Acme Security Incident Report

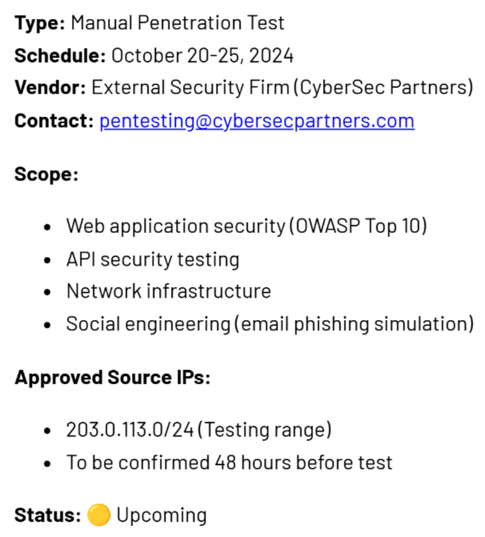
Section 1: Incident Analysis

Overview

From api_docs.pdf we identified a critical weakness around the GET /api/v1/portfolio/{account_id} endpoint used by the Portfolio Management module. This endpoint should only allow users to access their own accounts, but the documentation suggests authorization checks might be insufficient. That pointed us toward a possible **IDOR/BOLA** vulnerability.

There was also a Python request example in the docs showing user_id = null, 401 response codes, and repeated requests — which looked suspicious at first. After investigating, we concluded those were internal network scans and not malicious.

Another key document, security_test_schedule.pdf, gave us an important clue: the attacker's IP 203.0.113.45 came from an IP range approved for penetration testing. However, the authorized test window was **Oct 20–25**, while the incident happened on **Oct 15**. That confirms the activity was **not** an authorized test but a malicious intrusion.



Type: Manual Penetration Test
Schedule: October 20-25, 2024
Vendor: External Security Firm (CyberSec Partners)
Contact: pentesting@cybersecpartners.com

Scope:

- Web application security (OWASP Top 10)
- API security testing
- Network infrastructure
- Social engineering (email phishing simulation)

Approved Source IPs:

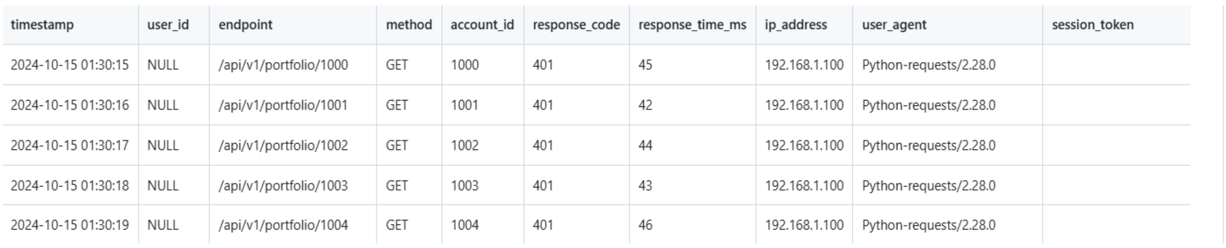
- 203.0.113.0/24 (Testing range)
- To be confirmed 48 hours before test

Status: 🟡 Upcoming

email_logs.csv shows email activity used for social engineering. The logs prove three employees clicked a fake link and gave the attacker either credentials or a valid session token. This happened at **09:00:23**.

Timeline
Initial traffic starts at **01:30**. api_logs.csv and waf_logs.csv show this traffic came from an internal IP (192.168.1.100) and returned 401 Unauthorized — this was a scheduled weekly security scan, not an attack. Because it was internal scanning, we ruled out vulnerability exploitation at that time.

The actual malicious activity unfolded in three stages:



