Incident Report

This report details a security incident that occurred on October 15, 2024, involving a coordinated attack on the Acme platform. The attack compromised the web application and mobile API, leading to data exfiltration and unauthorized access to user accounts. This report provides a detailed analysis of the attack, identifies the vulnerabilities that were exploited, and recommends architectural improvements to prevent similar incidents in the future.

Incident Analysis

Web Log Analysis

At 2024-10-15 06:20:30 the user 1523 attempted to web server with source IP 203.0.113.45 several times and 3 of them forbidden because of SQLi attempts. Obviously the user 1523 has been tried SQLi several times and at 2024-10-15 06:23:45 he/she carried out success.

WAF Logs Analysis

At 2024-10-15 06:20:30/06:23:45 suspicious SQLi attack which is coming from 203.0.113.45 IP address was not able to blocked. This timestamp same the web access suspicious SQL pattern and WAF was not able to blocked successful SQLi attempt. At 2024-10-15 06:00:23 the suspicious link pattern detected coming from 203.0.113.45 IP address this may sign of endpoint exploitation. At 2024-10-15 06:47:30 porfolio endpoint has been forced by 203.0.113.45 IP address several times this is a sign of IDOR attack.

Email Logs Analysis

At this analysis focused clicked links. There are 3 clicked mails between 10 mails and all of them is associated with the suspicious IP address which is mentioned above 203.0.113.45. The sender mail address is not different for 3 events and clicking events carried out at 2024-10-15 06:00:23/27/31 timestampt. Once examined mail subjects three of them are suspicious because contain immediate action and forcing user for the iteraction.

API Logs Analysis

According the investigation at 2024-10-15 06:46:30 a user with user_id 1523 IP address 203.0.113.45 started making a series of requests to the /api/v1/portfolio/{account_id} endpoint. As seen in the logs user was able to access the portfolies of the other users by changing account_id via url. Before all malicious activities not carried out, the attacker's successful first-try POST to "/api/v1/login" at 2024-10-15 03:45:10 was possible because user 1523 credentials obtained already. So that other /portfolio/{account_id} obtained same session_token named jwt_token_stolen_1523.

Correlation

2024-10-15 06:00:23 attacker sends phishing email with the subject "URGENT: Verify Your Account - Action Required" to multiple employees and 3 users clicked link and first click start attack. At the same time 2024-10-15 06:00:23 WAF detects Suspicious Link Pattern(verif-account.php) and IP address 203.0.113.45 and that is sign of directing user to malicious site for credentials theft. At the 2024-10-15 06:23:45 WAF detects Suspicious SQL Pattern and at the same time WEB logs records successful response for the " ticker" this time SQLi carried

out. After that at the 2024-10-15 06:24:10 attacker export download data as CSV format and 892341 size. After that at the 2024-10-15 06:45:10 attacker tried mobie app API for more access via IDOR. Attacker login with the user_id 1523 with the user_agent Acme-Mobile-Android/3.2.0 and same IP address. After that at the 2024-10-15 06:46:30 starts IDOR on "/api/v1/portfolio/{user_id}" enpoint and all of them successful.

Correlation flow

Note: the timestamps were normalized to UTC with the assumption that the API logs were in a different time zone

2024-10-15 06:00:23 (UTC)

Event: User clicks mail and access malicious site. The attacker steals the credentials of user_id 1523 via phishing email.

Evidence: email.logs(URGENT: Verify Your Account - Action Required),waf_logs(/verify-account.php, IP:203.0.113.45)

2024-10-15 06:18:30 (UTC)

Event: The attacker logs into the web application with the stolen credentials.

Evidence: web_logs.csv(login)

2024-10-15 06:23:45(UTC)

Event: The attacker successfully bypasses the WAF and performs a successful SQL injection.

Evidence: web_logs.csv(ticker=AAPL' /*!50000OR*/ 1=1-), waf_logs.csv(Suspicious SQL Pattern)

2024-10-15 06:24:10 (UTC)

Event: After the successful SQL injection, the attacker exports the data from the database (892341 bytes) in CSV format.

Evidence: web_logs.csv(format=csv)

2024-10-15 06:45:10(UTC -3)

Event: After the web attack, the attacker turns to the mobile API and logs into the API with the same stolen credentials.

Evidence: api_logs.csv(/api/v1/login)

2024-10-15 06:47:15(UTC -3)

Event: The attacker begins to exploit the IDOR vulnerability in the API to access the accounts of other users. Their goal is to collect more data or to be able to perform other actions(make financial transactions).

Evidence: api_logs.csv(/api/v1/portfolio/15xx)

Attack Vector Identification

Phishing: The attacker used a spearphishing campaign to steal the credentials of an employee

SQL Injection (SQLi): The attacker exploited a SQL injection vulnerability in the web application to exfiltrate data from the database.

Insecure Direct Object Reference (IDOR): The attacker exploited a broken access control vulnerability in the mobile API to access the data of other users

MITRE ATT&CK-Owasp

Initial Access: T1566.002 – Spearphishing, Link Execution: T1059.007 - SQL Injection

Credential Access: T1555 - Credentials from Password Stores

Discovery: T1087.004 - Account Discovery, Exfiltration: T1567.002 - Exfiltration to Cloud Storage

A02:2021 - Cryptographic Failures(Indirectly), A01:2021 - Broken Access Control A03:2021 - Injection

Root Cause

The root cause of the incident was the “successful phishing attack”. There were multiple technical vulnerabilities in the system but attacker cannot exploit the system without credentials. So that the phishing attack was the initial attack.

Impact

Data Breach: A large amount of sensitive user data was exfiltrated from the database(892341 byte).

