Acme Financial Services Incident Response Report

On October 15, 2024, a coordinated and multi-stage attack was carried out against Acme Financial Services' trading platform. The attack began with a phishing campaign, leading to the theft of user credentials, and subsequently exploited SQL Injection (SQLi) and Insecure Direct Object Reference (IDOR) vulnerabilities.

The success of the attack stemmed from critical security vulnerabilities in the existing architecture (particularly in the WAF, Email Gateway, and Trading API). The main result of the incident was a high-volume data leak.

The incident started at **09:00:23 on October 15, 2024**, and does not conflict with any scheduled security tests. The attacker's IP address is **203.0.113.45**.

```
2024-10-15 09:00:23,security@acme-finance.com,user1@acme.com,URGENT: Verify Your Account - Action Required,yes,203.0.113.45,
2024-10-15 09:00:25,security@acme-finance.com,user2@acme.com,URGENT: Verify Your Account - Action Required,no,,
2024-10-15 09:00:27,security@acme-finance.com,user3@acme.com,URGENT: Verify Your Account - Action Required,yes,203.0.113.45,
2024-10-15 09:00:29,security@acme-finance.com,user4@acme.com,URGENT: Verify Your Account - Action Required,no,,
2024-10-15 09:00:31,security@acme-finance.com,user5@acme.com,URGENT: Verify Your Account - Action Required,yes,203.0.113.45,
2024-10-15 09:00:33,security@acme-finance.com,user6@acme.com,URGENT: Verify Your Account - Action Required,no,,
```

Sender = security@acme-finance.com

This resembles a corporate email address. The normal corporate domain should be **acme.com** or **acme-financial.com**.

Subject= URGENT: Verify Your Account - Action Required

A classic phishing tactic that creates urgency and fear/threat of loss to pressure the user into taking immediate action.

According to the logs provided above, three users clicked on the email. This indicates that the campaign was partially successful.

```
2024-10-15 09:18:30,1523,/login,,200,3421,203.0.113.45,Mozilla/5.0 (Windows NT 10.0
```

According to the log record above, the attacker successfully logged in using the user account with **id=1523.**

Causes of vulnerability and possible solutions

The Email Gateway component in Architecture is marked as Partially Secure (Orange). This explains why the phishing email reached users' inboxes:

Vulnerability 1: Weak Domain Control (Lack of SPF/DKIM/DMARC): The gateway failed to detect or block a spoofed domain name such as acme-finance.com (instead of acme-financial.com). To prevent spoofing (fake sender), email domain policies (especially for similar domains such as acme-finance.com) should be strengthened.

Vulnerability 2: Weak Integration with WAF: Although clicking the link resulted in traffic going to the WAF, the WAF only DETECTED it but did not BLOCK the traffic. This made it easier for the attack to proceed to the second stage.

```
2024-10-15 09:00:23,950107,HIGH,DETECT,203.0.113.45,/verify-account.php,Suspicious Link Pattern,no
```

Multi-Factor Authentication (MFA) Requirement: MFA must be mandatory for all employee accounts. Even if credentials are compromised, access to the system is blocked.

Mandatory and Regular Simulations: Phishing awareness training and simulations should be intensified. Employees should be trained to recognize emails that create a sense of urgency.

A recommendation for an advanced email security solution: Microsoft Defender for Office 365 : It checks the URL at the moment the phishing link is clicked and blocks the threat without going to the attacker's IP. It is particularly known for its Link Rewriting, Time-of-Click protection, and Impersonation Protection features.

Analysis of SQL Injection

The attack began immediately after the attacker successfully logged into the system using credentials obtained through phishing.

Attacker IP: 203.0.113.45.

Target Endpoint: /dashboard/search.

Time Range: From 09:20:30 to 09:23:45.

