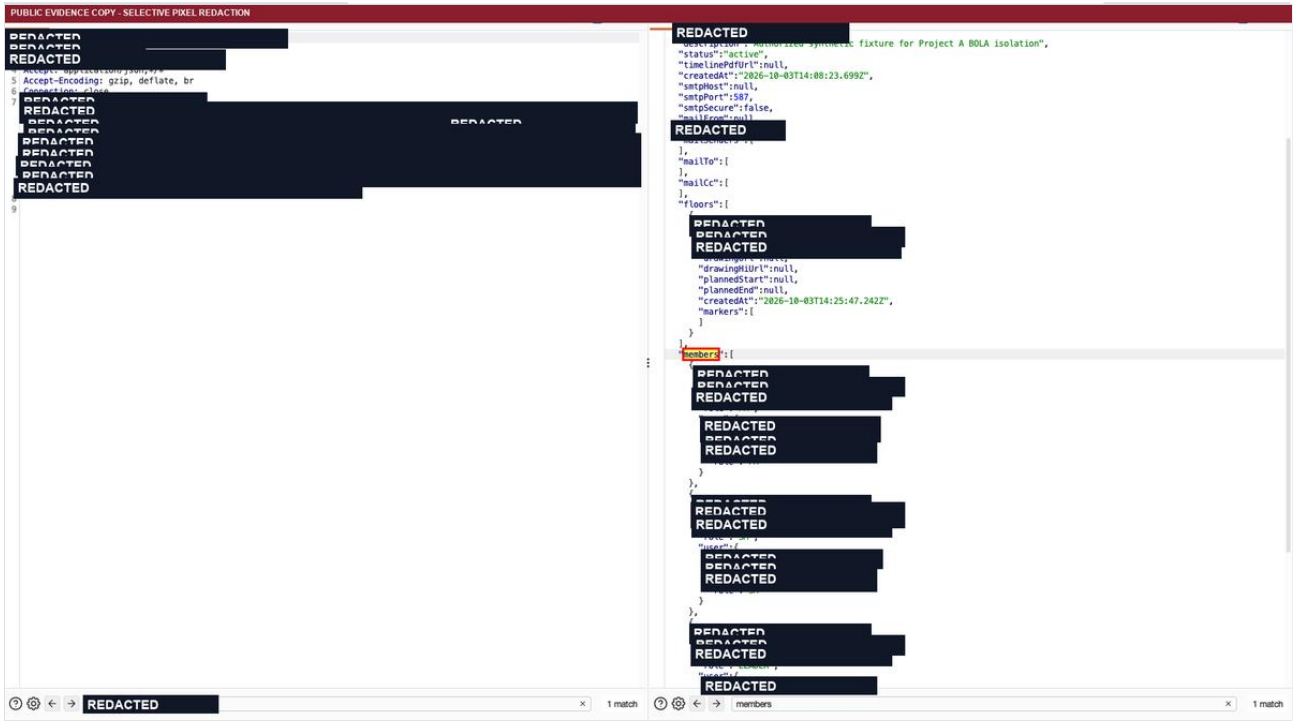


Figure 19 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 3 step 10

Affected scenario 3 — Data Pekerja - Worker List

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

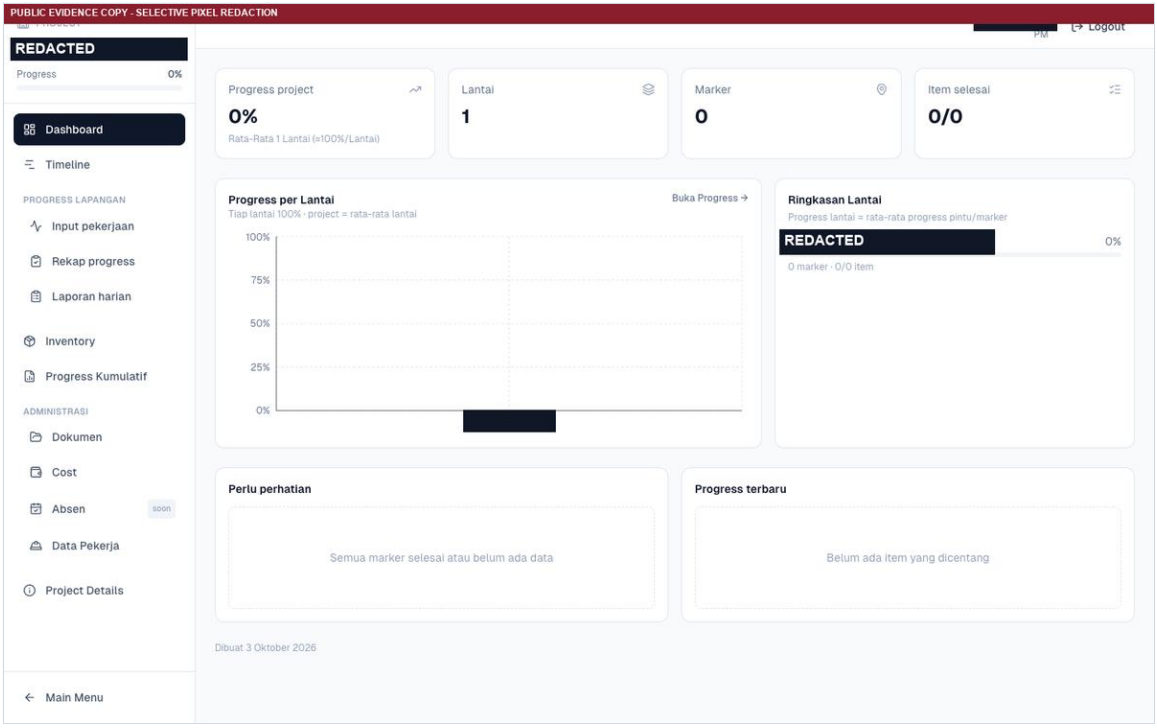


Figure 20 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 4 step 1

2. Click Data Pekerja in the sidebar, then open the Semua tab. The screenshot shows the worker roster with the [TEST FIXTURE] row, establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

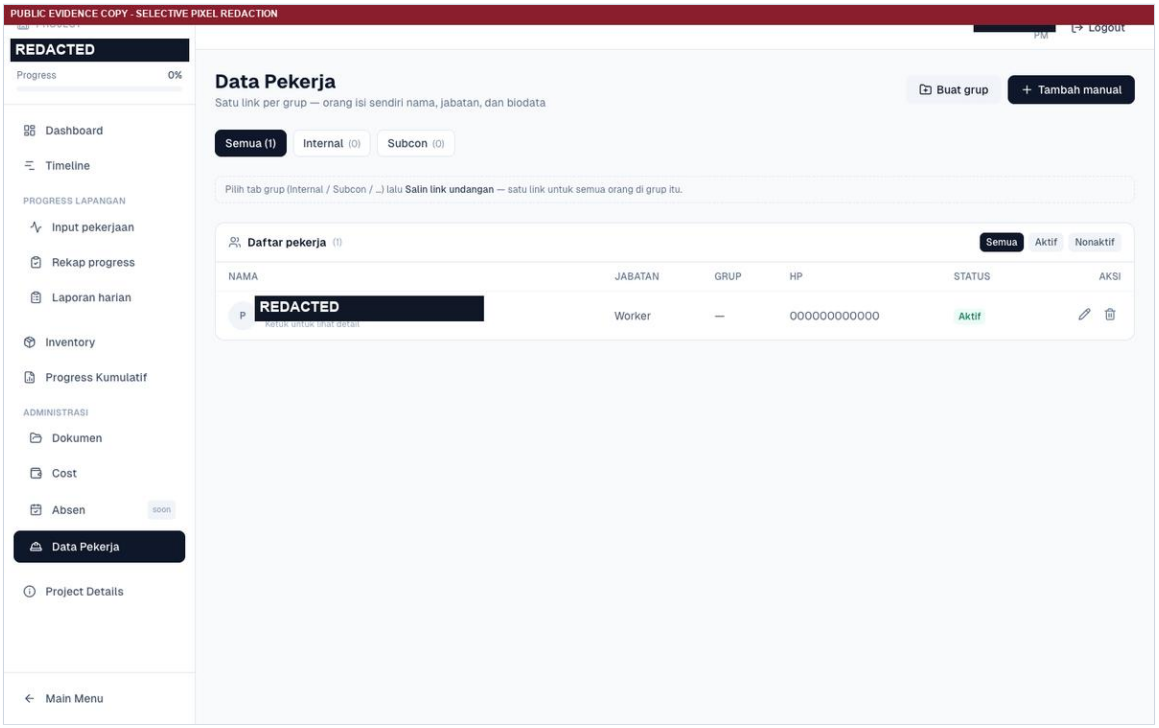


Figure 21 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 4 step 2

3. PM-A requests Project B's worker list. The screenshot shows the red-boxed foreign project identifier and the red-boxed nik field in the HTTP 200 foreign roster, proving disclosure.

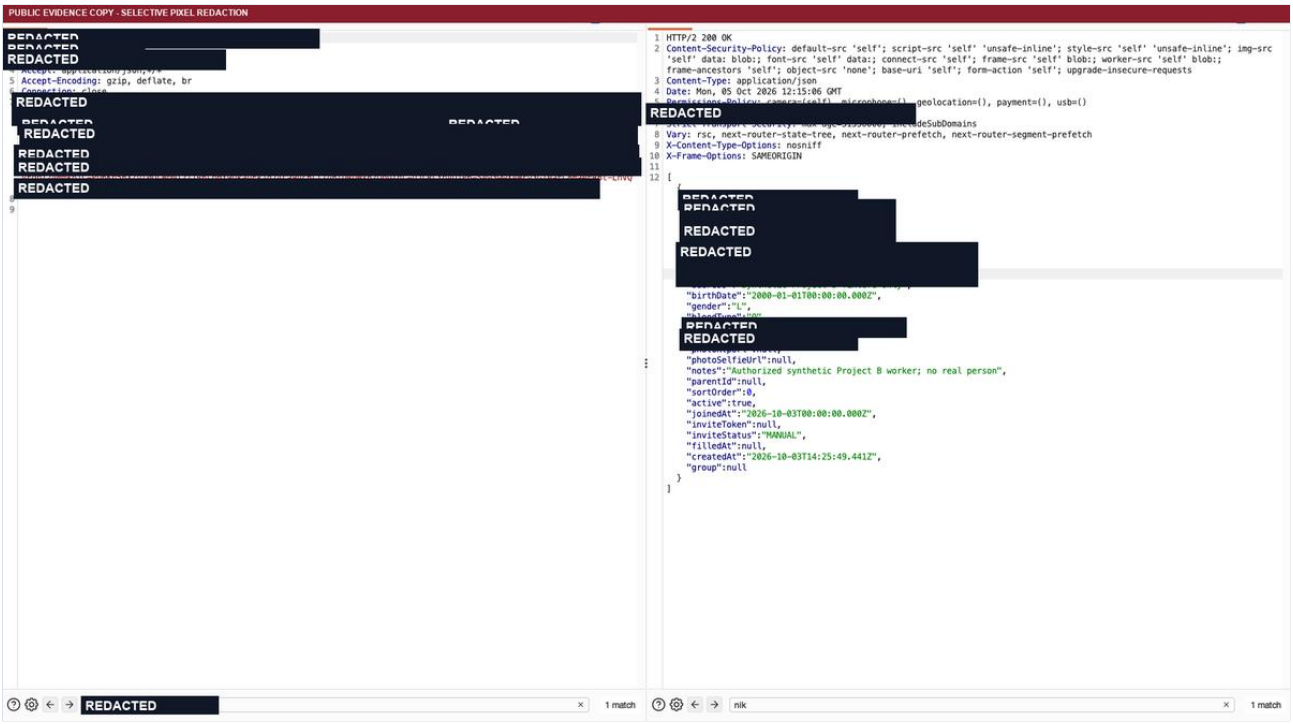


Figure 22 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 4 step 3

4. PM-B requests Project A's worker list. The screenshot shows the mirrored red-boxed foreign identifier and nik field, proving the bidirectional failure.

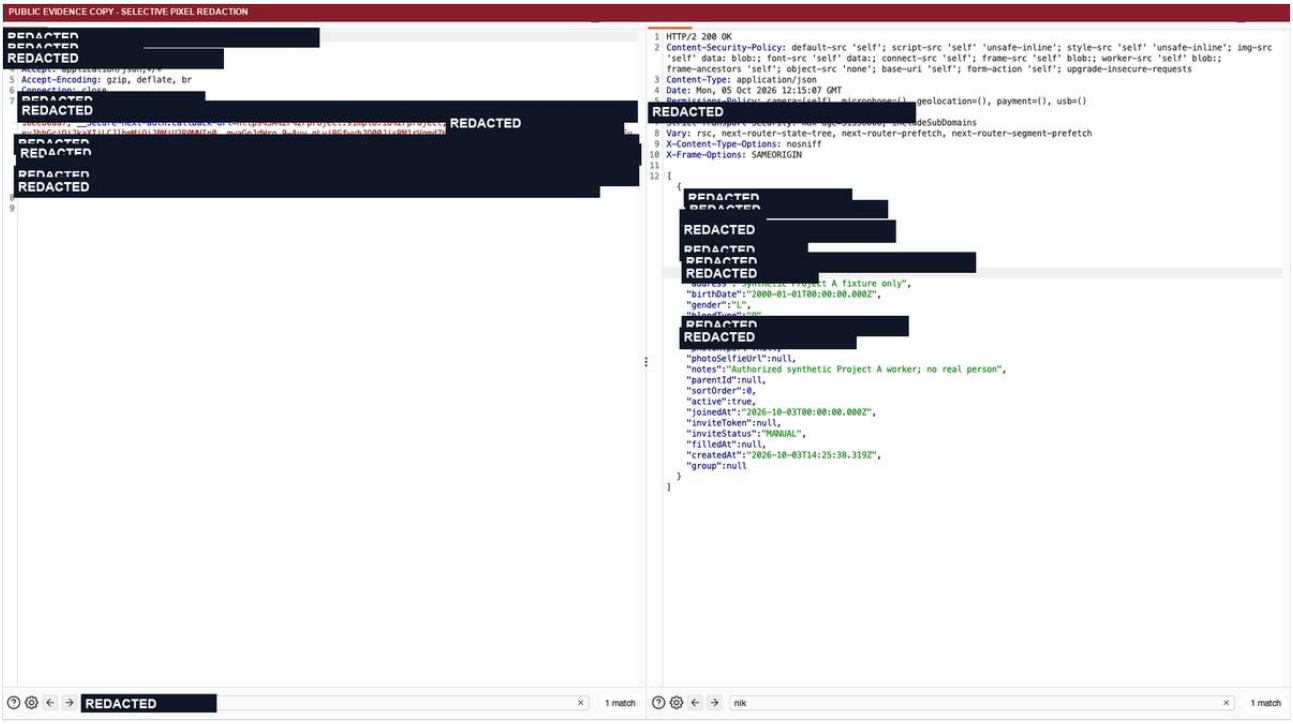


Figure 23 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 4 step 4

Affected scenario 4 — Inventory - Inventory Items

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

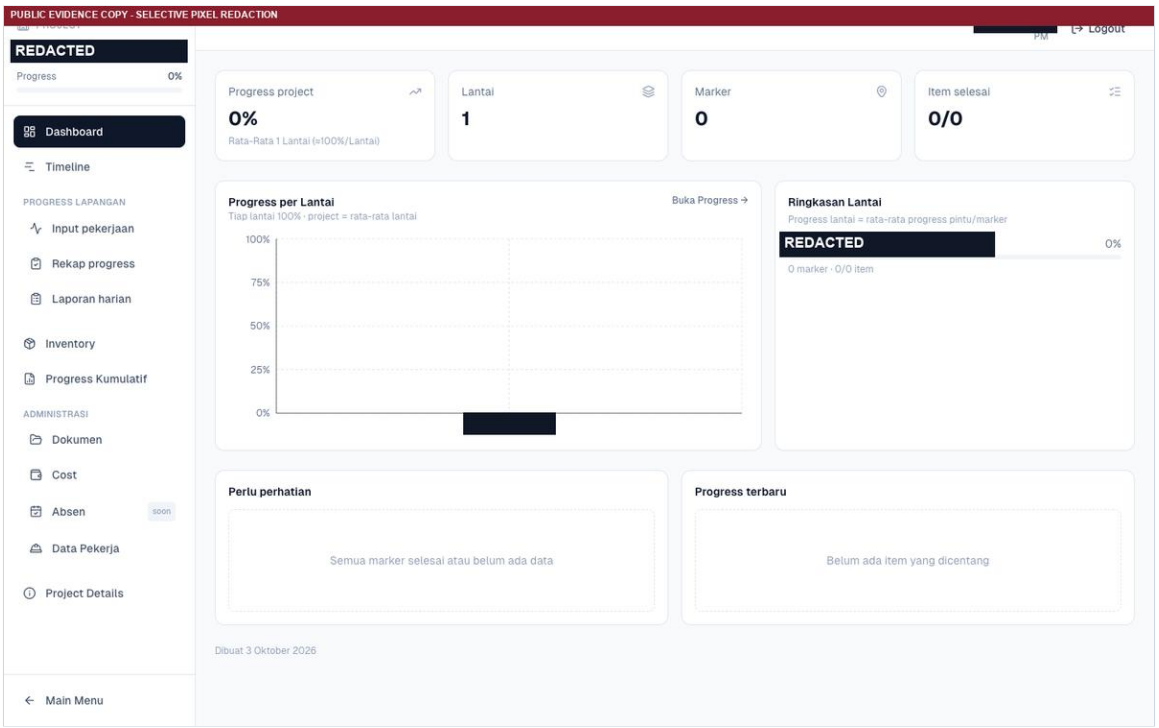


Figure 24 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 5 step 1

2. Click Inventory in the sidebar navigation. The screenshot shows the Inventory page with the project's own data visible ([TEST FIXTURE] stock row), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

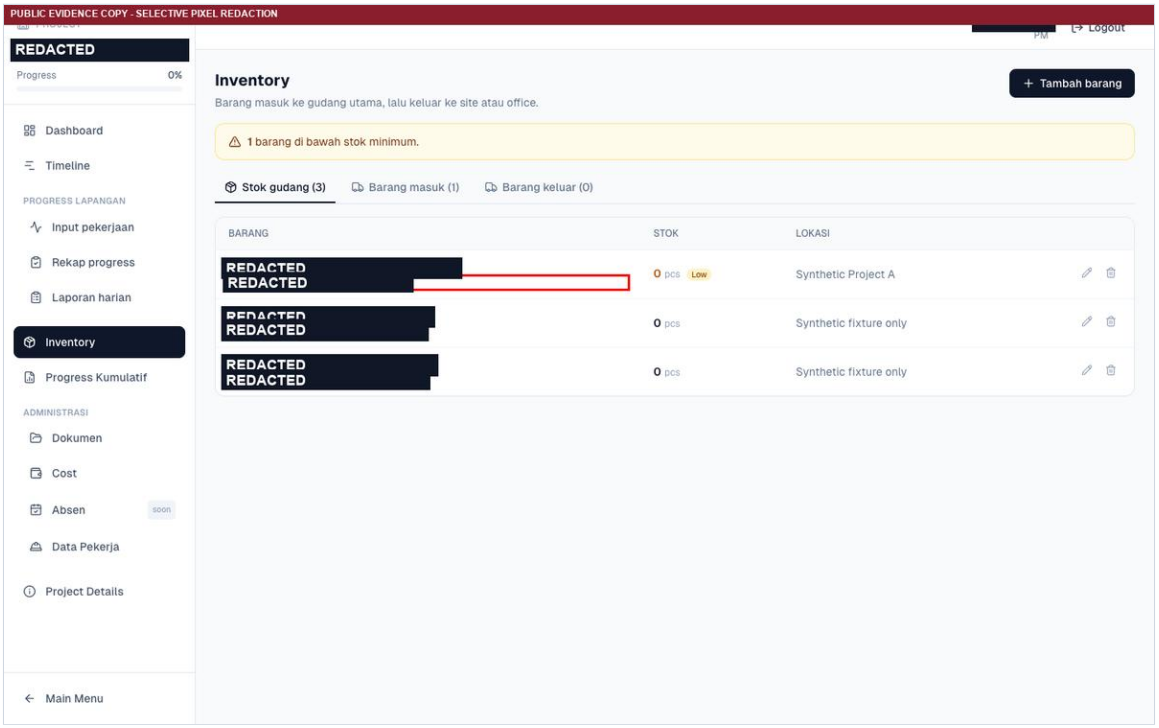


Figure 25 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 5 step 2

3. PM-A requests Project B's inventory. The screenshot shows the red-boxed foreign project identifier and the red-boxed [TEST FIXTURE] marker rows in the HTTP 200 foreign collection, proving disclosure.

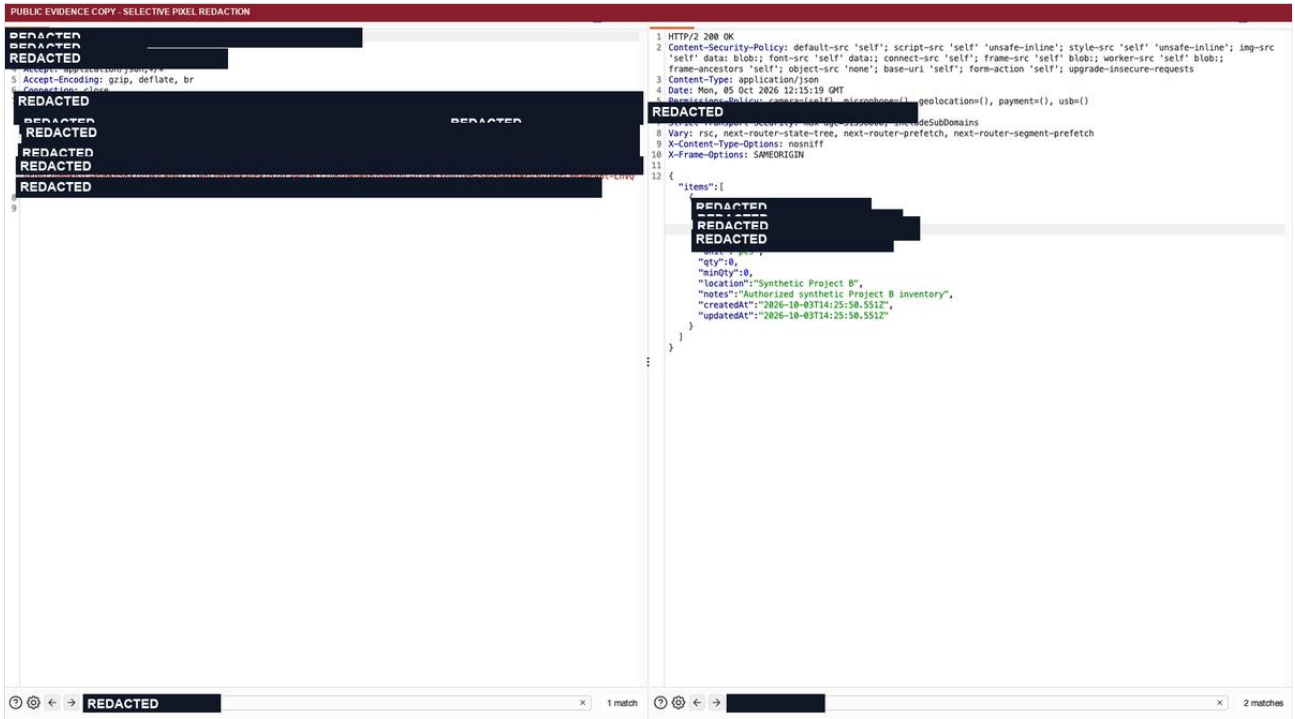


Figure 26 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 5 step 3

4. PM-B requests Project A's inventory. The screenshot shows the mirrored red-boxed foreign identifier and marker rows, proving the bidirectional failure.

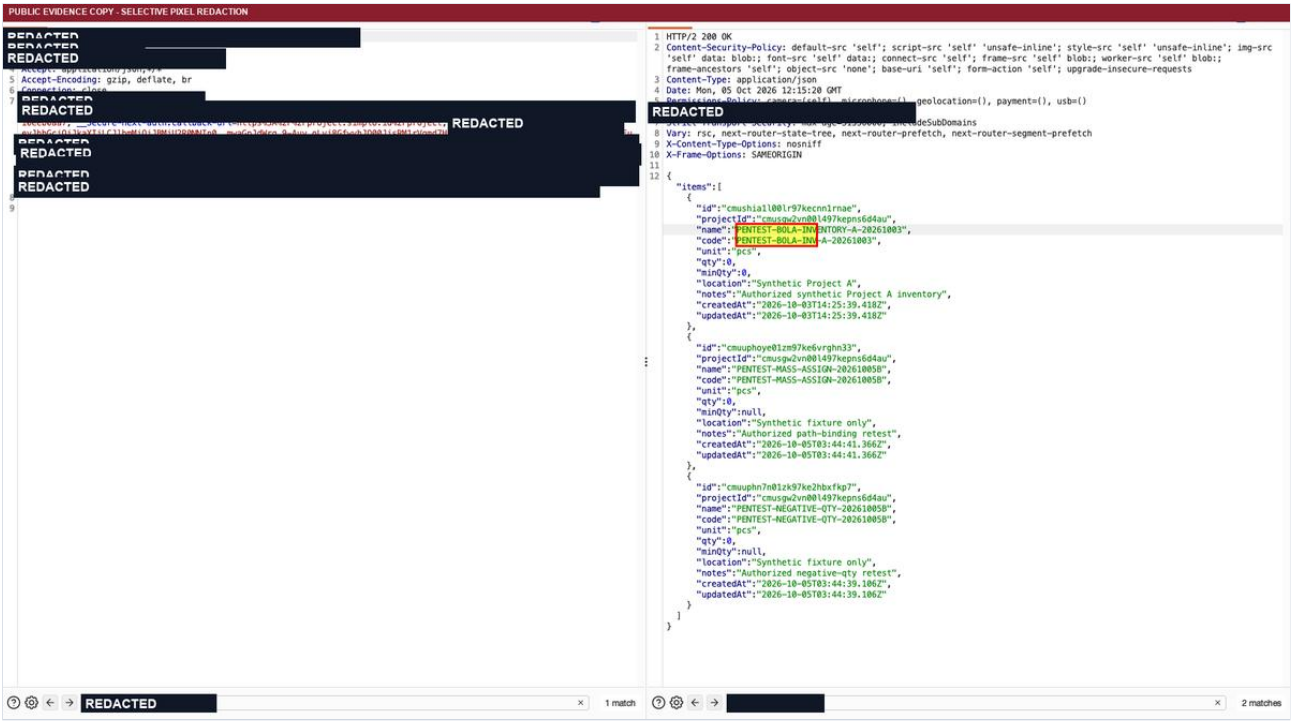


Figure 27 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 5 step 4

Affected scenario 5 — Cost - Cost Categories

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

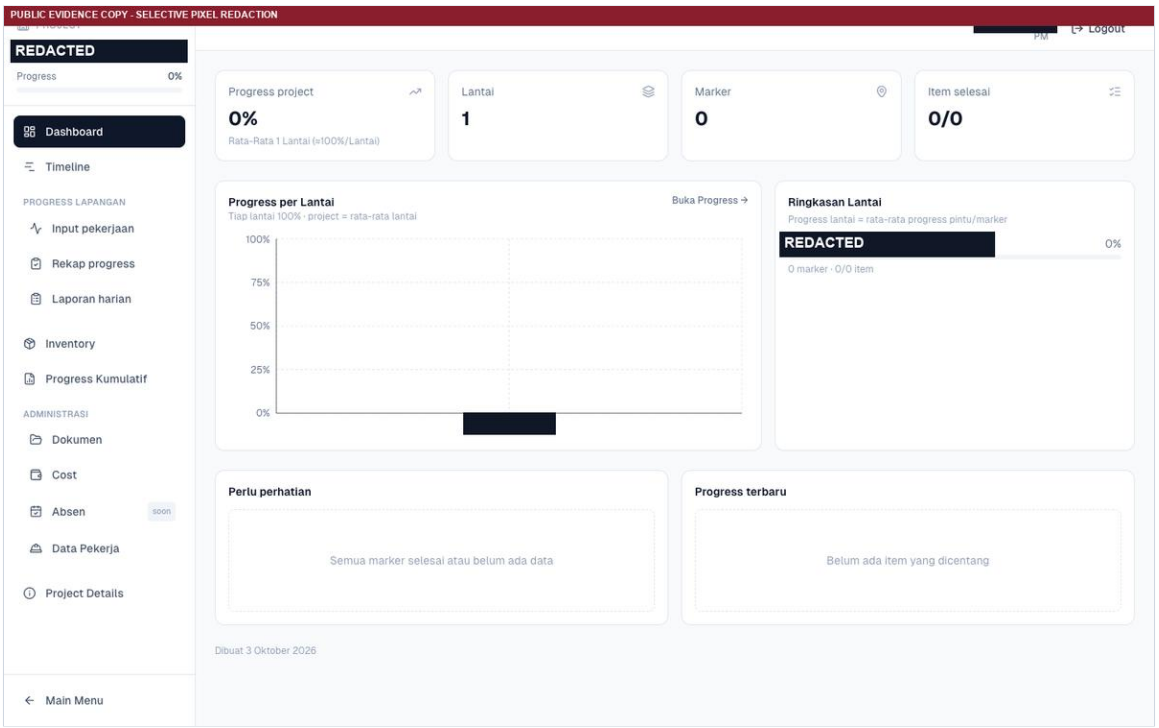


Figure 28 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 6 step 1

2. Click Cost in the sidebar navigation. The screenshot shows the Cost page with the project's own data visible (Ringkasan budget summary), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

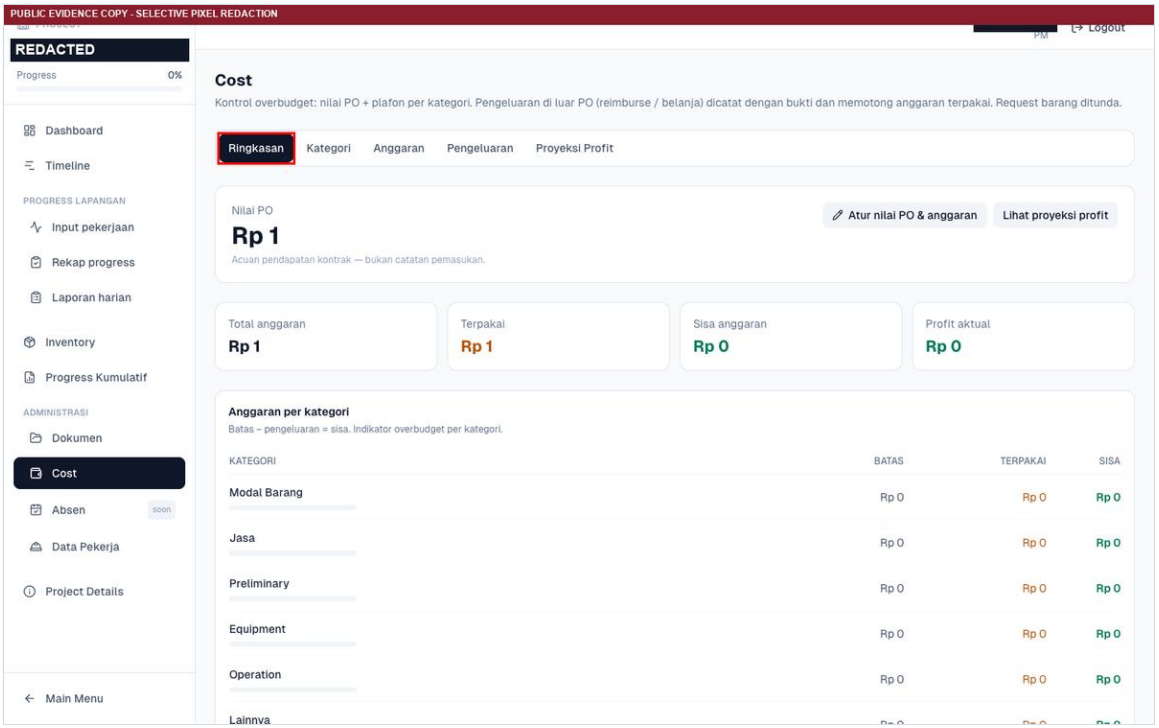


Figure 29 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 6 step 2

3. PM-A requests Project B's cost categories. The screenshot shows the red-boxed foreign project identifier in the request and red-boxed foreign projectId bindings on every returned row in the HTTP 200 response, proving foreign category disclosure.

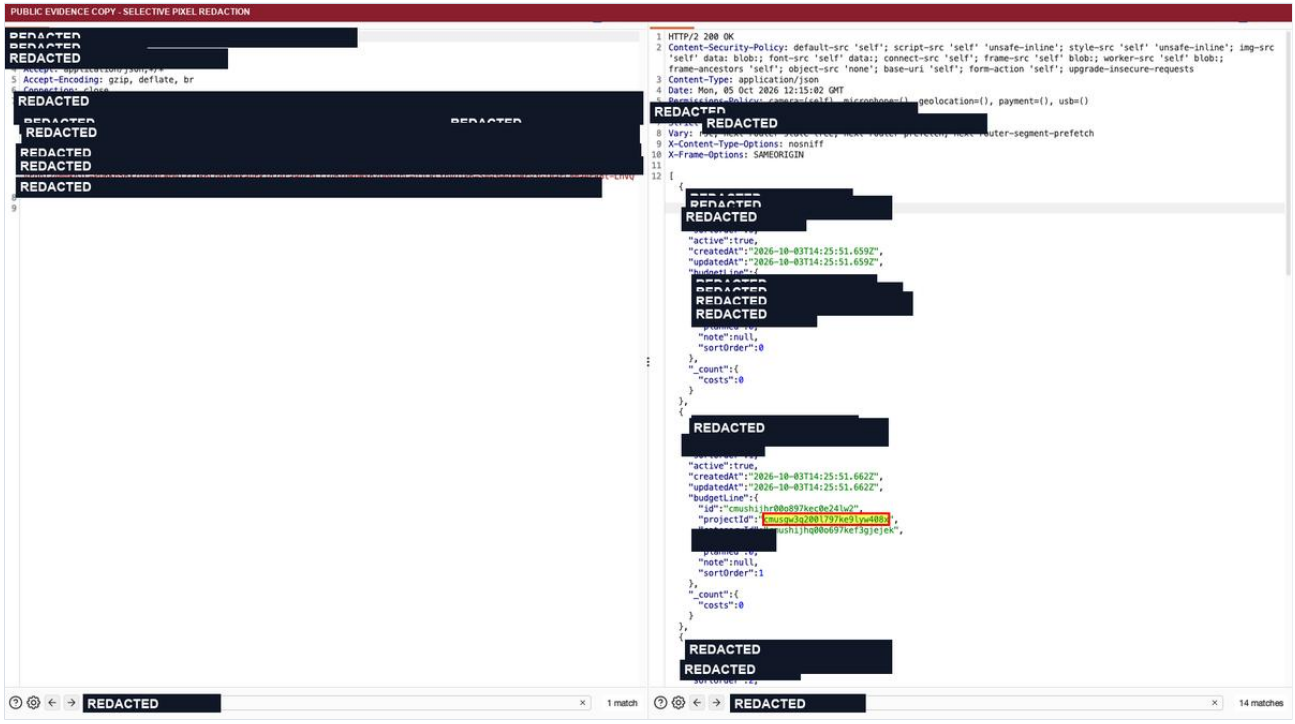


Figure 30 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 6 step 3

4. PM-B requests Project A's cost categories. The screenshot shows the mirrored red-boxed foreign identifier and foreign-bound rows in the HTTP 200 response, proving the bidirectional failure.

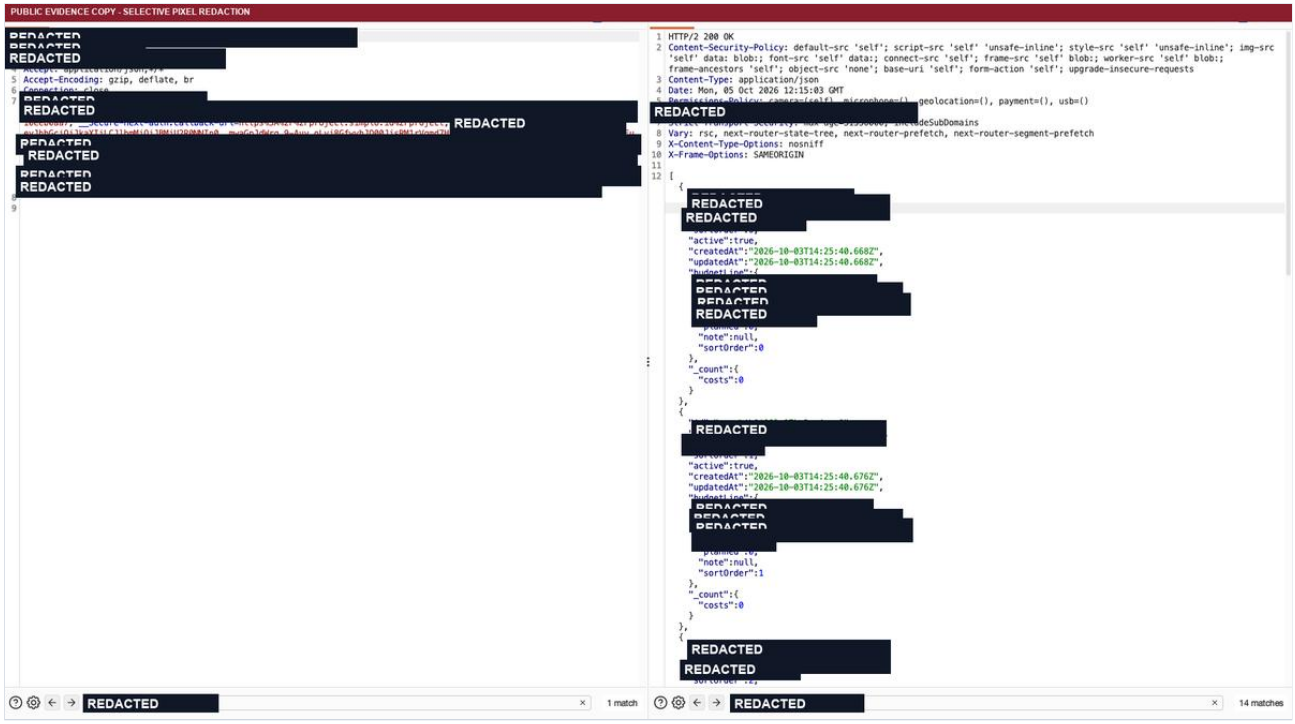


Figure 31 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 6 step 4

Affected scenario 6 — Budget - Financial Lines

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

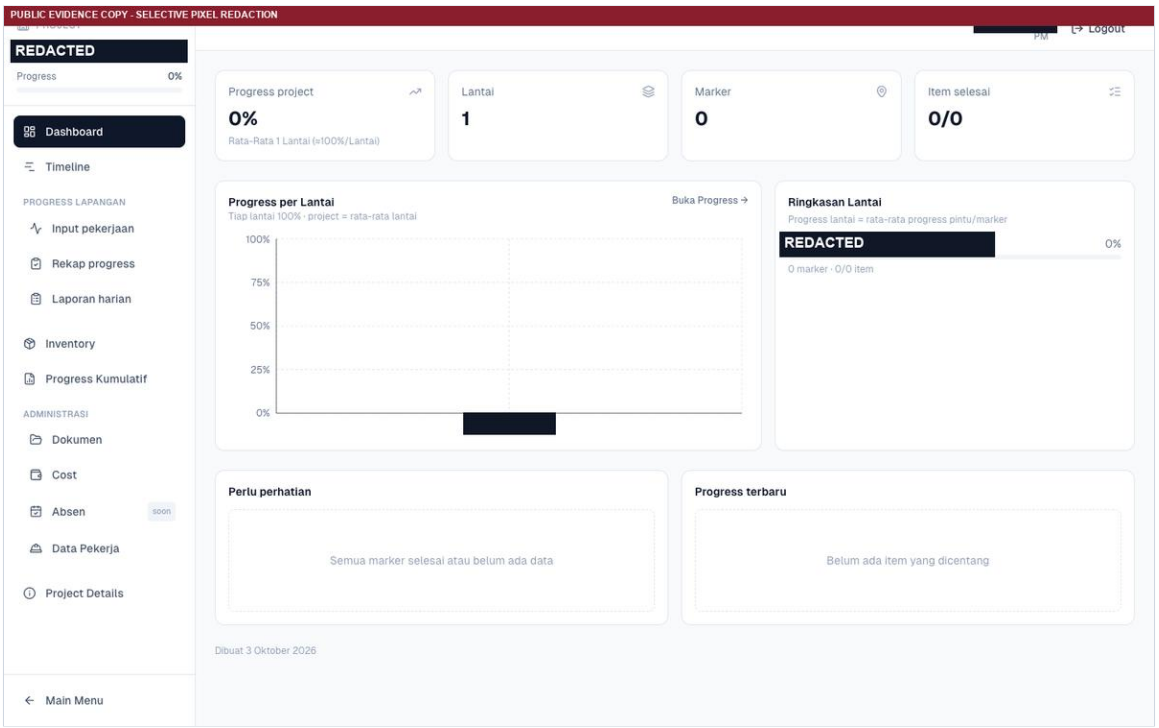


Figure 32 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 7 step 1

2. Click Cost in the sidebar navigation. The screenshot shows the Cost page with the project's own data visible (Ringkasan budget summary), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

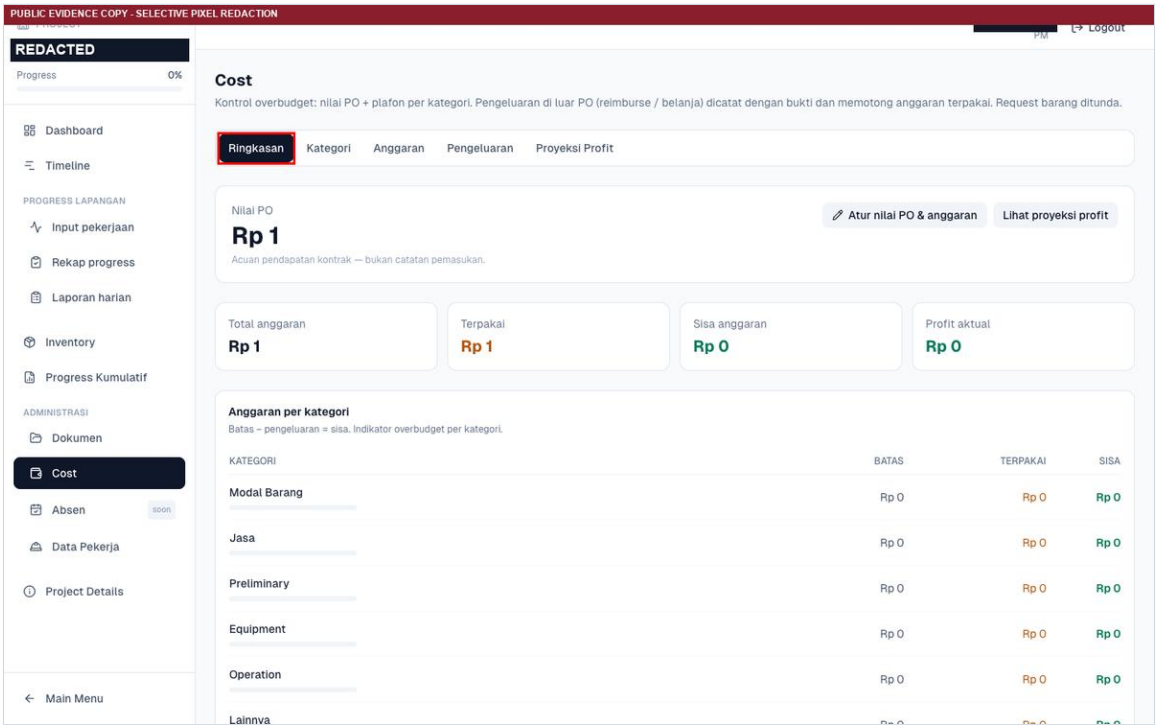


Figure 33 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 7 step 2

3. PM-A requests Project B's budget. The screenshot shows the red-boxed foreign project identifier in the request and the red-boxed totalValue field in the HTTP 200 response carrying Project B's financial payload, proving cross-project disclosure.

