

WEB, API AND CLOUD ASSESSMENT · REGULATED DIGITAL-SERVICES PLATFORM

Thousands of organizations' data reachable from one ordinary account

A regulated multi-tenant digital-services platform on public cloud

Anonymized with the client's written permission. The organization, systems, hostnames, addresses, accounts and other identifiers in the original report have been removed or generalized. Findings are described at a level that cannot be traced back to the client.

Client	A regulated digital-services platform (anonymized)
Engagement	Penetration test of a multi-tenant web application, its API, and the supporting public-cloud project
Approach	Grey-box with low- and high-privilege accounts and read-only cloud audit roles, across staging and production, roughly two weeks
Tester	Ismail Regragui, Vallum Security · OSCP, OSEP, BSCP, OSWP · Synack Red Team
Standards	OWASP WSTG, OWASP API Security Top 10, cloud security benchmarks; severity by CVSS v3.1

What we were asked to do

Assess whether the platform's access controls, tenant isolation and cloud configuration were ready for large-scale, regulated deployment: could one tenant reach another's data, and could a low-privilege user reach data or actions meant for administrators?

Results at a glance



Findings at a glance

ID	FINDING	SEVERITY
----	---------	----------

1	Broken object-level authorization (IDOR) allowing enumeration and modification of users across all organizations	Critical
2	Low-privilege users able to invite, list and delete other organizations' users	High
3	Unauthenticated access to business-process data through the API	High
4	Weak multi-tenant isolation allowing cross-tenant access	High
5	Unauthorized read and deletion of other tenants' files via a leaked storage path	High
6	File-upload validation bypass allowing dangerous file names and content	High
7	An outdated document-generation library with a known denial-of-service flaw	High
8	Session tokens not invalidated on logout	Medium
9	Read-only cloud audit roles able to download highly sensitive files from storage	Medium
10	Cloud image registry exposing internal architecture to low-privilege roles	Medium
11	Unauthenticated API schema introspection enabled in production	Medium
12	Unauthenticated disclosure of internal workflow logic through the API	Medium
13	Unauthenticated disclosure of user email addresses through the API	Medium
14	An outdated web framework with a known authentication-bypass flaw	Medium
15	A project-wide cloud SSH key present without strict access enforcement	Low
16	An outdated routing library with a known data-spoofing flaw	Low
17	Missing HTTP Strict Transport Security	Info

Findings are renumbered and retitled. Product names, domains, cloud project and storage names, specific version strings and the identity of the enumerated enterprise have been removed.

Key issue: authorization decided on the client, not the server

The most serious findings shared one root cause: the backend trusted identifiers and permissions supplied by the client instead of enforcing them server-side.

1. **Enumerate.** One API endpoint let any authenticated user list every organization in the system, exposing thousands of organization identifiers.
2. **Pivot.** Those identifiers fed into other endpoints that did not check ownership, so a single ordinary account could read and modify user records across organizations, and tens of thousands of corporate user records were retrievable within the test window.

3. **Cross tenants.** One set of credentials was accepted across tenant subdomains with no backend check that the user belonged to that tenant, dissolving the boundary between customers.
4. **Reach the cloud.** Error messages leaked internal storage paths, and read-only audit roles turned out to be enough to download sensitive files directly from cloud storage.
5. **Map the logic.** Unauthenticated API queries exposed internal workflow definitions and user email addresses, giving an attacker a full picture of how the platform worked before even logging in.

A single architectural change, enforcing authorization and tenant binding on the server for every request, would have closed the critical and most of the high findings at once.

What was working

- ✓ Some known framework and library flaws were present by version but not exploitable, because the surrounding implementation did not expose the vulnerable paths.
- ✓ HTTPS was enforced across the platform, with insecure requests redirected.
- ✓ The engagement covered both staging and production, confirming the same issues in both rather than assuming parity.

Remediation themes

- Enforce object-level and function-level authorization on the server for every request, keyed to the caller's identity and tenant, never to client-supplied identifiers.
- Disable API schema introspection in production and require authentication on every data query.
- Tighten cloud IAM so that audit roles cannot read production data, and remove verbose storage-path errors.
- Invalidate session tokens server-side on logout, and bring file-upload validation to the point of upload.

Outcome. The report gave the platform a prioritized path to close the systemic authorization gaps before scaling, with a retest to confirm the fixes and an attestation letter for compliance.