Table of Contents

Incident Analysis

Time zone corrected from PST to UTC
API Logs: material/api_logs.csv
Email Logs: material/email_logs.csv
WAF Logs: material/waf_logs.csv
Web Application Logs: material/web_logs.csv

The investigation identified a coordinated multi-vector attack originating from a single external IP address, `203.0.113.45`. This threat actor successfully executed an attack, compromising the Mobile API via Broken Object Level Authorization (BOLA), the Web Application via SQL Injection, and targeting employees with a phishing campaign.

Before analyzing the attacker's actions, all log activity was examined to eliminate false positives from legitimate testing. (`security_test_schedule.pdf`)

- API and WAF logs from 01:30 to 01:31 originating from the internal IP `192.168.1.100` directly aligns with the "*Test 1: Automated Vulnerability Scanning*", which is scheduled for every Tuesday at 01:30 AM.
- API logs at 01:45 from IP `10.0.0.50` (within the `10.0.0.0/24` test range) and using the `sec_team` user corresponds to authorized internal testing activity.

The malicious activity, however, is clearly distinct from this noise. It originates entirely from `203.0.113.45` on October 15, 2024.

A critical finding is that this IP address does fall within the approved source IP range (`203.0.113.0/24`) for the "*Test 2: Quarterly Penetration Test*". However, that test was explicitly scheduled to begin **five** days later, on October 20-25, 2024.

This date discrepancy, combined with two other key facts:

- The attacker compromised a real user account (`user_id: 1523`).
- This account is not one of the designated test accounts (`5001–5010`).

...leads to a clear conclusion: This was **not** an authorized, scheduled test. The event is a genuine, high-priority security incident, originating either from an external adversary or an unsanctioned, out-of-bounds action by the approved vendor.

Timeline reconstruction

06:45:10 (API Log):

The attacker (with the IP address ***203.0.113.45***) authenticates to the Mobile API (with the User-Agent *`Acme-Mobile-Android/3.2.0`*) using stolen credentials *`jwt_token_1523_stolen`*.

14:46:30 (API Logs):

The attacker sends a GET request to `/api/v1/portfolio/1523` to verify access and test account capabilities.

14:47:15 (API Logs):

The attacker launches an **IDOR (Insecure Direct Object Reference)** attack and successfully (`HTTP 200`) retrieves the portfolio data of user with the id **1524** using the GET endpoint `/api/v1/portfolio/1523` with the **session token of the user 1523.**

14:47:18 – 14:47:57 (API Logs):

The attacker iterates over sequential user IDs (from **1524** to **1538**) and retrieves portfolio data of those users.

14:47:30, 06:47:45 (WAF Logs):

WAF detects the event as "**Rapid Sequential Access**" (**Rule ID 942100, Severity MEDIUM**) but the flag 'Blocked = no' so no action is taken for the event. First event log is on `'/api/v1/portfolio/1529'` and the second one is on `'/api/v1/portfolio/1534'`

14:47:57 (WAF Logs):

WAF detects the behaviour as "**Possible Account Enumeration**" (**Rule ID 942100 though**) but event though the severity is **HIGH**, no action is taken (Blocked = no).

17:00:23 – 17:00:33 (Email Logs):

The attacker sends multiple phishing emails from a typosquatted address (**security@acme-finance.com**) to six employees (`user1, user2, user3, user4, user5, user6`). IP address is the same as the previous malicious events (**203.0.113.45**) Three of the users clicked the malicious link:

- **user1** clicks at 17:00:23
- **user3** clicks at 17:00:27
- **user5** clicks at 17:00:31

17:00:23 (WAF Logs):

The attacker checks if there is a `/verify-account.php` page on the application. WAF detects this event and classifies it as '**Suspicious Link Pattern'**. Severity is HIGH and request is not blocked by WAF.

17:18:30 (Web Application Logs):

The attacker logs into the web application with the User-Agent "`Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/118.0`" using the credentials of the user_id 1523 again.

17:20:30, 09:21:15, 09:22:00 (Web Application and WAF Logs):

The attacker's first SQL injection attempt is detected by WAF (Rule ID **981173**, severity **HIGH,** action **DETECT,** signature '`SQL Injection Attempt - OR 1=1'`) and the request is blocked (Blocked: yes) (HTTP 403). The attacker used the query '`ticker=AAPL' OR 1=1--`'.

