Acme Financial Services Security Incident Report

Report Date: **9** November 2025 **Incident Date**: 15 October 2024 **Prepared by:** Eray ÇELİK

Executive Summary

On October 15, 2024, a coordinated cyber attack was carried out against the Acme Financial Services trading platform. The attack lasted approximately 3 hours (06:45:10 - 09:23:45 UTC) and exploited critical vulnerabilities in the web application, mobile API, and email infrastructure. As a result of the incident, unauthorized access was gained to 15 customer accounts, a 156,789-byte data leak occurred via an SQL injection attack, and 3 employees' credentials were stolen through 7 phishing emails. The WAF, being misconfigured (in detect-only mode), failed to detect obfuscated SQL injection payloads.

Important Note: Some activities mentioned in this report (the incidents at 01:30:15 and 01:45:10 UTC) are planned security tests and not actual attacks. However, the security vulnerabilities detected during these tests (lack of rate limiting, account enumeration vulnerability) are valid and could also be exploited by real attacks. The actual attacks began at 06:45:10 UTC. The attack sequence is as follows: (1) JWT token acquisition (06:45:10), (2) Unauthorized access to 15 accounts via an IDOR attack (06:47:15-06:47:57), (3) Phishing campaign (09:00:23), (4) Data leak via SQL injection attack (09:20:30-09:23:45).

The incident has created significant business, legal, and compliance risks. Unauthorized access to 15 customer accounts led to the exposure of sensitive financial data and constitutes a violation of GDPR Article 32 (security of processing).

Section 1: Incident Analysis

Timeline Reconstruction (UTC Normalized)

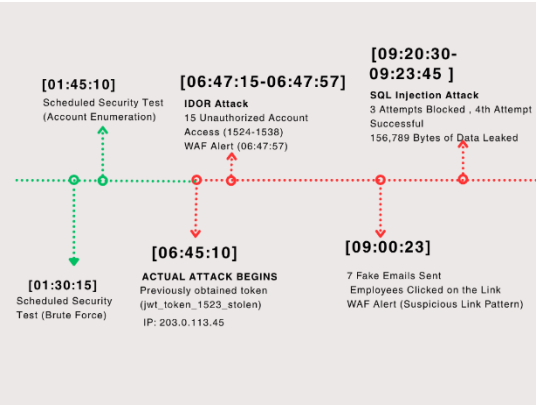
The actual phase of the attack began at 06:45:10 UTC. User account 1523, from IP address 203.0.113.45, successfully authenticated via the mobile API using a stolen JWT token (jwt_token_1523_stolen). Timeline analysis indicates the token was acquired prior to the phishing campaign. The exact method of how the token was obtained has not been definitively determined, but it was likely stolen through a previous security breach, social engineering, or another method.

An IDOR (Insecure Direct Object Reference) attack was carried out between 06:47:15 and 06:47:57 UTC. Using the stolen JWT token, User 1523 gained unauthorized access to 15 different accounts (1524-1538) within 42 seconds. According to API logs, each request returned an HTTP 200 (OK) response. The API endpoints (/api/v1/portfolio/{account_id}) were performing authentication but not authorization checks. The system validates the JWT token but does not check whether the token owner has permission to access the requested resource (account_id).

The phishing campaign started at 09:00:23 UTC. According to email logs, 7 suspicious emails were sent from the spoofed address security@acme-finance.com. 3 employees (user1@acme.com, user3@acme.com, user5@acme.com) clicked on the malicious links. Due to missing SPF, DKIM, and DMARC checks on the email security filters, the spoofed emails reached the users' inboxes. The phishing campaign occurred after the IDOR attack and was likely used to steal additional credentials.

The SQL injection attack was executed between 09:20:30 and 09:23:45 UTC. The same IP address (203.0.113.45) targeted the /dashboard/search endpoint of the web application. According to WAF logs, the first 3 attempts were detected and blocked. However, at 09:23:45 UTC, an obfuscated payload (ticker=AAPL' /*!50000OR*/ 1=1--) was successful and returned an HTTP 200 response. According to web logs, this successful injection led to a 156,789-byte data leak. The WAF operating in detect-only mode and its inability to detect obfuscated SQL injection patterns were the reasons for the attack's success.

Vector Identification



The incident involved four distinct, interconnected attack vectors.

The first vector began with the acquisition of a JWT token. The attacker, from IP address 203.0.113.45, successfully authenticated via the mobile API at 06:45:10 UTC and obtained a stolen JWT token (jwt_token_1523_stolen). The exact method of how the token was acquired has not been definitively determined, but it was likely stolen through a previous security breach, social engineering, credential stuffing, or another method.