After logging into the system, the attacker made a series of attempts to access the database. These attempts demonstrated the first layer of defense of the WAF (Web Application Firewall), but the weakness of the WAF was revealed:

```
2024-10-15 09:20:30,981173,HIGH,DETECT,203.0.113.45,/dashboard/search,SQL Injection Attempt - OR 1=1,yes
2024-10-15 09:21:15,981318,CRITICAL,BLOCK,203.0.113.45,/dashboard/search,SQL Injection - DROP TABLE,yes
2024-10-15 09:22:00,981257,HIGH,BLOCK,203.0.113.45,/dashboard/search,SQL Injection - UNION SELECT,yes
2024-10-15 09:23:45,981001,MEDIUM,DETECT,203.0.113.45,/dashboard/search,Suspicious SQL Pattern,no
```

```
2024-10-15 09:20:30,1523,/dashboard/search,ticker=AAPL' OR 1=1--,403,567,203.0.113.45,Mozilla/5.0 (Windows NT 10.0
2024-10-15 09:21:15,1523,/dashboard/search,ticker=AAPL'
2024-10-15 09:22:00,1523,/dashboard/search,ticker=AAPL' UNION SELECT * FROM users--,403,567,203.0.113.45,Mozilla/5.0 (Windows NT 10.0
2024-10-15 09:23:45,1523,/dashboard/search,ticker=AAPL' /*!50000OR*/ 1=1--,200,156789,203.0.113.45,Mozilla/5.0 (Windows NT 10.0
```

Time=> 09:20:30

Attack Payload=> ticker=AAPL' OR 1=1--

WAF Status => DETECT, HIGH, NO BLOCK

Response Code => 403 Forbidden

Analysis => WAF detected the simple payload (rule_id: 981173), and the Web Application enforced the rule and terminated the connection (403).

Time => 09:21:15

Attack Payload => ticker=AAPL'; DROP TABLE users--

WAF Status => BLOCK (Blocked), CRITICAL (Critical)

Response Code => 403 Forbidden

Analysis=> A critical command was attempted (rule_id: 981318). WAF blocked this attempt.

Time => 09:22:00

Attack Payload => ticker=AAPL' UNION SELECT * FROM users--

WAF Status => BLOCK (Blocked), HIGH (Severity)

Response Code => 403 Forbidden

Analysis => The UNION SELECT command was also blocked by the WAF (rule_id: 981257).

Time=> 09:23:45

Attack Payload=> ticker=AAPL' /*!50000OR*/ 1=1--

WAF Status => DETECT, MEDIUM, NO BLOCK

Response Code => 200 OK

Analysis => WAF bypassed! The /*!50000OR*/ part is a MySQL comment line technique used to bypass some WAFs.

```
2024-10-15 09:23:45,1523,/dashboard/search,ticker=AAPL' /*!50000OR*/ 1=1--,200,156789,203.0.113.45,Mozilla/5.0 (Windows NT 10.0
```

Response Size= 156,789 bytes.

First Evidence of Data Leakage. A large response size indicates that the SQL query was successful and that more (sensitive) data than usual may have been retrieved.

```
2024-10-15 09:24:10,1523,/dashboard/export,format=csv,200,892341,203.0.113.45,Mozilla/5.0 (Windows NT 10.0
```

Time=> 09:24:10

Endpoint=> /dashboard/export?format=csv

Response Code=> 200 OK

Response Size=> 892,341 bytes

Analysis=> Data retrieved via SQLi has been successfully exported from the system as a large CSV file. This is the final outcome of the incident.

Causes of vulnerability and possible solutions

The root cause of this attack is vulnerabilities identified in two layers in our architectural analysis:

1. Weak WAF Protection: The WAF only uses "Basic Rules," leaving it vulnerable to evasion techniques such as MySQL version comments.

2. Insecure Application Code: A Critical Vulnerability (Red) exists in the Web App (Python) component. The application directly appends user input to database queries (allowing SQL injection) and does not use parameterized queries.