Second SQL injection attempt (query: `ticker=AAPL'; DROP TABLE users--`, payload: `DROP TABLE users--`), blocked by WAF (Rule ID 981318, severity CRITICAL, signature 'SQL Injection - DROP TABLE')

Third SQL injection attempt (query: "ticker=`AAPL' UNION SELECT * FROM users--` payload: `UNION SELECT * FROM users--`), blocked by WAF (Rule ID 981257, severity CRITICAL, signature 'SQL Injection - UNION SELECT')

17:23:45 (Web Application and WAF Logs)

Successful SQL injection, the attacker bypassed the WAF. A MySQL comment-based payload (ticker= `/*!50000OR*/`) is used to evade detection. WAF detects the event (Rule **981001**, signature **Suspicious SQL Pattern**) but does not block the request.

Web log shows HTTP 200 Ok and the response_size has a remarkably increased from (Response_size_bytes: `156789 bytes`) indicates data exfiltration.

17:24:10 (Web Application Logs)

The attacker accesses `/dashboard/export` and exfiltrates `892341 bytes` **of data.**

Attack Vector Identification

API: Broken Access Control on '`/api/v1/portfolio/{account_id}`' leading IDOR.

Web Application: SQL İnjection on '`/dashboard/search'`.

Employees: Targeted phishing emails with a typosquatted sender domain.

Attack Classification

OWASP Top 10 (2025)

Vector	Category	ID
API	Broken Access Control	A01:2025

API	Broken Object Level Authorization	API1:2023
Web Application	Injection	A05:2025
WAF Misconfiguration	Security Misconfiguration	A02:2025

MITRE ATT&CK

Phase	Technique
Initial Access	T1566.002 Spearphishing Link
Defense Evasion	T1027 Obfuscated/Concealed Payloads
Discovery	T1087.004 Account Discovery
Collection	T1074 Data Staging
Exfiltration	T1041 Exfiltration Over Command-and-Control Channel

ISO 27001

Incident	A control
SQL Injection & BOLA/IDO	A.14.2.1, A.14.2.5, A.9.4.1
WAF Misconfiguration	A.13.1.1
Phishing & Email Gaps	A.7.2.2, A.13.1.2
Credential Compromise	A.9.4.2

Root Cause Analysis

RCA	Description
1 API Authorization Failure	The API validates authentication but not authorization. Any authenticated user can access another user's portfolio by modifying account_id.
2 SQL Injection	The web app constructs SQL dynamically instead of using prepared statements.
3 WAF Misconfiguration	The WAF rules for the API are set to detect-only. The web WAF relies on blacklist patterns and is bypassed using obfuscated SQL payloads.
4 Email Authentication Gaps	Missing or weak SPF/DKIM/DMARC allows typosquatted email spoofing.
5 Account Compromise	user_id: 1523 credentials were already compromised prior to the incident. Initial compromise source is unknown.

Impact Assessment

Impact	Description
Data Breach	Portfolio data of at least 15 customers (IDs 1524–1538) exposed via API.

Data Exfiltration	892 KB of sensitive data extracted through SQL injection.
Credential Compromise	Phishing results in 3 employees clicking the malicious link

Architecture Review

Based on the provided current_architecture.png:

Weaknesses

WAF

The WAF is labeled “Basic Rules” and “Partially Secure”. This is insufficient as it was bypassed by the attacker’s obfuscated SQL injection payload (`‘/*!50000OR*/`).

API Gateway

The API Gateway is labeled “Partially Secure” and its only documented function is “TLS Termination”. It fails to validate the JWT token to check if the `user_id` in the token matches the `account_id`, leading to IDOR attacks.

Trading API

Labeled “Critical Vulnerability”, this application has an IDOR (BOLA) vulnerability. It executes the queries without performing object-level authorization it receives from the application.

Web Application

Labeled “Critical Vulnerability”, this application is vulnerable to SQL injection. The diagram shows it sends the SQL queries directly to the database, unsanitized queries from user input.

Email Gateway

Labeled as “Partially Secure”, email gateway fails to detect and block the phishing mails from a typosquatted domain. This is due to lack of security contrls like DMARC and anti-spoofing filters.

Recommended Security Controls

Email Gateway → Add security controls such as DMARC, DKIM, SPF Enforcement.

WAF → Add Managed Rules based on OWASP Core Set, add custom rules dedicated to block API enumeration patterns.

API Gateway

The Gateway must validate the JWT token from Auth Service.

- Extract the `user_id` and compare it with the `account_id`
- If ids don’t match, return `403 Forbidden`, otherwise, forward to backend services.