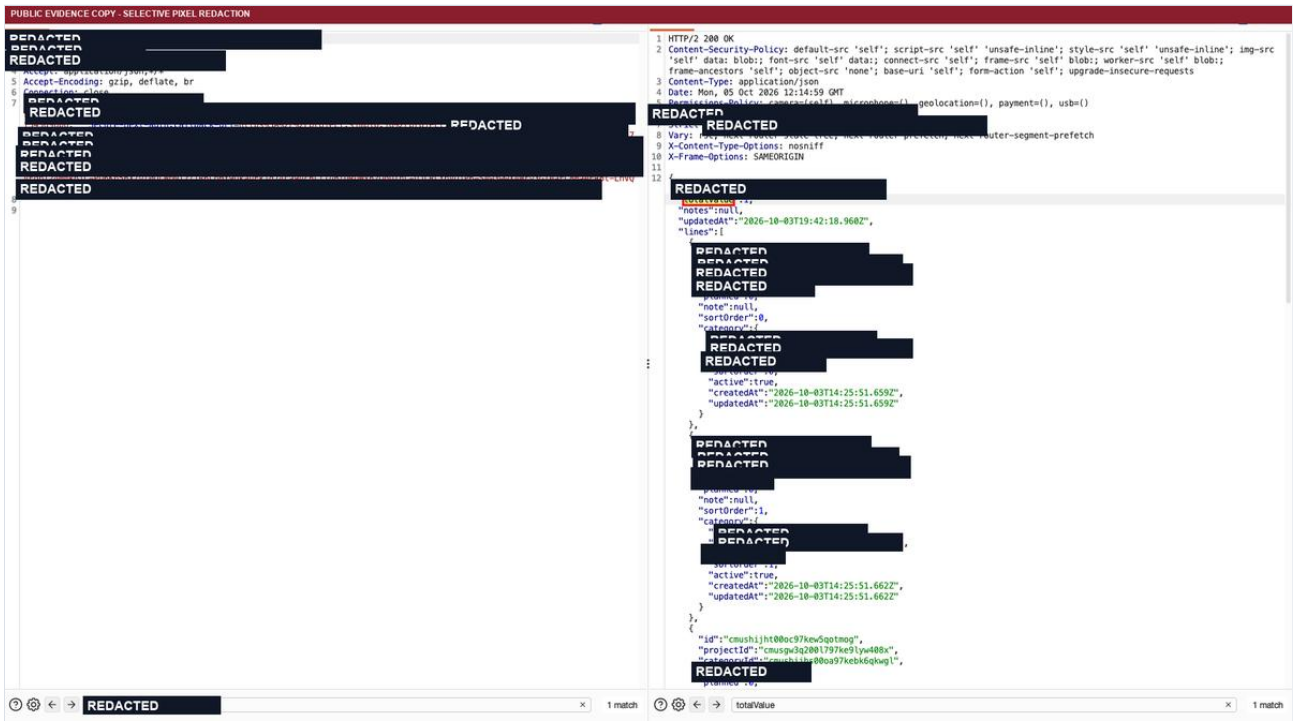


Figure 34 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 7 step 3

4. PM-B requests Project A's budget. The screenshot shows the red-boxed foreign project identifier and the red-boxed totalValue in the HTTP 200 response carrying Project A's financial payload, proving the bidirectional failure.

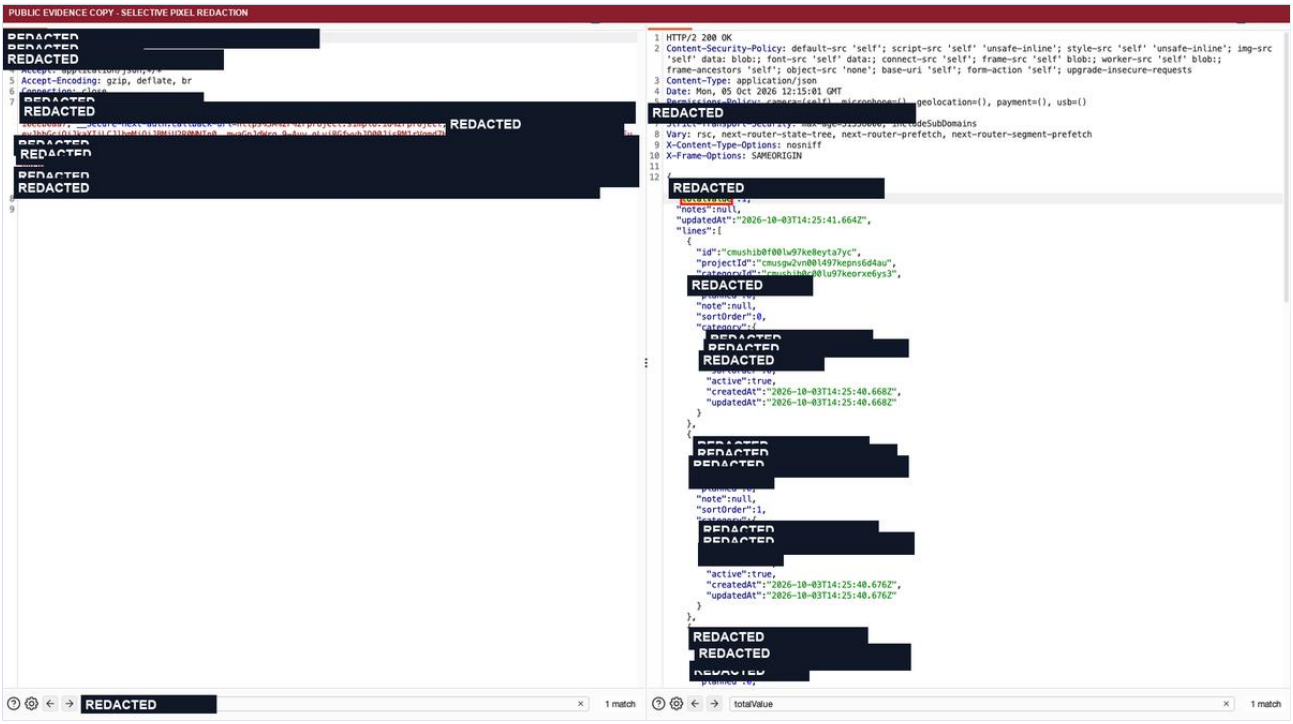


Figure 35 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 7 step 4

Affected scenario 7 — Cost - Cost Records

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

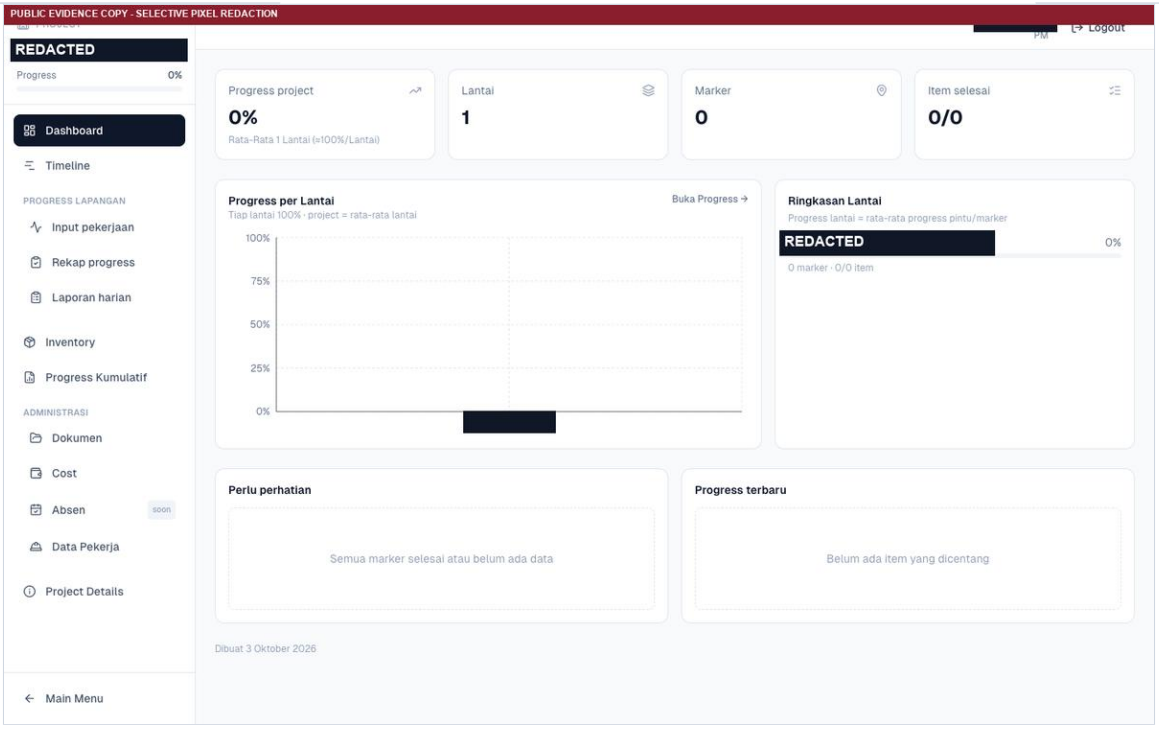


Figure 36 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 8 step 1

2. Click Cost in the sidebar navigation. The screenshot shows the Cost page with the project's own data visible (Ringkasan budget summary), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

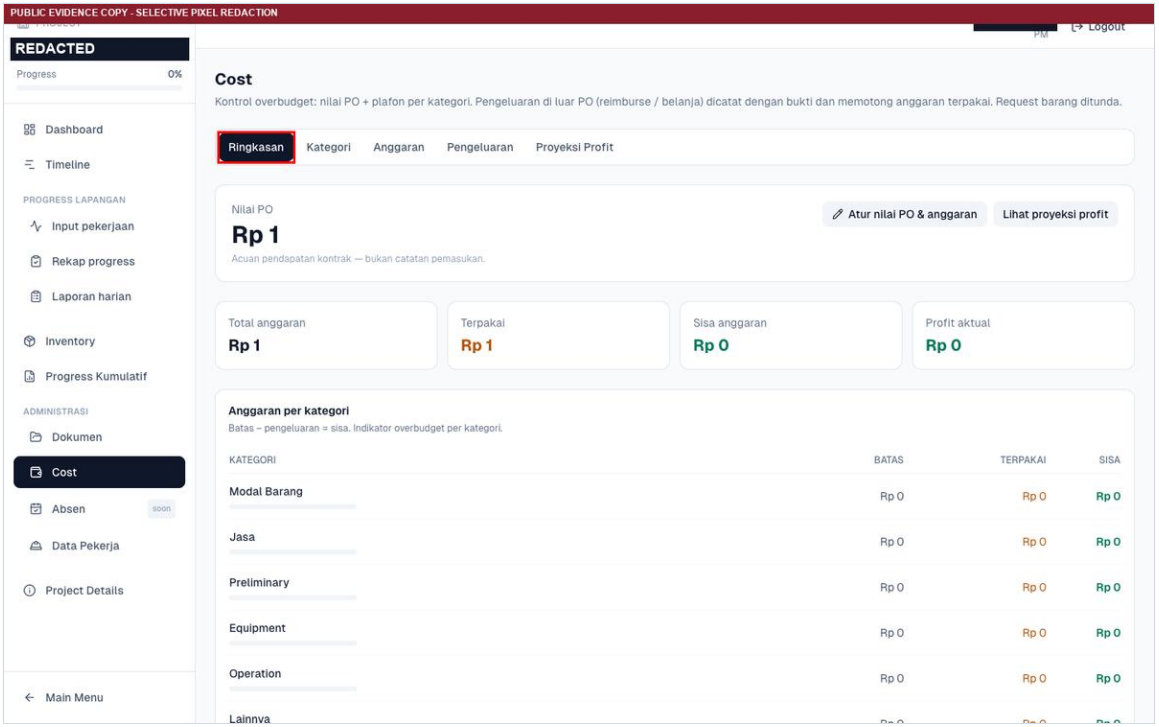


Figure 37 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 8 step 2

3. PM-A requests Project B's cost records. The screenshot shows the red-boxed foreign project identifier and the red-boxed receiptUrl in the HTTP 200 foreign financial record, proving disclosure including receipt metadata.

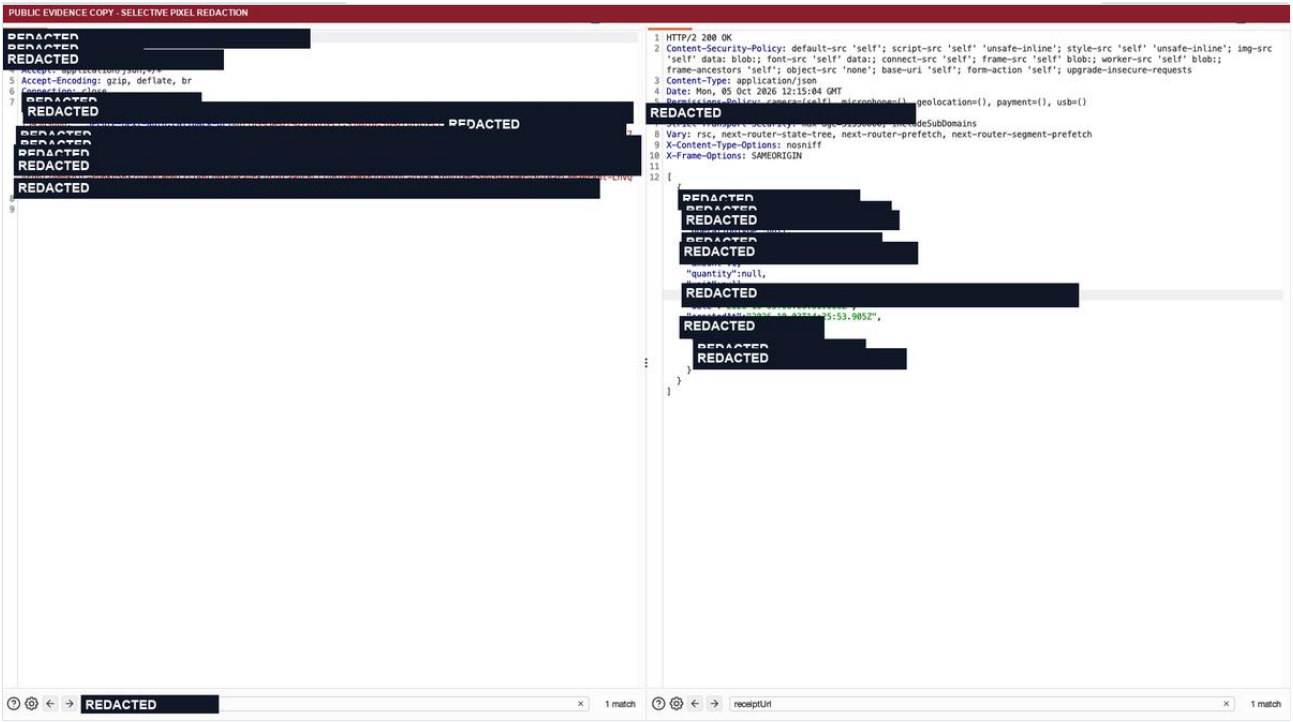


Figure 38 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 8 step 3

4. PM-B requests Project A's cost records. The screenshot shows the mirrored red-boxed foreign identifier and receiptUrl in the HTTP 200 response, proving the bidirectional failure.

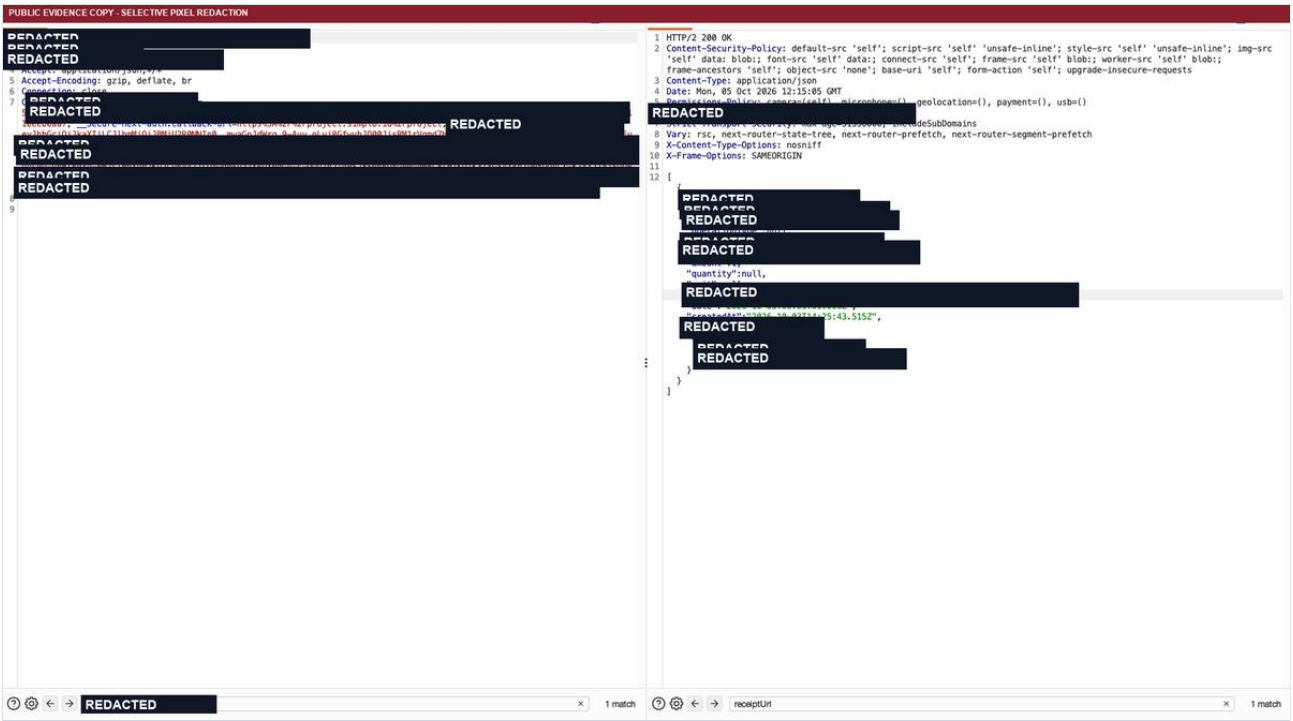


Figure 39 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 8 step 4

Affected scenario 8 — Dokumen - Document Metadata

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

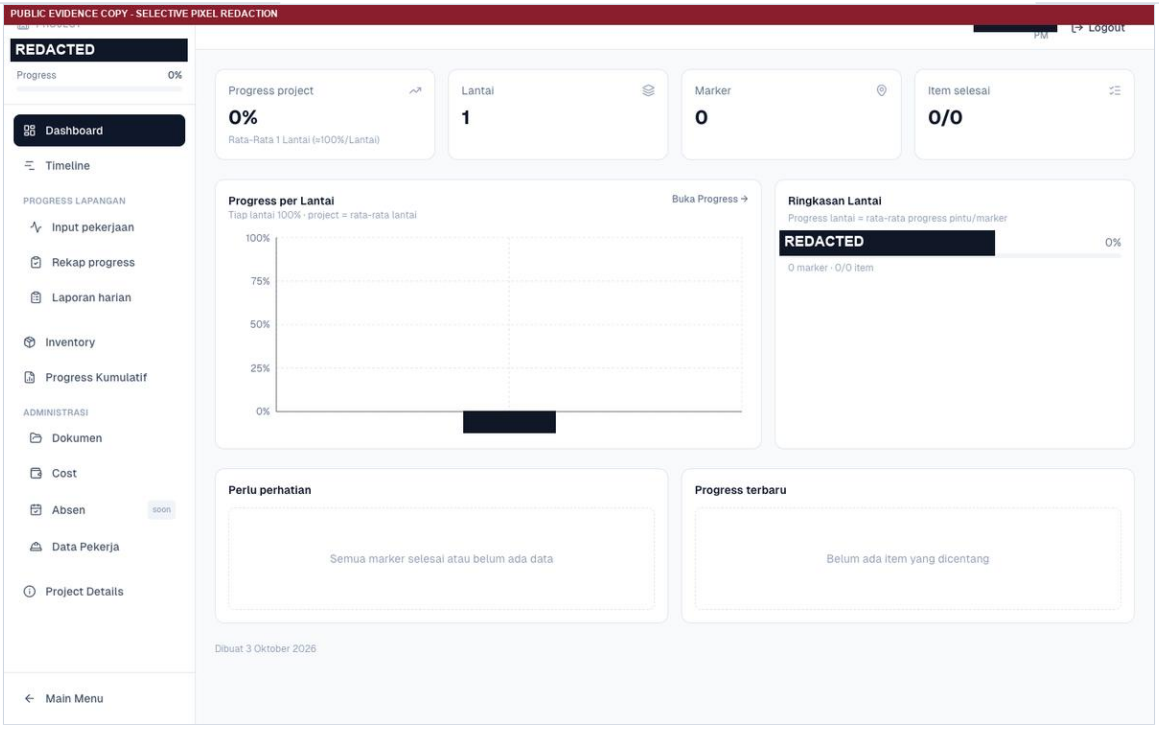


Figure 40 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 9 step 1

2. Click Dokumen in the sidebar navigation. The screenshot shows the Dokumen page with the project's own data visible ([TEST FIXTURE] file row), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

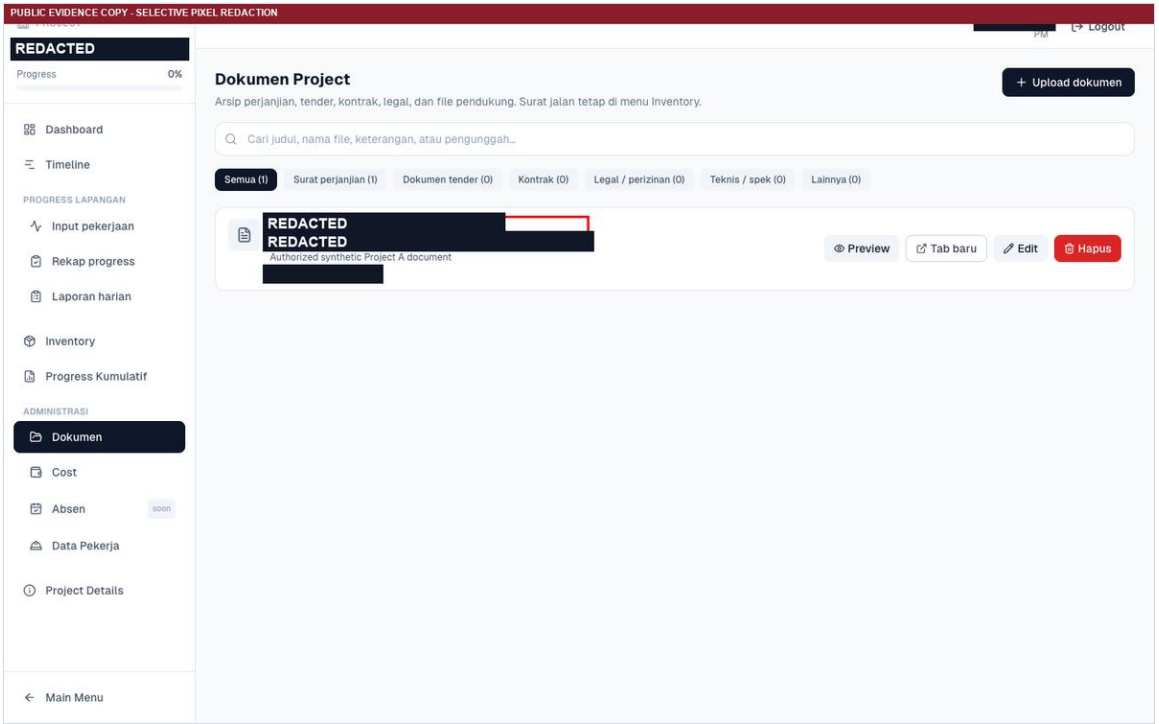


Figure 41 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 9 step 2

3. PM-A requests Project B's document list. The screenshot shows the red-boxed foreign project identifier and the red-boxed fileUrl in the HTTP 200 foreign metadata, proving disclosure including download URLs.

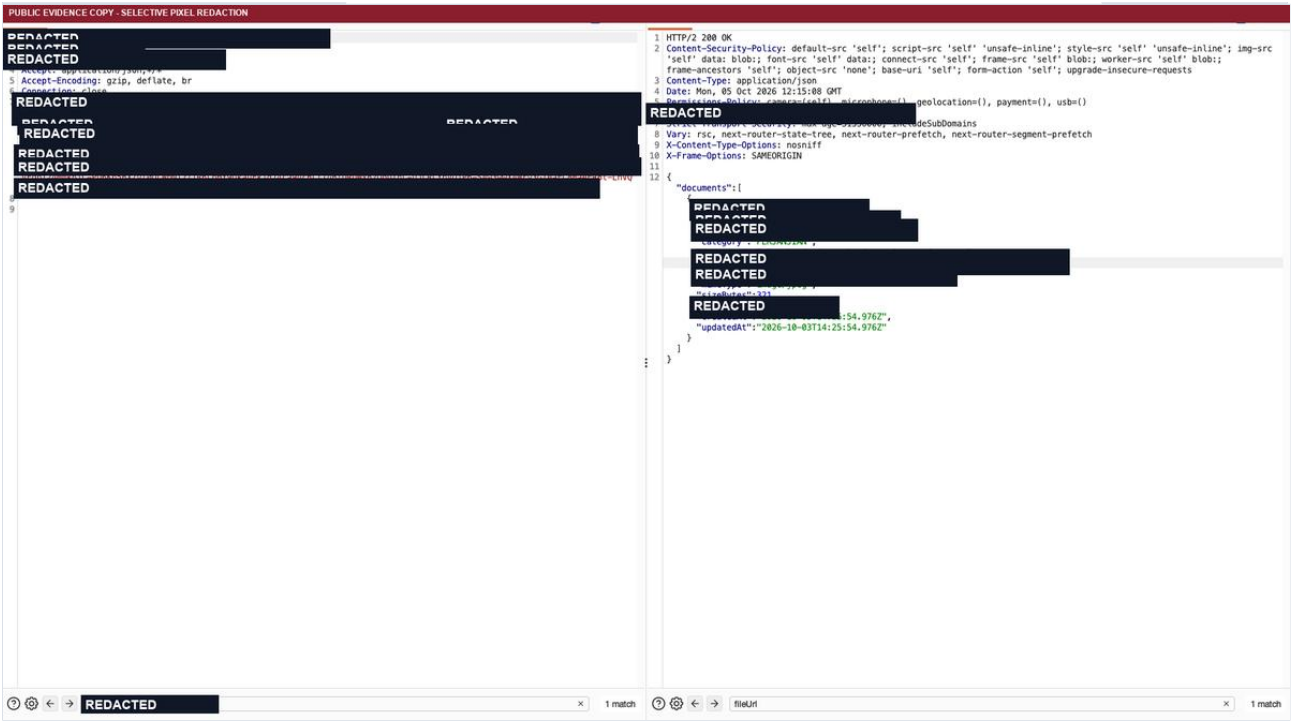


Figure 42 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 9 step 3

4. PM-B requests Project A's document list. The screenshot shows the mirrored red-boxed foreign identifier and fileUrl, proving the bidirectional failure.

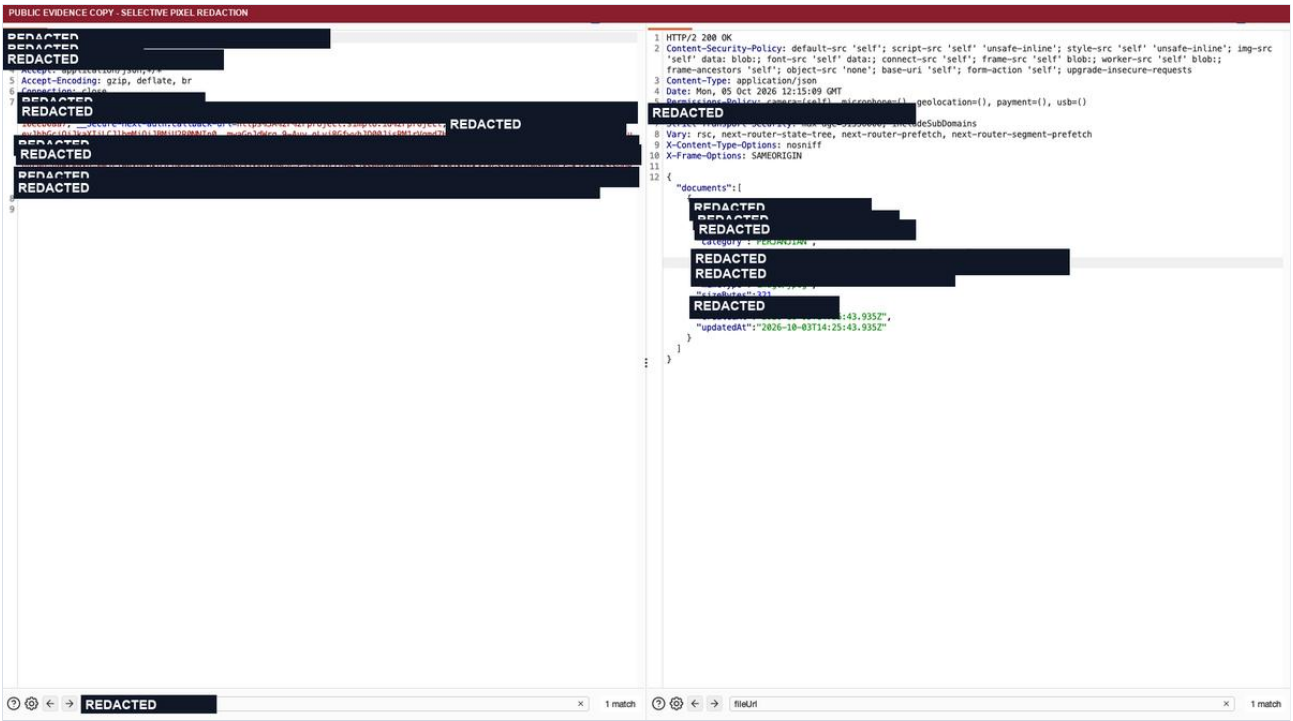


Figure 43 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 9 step 4

Affected scenario 9 — Purchase Order - Purchase Orders

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

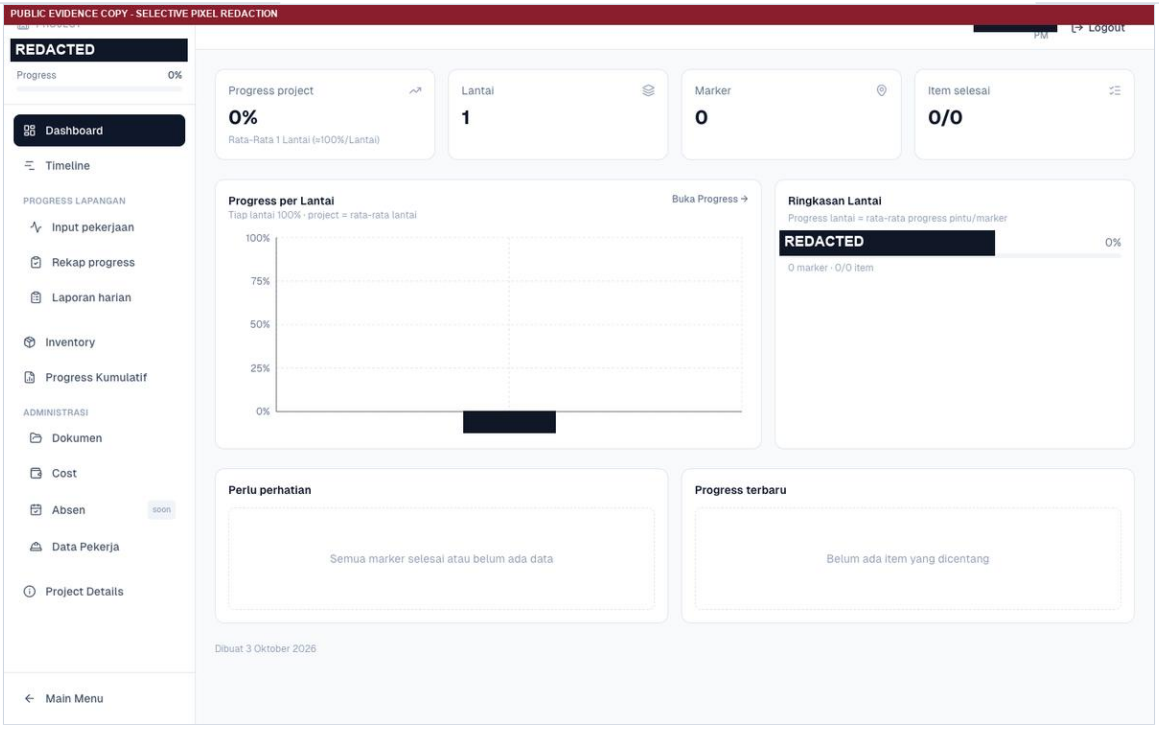


Figure 44 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 10 step 1

2. Click Progress Kumulatif in the sidebar navigation. The screenshot shows the PO page with the project's own data visible ([TEST FIXTURE] record), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

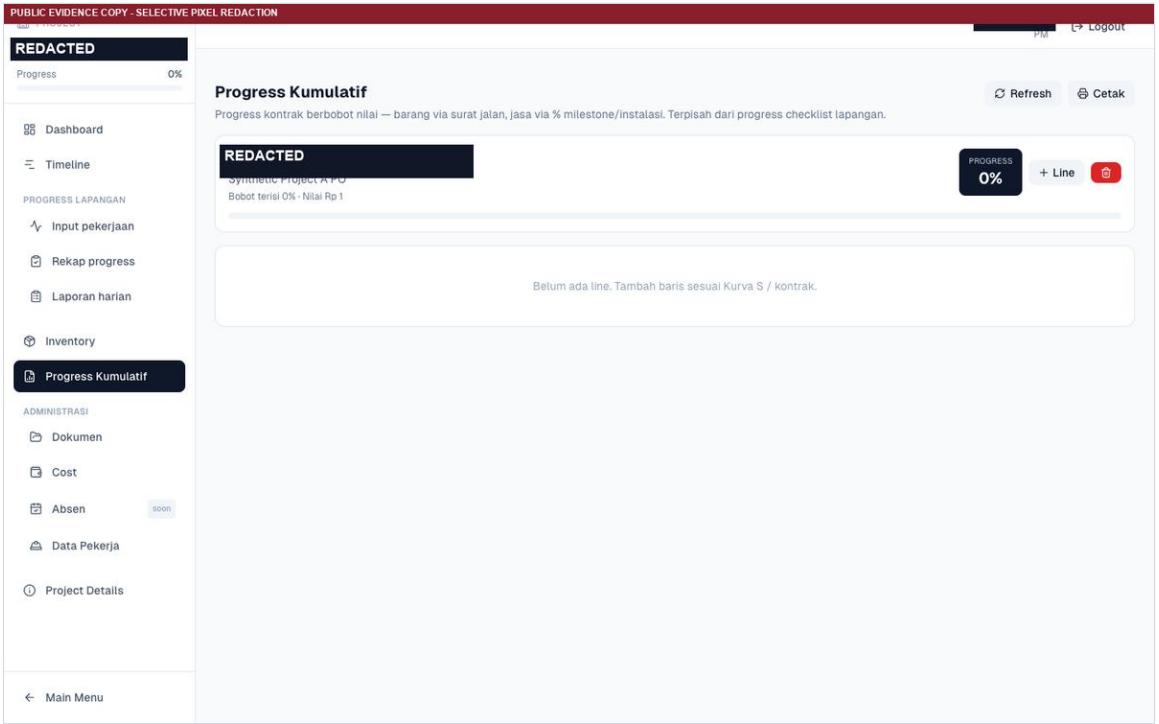


Figure 45 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 10 step 2

3. PM-A requests Project B's purchase orders. The screenshot shows the red-boxed foreign project identifier and the red-boxed totalValue field in the HTTP 200 foreign PO record, proving disclosure.

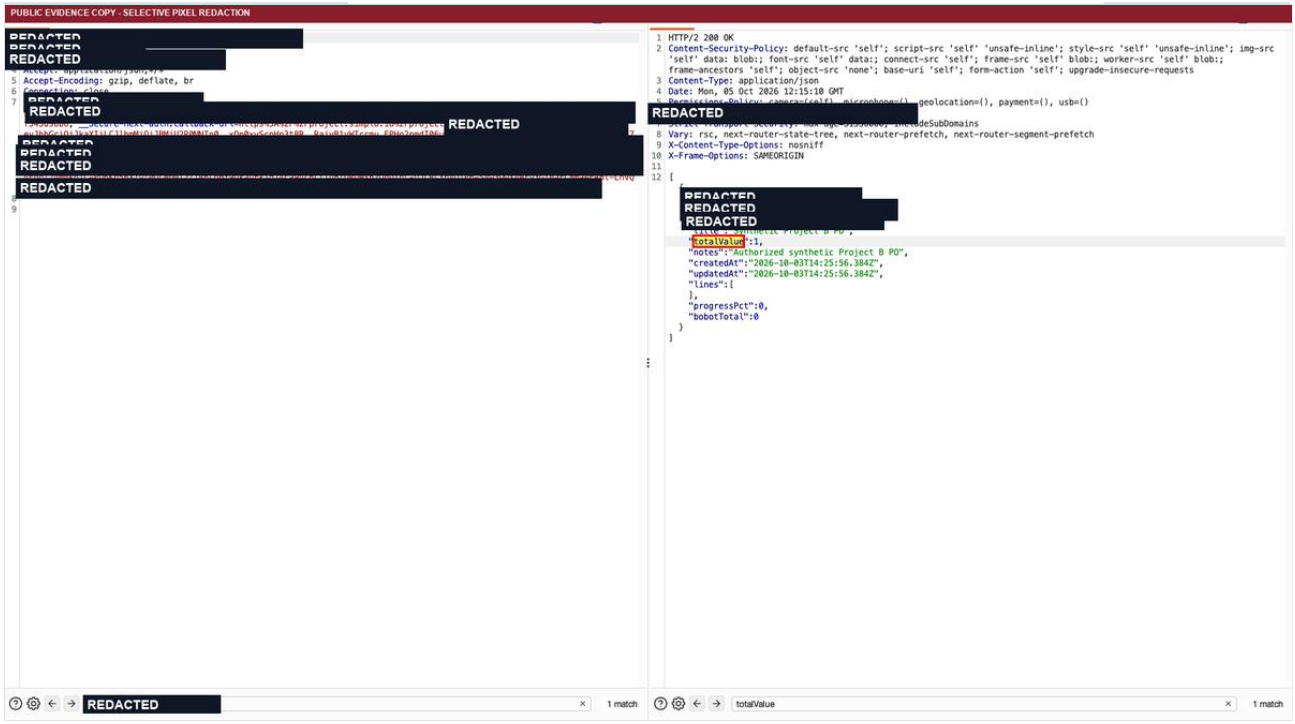


Figure 46 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 10 step 3

4. PM-B requests Project A's purchase orders. The screenshot shows the mirrored red-boxed foreign identifier and totalValue, proving the bidirectional failure.

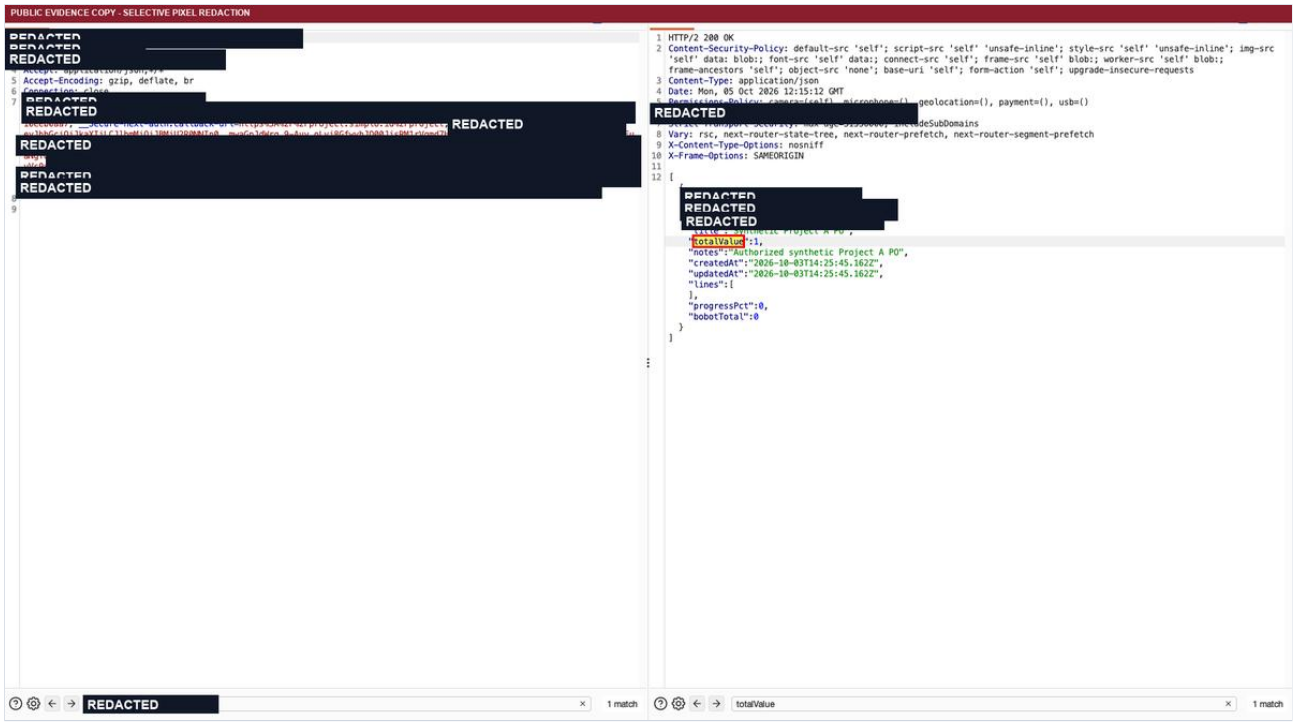


Figure 47 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 10 step 4

Affected scenario 10 — Organization - Organization Entries

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

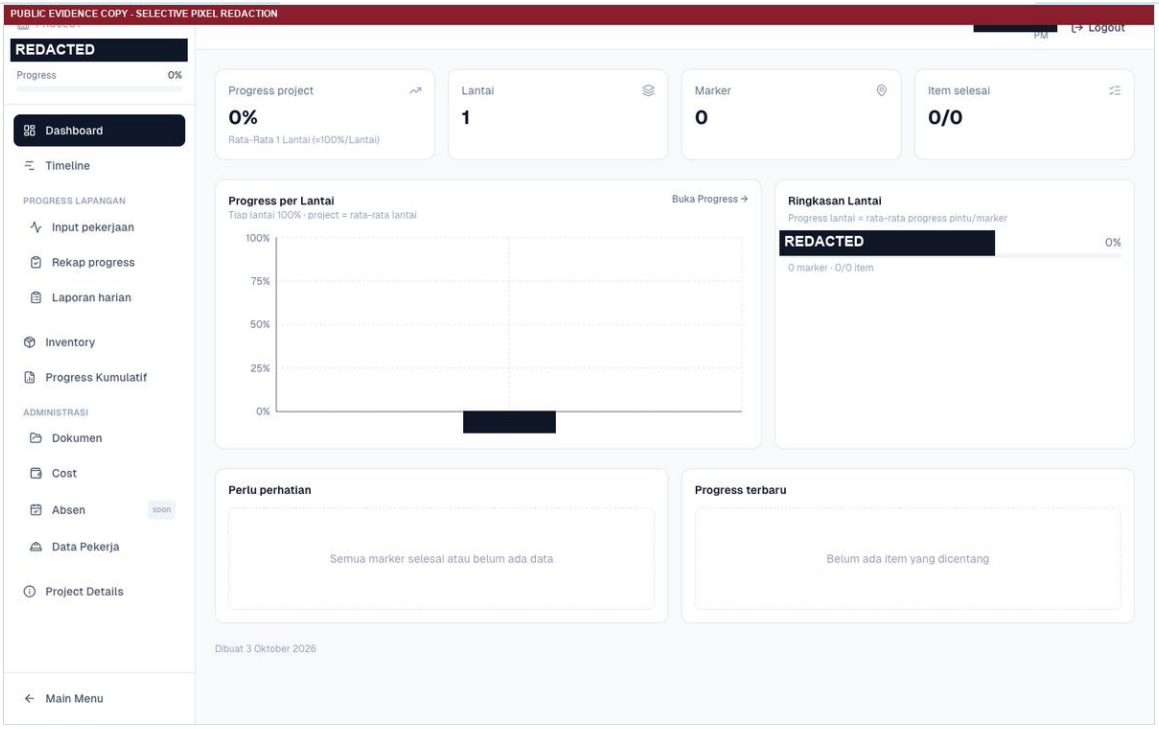


Figure 48 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 11 step 1

2. Click Project Details in the sidebar. The screenshot shows the details page with the Struktur Organisasi card carrying the [TEST FIXTURE] entry, establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

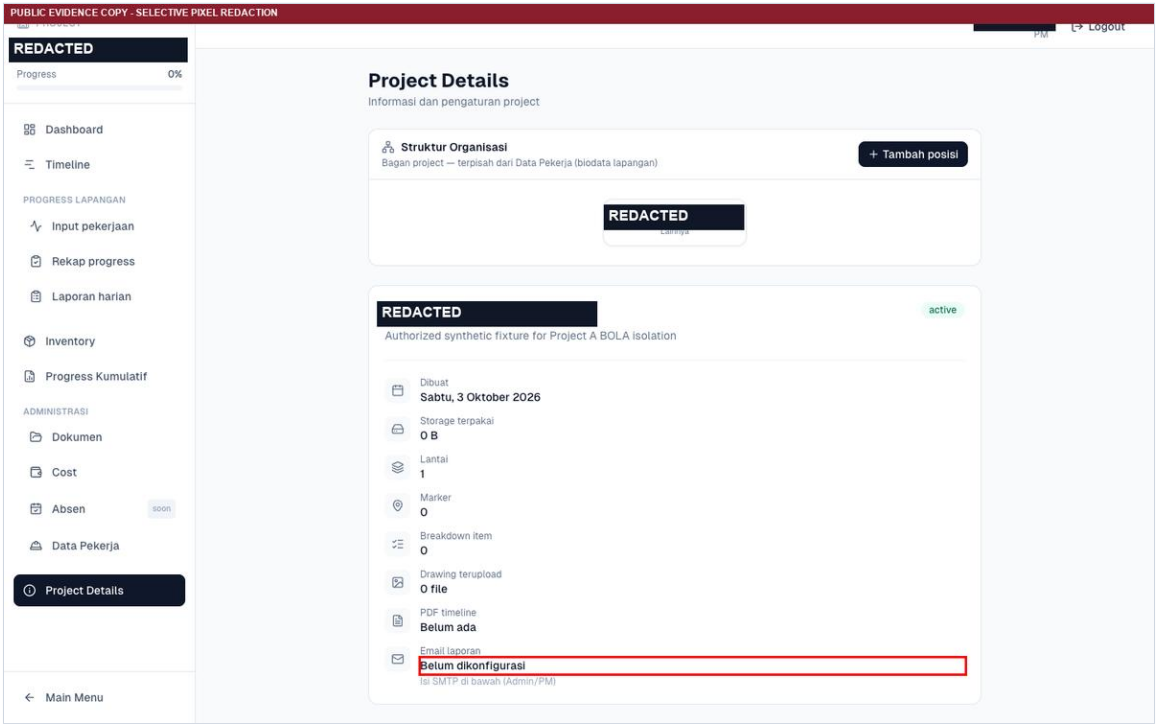


Figure 49 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 11 step 2

3. PM-A requests Project B's organization entries. The screenshot shows the red-boxed foreign project identifier and red-boxed foreign projectId bindings on the returned entries in the HTTP 200 response, proving disclosure including contact PII.

