Executive Summary

On October 15, 2024, a security incident was detected within Acme Financial Services' systems.
The incident originated from a fake *"account security"* email used for a phishing attack.
After the user clicked the malicious link, the attacker obtained valid session credentials, gained access to the system, and exploited vulnerabilities in API authorization to access multiple accounts.

Shortly afterward, weaknesses in the WAF configuration were abused to perform an SQL injection, and finally, the attacker used the dashboard's "export" function to extract a large amount of data.

The event demonstrated how multiple small gaps—email security, API authorization, query validation, and WAF configuration—can combine into a serious data-breach scenario.

Incident Timeline

Time	Event	Source	IP Address	Finding
09:00	User clicked the phishing link	Email logs	203.0.113.45	Phishing attempt confirmed
09:18	Successfull login from same IP	Web/API logs	203.0.113.45	user_id=1523 session established
06:47	Consecutive API access attempts	WAF logs	203.0.113.45	"Rapid Sequential Access" alert
09:20-09:23	SQL Injection attempts	Web/API logs	203.0.113.45	WAF detected but did not block
09:24	CSV export activity	Web logs	203.0.113.45	~900 KB of data exfiltrated

Technical Analysis

Phishing and Session Compromise

Email records show multiple link clicks from the same IP address.
These links redirected users to a domain controlled by the attacker.
Minutes later, a successful login occurred on the same IP under user_id = 1523.
Because the system treated the login as legitimate, the attacker gained full access to the user interface.

Unauthorized API Access (BOLA)

WAF logs indicate consecutive requests to different portfolio IDs from the same IP address.
The activity triggered *"Rapid Sequential Access"* detections but was not blocked.
This shows that while token validation was in place, the system failed to verify whether the accessed data actually belonged to the authenticated user.

SQL Injection and WAF Bypass

The attacker performed SQL injection attempts through a search-parameter field.
The WAF successfully blocked standard payloads but only *detected* an obfuscated payload (/*!50000OR*/ 1=1--), which produced an unusually large response from the database — clear evidence that the injection succeeded.

Data Exfiltration

Immediately after the successful injection, the attacker used the dashboard's "export" function to download data.
The export file was approximately 900 KB in size, and there were no size or approval restrictions configured for that function.
As a result, sensitive data was extracted without any additional authorization.

Containment Plan

Once the incident is identified, the immediate objective is to stop the attacker's connection, prevent further data leakage, and preserve forensic evidence.

1. **Revoke all active sessions and tokens** associated with user_id = 1523 and any linked accounts.
2. **Block source IP 203.0.113.45** at both WAF and perimeter-firewall levels.
3. **Temporarily disable** the /dashboard/export endpoint to halt ongoing data exports.
4. **Change WAF rule 981001** from detect-only to block mode and review all detection-only signatures.
5. **Collect and secure logs** (WAF, web, API, email) using SHA-256 hashing to preserve integrity.
6. **Coordinate internal communication** between SOC, application, network, and legal/compliance teams under the supervision of the CISO.

These actions aim to isolate the attacker, contain the breach, and ensure all forensic data remains intact.

Long-Term Remediation and Security Hardening

Following containment, long-term measures should focus on preventing recurrence and strengthening the security posture across all layers of the environment.

- Convert all SQL queries to **parameterized statements** and enforce strict input validation.
- Implement **ownership verification** on all sensitive API endpoints to ensure data access aligns with the authenticated user.
- Review WAF configuration: move detection-only signatures to **block mode**, and update rules to detect obfuscated SQLi patterns.
- Apply **size and approval limits** to data-export functions, and integrate them with **DLP controls** for continuous monitoring.
- Enforce **multi-factor authentication (MFA)** and adopt **short-lived tokens** tied to device/IP context.
- Conduct **regular phishing simulations** and employee awareness training.
- In the long term, adopt a **Zero Trust architecture**, implement **mTLS** for service-to-service traffic, and integrate automated security testing (SAST, DAST, API fuzzing) into CI/CD pipelines.

Conclusion

This incident clearly illustrates how a single phishing email can escalate into a broader data-exfiltration scenario when multiple security layers fail simultaneously.
The compromise originated from human error but was amplified by insufficient API authorization, weak SQL input controls, and a misconfigured WAF policy.

By applying the remediation steps outlined in this report, Acme Financial Services can significantly reduce exposure to similar multi-vector attacks and improve its overall resilience.
A secure infrastructure requires not only robust technical defenses but also continuous awareness, disciplined processes, and proactive testing.