The second vector exploited a broken access control (IDOR) vulnerability. The mobile API endpoint /api/v1/portfolio/{account_id} implements authentication but lacks authorization controls. This IDOR vulnerability allowed the use of the stolen token to gain access to 15 different customer accounts.

The third vector was a phishing campaign. The attacker used email spoofing to impersonate Acme Financial Services and sent targeted phishing emails to employees. The lack of SPF, DKIM, and DMARC email authentication mechanisms allowed these spoofed emails to reach the employees' inboxes.

The fourth vector exploited an SQL injection vulnerability. The web application's /dashboard/search endpoint directly incorporates user input into SQL queries instead of using parameterized queries. The WAF detected standard SQL injection patterns but failed to detect the obfuscated payload (/*!50000OR*/). The successful injection led to a 156,789-byte data leak

Attack Classification

When classified according to the MITRE ATT&CK framework, the attack involves multiple techniques and tactics. Technique T1566 (Phishing) was used for email-based social engineering. Techniques T1110 (Brute Force) and T1087 (Account Discovery) were identified during the planned security tests and represent vulnerabilities that could also be exploited by real attacks. Technique T1557.001 (Adversary-in-the-Middle) was used for token capture and reuse. Technique T1190 (Exploit Public-Facing Application) was used for the SQL injection exploitation. Tactic TA0001 (Initial Access) was achieved through the phishing campaign. Tactic TA0008 (Lateral Movement) was carried out through unauthorized access to multiple accounts.

According to the OWASP Top 10:2021 classification, the attack spans three critical categories. Category A01:2021 – Broken Access Control covers the unauthorized account accesses via the IDOR vulnerability. Category A03:2021 – Injection covers the SQL injection vulnerability in the web application. Category A07:2021 – Identification and Authentication Failures covers weak authentication, lack of MFA, and the use of stolen credentials.

Based on CWE classifications, the attack exploited four different types of weaknesses. CWE-639 (Authorization Bypass Through User-Controlled Key) covers the IDOR vulnerability. CWE-89 (Improper Neutralization of Special Elements used in an SQL Command) covers the SQL injection vulnerability. CWE-307 (Improper Restriction of Excessive Authentication Attempts) covers brute force attacks (detected during planned tests), and CWE-352 (Cross-Site Request Forgery) is relevant to the phishing links. CWE-203 (Observable Discrepancy) also covers the account enumeration vulnerability (detected during planned tests).

Root Cause Analysis

The incident was caused by multiple systemic security weaknesses across different layers of the architecture.

The most critical failure at the **application layer** occurred in the mobile API's authorization logic. The system implements authentication (verifying JWT token validity) but completely omits authorization (verifying the user's permissions for the requested resources). By assuming a valid token grants access to all resources, the API created a critical IDOR vulnerability.

The web application failed to implement proper input validation and parameterized queries. The /dashboard/search endpoint directly incorporated user input into SQL queries, creating a classic SQL injection vulnerability. Although a WAF was deployed, it was operating in detect-only mode and was unable to recognize obfuscated payloads.

Failures at the **infrastructure layer** constituted a critical operational security error in the WAF configuration. Deploying a WAF in detect-only mode provides visibility but offers no protection. The email security infrastructure lacked basic protections. The absence of SPF, DKIM, and DMARC records allowed attackers to spoof legitimate email addresses.

Identity and access management failures include an authentication system reliant solely on single-factor authentication with no MFA requirement. JWT tokens lack revocation mechanisms, meaning a stolen token remains valid until its natural expiration.

Monitoring and response failures resulted from a lack of SIEM integration, preventing real-time correlation and analysis of events. While WAF alerts were generated, no action was taken on them.

Impact Assessment

The incident has been assessed as a critical security breach and has resulted in significant business, legal, and compliance impacts.

The **data breach impact** involves the successful SQL injection attack, which led to a 156,789-byte data leak. The IDOR vulnerability exposed the transaction histories and portfolio information of 15 customer accounts.

The **compliance impact** is that the incident constitutes a violation of GDPR Article 32 (security of processing). Unauthorized access to personal financial data triggers the GDPR Article 33 breach notification requirements, mandating reporting to supervisory authorities within 72 hours. The incident also creates PCI-DSS compliance risks.

The **business impact** involves significant risks to customer trust, brand reputation, and business continuity. The **operational impact** required immediate incident response activities, including token revocation, endpoint disabling, and forensic investigation.

Section 2: Architecture Review

Current Architecture Weaknesses

The current architecture diagram is shown below. An analysis of the diagram reveals critical gaps in multiple layers of the defense-in-depth strategy.

