



Breaking the Trust Boundary: Uncovering Six Critical Account-Takeover and Privilege-Escalation Flaws in a Multi-Role Enterprise Portal

Research & Assessment: Pranesh Kohli, Penetration Tester | Ampcus Cyber

01 Client Profile:

Client profile	B2B distribution Medical & Healthcare US
Platform assessed	Web-based, multi-role portal (administrator and distributor-tier accounts); self-service registration, OTP password reset, profile management, and an admin user-management console
Engagement type	Manual application penetration test focused on authentication, authorization, and account-management logic
Methodology	OWASP Web Security Testing Guide (WSTG v4.2) · OWASP Top 10 (A01) · OWASP API Security Top 10 (API1: BOLA) · NIST SP 800-115
Key findings	6 authentication & authorization flaws 4 High / 1 High-Critical / 1 Medium (see Severity Matrix)
Headline outcome	Prevented a distributor-to-admin account-takeover chain and closed systemic client-side trust failures before they could be exploited in production

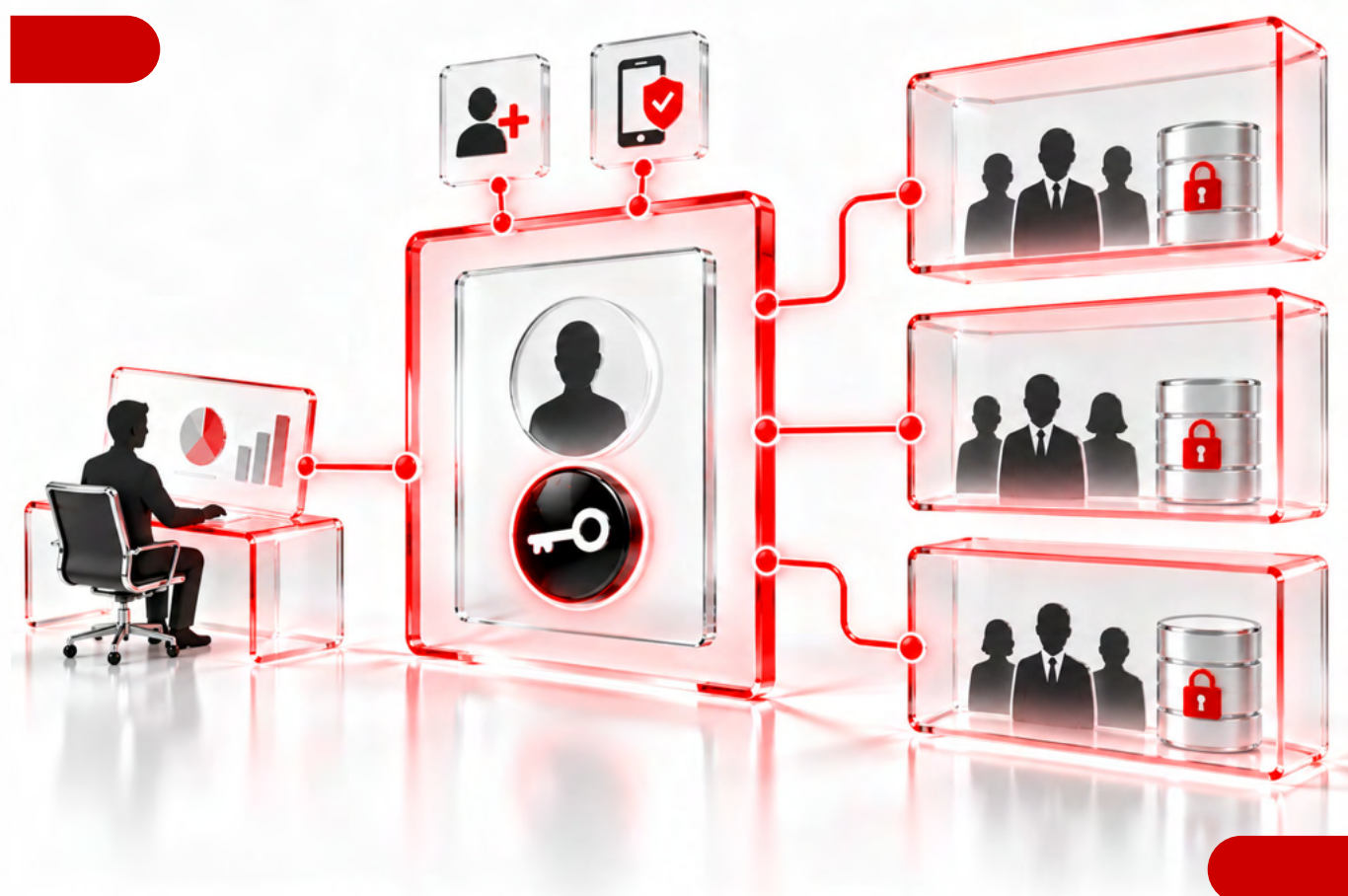
02 Executive Summary

A single low-privilege distributor account should never be able to become an administrator, reset another user's password, or read another customer's data. During a manual penetration test of the client's multi-role portal, Ampcus Cyber demonstrated that it could do all three: repeatedly, and by different routes.

The portal enforced its security decisions where users could see them, hiding fields and disabling controls in the interface, but failed to re-verify identity, role, and record ownership on the server, where enforcement counts. That single design flaw surfaced as six distinct vulnerabilities spanning privilege escalation, CAPTCHA/anti-automation bypass, one-time-passcode replay, and Broken Object Level Authorization (BOLA/IDOR).

For a multi-role distribution platform, the business exposure is not abstract. An attacker holding nothing more than a valid distributor login could have escalated to administrator, broken the isolation between distributor tenants, tampered with account and order data, and taken over arbitrary user and administrator accounts, the kind of