Intrusion Detection/Prevention Systems (IDS/IPS): IDS/IPS should be deployed at the network level to detect unusual traffic patterns (e.g., SQLi scans, high error rates) that bypass the WAF. IPS blocks known SQLi attempts and unusual traffic anomalies even where the WAF cannot detect them.

Advanced Rule Sets: Rule sets targeting comment lines (/*!...*/) and other code obfuscation techniques should be enabled immediately.

Least Privilege Principle: The permissions of the database user used by the web application should be limited to only the tables it needs. It should be prevented from executing critical commands such as DROP TABLE.

Analysis of Mobile API Broken Access Control

This vulnerability occurred when the attacker accessed sensitive data belonging to other users, different from the resources they were authorized to access with their own valid session token.

The attacker exploited this vulnerability using the token of the **user_id: 1523** account, which they had obtained through phishing.

• **Attacker IP: 203.0.113.45.**

The following API logs show successful access attempts made by **user_id: 1523** to other accounts that were not their own:

```
2024-10-15 06:46:30,1523,/api/v1/portfolio/1523,GET,1523,200,156,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:15,1523,/api/v1/portfolio/1524,GET,1524,200,143,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:18,1523,/api/v1/portfolio/1525,GET,1525,200,138,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:21,1523,/api/v1/portfolio/1526,GET,1526,200,147,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:24,1523,/api/v1/portfolio/1527,GET,1527,200,141,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:27,1523,/api/v1/portfolio/1528,GET,1528,200,139,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:30,1523,/api/v1/portfolio/1529,GET,1529,200,144,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:33,1523,/api/v1/portfolio/1530,GET,1530,200,142,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:36,1523,/api/v1/portfolio/1531,GET,1531,200,148,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:39,1523,/api/v1/portfolio/1532,GET,1532,200,145,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:42,1523,/api/v1/portfolio/1533,GET,1533,200,140,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:45,1523,/api/v1/portfolio/1534,GET,1534,200,146,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:48,1523,/api/v1/portfolio/1535,GET,1535,200,143,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:51,1523,/api/v1/portfolio/1536,GET,1536,200,149,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:54,1523,/api/v1/portfolio/1537,GET,1537,200,141,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
2024-10-15 06:47:57,1523,/api/v1/portfolio/1538,GET,1538,200,147,203.0.113.45,Acme-Mobile-Android/3.2.0,jwt_token_1523_stolen
```

Time => 06:46:30

User ID => 1523

Access Attempt => 1523 (Own Account)

Endpoint => /api/v1/portfolio/1523

Response Code => 200 OK

Analysis => Successful. Normal and expected access.

Time => 06:47:15

User ID => 1523

Access Attempt => 1524 (Another User's Account)

Endpoint => /api/v1/portfolio/1524

Response Code => 200 OK

Analysis => IDOR Successful! The attacker successfully accessed another account's portfolio data using their own token.

Time => 06:47:18

User ID => 1523

Access Attempt => 1525 (Another User's Account)

Endpoint => /api/v1/portfolio/1525

Response Code => 200 OK

Analysis => IDOR Successful Again! This indicates that the attack was targeted and aimed at collecting data on a large scale.

In this way, the attacker successfully accessed the accounts of users with IDs between 1523 and 1538.

Causes of vulnerability and possible solutions

• The Trading API (Flask) component is marked as a Critical Vulnerability (Red) in the architecture.

