Incident Response Report:

Acme Financial Coordinated Threat Analysis

Incident ID: IR-20241015-001

Analyst: Oğuzhan Çilkurt

Report Date: November 8, 2025

CONTENTS

1. Executive Summary

On October 15, 2024, a multi-vector, coordinated cyber attack was detected against the Acme Financial Services platform. Following a detailed investigation, it has been confirmed that this activity was a planned external penetration test, as defined in the security_test_schedule.pdf document and executed from the 203.0.113.45 IP address.

While this test was not an active threat, it successfully exploited three critical vulnerabilities in our production systems:

API - Broken Object Level Authorization (BOLA): An authorization token (jwt_token_1523_stolen) belonging to one user was used to gain unauthorized access to other users' portfolio data (1524 - 1538). This exploited a vulnerability noted in the api_docs.pdf which states that checks "may not verify account ownership".

Web Application - SQL Injection (SQLi): The Web Application Firewall (WAF) was bypassed using an advanced query (/*!50000OR*/) due to a rule (981001) being left in DETECT mode, allowing unauthorized database access.

API - Weak Rate Limiting: A rate limiting vulnerability, noted in the API documentation as "may not be strictly enforced", allowed the BOLA attack to be executed rapidly. This activity was detected by the WAF as "Rapid Sequential Access" but was not blocked (blocked: no).

This report details the technical evidence for each finding and presents an action plan for the permanent remediation of these vulnerabilities.

Section 1: Incident Analysis

1.1. Attack Timeline (UTC Normalized)

The analysis was conducted across all log files (api_logs.csv, web_logs.csv, waf_logs.csv, email_logs.csv) by pivoting on the planned test IP, 203.0.113.45. Other IPs, such as 192.168.1.100 (automated scanning) and 172.89.15.67 (normal user), were triaged as "False Positives" and excluded from this timeline.

October 15, 2024 (All Times UTC)

03:45:10: The test analyst (203.0.113.45) successfully logs in via /api/v1/login as user 1523, obtaining jwt_token_1523_stolen.

03:47:15 - 03:47:57: (BOLA & Rate Limit Attack) Using the token for user 1523, the analyst queries the /api/v1/portfolio/ endpoint for different users (from 1524 to 1538) within 42 seconds. All requests return a 200 OK (Success) response.

03:47:45: (WAF Detection - Failed Block) The WAF detects this rapid activity with rule_id: 942100 as "Rapid Sequential Access", but due to action: DETECT and blocked: no, it does not block the requests. This confirms the "Rate Limit" vulnerability.

06:00:23 - 06:00:31: (Phishing Attack) From the same IP (203.0.113.45), phishing emails with the subject "URGENT" are sent using the spoofed address security@acme-finance.com.

06:00:27: Users user3 and user5 click the malicious link (link_clicked: yes).

06:20:30 - 06:22:00: (SQLi Attempt - Failed) The analyst attempts standard SQLi queries (OR 1=1, UNION SELECT) on /dashboard/search. The WAF flags these as "CRITICAL" and successfully blocks them (blocked: yes).

06:23:45: (SQLi WAF Bypass - Successful) The analyst sends a MySQL-specific query (/*!50000OR*/ 1=1--) to bypass the WAF rule.

06:23:45 (WAF Detection - Failed Block): The WAF detects this attempt with rule_id: 981001 as a "Suspicious SQL Pattern", but as the rule is in DETECT mode, it is not blocked (blocked: no).

06:24:15: A web_logs.csv entry confirms data exfiltration via the /dashboard/export endpoint immediately following the successful SQLi.

1.2. Attack Classification (OWASP & MITRE ATT&CK)

The identified vulnerabilities are classified according to industry-standard frameworks as follows:

1. API Vulnerability (BOLA):

OWASP API Top 10: API1:2023 - Broken Object Level Authorization (BOLA). This is confirmed by the analyst's ability to access data for account 1538 using credentials for account 1523. This is directly correlated with the note in api_docs.pdf stating "may not verify account ownership".

2. Web Application Vulnerability (SQLi):

OWASP Top 10: A03:2021 - Injection. An endpoint was identified that accepts non-parameterized queries, allowing WAF bypass techniques like /*!50000OR*/.
MITRE ATT&CK: T1190 - Exploit Public-Facing Application.

3. API Vulnerability (Rate Limit):

OWASP API Top 10: API4:2023 - Unrestricted Resource Consumption. The vulnerability, noted in api_docs.pdf as "may not be strictly enforced", was confirmed by the WAF's failure to block (blocked: no) the "Rapid Sequential Access" attempt.

4. Email Vulnerability (Phishing):

MITRE ATT&CK: T1566.002 - Spearphishing Link. Targeted phishing emails were sent to users (user3, user5). This indicates a lack of email authentication mechanisms such as SPF, DKIM, and DMARC.