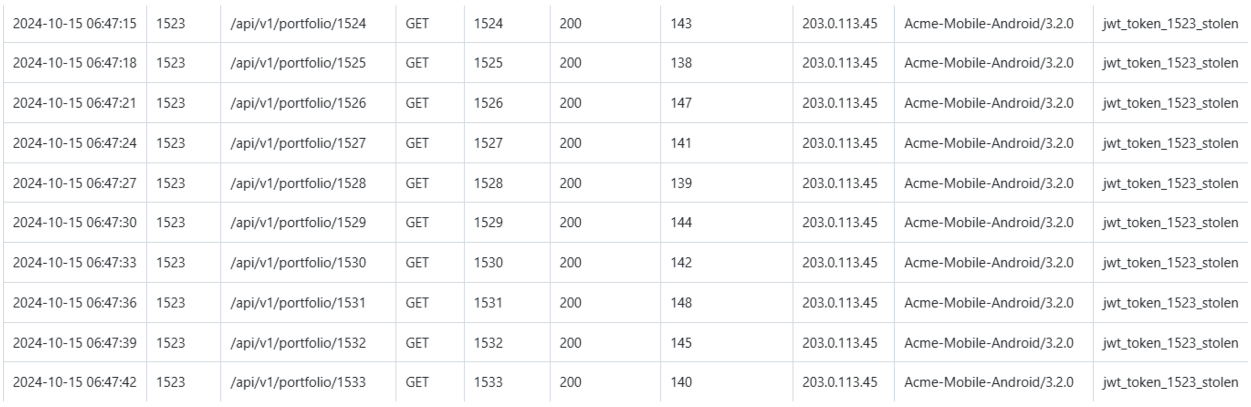
timestamp	user_id	endpoint	method	account_id	response_code	response_time_ms	ip_address	user_agent	session_token
2024-10-15 01:30:15	NULL	/api/v1/portfolio/1000	GET	1000	401	45	192.168.1.100	Python-requests/2.28.0	
2024-10-15 01:30:16	NULL	/api/v1/portfolio/1001	GET	1001	401	42	192.168.1.100	Python-requests/2.28.0	
2024-10-15 01:30:17	NULL	/api/v1/portfolio/1002	GET	1002	401	44	192.168.1.100	Python-requests/2.28.0	
2024-10-15 01:30:18	NULL	/api/v1/portfolio/1003	GET	1003	401	43	192.168.1.100	Python-requests/2.28.0	
2024-10-15 01:30:19	NULL	/api/v1/portfolio/1004	GET	1004	401	46	192.168.1.100	Python-requests/2.28.0	

Section 2: Architecture Review

Stage 1 — IDOR / BOLA (Broken Object Level Authorization) **(06:47:36)**

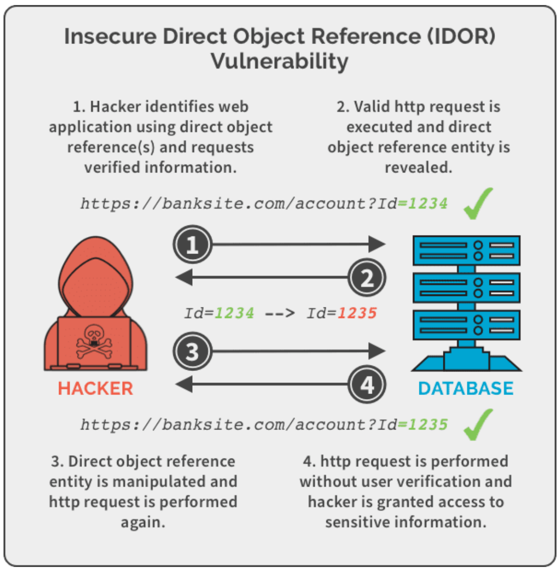
MITRE ATT&CK: T1078 — Valid Accounts

- **Details:** The attacker used a valid JWT token stolen from user_id: 1523 and iterated the path parameter to access other account numbers (e.g., 1532, 1533). api_logs.csv shows these unauthorized requests returned **200 OK**.



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

- **What IDOR is:** IDOR allows a user to access another user's private data because object-level authorization checks are missing. Here's a quick flow to illustrate:


Insecure Direct Object Reference (IDOR) Vulnerability
1. Hacker identifies web application using direct object reference(s) and requests verified information.
2. Valid http request is executed and direct object reference entity is revealed.
https://banksite.com/account?Id=1234
1
2
Id=1234 --> Id=1235
3
4
HACKER
DATABASE
https://banksite.com/account?Id=1235
3. Direct object reference entity is manipulated and http request is performed again.
4. http request is performed without user verification and hacker is granted access to sensitive information.