Web Application → Remediate the unsanizited and not validated user inputs, use parameterized queries AKA prepared statement for database queries.

Trading API → Remediate the object-level authorization on code level, validate the user and the resource owner.

Database Layer → Add a new component “Firewall” between the applications and the database. Whitelist allowed queries.

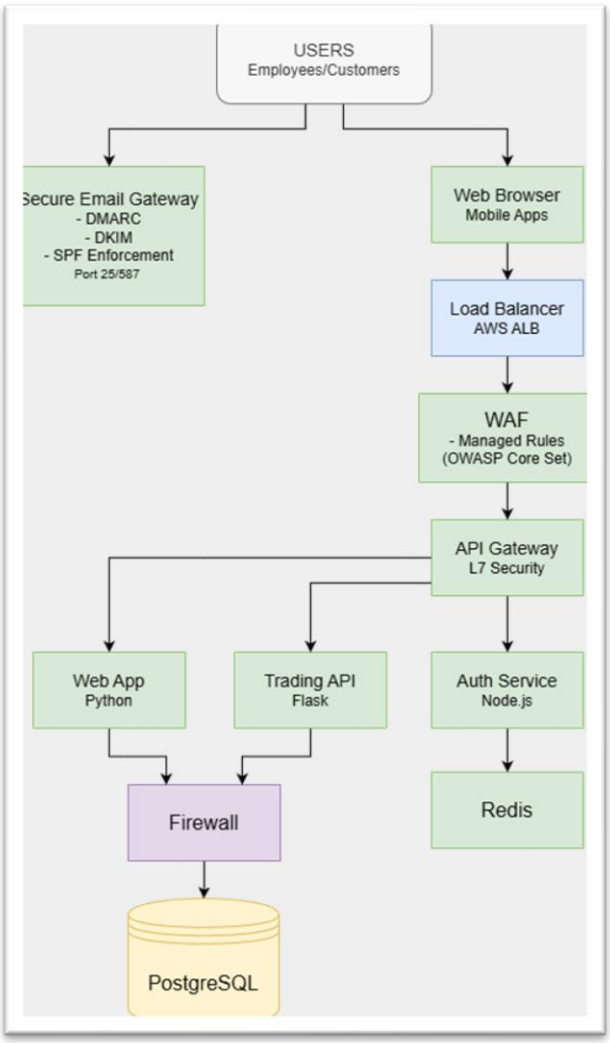
Email → DMARC, DKIM, SPF Enforcement

Web → Parameterized Queries

API → Centralized API Gateway Authorization

WAF → Managed & Custom Rules

Database → Database Access Control/Firewall



Defense-in-Depth Strategy

Layer 1 (Edge) → The Secure Email Gateway blocks phishing. The WAF blocks common attacks. The API Gateway blocks BOLA/IDOR.

Layer 2 (Application) → Secure code (Parameterized Queries) in the Web App and Authorization checks in the Trading API provides a redundant defense against BOLA.

Layer 3 (Data) → The Database Firewall acts as a final backstop, preventing unauthorized query types from reaching the data itself.

Response & Remediation

Immediate Actions (Containment - 0-24 Hours)

Block the Attacker IP → Immediately add a "block" rule to the edge firewall/WAF for the source IP `203.0.113.45` to stop the ongoing attack.

Rotate All Credentials

- Force a password reset for `user_id: 1523` immediately.
- Force password resets for `user1, user3`, and `user5` (the phishing victims).
- Invalidate Active Sessions: Expire all active web and API sessions for the compromised users (especially `jwt_token_1523_stolen`).

Short-Term Fixes (Remediation - 1-2 Weeks)

Patch the BOLA (IDOR) Vulnerability

- Implement the authorization logic check at the API Gateway level.

Patch the SQLi Vulnerability

- The Web App team must immediately rewrite the /dashboard/search function to use parameterized queries (prepared statements).

Upgrade WAF to Blocking Mode

- Switch all relevant WAF rules (for SQLi and API enumeration) from "DETECT" to "BLOCK".

Deploy DMARC

- Implement DMARC, DKIM, and SPF records for acme.com stop email spoofing.

Long-Term Improvements (Hardening - 1-3 Months)

- Implement the new WAF -> API Gateway architecture we designed.
- Enforce Code-Level Authorization, While the Gateway provides the edge fix, the Trading API (Flask) application itself must be refactored to perform its own object-level authorization checks.
- Launch a mandatory, company-wide security awareness program focused on identifying phishing attacks.