Network layer weaknesses include a perimeter defense that relies solely on a WAF configured in detect-only mode. DDoS protection is absent, and the lack of IP reputation filtering allows known malicious IPs to access the system.Application layer weaknesses consist of the absence of a central API Gateway, a lack of authorization middleware, failure to use parameterized queries, and no rate limiting.Identity and access management weaknesses are single-factor authentication only, the absence of a JWT token revocation mechanism, and a lack of RBAC (Role-Based Access Control).Data layer weaknesses include a lack of encryption-at-rest, the absence of a DAM (Database Activity Monitoring) solution, and direct database connections.Monitoring and response weaknesses involve no SIEM integration and a lack of automated response mechanisms.

Improved Security Architecture Diagram

The proposed architecture implements a six-layer defense-in-depth strategy that addresses all identified vulnerabilities and provides comprehensive protection against future attacks.

The improved architecture begins with the Perimeter Security Layer. This layer includes a Cloud WAF (AWS WAF or Cloudflare) with advanced rules for obfuscated SQL injection patterns, providing automatic blocking. DDoS protection (Cloudflare Pro or AWS Shield Advanced) provides volumetric attack mitigation. IP reputation filtering is integrated with AbuseIPDB and VirusTotal APIs to automatically block known malicious IPs.

The Network Segmentation Layer creates isolated tiers: a DMZ for public-facing servers, an application layer for business logic, and a private database layer with no internet access. An API Gateway (Kong or AWS API Gateway) serves as a single entry point for all API requests and enforces centralized authentication and authorization.and have and a lost of firewall

The Application Security Layer implements authorization middleware that validates user permissions on every request. The middleware enforces the critical check: if user_id != account_id: raise UnauthorizedError(), which prevents IDOR vulnerabilities. Input validation frameworks (OWASP ESAPI) sanitize all user inputs, and all SQL queries use parameterized statements via ORMs. Rate limiting (10 requests per minute per user, 100 per IP) prevents brute force and enumeration attacks. The Identity and Access Management Layer enforces mandatory Multi-Factor Authentication (MFA) using TOTP (Google Authenticator) or SMS. An Identity Provider (Auth0, Okta, or AWS Cognito) centralizes user management. JWT tokens are short-lived (15-minute access tokens, 7-day refresh tokens) and work with a revocation list stored in Redis, ensuring stolen tokens are revoked immediately.

The Data Protection Layer implements AES-256 encryption at rest for all database storage, with key management via AWS KMS or HashiCorp Vault. All network connections use TLS 1.3 with perfect forward secrecy. Database Activity Monitoring (DAM) provides real-time SQL query analysis, detecting and alerting on suspicious patterns like SQL injection attempts. Data Loss Prevention (DLP) scans outbound traffic to detect PII and financial data.

The Monitoring and Response Layer implements SIEM integration with Splunk, ELK Stack, or AWS Security Hub, including correlation rules that detect attack patterns across multiple log sources. Real-time alerting immediately notifies security teams upon detection of SQL injection, unauthorized access, or brute force attempts. Automated incident response triggers IP blocking, token revocation, and endpoint disabling upon threat detection.

Recommended Security Controls (with Justification)

The recommended security controls have been selected to address the identified vulnerabilities and reduce the risk of similar incidents by 80-90%.Adding authorization middleware to all API endpoints is critical to prevent IDOR vulnerabilities. The current API validates token validity but does not check if the user is authorized for the requested resource. This control would have prevented the unauthorized access to 15 accounts, as seen in the API logs.

Implementing parameterized queries for all SQL statements eliminates SQL injection vulnerabilities at the source. The current web application directly concatenates user input into SQL queries. This control would have prevented the successful SQL injection attack documented in the web logs.

Enforcing rate limiting at the API Gateway level prevents brute force and enumeration attacks. The current system allows unlimited authentication attempts. This control imposes a limit of 10 requests per minute per user and 100 requests per IP.

Mandatory Multi-Factor Authentication (MFA) prevents stolen credentials from being immediately usable. The current system relies solely on single-factor authentication. This control requires an Identity Provider solution (Auth0, Okta, or AWS Cognito).

Configuring SPF, DKIM, and DMARC records prevents email spoofing. The current email infrastructure lacked these records, allowing spoofed emails to reach users' inboxes. This control prevents the delivery of phishing emails.

SIEM integration provides real-time event correlation and analysis. The current system captures logs but cannot correlate events without SIEM integration. This control ensures that WAF alerts are processed automatically.

Defense-in-Depth Strategy

The six-layer defense-in-depth strategy ensures that a failure in any single layer does not lead to a full system compromise. Each layer provides independent protection while complementing the others.

The Perimeter layer: blocks known attack patterns and malicious IPs before they reach the application infrastructure.

Network segmentation: isolates critical systems and prevents lateral movement.

Application security: implements secure coding practices and authorization controls.

Identity management: ensures that only authenticated and authorized users can access resources.

Data protection:safeguards information even if the other layers are compromised.

Monitoring and response: provides visibility and automated containment.

