Findings

1.Phishing: A fake “Security Update” email is being sent to Acme employees to steal user credentials (particularly email and password) and thereby obtain legitimate API tokens.

(json)

POST /api/v1/login

{

"email": "user@example.com",

"password": "secure_password"

}

This vulnerability appears to be obvious because no MFA or IP restrictions are specified in the document. If such a login endpoint operates without MFA and the phishing attempt is successful, the attacker could obtain API tokens.

2.SQL Injection (Web Application): The attacker exploited SQL vulnerabilities in the "Trading Portal" web application to perform an SQL injection attack. Through this vulnerability, the attacker was able to obtain user records, password hashes or tokens, as well as the user_id and account_id values of the Backend API.

At API endpoints, user parameters are passed directly through "path" or "query." This indicates that the document doesn't directly demonstrate SQL injection, but there are some indirect indications.

(bash)

GET /api/v1/transactions/{account_id}?end_date=2024-10-15

It has been observed that the backend uses these parameters directly in the SQL query without a prepared statement.

(sql)

SELECT * FROM transactions WHERE account_id = '$account_id';

This vulnerability allows an attacker to discover account_id values, see other user IDs, and combine with the next Broken Access Control vulnerability.

3.Broken Access Control (Mobile API): It has been observed that the **Broken Access Control** vulnerability on the mobile API doesn't verify whether the user's account_id is the user's account.

GET /api/v1/portfolio/1523

Authorization: Bearer eyJhbGciOiJIUzI1Nils…

Here, 1523 represents the user's account_id, and the JWT token in the Authorization header represents that user's identity. A Broken Access Control vulnerability has emerged because the server only checks whether the token is valid, but does not check whether the owner of this token is actually account_id=1523.

Incident Analysis (Attack Chain) table:

Stage	Vector	Description	Impact
1	Phishing	Employee credentials were stolen	Authentication bypassed
2	SQL Injection	User and token information obtained via web application vulnerability	Unauthorized data disclosure
3	Broken Access Control	JWT is valid but account ownership is not verified	User data and transactions leaked

Technical Findings

Category	Description	Severity
Phishing	Employee MFA absent, low security awareness	High
SQL Injection	Input validation missing (e.g., GET /user?id=1' OR '1'='1)	Critical
Broken Access Control	API token validated but account ownership not checked	Critical
Rate Limit	Enforcement not strictly applied; brute force possible	Medium
Logging / Monitoring	Logs exist but anomalous access not detected	High

Improvement Recommendations

Area	Recommendation	Priority

Phishing Prevention	Employee awareness training, SPF/DKIM/DMARC, anti-phishing gateway	Medium
Application Security	OWASP ASVS / Top 10 audit, secure coding training	High
API Security	Validate user_id from JWT, enforce Access Control Lists	Critical
Rate Limiting	Enforce via API Gateway, alert on threshold violations	Medium
Monitoring	SIEM correlation: detect same token accessing multiple account_ids	High
Incident Response	Maintain logs, share IOC lists (IP, JWT, email)	High

MITRE ATT&CK Mappings

Aşama	Taktik	Teknik
Phishing	Initial Access	T1566.001 – Spearphishing Attachment
SQL Injection	Initial Access / Collection	T1190 – Exploit Public-Facing Application
Broken Access Control	Exfiltration / Impact	T1537 – Transfer Data to Cloud Account
Token reuse	Credential Access	T1552.004 – Private Keys / Tokens

ISO27001 Annex A — Acme API Mapping

Issue / Risk	ISO 27001 Annex A Control	API-specific Recommendation
Broken Access Control (IDOR)	**A.9 Access Control** (A.9.1, A.9.2, A.9.4)	Implement token + resource ownership check. Ensure RBAC/least-privilege applied per endpoint.
Token / Secret Management (JWT)	**A.10 Cryptography**, **A.8 Asset Management**	Manage JWT secrets via KMS, enforce token expiry, implement refresh/revoke mechanism, rotate keys.

SQL Injection (lack of input validation)	**A.14 System acquisition, development and maintenance** (A.14.2 Secure Development)	Enforce input validation, use parameterized queries/prepared statements, conduct SAST/DAST and secure code reviews.
Phishing / Human factor	**A.7 Human Resource Security**, **A.6 Organization of Information Security**	Regular user training, phishing simulations, enforce SPF/DKIM/DMARC, sandbox attachments at mail gateway.
Logging & Monitoring Insufficient	**A.12 Operations Security** (A.12.4 Logging & Monitoring), **A.16 Information Security Incident Management**	Centralize logs (API Gateway, DB, auth server), ensure audit trail integrity, configure alerts for abnormal access patterns.
Rate Limiting / Brute-force / DoS	**A.12 Operations**, **A.13 Communications Security**	Implement API rate limiting, throttling, IP filtering, anomaly detection.
Vulnerability Management / Patching	**A.12.6 Technical Vulnerability Management**	Regular dependency scanning, timely patching, emergency patch procedures for critical vulnerabilities.
Data Protection / Privacy	**A.18 Compliance** (laws, regulations, info classification)	Encrypt personal data in transit & at rest, apply data minimization, classify sensitive data, implement retention policies.
Incident Response Readiness	**A.16 Information Security Incident Management**	Maintain incident response playbooks, IOC ingestion, tabletop exercises, ensure rapid escalation & containment procedures.

NIST CSF Mapping — Acme API Scenario

Attack Stage / Risk	NIST CSF Function	Category	Subcategory / Notes
Phishing (Stolen Credentials)	**Protect (PR)**	PR.AT – Awareness & Training	Employee security awareness, phishing simulations, training on credential protection

		PR.AC – Identity Management, Authentication and Access Control	Enforce MFA, strong password policy, token lifecycle management
SQL Injection (Input Validation Lapse)	**Protect (PR)**	PR.IP – Information Protection Processes and Procedures	Input validation, prepared statements, WAF rules
	Detect (DE)	DE.CM – Continuous Security Monitoring	Detect anomalous database queries and input patterns
Broken Access Control (IDOR)	**Protect (PR)**	PR.AC – Access Control	Validate resource ownership with token; enforce RBAC / ACLs
	Detect (DE)	DE.CM – Continuous Monitoring	Alert SIEM on token accessing multiple account_ids
Rate Limiting / Brute-force	**Protect (PR)**	PR.DS – Data Security	API Gateway rate limiting, IP / user throttling
	Detect (DE)	DE.CM – Continuous Monitoring	Detect repeated failed login attempts / excessive requests
Logging & Monitoring Gap	**Detect (DE)**	DE.CM – Continuous Monitoring	Centralized logging (API, auth, DB), correlation of anomalies
Incident Response Readiness	**Respond (RS)**	RS.MI – Mitigation	Incident playbooks, IOC ingestion, response actions
	Recover (RC)	RC.IM – Improvements	Lessons learned, policy/process updates post-incident
Data Protection / Privacy	**Protect (PR)**	PR.DS – Data Security	Encrypt sensitive data in transit & at rest, enforce retention policy

VIDEO Bağlantı Linki: https://vimeo.com/1135097518?fl=ip&fe=ec