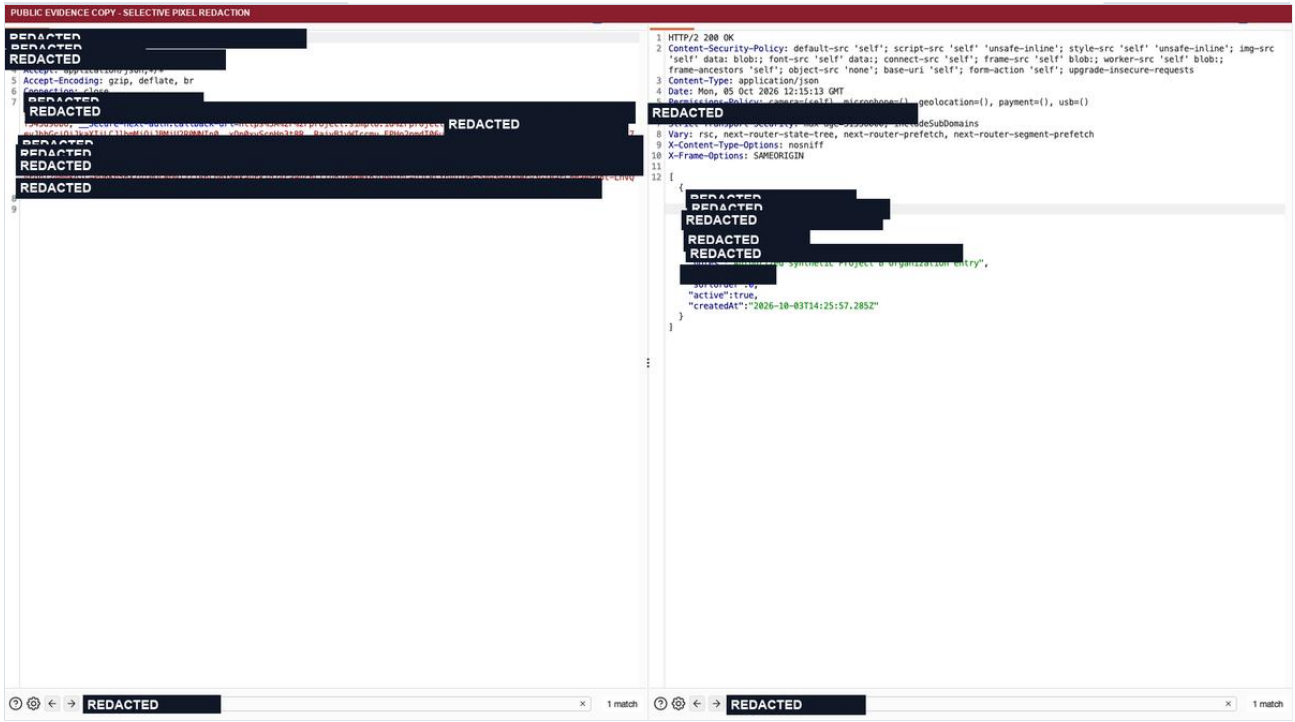


Figure 50 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 11 step 3

4. PM-B requests Project A's organization entries. The screenshot shows the mirrored red-boxed foreign identifier and bindings, proving the bidirectional failure.

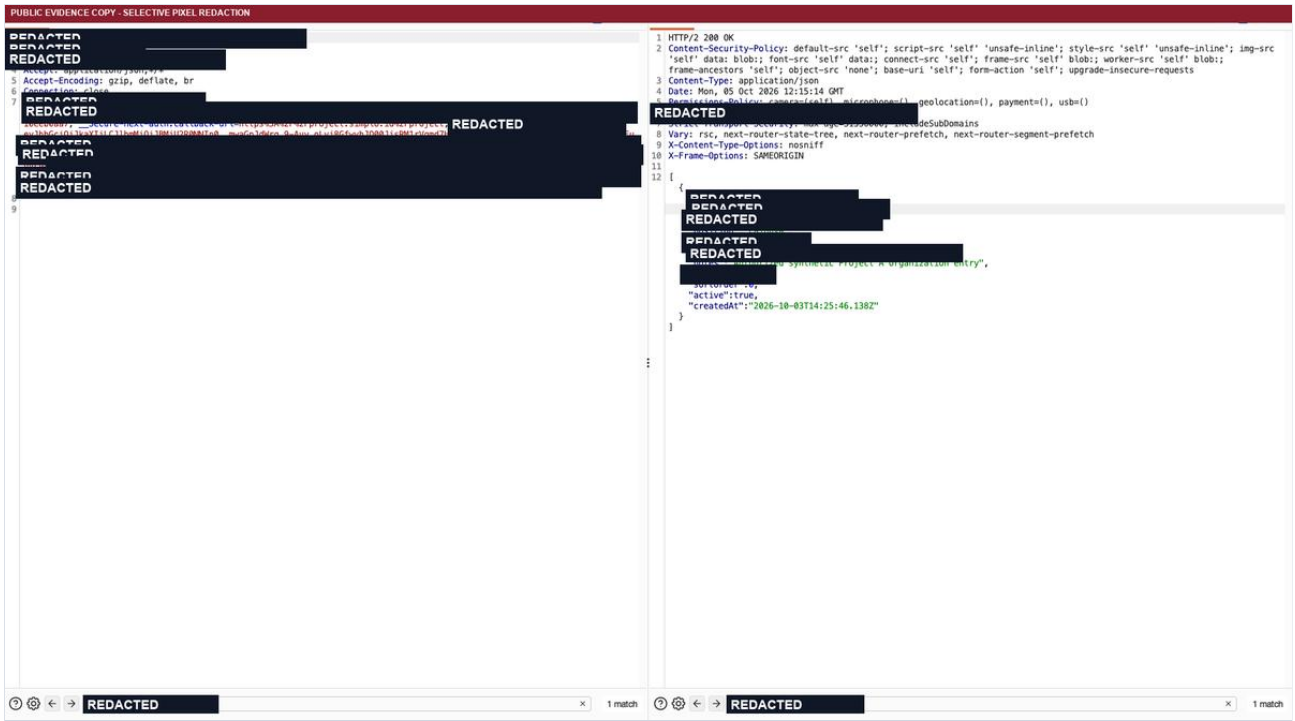


Figure 51 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 11 step 4

Affected scenario 11 — Project - Floors

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

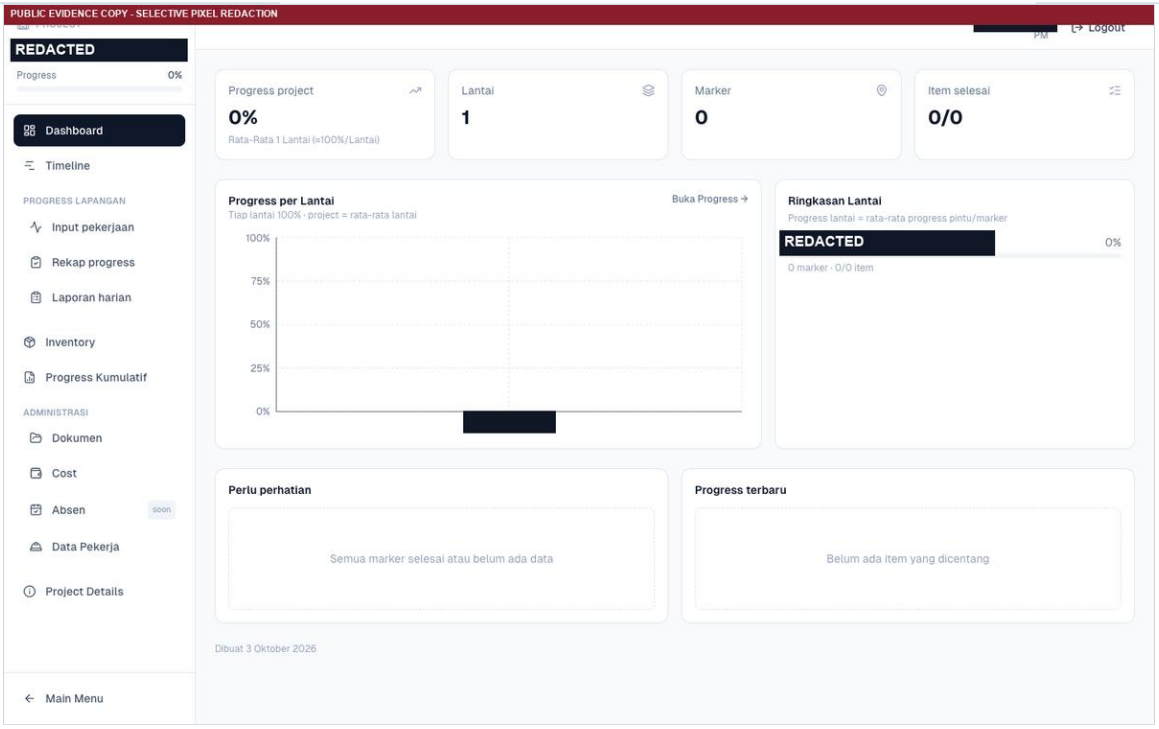


Figure 52 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 12 step 1

2. The screenshot additionally shows the Ringkasan Lantai card with the [TEST FIXTURE] floor. Click Project Details in the sidebar; The screenshot shows the details page, establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

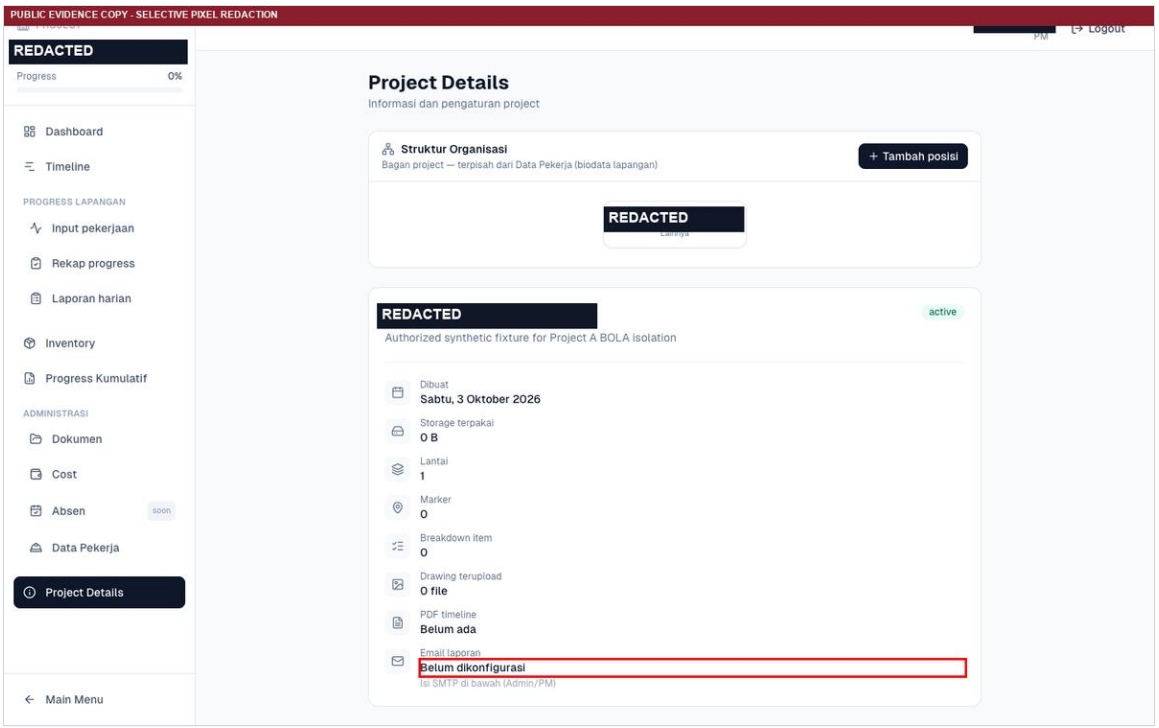


Figure 53 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 12 step 2

3. PM-A requests Project B's floors. The screenshot shows the red-boxed foreign project identifier and red-boxed foreign projectId bindings on the returned floor rows in the HTTP 200 response, proving disclosure.

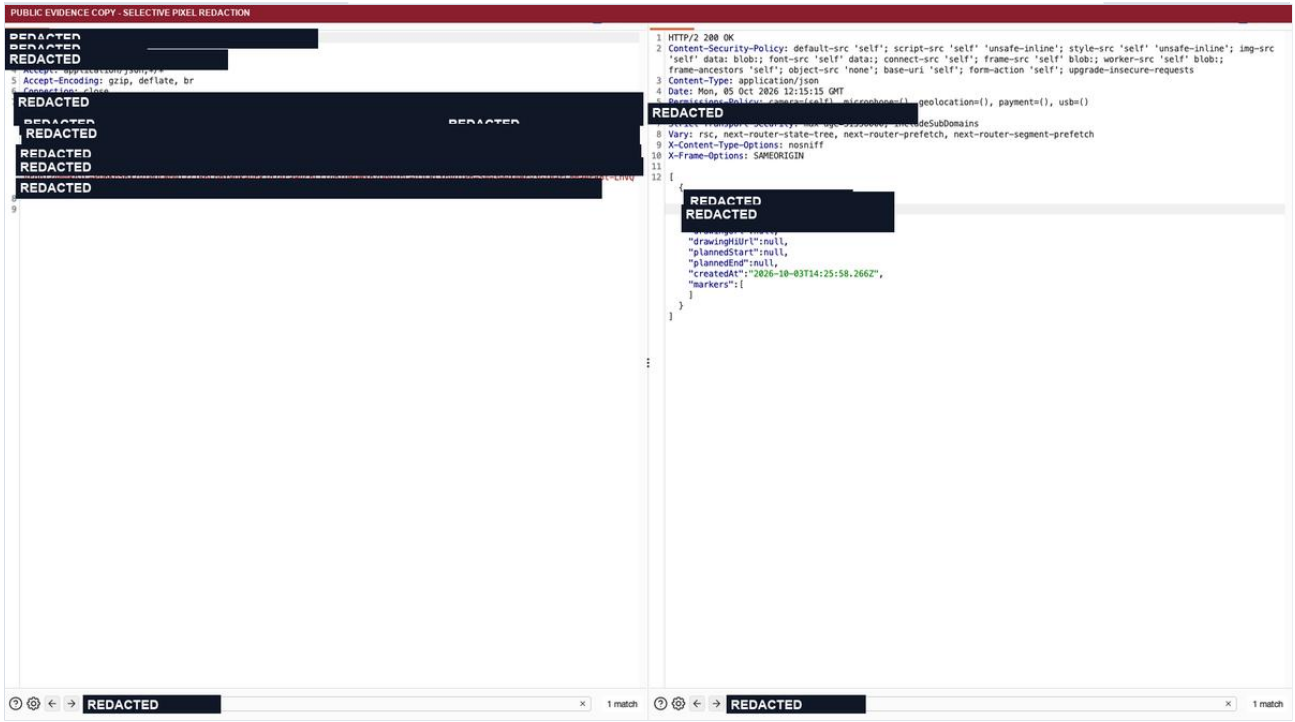


Figure 54 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 12 step 3

4. PM-B requests Project A's floors. The screenshot shows the mirrored red-boxed foreign identifier and bindings, proving the bidirectional failure.

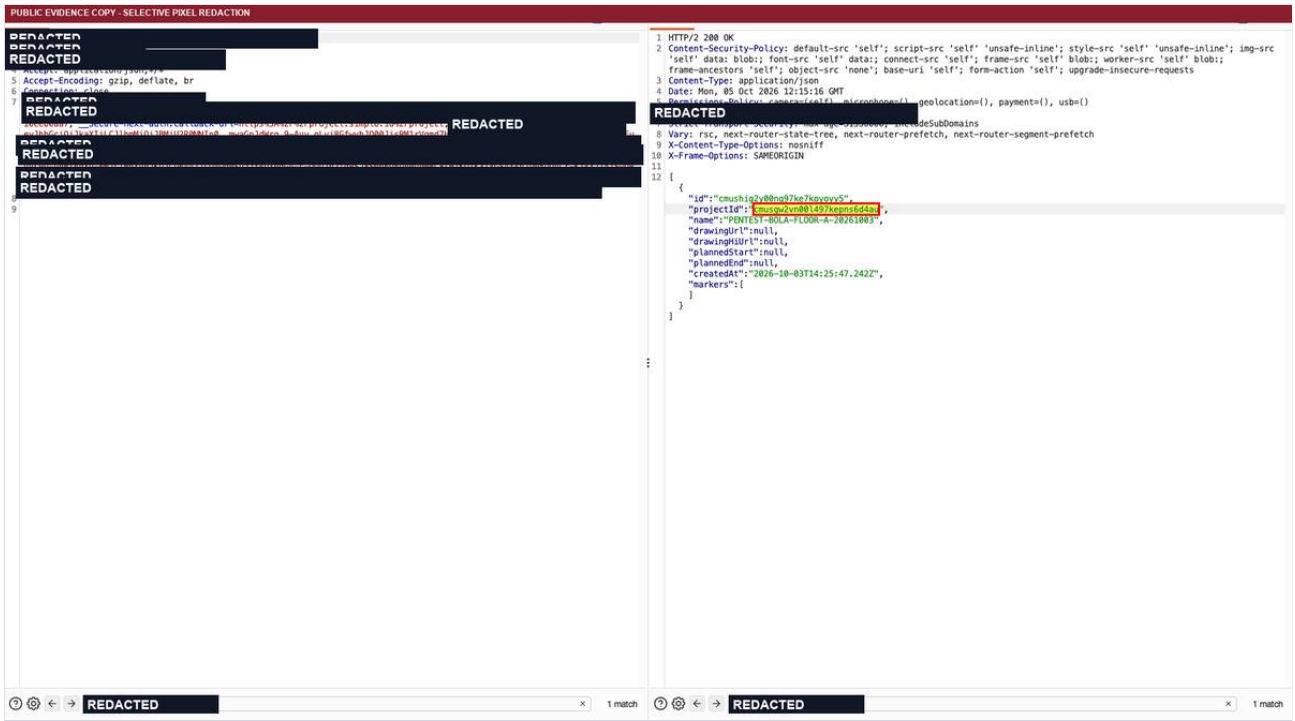


Figure 55 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 12 step 4

Affected scenario 12 — Inventory - Deliveries

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

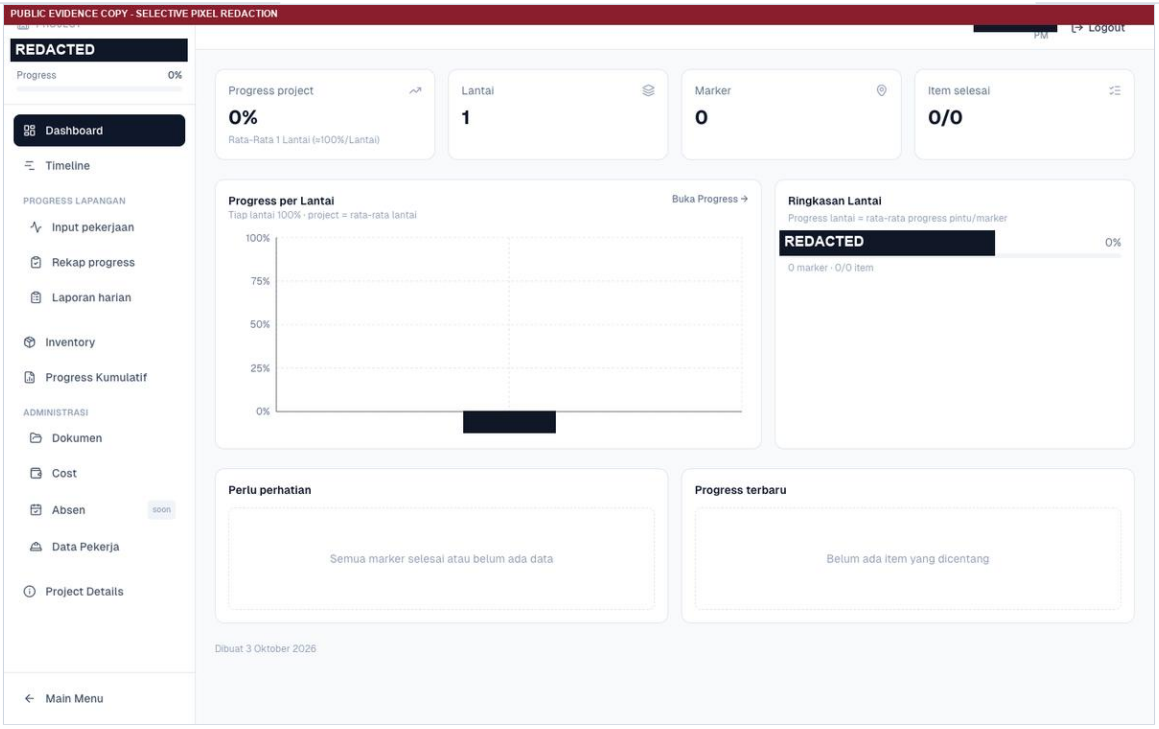


Figure 56 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 13 step 1

2. Click Inventory in the sidebar navigation. The screenshot shows the Inventory page with the project's own data visible (Stok gudang rows including the BOLA fixture item; Barang masuk covers deliveries), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

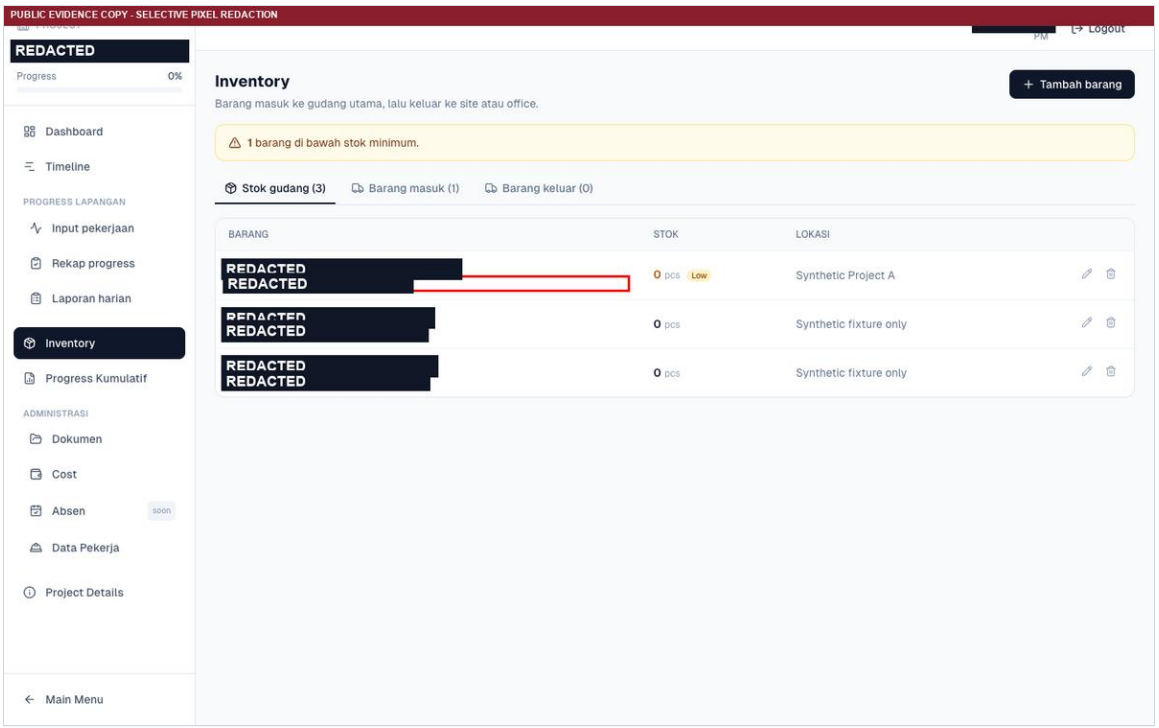


Figure 57 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 13 step 2

3. PM-A requests Project B's deliveries. The screenshot shows the red-boxed foreign project identifier and red-boxed foreign projectId bindings on the returned delivery rows in the HTTP 200 response, proving disclosure including party and line detail.

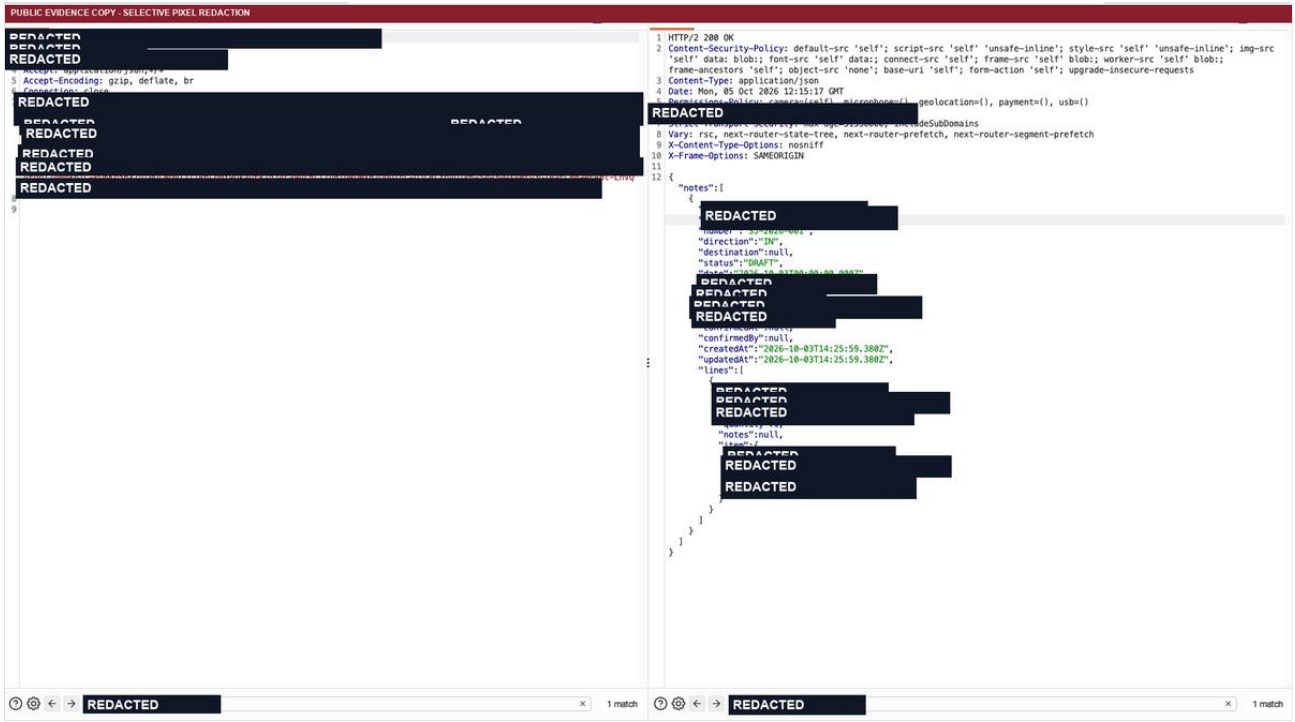


Figure 58 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 13 step 3

4. PM-B requests Project A's deliveries. The screenshot shows the mirrored red-boxed foreign identifier and bindings, proving the bidirectional failure.

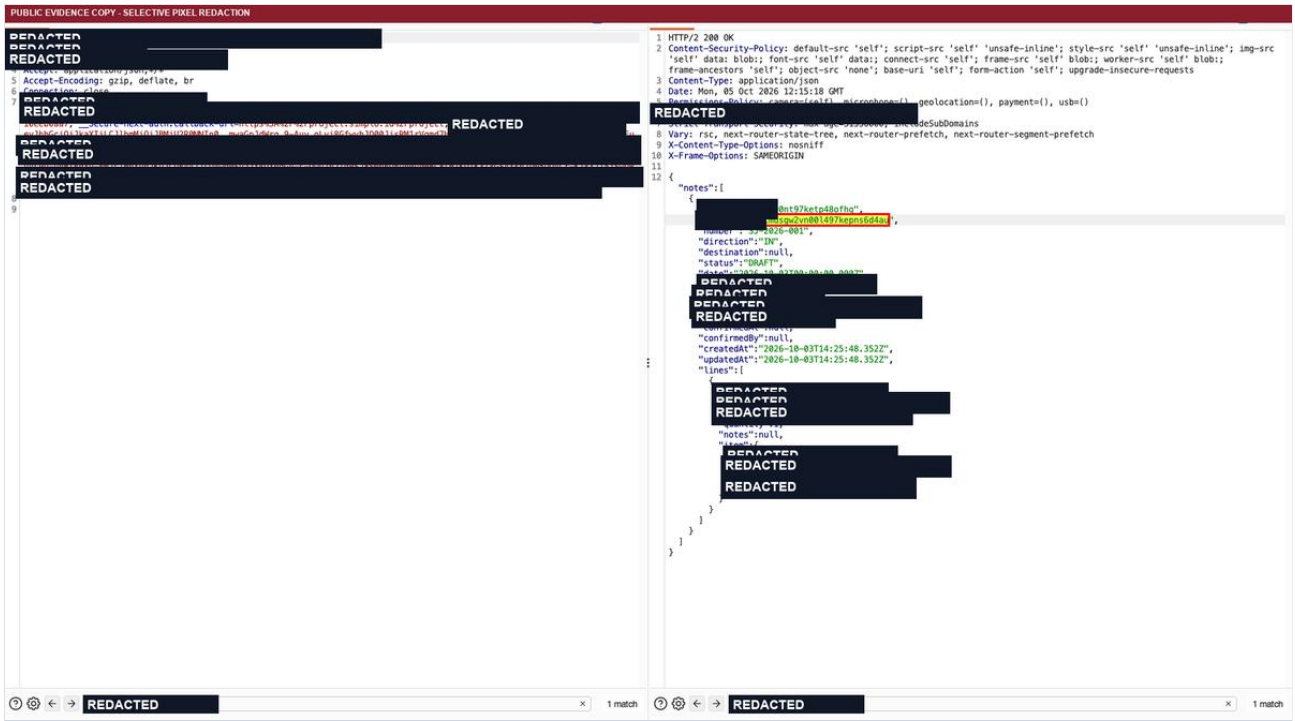


Figure 59 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 13 step 4

Affected scenario 13 — Project Settings - Mail Settings

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

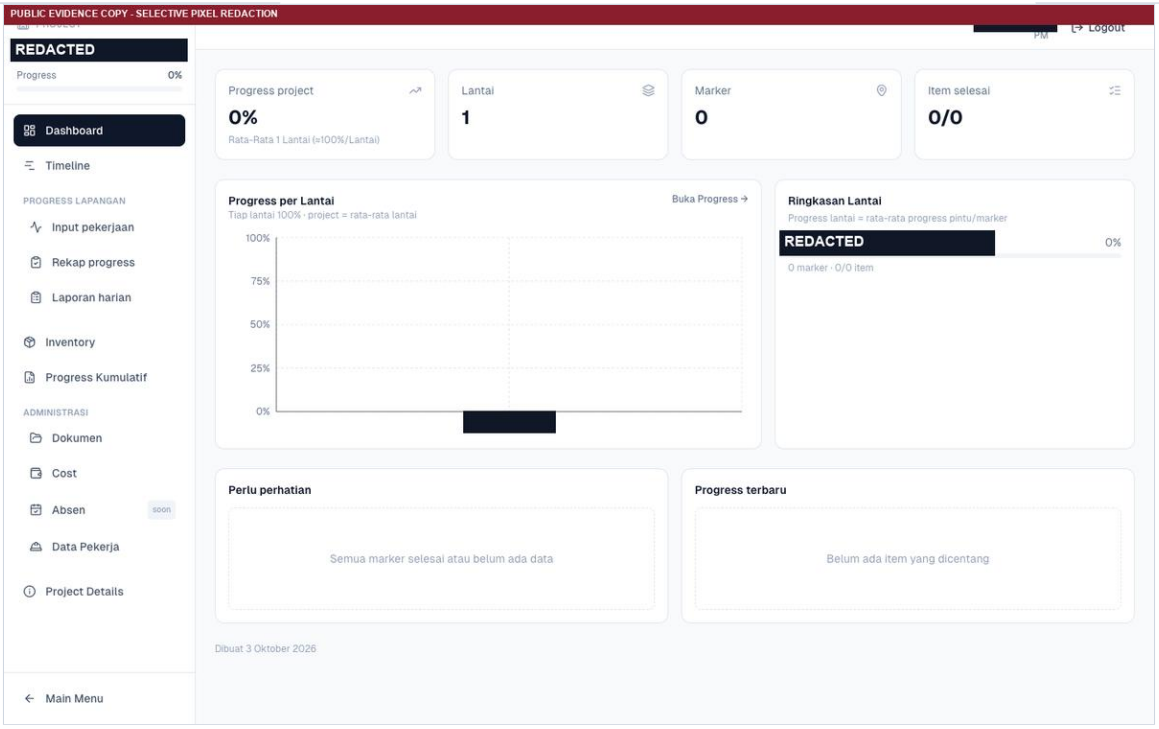


Figure 60 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 14 step 1

2. Click Project Details in the sidebar. The screenshot shows the details page with the mail state Belum dikonfigurasi (Isi SMTP di bawah (Admin/PM)); there is no dedicated mail-settings page, so the authorization check is API-level in the steps below.

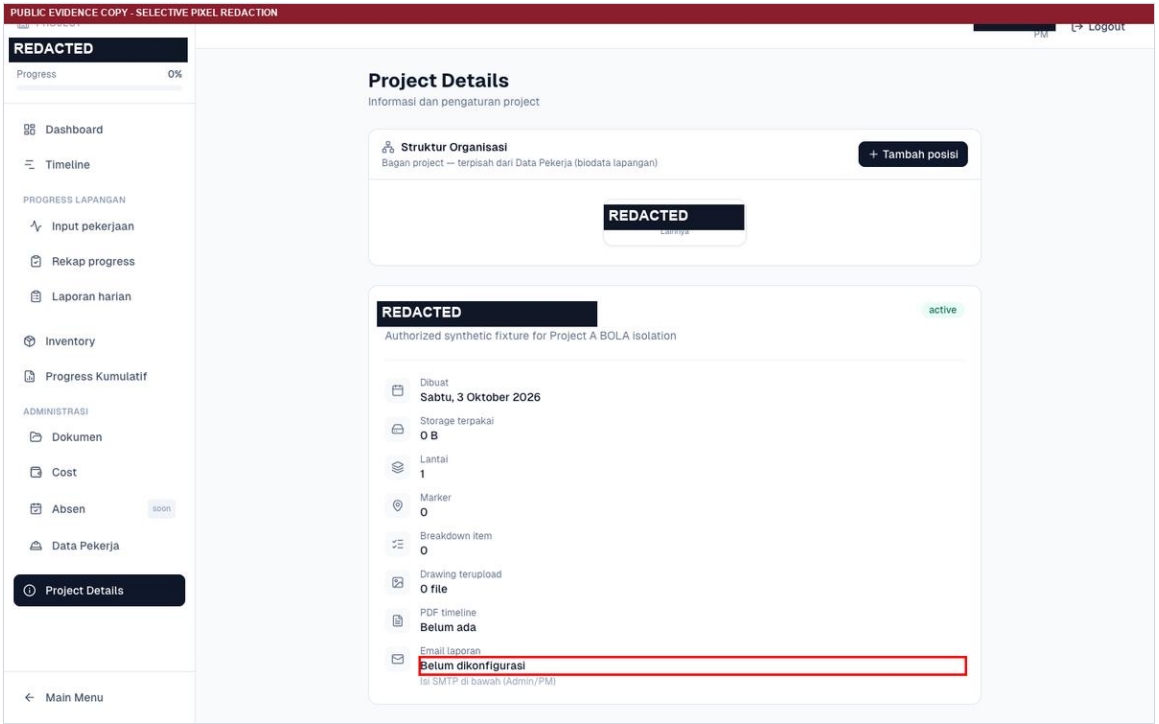


Figure 61 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 14 step 2

3. PM-A requests Project B's mail settings. The screenshot shows the red-boxed foreign route and the red-boxed smtpPassSet field in the HTTP 200 foreign configuration, proving the route is reachable cross-project.

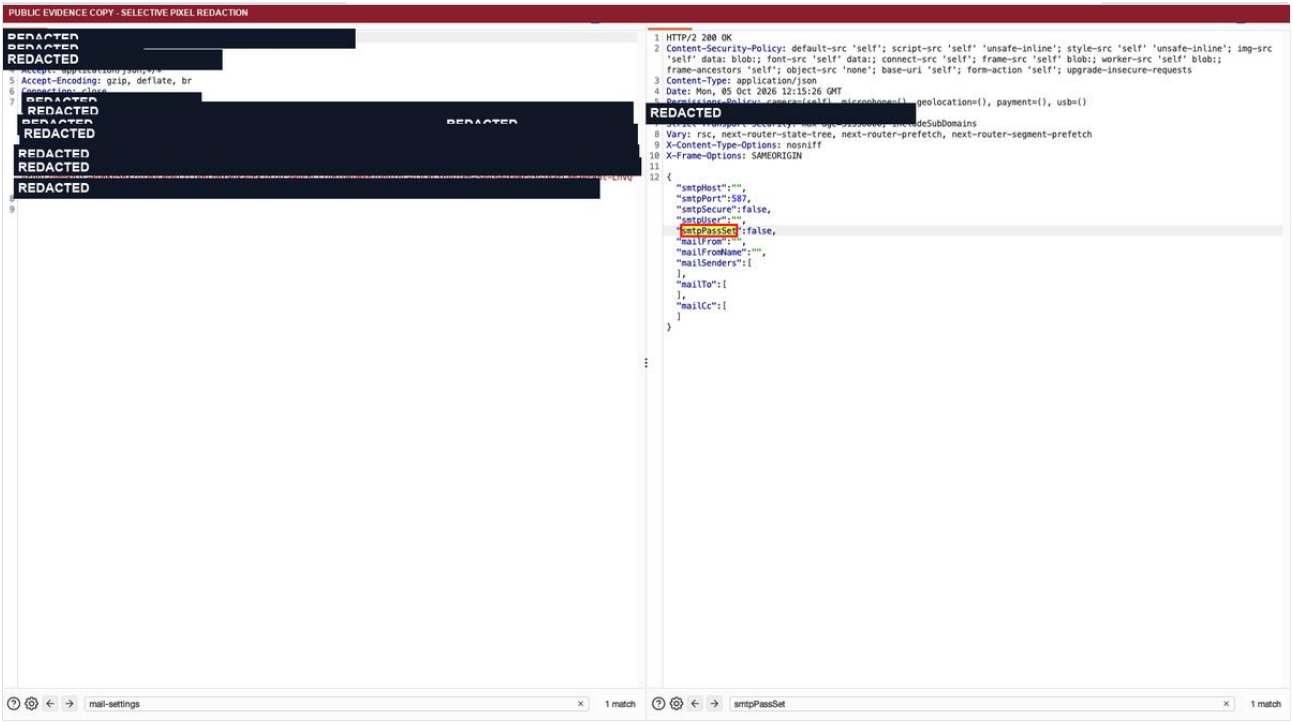


Figure 62 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 14 step 3

4. PM-B requests Project A's mail settings. The screenshot shows the mirrored red-boxed route and configuration, proving the bidirectional failure.

