Acme Financial Services – Security Incident Report

Analyst: Enes Özkırdeniz

Date: November 08, 2025

Incident Date: October 15, 2024

Section 1: Incident Analysis

1.1 Reconstructed Timeline

The incident was a coordinated, multi-vector attack managed from a single external IP (203.0.113.45).

Crucially, analysis of the security_test_schedule.pdf document reveals the attacker's true intent. The attacker's IP (203.0.113.45) falls within the "Approved Source IP" range (203.0.113.0/24) designated for an upcoming external penetration test ("Test 2") scheduled for **October 20-25**. The attack took place on **October 15**, five days *before* the scheduled test. This indicates the attacker had possession of this internal document and was impersonating the trusted pentest vendor (CyberSec Partners) to evade detection by the SOC.

Attacker Timeline (203.0.113.45):

- **06:46:30 (API Recon):** Attacker authenticated as user_id=1523 (using jwt_token_1523_stolen) and accessed their own portfolio (Response: 200 OK).
- **06:47:15 - 06:47:57 (API Exploit):** Using the same token, the attacker requested 15+ *other* user portfolios (/api/v1/portfolio/1524...1538). **All requests returned 200 OK**.
- **09:00:12 (Initial Access - Phishing):** Attacker spoofed security@acme-financial.com and sent a phishing email to john.doe@acmefinancial.services.
- **09:00:12 (Gateway Failure):** Email Gateway logged the mail as Low Reputation Sender but **DELIVERED** it instead of blocking.

- **09:18:30 (Account Compromise):** Attacker used credentials stolen from the phishing link to successfully /login as user_id=1523.
- **09:20:30 - 09:22:00 (WAF Successful):** WAF successfully **BLOCKED** basic SQLi attempts (OR 1=1, DROP TABLE).
- **09:23:45 (WAF BYPASS):** Attacker used an obfuscated payload: ticker=AAPL' /*!50000OR*/ 1=1--. The WAF's "Basic Rules" failed to block this.
- **09:23:45 (SQL INJECTION BREACH):** The Web App log confirms the payload worked, returning 200 OK with 156,789 bytes of data.
- **09:24:10 (Data Exfiltration):** Attacker exfiltrated 892,341 bytes of data via /dashboard/export?format=csv.

1.2 Attack Vector Identification

Vector 1 (Phishing): Theft of employee credentials (user_id=1523) via a weak Email Gateway.

Vector 2 (API - Broken Access Control): Unauthorized access to other user accounts (1524-1538) via the Trading API.

Vector 3 (Web App - SQL Injection): Bypassing the WAF and exploiting the Web App using an obfuscated payload.

1.3 Attack Classification

OWASP Top 10:

- **A01:2021 – Broken Access Control:** The API failed to check if user_id=1523 was authorized for account_id=1524.
- **A03:2021 – Injection:** The Web App was vulnerable to the /*!50000OR*/ SQLi payload.
- **A05:2021 – Security Misconfiguration:** Email Gateway failed to block known spam; WAF used "Basic Rules"; admin privileges were exposed internally.
- **A07:2021 – Identification and Authentication Failures:** Successful Phishing due to a **lack of MFA**.

MITRE ATT&CK:

- **T1566.002 (Spearphishing Link): Targeting john.doe.**
- **T1078.001 (Valid Accounts):** Use of compromised user_id=1523 and admin_5678.
- **T1190 (Exploit Public-Facing Application):** SQL Injection.

1.4 Root Cause Analysis

Lack of MFA (Technical): The single most critical failure. MFA would have stopped the 09:18:30 login, rendering the phishing credential theft useless.

Insecure Coding Practices (Technical): The Web App code was not using Parameterized Queries. The Trading API code had no authorization checks.

Inadequate Configuration (Technical/Process): The WAF was left on "Basic Rules." The Email Gateway was not in "Block" mode.

1.5 Impact Assessment

Confirmed exfiltration of 892KB of data by the external attacker, 15+ portfolio records from the API, and a 245KB full user database export from the internal threat.

Complete compromise of customer PII (Personally Identifiable Information) and financial portfolio data.

Section 2: Architecture Review

2.1 Current Architecture Weaknesses

The current_architecture.png diagram visually confirms the log file analysis

Email Gateway (Orange): Marked "Partially Secure," explaining why it DELIVERED the Low Reputation phishing email.

WAF (Basic Rules) (Orange): "Basic Rules" explains why it blocked simple SQLi but was bypassed by the /*!50000...*/ payload.