event that triggers regulatory breach-notification obligations (e.g., GDPR / CCPA), erodes partner trust, and opens the door to supply-chain manipulation. Critically, the client's automated scanners had not flagged these issues: everyone was a business-logic and trust failure that only adversarial manual testing reliably exposes.

03 Company Overview



The client operates a web-based portal that serves multiple classes of users, internal administrators alongside external, distributor-tier accounts. The platform supports self-service account creation, OTP-based password reset, user profile management, and an administrative console used to oversee accounts across the environment.

Because the portal governs different privilege levels and custodies account credentials and partner data, the integrity of its authentication and authorization logic is foundational. Any weakness that lets one user act as another or a lower-privilege user assume a higher-privilege role directly threatens the confidentiality, integrity, and tenant isolation of every account on the system.

04 The Problem: When the User Interface Enforces Security, the Server Doesn't

Modern portals are typically built as decoupled applications, a rich browser front end (single-page application) talking to a REST or API back end. This architecture is fast and flexible, but it invites a specific and dangerous antipattern: the front end is trusted to enforce the rules. Buttons are hidden, fields are marked read-only, and roles are read from whatever the server returned. The problem is that anything the browser can hide, an attacker can un-hide and anything the client sends, an attacker can rewrite.

The client engaged Ampcus Cyber to pressure-test exactly this seam. Early in the assessment a consistent theme emerged: the application repeatedly trusted data it should never have trusted, roles returned in responses, identifiers submitted in requests, and OTP-validation outcomes that could be substituted. Individually, each flaw was serious. Collectively, they meant the boundary separating a normal user from an administrator, and one account holder from another, could be crossed at will.