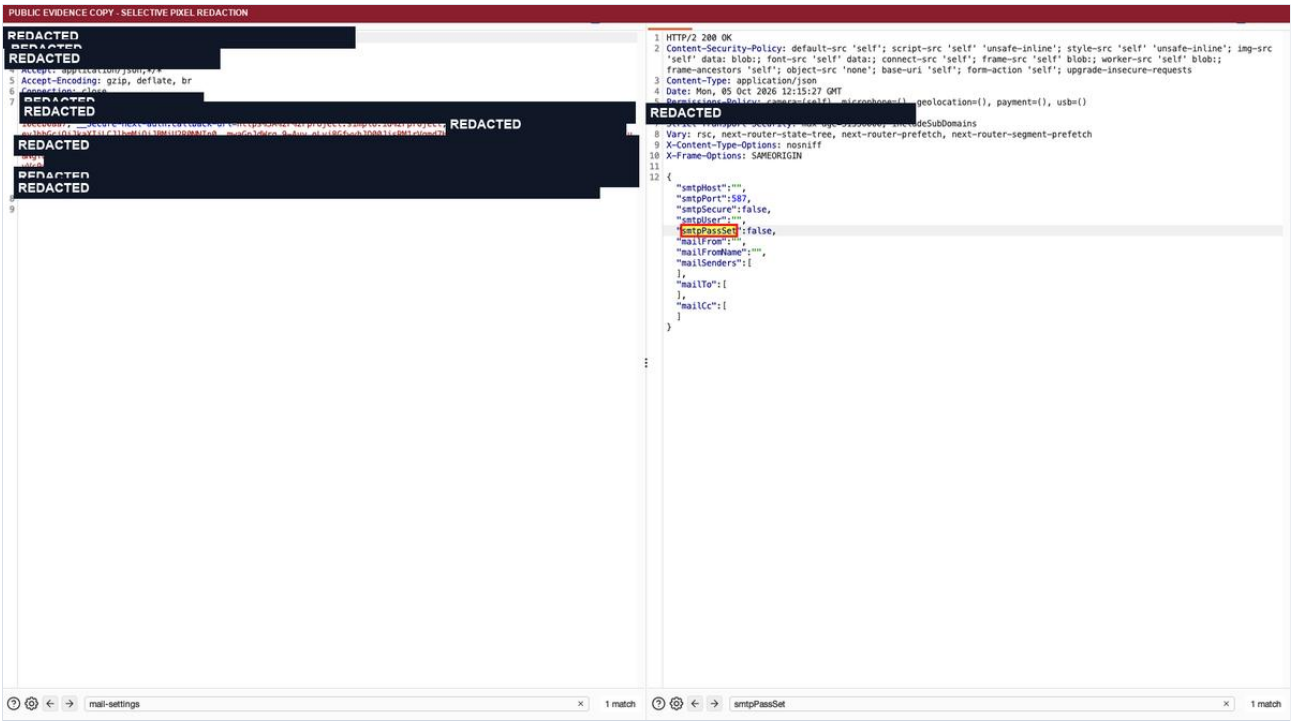


Figure 63 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 14 step 4

Affected scenario 14 — Daily Report - Daily Reports

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

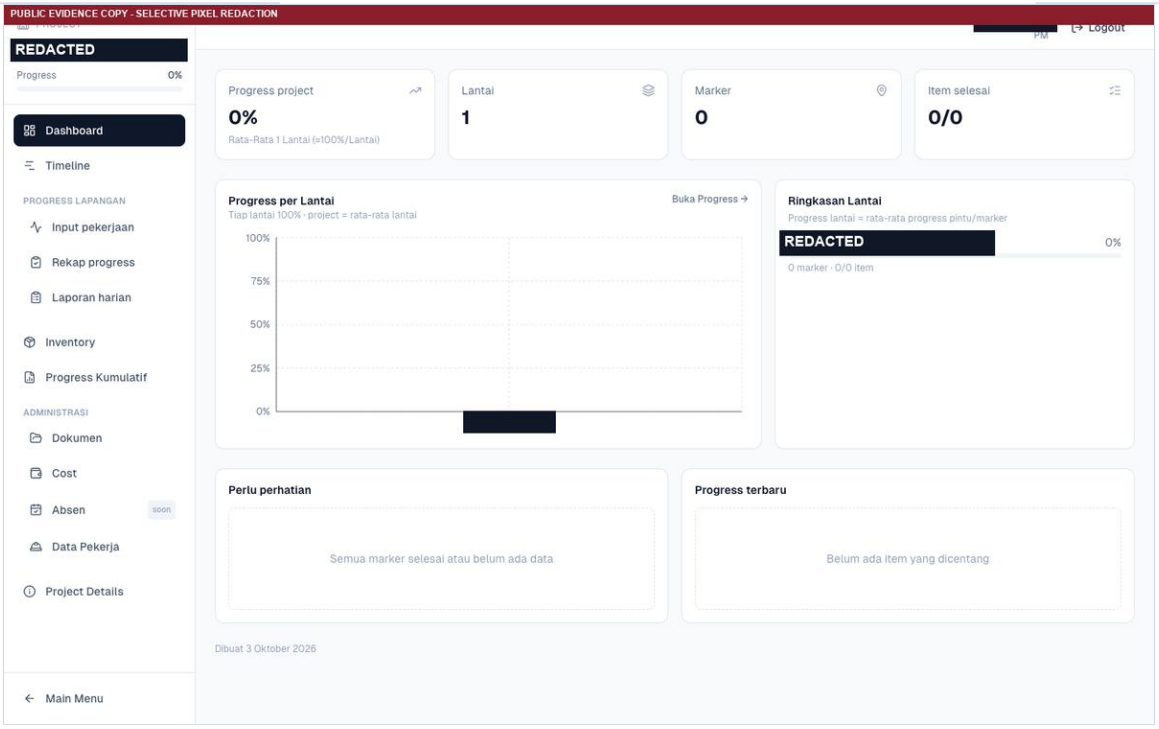


Figure 64 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 15 step 1

2. Click Laporan harian in the sidebar navigation. The screenshot shows the Laporan Harian daily-report form page with the project's own data visible (report history and entry form), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

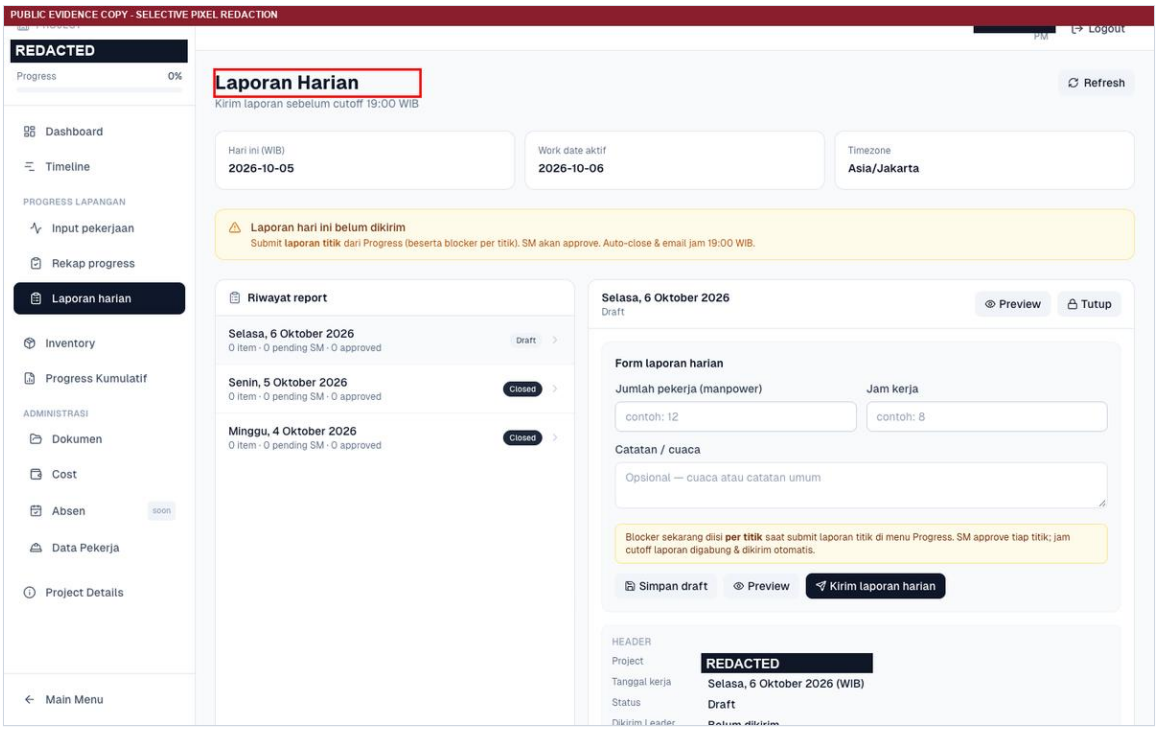


Figure 65 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 15 step 2

3. PM-A requests Project B's daily reports. The screenshot shows the red-boxed foreign project identifier and the red-boxed reports array populated with Project B report entries in the HTTP 200 response, proving disclosure.

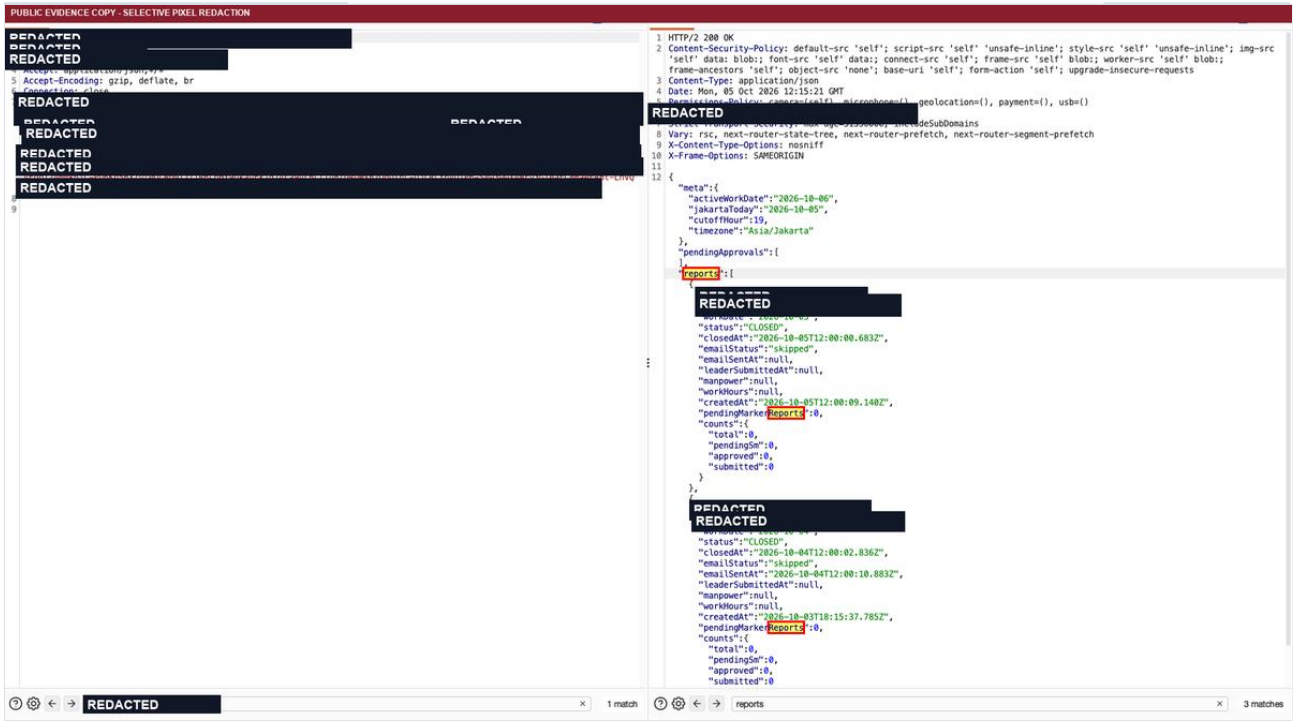


Figure 66 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 15 step 3

4. PM-B requests Project A's daily reports. The screenshot shows the mirrored red-boxed foreign identifier and populated reports array, proving the bidirectional failure.

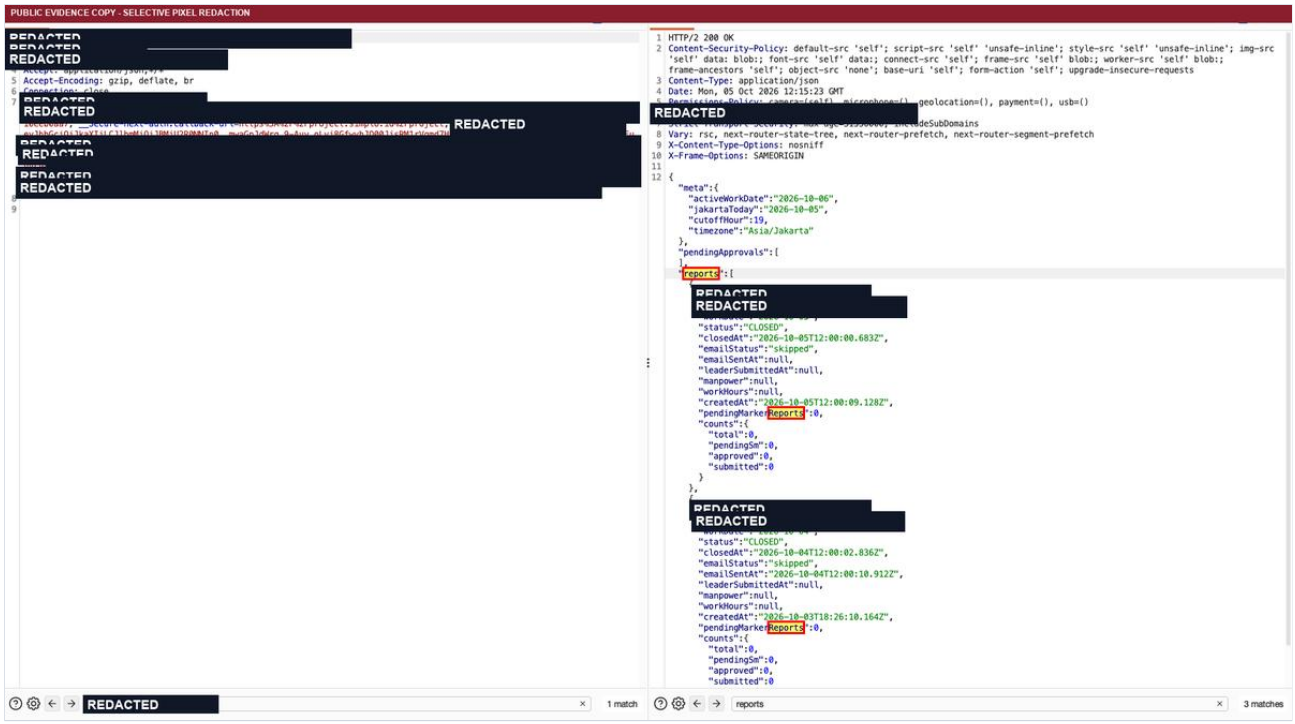


Figure 67 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 15 step 4

Affected scenario 15 — Data Pekerja - Direct Worker Record

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

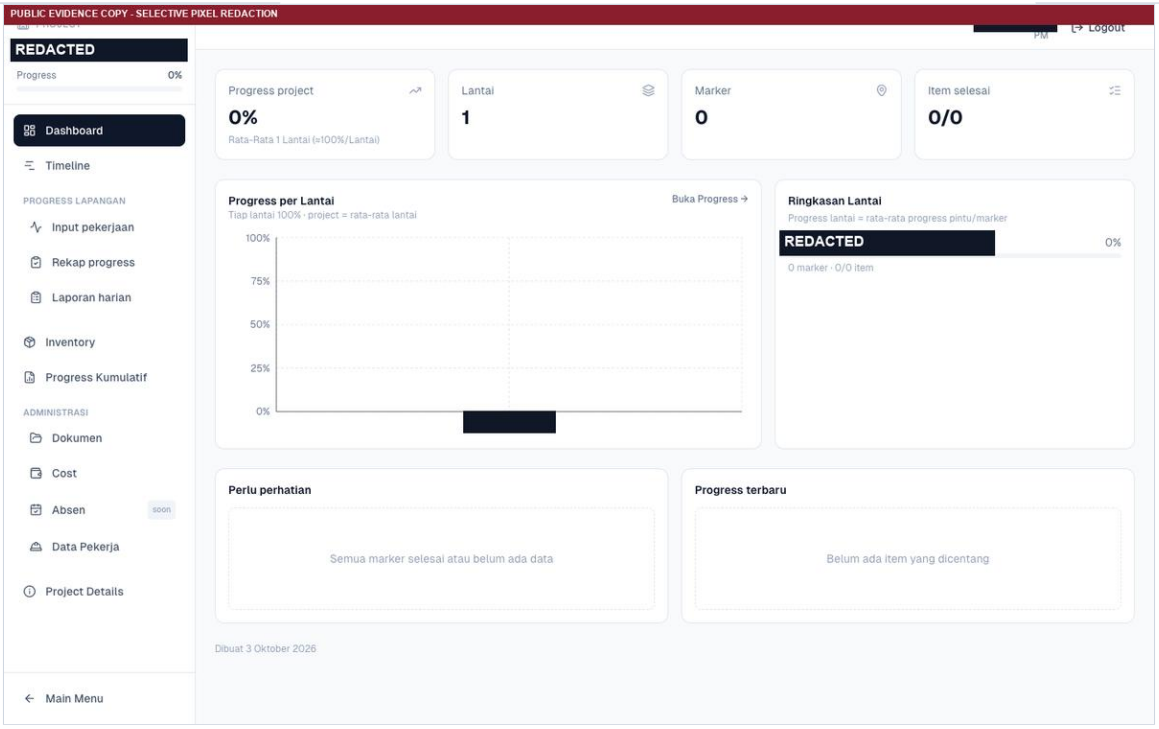


Figure 68 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 16 step 1

2. Click Data Pekerja in the sidebar, then open the Semua tab. The screenshot shows the worker roster with the [TEST FIXTURE] row, establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

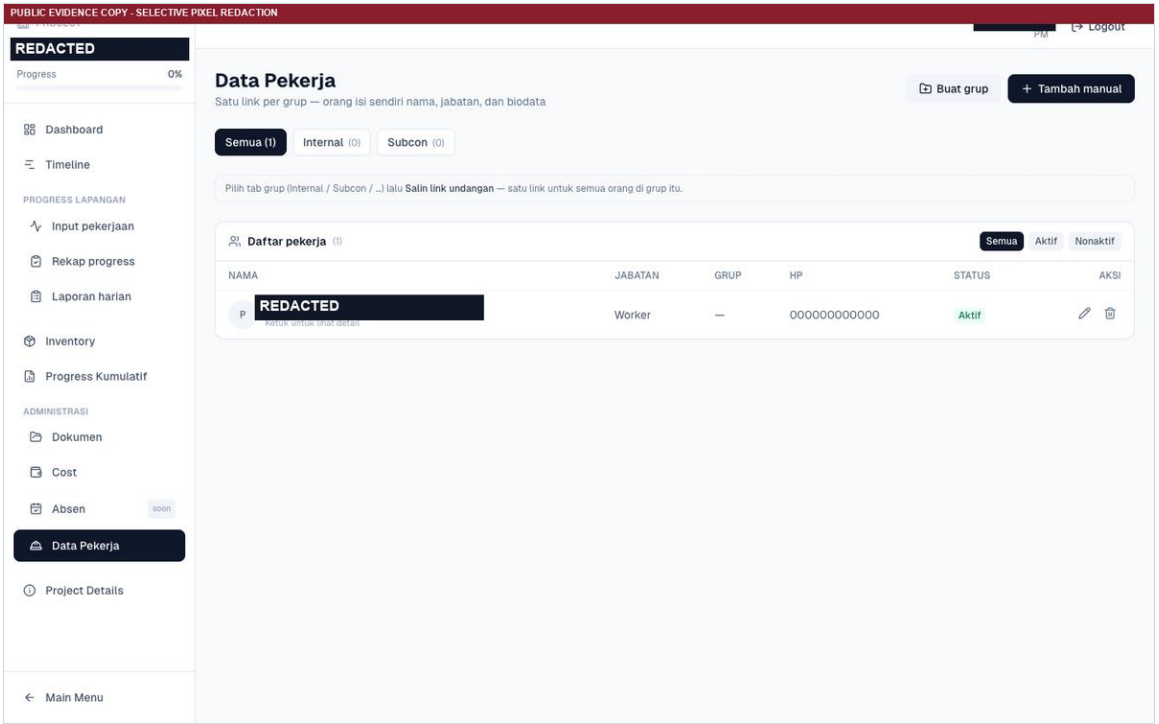


Figure 69 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 16 step 2

3. PM-A requests Project B's worker record. The screenshot shows the red-boxed foreign worker identifier in the request and the red-boxed nik field in the HTTP 200 full foreign worker record, proving PII disclosure.

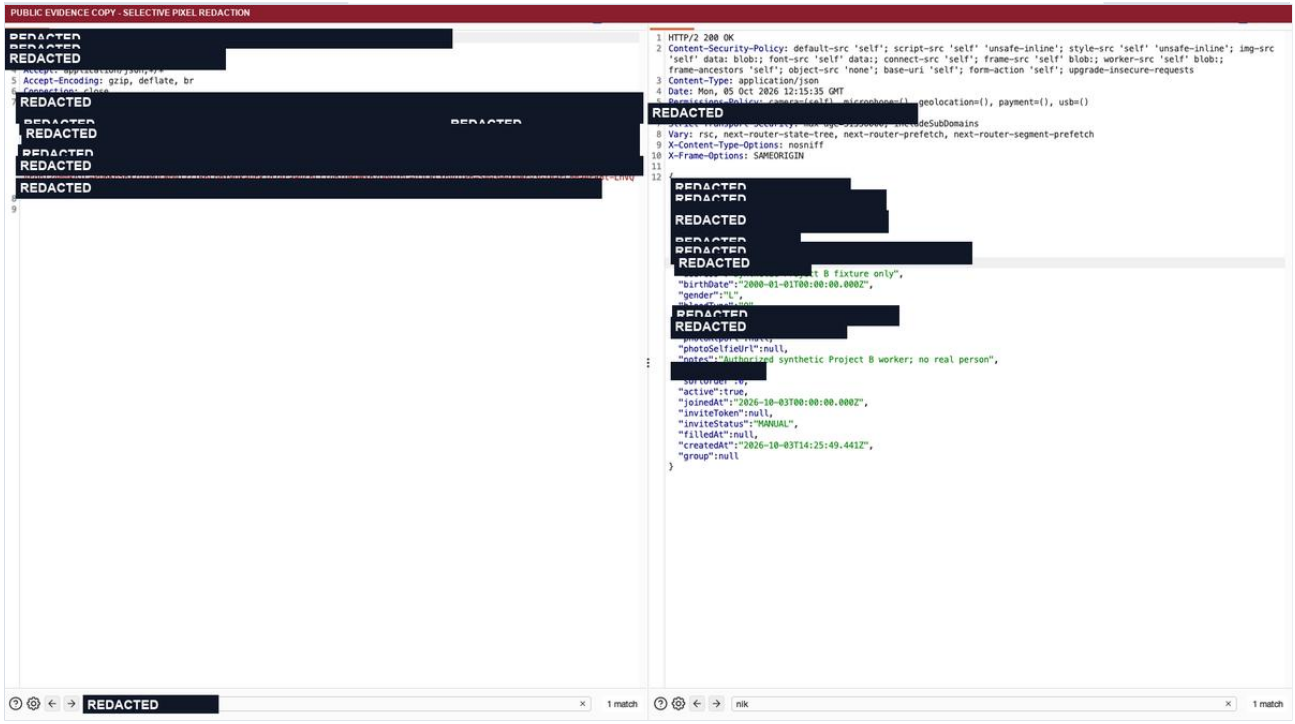


Figure 70 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 16 step 3

4. PM-B requests Project A's worker record. The screenshot shows the mirrored red-boxed foreign worker identifier and nik field, proving the bidirectional failure.

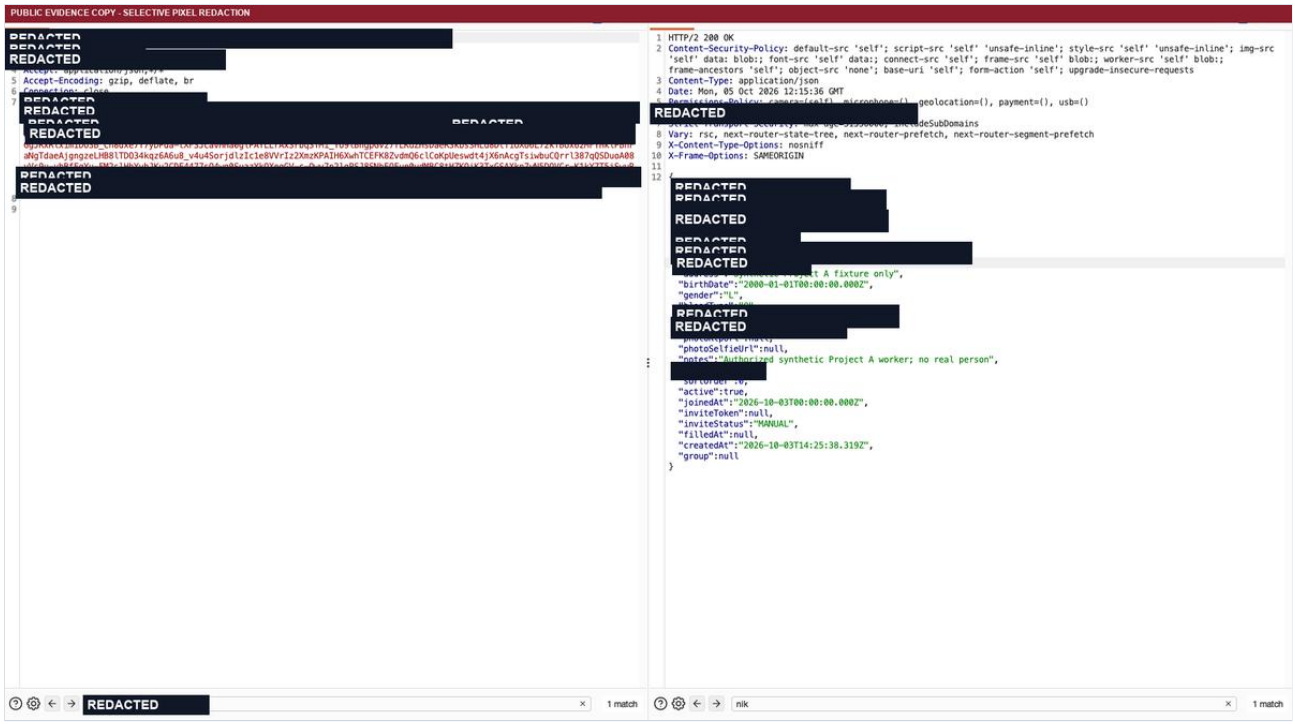


Figure 71 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 16 step 4

Affected scenario 16 — Dokumen - Direct File Download

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

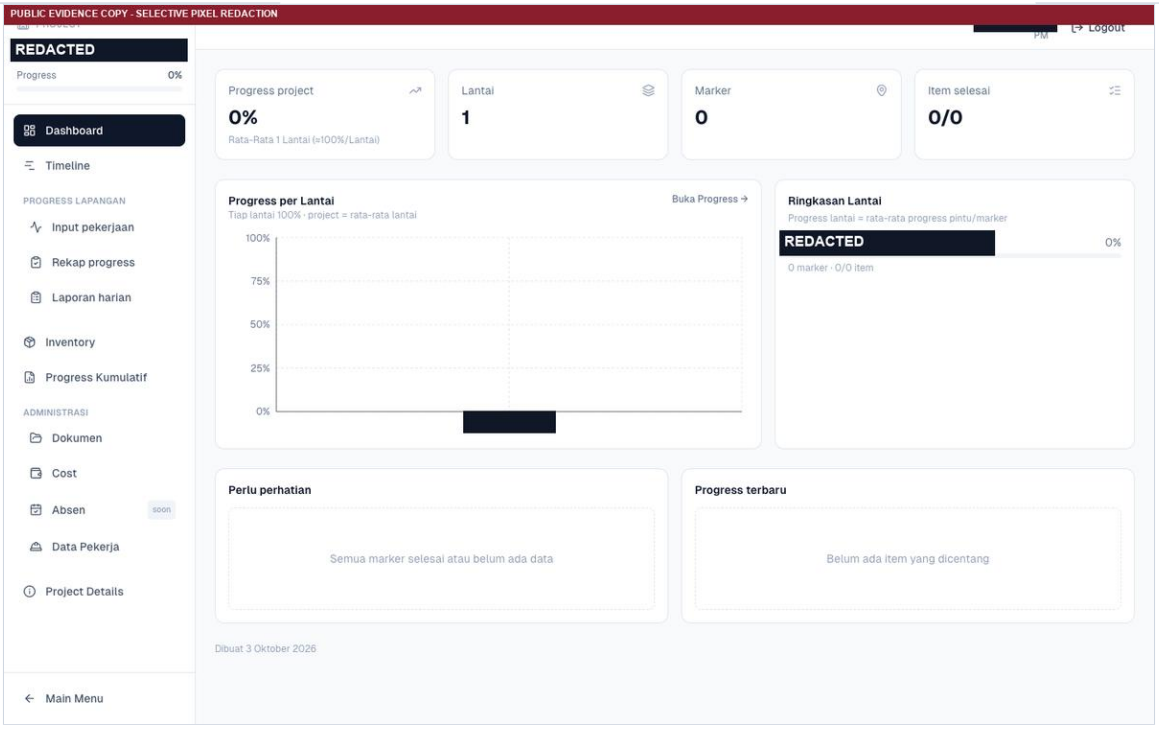


Figure 72 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 17 step 1

2. Click Dokumen in the sidebar navigation. The screenshot shows the Dokumen page with the project's own data visible ([TEST FIXTURE] file row), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

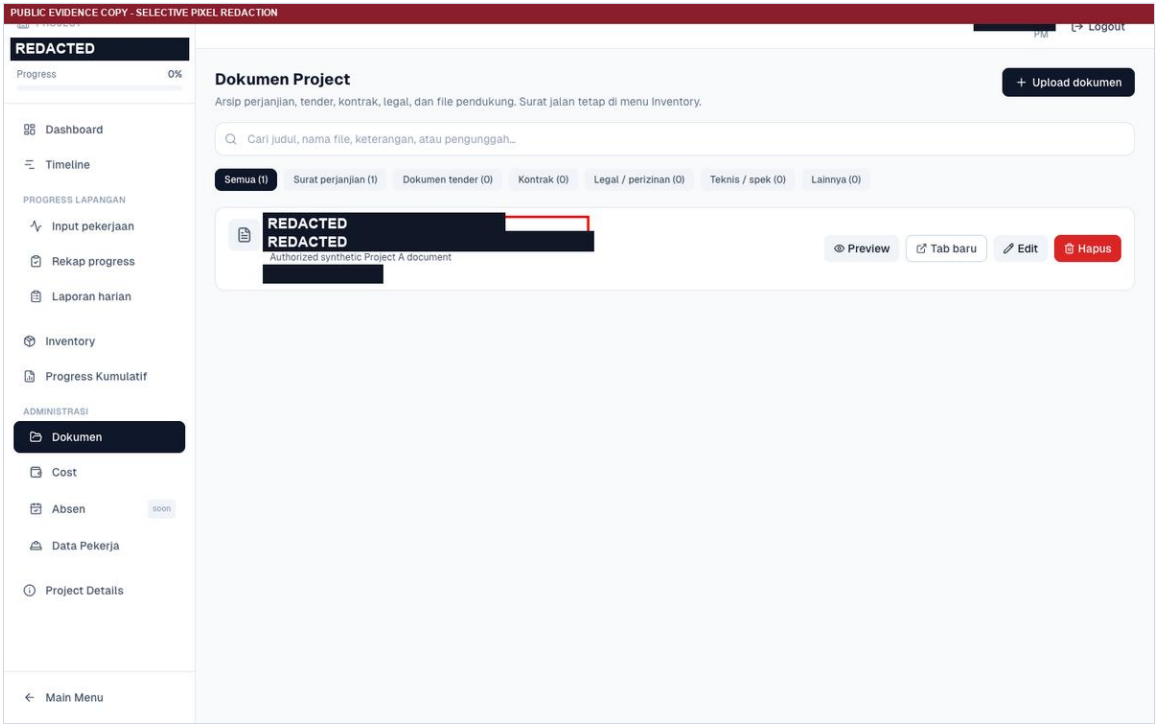


Figure 73 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 17 step 2

3. PM-A requests Project B's document file. The screenshot shows the red-boxed foreign file path and the red-boxed HTTP 200 OK response with Content-Type image/jpeg, proving foreign file bytes are served.

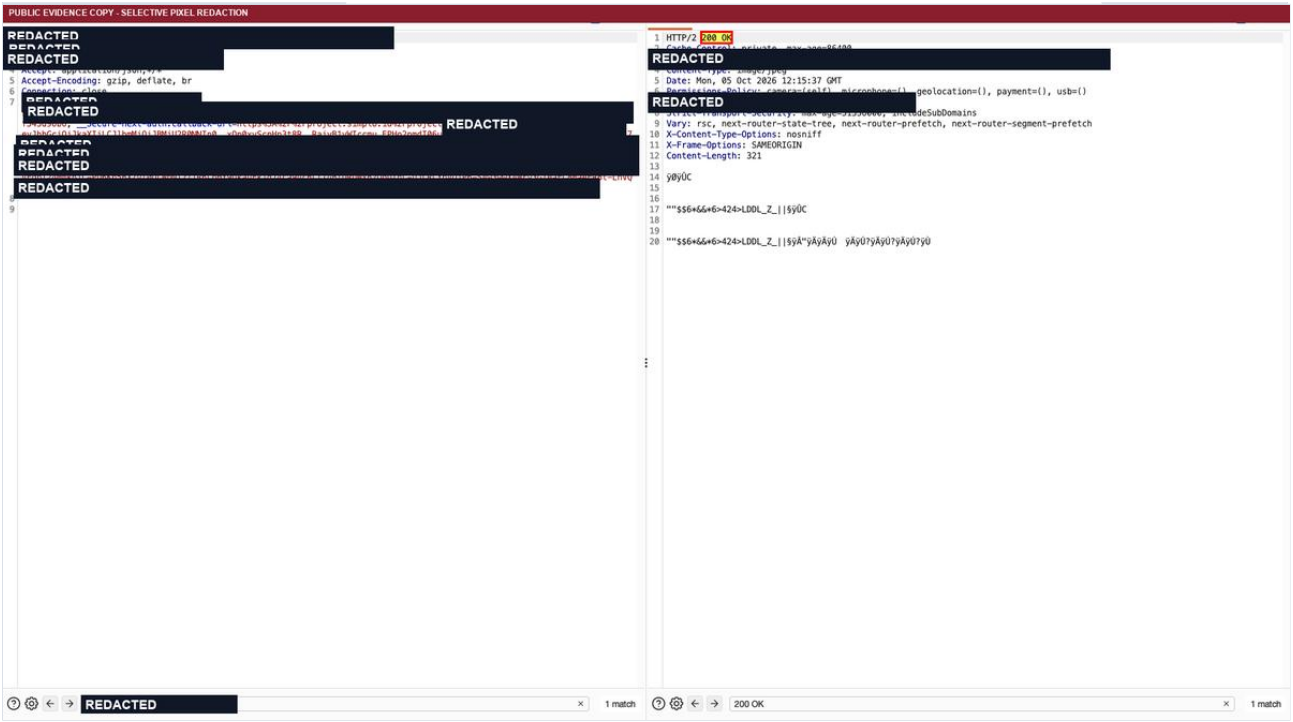


Figure 74 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 17 step 3

4. PM-B requests Project A's document file. The screenshot shows the mirrored red-boxed foreign file path and HTTP 200 OK image response, proving the bidirectional failure.

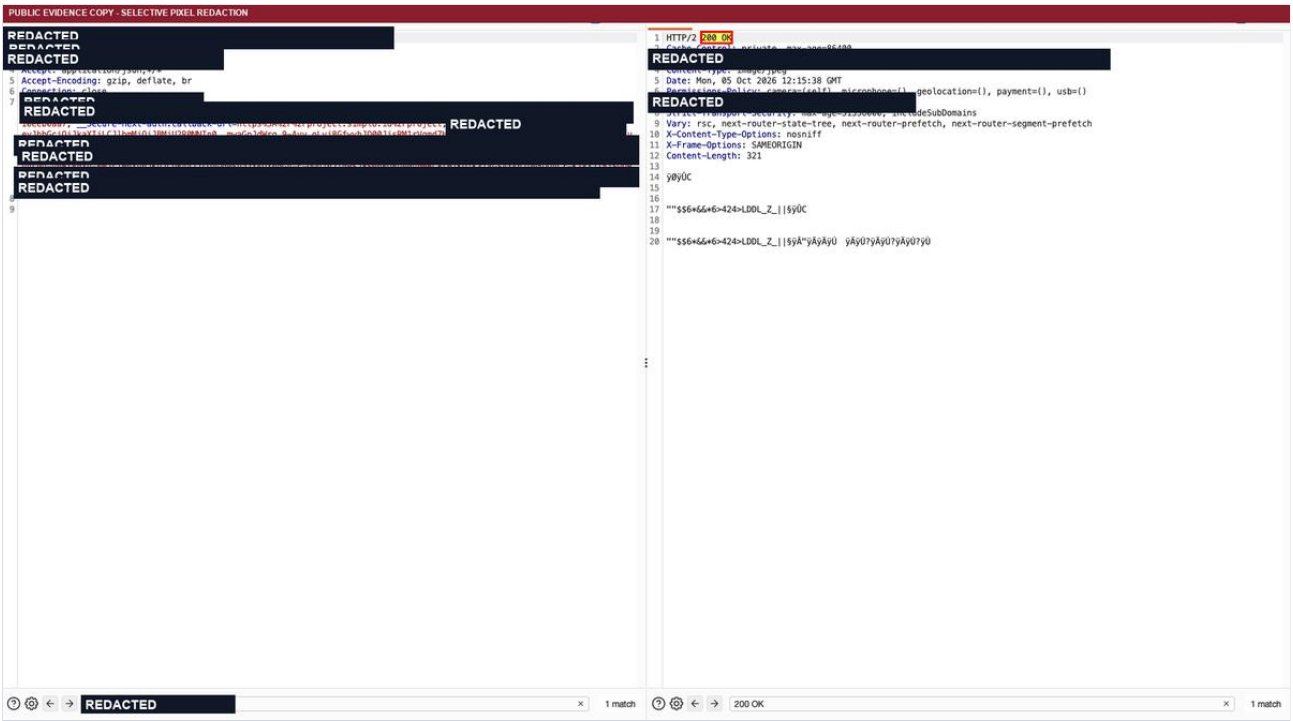


Figure 75 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 17 step 4

Affected scenario 17 — Cost - Receipt Download

1. Log in with the authorized test account (PM) test account and open the Project A dashboard at / project/[OBJECT ID]/overview. The screenshot shows the [TEST FIXTURE] project context, the PM role badge, and the sidebar navigation used in the next step.

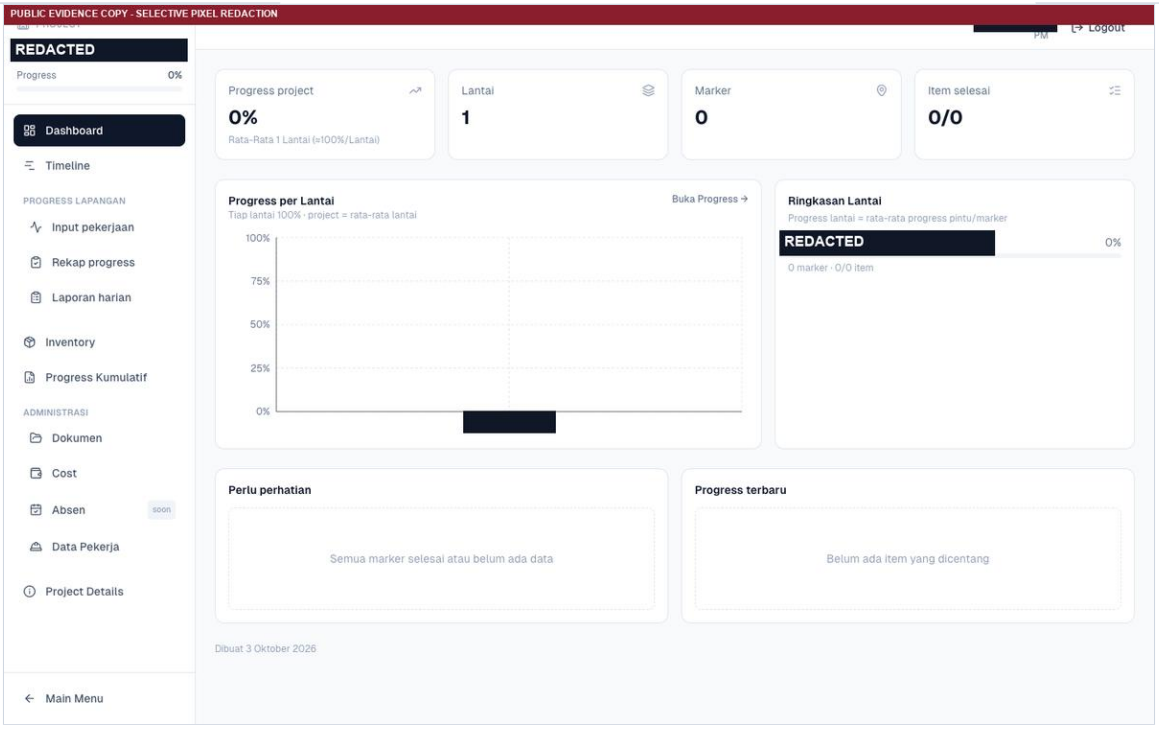


Figure 76 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 18 step 1

2. Click Cost in the sidebar navigation. The screenshot shows the Cost page with the project's own data visible (Ringkasan budget summary), establishing the pre-test state. No cross-project data is visible here; the cross-project check is performed at the API layer in the steps below.

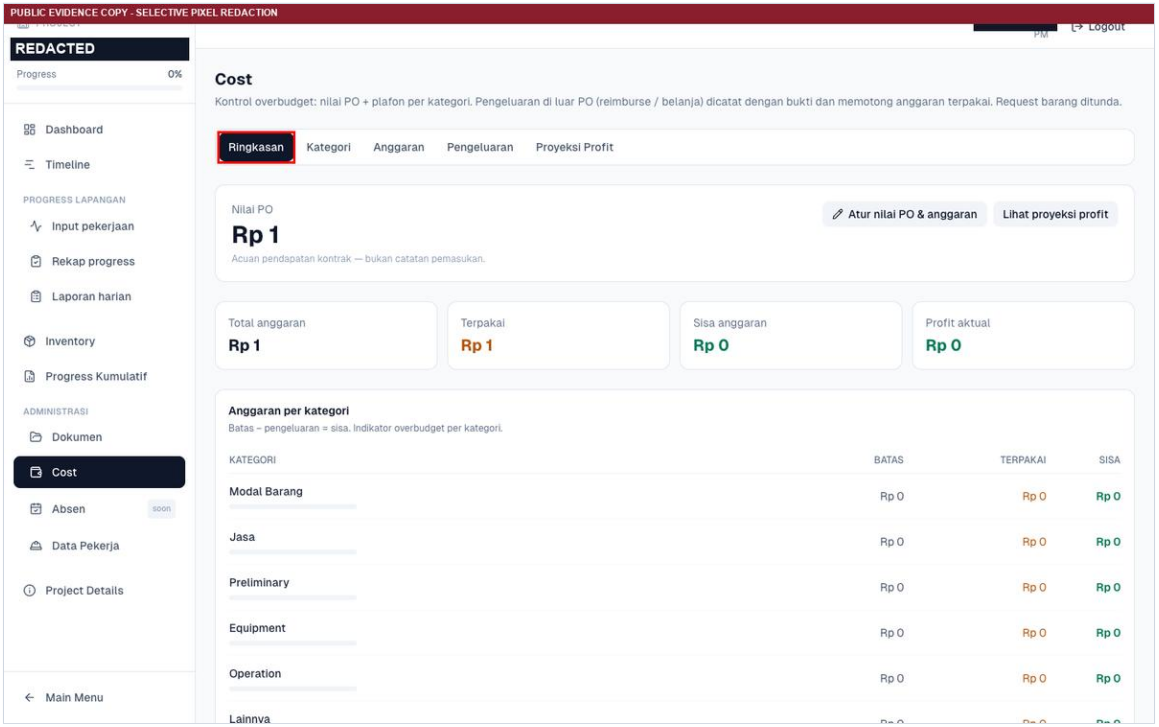


Figure 77 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 18 step 2

3. PM-A requests Project B's receipt file. The screenshot shows the red-boxed foreign file path and the red-boxed HTTP 200 OK response with Content-Type image/jpeg, proving foreign file bytes are served.

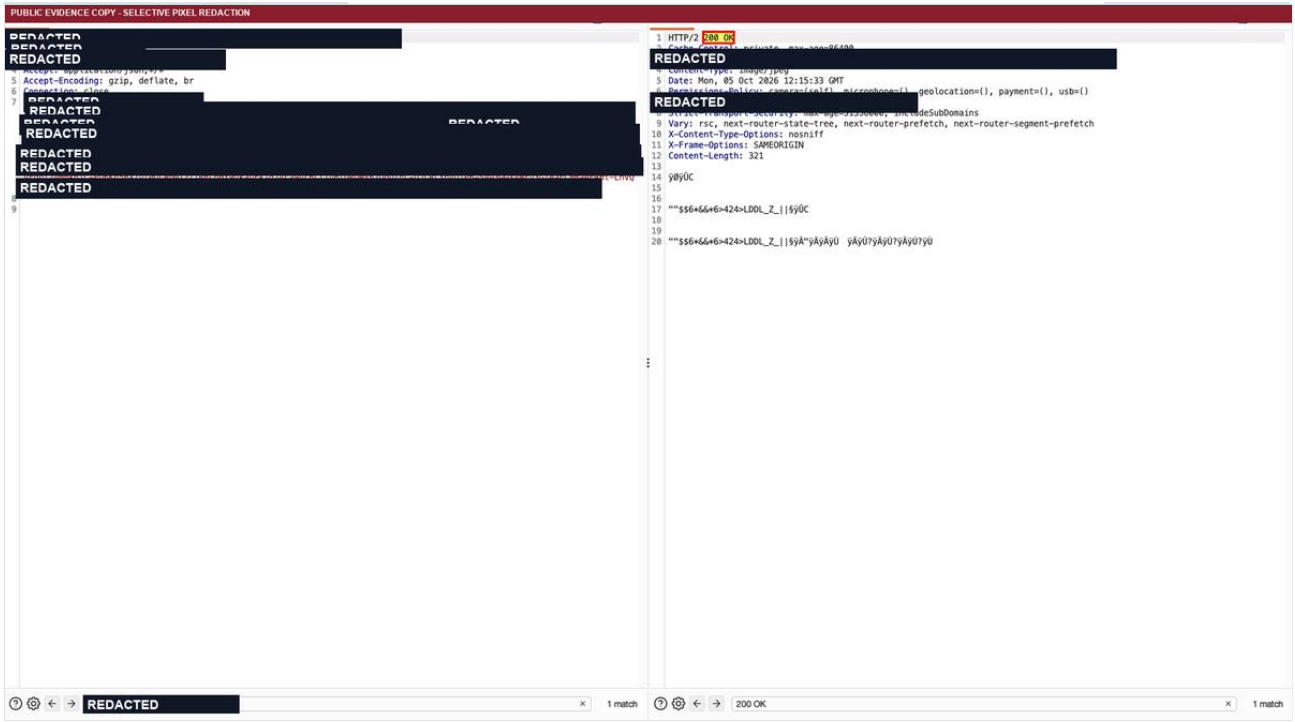


Figure 78 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 18 step 3

4. PM-B requests Project A's receipt file. The screenshot shows the mirrored red-boxed foreign file path and HTTP 200 OK image response, proving the bidirectional failure.