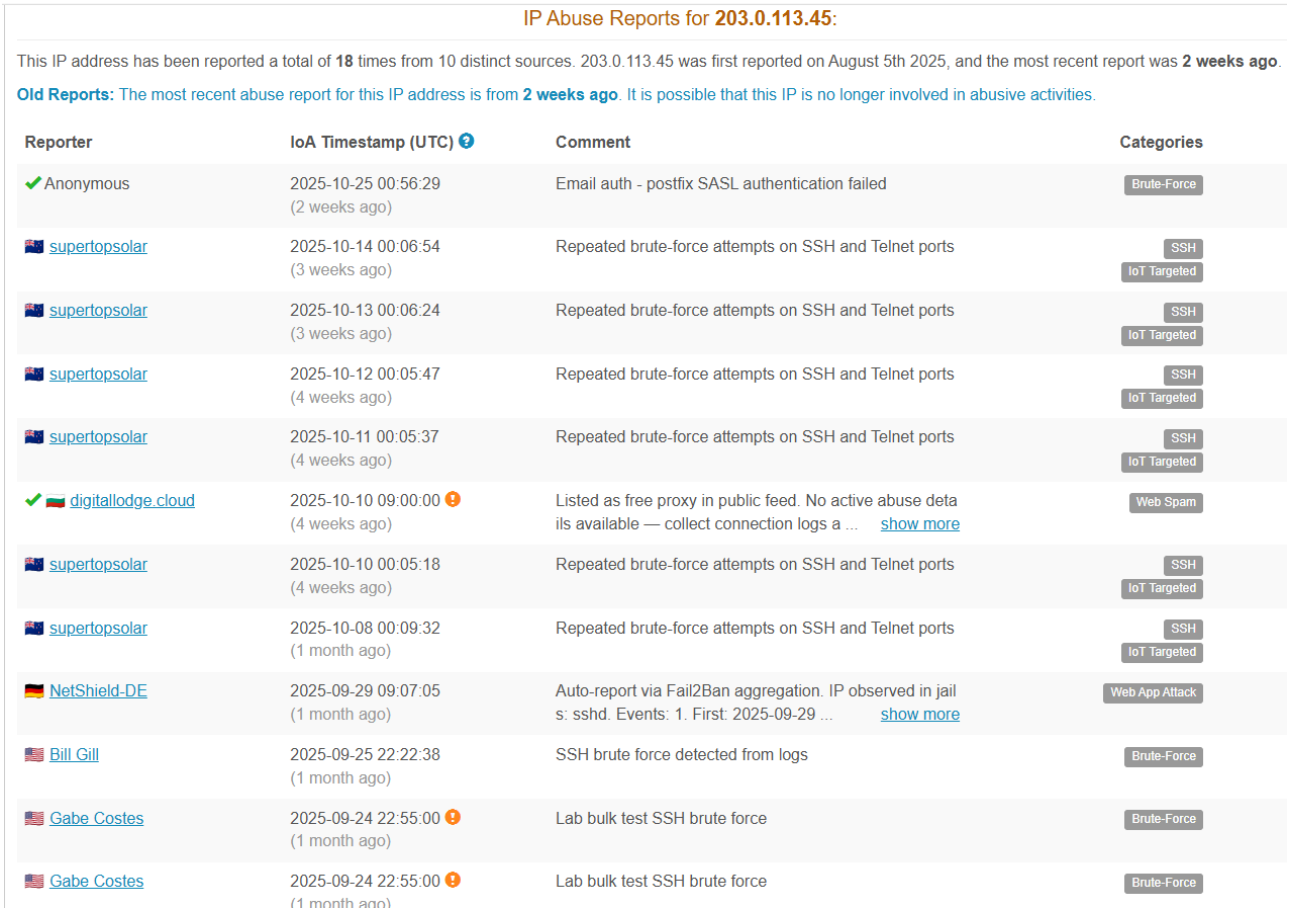
• The cause of the vulnerability is clearly stated in the API documentation: Authorization checks only verify the token's validity but “may not verify account ownership.” (3.1 Portfolio Management)

• This indicates an IDOR (Insecure Direct Object Reference) vulnerability, which is listed in the OWASP Top 10. The object reference used in the API ({account_id}) allows direct access without user authorization.

Mandatory Account Ownership Check: For all Portfolio (/portfolio/{account_id}) and Transaction endpoints, before processing the request, the user_id extracted from the JWT token must be checked to see if it matches the {account_id} requested in the URL path.

Implementing Speed Limits in the API Layer: IDOR attacks typically attempt to retrieve data at high speeds. User-based speed limits for portfolio endpoints (/portfolio) (the 60/minute limit specified in the API documentation) must be enforced in the API Gateway.