Reputational Damage: The incident could damage the reputation of Acme Financial Services and lead to a loss of customer trust.

Regulatory Compliance: The incident may have violated data protection regulations such as GDPR or CCPA, which could result in significant fines.

Financial Loss: The company could face financial losses due to regulatory fines, legal fees, and the cost of remediation.

Architecture Review

Current Architecture Weaknesses

After the review and some of weaknesses observed and listed below.

Human: All of the events started with only clicking suspicious link. So that absence of employee awareness is the first reason at this point.

Weak Input Validation: The web application did not properly validate user input, which is one of the most basic security vulnerabilities.

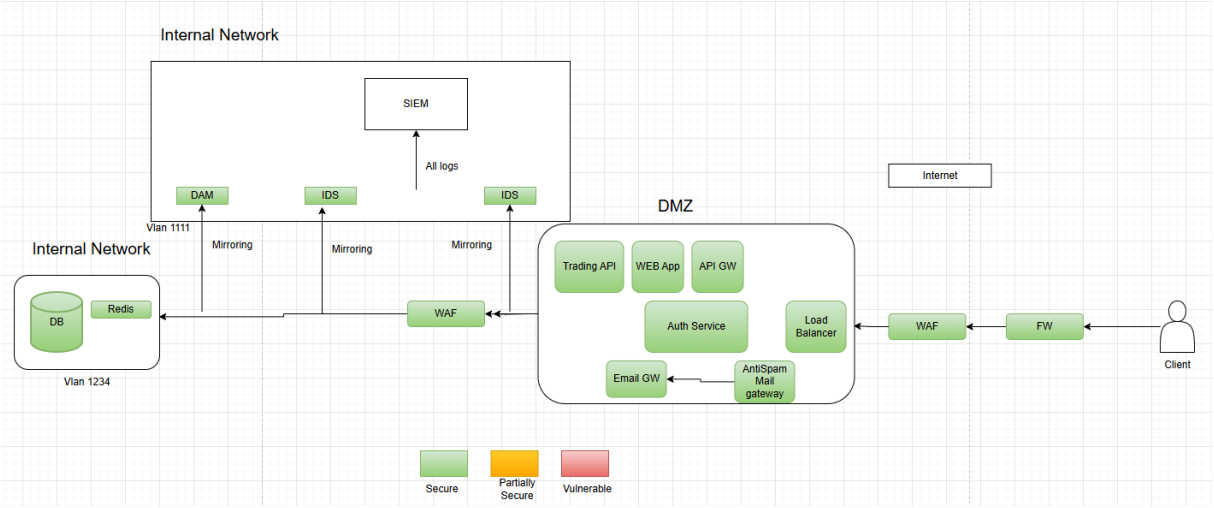
WAF Configuration: The WAF was not configured to block advanced SQL injection techniques It only attempted to detect and block some known patterns.

API: The mobile API did not properly check whether a user could access another user's data.

Lack of MFA: The absence of MFA meant that a password stolen through a phishing attack was sufficient to log into the system.

Absence of Logging and Monitoring: There is no any logging technology for the real time monitoring and taking action immediately.

Improved Security Architecture Diagram



Recommended Security Controls

Preventative layer solutions

Security Awareness Training: Educates employees against social engineering attacks. In addition using phishing simulation tools(Gophish)

MFA: Prevents unauthorized access even if credentials are stolen, neutralizing the impact of phishing attacks.

Parameterized Queries: The most effective way to prevent SQL injection.

Strict Access Control on the API: The user only access their own data.

Updated WAF rules: Rules should be updated to block advanced attack techniques such as WAF bypass.

EDR/AV: Prevents malicious software from running on servers.

Network Segmentation: Creating DMZ to gather crucial assets.

Detection Layer

SIEM: Collects and correlates logs from all assets to early detection.

AV/EDR: Detects abnormal activities such as file changes or suspicious process.

IDS/IPS: To monitor network traffic to detect attack signature. It can detect WAF-bypassing or abnormal API access patterns(such IDOR).

Defense-in-depth strategy

Network Architecture and Segmentation: A DMZ network should be established between the internet-facing services (Web App, Trading API, Auth Service) and the internal network that sensitive data (PostgreSQL, Redis).

Monitoring and Detection Layer: SIEM system should be established that collects logs from all systems and generates real-time alerts. IDS/IPS should monitor network traffic and server activities to detect suspicious activities that bypass the WAF or occur on the internal network.

Security Awareness Training: Employees should be regularly trained to recognize and report social engineering attacks such as phishing.

Response & Remediation

Immediate actions

Containment: Immediately reset the passwords for user_id 1523 and the other users which are clicked link. Block the attacker's IP address (203.0.113.45) at the firewall. Temporarily disable the /dashboard/export endpoint and the vulnerable API endpoints until they can be patched.

Investiagtion: Create a backup of all relevant logs (web, api, waf, email) for forensic analysis. Begin a full investigation to determine exactly what data was exfiltrated and which user accounts were accessed.

Communication: Notifying Immediately notify the legal, compliance, and executive teams. The data breaches should be published for the people.

Short-term Fixes

Vulnerability Remediation: Patch the SQL injection vulnerability by implementing parameterized queries and input validation.

Update WAF: Update the WAF rules to block the specific SQL injection bypass technique that was used in the attack.

Long-term Improvements

Implemet MFA for all users. Implement SIEM for real-time centralized logging and monitoring system. Implement a regular, mandatory security awareness training program for all employees may be Gophish could assist that.

Compliance Considerations

GDPR/CCPA: The company must notify the relevant data protection authorities and the affected individuals within the required timeframe.

PCI DSS: If the company stores credit card data, this incident could be a violation of PCI DSS requirements, which could result in fines and other penalties.