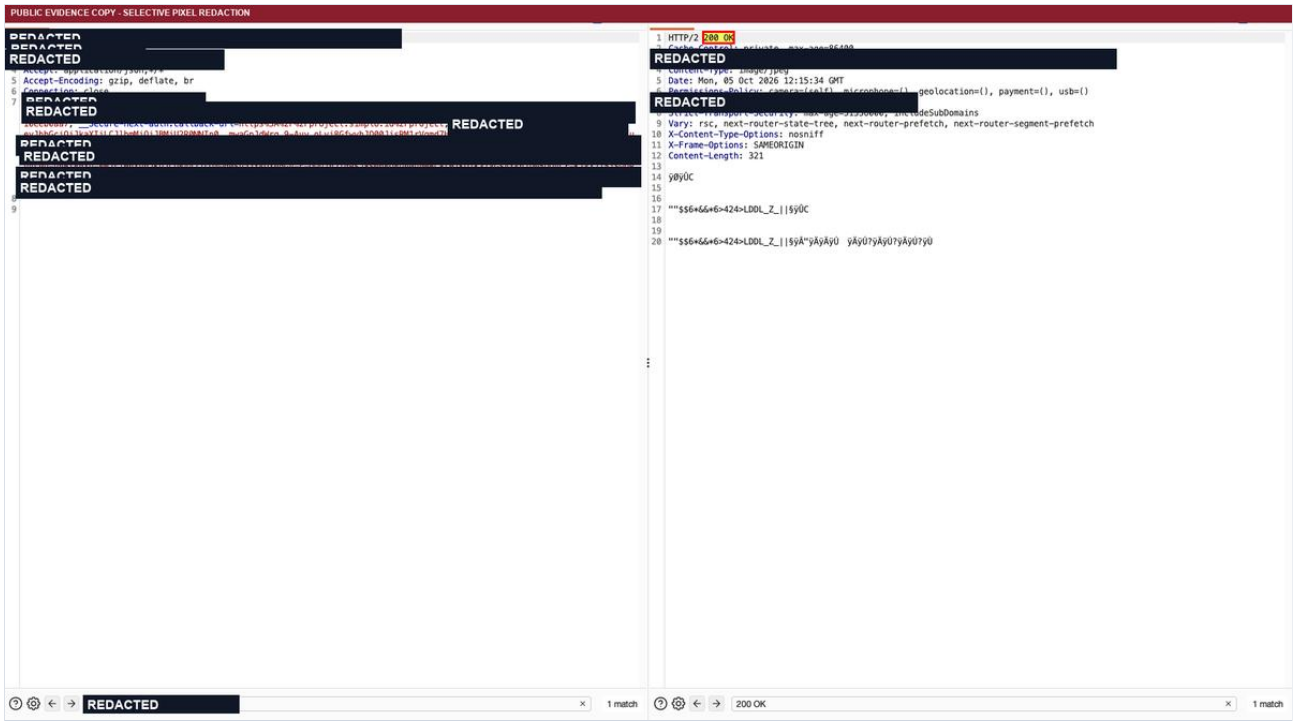


Figure 79 - Broken Object Level Authorization Across Project-Scoped Resources - scenario 18 step 4

3.1.2 Broken Function Level Authorization (API5) – F-02 - Broken Function Level Authorization Across Restricted Features

SEVERITY	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS
Medium	6.5 / 10	Yes	CWE-862

Standard mapping	**API5:2023** Broken Function Level Authorization (OWASP API Security Top 10 2023)
Impact rating	Medium
CVSS vector	`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N`
Affected feature / URL	• Budget / financial lines (read) • Cost records (read) • Personnel records (read) • Documents: metadata and file references (read) • Inventory items (read) • Project details and project list (read)
Testing approach	Grey-box

DESCRIPTION

The server did not consistently enforce the approved role-to-function matrix, even where the interface hid or denied the corresponding feature. Two restricted roles retrieved data from seven restricted read functions by requesting them directly; selected viewer-role controls were correctly denied, which narrows but does not remove the finding.

Attack prerequisites: The demonstrated access used authenticated accounts whose roles were not authorized for the requested function; no victim interaction was required.

Expected: A role that is not authorized for a function is denied by the server even when the function is requested directly, outside the normal interface flow.

Actual: Two restricted roles retrieved data from seven restricted read functions by requesting them directly; selected viewer-role controls were correctly denied, which narrows but does not remove the finding.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

Restricted authenticated roles retrieved financial records, personnel personal-data fields, document metadata/file references, inventory, foreign project details and project-list entries reserved for other roles. This is function-level enforcement failure; cross-project binding is reported separately. No unrestricted file-download or write capability is inferred.

RECOMMENDATION

1. Enforce the approved role-to-function matrix before loading business data.
2. Deny unauthorized direct API calls independently of menu visibility and client-side routing.
3. Centralize policy across list, detail and file functions.
4. Add positive and negative tests for every role/function combination and log denied attempts.

Public-edition evidence: Each recorded step is followed by its corresponding selectively redacted screenshot. Client identity, origins, credentials, opaque identifiers, test-account identifiers, and personal data are removed at the pixel level.

Affected scenario 1 — Budget - Financial Lines

1. Log in as LEADER and record the project menu before testing the restricted Budget API directly.

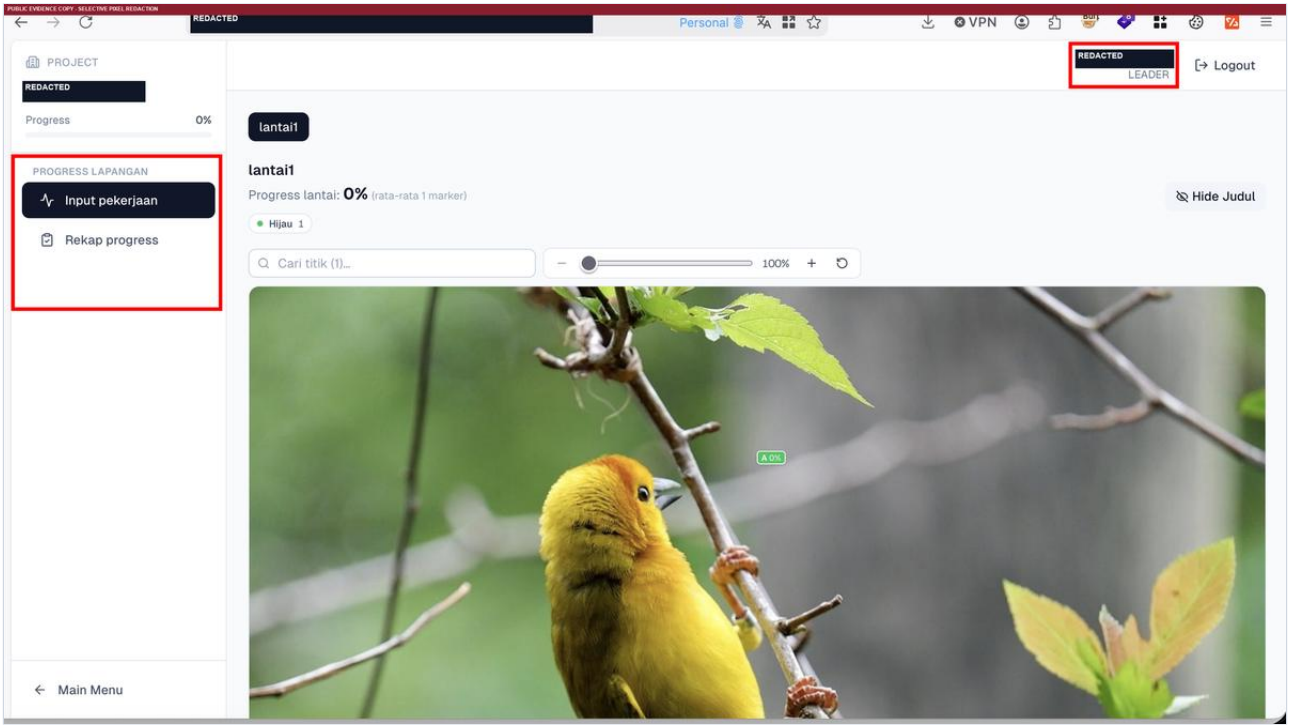


Figure 80 - Broken Function Level Authorization Across Restricted Features - scenario 1 step 1

2. Send the fixture A Budget request as LEADER; the server returns HTTP 200 with the PM financial payload.

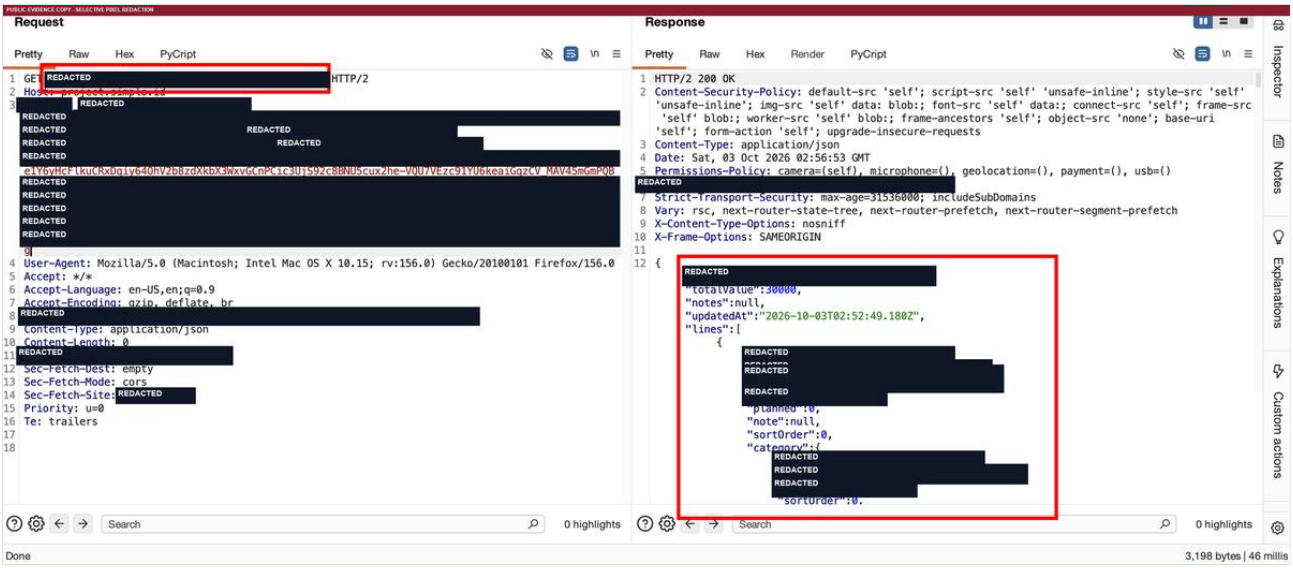


Figure 81 - Broken Function Level Authorization Across Restricted Features - scenario 1 step 2

3. Log in as SM and record the project menu before the direct API request.

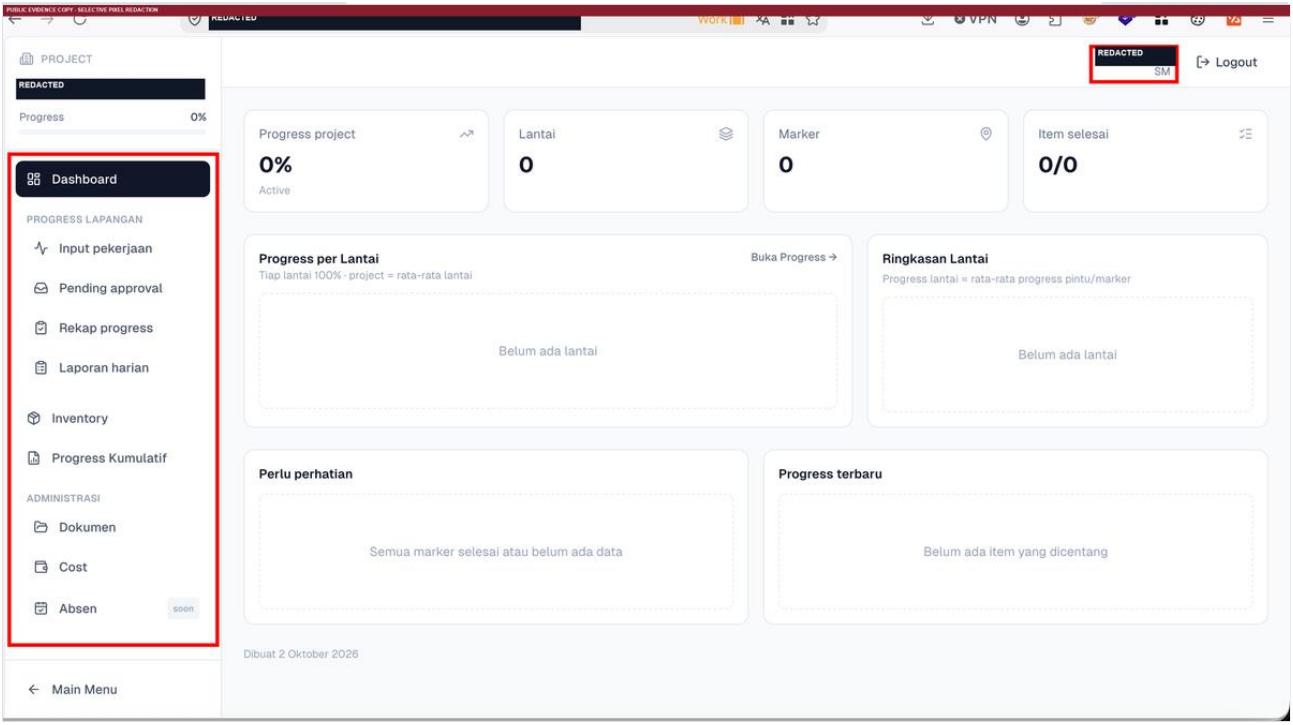


Figure 82 - Broken Function Level Authorization Across Restricted Features - scenario 1 step 3

4. Send the same fixture A Budget request as SM; the server returns HTTP 200 with the same PM financial payload.

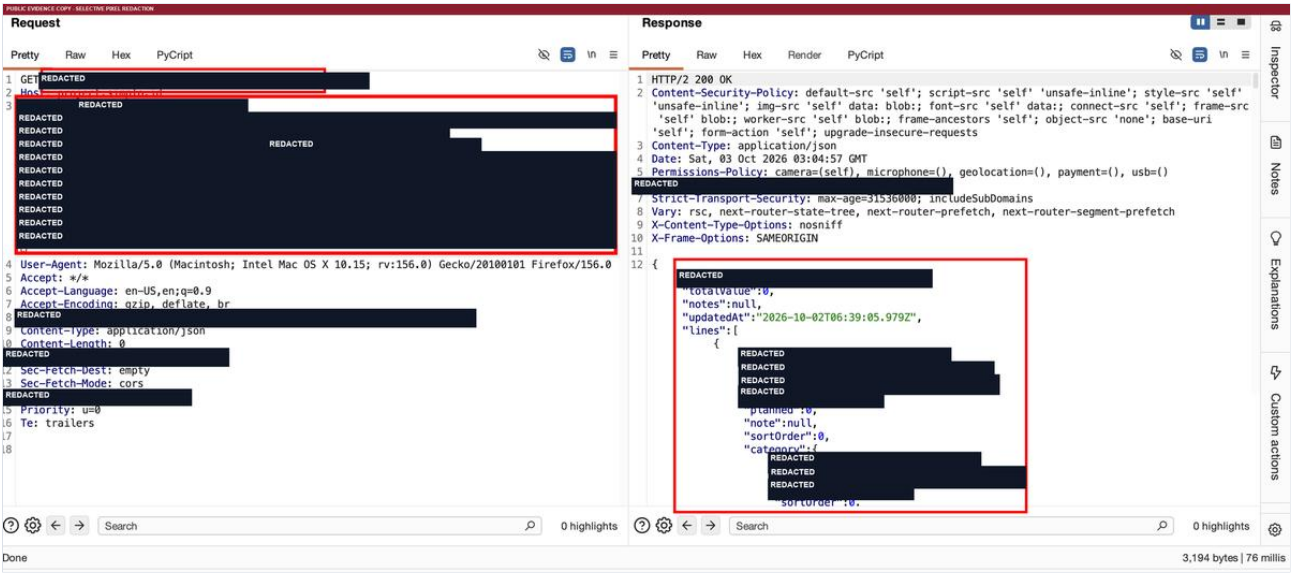


Figure 83 - Broken Function Level Authorization Across Restricted Features - scenario 1 step 4

Affected scenario 2 — Cost - Cost Records

1. Log in as LEADER and confirm that the Cost menu is not available in the application interface.

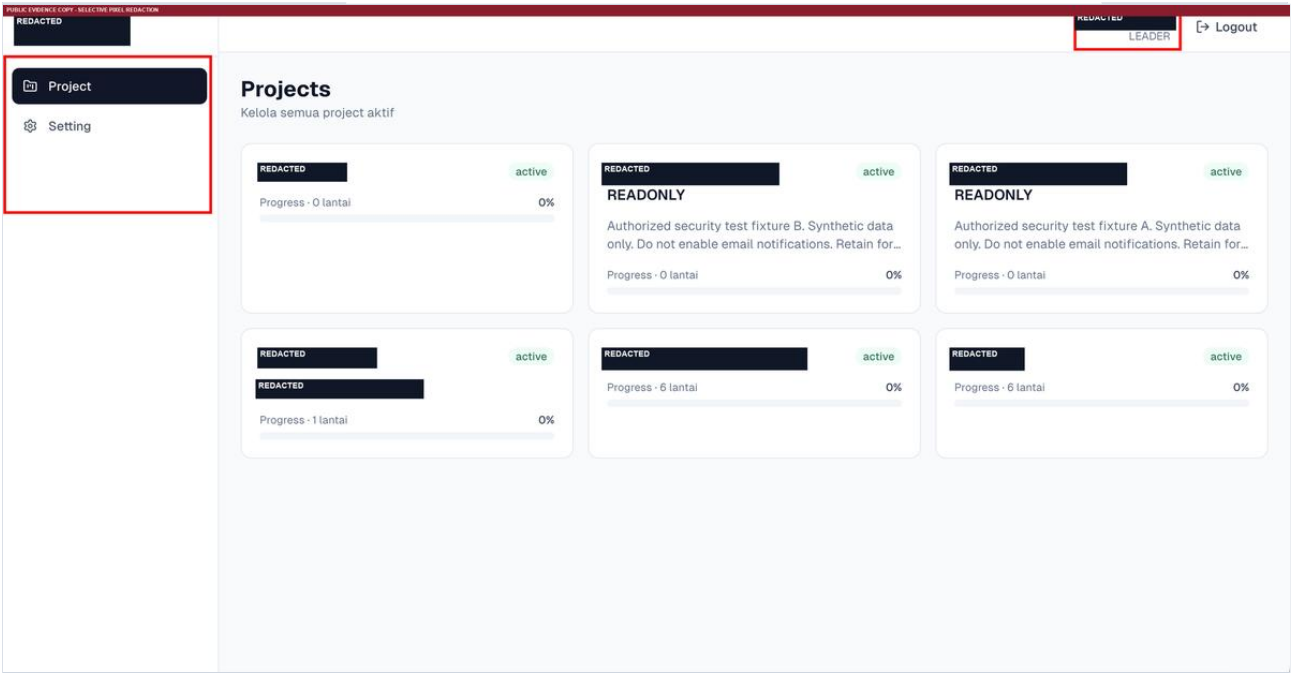


Figure 84 - Broken Function Level Authorization Across Restricted Features - scenario 2 step 1

2. Open the Cost page directly in the browser; the application returns 404 Not Found.

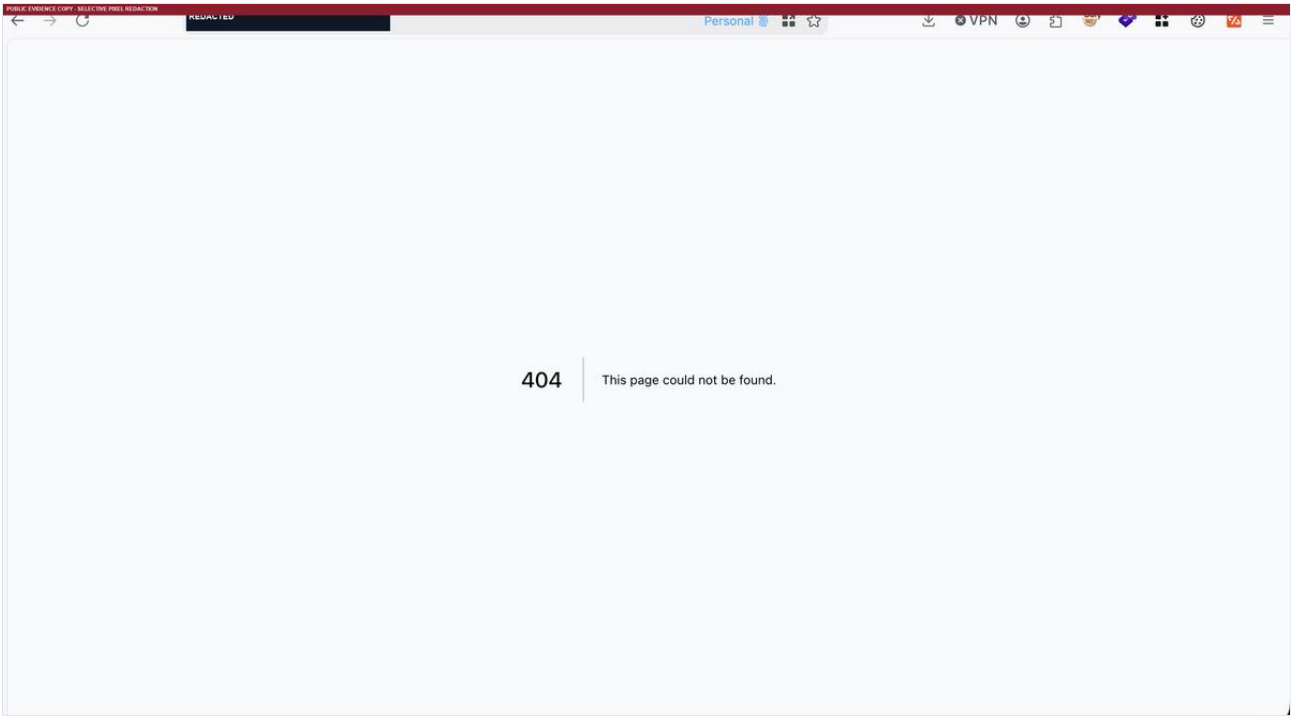


Figure 85 - Broken Function Level Authorization Across Restricted Features - scenario 2 step 2

3. Request the Cost API directly as LEADER; the server returns HTTP 200 with the same synthetic financial record returned to PM.

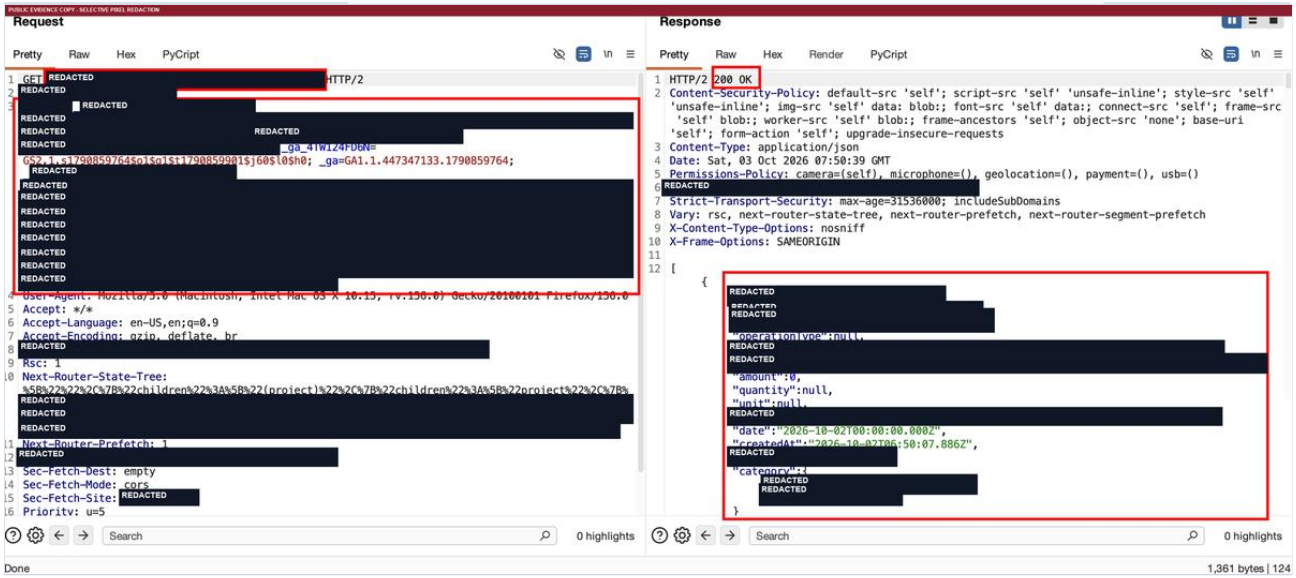


Figure 86 - Broken Function Level Authorization Across Restricted Features - scenario 2 step 3

Affected scenario 3 — Data Pekerja - Worker API

1. Request the fixture A Worker API as PM; the server returns HTTP 200 with the authorized synthetic worker record.

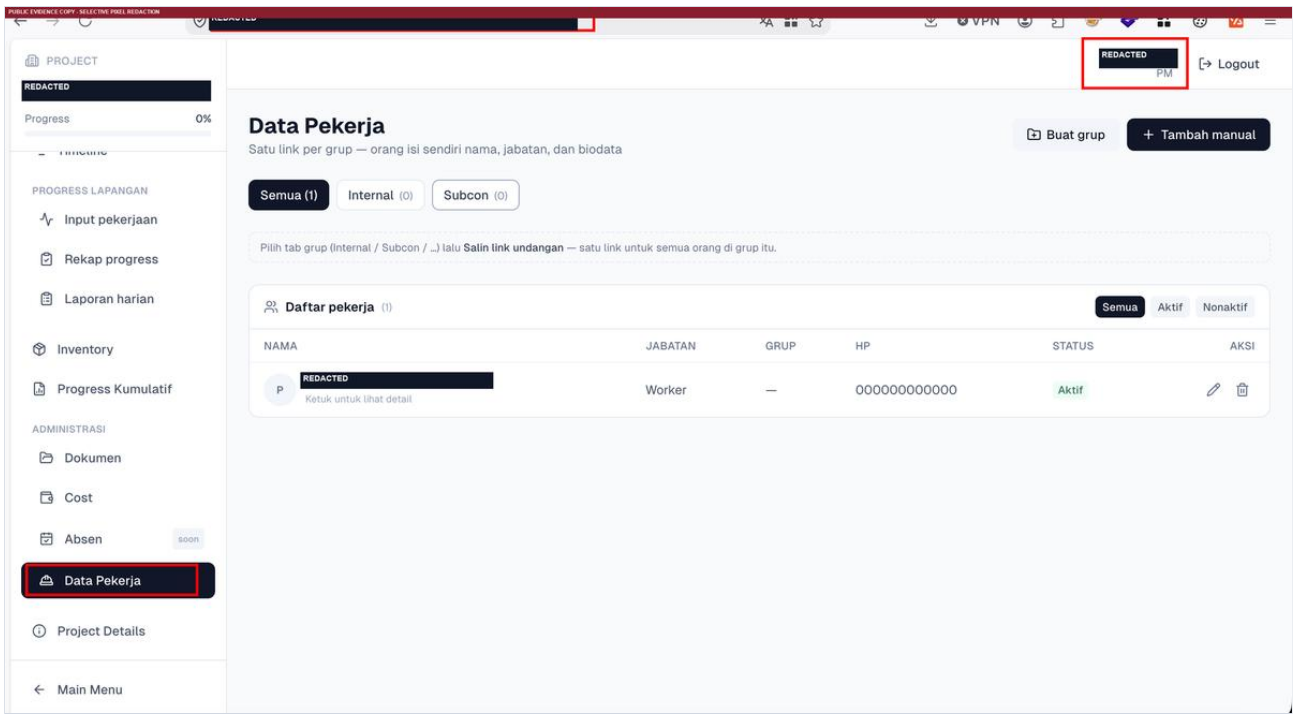


Figure 87 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 1

2. Log in as LEADER and confirm that the Data Pekerja menu is not available in the application interface.

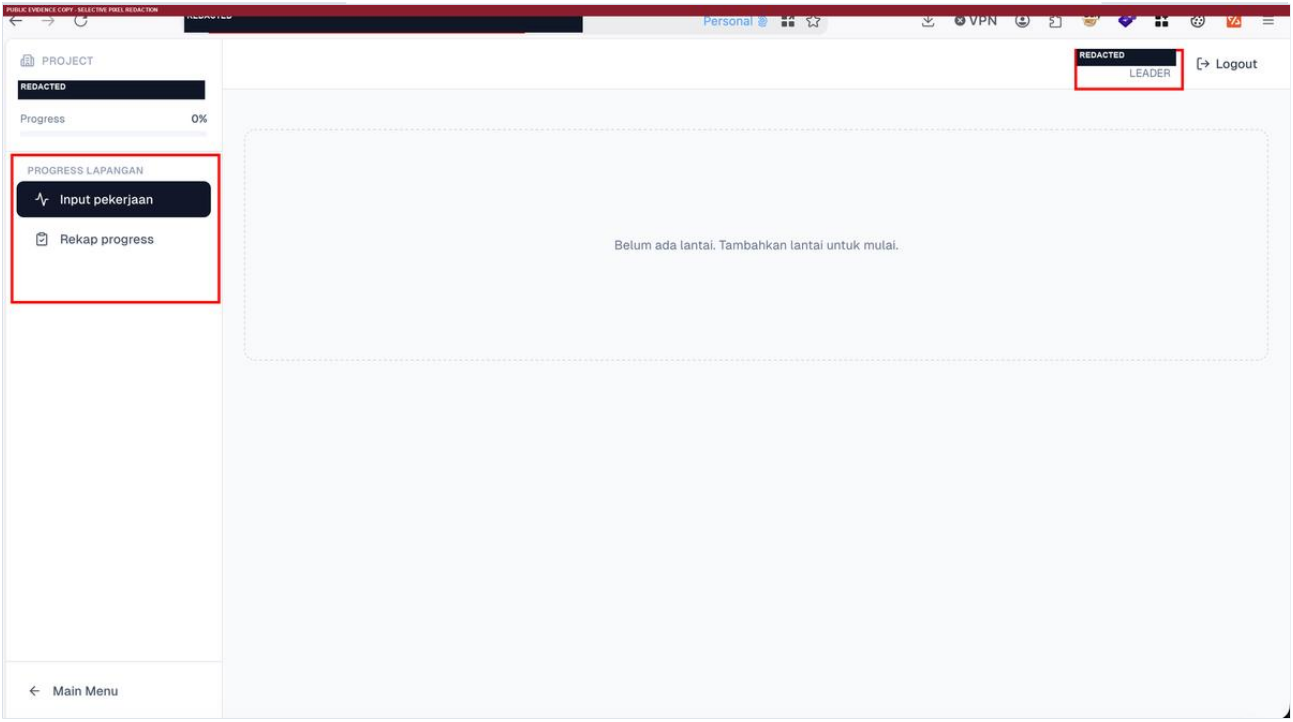


Figure 88 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 2

3. Open the Worker page directly as LEADER; the application returns 404 Not Found.

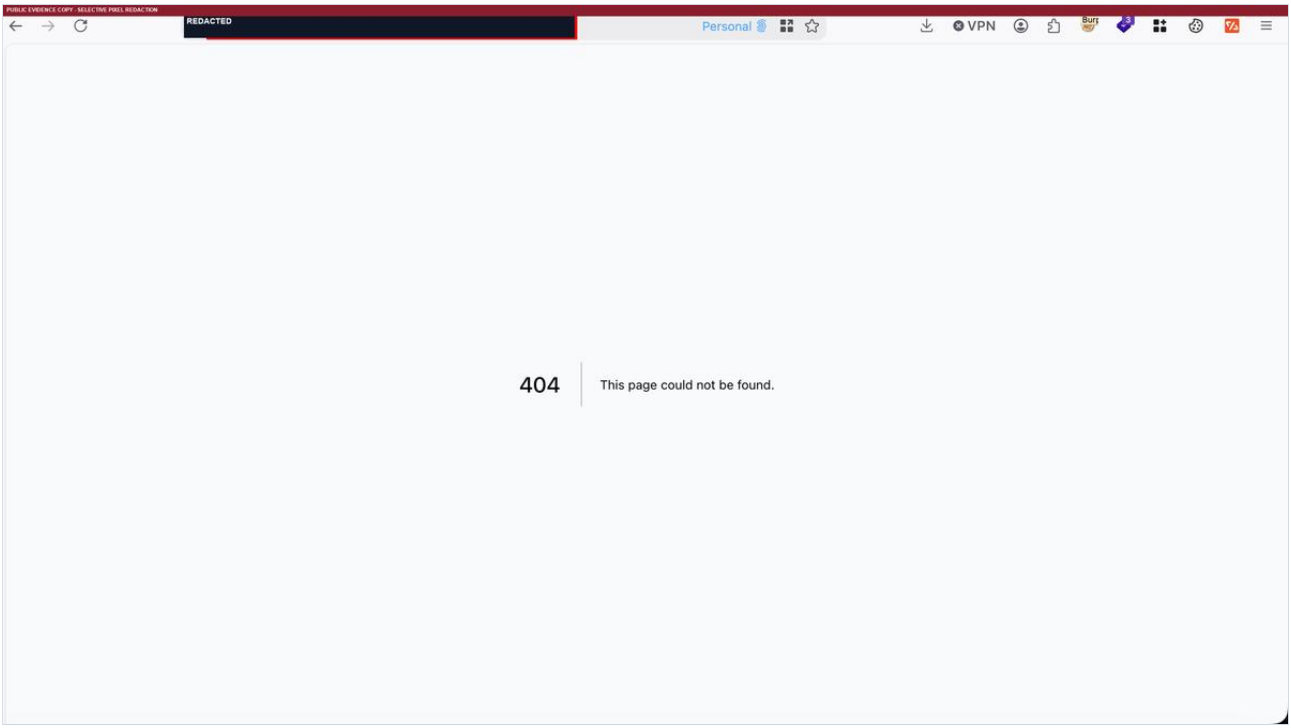


Figure 89 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 3

4. Request the same Worker API directly as LEADER; the server returns HTTP 200 with the same worker PII returned to PM.

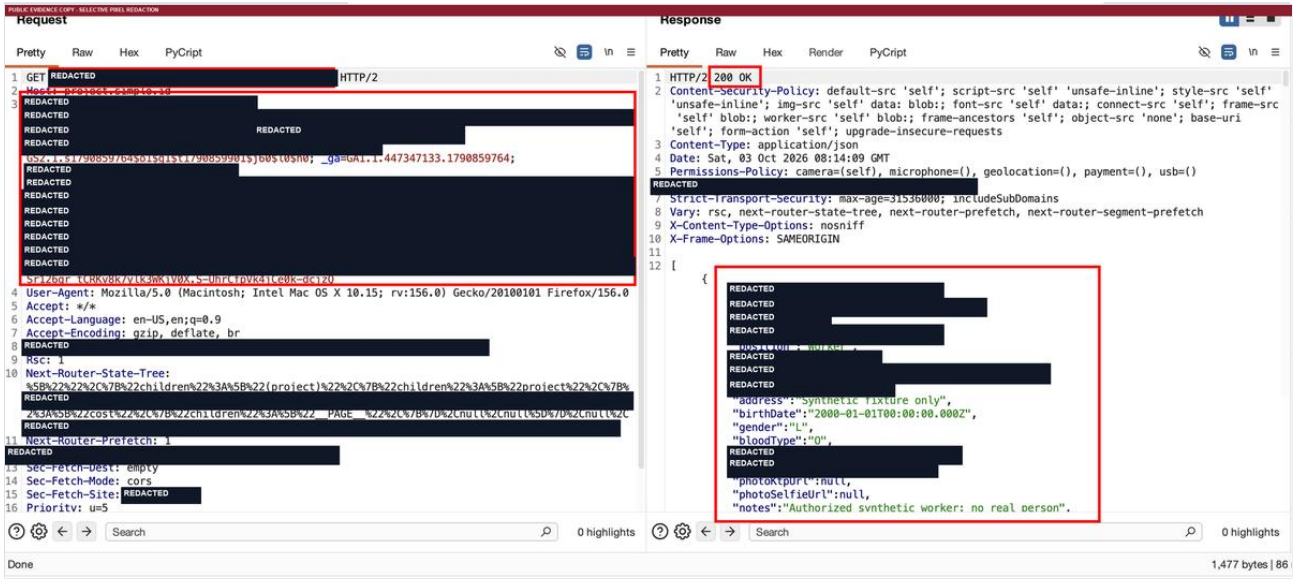


Figure 90 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 4

5. Request the same Worker API directly as SM; the server returns HTTP 200 with the same worker PII returned to PM.

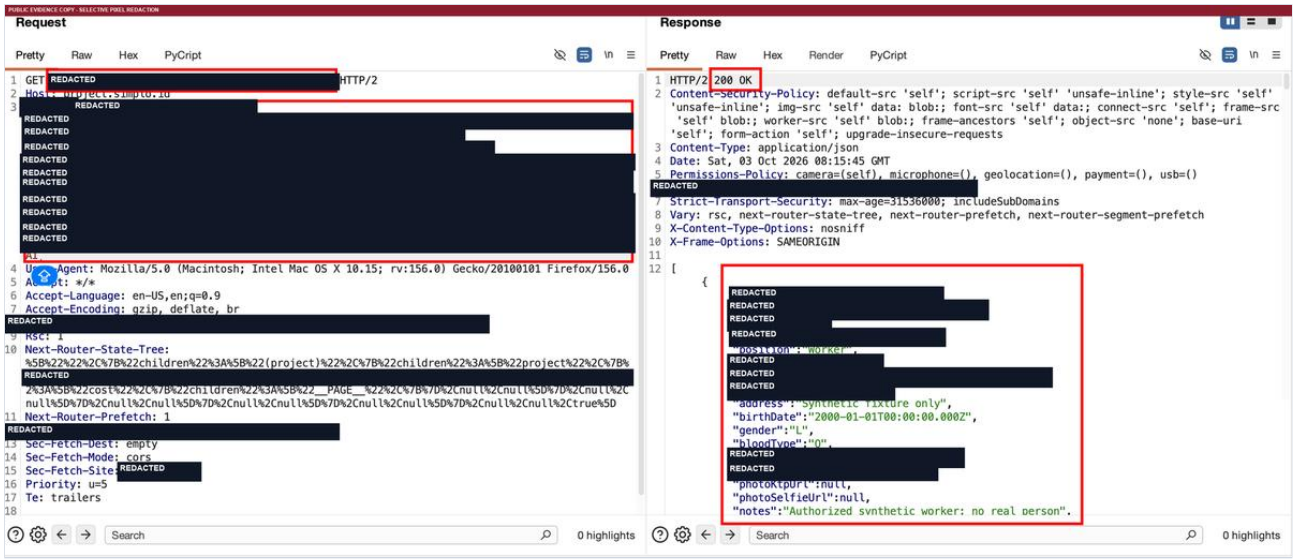


Figure 91 - Broken Function Level Authorization Across Restricted Features - scenario 3 step 5

Affected scenario 4 — Dokumen - Document Metadata

1. Log in as PM and confirm that the Dokumen menu provides the authorized document view.

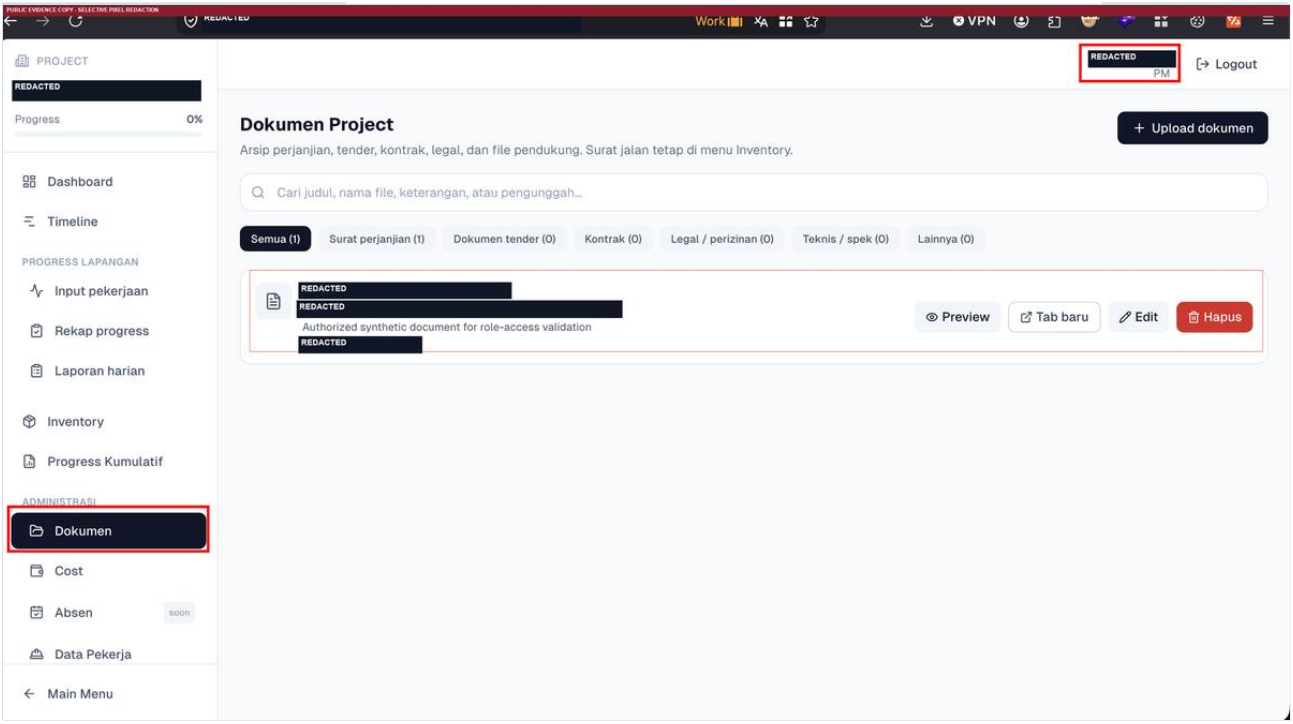


Figure 92 - Broken Function Level Authorization Across Restricted Features - scenario 4 step 1

2. Log in as LEADER and confirm that the Dokumen menu is not available in the application interface.

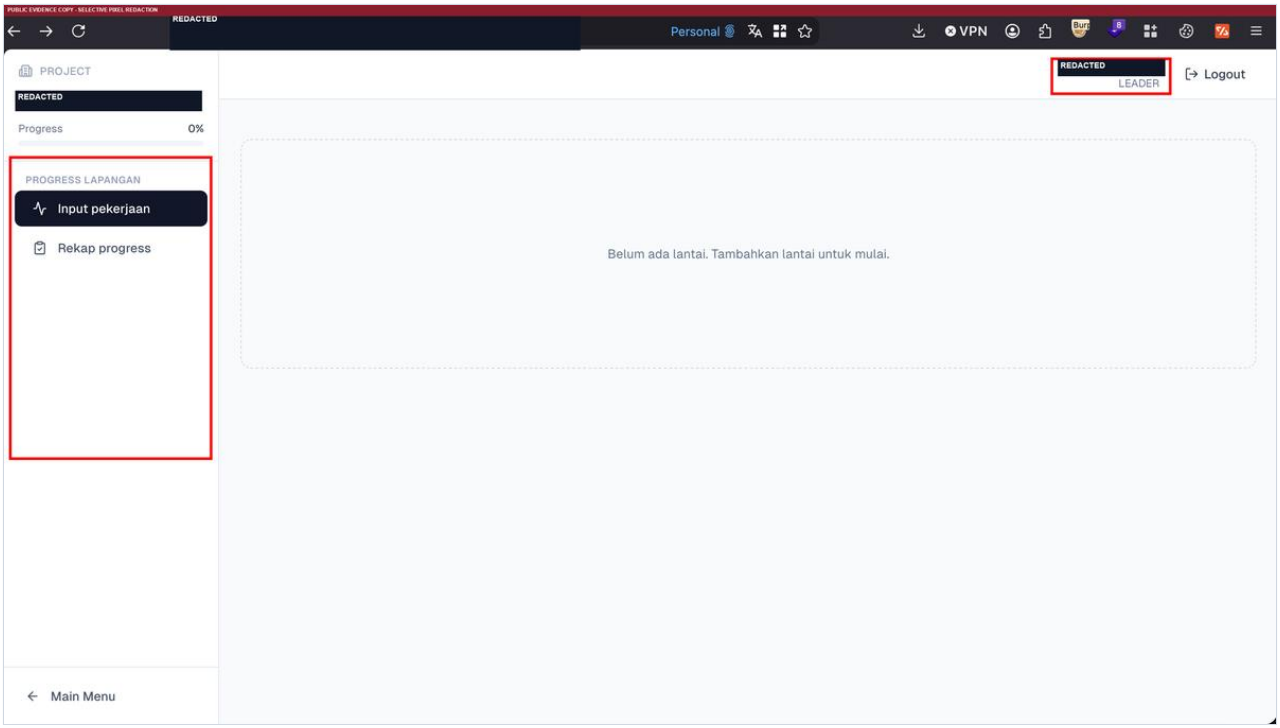


Figure 93 - Broken Function Level Authorization Across Restricted Features - scenario 4 step 2

3. Request the Document list API directly as LEADER; the server returns HTTP 200 with the same synthetic document metadata and file URL returned to PM.