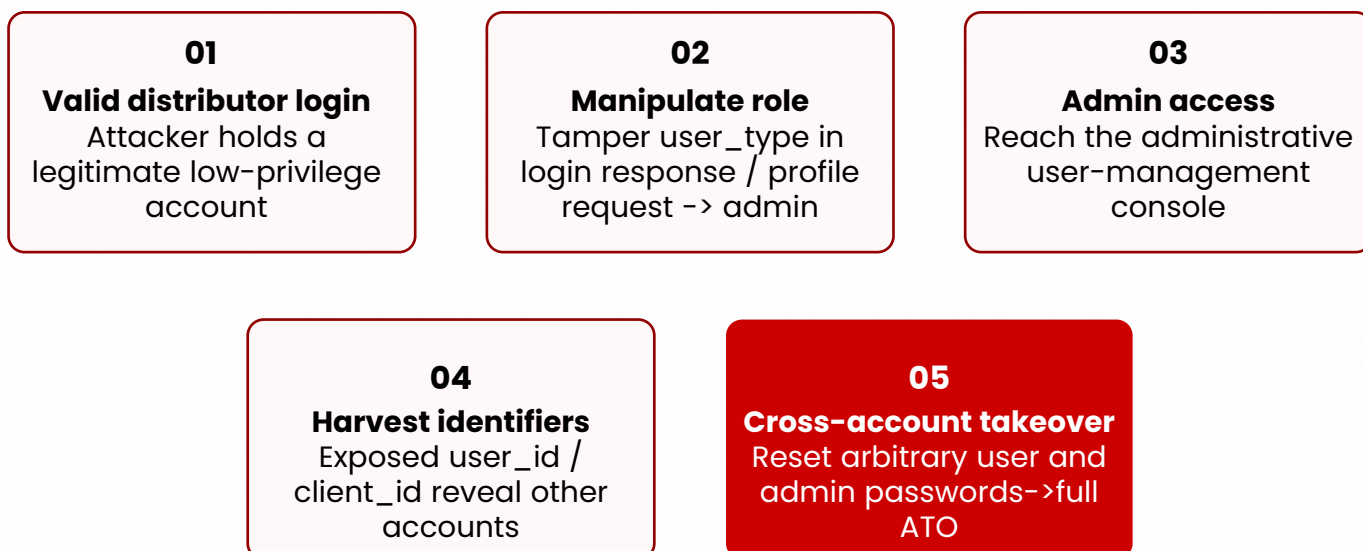
This class of weakness dominates real-world breach data. Broken Access Control is ranked the No. 1 risk in the OWASP Top 10 (2021), and Broken Object Level Authorization (BOLA) sits at the top of the OWASP API Security Top 10 (2023). The findings below are textbook illustrations of why: they are simple to exploit, invisible to automated scanners, and severe in impact.

05 Vulnerability Severity & Impact Matrix

#	Vulnerability	CWE	CVSS	Severity	Retest
1	Privilege escalation via login-response manipulation	CWE-269/ CWE-284	8.8	High	[Verified closed]
2	Persistent vertical privilege escalation via profile parameter tampering	CWE-639/ CWE-269	8.8	High	[Verified closed]
3	CAPTCHA / anti-automation bypass via single-packet race condition	CWE-362/ CWE-799	5.3	Medium	[Verified closed]
4	Account takeover via OTP response replay	CWE-294/ CWE-640	8.1	High	[Verified closed]
5	Unauthorized password change via user-ID manipulation (IDOR/BOLA)	CWE-639/ CWE-284	8.1	High	[Verified closed]
6	Account takeover via excessive data exposure + broken access control	CWE-200/ CWE-639	8.1	High	[Verified closed]

06 Attack Flow: Distributor-to-Admin Takeover Chain

The findings were not isolated. Chained together, they form a complete kill chain from a single low-privilege account to full administrative control:



07 The Ampcus Cyber Approach



Ampcus Cyber conducted a manual, business-logic-driven penetration test, which routinely miss authorization flaws because each malicious request is, on its own, perfectly well-formed. Working through an intercepting proxy (Burp Suite) to observe and rewrite live traffic, our team applied a structured adversarial authorization methodology aligned to OWASP WSTG v4.2 and NIST SP 800-115.

Finding 1: Privilege Escalation via Login-Response Manipulation

Severity: High

CVSS v3.1: 8.8

CWE-269 · CWE-284

Exploit mechanism: After authenticating with valid low-privilege credentials, the team intercepted the server's login response and observed the account role returned in a `user_type` field set to `distributor`. Altering that value to `admin` before the response reached the browser, then forwarding it, granted immediate access to administrative functionality.