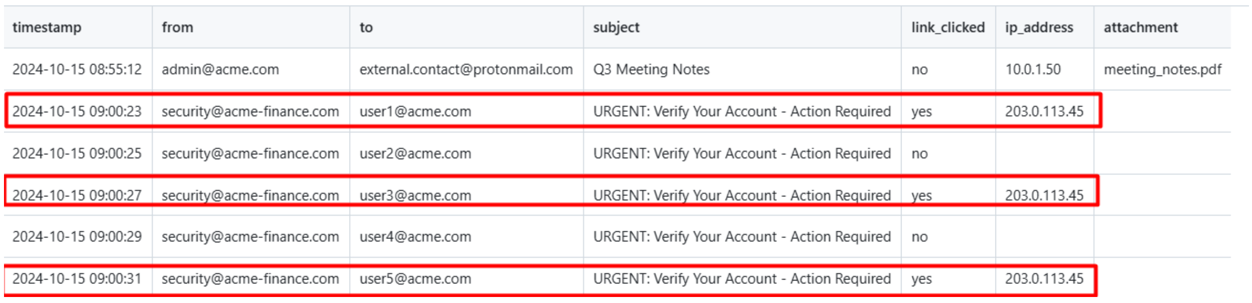
Steps:

1. Attacker discovers the app uses direct object references (IDs) and requests a valid resource (e.g., Id=1234).
2. The app responds successfully.
3. Attacker modifies the ID (e.g., Id=1235) and resends the request.
4. If the server does not verify ownership, the attacker gets access to someone else's data — which is what happened here.

Stage 2 — Phishing / Social Engineering (09:00:23)

MITRE ATT&CK: T1566 — Phishing

- **Details:** email_logs.csv shows the sender used a spoofed domain (security@acme-finance.com). Three employees clicked the malicious link and the attacker obtained credentials or a session, providing the initial foothold.

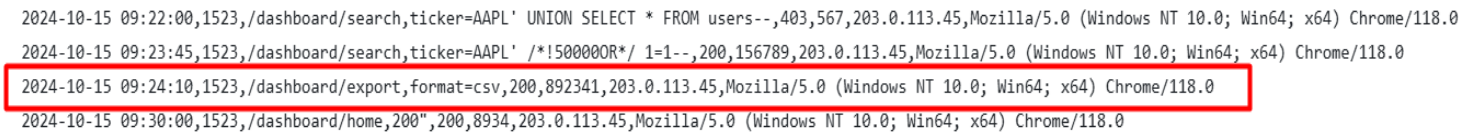


timestamp	from	to	subject	link_clicked	ip_address	attachment
2024-10-15 08:55:12	admin@acme.com	external.contact@protonmail.com	Q3 Meeting Notes	no	10.0.1.50	meeting_notes.pdf
2024-10-15 09:00:23	security@acme-finance.com	user1@acme.com	URGENT: Verify Your Account - Action Required	yes	203.0.113.45	
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	

Stage 3 — WAF Bypass & SQL Injection (09:23:45)

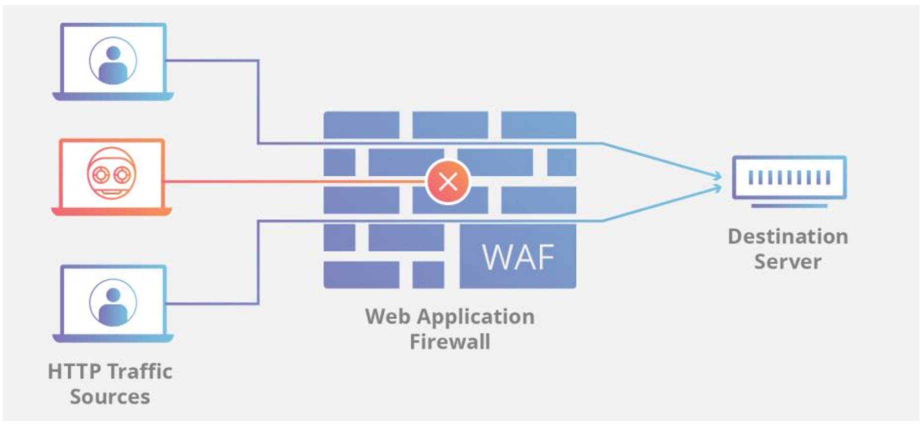
MITRE ATT&CK: T1190 — Web Application Compromise

- **Details:** web_logs.csv and waf_logs.csv show initial SQLi attempts were BLOCKed by the WAF. The attacker then used an obfuscated payload (/*!50000OR*/) to bypass WAF rule **981001**. The WAF logged (DETECT) but did not block this bypass. Right after, at **09:24:10**, about **892 KB** of sensitive data was exfiltrated via /dashboard/export.



```
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; Win64; x64) Chrome/118.0
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; Win64; x64) Chrome/118.0
2024-10-15 09:24:10,1523,/dashboard/export,format=csv,200,892341,203.0.113.45,Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/118.0
2024-10-15 09:30:00,1523,/dashboard/home,200",200,8934,203.0.113.45,Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/118.0
```

- **What SQLi is:** SQL injection is when an attacker injects malicious SQL into input fields to manipulate database queries. If application-side protections are weak, WAF evasions can still succeed.