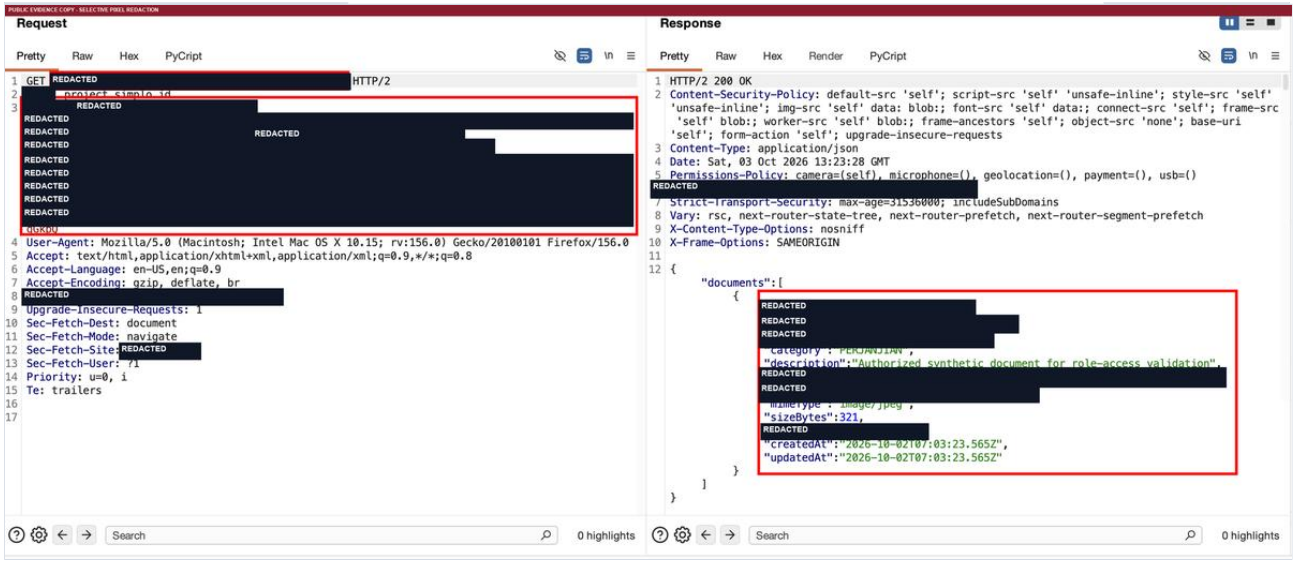


Figure 94 - Broken Function Level Authorization Across Restricted Features - scenario 4 step 3

Affected scenario 5 — Inventory - Inventory Items

1. Log in with the authorized test account account and open the Project A dashboard; the page establishes the PM role and project context.

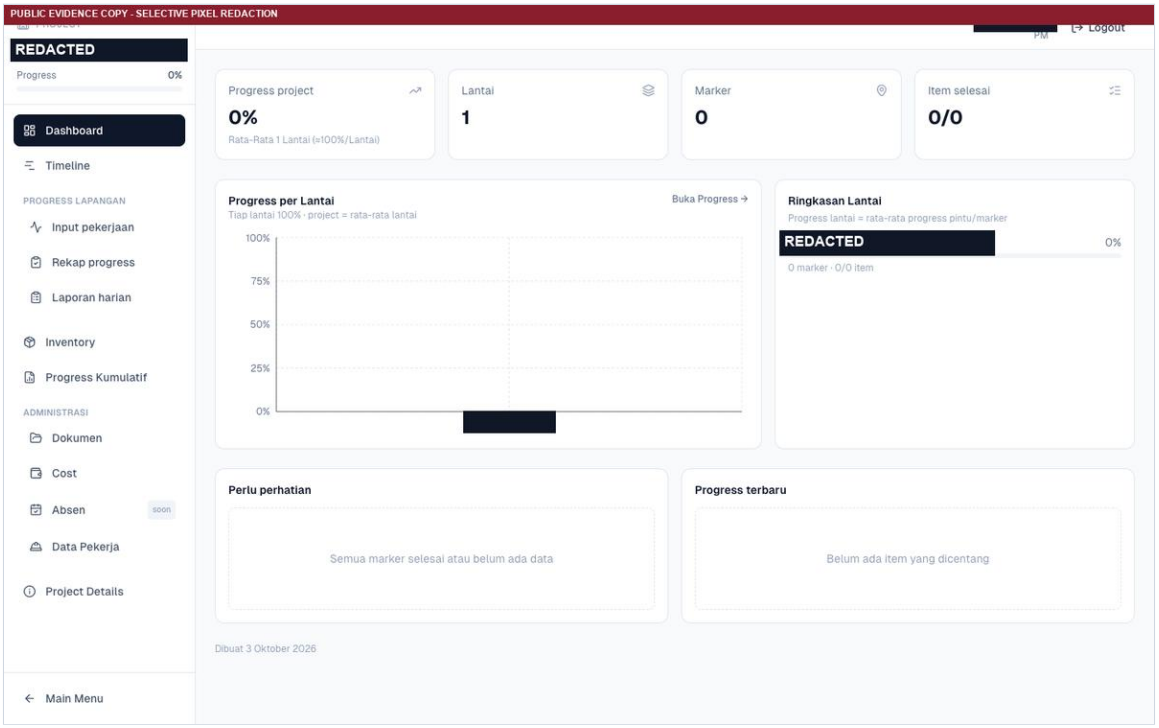


Figure 95 - Broken Function Level Authorization Across Restricted Features - scenario 5 step 1

2. Open Inventory from the sidebar; the page displays the fixture stock rows and establishes the authorized baseline.

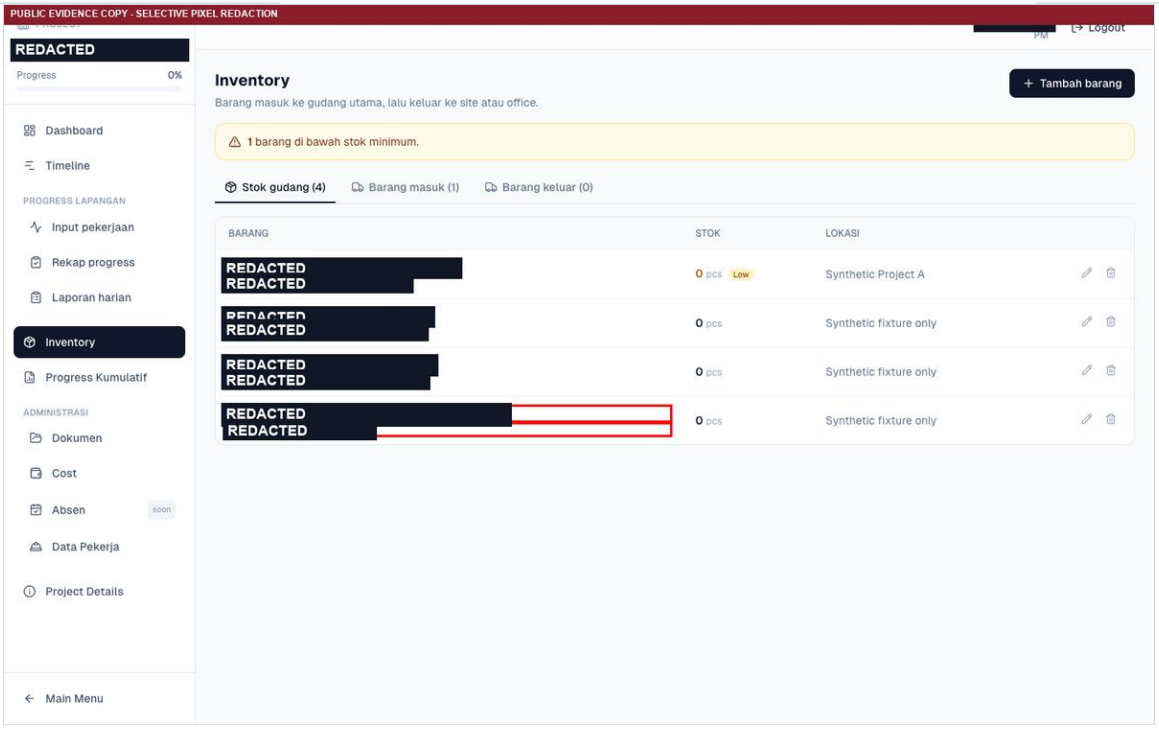


Figure 96 - Broken Function Level Authorization Across Restricted Features - scenario 5 step 2

3. Request the same Inventory API directly as LEADER; the HTTP 200 response contains the [TEST FIXTURE] rows despite the role exclusion.

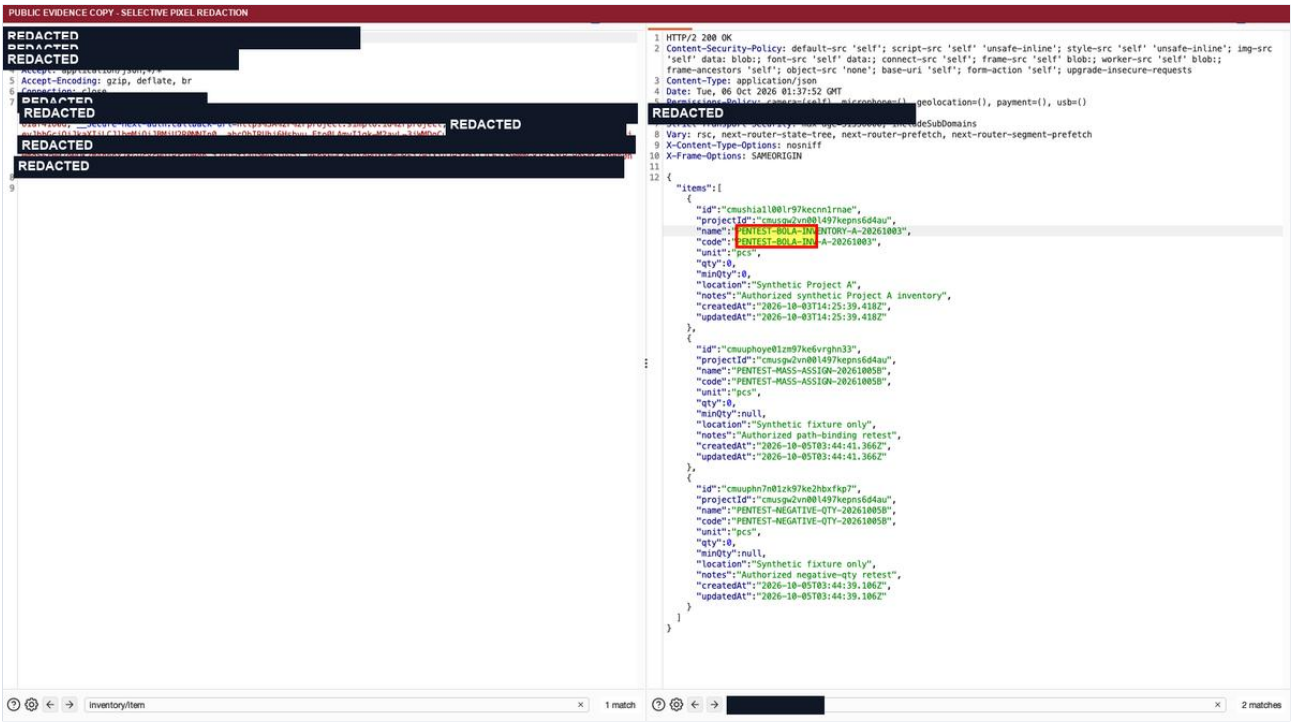


Figure 97 - Broken Function Level Authorization Across Restricted Features - scenario 5 step 3

Affected scenario 6 — Project Details - Non-Member Access

1. Log in with the authorized test account account and open the Project A dashboard to establish the project context.

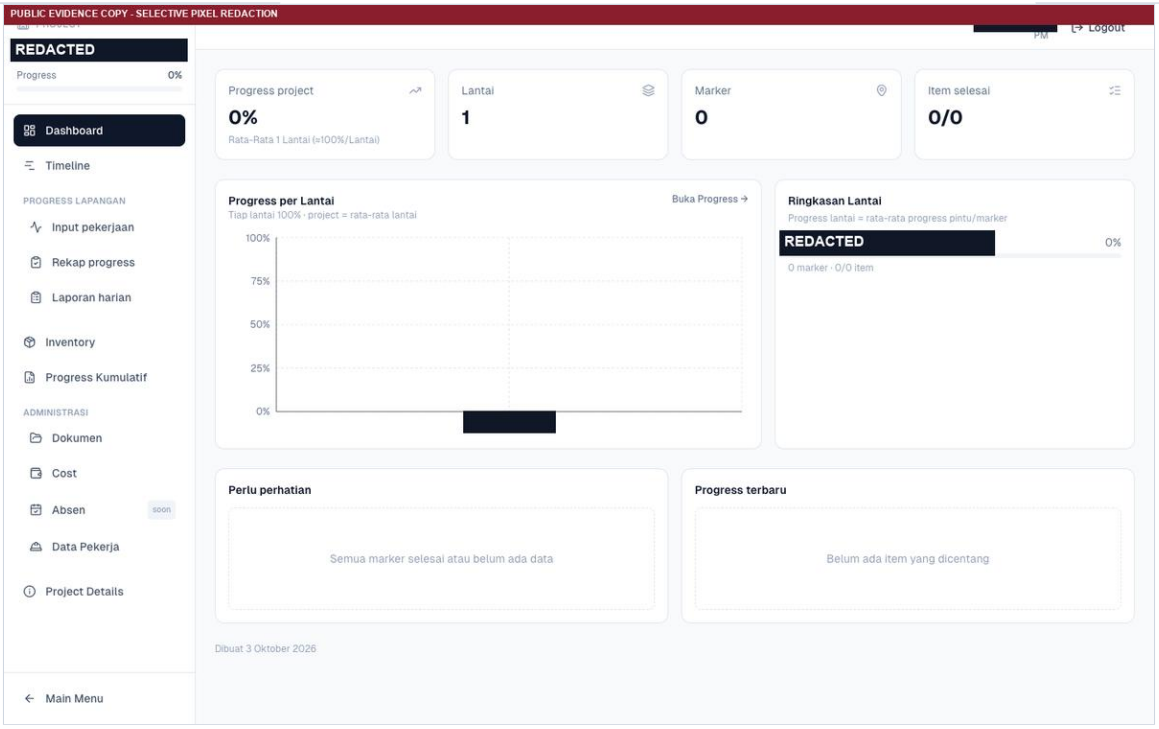


Figure 98 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 1

2. Open Project Details; the own-project page displays the Struktur Organisasi card and establishes the normal UI flow.

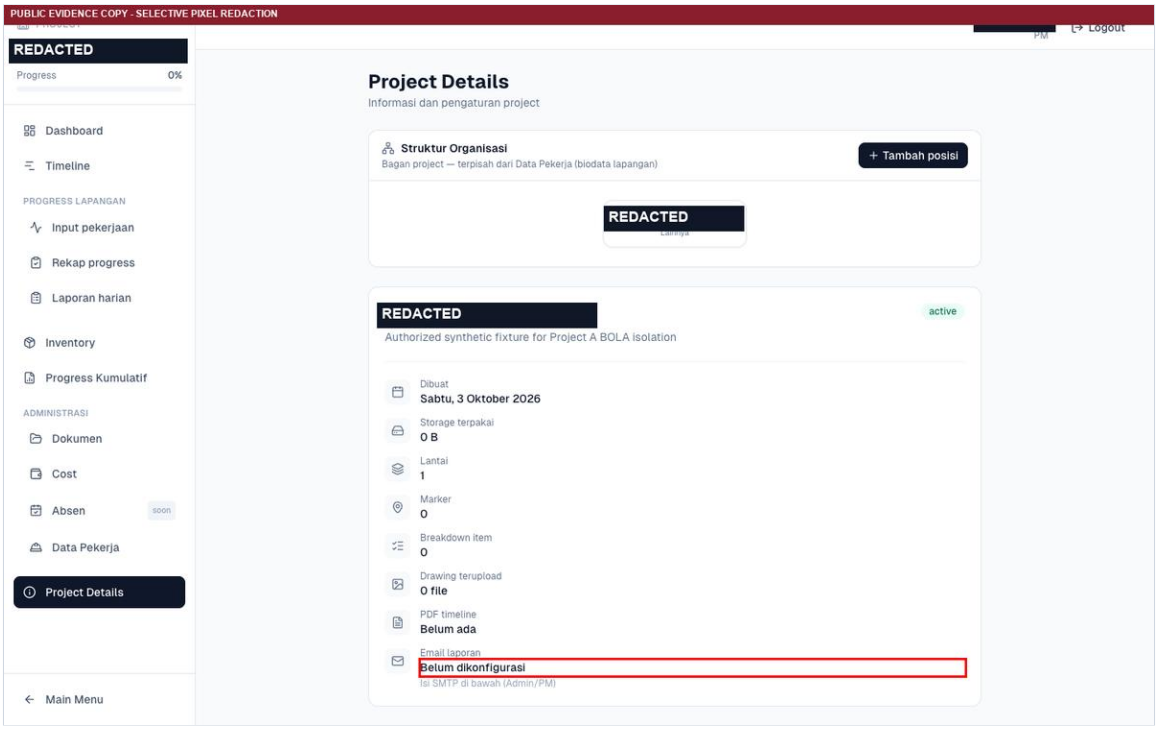


Figure 99 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 2

3. Request fixture B as a non-member CLIENT; the server returns HTTP 403, proving that the membership control exists.

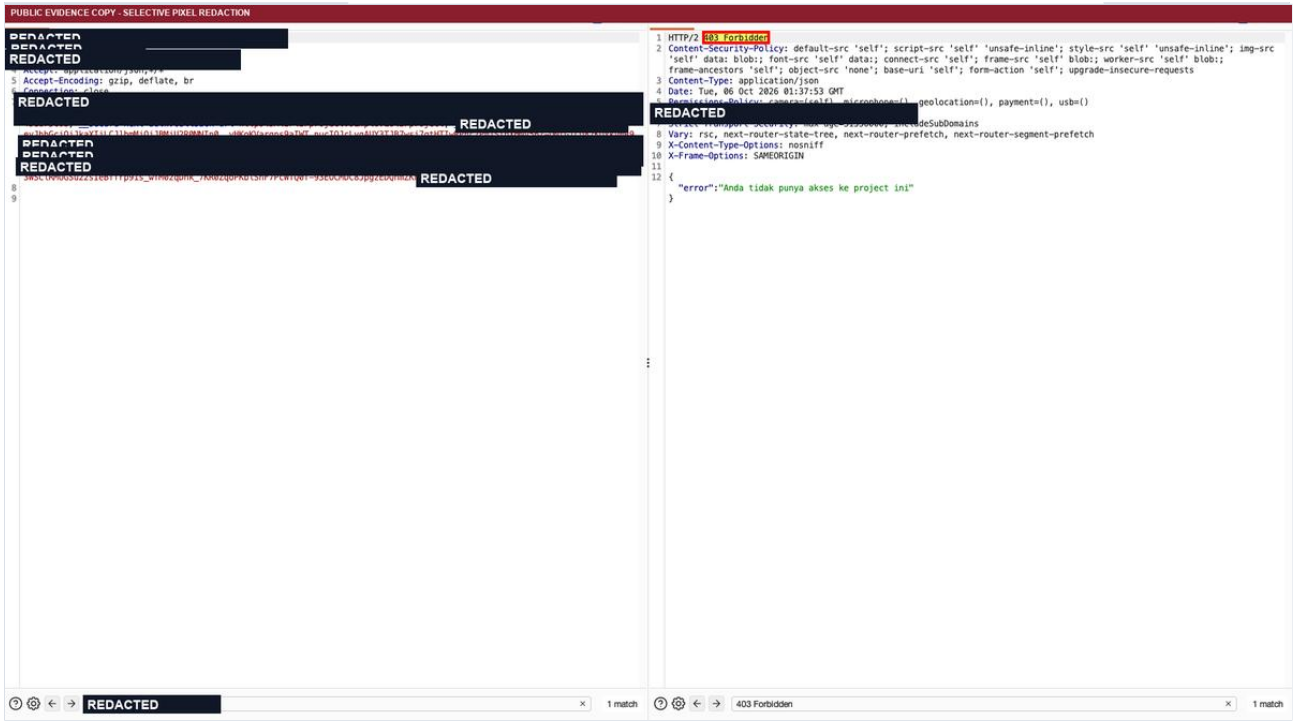


Figure 100 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 3

4. Request the same fixture B object as a non-member LEADER; the server returns HTTP 200 with the foreign members roster and user PII.

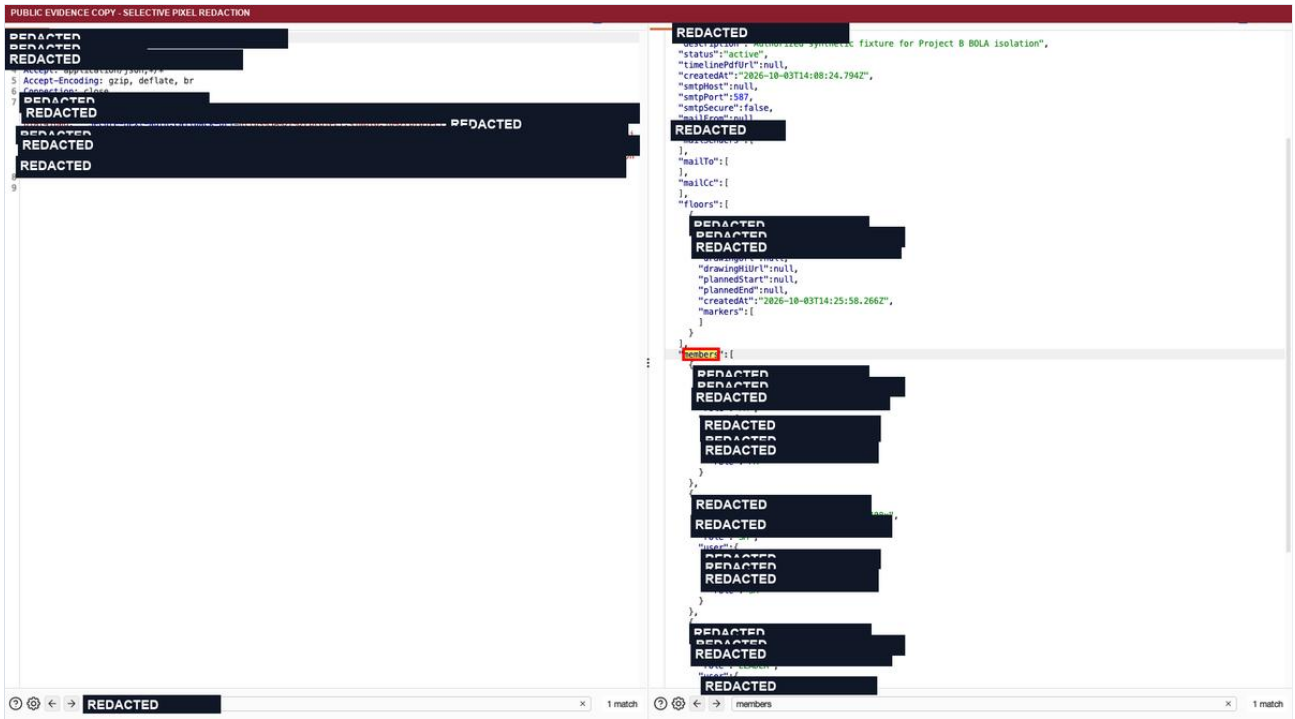


Figure 101 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 4

5. Request the same fixture B object as a non-member SM; the server returns HTTP 200 with the foreign members roster, confirming a second role bypass.

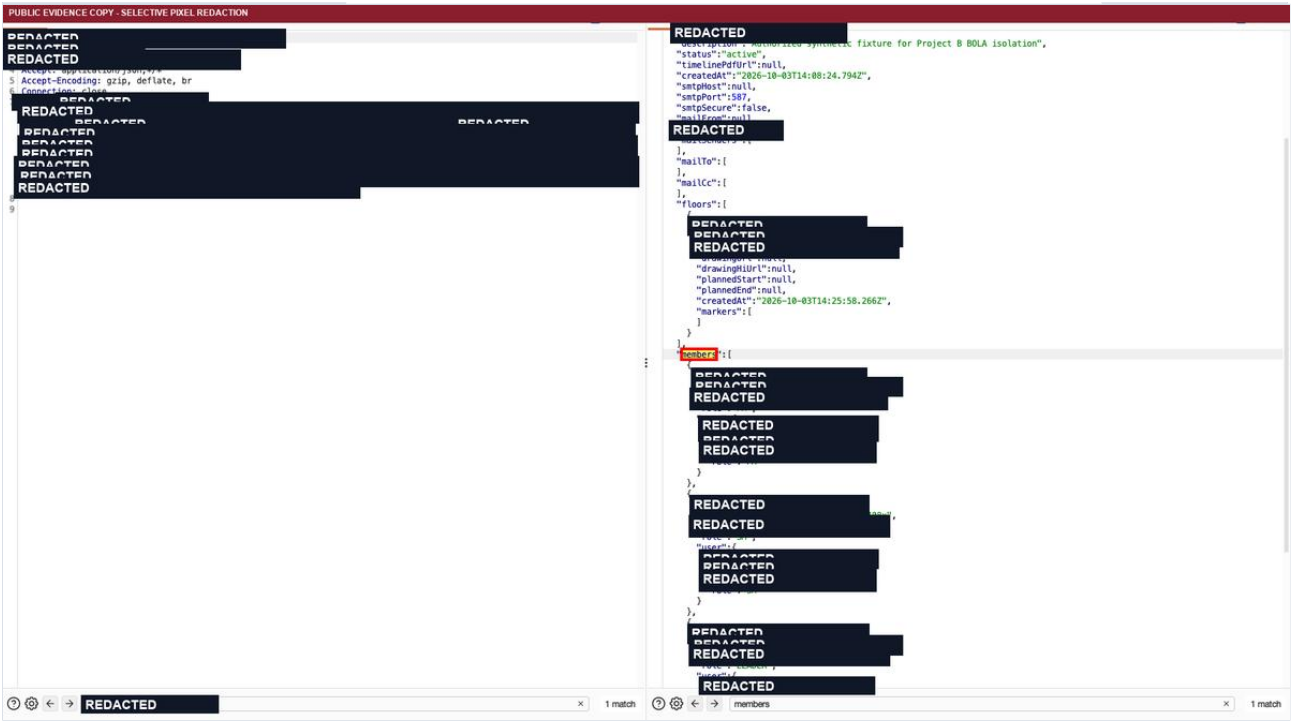


Figure 102 - Broken Function Level Authorization Across Restricted Features - scenario 6 step 5

Affected scenario 7 — Project List - Tenant Enumeration

1. Log in as CLIENT and open the Project page; the interface displays only the assigned [TEST FIXTURE] project.

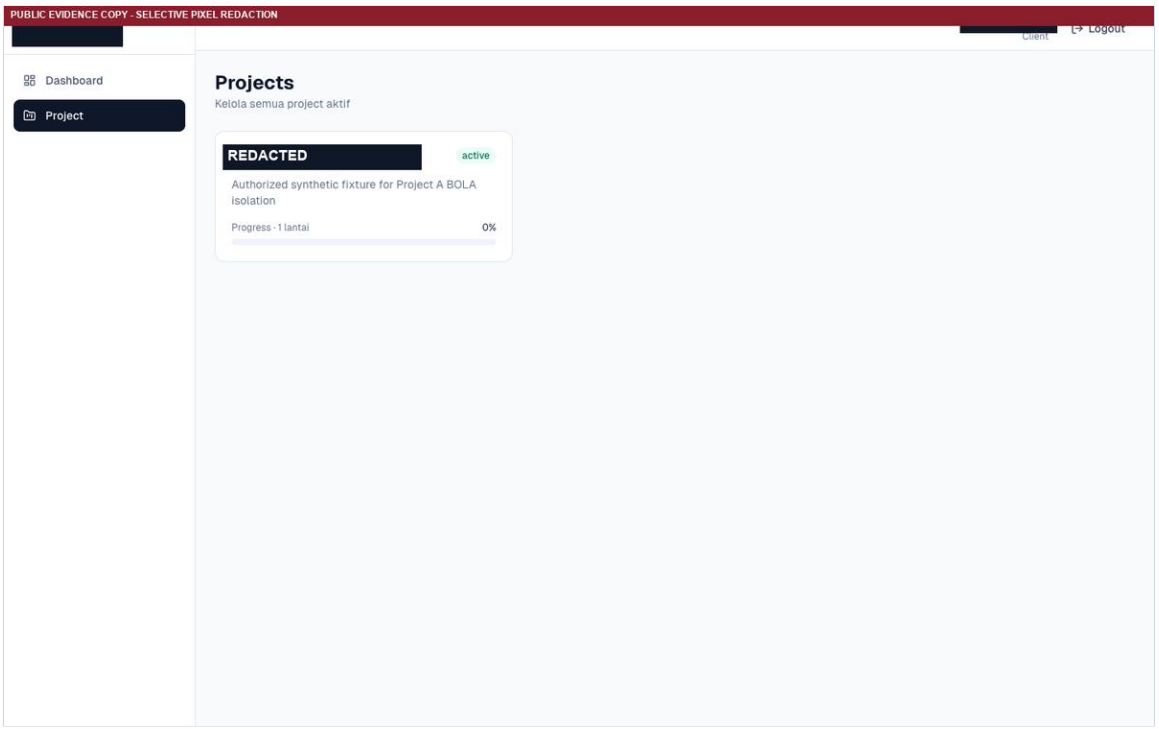


Figure 103 - Broken Function Level Authorization Across Restricted Features - scenario 7 step 1

2. Log in as PM and open the Project page to establish the collection context used in the API checks.

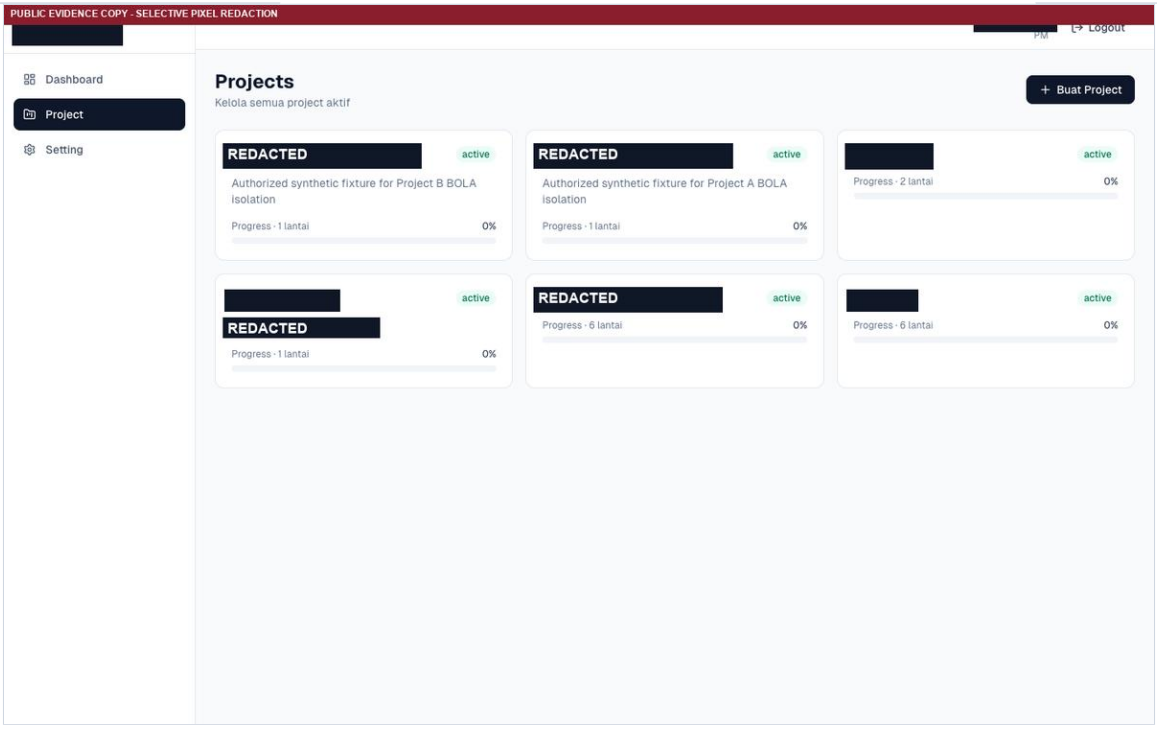


Figure 104 - Broken Function Level Authorization Across Restricted Features - scenario 7 step 2

3. Request the project collection as LEADER; the HTTP 200 response includes foreign project [OBJECT ID] despite no membership.

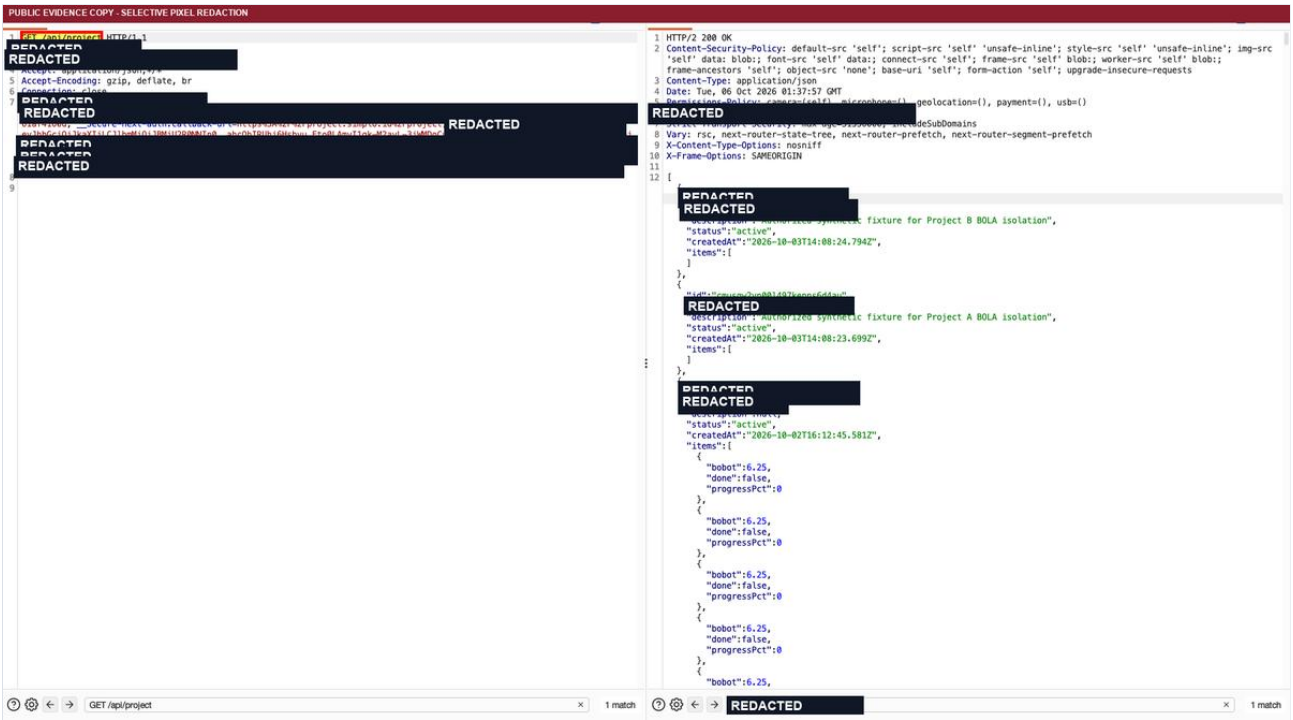


Figure 105 - Broken Function Level Authorization Across Restricted Features - scenario 7 step 3

4. Request the project collection as SM; the HTTP 200 response includes the same foreign project, confirming tenant enumeration by a second role.

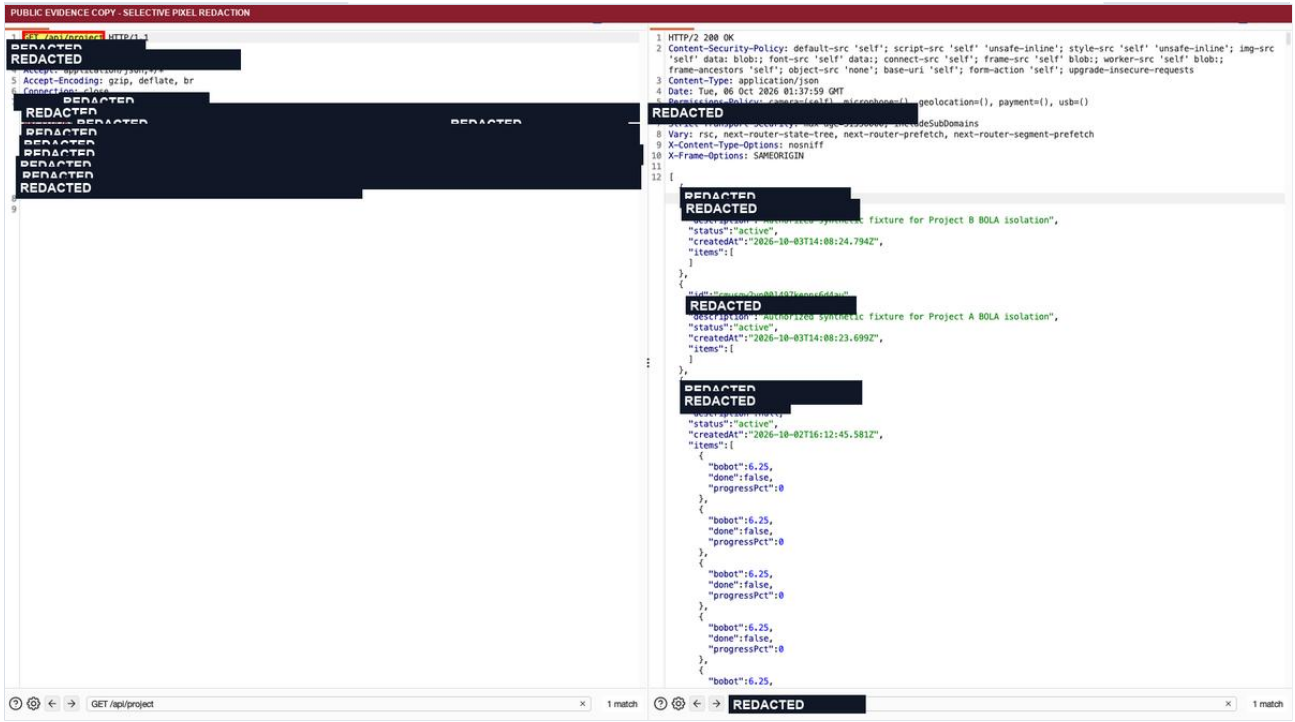


Figure 106 - Broken Function Level Authorization Across Restricted Features - scenario 7 step 4

3.1.3 Unrestricted Access to Sensitive Business Flows (API6) – F-04 - Arbitrary Historical Date Accepted for Submitted Daily Reports

SEVERITY	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS
Medium	4.3 / 10	Yes	CWE-841

Standard mapping	**API6:2023** Unrestricted Access to Sensitive Business Flows (OWASP API Security Top 10 2023)
Impact rating	Medium
CVSS vector	`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N`
Affected feature / URL	Daily reporting (submission and stored report date)
Testing approach	Grey-box

DESCRIPTION

The submission flow trusted a client-controlled report date without enforcing the active reporting window on the server side. A report dated far before the active reporting period was stored and displayed as submitted.

Attack prerequisites: An authenticated account with daily-report submission permission for its own project was used; no victim interaction was required.

Expected: The report date is derived from, or validated against, the server-side active reporting window; impossible historical dates are rejected.

Actual: A report dated far before the active reporting period was stored and displayed as submitted.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

An account with reporting permission created and retrieved a submitted report far outside the active reporting period, compromising record integrity and audit chronology.

RECOMMENDATION

1. Derive report dates from the server-side active work date and enforce an allowed reporting window.
2. Reject out-of-window dates; permit duplicate or retroactive corrections only through an explicit audited workflow.
3. Add past, future, timezone and closed-period boundary tests.

EVIDENCE

Public-edition evidence: Each recorded step is followed by its corresponding selectively redacted screenshot. Client identity, origins, credentials, opaque identifiers, test-account identifiers, and personal data are removed at the pixel level.