Business impact: The application derived effective privilege from a value it handed to the client and trusted back, allowing any authenticated user to self-promote to administrator and reach controls intended only for internal staff.

Remediation: Never trust client-visible role data. Derive authorization from the authenticated server-side session on every request; treat any role field in a response as display-only and non-authoritative.

Finding 2: Persistent Vertical Privilege Escalation via Profile Parameter Tampering

Severity: High

CVSS v3.1: 8.8

CWE-639 · CWE-269

Exploit mechanism: A structurally distinct instance of the same trust failure appeared after login. Within the "My Profile" section an `account_role` field was present but restricted at the interface level. Intercepting the underlying request revealed the role set to `distributor`; changing it to `admin` and forwarding the request caused the elevated role to persist. After logging out and back in, the account retained full administrative access.

Business impact: Unlike Finding 1, this escalation was durable — it survived re-authentication, converting a transient bypass into a permanently compromised, attacker-controlled admin account.

Remediation: Enforce field-level authorization server-side. Reject any attempt to modify privilege-bearing attributes (`account_role`, `user_type`) from a self-service profile context, regardless of what the UI exposes.

Finding 3: CAPTCHA / Anti-Automation Bypass via Single-Packet Race Condition

Severity: Medium**CVSS v3.1: 5.3****CWE-362 · CWE-799**

Exploit mechanism: The account-creation workflow was protected by a CAPTCHA intended to block bulk or automated registration. The team captured one valid creation request, replicated it across multiple Repeater tabs, grouped them, and executed a single-packet attack, a technique that leverages HTTP/2 request multiplexing to deliver many requests within a single TCP packet, eliminating network jitter so they arrive effectively simultaneously and defeat checks that assume requests are processed one at a time. The CAPTCHA was bypassed and multiple account-creation requests were accepted for the same email address.

Business impact: The anti-automation control gave a false sense of protection. In practice it could be circumvented to enable bulk or duplicate account creation — a precursor to abuse, resource exhaustion, spam, and downstream fraud.

Remediation: Enforce anti-automation server-side with atomic, idempotent handling of registration and rate limiting resilient to concurrent / single-packet request patterns; validate CAPTCHA state within the same transaction that creates the account.

Finding 4: Account Takeover via OTP Response Replay

Severity: High**CVSS v3.1: 8.1****CWE-294 · CWE-640**

Exploit mechanism: The password-reset workflow relied on a one-time passcode. The team first completed a legitimate reset for a controlled account (User A) and captured the server's success response. They then initiated a reset for a different account (User B), submitted an arbitrary incorrect OTP, and substituted the previously captured success response in place of the genuine failure. The application treated verification as passed and allowed User B's password to be reset without the correct code ever being known.

Business impact: OTP validation was decided on the client side rather than confirmed authoritatively by the server. Because the success indicator could be replayed across accounts, any account with a known email address was open to complete takeover.

Remediation: Validate OTPs server-side against a single-use, time-bound, account-bound token. Bind the verification result to the server session — never to a response the client can capture and replay — and invalidate the code on first use.

Finding 5: Unauthorized Password Change via User-ID Manipulation (IDOR / BOLA)

Severity: High**CVSS v3.1: 5.3****CWE-362 · CWE-799**

Exploit mechanism: The change-password function identified its target account using a system-assigned `user_id` submitted in the request, without verifying that the identifier belonged to the authenticated session. This produced two exploitable scenarios, documented below.

Business impact: The server decided whose password to change purely from a client-supplied identifier, a classic Broken Object Level Authorization flaw affecting both ordinary users and privileged administrators.

Remediation: Bind every object-level action to the authenticated session. Resolve the target account from the server session, not from a client-supplied `user_id`; enforce an explicit ownership/permission check before any password mutation.

Sub-case A: Cross-user password change.