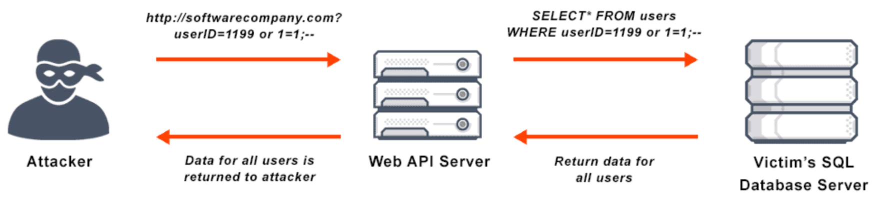


Section 3: Response & Remediation:

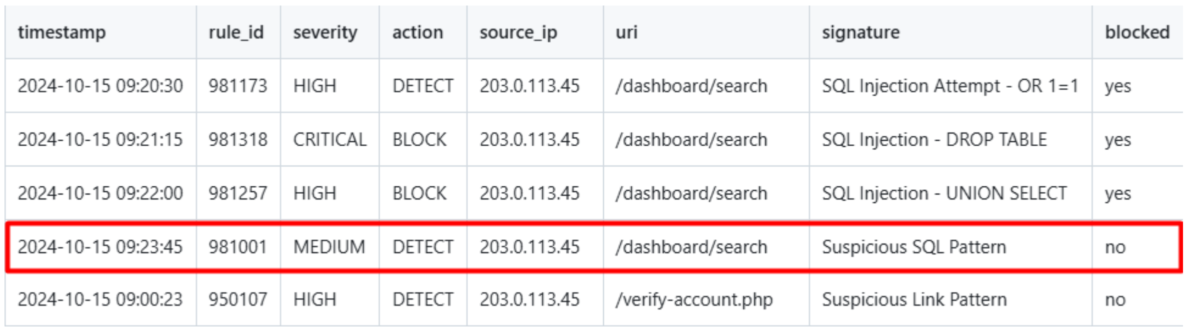
How to Fixes:

1. **IDOR / BOLA Vulnerability**
   - IDOR/BOLA happens when authentication exists but object-level authorization is missing. An attacker can escalate privileges or access others' data.
   - Developers must enforce strict authentication *and* authorization checks. **BOLA must be enforced** for APIs.
   - Quick comparison:
     - **IDOR** → web apps (e.g., /user?id=96 → access another user)
     - **BOLA** → APIs (e.g., /api/v1/portfolio/1523 → another user's portfolio)

2. **SQL Injection (SQLi) Vulnerability**
   - Use prepared statements / parameterized queries — never concatenate user input into SQL.
   - Keep WAF enabled, but don't rely on it as the only line of defense. Block traffic from repeated sqlmap-like patterns.
   - Do not expose raw DB errors to users.


http://softwarecompany.com?
userID=1199 or 1=1;--
SELECT* FROM users
WHERE userID=1199 or 1=1;--
Attacker
Data for all users is returned to attacker
Web API Server
Return data for all users
Victim's SQL Database Server

3. **Network & WAF**
   - Move WAF rules like **981001** from DETECT to **BLOCK** for bypass attempts and similar patterns.
   - Ensure test IP ranges are only allowed during approved windows; traffic from those IPs outside the window should be blocked and escalated.



timestamp	rule_id	severity	action	source_ip	uri	signature	blocked
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 09:00:23	950107	HIGH	DETECT	203.0.113.45	/verify-account.php	Suspicious Link Pattern	no

4. **People & Process**
   - Enforce **MFA(Multi-Factor Authentication)** for all staff.
   - SOC team must treat test-range traffic that occurs outside scheduled windows as hostile and respond immediately.
   - Run regular phishing awareness and response drills.

01:30
06:47
09:00
09:20
09:23
09:24

Author: Emirhan Köseoğlu

Date: 10/11/2025