Affected scenario 1 — Daily Report - Historical Date Submission

1. Prepare a Daily Report for the active 2026 work date and select Submit Daily Report.

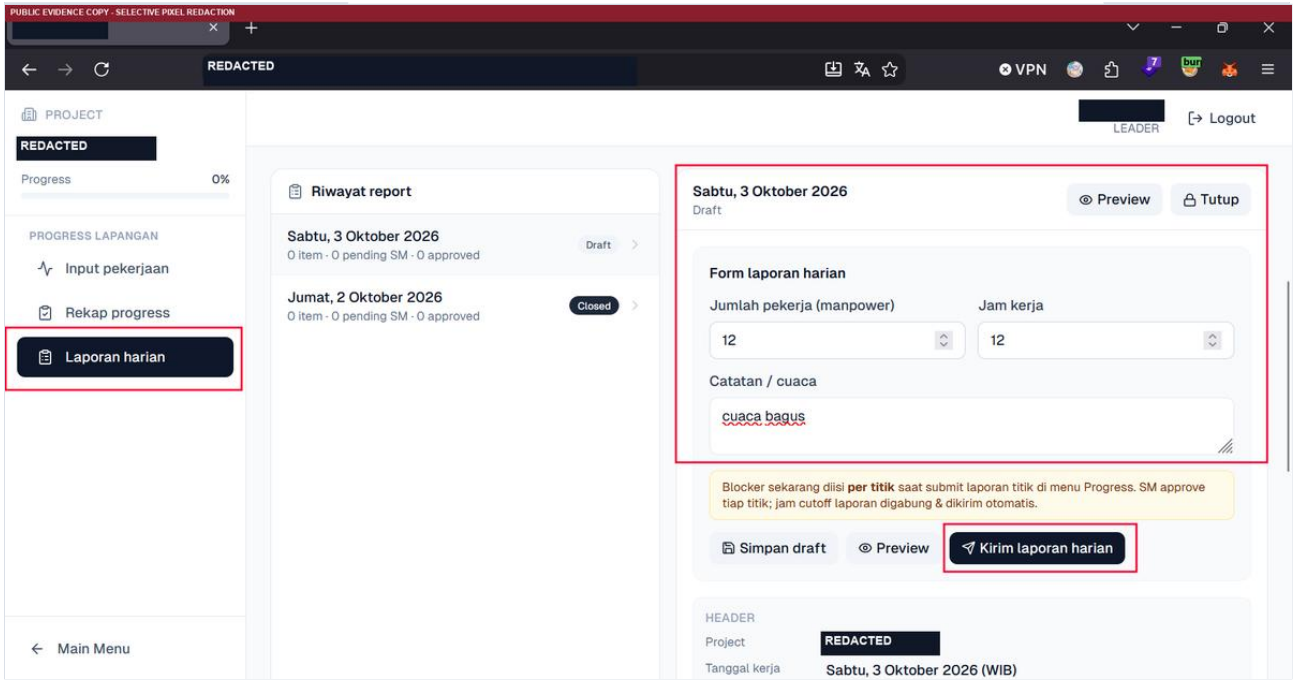


Figure 107 - Arbitrary Historical Date Accepted for Submitted Daily Reports - scenario 1 step 1

2. Intercept the request and identify the client-controlled date parameter.

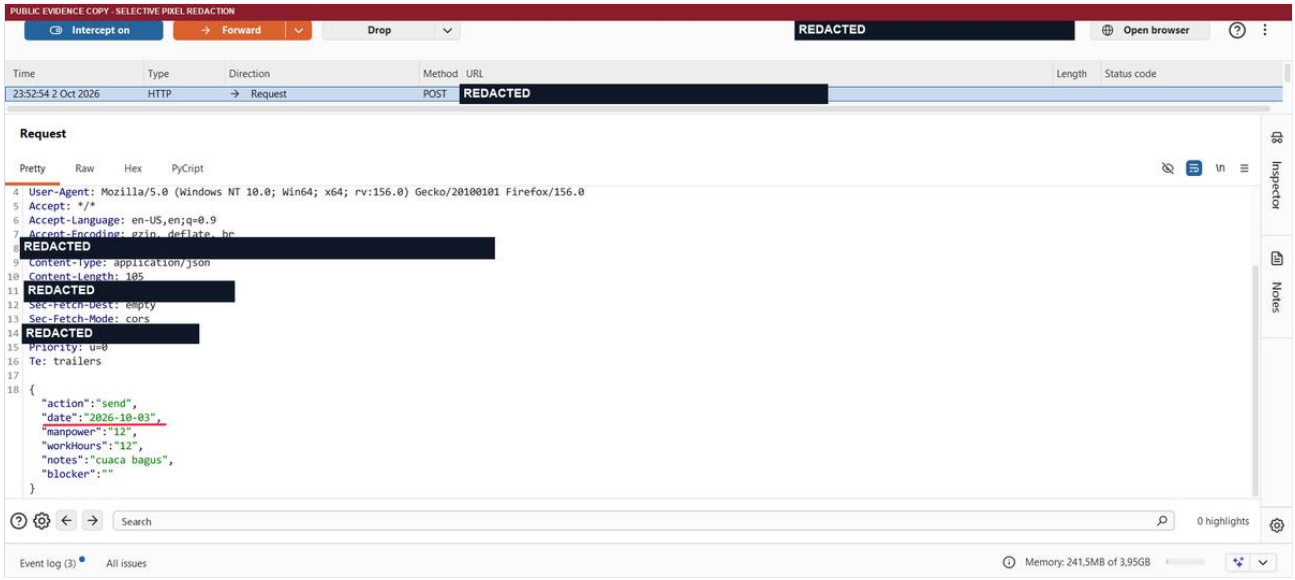


Figure 108 - Arbitrary Historical Date Accepted for Submitted Daily Reports - scenario 1 step 2

3. Replace the active work date with 1900-10-03 and forward the request.

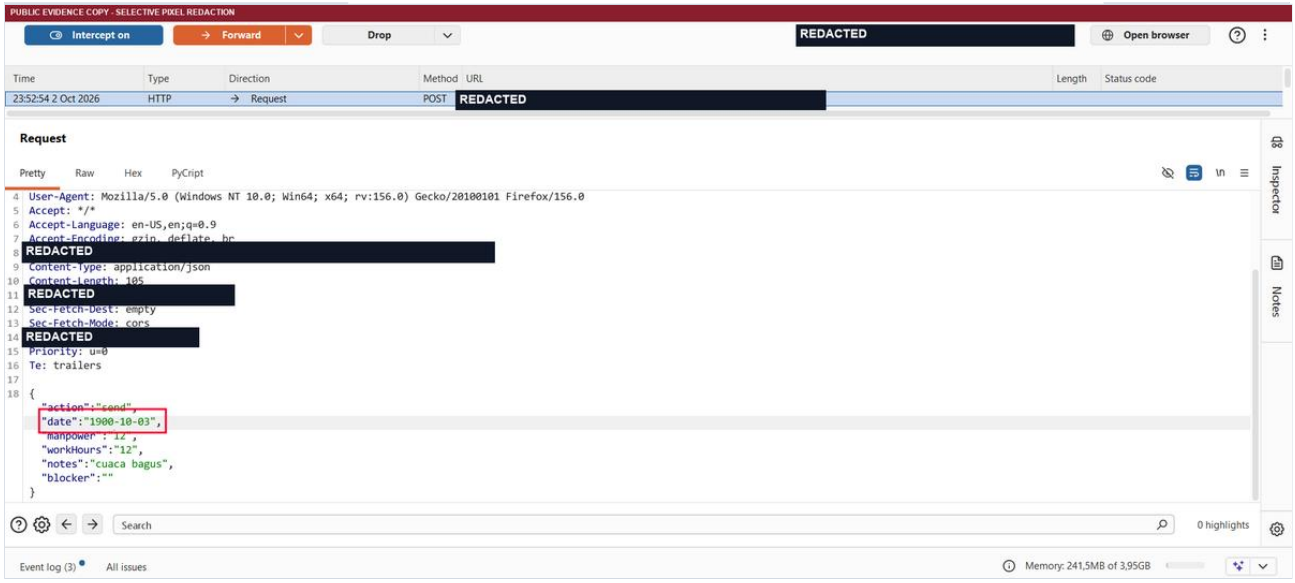


Figure 109 - Arbitrary Historical Date Accepted for Submitted Daily Reports - scenario 1 step 3

4. Confirm that the application stores and displays the submitted report dated 3 October 1900.

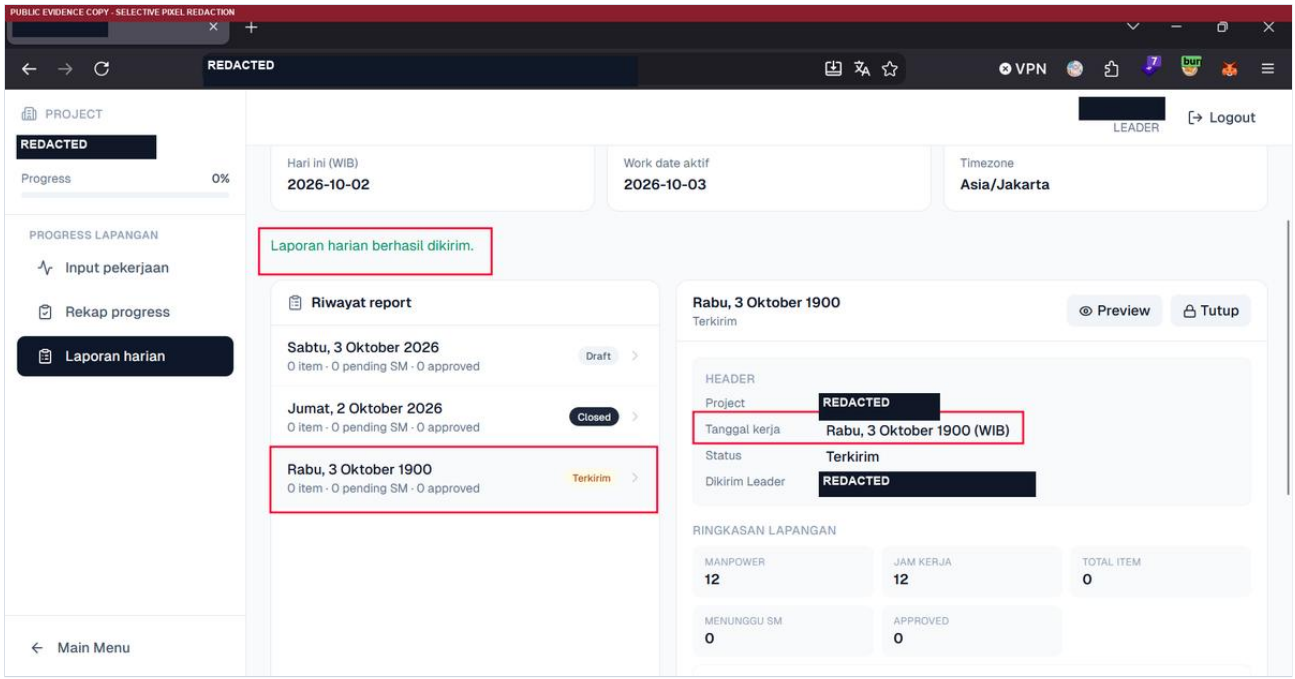


Figure 110 - Arbitrary Historical Date Accepted for Submitted Daily Reports - scenario 1 step 4

3.1.4 Unrestricted Access to Sensitive Business Flows (API6) – F-05 - Future-Dated Delivery Accepted as Completed Progress

SEVERITY	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS
Medium	4.3 / 10	Yes	CWE-841

Standard mapping	**API6:2023** Unrestricted Access to Sensitive Business Flows (OWASP API Security Top 10 2023)
Impact rating	Medium
CVSS vector	`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N`
Affected feature / URL	Delivery submission and cumulative completion state
Testing approach	Grey-box

DESCRIPTION

The delivery flow accepted a future client-controlled receipt date and immediately counted the item as received instead of rejecting the invalid date. A delivery dated far in the future was stored and the affected line and project progress immediately displayed 100 percent completion.

Attack prerequisites: An authenticated account with delivery submission permission was used; no victim interaction was required.

Expected: Receipt dates later than the server date are rejected, and completion is calculated only from valid, approved receipt records.

Actual: A delivery dated far in the future was stored and the affected line and project progress immediately displayed 100 percent completion.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

An account with delivery permission recorded goods as received on a future date and prematurely marked procurement progress complete. The application immediately displayed 100 percent completion, compromising delivery and progress integrity.

RECOMMENDATION

1. Reject delivery dates after the authoritative server date.
2. Enforce project and purchase-order lifecycle boundaries.
3. Calculate completion only from valid, approved receipt records.
4. Add past/future date and timezone boundary tests.

EVIDENCE

Public-edition evidence: Each recorded step is followed by its corresponding selectively redacted screenshot. Client identity, origins, credentials, opaque identifiers, test-account identifiers, and personal data are removed at the pixel level.

Affected scenario 1 — Progress Completion - Future-Dated Delivery

1. Open Progress Kumulatif and select Delivery for the synthetic purchase-order line.

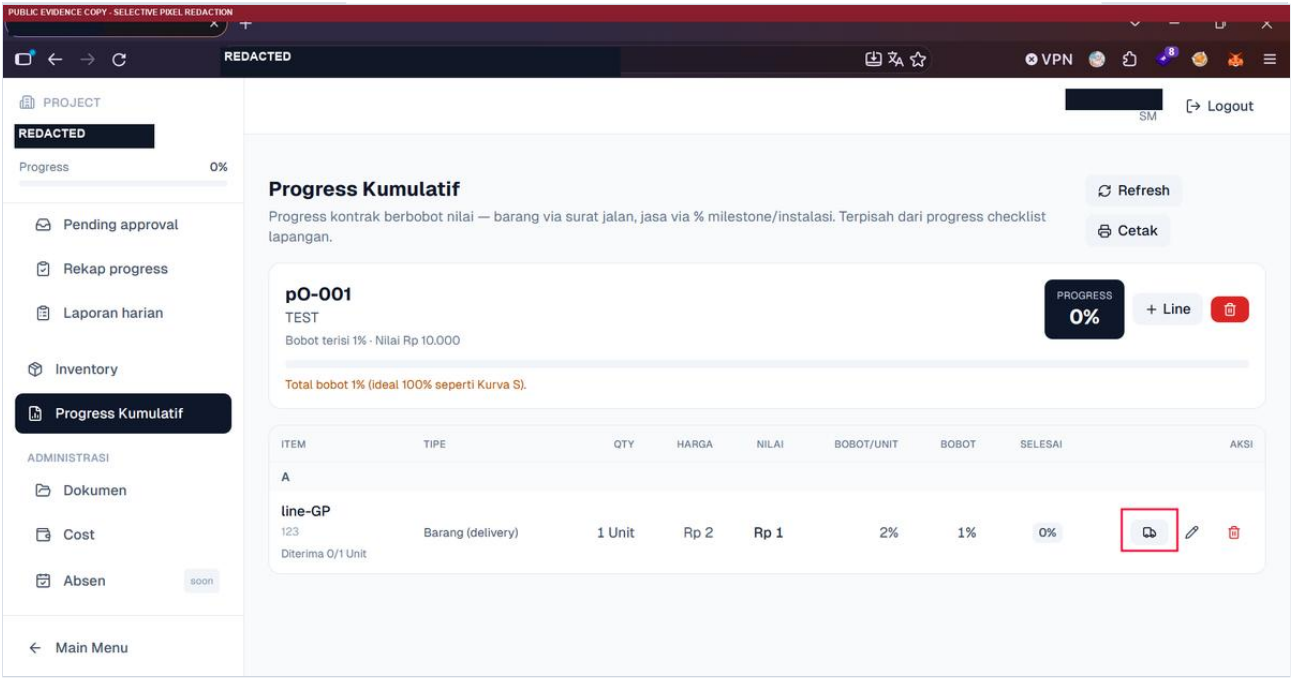


Figure 111 - Future-Dated Delivery Accepted as Completed Progress - scenario 1 step 1

2. Set the receipt date to 30 October 2043 and save the Delivery.

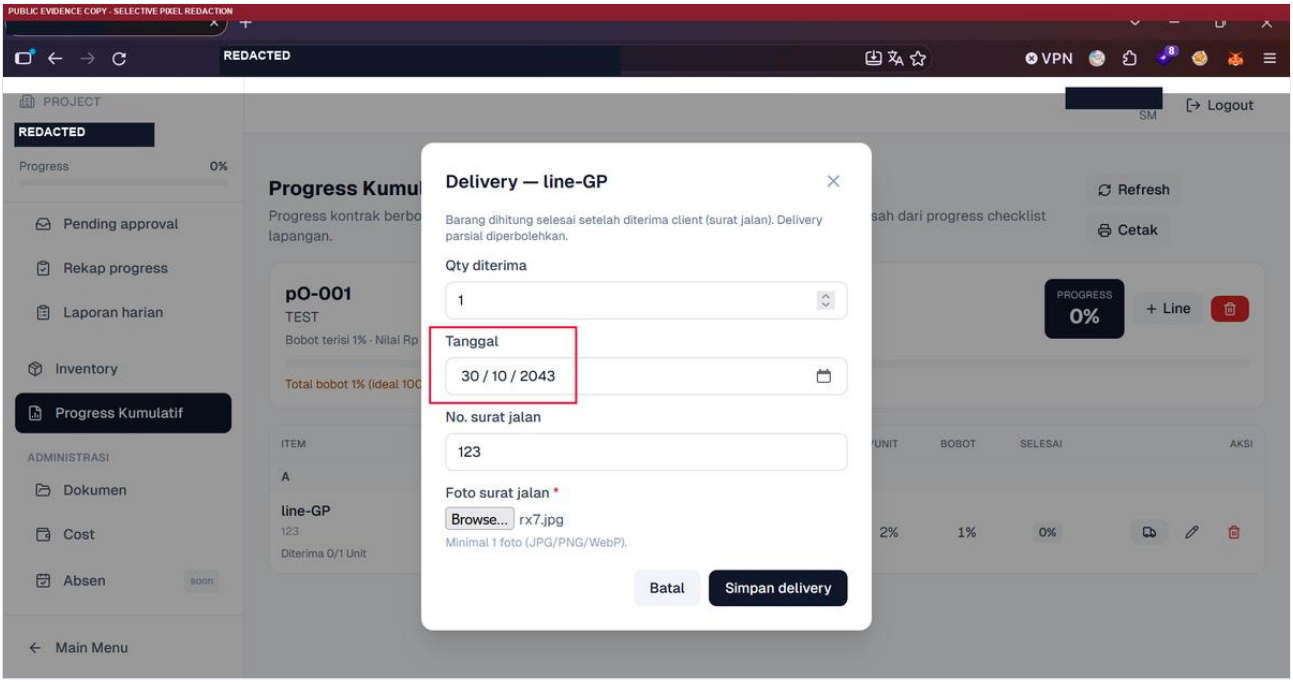


Figure 112 - Future-Dated Delivery Accepted as Completed Progress - scenario 1 step 2

3. Confirm that the application stores the 2043 delivery record and immediately displays 100 percent completion.

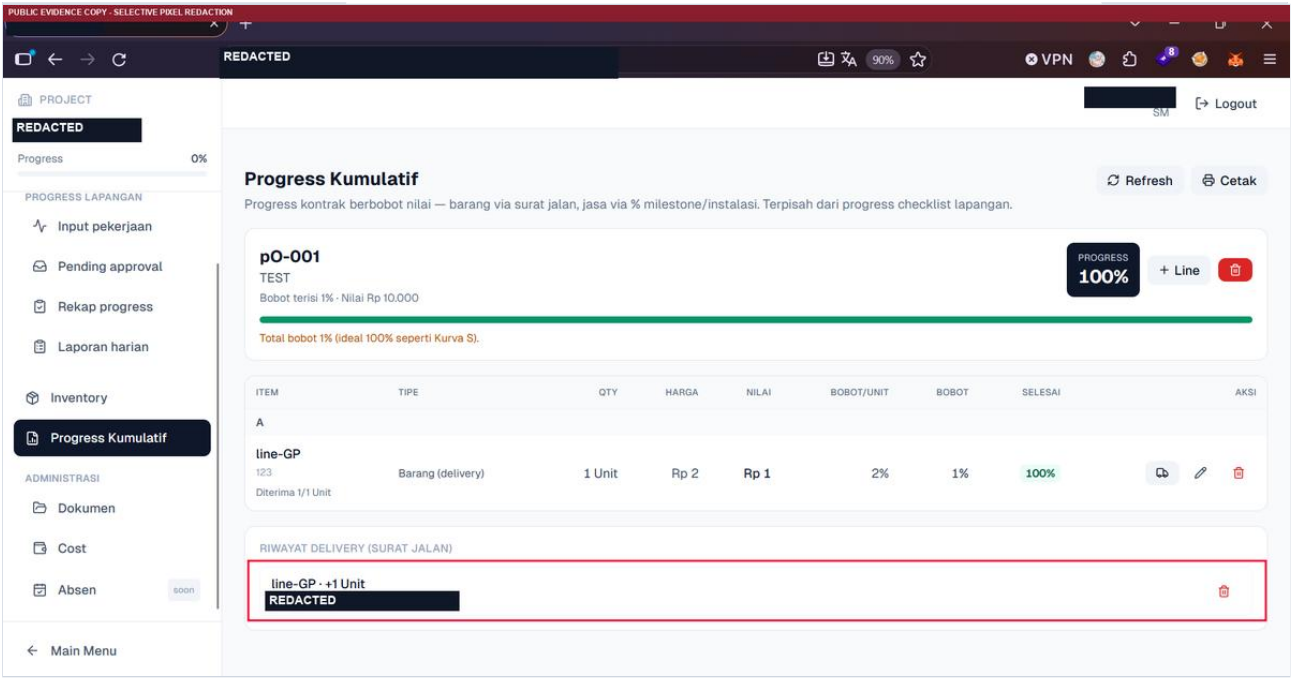


Figure 113 - Future-Dated Delivery Accepted as Completed Progress - scenario 1 step 3

3.1.5 Broken Authentication (API2) – F-03 - Insufficient Session Invalidation After Logout

SEVERITY	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS
Medium	4.2 / 10	Yes	CWE-613

Standard mapping	**API2:2023** Broken Authentication (OWASP API Security Top 10 2023)
Impact rating	Medium
CVSS vector	`CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:N`
Affected feature / URL	• Session termination (logout) and post-logout session acceptance • One organization-structure write function (post-logout replay)
Testing approach	Grey-box

DESCRIPTION

Logout cleared the browser cookie without revoking or invalidating the corresponding server-side session. A saved pre-logout session credential was accepted again after logout: it returned the original identity and protected project data, and a replayed write created an organization entry.

Attack prerequisites: Possession of a copied pre-logout session credential is required. Read access was demonstrated with the viewer test session; the write demonstration used a separate administrative test session. No victim interaction was required. The source vector records high attack complexity (AC:H).

Expected: After logout, the previous session credential is no longer accepted for reading protected data or performing writes.

Actual: A saved pre-logout session credential was accepted again after logout: it returned the original identity and protected project data, and a replayed write created an organization entry.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

Possession of a copied session credential allowed continued access after logout. The recorded viewer session read a protected project object; a separate administrative session created an organization entry. This does not demonstrate gaining administrative rights from a viewer session.

RECOMMENDATION

1. Revoke or invalidate the server-side session when logout succeeds.
2. Check revocation or session version on every authenticated request.
3. Rotate tokens after authentication-state changes.
4. Add post-logout read and write replay regression tests.

EVIDENCE

Public-edition evidence: Each recorded step is followed by its corresponding selectively redacted screenshot. Client identity, origins, credentials, opaque identifiers, test-account identifiers, and personal data are removed at the pixel level.

Affected scenario 1 — Broken Authentication - Session Management - Logout Does Not Invalidate Session

1. Establish the session. The screenshot shows the red-boxed GET /api/auth/session request and the red-boxed [TEST ACCOUNT] identity in the HTTP 200 response, proving the pre-logout authenticated state.

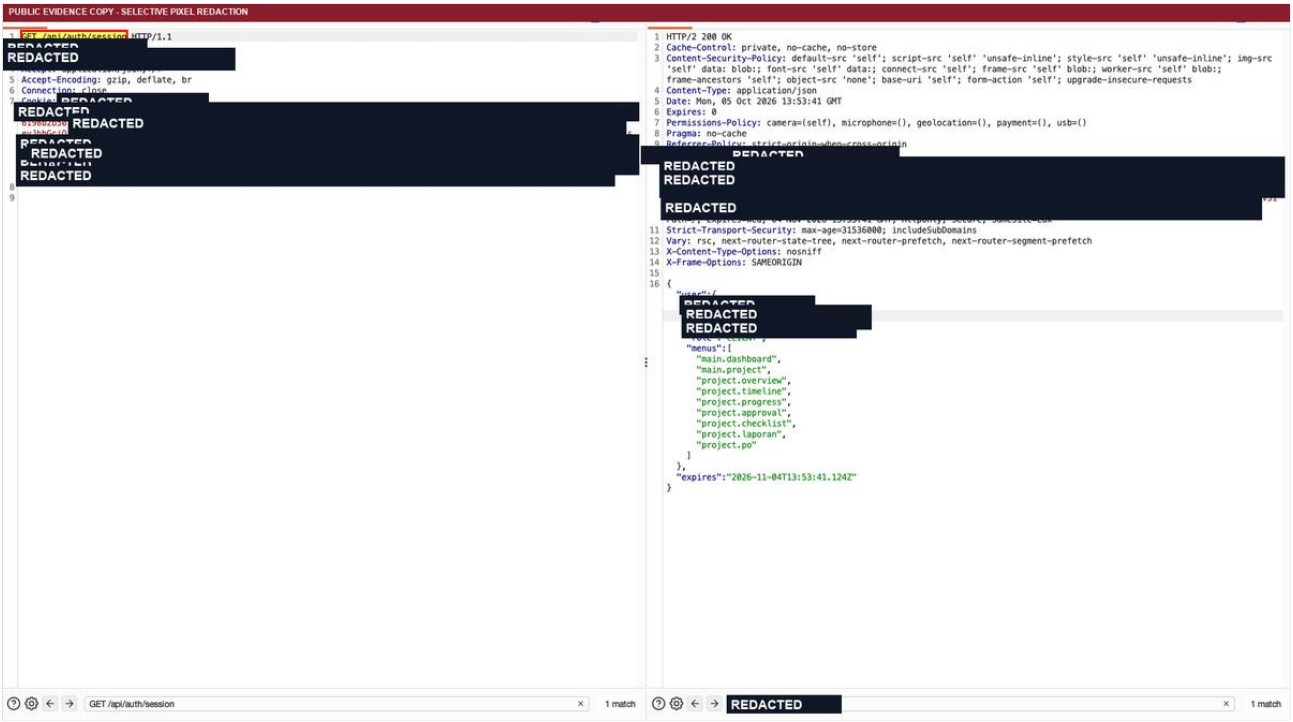


Figure 114 - Insufficient Session Invalidation After Logout - scenario 1 step 1

2. Sign out. The screenshot shows the red-boxed POST /api/auth/signout request and the red-boxed HTTP 200 OK response, which clears the browser cookie via Max-Age=0 while leaving the server-side token valid.

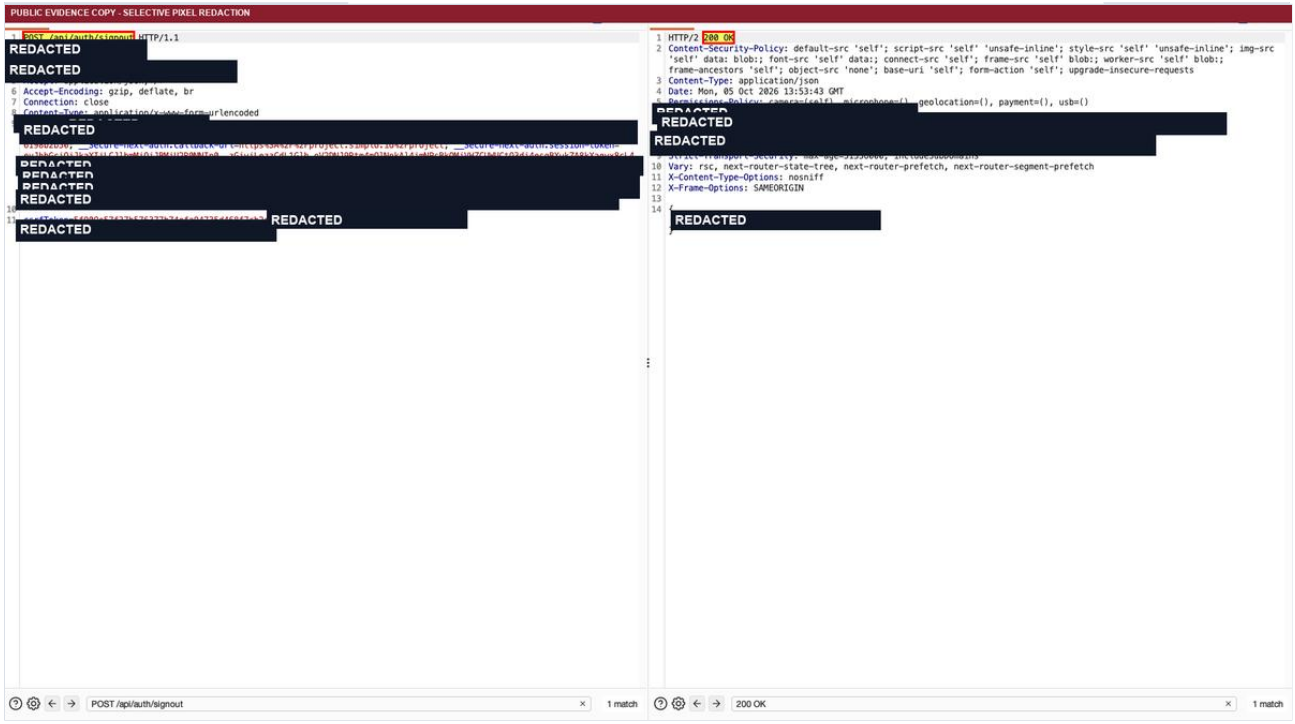


Figure 115 - Insufficient Session Invalidation After Logout - scenario 1 step 2

3. Replay the saved pre-logout Cookie after logout. The screenshot shows the red-boxed session request with the restored pre-logout Cookie and the red-boxed [TEST ACCOUNT] identity returned with HTTP 200, proving the logged-out session is still accepted.

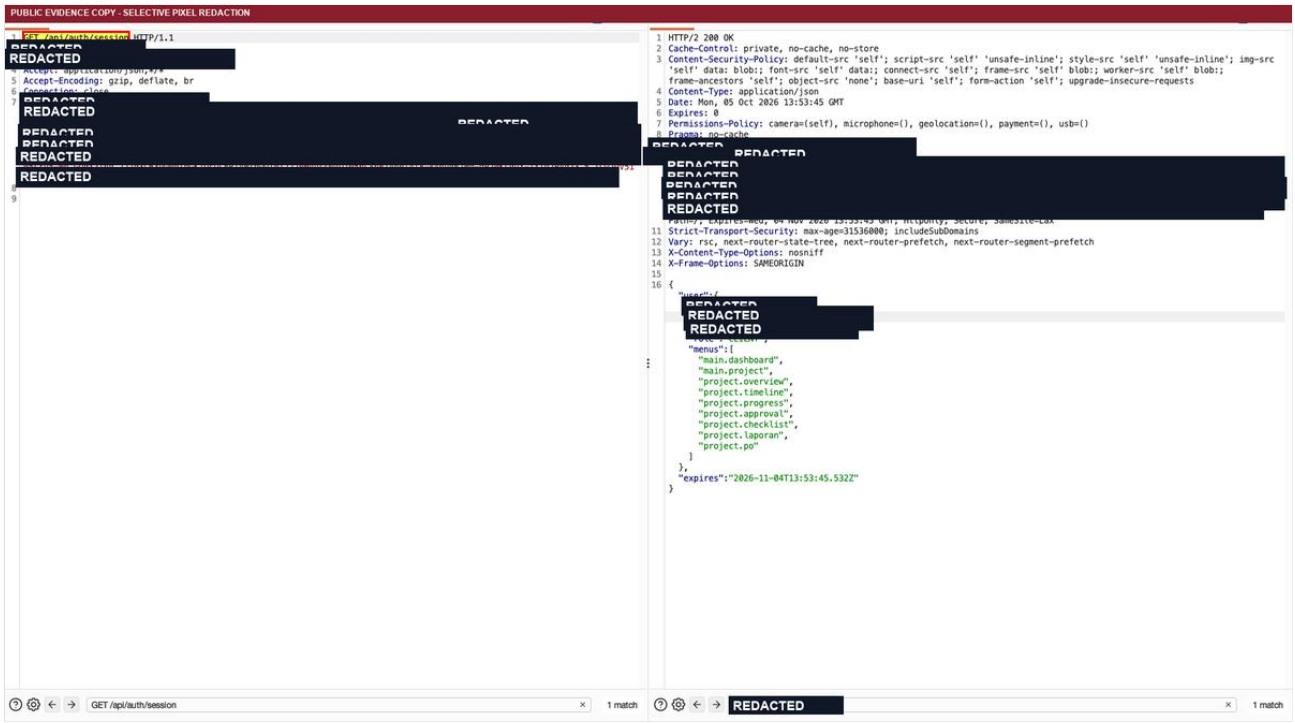


Figure 116 - Insufficient Session Invalidation After Logout - scenario 1 step 3

4. Access a protected object with the replayed cookie. The screenshot shows the red-boxed project request and the red-boxed [TEST FIXTURE] project name in the HTTP 200 response body, proving full authenticated access persists after logout.

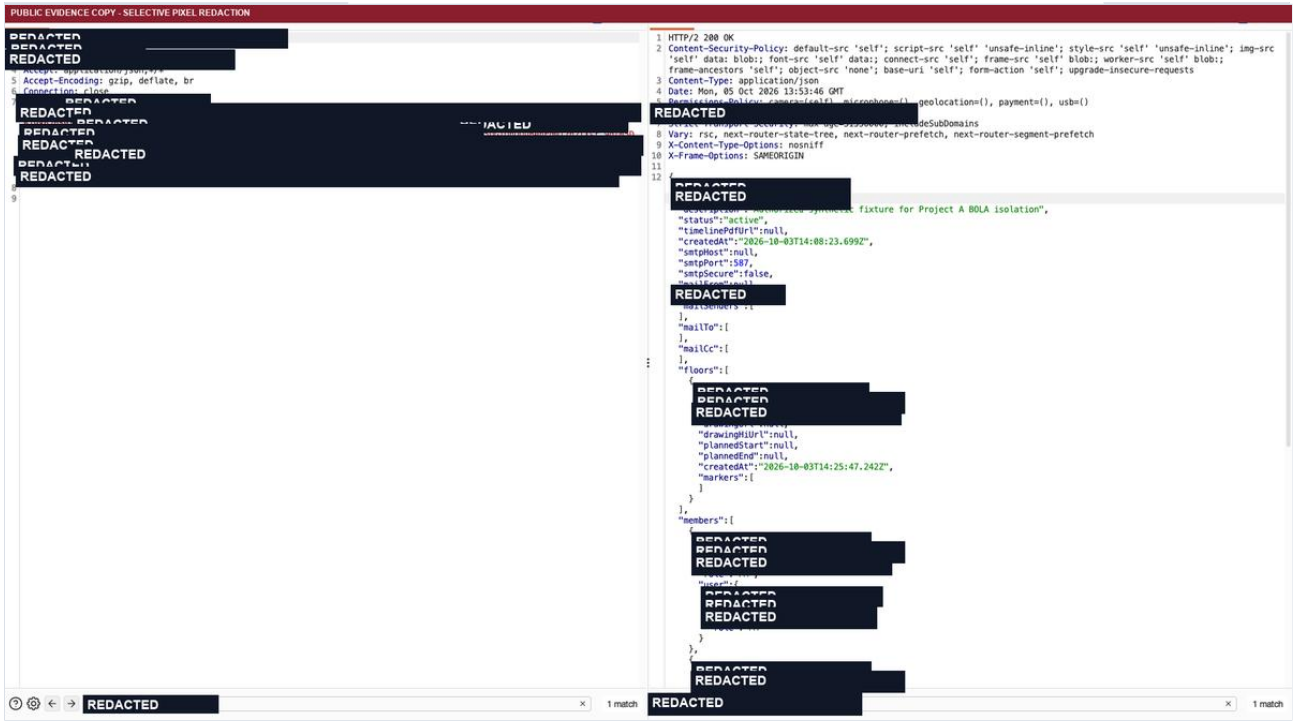


Figure 117 - Insufficient Session Invalidation After Logout - scenario 1 step 4

Affected scenario 2 — Session Invalidation - Organization Position Write Replay

1. Log out from the administrative test session used for the write-replay check.

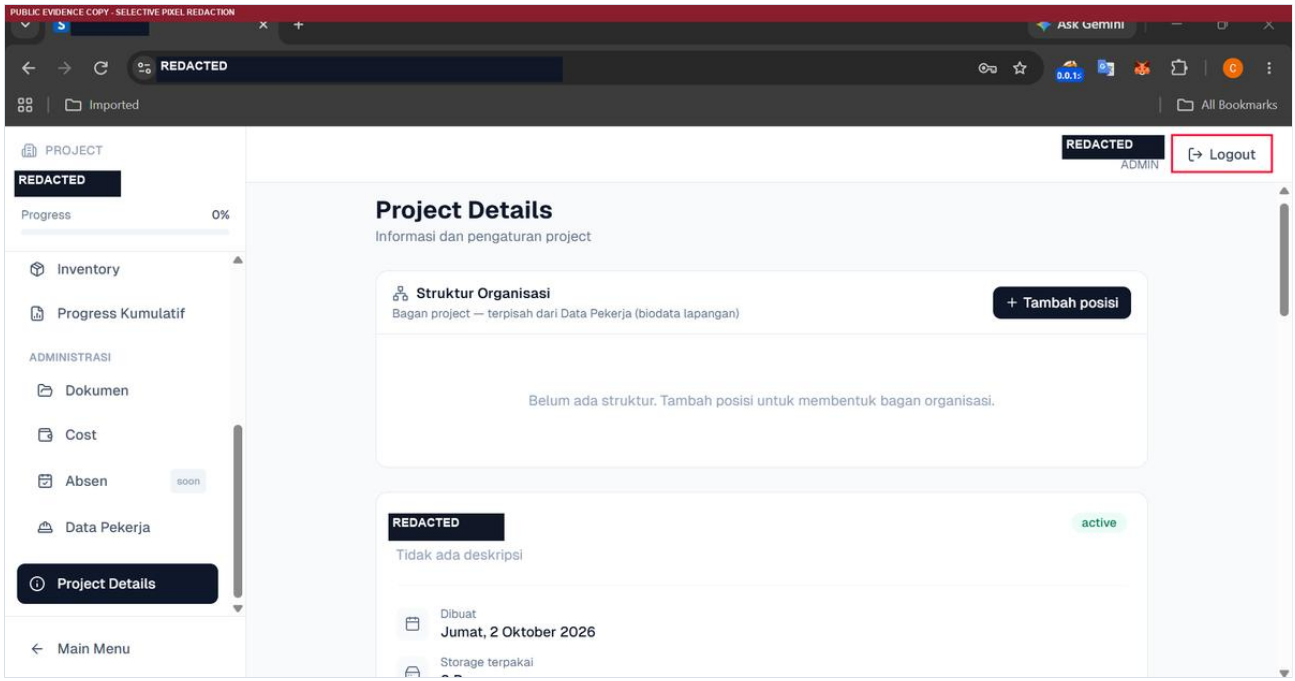


Figure 118 - Insufficient Session Invalidation After Logout - scenario 2 step 1

2. Confirm that the browser has returned to the login page.

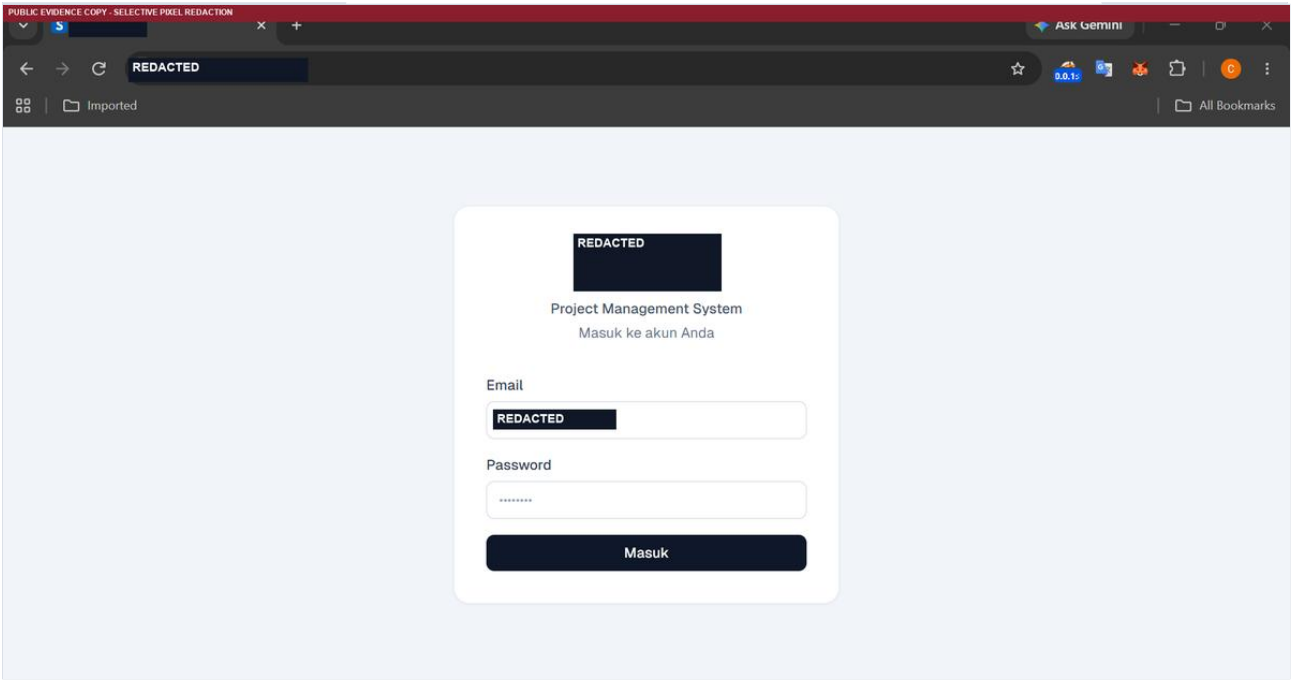


Figure 119 - Insufficient Session Invalidation After Logout - scenario 2 step 2

3. Replay the saved pre-logout organization-position request; the server returns HTTP 200 and creates the position.

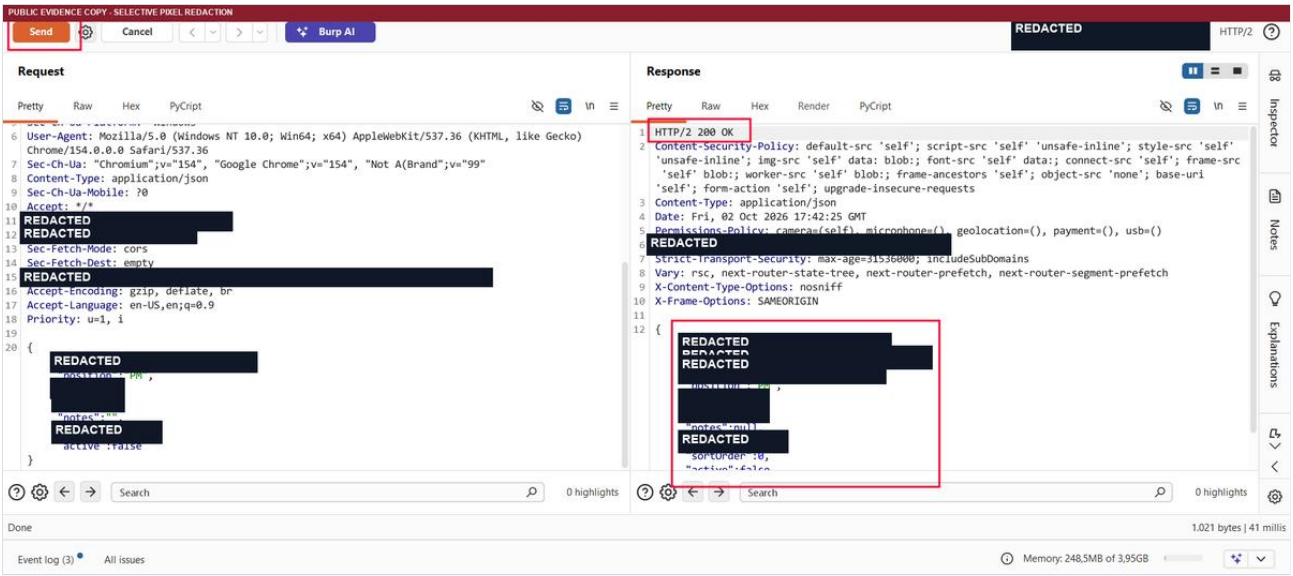


Figure 120 - Insufficient Session Invalidation After Logout - scenario 2 step 3

4. Reload the project organization view and confirm that the replay-created position is present.

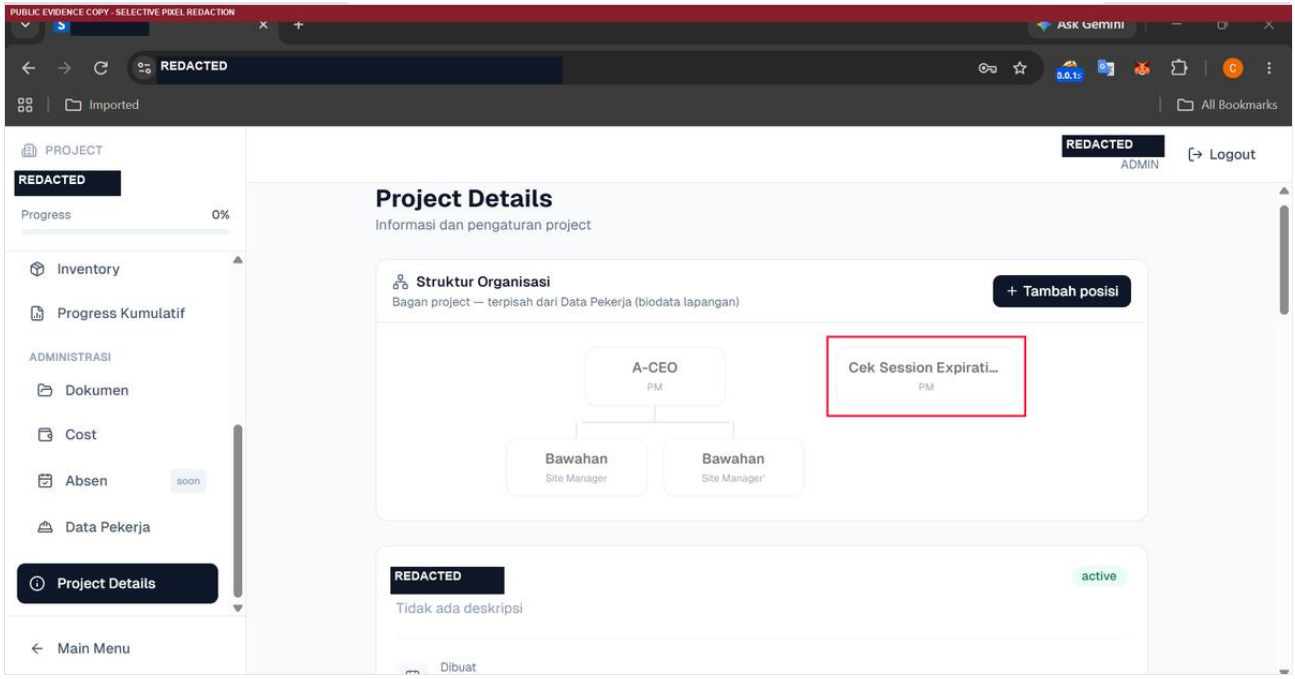


Figure 121 - Insufficient Session Invalidation After Logout - scenario 2 step 4

3.1.6 Security Misconfiguration (API8) – F-06 - Improper Input Validation in Worker Profile Fields

SEVERITY	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS
Low	2.7 / 10	Yes	CWE-20

Standard mapping	**API8:2023** Security Misconfiguration (OWASP API Security Top 10 2023)
Impact rating	Low
CVSS vector	`CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:N/I:L/A:N`
Affected feature / URL	Personnel profile creation and editing (identity and enumerated fields)
Testing approach	Grey-box

DESCRIPTION

Personnel create/edit handling did not enforce server-side format rules for identity fields or the enumerated value set offered by the interface. Malformed phone / national identity number values and an unsupported enumerated value were stored. The stored values remained inert: no script execution, query manipulation, authorization bypass, privilege change, or cross-user effect was demonstrated.