Logged in as User A, the team captured the unique `user_id` from a change-password request. After re-authenticating as User B and initiating the same action, they swapped in User A's identifier and successfully changed User A's password from an entirely separate session.

Sub-case B: Admin-to-admin password change.

Logged in as an administrator, the team opened the user-management console. While viewing another administrator's profile, a `client_id` was exposed directly in the URL. Combined with the `user_id` pattern from the change-password request, the team altered the target identifier and reset another administrator's password, despite the acting administrator having no authorized permission to do so. This demonstrated that even the platform's own permission model for privileged users could be circumvented.

Finding 6: Account Takeover via Excessive Data Exposure & Broken Access Control

Severity: High**CVSS v3.1: 8.1****CWE-294 · CWE-640**

Exploit mechanism: While authenticated, the team observed that the user-management functionality returned — through a backing API, information about other users, including their roles, that the logged-in account was not authorized to see. By capturing that exposed data and manipulating the corresponding response to reference a different, higher-privilege user, the team accessed that user's account and its associated privileges, resulting in full account takeover.

Business impact: Excessive data exposure (leaking other users' identifiers and roles) combined with weak object-level authorization gave an attacker both the target and the means to impersonate it, the two ingredients of a reliable, repeatable takeover.

Remediation: Apply least-privilege response filtering so an API never returns records or role data beyond the requester's authorization; enforce object-level access checks on the server for every read and write.

07 Root Cause & Remediation Roadmap

Every finding traced back to a single systemic weakness: the application trusted client-controlled data to make security decisions. Whether the trusted value was a role in a response, an identifier in a request, or an OTP outcome, the server failed to independently and authoritatively verify identity, role, and object ownership on the operations that mattered most.

Ampcus Cyber delivered prioritized, developer-ready remediation guidance addressing this root cause:

- ❑ Server-side authorization on every sensitive action, verify role and object ownership from the authenticated session, never from a client-supplied `user_type`, `user_id`, or `client_id`.
- ❑ Authoritative server-side validation of OTPs and reset flows, so success is determined by the server and cannot be replayed or substituted across accounts.
- ❑ Anti-automation enforced server-side, with atomic handling and rate limiting resilient to concurrent / single-packet request patterns.
- ❑ Elimination of sensitive identifiers and role data from responses, URLs, and APIs beyond what the requesting user is authorized to see, following least-privilege and secure-by-design principles.
- ❑ A validation retest to confirm each fix closed the underlying gap, not merely its surface symptom.

08 Outcomes

Working from Ampcus Cyber's proof-of-concept exploits and remediation guidance, the client's engineering team was able to:

- ▣ Close the systemic client-side trust failures underlying all six findings by re-basing authorization on server-side session state.
- ▣ Remediate [100% of Critical and High findings].
- ▣ Achieve [all findings verified closed] during Ampcus Cyber's validation retest.
- ▣ Harden the portal's authentication and account-management surface before the exposure could be exploited in production.

09 Conclusion

What began as a routine application assessment revealed a chain of authentication and authorization flaws severe enough that, left unaddressed, an attacker could have escalated from a low-privilege distributor account to administrator, bypassed anti-automation defenses, and taken over arbitrary user and administrator accounts, all without brute-forcing a single password.

The engagement underscores a lesson the OWASP Top 10 has reflected for years: the most damaging web-application vulnerabilities are rarely exotic. They are logic and trust failures that automated tools miss and that only manual, adversarial testing reliably uncovers. By exercising the application the way a real attacker would, manipulating responses, tampering with parameters, and probing every trust boundary, Ampcus Cyber surfaced issues a compliance checklist or vulnerability scan would almost certainly have overlooked, and gave the client's team the blueprint to rebuild the portal's trust boundaries on server-side enforcement.



Could Your Application Withstand a Real-World Attack?

Identify hidden vulnerabilities, strengthen access controls, and validate your defences with Ampcus Cyber's Advanced Application Penetration Testing.

Connect with Our Experts to Secure Your Applications.