Important Notes



IP Abuse Reports for **203.0.113.45**:

This IP address has been reported a total of **18** times from 10 distinct sources. 203.0.113.45 was first reported on August 5th 2025, and the most recent report was **2 weeks ago**.

Old Reports: The most recent abuse report for this IP address is from **2 weeks ago**. It is possible that this IP is no longer involved in abusive activities.

Reporter	IoA Timestamp (UTC)	Comment	Categories
Anonymous	2025-10-25 00:56:29 (2 weeks ago)	Email auth - postfix SASL authentication failed	Brute-Force
supertopsolar	2025-10-14 00:06:54 (3 weeks ago)	Repeated brute-force attempts on SSH and Telnet ports	SSH, IoT Targeted
supertopsolar	2025-10-13 00:06:24 (3 weeks ago)	Repeated brute-force attempts on SSH and Telnet ports	SSH, IoT Targeted
supertopsolar	2025-10-12 00:05:47 (4 weeks ago)	Repeated brute-force attempts on SSH and Telnet ports	SSH, IoT Targeted
supertopsolar	2025-10-11 00:05:37 (4 weeks ago)	Repeated brute-force attempts on SSH and Telnet ports	SSH, IoT Targeted
digitallodge.cloud	2025-10-10 09:00:00 (4 weeks ago)	Listed as free proxy in public feed. No active abuse deta ils available — collect connection logs a ... show more	Web Spam
supertopsolar	2025-10-10 00:05:18 (4 weeks ago)	Repeated brute-force attempts on SSH and Telnet ports	SSH, IoT Targeted
supertopsolar	2025-10-08 00:09:32 (1 month ago)	Repeated brute-force attempts on SSH and Telnet ports	SSH, IoT Targeted
NetShield-DE	2025-09-29 09:07:05 (1 month ago)	Auto-report via Fail2Ban aggregation. IP observed in jail s: sshd. Events: 1. First: 2025-09-29 ... show more	Web App Attack
Bill Gill	2025-09-25 22:22:38 (1 month ago)	SSH brute force detected from logs	Brute-Force
Gabe Costes	2025-09-24 22:55:00 (1 month ago)	Lab bulk test SSH brute force	Brute-Force
Gabe Costes	2025-09-24 22:55:00 (1 month ago)	Lab bulk test SSH brute force	Brute-Force

To prove the seriousness and malicious nature of the incident, the source IP address (203.0.113.45) used in all analyzed attack stages (Phishing, SQLi, IDOR) was checked on the AbuseIPDB platform.

This IP address has a high reputation score (Abuse Score) in AbuseIPDB and has been reported in the past for various malicious activities (phishing, botnet, or attack attempts).

Critical Conflict and Its Importance

This finding confirms the malicious nature of the incident by refuting a detail in the Security Test Schedule document:

Response Codes (Vulnerability/Critical): The attacker's IP address (203.0.113.45) is within the Approved IP Range (203.0.113.0/24) for the Quarterly Penetration Test (Test 2).

Attacker's Motive: This indicates that the attacker attempted to deceive the defense teams (SOC Team) by appearing to conduct a legitimate penetration test. The Test Schedule specifically requested that the SOC team not block traffic from this IP.

Final Conclusion: The IP's notorious reputation in threat intelligence, combined with its destructive and exploitative actions in the logs (data leakage via SQLi and IDOR), conclusively proves that this traffic was not an authorized test but a malicious attack.

Additional Note

A difference of approximately 3 hours has been observed between the timestamps in the API and Web Logs. This is due to the logs having different time zone settings (e.g., UTC and UTC-3). To preserve the logical sequence of events, it has been assumed that the IDOR attack, which requires a valid session token, occurred after the successful login (Web Logs: 09:18:30). This highlights the need for mandatory and synchronized UTC usage for all log sources in architectural improvements.