Attack prerequisites: The demonstrated submissions used an authenticated account permitted to create or edit personnel profiles. The source vector records high privileges (PR:H); no victim interaction was required.

Expected: The server rejects malformed identity values and any enumerated value outside the offered set.

Actual: Malformed phone / national identity number values and an unsupported enumerated value were stored. The stored values remained inert: no script execution, query manipulation, authorization bypass, privilege change, or cross-user effect was demonstrated.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

Malformed personnel identity values and an unsupported blood-type value persisted, degrading personnel record integrity. No script execution, query manipulation, authorization bypass, privilege change or cross-user effect was demonstrated. Effects on exports or downstream integrations were not demonstrated.

RECOMMENDATION

1. Enforce server-side format and length rules for phone and national identity number fields.
2. Use the same blood-type enumeration in the API and interface.
3. Reject unknown values with a generic HTTP 400 response.
4. Validate exports and downstream integrations before assigning any higher security impact.

EVIDENCE

Public-edition evidence: Each recorded step is followed by its corresponding selectively redacted screenshot. Client identity, origins, credentials, opaque identifiers, test-account identifiers, and personal data are removed at the pixel level.

Affected scenario 1 — Worker Edit - Malformed Phone and NIK Storage

1. Open the Edit action for the synthetic worker record.

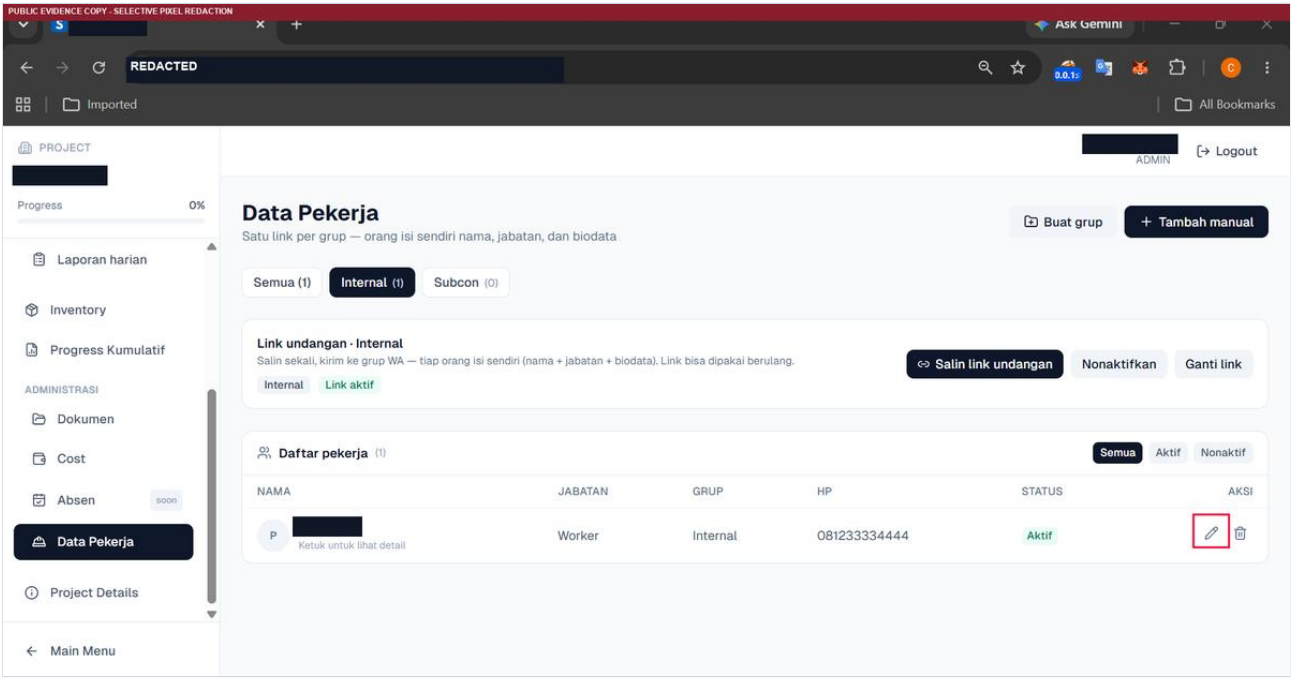


Figure 122 - Improper Input Validation in Worker Profile Fields - scenario 1 step 1

2. Record the normal worker values before changing the Phone and NIK fields.

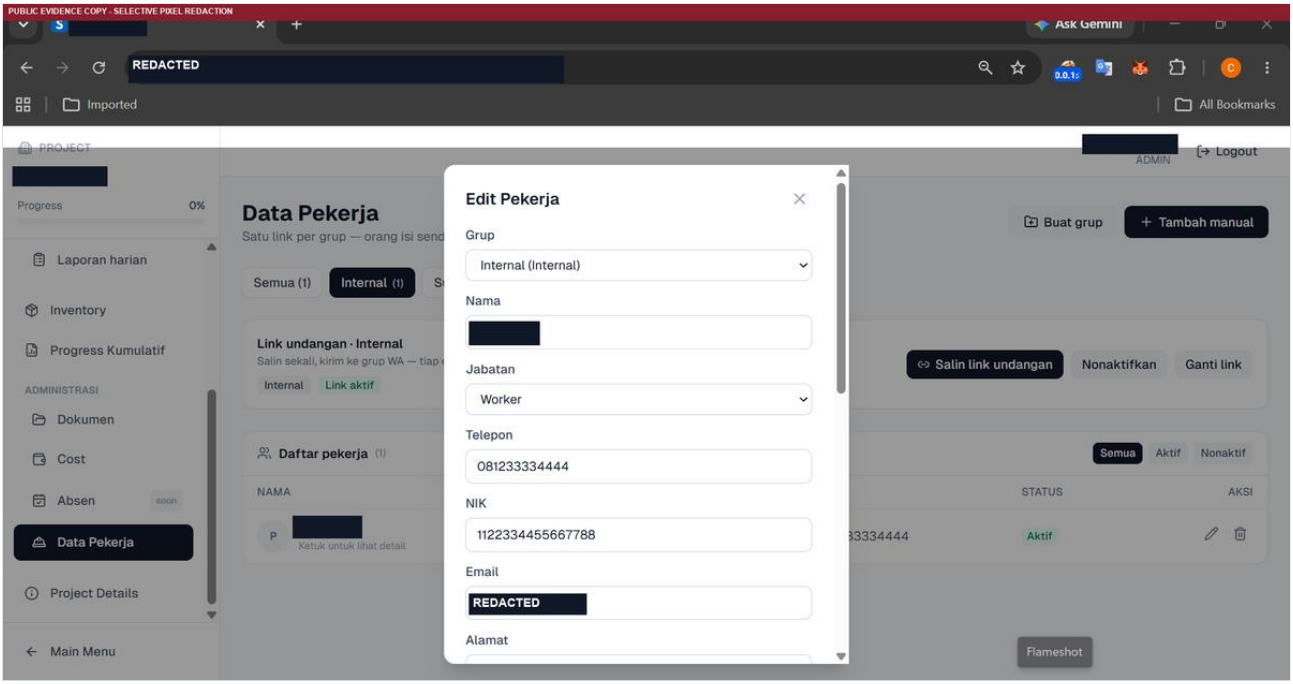


Figure 123 - Improper Input Validation in Worker Profile Fields - scenario 1 step 2

3. Insert nonconforming character canaries into Phone and NIK and submit the edit.

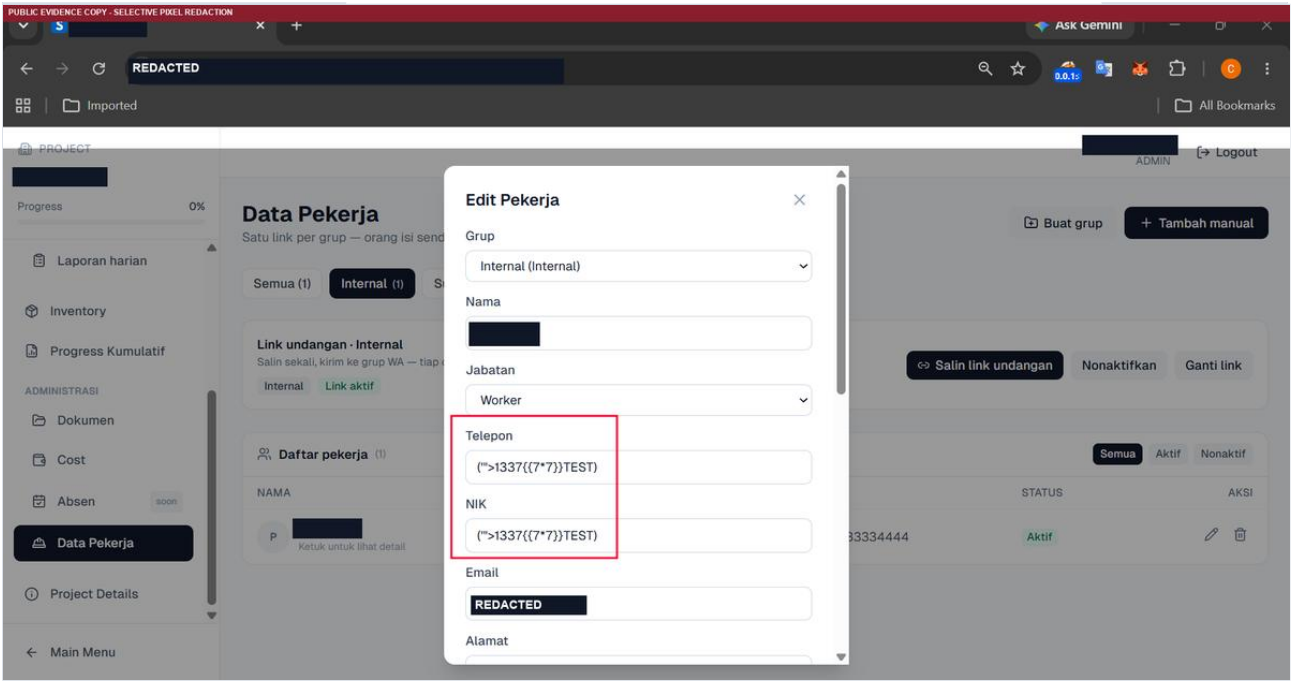


Figure 124 - Improper Input Validation in Worker Profile Fields - scenario 1 step 3

4. Confirm that the server stores and displays the malformed values as inert text; no script execution, query manipulation, privilege change, or cross-user effect was demonstrated.

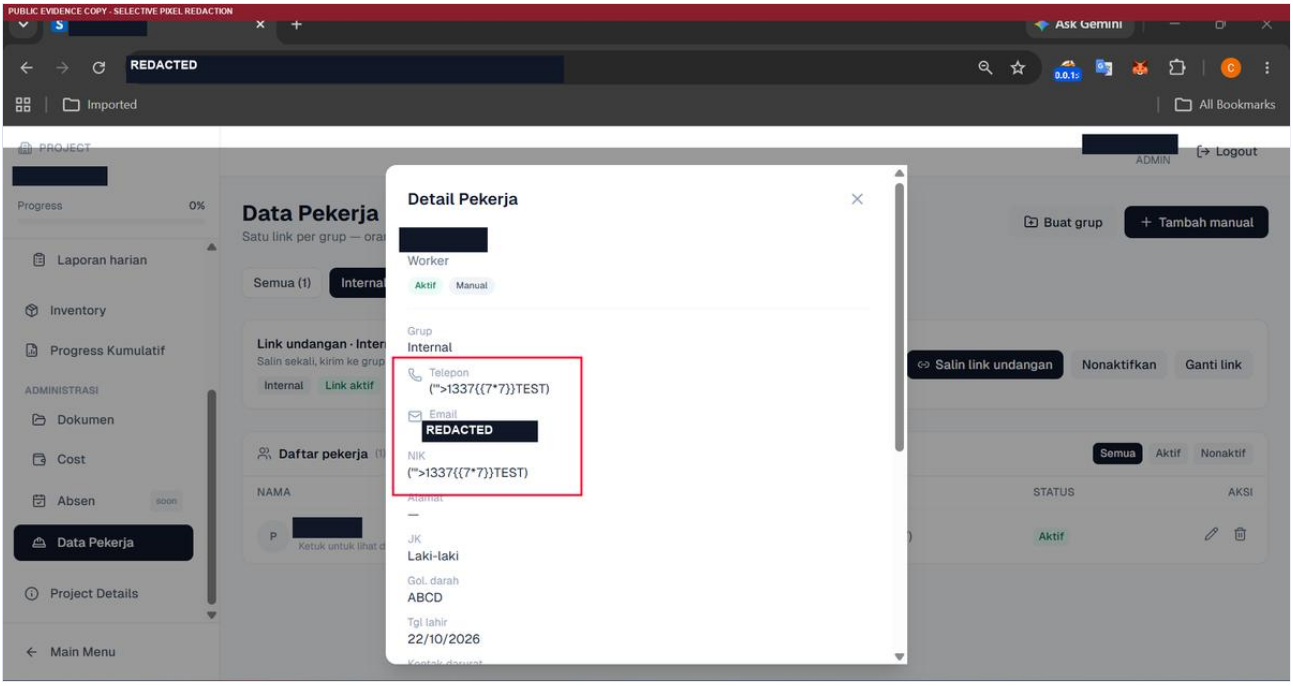


Figure 125 - Improper Input Validation in Worker Profile Fields - scenario 1 step 4

Affected scenario 2 — Manual Worker Creation - Unsupported Blood Type Storage

1. Open the manual worker-creation form.

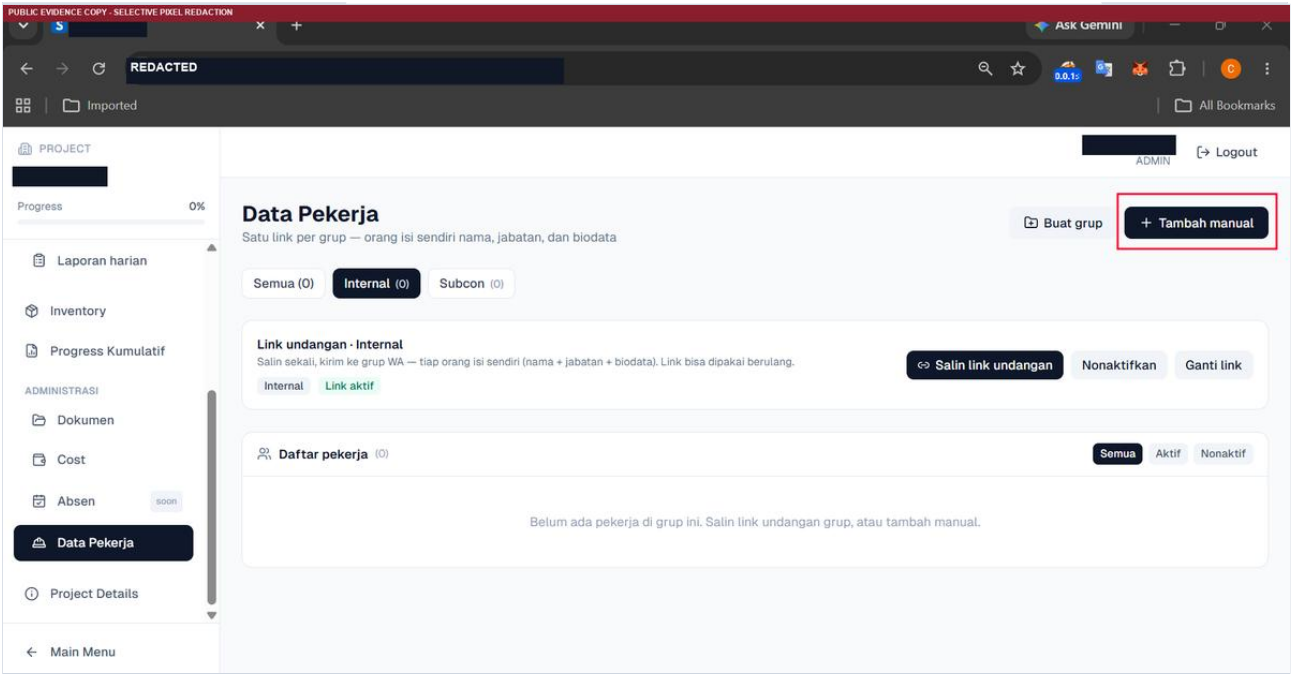


Figure 126 - Improper Input Validation in Worker Profile Fields - scenario 2 step 1

2. Record the blood-type values offered by the user interface.

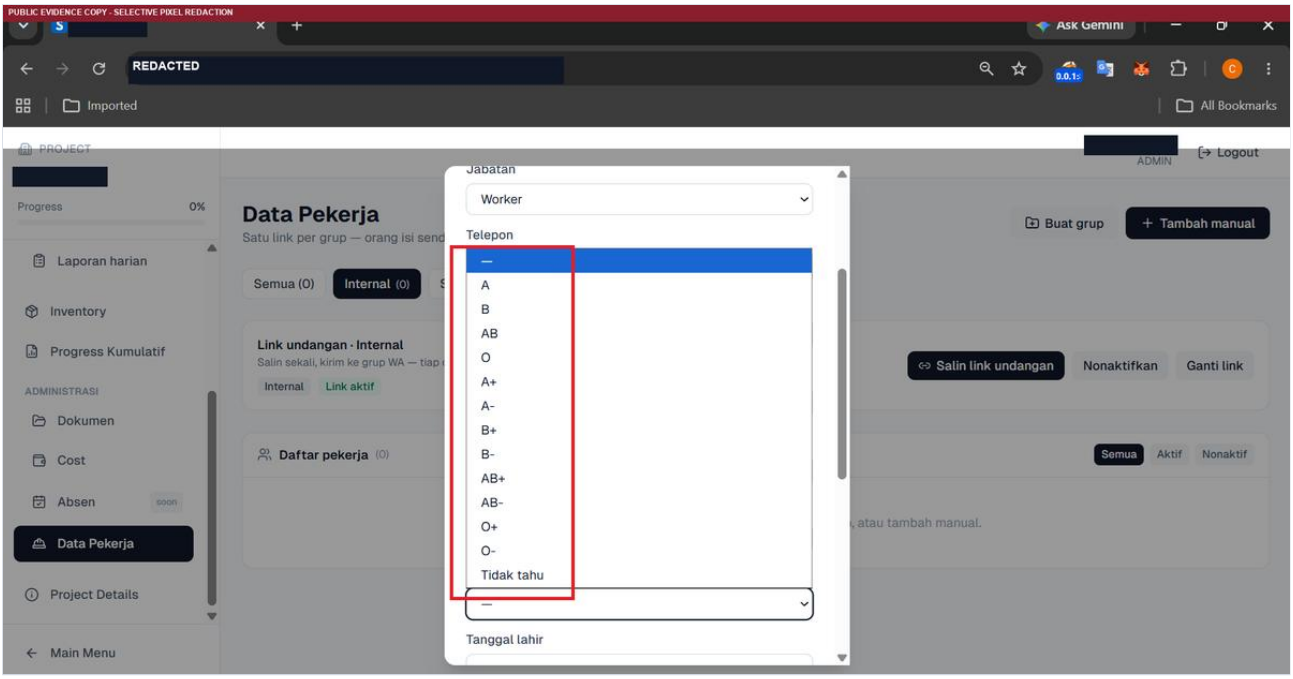


Figure 127 - Improper Input Validation in Worker Profile Fields - scenario 2 step 2

3. Intercept the worker-creation request and identify the bloodType property.

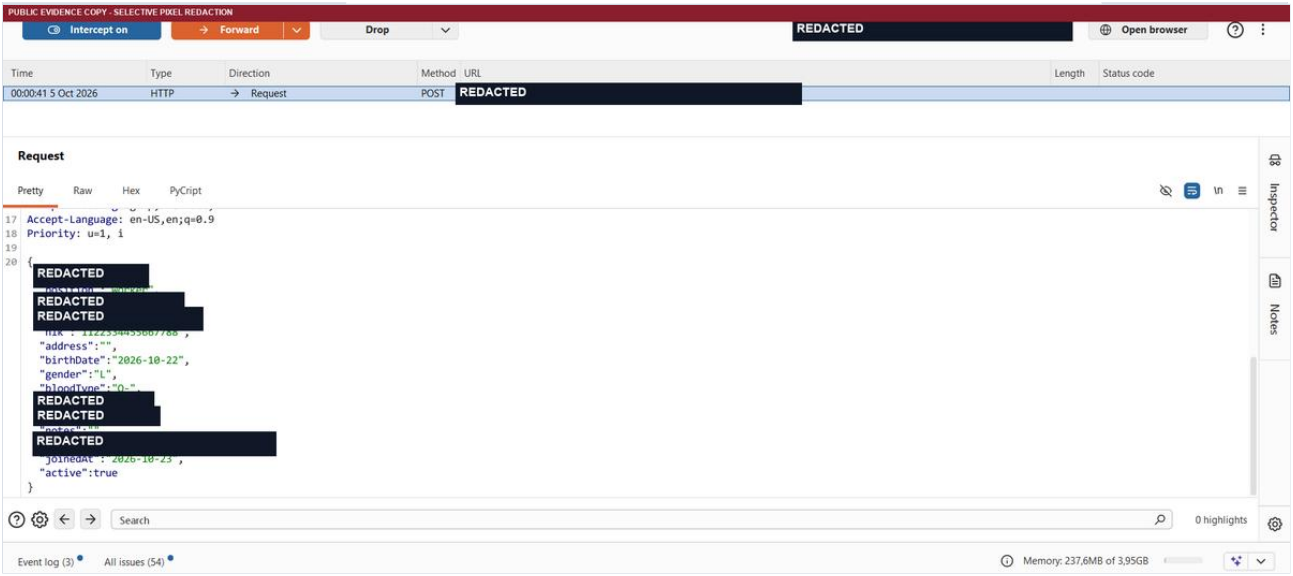


Figure 128 - Improper Input Validation in Worker Profile Fields - scenario 2 step 3

4. Replace bloodType with the unsupported value ABCD and submit the request.

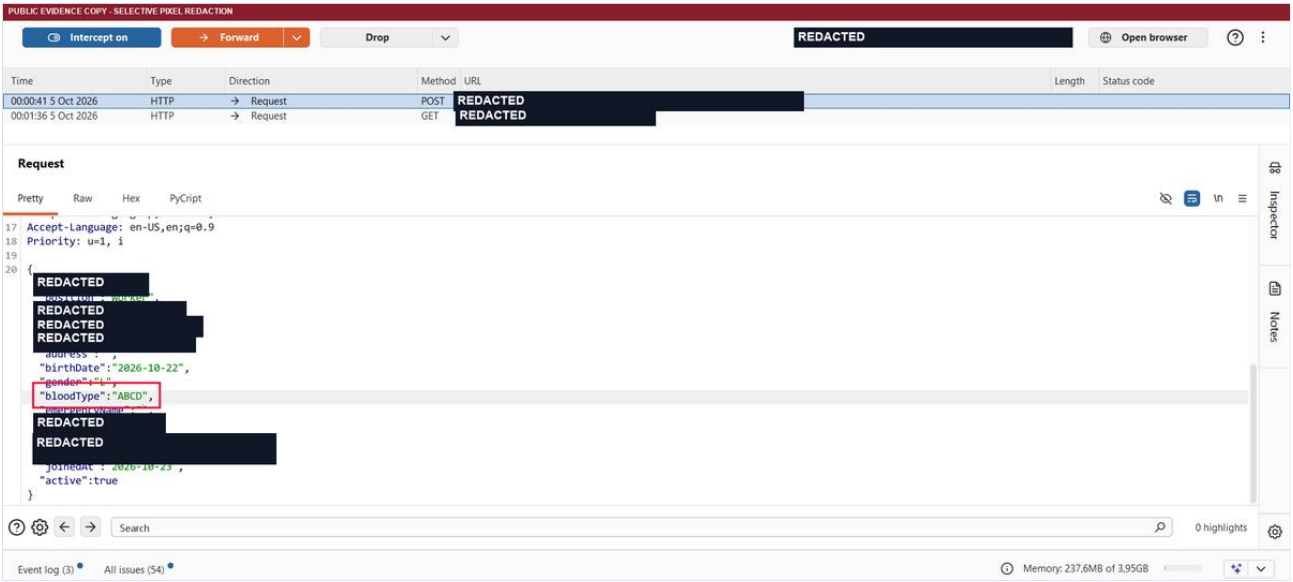


Figure 129 - Improper Input Validation in Worker Profile Fields - scenario 2 step 4

5. Confirm that ABCD is stored in the worker record; no authorization bypass, injection, or security-sensitive downstream decision was demonstrated.

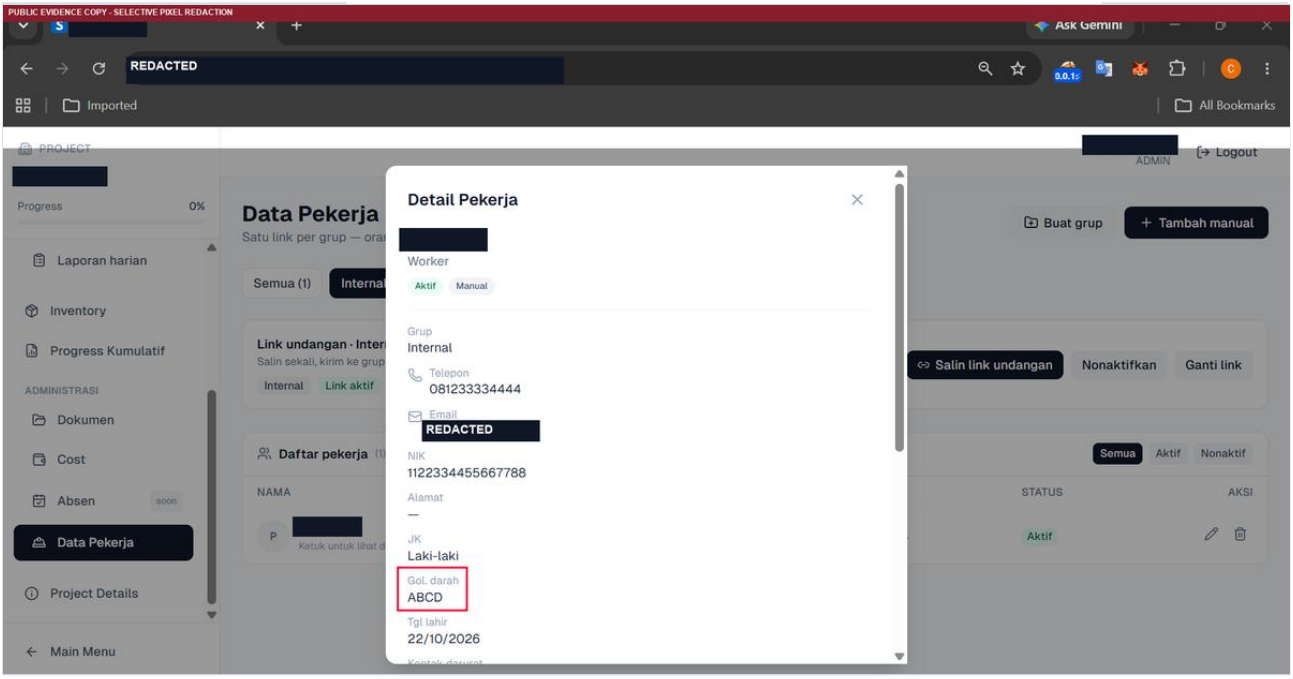


Figure 130 - Improper Input Validation in Worker Profile Fields - scenario 2 step 5

3.1.7 Broken Object Level Authorization (API1) – P-01 - Authorization and Object Access Controls

RESULT	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS TESTED
PASS	—	No	N/A

Standard mapping	**API1:2023** Broken Object Level Authorization (OWASP API Security Top 10 2023) **API5:2023** Broken Function Level Authorization (OWASP API Security Top 10 2023)
Affected feature / URL	—
Testing approach	Grey-box
Result	**PASS** — Retain the tested controls and add regression tests for the exact role, object and action combinations.

DESCRIPTION

Eleven exact role, membership, object and action combinations were denied or filtered. These narrow results do not override the separate object-level and function-level authorization findings.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

No vulnerability was demonstrated in the recorded scenario(s). This result applies only to the exact control boundaries described.

RECOMMENDATION

Retain the tested controls and add regression tests for the exact role, object and action combinations.

EVIDENCE

Public-edition evidence: Sanitized narrative only. Raw traffic, screenshots, hosts, routes, identifiers, credentials, and exact test values are intentionally omitted.

Scenario 1 - Viewer budget access

- The viewer's isolated project and Role A's financial view established context. The viewer's direct budget request was denied with HTTP 403.

Scenario 2 - Viewer cost access

- The viewer's direct request for a synthetic cost record was denied with HTTP 403; the restricted record was not returned.

Scenario 3 - Viewer document metadata access

- The viewer's isolated project and Role A's document list established context. The viewer's direct document-list request was denied with HTTP 403.

Scenario 4 - Role C document download

- Role C received HTTP 404 with no content for a file reference. Role A received HTTP 200 and image bytes for the identical reference, confirming that the file existed.

Scenario 5 - Viewer inventory access

- Role A's inventory view established the baseline. The viewer's direct inventory request was denied with HTTP 403.

Scenario 6 - Role C inventory write

- Role C's synthetic inventory write was refused with HTTP 403. A subsequent Role A inventory read contained no row with the submitted marker.

Scenario 7 - Viewer project membership

- The viewer interface showed only its assigned test project. Direct access to the other project's details was denied with HTTP 403.

Scenario 8 - Viewer project-list filtering

- The viewer received HTTP 200 with exactly its assigned project in a single-element collection and no second project.

Scenario 9 - Role C member administration

- After establishing the own-project context, Role C's direct member-administration request was denied with HTTP 403.

Scenario 10 - Non-administrative user management

- Roles A, B, C and the viewer each received HTTP 403 from the user-management function.

Scenario 11 - Viewer cross-project daily reports

- The viewer began in its assigned project's reporting feature. A request changed to reference the other project was denied with HTTP 403 and returned no target report data.

3.1.8 Broken Authentication (API2) – P-02 - Authentication and Session Cookie Controls

RESULT	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS TESTED
PASS	—	No	N/A

Standard mapping	**API2:2023** Broken Authentication (OWASP API Security Top 10 2023) **A07:2025** Authentication Failures (OWASP Top 10 2025) **A04:2025** Cryptographic Failures (OWASP Top 10 2025)
Affected feature / URL	—
Testing approach	Grey-box
Result	**PASS** — Retain the tested token, CSRF and cookie controls; separately invalidate sessions at logout.

DESCRIPTION

Fabricated session credentials were rejected, the tested login CSRF token/cookie binding held, and reviewed session-cookie attributes were present. These controls do not remediate post-logout session reuse.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

No vulnerability was demonstrated in the recorded scenario(s). This result applies only to the exact control boundaries described.

RECOMMENDATION

Retain the tested token, CSRF and cookie controls; separately invalidate sessions at logout.

EVIDENCE

Public-edition evidence: Sanitized narrative only. Raw traffic, screenshots, hosts, routes, identifiers, credentials, and exact test values are intentionally omitted.

Scenario 1 - Fabricated session rejection

- A valid Role A session established the authenticated baseline. Anonymous protected-project access returned HTTP 401.
- A fabricated unsigned administrator-claim token established no identity: session inspection returned an empty object with HTTP 200, and protected-project access returned HTTP 401. A malformed session credential also returned HTTP 401.

Scenario 2 - Login CSRF token/cookie binding

- The authentication service issued a fresh CSRF token and bound cookie with HTTP 200. A matching pair with deliberately invalid credentials reached credential validation and returned HTTP 401 with a credential-sign-in error.
- Omitting the submitted token, supplying a mismatched token, or omitting the bound cookie each produced a CSRF rejection. No token, credential, cookie value or request body is reproduced here.

Scenario 3 - Session cookie hardening

- Successful Role A and viewer authentication responses carried Secure, HttpOnly and SameSite=Lax attributes with a scoped path on the session cookie.
- These attributes do not establish server-side revocation after logout; that failure is reported separately.

3.1.9 Broken Object Property Level Authorization (API3) – P-03 - Object Property and Project Binding Controls

RESULT	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS TESTED
PASS	—	No	N/A

Standard mapping	**API3:2023** Broken Object Property Level Authorization (OWASP API Security Top 10 2023)
Affected feature / URL	—
Testing approach	Grey-box
Result	**PASS** — Retain tested property allowlists, role checks, candidate isolation and parent-project binding.

DESCRIPTION

Recorded responses omitted mail credentials from empty configurations, restricted dedicated mail-settings/member functions, and preserved project binding for the selected inventory write. Populated mail-property exposure remains inconclusive.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

No vulnerability was demonstrated in the recorded scenario(s). This result applies only to the exact control boundaries described.

RECOMMENDATION

Retain tested property allowlists, role checks, candidate isolation and parent-project binding.

EVIDENCE

Public-edition evidence: Sanitized narrative only. Raw traffic, screenshots, hosts, routes, identifiers, credentials, and exact test values are intentionally omitted.

Scenario 1 - Mail-configuration credential omission

- The interface indicated that mail credentials were not configured. Role A's HTTP 200 project-detail response had an empty mail host and no credential values.
- This does not establish whether populated sensitive values would be exposed; that question remained inconclusive.

Scenario 2 - Dedicated mail-settings authorization

- Role A received an empty/default configuration with HTTP 200 and a credential-configured flag, not a populated credential.
- Roles B, C and the viewer were denied with HTTP 403; the viewer response also indicated view-only access.

Scenario 3 - Candidate-directory isolation

- Role A received a five-entry candidate directory with HTTP 200. Roles B, C and the viewer received HTTP 403 for the dedicated members function.

Scenario 4 - Parent-project binding

- Role A submitted a synthetic inventory item to Project A while supplying a conflicting Project B identifier in the body. The HTTP 201 response bound it to Project A.
- Read-back showed the new item only in Project A. Project B retained only its pre-existing fixture, with no submitted marker.

3.1.10 Unrestricted Access to Sensitive Business Flows (API6) – P-04 - Business Workflow Controls

RESULT	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS TESTED
PASS	—	No	N/A

Standard mapping	**API6:2023** Unrestricted Access to Sensitive Business Flows (OWASP API Security Top 10 2023)
Affected feature / URL	—
Testing approach	Grey-box
Result	**PASS** — Retain server-side state-transition and quantity invariants, with regression tests.

DESCRIPTION

The selected inventory workflows normalized negative quantities to zero and rejected deletion of a cancelled incoming-goods record through reference substitution.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

No vulnerability was demonstrated in the recorded scenario(s). This result applies only to the exact control boundaries described.

RECOMMENDATION

Retain server-side state-transition and quantity invariants, with regression tests.

EVIDENCE

Public-edition evidence: Sanitized narrative only. Raw traffic, screenshots, hosts, routes, identifiers, credentials, and exact test values are intentionally omitted.

Scenario 1 - Negative-quantity normalization

- A Role A item-creation request with a negative quantity was accepted with HTTP 201.
- Inventory read-back showed zero in both stored rows, including the uniquely marked test row. No negative quantity persisted.

Scenario 2 - Cancelled-record deletion

- The interface offered deletion for a draft incoming-goods record but not a cancelled one.
- Substituting the cancelled record's reference into the draft deletion request was rejected; the application stated that only draft records can be deleted.

3.1.11 Injection (A05) – P-05 - Injection and Output-Encoding Controls

RESULT	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS TESTED
PASS	—	No	N/A

Standard mapping	**A05:2025** Injection (OWASP Top 10 2025) **API8:2023** Security Misconfiguration (OWASP API Security Top 10 2023)
Affected feature / URL	—
Testing approach	Grey-box
Result	**PASS** — Retain strict parsing and context-aware output encoding; extend coverage to other input/rendering contexts.

DESCRIPTION

Five bounded input-handling scenarios rejected malformed values or kept them inert. The recorded browser canaries did not execute. This is not an application-wide injection or output-encoding conclusion.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

No vulnerability was demonstrated in the recorded scenario(s). This result applies only to the exact control boundaries described.

RECOMMENDATION

Retain strict parsing and context-aware output encoding; extend coverage to other input/rendering contexts.

EVIDENCE

Public-edition evidence: Sanitized narrative only. Raw traffic, screenshots, hosts, routes, identifiers, credentials, and exact test values are intentionally omitted.

Scenario 1 - Date syntax validation

- A syntactically valid date reached report handling with HTTP 200 and an active-work-date field. A malformed date returned HTTP 400 with a date-format rejection.
- Boolean-style SQL true and false canaries each returned the same HTTP 400 rejection, with no differential behavior in those requests. This syntax control did not prevent the separate historical-date workflow flaw.

Scenario 2 - Inventory stored-input encoding

- Role A's canary item was accepted with HTTP 201 as literal strings. The interface rendered the canary as plain text without executing markup.
- HTTP 200 read-back preserved the literal name and notes; no evaluation was observed.

Scenario 3 - Malformed project-reference handling

- A nonexistent control reference returned HTTP 404. Boolean-style SQL true and false reference canaries returned the identical not-found status and body, with no observed differential behavior.

Scenario 4 - Public sign-in query rendering

- An anonymous baseline and requests carrying markup, script-scheme callback and sign-in-error canaries displayed the same plain sign-in form.
- The recorded DOM checks found no injected graphics element or script-scheme link; no error banner or script execution was observed in those contexts.

Scenario 5 - Stored floor-name encoding

- A markup canary submitted as a floor name was rendered as text; no script execution occurred in the recorded browser context.

3.1.12 Security Misconfiguration (API8) – P-06 - Active Content Upload Controls

RESULT	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS TESTED
PASS	—	No	N/A

Standard mapping	**API8:2023** Security Misconfiguration (OWASP API Security Top 10 2023) **A08:2025** Software or Data Integrity Failures (OWASP Top 10 2025)
Affected feature / URL	—
Testing approach	Grey-box
Result	**PASS** — Retain server-side extension, MIME, content-signature and storage controls on every upload function.

DESCRIPTION

The selected document, delivery-photo and timeline uploads rejected tested active-content or mismatched-type files. Results are limited to the recorded submissions.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

No vulnerability was demonstrated in the recorded scenario(s). This result applies only to the exact control boundaries described.

RECOMMENDATION

Retain server-side extension, MIME, content-signature and storage controls on every upload function.

EVIDENCE

Public-edition evidence: Sanitized narrative only. Raw traffic, screenshots, hosts, routes, identifiers, credentials, and exact test values are intentionally omitted.

Scenario 1 - Document active-content rejection

- The authorized document upload interface declared document/image type and size restrictions.
- An active HTML document and an image-named file with an HTML content type each returned HTTP 400 for the tested format mismatch. No file reference was returned for the active document.

Scenario 2 - Delivery-photo type rejection

- The recorded test selected a disallowed file type, then submitted a photo upload without a filename extension and with an HTML content type.
- The application rejected the upload and required a JPEG, PNG or WebP image.

Scenario 3 - Timeline document type rejection

- The recorded timeline upload omitted the filename extension and used an HTML content type.
- The application rejected the upload and required a PDF file.

3.1.13 Security Misconfiguration (API8) – P-07 - Cross-Origin Resource Sharing Controls

RESULT	CVSS V3.1 BASE SCORE	EXPLOITED	WEAKNESS TESTED
PASS	—	No	N/A

Standard mapping	**API8:2023** Security Misconfiguration (OWASP API Security Top 10 2023)
Affected feature / URL	—
Testing approach	Grey-box
Result	**PASS** — Retain explicit trusted-origin handling and credential restrictions on every API response.

DESCRIPTION

The selected anonymous authentication and authenticated project requests did not grant the tested hostile origins credentialed cross-origin read access.

Remediation status: Not stated in source report
Retest status: Not stated in source report

IMPACT

No vulnerability was demonstrated in the recorded scenario(s). This result applies only to the exact control boundaries described.

RECOMMENDATION

Retain explicit trusted-origin handling and credential restrictions on every API response.

EVIDENCE

Public-edition evidence: Sanitized narrative only. Raw traffic, screenshots, hosts, routes, identifiers, credentials, and exact test values are intentionally omitted.

Scenario 1 - Anonymous cross-origin checks

- Anonymous session inspection returned HTTP 200 with an empty session. Protected-project requests with external, null and suffix-lookalike origins each returned HTTP 401 without a credentialed cross-origin read grant.
- The preflight returned HTTP 204 without a grant. Anonymous CSRF-token retrieval returned HTTP 200 without a cross-origin grant.

Scenario 2 - Authenticated cross-origin checks

- Authenticated external-origin and null-origin reads each returned HTTP 200. Their recorded full response headers contained neither an allowed-origin nor an allowed-credentials grant.
- The authenticated preflight returned HTTP 204 without a cross-origin grant.

4 Summary of Recommendations

Confirmed vulnerabilities: **6** (5 Medium, 1 Low). Prioritize project/object isolation and role-to-function enforcement as separate controls, then server-side logout revocation, semantic date validation, and personnel field schemas. Passed controls apply only to their recorded scenarios.

Scope and limitations

- Assessment type: grey-box security testing of one web application and its API, using dedicated test accounts and synthetic fixture records.
- Passed controls are narrow results: each applies only to the exact role, object, and function combination recorded in the evidence narrative, and does not clear the related vulnerability findings.
- One mail-property exposure question remained inconclusive in the source because no populated sensitive value was disclosed during testing.
- Not fully tested in the source assessment: resource consumption, supply chain, logging, server-side request forgery, unsafe upstream consumption, broad write workflows, and unidentified routes.
- Screenshots and raw requests/responses are omitted from this public edition; evidence is provided as sanitized narratives of the recorded steps and outcomes.
- No remediation or retest status is stated anywhere in the source report; none is claimed here. Any status update requires a separate retest statement.

Broken Object Level Authorization (API1) – F-01 - Broken Object Level Authorization Across Project-Scoped Resources — **Medium**

1. Resolve the authenticated identity to explicit project membership before loading project resources.
2. Enforce deny-by-default authorization using identity, membership, object ownership and the parent project.
3. Bind parent and child resources, including files, to the same authorized project; filter collections by membership.
4. Add bidirectional cross-project access regression tests for every resource.

Broken Function Level Authorization (API5) – F-02 - Broken Function Level Authorization Across Restricted Features — **Medium**

1. Enforce the approved role-to-function matrix before loading business data.
2. Deny unauthorized direct API calls independently of menu visibility and client-side routing.
3. Centralize policy across list, detail and file functions.
4. Add positive and negative tests for every role/function combination and log denied attempts.

Unrestricted Access to Sensitive Business Flows (API6) – F-04 - Arbitrary Historical Date Accepted for Submitted Daily Reports — **Medium**

1. Derive report dates from the server-side active work date and enforce an allowed reporting window.
2. Reject out-of-window dates; permit duplicate or retroactive corrections only through an explicit audited workflow.
3. Add past, future, timezone and closed-period boundary tests.

Unrestricted Access to Sensitive Business Flows (API6) – F-05 - Future-Dated Delivery Accepted as Completed Progress	Medium

1. Reject delivery dates after the authoritative server date.
2. Enforce project and purchase-order lifecycle boundaries.
3. Calculate completion only from valid, approved receipt records.
4. Add past/future date and timezone boundary tests.

Broken Authentication (API2) – F-03 - Insufficient Session Invalidation After Logout	Medium

1. Revoke or invalidate the server-side session when logout succeeds.
2. Check revocation or session version on every authenticated request.
3. Rotate tokens after authentication-state changes.
4. Add post-logout read and write replay regression tests.

Security Misconfiguration (API8) – F-06 - Improper Input Validation in Worker Profile Fields	Low

1. Enforce server-side format and length rules for phone and national identity number fields.
2. Use the same blood-type enumeration in the API and interface.
3. Reject unknown values with a generic HTTP 400 response.
4. Validate exports and downstream integrations before assigning any higher security impact.

5 Conclusion

OVERALL RISK RATING

Medium

The assessment confirmed object-level and function-level authorization failures, continued session acceptance after logout, historical reporting-date and future delivery-date workflow flaws, and limited personnel input-validation failures. The Low finding demonstrated data-integrity impact only: no code execution, authorization bypass or cross-user effect was proven. Thirty narrow controls passed. Populated mail-property exposure remained inconclusive; several other categories were not fully tested.

No remediation or retest status is asserted in this edition.

Residual disclosure limitations

- Direct identifiers, hosts, routes, credentials and personal data are omitted. The retained combination of functional areas and outcomes can still identify the general class of application to a knowledgeable reader.
- Severity, CVSS vectors, CWE/OWASP categories, and demonstrated outcomes are intentionally retained because they are necessary for remediation and risk communication; they describe vulnerability classes, not the tested instance.
- Only the reviewed source revision may be rebuilt. Pattern scans are defense in depth, not a guarantee against every indirect identifier. Review any changed projection before external release; do not reintroduce private screenshots or traffic.
- Dates and payload values are described by category only. Historical/future date outcomes, control response statuses, and negative-quantity normalization to zero are retained without reusable request details.

6 Attachments

Attachment	Description