This multi-layered approach addresses all identified vulnerabilities: WAF blocking prevents SQL injection, authorization middleware prevents IDOR, MFA prevents the exploitation of stolen credentials, rate limiting prevents brute force attacks, and SIEM correlation enables rapid detection and response.

Section 3: Response & Remediation

Immediate Actions (0-24 hours)

The attacker's IP address (203.0.113.45) and the brute force source IP (192.168.1.100) should be blocked at the firewall level. The WAF configuration must be changed from detect-only to blocking mode. According to WAF logs, the WAF detected SQL injection attempts but did not block them.A forensic investigation must be initiated immediately.Database logs should be analyzed to determine which data was accessed via the SQL injection attack. Web logs indicate a 156,789-byte data leak occurred. The 15 customer accounts must be notified in accordance with GDPR requirements. According to API logs, these accounts were subjected to unauthorized access via the IDOR vulnerability. The affected accounts should be locked, pending password resets and a security review.

Short-term Fixes (1-2 weeks)

The short-term remediation phase includes high-priority fixes that address critical security vulnerabilities and significantly improve the system's security posture.

Authorization middleware must be added to all API endpoints. According to API logs, the current API checks token validity but does not verify if the user is authorized for the resource they are trying to access. The middleware must enforce the critical check: if user_id != account_id: raise UnauthorizedError(). This single change will prevent all IDOR vulnerabilities.Rate limiting must be deployed at the API Gateway level. According to API logs, the system allowed unlimited authentication attempts. Rate limiting should enforce a limit of 10 requests per minute per user and 100 requests per IP.An input validation framework must be integrated into the web application. According to web logs, the current application directly concatenated user input into SQL queries. All user input must be sanitized before processing.SPF, DKIM, and DMARC records must be configured. According to email logs, the current email infrastructure lacked these records, allowing spoofed emails to reach users' inboxes. These email authentication mechanisms prevent domain spoofing and block the delivery of phishing emails.Account lockout policies must be implemented. According to API logs, the system permitted an unlimited number of failed authentication attempts. The account lockout policy should lock an account for 15 minutes after 3 failed authentication attempts.

Long-term Improvements (1-3 months)

The long-term improvement phase involves strategic security enhancements that transform the system's security architecture and provide comprehensive protection against future attacks.

A comprehensive API Gateway (Kong or AWS API Gateway) must be deployed. The current architecture lacks a central API Gateway, resulting in inconsistent security controls across endpoints. The API Gateway will serve as a single entry point for all API requests, enforcing centralized authentication, authorization, rate limiting, and request routing.Multi-factor authentication must be made mandatory for all users. The current system used only single-factor authentication. An Identity Provider solution (Auth0, Okta, or AWS Cognito) should be deployed, and MFA must be enforced.

All SQL queries must be converted to use parameterized queries via an ORM framework. According to web logs, the current application directly concatenated user input into SQL queries. A comprehensive code review should identify and remediate all instances of raw SQL query construction.SIEM integration must be implemented. The current system captures logs but cannot correlate events without SIEM integration. SIEM integration provides centralized log collection, correlation, and real-time alerting. Correlation rules should be configured to detect attack patterns such as SQL injection attempts, unauthorized access patterns, and brute force activities. Automated incident response playbooks should be developed to ensure containment as soon as threats are detected.Network segmentation must be implemented. The current architecture has a flat network structure where all components reside on the same network segment. DMZ, application, and database tiers must be segregated. Database systems should be moved to private subnets with no internet gateway.Database encryption at rest must be enabled. The current architecture does not demonstrate evidence of encryption at rest. AES-256 encryption should be enabled for all database storage. Key management should be handled via AWS KMS or HashiCorp Vault. Database Activity Monitoring (DAM) should be deployed.

Compliance Considerations

The incident has triggered multiple regulatory and compliance requirements.

GDPR Compliance: The incident triggers the GDPR Article 33 breach notification requirements. Supervisory authorities must be informed within 72 hours of the incident's detection. Affected data subjects must be notified without undue delay if the breach is likely to result in a high risk to their rights and freedoms. The remediation efforts must be compliant with GDPR Article 32 (security of processing).

PCI-DSS Compliance: If payment card data was accessed, an immediate PCI-DSS assessment is required. The Payment Card Industry Security Standards Council must be informed, and a forensic investigation by a PCI-DSS approved forensic auditor may be necessary.

ISO 27001 Alignment: The remediation efforts must be aligned with the ISO 27001:2022 controls. Specifically, the A.9.2 (User Access Management), A.12.6 (Technical Vulnerability Management), and A.12.4 (Logging and Monitoring) controls must be addressed.

Regulatory Reporting: Depending on the jurisdiction and the nature of the data accessed, financial services regulators may require incident reporting.