Web App (Python) (Red): Marked "Critical Vulnerability," explaining why the bypassed SQLi query executed successfully (200 OK).

API Gateway (Orange) & Trading API (Red): This confirms the "Broken Access Control" failure. The Gateway (Orange) authenticated, but the API (Red) failed to authorize.

2.2 Improved Security Architecture Diagram

The goal is to turn all vulnerable (Orange and Red) components Green (Secure). This is achieved by enforcing MFA (Multi-Factor Authentication) for all users. The Email Gateway will be upgraded to ATP (Advanced Threat Protection) with sandboxing, and the WAF will use Advanced Managed Rulesets (e.g., OWASP Core Rule Set) in "Block" mode. The API Gateway will be reconfigured for Centralized Authorization to validate user_id against the requested account_id. At the code level, the Web App must be refactored to use Parameterized Queries (Prepared Statements), and the Trading API must add its own authorization checks as a defense-in-depth measure. Finally, monitoring will be

enhanced with P0 SIEM alerts for critical actions (like /admin/export) and DLP (Data Loss Prevention) for sensitive documents.

2.3 Recommended Security Controls (Justified)

Configure Cloudflare WAF to detect and block MySQL inline-comment patterns such as /*\!\d{5}.*?*/; the 09:23 exploit shows this gap is actionable. Enforce object-level authorization in the Kong API Gateway so that user_id must equal account_id, closing the IDOR issue evidenced by 15 unauthorized requests. Replace dynamic SQL with parameterized queries to remove injection avenues. Throttle /portfolio/ to ~5 requests per minute per token to deter enumeration, as demonstrated by the 57-second probing burst. Use PostgreSQL Row-Level Security to confine records to their owners and shrink the blast radius. Require MFA to stop post-phishing credential reuse. Correlate email, API, and web logs in the SIEM to surface multi-stage activity early.

2.4 Defense-in-depth Strategy

We deploy complementary controls across layers so that a single failure does not lead to compromise. Controls map to the observed attack path (phishing → session reuse → IDOR → SQL injection bypass → export) and provide prevent, detect, and respond capabilities at each tier.

Section 3: Response & Remediation

3.1 Recommended Security Controls (Justified)

In the first 24 hours, I isolate the Python web app and the 10.0.1.50 server from the network. I block 203.0.113.45 and 10.0.1.50 on the firewalls. I check that 192.168.1.100 is our internal scanner and put it on a monitored allow-list (do not block). I suspend user_id=1523 and admin_5678, and I force password reset and MFA for all staff. I save all WAF, web, API, email, and database logs to write-once storage and take a forensic image of 10.0.1.50. I also inform Legal and Compliance about the possible PII and financial data leak.

3.2 Short-term Fixes (1-2 Weeks)

In the next two weeks, the team should add a custom WAF rule to block MySQL inline comments like /*! and /*!50000 as a virtual patch for SQL injection, set the WAF to OWASP CRS in Block mode, and apply rate limiting on /api/v1/portfolio/ to stop enumeration. The team should also start a forensic analysis of the **10.0.1.50** server image to find the entry path and how the **admin_5678** credentials were taken. Finally, document sharing and logs should be reviewed to learn how **security_test_schedule.pdf** leaked, and the access rules should be fixed.

3.3 Long-term improvements (1-3 months)

In the next one to three months, the team should roll out and enforce MFA for all users, starting with employees and admins. The secure SDLC should include refactoring: use parameterized queries in the web app to remove SQLi, and make the Trading API check server-side authorization so user_id must match account_id. SIEM monitoring should add P0 alerts for any /admin/export action, for access to sensitive documents (when DLP is enabled), and for a high number of 401 errors from one internal IP. A Data Classification and DLP program should start: mark files like pentest schedules as Confidential and control or monitor sharing. Finally, apply least privilege by moving admin functions (e.g., /admin/export) to an internal-only network behind VPN, and reduce admin rights on general servers such as 10.0.1.50.

3.4 Compliance considerations

Because the incident includes PII and financial data, the company must notify affected customers and the regulators (e.g., GDPR, KVKK) within required timelines. The breach type (financial data, common OWASP issues, missing MFA) may lead to significant fines and penalties. Public disclosure will damage trust, so a customer communication plan and FAQ should be prepared. Regulators will likely require an independent external audit of the security program (e.g., SOC 2, PCI-DSS) after the incident; the company should cooperate, provide evidence, and track remediation commitments.