1.3. Root Cause Analysis and Impact Assessment

Root Cause: The root cause of this incident is a multi-layered failure in the "Defense-in-Depth" strategy. The vulnerabilities were known at the documentation level (BOLA, Rate Limiting), yet they were not remediated at the code level (Trading API, Web App) nor mitigated at the security layer (WAF). Leaving critical WAF rules in DETECT mode turned a security control into a passive monitoring tool.

Impact: The successful BOLA and SQLi attacks demonstrate that the analyst (in this case, a tester) gained full access to all customer portfolio data, balances, and potentially transaction histories. This could lead to mass data exfiltration, financial loss, and severe legal penalties under regulations like GDPR and PCI-DSS .

Section 2: Architecture Review

2.1. Current Architecture Weaknesses

The current_architecture.png diagram reveals the core design weaknesses that allowed these attacks to succeed:

Insufficient Authorization (BOLA): The most critical weakness is the gap between the Auth Service (Authentication) and the Trading API (Resource Access). The diagram shows the Auth Service issues a token, but the Trading API only validates it. There is no centralized authorization mechanism to check if the token's owner matches the requested resource (account_id).

Ineffective WAF Configuration: The WAF is at a critical point, yet logs prove that critical SQLi (981001) and Rate Limit (942100) rules were left in DETECT mode, rendering the WAF control ineffective.

Insecure Direct Data Access: The red SQL arrows in the diagram show the Web App and Trading API sending queries directly to the PostgreSQL database. This design discourages the enforcement of prepared statements and maximizes the risk of SQL injection.

Lack of Email Security: The Email Gateway failed to filter spoofed emails from an external IP impersonating the internal domain (acme-finance.com), indicating a lack of SPF, DKIM, and DMARC controls.

2.2. Recommended Secure Architecture Diagram & Controls

To address the identified weaknesses, the following architectural improvements, based on the "Defense-in-Depth" principle, are recommended:

Control: Centralized Authorization (API Gateway): The API Gateway must be elevated from TLS termination to perform Centralized Policy Enforcement. It must integrate with the Auth Service to validate every JWT token and enforce that the user_id in the token matches the requested account_id. All non-matching requests must be rejected with 403 Forbidden.

Control: Set WAF to "Block Mode": The WAF must be configured in BLOCK mode for all SQLi (981001), Rate Limit (942100), and other critical OWASP rules.

Control: Email Security (DMARC): A DMARC (p=reject) policy must be implemented for the Email Gateway to prevent domain spoofing.

Control: Data Access Layer (DAL): Direct SQL access from applications must be forbidden. All database communication must go through a Data Access Layer (DAL) that strictly enforces the use of prepared statements.

Section 3: Remediation and Response

3.1. Immediate Containment (0-24 Hours)

- Revoke Test Assets: The jwt_token_1523_stolen used in the test must be immediately revoked, and the 1523 test account must be disabled.
- Update WAF Rules: All critical rules identified in waf_logs.csv as being in DETECT mode (including 981001 and 942100) must be immediately moved to BLOCK mode.
- Verify Data Exposure: Confirm whether the accounts accessed (1524 - 1538) contained real customer data or were within the defined scope of the penetration test.

3.2. Short-Term Remediation (1-2 Weeks)

- API BOLA Hotfix: An emergency patch must be deployed to the Trading API code. This patch must add a check to the /portfolio/{account_id} endpoint to compare the user_id from the token with the account_id in the path. If they do not match, return 403 Forbidden.
- Implement DMARC/SPF/DKIM: Create SPF and DKIM records for the acme-finance.com domain. Implement a DMARC policy set to p=quarantine for monitoring.
- Secure Coding Training: Schedule an immediate OWASP Top 10 (API & Web) training session for the development team responsible for the api_docs.pdf and its insecure notations.

3.3. Long-Term Improvements (1-3 Months)

- Secure Coding Refactor (SQLi): The entire Web App codebase must be refactored to ensure all database queries use Prepared Statements or an ORM (Object-Relational Mapping).
- Centralized Authorization (BOLA & Rate Limit): Implement the recommended architecture change: establish a centralized Policy-Based Access Control system at the API Gateway level. The insecure notes in api_docs.pdf must be removed, and all APIs integrated into this new system.
- DMARC Policy Enforcement: After the monitoring period, update the DMARC policy to p=reject.

3.4. Compliance Considerations

The identified BOLA and SQLi vulnerabilities directly jeopardize Acme Financial Services' compliance with regulations such as PCI-DSS and GDPR. Unauthorized access to user portfolio data qualifies as a data breach and could lead to severe legal and financial penalties. The recommended remediations are critical for closing these compliance gaps.